Mathématique

&

Informatique

Copyright © 2015
Loïc Vrillon
✉ **vrillon.loic@gmail.com**

Mathématique & Informatique

Scilab - *Scientific Laboratory* - est un logiciel de calcul scientifique (équations, graphiques, optimisations, statistiques, graphes, ...) disposant d'une bibliothèque de plusieurs centaines de fonctions. Créé à l'INRIA, il est distribué sous licence CeCILL (Open Source) par Scilab Enterprises et téléchargeable sur le site internet : **http://www.scilab.org/**

Notations matricielles.

Une matrice se présente sous la forme de données encadrées par 2 crochets. Chaque élément d'une ligne est séparé du suivant par une virgule ou des espaces. Le point-virgule sert à passer à la ligne suivante.
$A = [a_{11} , ... , a_{1q} ; ... ; a_{p1} , ... , a_{pq}]$ désigne une matrice de dimension (p , q) , c'est-à-dire de p lignes et q colonnes.
Les éléments $A(i , j) = a_{ij}$ sont référencés en colonne : $A(i , j) = A(i + (j - 1)*p)$ avec i = 1, ... , p et j = 1, ... , q
La matrice transposée et conjuguée de A est notée A' : $A'(i,j) = \overline{A(j,i)}$. Noter A.' pour une transposition seule.
L'application d'une fonction f usuelle (sin, exp, sqrt, ...) aux éléments de A s'écrit f(A) : $f(A) = f([a_{ij}]) = [f(a_{ij})]$
Si $I = [i_1, ... , i_r]$ et $J = [j_1, ... , j_s]$ sont 2 vecteurs d'indices, alors $A(I , J) = [a_{ij}]_{i \in I, j \in J}$ est une matrice extraite de A.
NB. Par convention [λ] est assimilée au scalaire λ et [ ] symbolise la matrice vide.

Cas p = 1 ou q = 1 :

$x = [v_1 , ... , v_q]$ désigne un vecteur ligne (1 , q) composé de q valeurs v_j ⇒ $x(j) = v_j$
$y = [w_1 ; ... ; w_p]$ désigne un vecteur colonne (p , 1) composé de p valeurs w_i ⇒ $y(i) = w_i$
Soit n vecteurs lignes réels $x_1 , ... , x_n$ de même dimension :
$X = [x_1 ; ... ; x_n]$ désigne la matrice composée des vecteurs x_i en ligne ⇒ $X(i , :) = x_i$
$X' = [x_1 ; ... ; x_n]'$ désigne la matrice composée des vecteurs x_i en colonne ⇒ $X'(: , j) = x_j'$

Opérations de base :

Les symboles +, -, *, /, ^ sont utilisés pour les opérations habituelles sur les matrices et les symboles .*, ./, .^ sont utilisés pour les opérations élément par élément : $A \; .op \; B = [a_{ij} \; op \; b_{ij}]$
L'antislash \ sert à inverser une matrice : $A \setminus B = A^{-1} * B$
L'expression x = A \ b revient à résoudre A* x = b , et l'expression x = A / b revient à résoudre x * b = A
Le symbole .*. représente le produit tensoriel : $A .*. B = A \otimes B = [a_{ij} * B]$
NB. Validité des opérations matricielles :

$(p , q) \pm (p , q) = (p , q)$
$(p , q) * (q , r) = (p , r)$
$(p , q) / (r , q) = (p , r)$
$(p , q) \setminus (p , r) = (q , r)$

Fonctions usuelles.

- zeros(p , q) fournit la matrice p lignes, q colonnes dont les éléments sont des 0.
- ones(p , q) fournit la matrice p lignes, q colonnes dont les éléments sont des 1.
- eye(n , n) fournit la matrice identité d'ordre n.
- norm(x) fournit la norme euclidienne du vecteur x.
- det(A) fournit le déterminant de la matrice A.
- inv(A) fournit la matrice inverse de la matrice A.
- spec(A) fournit les valeurs propres de la matrice A.

Discrétisation.

La fonction *linspace* permet de créer un vecteur ligne constitué de points régulièrement espacés entre une valeur a et une valeur b. Sa syntaxe est : linspace(valeur a, valeur b, nombre de points). Le pas est égal à (b - a) / (nb_pts - 1).
Par exemple : x = linspace(0, 10, 6) est équivalent à x = [0, 2, 4, 6, 8, 10]
Pour obtenir des points équidistants à partir d'une valeur a jusqu'à une valeur limite b, on peut aussi écrire :
valeur a : valeur du pas : valeur b. Par défaut, la valeur du pas est de +1 (a:b ⇔ a:1:b).
Par exemple : x = 0:2:11 est équivalent à x = [0, 2, 4, 6, 8, 10]
NB. Avec linspace les bornes a et b sont toujours atteintes.

Mathématique & Informatique

EXEMPLES.

```
A = [1,2;3,4] = 1.   2.  ;  B = [5,6;7,8] = 5.   6.
                3.   4.                     7.   8.

x = [9,10] = 9.   10.  ;  y = [11,12] = 11.   12.  ;  z = [13;14] = 13.
                                                          14.
```

Addition :

```
A + B = B + A = 6.    8.  ;  A + 1 = 2.   3.  ;  x + y = 20.  22.
                10.   12.            4.   5.
```

Soustraction :

```
A - B = -4.   -4.  ;  B - A = 4.   4.  ;  x - y = -2.  -2.
        -4.   -4.             4.   4.
```

Multiplication :

```
5 * A = 5 .* A = A * 5 = A .* 5 = 5.    10.
                                  15.   20.

A * B = 19.   22.  ;  B * A = 23.   34.  ;  A .* B = B .* A = 5.    12.
        43.   50.             31.   46.                      21.   32.

A * z = 41.  ;  x * y' = 219.  ;  x' * y = 99.    108.
        95.                               110.   120.
```

Division :

```
A / 2 = A ./ 2 = 0.5   1.  ;  3 / A = -6.     3.   ;  3 ./ A = 3.   1.5

                 1.5   2.             4.5   -1.5             1.   0.75

A / B = 3.   -2.  ;  A ./ B = 0.2         0.3333333
        2.   -1.             0.4285714   0.5

x / A = -3.   4.  ;  x ./ y = 0.8181818  0.8333333
```

Exponentiation :

```
A ^ 2 = 7.    10.  ;  A .^ 2 = 1.   4.   ;  x ^ 3 = x .^ 3 = 729.   1000.
        15.   22.             9.   16.
```

Inversion :

```
A \ B = -3.   -4.  ;  A \ 1 = -2.     1.   ;  A \ z = -12.
         4.    5.             1.5   -0.5             12.5
```

Fonction :

```
exp(A) = 2.7182818   7.3890561  ;  spec(A) =  5.3722813
         20.085537   54.59815                -0.3722813
```

Produit tensoriel :

```
A .*. B = 5.    6.    10.   12.  ;  x .*. y = 99.   108.   110.   120.
          7.    8.    14.   16.
          15.   18.   20.   24.
          21.   24.   28.   32.
```

Mathématique & Informatique

Graphisme en 2 dimensions :

On utilise la fonction ***plot2d*** pour représenter une courbe dans le plan :

plot2d(x, y, options)

x est le vecteur ligne des abscisses : (1 , n) $\Rightarrow$ x(i)
y est le vecteur ligne des ordonnées : (1 , n) $\Rightarrow$ y(j)

- **Cas 1 : la courbe est donnée par une équation cartésienne y(x)**

  1) Définir le domaine de variation de l'abscisse x :
     x = linspace(x_{min} , x_{max} , n_x)

  2) Ecrire l'expression analytique de l'ordonnée y :
     y = f (x)

  3) Dessiner la courbe : plot2d(x, y, options)

- **Cas 2 : la courbe est donnée par des équations paramétriques x(t), y(t)**

  1) Définir le domaine de variation du paramètre t :
     t = linspace(t_{min} , t_{max} , n_t)

  2) Ecrire les expressions analytiques de l'abscisse x et de l'ordonnée y des points :
     x = f_1 (t)
     y = f_2 (t)

  3) Dessiner la courbe : plot2d(x, y, options)

- **Cas 3 : la courbe est donnée une équation polaire r(θ)**

  1) Définir le domaine de variation de l'angle theta :
     theta = linspace($theta_{min}$, $theta_{max}$, n_{theta})

  2) Ecrire les expressions analytiques du rayon r, de l'abscisse x et de l'ordonnée y :
     r = f(theta)
     x = r .* cos(theta)
     y = r .* sin(theta)

  3) Dessiner la courbe : plot2d(x, y, options)

Les options *style* = n_1 , *frameflag* = n_2 , *axesflag* = n_3 , permettent de choisir respectivement la couleur de la courbe, l'échelle du tracé, le type d'axes, en fonction des valeurs des entiers n_1 , n_2 , n_3 .
Pour plus de détails, accéder à l'aide de Scilab : *F1* $\longrightarrow$ *Graphiques* $\longrightarrow$ *plot2d*

Remarques :

- Lorsqu'on dessine une courbe avec *plot2d*, on peut écrire indifféremment ses coordonnées en ligne ou en colonne.
- La fonction *plot2d* relie 2 points successifs d'une même courbe par un trait.
  Par exemple : plot2d([a_1 , a_2], [b_1 , b_2]) trace un trait entre le point $P_1(a_1, b_1)$ et le point $P_2(a_2, b_2)$.
  Il est donc inutile de discrétiser des segments de droite, il suffit de préciser leurs extrémités.
- Pour définir uniquement les options d'un dessin, on peut utiliser l'expression : plot2d(%inf, %inf, options)
- Il est possible de dessiner simultanément N courbes sur le même graphique en utilisant la syntaxe suivante :
  x_i = linspace(min x_i , max x_i , n) ; $y_i = f_i(x_i)$; plot2d([x_1 ; ... ; x_N]' , [y_1 ; ... ; y_N]' , options)
  Les coordonnées x_i et y_i sont fournies en colonne, d'où la présence du symbole ' (apostrophe droite).
  Si x_1 = ... = x_N = x, on utilise la formulation simplifiée : plot2d(x' , [y_1 ; ... ; y_N]' , options)
  A noter que le nombre n de points de calcul doit être le même pour toutes les courbes, sinon il faut faire appel à la fonction *plot2d* pour chaque courbe séparément :
  x_i = linspace(min x_i , max x_i , n_i) ; $y_i = f_i(x_i)$; plot2d(x_i , y_i , options)

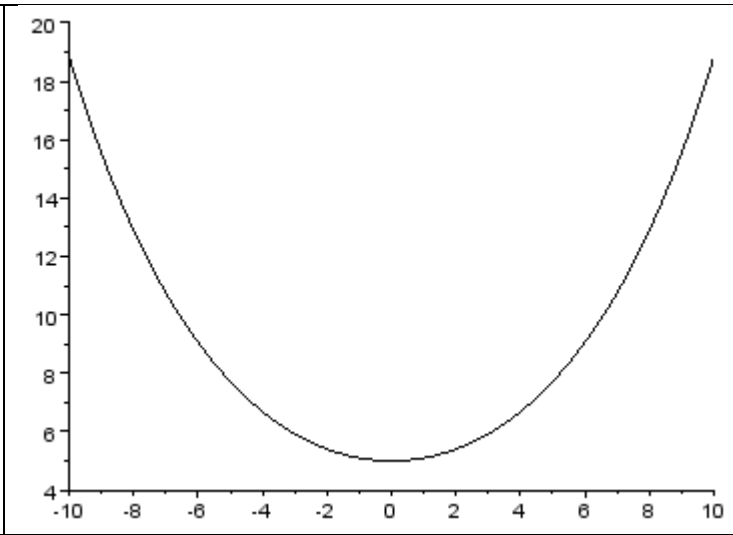
Mathématique & Informatique

Exemples de programmes Scilab :

$$y = a\ \mathrm{ch}\left(\frac{x}{a}\right)$$

// Chaînette

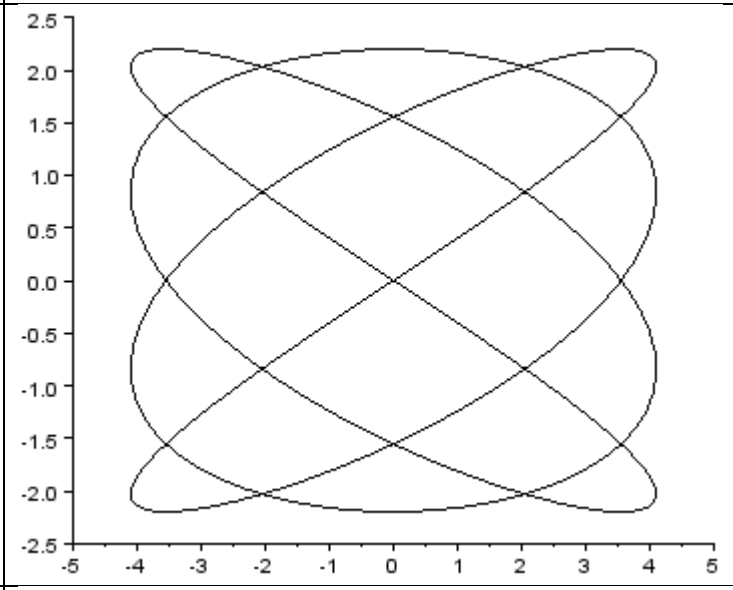
```
a = 5 ;
x = linspace(-10, +10, 200) ;
y = a*cosh(x/a) ;
plot2d(x, y)
```



$$\begin{cases} x = a_1\ \sin(2\pi f_1 t) \\ y = a_2\ \sin(2\pi f_2 t) \end{cases}$$

// Figure de Lissajous

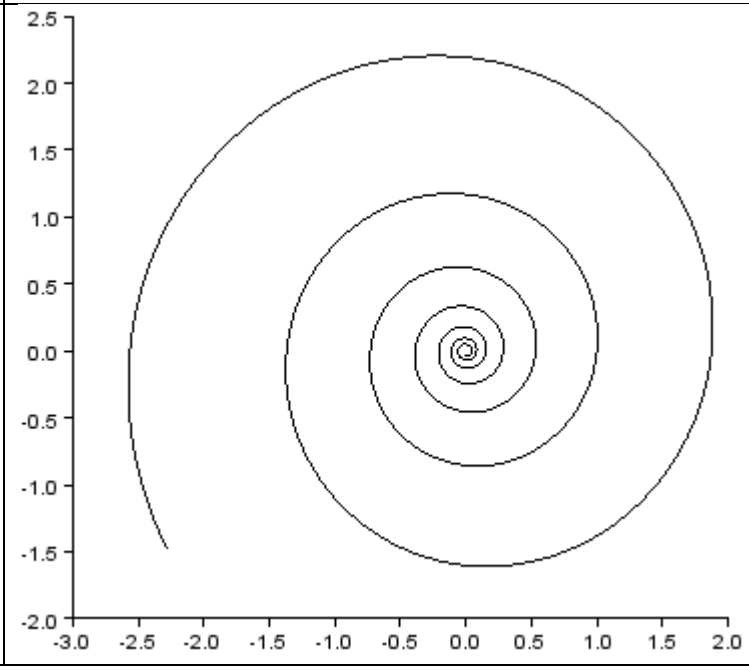
```
a1 = 4.1 ; f1 = 4 ; a2 = 2.2 ; f2 = 3 ;
t = linspace(0, 1, 200) ;
x = a1*sin(2*%pi*f1*t) ;
y = a2*sin(2*%pi*f2*t) ;
plot2d(x, y)
```



$$r = a\ e^{\,m\theta}$$

// Spirale logarithmique

```
a = 1 ; m = 0.1 ;
theta = linspace(-33,+10,1000) ;
r = a*exp(m*theta) ;
x = r.*cos(theta) ;
y = r.*sin(theta) ;
plot2d(x, y)
```



Mathématique & Informatique

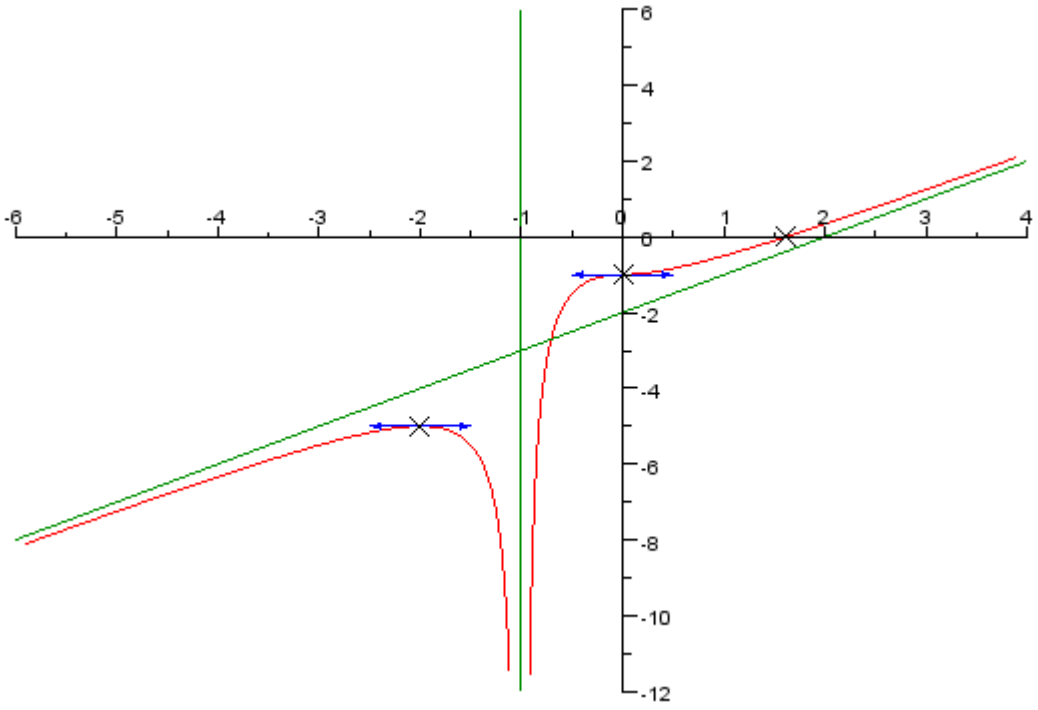
GRAPHIQUE D'UNE FONCTION y = f(x)

- Pour définir un repère orthogonal d'axes (Ox, Oy) centré en 0 (origine) :
  a = gca() ; a.x_location = "origin" ; a.y_location = "origin" ;
- Pour dessiner la courbe d'une fonction f définie sur un segment [a,b] :
  x = linspace(a , b , 300) ; y = f(x) ; plot2d(x , y , cc)
- Pour tracer une droite asymptote entre les points (x_1 , y_1) et (x_2 , y_2) :
  plot2d([x_1 , x_2] , [y_1 , y_2] , ca)
- Pour figurer la double flèche de la tangente au point (x_0 , y_0) :
  xarrows([x_0 , x_0 - dx , x_0 + dx] , [y_0 , y_0 ; y_0 - dy , y_0 + dy] , 4 , ct)
- Pour marquer d'une croix des points particuliers (x_1 , y_1) , ... , (x_n , y_n) :
  plot2d([x_1 , ... , x_n] , [y_1 , ... , y_n] , -2)

NB. Les constantes cc, ca, ct sont des nombres entiers qui représentent les couleurs des pixels.

Exemple : $y = x \left| 1 - \frac{1}{x + 1} \right| - 1$

// Axes
a = gca() ; a.x_location = "origin" ; a.y_location = "origin" ;
// Courbes
x = linspace(-5.90,-1.12,300) ; y = x.*abs(1 - (x+1)^(-1)) - 1 ; plot2d(x, y, 5)
x = linspace(-0.92,+3.90,300) ; y = x.*abs(1 - (x+1)^(-1)) - 1 ; plot2d(x, y, 5)
// Asymptotes
plot2d([-1,-1], [-12,+6], 13) ; plot2d([-6,+4], [-8,+2], 13)
// Tangentes
xarrows([0,0;-0.5,+0.5], [-1,-1;-1,-1], 4, 2) ; xarrows([-2,-2;-2.5,-1.5], [-5,-5;-5,-5], 4, 2)
// Points
plot2d([-2,0,(1+sqrt(5))/2], [-5,-1,0], -2)



Mathématique & Informatique

COURBES DE DÉSINTÉGRATION RADIOACTIVE

La radioactivité naturelle a été découverte par **Henri Becquerel** en 1896 (uranium), confirmée par **Pierre** et **Marie Curie** en 1898 (polonium, radium). La découverte de la radioactivité artificielle est due à **Frédéric** et **Irène Joliot-Curie** en 1934. Rappelons les expressions des 2 lois usuelles de la désintégration radioactive :

1) Simple désintégration du type : $A^* \to B$

Pendant le temps dt, il disparaît un nombre dA d'atomes proportionnel au nombre A d'atomes présents :

$$dA = -\lambda_A A\,dt \;\Rightarrow\; \frac{dA}{dt} + \lambda_A A = 0$$

Solution de l'équation différentielle linéaire du premier ordre :

$$\boxed{A = A_0 e^{-\lambda_A t}}$$

$\begin{cases} A = \text{nombre d'atomes de l'élément radioactif } A \text{ présents à l'instant } t \\ A_0 = \text{nombre d'atomes de l'élément radioactif } A \text{ présents à l'instant } t = 0 \\ \lambda_A = \text{constante radioactive de l'élément } A \end{cases}$

La courbe de A est une exponentielle décroissante, le maximum A_{max} est atteint pour $t = 0$.
Par définition, la période (ou demi-vie) T_A de l'élément radioactif A est le temps nécessaire pour que la moitié du nombre initial A_0 d'atomes se soient désintégrés : $A = \frac{A_0}{2} \;\Rightarrow\; T_A = \frac{\ln 2}{\lambda_A}$

Si l'élément B est stable, le nombre d'atomes de B à l'instant t est : $B = \int_0^t \lambda_A A\,dt = A_0\left(1 - e^{-\lambda_A t}\right)$

2) Double désintégration du type : $A^* \to B^* \to C$

Pendant le temps dt, il disparaît $dB_1 = -\lambda_B B\,dt$ atomes de B par désintégration de B et il apparaît $dB_2 = \lambda_A A\,dt$ atomes de B par désintégration de A. Notons dB la variation du nombre d'atomes de B dans l'intervalle dt, on a :

$$dB = dB_1 + dB_2 = -\lambda_B B\,dt + \lambda_A A\,dt \;\Rightarrow\; \frac{dB}{dt} + \lambda_B B = \lambda_A A_0 e^{-\lambda_A t}$$

Solution de l'équation différentielle linéaire du premier ordre :

$$\boxed{B = A_0 \frac{\lambda_A}{\lambda_B - \lambda_A}\left(e^{-\lambda_A t} - e^{-\lambda_B t}\right)}$$

$\begin{cases} B = \text{nombre d'atomes de l'élément radioactif } B \text{ présents à l'instant } t \\ A_0 = \text{nombre d'atomes de l'élément radioactif } A \text{ présents à l'instant } t = 0 \\ \lambda_A = \text{constante radioactive de l'élément } A \\ \lambda_B = \text{constante radioactive de l'élément } B \end{cases}$

La courbe de B est un dôme étalé, le maximum B_{max} est atteint pour $t_{max} = \frac{\ln \lambda_B - \ln \lambda_A}{\lambda_B - \lambda_A}$.

Lorsque $\lambda_A << \lambda_B$, aux alentours du temps t_{max} , il s'établit un équilibre radioactif entre A et B : $\lambda_A A = \lambda_B B$

Si l'élément C est stable, le nombre d'atomes de C à l'instant t est : $C = \int_0^t \lambda_B B\,dt = A_0\left(1 - \frac{\lambda_B e^{-\lambda_A t} - \lambda_A e^{-\lambda_B t}}{\lambda_B - \lambda_A}\right)$

Note

Solution générale d'une équation différentielle linéaire du premier ordre :

$$y' + a(x)y = b(x) \;\Rightarrow\; y = \left(\int b(x) e^{\int a(x)dx} dx + C\right) e^{-\int a(x)dx}$$

La constante C dépend des conditions initiales.

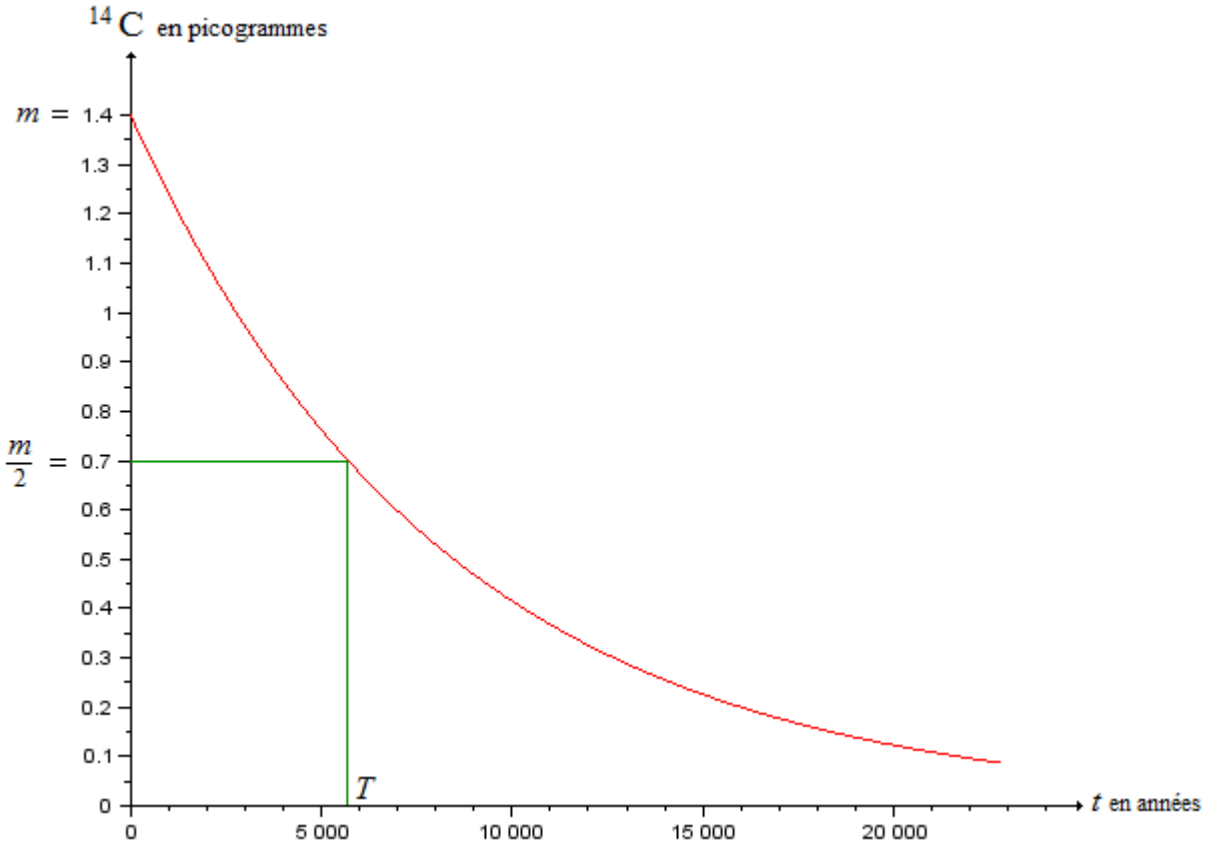
Mathématique & Informatique

Application 1

Tracer la courbe de désintégration du carbone 14 $\left({}^{14}_{6}C\right)$ dont la période radioactive T est approximativement de 5700 ans.

Le programme de la courbe du carbone radioactif se déduit de la formule : $m_C = m e^{-\lambda_C t}$

```
scf(1) ;
TC = 5700 ;
lambdaC = log(2)/TC ;
m = 1.4 ;
t = linspace(0,4*TC,300) ;
mC = m*exp(-lambdaC*t) ;
plot2d(t,mC,style = 5)
plot2d([TC,0;TC,TC],[0,m/2;m/2,m/2],style = [13,13])
```



Pour évaluer la masse m_A d'une substance radioactive A, on mesure son activité c'est-à-dire sa vitesse de désintégration $-dA/dt$ exprimée en becquerels (1 Bq = 1 désintégration par seconde). Si A désigne le nombre d'atomes radioactifs présents, m_A leur masse, M_A la masse molaire du radioélément, T_A sa période et N le nombre d'**Avogadro**, on a la relation :

$$\left(-\frac{dA}{dt}\right) = \lambda_A A = \frac{\ln 2}{T_A}\frac{m_A N}{M_A} \Rightarrow m_A = \frac{M_A T_A}{N \ln 2}\left(-\frac{dA}{dt}\right)$$

Pour le carbone 14, **Libby** a obtenu une activité de 13.56 dpmg (désintégrations par minute et par gramme de carbone) :

$$m = \frac{14 \times (5700 \times 365.25 \times 24 \times 3600)}{6.022\ 10^{23} \times 0.693} \times \frac{13.56}{60} = 1.36\ 10^{-12}\ g$$

Ainsi, un organisme vivant (animal, végétal) contient environ m = 1.4 picogrammes de carbone 14 par gramme de carbone. Après sa mort, cette quantité diminue lentement, conformément à la courbe ci-dessus.

Mathématique & Informatique

Application 2

Considérons la double désintégration radioactive suivante : $^{226}_{88}Ra \xrightarrow{\alpha} {}^{222}_{86}Rn \xrightarrow{\alpha} {}^{218}_{84}Po$

La particule α émise est un noyau d'hélium $^{4}_{2}He^{2+}$ (hélion).

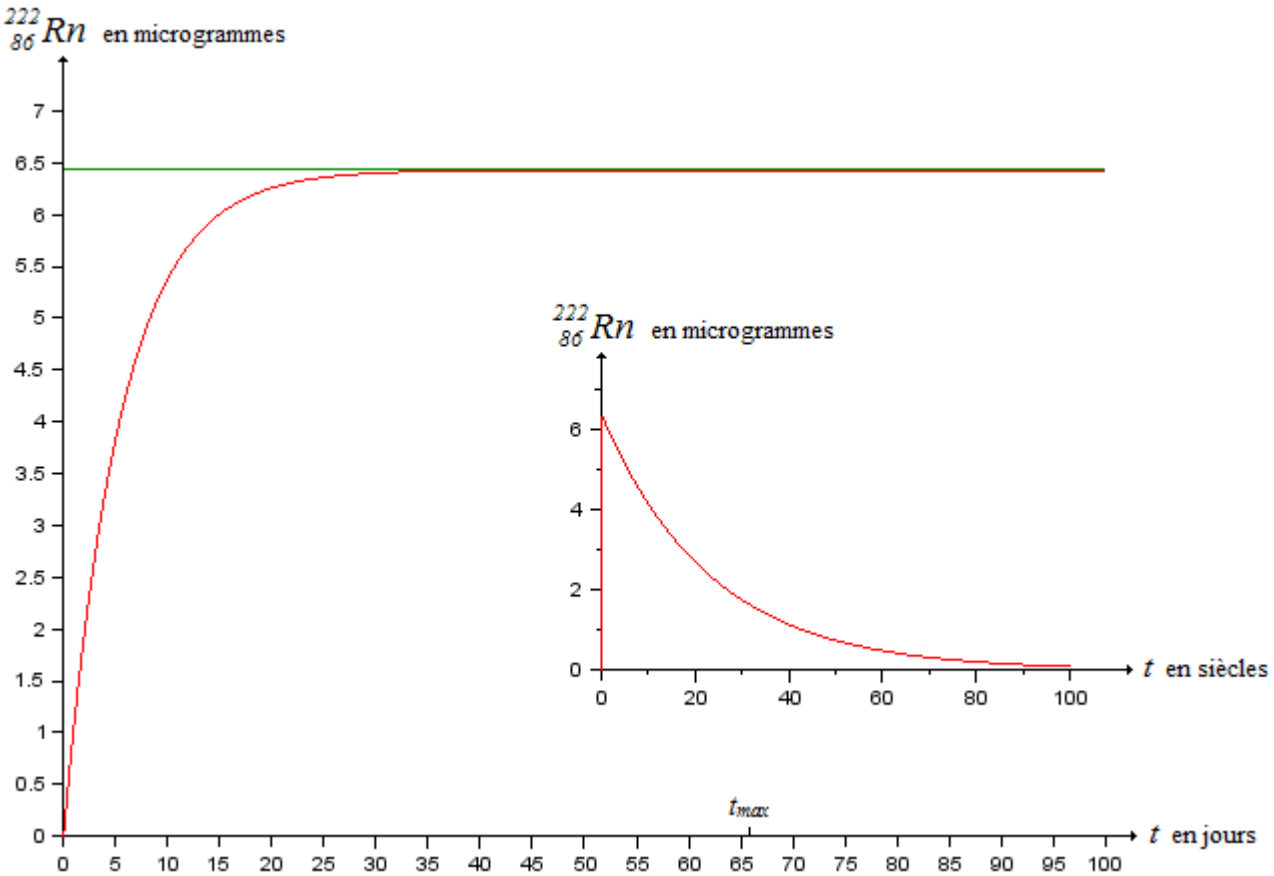
La période du radium $^{226}_{88}Ra$ est T_{Ra} = 1600 années.

La période du radon $^{222}_{86}Rn$ est T_{Rn} = 3.824 jours.

Partant d'un gramme de radium, tracer la courbe donnant la quantité de radon présente au cours du temps.

Le programme de la courbe du radon radioactif se déduit de la formule : $m_{Rn} = m_{Ra}\dfrac{M_{Rn}}{M_{Ra}}\dfrac{\lambda_{Ra}}{\lambda_{Rn}-\lambda_{Ra}}\left(e^{-\lambda_{Ra}t} - e^{-\lambda_{Rn}t}\right)$

```
scf(2) ;
lambdaRa = log(2)/(1600*365.25) ;
lambdaRn = log(2)/3.824 ;
K = (1D6)*(222/226) ;
j = 100 ;
t = linspace(0,j,300) ;
mRn = K*(lambdaRa/(lambdaRn - lambdaRa))*(exp(-lambdaRa*t) - exp(-lambdaRn*t)) ;
plot2d((log(lambdaRn) - log(lambdaRa))/(lambdaRn - lambdaRa),0,style = -1)
plot2d([0,j],[1,1]*K*lambdaRa/lambdaRn,style = 13)
plot2d(t,mRn,style = 5)
```



Après 30 jours, on atteint un équilibre radioactif où il se forme approximativement autant de radon qu'il en disparaît.
Cette quasi-stabilité de la quantité de radon se prolonge environ une centaine d'années, d'où le nom d'équilibre séculaire.
Sur le schéma, la transition est représentée par une droite horizontale en vert, elle correspond à la masse suivante de $^{222}_{86}Rn$:

$$m_{max} \approx m_{Ra}\frac{M_{Rn}}{M_{Ra}}\frac{T_{Rn}}{T_{Ra}} = 1\times\frac{222}{226}\times\frac{3.824}{1600\times365.25} = 6.42\ 10^{-6}\,g$$

Mathématique & Informatique

Chaîne de désintégration radioactive

D'une manière générale, soit la chaîne de désintégration radioactive : $X_1 \to X_2 \to \cdots X_i \to X_{i+1} \cdots \to X_{n-1} \to X_n$

Chaque transmutation d'un élément père X_i en élément fils X_{i+1} émet une particule α ou β (suivant le cas) et s'accompagne le plus souvent d'un rayonnement électromagnétique γ.
Les éléments X_1 , X_2 , ... , X_{n-1} sont radioactifs ; l'élément terminal X_n est stable, il ne se désintègre pas (non radioactif).
Par convention on désignera par $X_i(t)$, ou plus simplement par X_i, le nombre d'atomes de l'élément X_i présents à l'instant t et par λ_i $(1 \le i \le n-1)$ la constante radioactive de l'élément X_i $(1 \le i \le n-1)$.
Ecrivons les équations différentielles régissant les désintégrations successives :

$$\begin{cases} \dfrac{dX_1}{dt} = -\lambda_1 X_1 \\ \dfrac{dX_2}{dt} = \lambda_1 X_1 - \lambda_2 X_2 \\ \vdots \\ \dfrac{dX_i}{dt} = \lambda_{i-1} X_{i-1} - \lambda_i X_i \\ \vdots \\ \dfrac{dX_{n-1}}{dt} = \lambda_{n-2} X_{n-2} - \lambda_{n-1} X_{n-1} \\ \dfrac{dX_n}{dt} = \lambda_{n-1} X_{n-1} \end{cases}$$

Ce système différentiel se résoud de proche en proche en supposant qu'à l'instant initial on a :

$$X_1 = X_0 \quad \text{et} \quad X_2 = \cdots = X_i = \cdots = X_n = 0$$

La solution de l'équation linéaire n° i permet d'intégrer l'équation linéaire n° $i+1$, on obtient :

$$X_1 = X_0 e^{-\lambda_1 t}$$

$$X_i = X_0 \lambda_1 \lambda_2 \cdots \lambda_{i-1} \sum_{k=1}^{i} \frac{e^{-\lambda_k t}}{(\lambda_1 - \lambda_k) \cdots (\lambda_{k-1} - \lambda_k)(\lambda_{k+1} - \lambda_k) \cdots (\lambda_i - \lambda_k)} \qquad \text{pour } i = 2, \cdots, n-1$$

$$X_n = X_0 \left(1 - \sum_{k=1}^{n-1} \frac{\lambda_1 \cdots \lambda_{k-1} \lambda_{k+1} \cdots \lambda_{n-1}}{(\lambda_1 - \lambda_k) \cdots (\lambda_{k-1} - \lambda_k)(\lambda_{k+1} - \lambda_k) \cdots (\lambda_{n-1} - \lambda_k)} e^{-\lambda_k t} \right)$$

Un nombre X d'atomes d'un même élément correspond à une masse $m = \dfrac{M}{N} X$ où M est la masse molaire de l'élément considéré (masse d'une mole de ses atomes) et N est le nombre d'**Avogadro** ($N = 6.022\ 10^{23}$).

Par suite, m_0 représentant la masse initiale de l'élément X_1, m_i la masse de l'élément X_i à l'instant t et M_i la masse molaire de l'élément X_i, on a :

$$m_1 = m_0 e^{-\lambda_1 t}$$

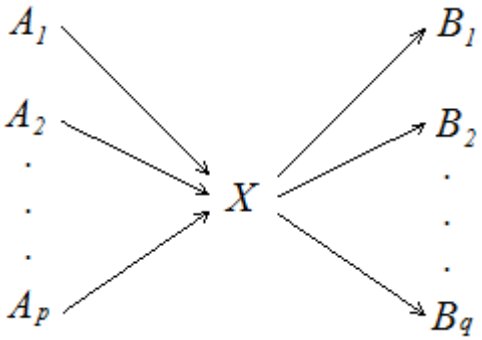
$$m_i = m_0 \frac{M_i}{M_1} \lambda_1 \lambda_2 \cdots \lambda_{i-1} \sum_{k=1}^{i} \frac{e^{-\lambda_k t}}{(\lambda_1 - \lambda_k) \cdots (\lambda_{k-1} - \lambda_k)(\lambda_{k+1} - \lambda_k) \cdots (\lambda_i - \lambda_k)} \qquad \text{pour } i = 2, \cdots, n-1$$

$$m_n = m_0 \frac{M_n}{M_1} \left(1 - \sum_{k=1}^{n-1} \frac{\lambda_1 \cdots \lambda_{k-1} \lambda_{k+1} \cdots \lambda_{n-1}}{(\lambda_1 - \lambda_k) \cdots (\lambda_{k-1} - \lambda_k)(\lambda_{k+1} - \lambda_k) \cdots (\lambda_{n-1} - \lambda_k)} e^{-\lambda_k t} \right)$$

Mathématique & Informatique

Complément

Dans une chaîne élémentaire de désintégration radioactive, chaque radioélément X_i possède au plus un prédécesseur X_{i-1} et au plus un successeur X_{i+1}. Dans le cas d'un processus plus complexe de désintégration radioactive, chaque radioélément X est issu de p radioéléments différents A_1 , A_2 , … , A_p et peut générer q radioéléments différents B_1 , B_2 , … , B_q.
Schématiquement :



Notons a_i $(1 \le i \le p)$ le coefficient de proportionnalité, supposé connu, de la désintégration de A_i en X et b_j $(1 \le j \le q)$ le coefficient de proportionnalité, supposé connu, de la désintégration de X en B_j.
Le nombre dA_i d'atomes de A_i $(1 \le i \le p)$ qui disparaissent lors de la désintégration du radioélément A_i est proportionnel à la fois au nombre A_i d'atomes présents et à l'intervalle de temps dt : $dA_i = -a_i A_i dt$.
Le nombre dB_j d'atomes de B_j $(1 \le j \le q)$ qui apparaissent lors de la désintégration du radioélément X est proportionnel à la fois au nombre X d'atomes présents et à l'intervalle de temps dt : $dB_j = b_j X dt$.
On en déduit la variation dX du nombre d'atomes de X pendant l'intervalle de temps dt :

$$dX = -\sum_{i=1}^{p} dA_i - \sum_{j=1}^{q} dB_j = \sum_{i=1}^{p} a_i A_i dt - \sum_{j=1}^{q} b_j X dt$$

D'où l'équation différentielle linéaire (E) relative à l'élément X :

$$\frac{dX}{dt} = a_1 A_1 + a_2 A_2 + \cdots + a_p A_p - (b_1 + b_2 + \cdots + b_q) X$$

Supposons que le graphe de désintégration radioactive possède au total n sommets X_1 , X_2 , … , X_n chacun étant associé à une équation différentielle linéaire (E_1) , (E_2) , … , (E_n). On aboutit à un système différentiel linéaire homogène à coefficients constants de n équations à n inconnues :

$$\frac{d}{dt}\begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} = \begin{pmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & & \vdots \\ M_{n1} & \cdots & M_{nn} \end{pmatrix} \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} \quad \Leftrightarrow \quad \frac{dY}{dt} = M\,Y$$

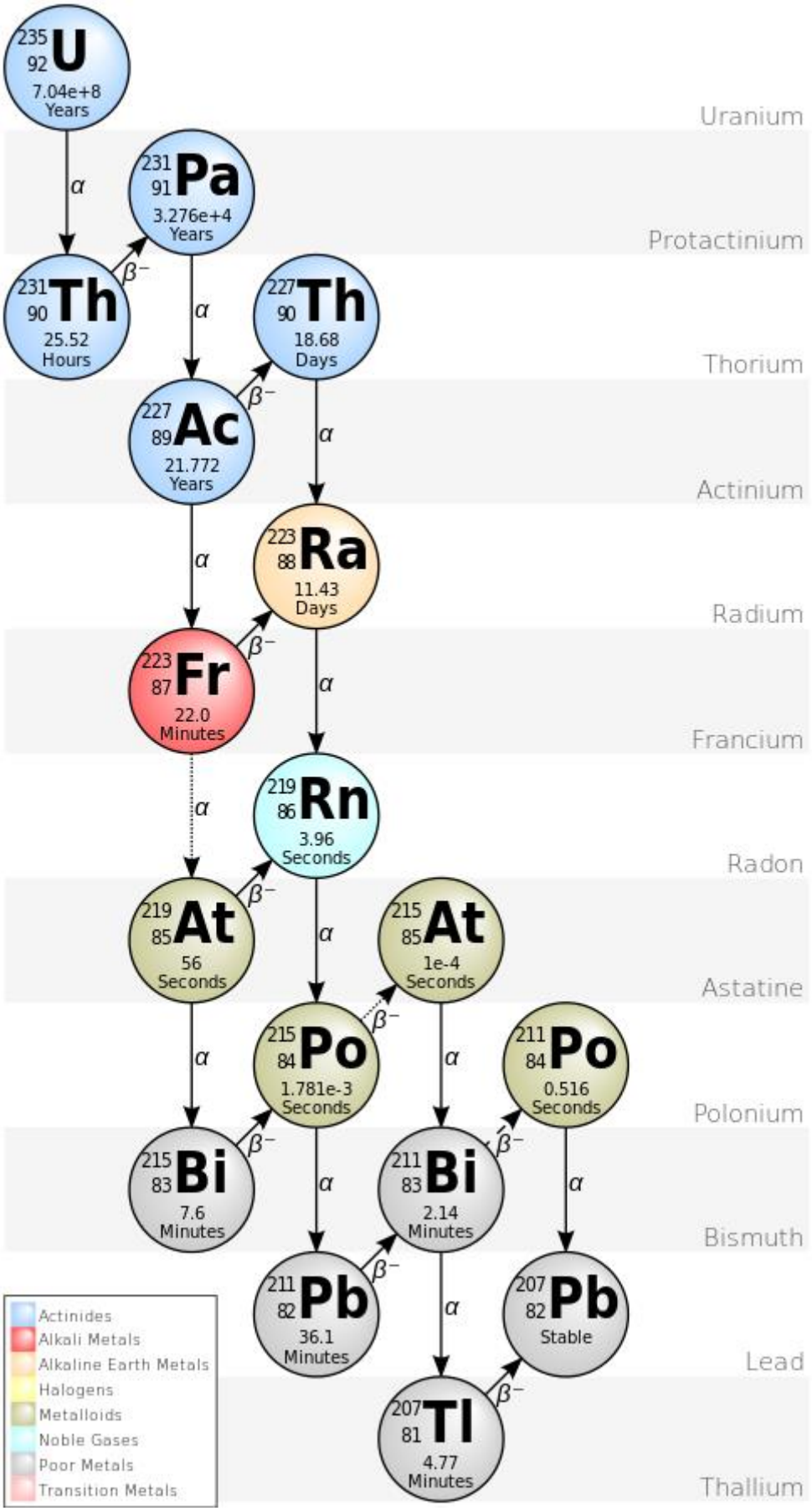
La notation M désigne une matrice carrée réelle d'ordre n, la notation Y désigne un vecteur colonne de n fonctions de t.
On sait résoudre un tel système, de manière analytique (en mettant M sous la forme $M = D + N$ où D est diagonalisable et N nilpotente avec $DN = ND$, puis en exprimant la solution $Y = e^{tM} Y_0$) et de manière numérique (voir la fonction Scilab *ode*).
En particulier, une chaîne élémentaire de désintégration radioactive correspond à une matrice M bidiagonale et singulière :

$$X_1 \to X_2 \to X_3 \to \cdots \to X_{n-1} \to X_n \quad \Rightarrow \quad \frac{d}{dt}\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_{n-1} \\ X_n \end{pmatrix} = \begin{pmatrix} -\lambda_1 & 0 & 0 & \cdots & 0 & 0 \\ \lambda_1 & -\lambda_2 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2 & -\lambda_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\lambda_{n-1} & 0 \\ 0 & 0 & 0 & \cdots & \lambda_{n-1} & 0 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_{n-1} \\ X_n \end{pmatrix}$$

où $\lambda_i = \dfrac{\ln 2}{T_i}$ $(1 \le i \le n - 1)$ est la constante radioactive de l'élément X_i de période T_i.

Mathématique & Informatique

Exemple : schéma de désintégration radioactive de l'uranium 235 (**Edgar Bonet**)



Mathématique & Informatique

Application 1

Considérons la chaîne de désintégration de l'isotope ^{239}U de l'uranium : $^{239}U \xrightarrow{\beta} {}^{239}Np \xrightarrow{\beta} {}^{239}Pu \xrightarrow{\alpha} {}^{235}U$

$$^{239}_{92}U \rightarrow {}^{239}_{93}Np + {}^{0}_{-1}e$$
$$^{239}_{93}Np \rightarrow {}^{239}_{94}Pu + {}^{0}_{-1}e$$
$$^{239}_{94}Pu \rightarrow {}^{235}_{92}U + {}^{4}_{2}He$$

Tracer la courbe donnant la masse de plutonium ^{239}Pu obtenue à partir d'un gramme d'uranium ^{239}U durant 30 jours.
On donne les périodes radioactives de l'uranium 239, du neptunium 239 et du plutonium 239 :

$T_{U\,239}$ = 23.45 minutes ; $T_{Np\,239}$ = 2.3565 jours ; $T_{Pu\,239}$ = 24 110 ans

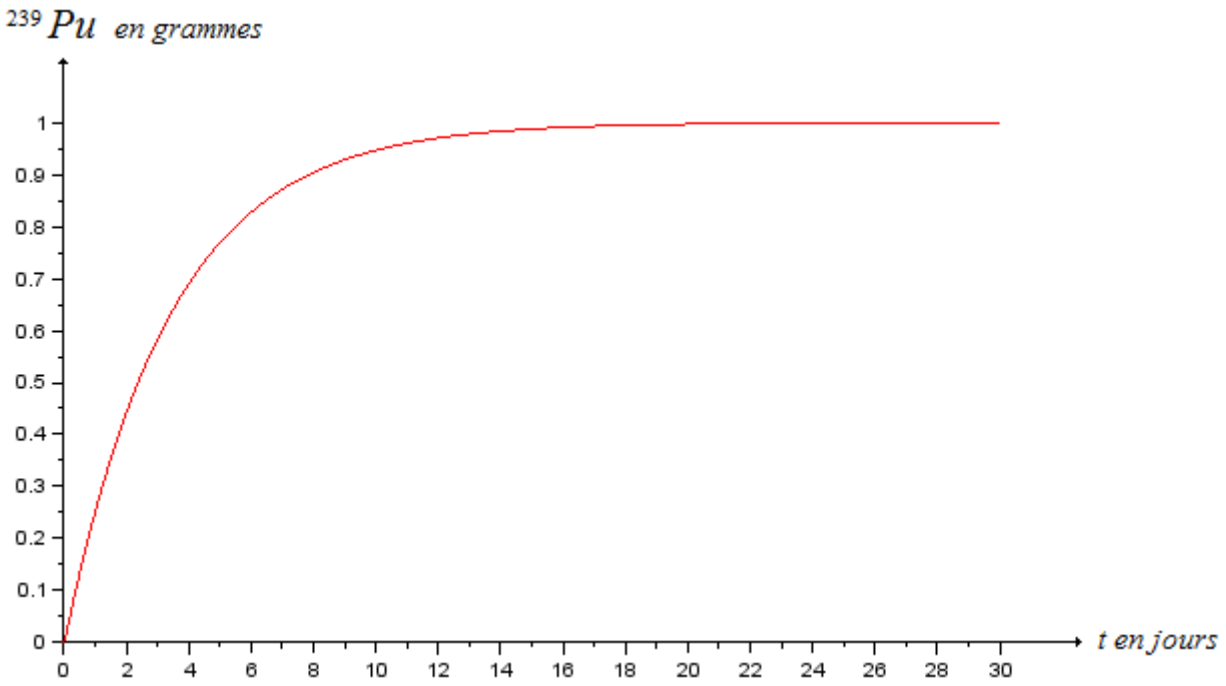
---

Notons λ_1 , λ_2 , λ_3 les constantes radioactives de l'uranium 239, du neptunium 239 et du plutonium 239.
Si m_0 désigne la masse initiale d'uranium 239, la masse m_3 de plutonium 239 au cours du temps est donnée par :

$$m_3 = m_0 \frac{M_3}{M_1} \lambda_1 \lambda_2 \left(\frac{e^{-\lambda_1 t}}{(\lambda_2 - \lambda_1)(\lambda_3 - \lambda_1)} + \frac{e^{-\lambda_2 t}}{(\lambda_1 - \lambda_2)(\lambda_3 - \lambda_2)} + \frac{e^{-\lambda_3 t}}{(\lambda_1 - \lambda_3)(\lambda_2 - \lambda_3)} \right)$$

La masse molaire de l'uranium 239 est M_1 = 239.0542933 g, celle du plutonium 239 est M_3 = 239.0521634 g.
En prenant comme unité de temps 1 jour et comme unité de masse 1 gramme, le programme Scilab de la courbe $m_3(t)$ est :

```
T1 = 23.45/60/24 ; T2 = 2.3565 ; T3 = 24110*365.25 ;
lambda1 = log(2)/T1 ; lambda2 = log(2)/T2 ; lambda3 = log(2)/T3 ;
M1 = 239.0542933 ; M3 = 239.0521634 ; m0 = 1 ;
tm = 30 ; t = linspace(0,tm,300) ;
m3 = m0*(M3/M1)*lambda1*lambda2*...
     (exp(-lambda1*t)/((lambda2 - lambda1)*(lambda3 - lambda1)) + ...
      exp(-lambda2*t)/((lambda1 - lambda2)*(lambda3 - lambda2)) + ...
      exp(-lambda3*t)/((lambda1 - lambda3)*(lambda2 - lambda3))) ;
plot2d(t,m3,style = 5)
```



Noter qu'ici $T_3 >> T_1$, $T_3 >> T_2$, $T_3 >> t$ d'où : $\lambda_1 - \lambda_3 \approx \lambda_1$, $\lambda_2 - \lambda_3 \approx \lambda_2$, $e^{-\lambda_3 t} \approx 1 \Rightarrow m_3 \approx m_0 \frac{M_3}{M_1} \left(1 - \frac{\lambda_2 e^{-\lambda_1 t} - \lambda_1 e^{-\lambda_2 t}}{\lambda_2 - \lambda_1} \right)$

Cette approximation revient à considérer que l'élément ^{239}Pu est stable dans l'intervalle de temps étudié.

Mathématique & Informatique

Application 2

Lors de la formation de la terre est apparu l'uranium 235 $\left({}^{235}_{92}U\right)$, élément naturellement radioactif qui se transforme progressivement en plomb 207 $\left({}^{207}_{82}Pb\right)$ stable. La chaîne de désintégration est la suivante :

$${}^{235}U \xrightarrow{\alpha} {}^{231}Th \xrightarrow{\beta} {}^{231}Pa \xrightarrow{\alpha} {}^{227}Ac \xrightarrow{\beta} {}^{227}Th \xrightarrow{\alpha} {}^{223}Ra \xrightarrow{\alpha} {}^{219}Rn \xrightarrow{\alpha} {}^{215}Po \xrightarrow{\alpha} {}^{211}Pb \xrightarrow{\beta} {}^{211}Bi \xrightarrow{\alpha} {}^{207}Tl \xrightarrow{\beta} {}^{207}Pb$$

Périodes radioactives des différents éléments :

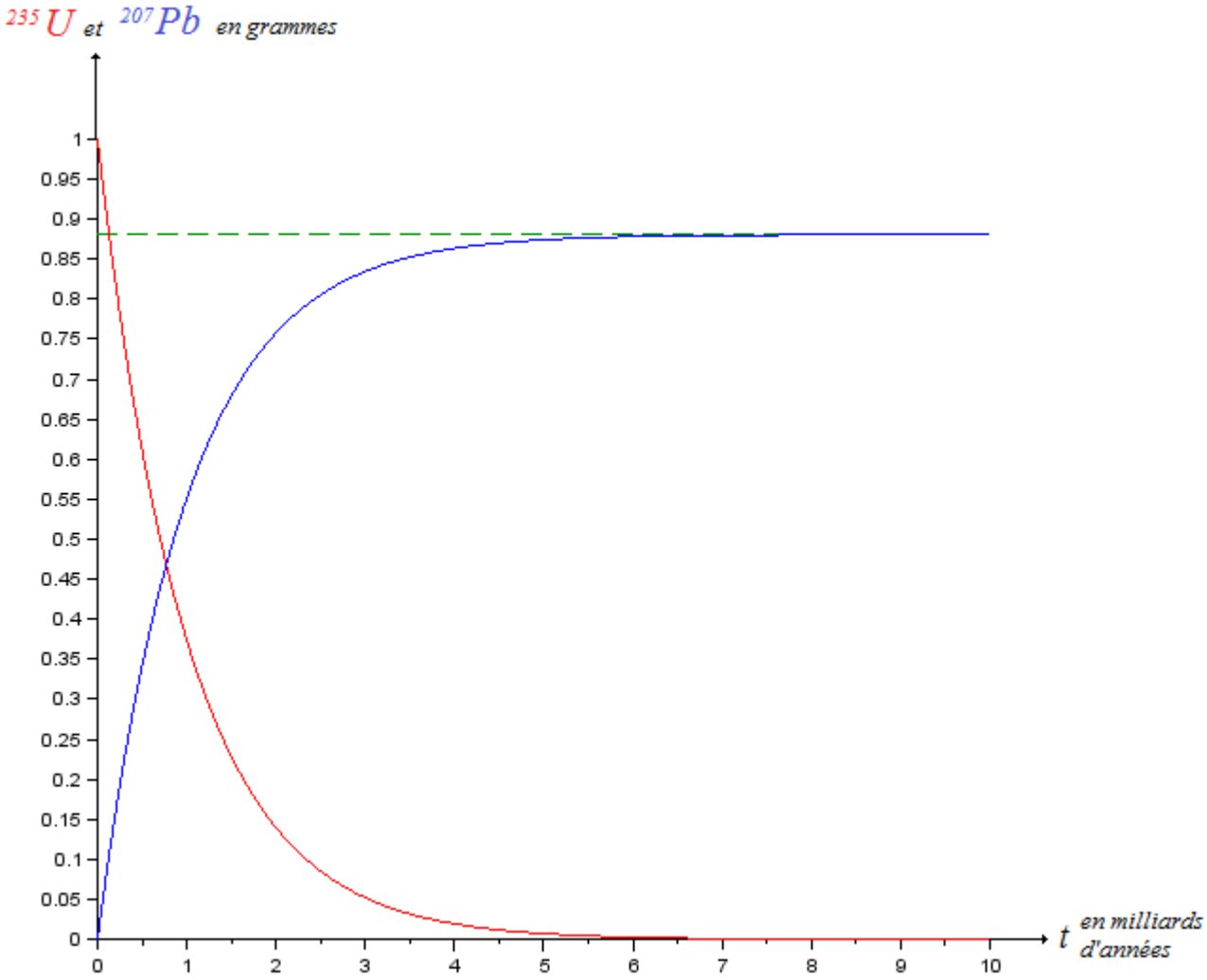
- Uranium 235 = 704 millions d'années
- Thorium 231 = 25.52 heures
- Protactinium 231 = 32 760 ans
- Actinium 227 = 21.772 ans
- Thorium 227 = 18.68 jours
- Radium 223 = 11.43 jours
- Radon 219 = 3.96 secondes
- Polonium 215 = 1.78 millisecondes
- Plomb 211 = 36.12 minutes
- Bismuth 211 = 2.14 minutes
- Thallium 207 = 4.77 minutes

On se fixe une échelle des temps en milliards d'années (l'âge de la terre est estimé à 5 milliards d'années).
Tracer la courbe donnant la masse de plomb 207 issue de la désintégration d'une masse initiale de 1 gramme d'uranium 235.
Tracer également les courbes donnant les masses des différents radioéléments intermédiaires apparaissant au cours du temps.

Programme Scilab des courbes de l'Uranium 235 et du Plomb 207 :

```
T1 = 704D6/1D9 ;
T2 = 25.52/24/365.25/1D9 ;
T3 = 32.76D3/1D9 ;
T4 = 21.772/1D9 ;
T5 = 18.68/365.25/1D9 ;
T6 = 11.43/365.25/1D9 ;
T7 = 3.96/3600/24/365.25/1D9 ;
T8 = 1.78D-3/3600/24/365.25/1D9 ;
T9 = 36.12/60/24/365.25/1D9 ;
T10 = 2.14/60/24/365.25/1D9 ;
T11 = 4.77/60/24/365.25/1D9 ;
T = [T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11] ;    // périodes en milliards d'années
lambda = log(2)./T ;    // constantes de radioactivité
n = 12 ;    // nombre d'éléments de la chaîne
m0 = 1 ;    // 1 gramme d'Uranium 235
M1 = 235.0439299 ;    // masse molaire de U 235
Mn = 206.9758969 ;    // masse molaire de Pb 207
tm = 10 ;    // intervalle de temps de 10 milliards d'années
p = 200 ;    // nombre de points de calcul de la courbe
t = linspace(0,tm,p) ;
sigma = zeros(1,p) ;
for k = 1:n-1
    a = 1 ;
    for j = 1:n-1
        if j <> k then
            a = a*lambda(j)/(lambda(j) - lambda(k)) ;
        end
    end
    sigma = sigma + a*exp(-lambda(k)*t) ;
end
mU = m0*exp(-lambda(1)*t) ;    // masse d'Uranium au cours du temps
mPb = m0*(Mn/M1)*(1 - sigma) ;    // masse de Plomb au cours du temps
plot2d([0,tm],[m0*Mn/M1,m0*Mn/M1])
e = gce() ; p = e.children ; p.line_style = 2 ; p.foreground = 13 ;
plot2d(t,mU,style = 5,frameflag = 1,rect = [0,0,tm,m0])
plot2d(t,mPb,style = 2,frameflag = 1,rect = [0,0,tm,m0])
```

Courbes obtenues :



Noter qu'à $t = 0$ on a $m_0 = 1$ g de ^{235}U et qu'à $t = \infty$ la masse de ^{207}Pb est $m_\infty = m_0 \dfrac{M_{Pb}}{M_U} = 1 \times \dfrac{207}{235} \approx 0.88 g$.

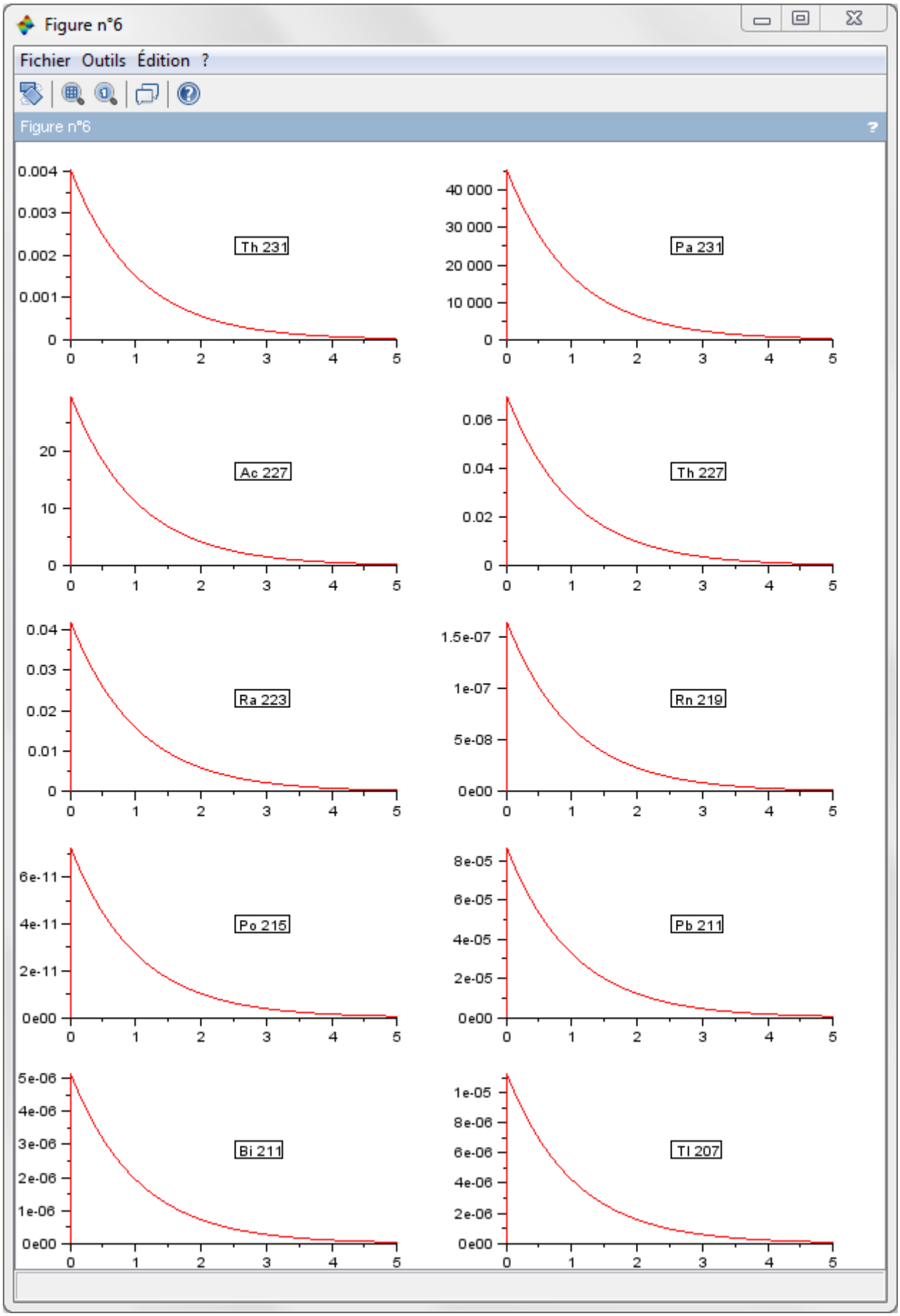
Cette perte de masse d'environ 0.12 g correspond aux émissions de particules α lors de la transmutation de l'uranium en plomb.

Mathématique & Informatique

Programme Scilab des courbes du Thorium 231 au Thallium 207 :

```
E1 = 'U 235' ;
E2 = 'Th 231' ;
E3 = 'Pa 231' ;
E4 = 'Ac 227' ;
E5 = 'Th 227' ;
E6 = 'Ra 223' ;
E7 = 'Rn 219' ;
E8 = 'Po 215' ;
E9 = 'Pb 211' ;
E10 = 'Bi 211' ;
E11 = 'Tl 207' ;
E = [E1,E2,E3,E4,E5,E6,E7,E8,E9,E10,E11] ;    // symboles des radioéléments
M1 = 235.0439299 ;
M2 = 231.0363043 ;
M3 = 231.0358840 ;
M4 = 227.0277521 ;
M5 = 227.0277041 ;
M6 = 223.0185022 ;
M7 = 219.0094802 ;
M8 = 214.9994200 ;
M9 = 210.9887370 ;
M10 = 210.987269 ;
M11 = 206.977419 ;
M = [M1,M2,M3,M4,M5,M6,M7,M8,M9,M10,M11] ;    // masses molaires des radioéléments
T1 = 704D6/1D9 ;
T2 = 25.52/24/365.25/1D9 ;
T3 = 32.76D3/1D9 ;
T4 = 21.772/1D9 ;
T5 = 18.68/365.25/1D9 ;
T6 = 11.43/365.25/1D9 ;
T7 = 3.96/3600/24/365.25/1D9 ;
T8 = 1.78D-3/3600/24/365.25/1D9 ;
T9 = 36.12/60/24/365.25/1D9 ;
T10 = 2.14/60/24/365.25/1D9 ;
T11 = 4.77/60/24/365.25/1D9 ;
T = [T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11] ;    // périodes des radioéléments
lambda = log(2)./T ;
n = 12 ;
m0 = 1D9 ;    // 10^9 nanogrammes d'Uranium 235
tm = 5 ;    // intervalle de temps de 5 milliards d'années
p = 500 ;
t = linspace(0,tm,p) ;
for i = 2:n-1
    sigma = zeros(1,p) ;
    for k = 1:i
        a = 1 ;
        for j = 1:i
            if j <> k then
                a = a*(lambda(j) - lambda(k)) ;
            end
        end
        sigma = sigma + exp(-lambda(k)*t)/a ;
    end
    b = 1 ;
    for j = 1:i-1
        b = b*lambda(j) ;
    end
    mi = m0*(M(i)/M(1))*b*sigma ;
    subplot(5,2,i-1)
    xstring(tm/2,max(mi)/2,E(i),0,1)
    plot2d(t,mi,style = 5,frameflag = 1,rect = [0,0,tm,max(mi)])
end
```

Mathématique & Informatique



Pour ces 10 courbes, le temps (abscisse) est exprimé en milliards d'années, la masse (ordonnée) en nanogrammes.

Mathématique & Informatique

Transformation exponentielle réversible

Une substance A peut se transformer, molécule par molécule, en une substance B selon une loi de décroissance exponentielle de paramètre α. Réciproquement, on suppose que B est instable et susceptible de redonner A selon une autre loi de décroissance exponentielle de paramètre β.
De manière symbolique, cette double transformation est notée :

$$A \rightleftarrows B$$

Exprimer les quantités de A et de B en fonction du temps t sachant qu'au départ (à $t = 0$) on a : $A = A_0$ et $B = 0$.
Montrer que l'on aboutit à un équilibre (à $t = \infty$), calculer les valeurs correspondantes A_∞ et B_∞.
Tracer l'allure des courbes $A(t)$ et $B(t)$ dans les 3 cas suivants : $\alpha < \beta$, $\alpha > \beta$, $\alpha = \beta$.

Désignons par A le nombre de molécules du composé A à l'instant t et par B le nombre de molécules du composé B à l'instant t.

L'équation différentielle régissant la transformation de A s'écrit :

$$\frac{dA}{dt} + \alpha A = \beta B$$

L'équation différentielle régissant la transformation de B s'écrit :

$$\frac{dB}{dt} + \beta B = \alpha A$$

D'où le système différentiel linéaire :

$$\begin{cases} \dfrac{dA}{dt} = -\alpha A + \beta B \\ \dfrac{dB}{dt} = \alpha A - \beta B \end{cases}$$

La matrice de ce système est : $M = \begin{pmatrix} -\alpha & \beta \\ \alpha & -\beta \end{pmatrix}$

- Calculons ses valeurs propres :

$$\det(\lambda I - M) = \begin{vmatrix} \lambda + \alpha & -\beta \\ -\alpha & \lambda + \beta \end{vmatrix} = \lambda[\lambda + (\alpha + \beta)] = 0 \quad \Rightarrow \quad \lambda_1 = 0 \ , \ \lambda_2 = -(\alpha + \beta)$$

- Calculons ses vecteurs propres :
  Associé à λ_1

$$\begin{pmatrix} \alpha & -\beta \\ -\alpha & \beta \end{pmatrix}\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \Rightarrow \quad V_1 = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$$

  Associé à λ_2

$$\begin{pmatrix} -\beta & -\beta \\ -\alpha & -\alpha \end{pmatrix}\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \Rightarrow \quad V_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

La solution générale du système différentiel est donnée par l'expression :

$$\begin{pmatrix} A \\ B \end{pmatrix} = c_1 e^{\lambda_1 t} V_1 \ + \ c_2 e^{\lambda_2 t} V_2 \qquad\qquad (c_1 \ , \ c_2) \in \mathbb{R}^2$$

Soit :

$$\begin{cases} A = \beta c_1 + c_2 e^{-(\alpha+\beta)t} \\ B = \alpha c_1 - c_2 e^{-(\alpha+\beta)t} \end{cases}$$

Les conditions initiales ($t = 0$) permettent de déterminer les constantes c_1 et c_2 :

$$\begin{cases} A_0 = \beta c_1 + c_2 \\ 0 = \alpha c_1 - c_2 \end{cases} \Rightarrow \quad c_1 = \frac{1}{\alpha+\beta} A_0 \ , \ c_2 = \frac{\alpha}{\alpha+\beta} A_0$$

Finalement, on obtient :

$$\boxed{\begin{aligned} A &= A_0 \frac{\beta}{\alpha+\beta}\left(1 + \frac{\alpha}{\beta} e^{-(\alpha+\beta)t}\right) \\ B &= A_0 \frac{\alpha}{\alpha+\beta}\left(1 - e^{-(\alpha+\beta)t}\right) \end{aligned}}$$

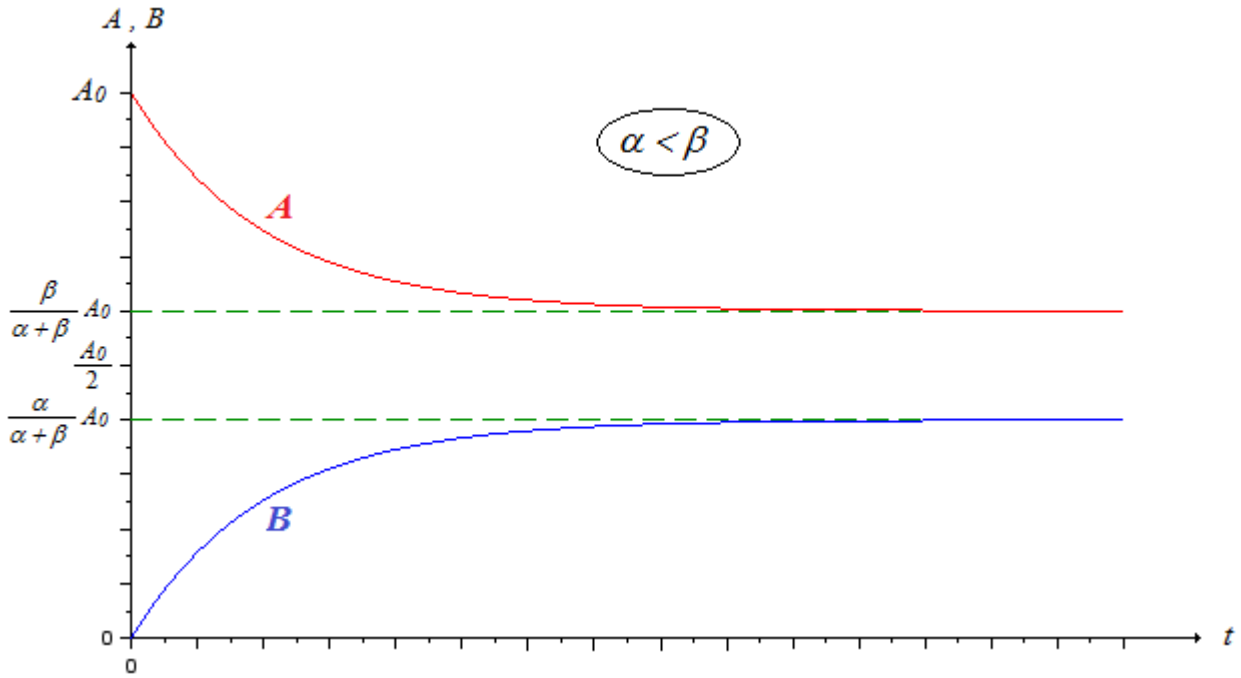
En permanence ($\forall\ t$), on a l'identité : $A + B = A_0$

A $t = \infty$, on aboutit à un équilibre caractérisé par les valeurs limites de A et de B : $A_\infty = \frac{\beta}{\alpha+\beta} A_0$, $B_\infty = \frac{\alpha}{\alpha+\beta} A_0$

Traçons les différentes courbes en comparant les paramètres α et β :

1) $\alpha < \beta$

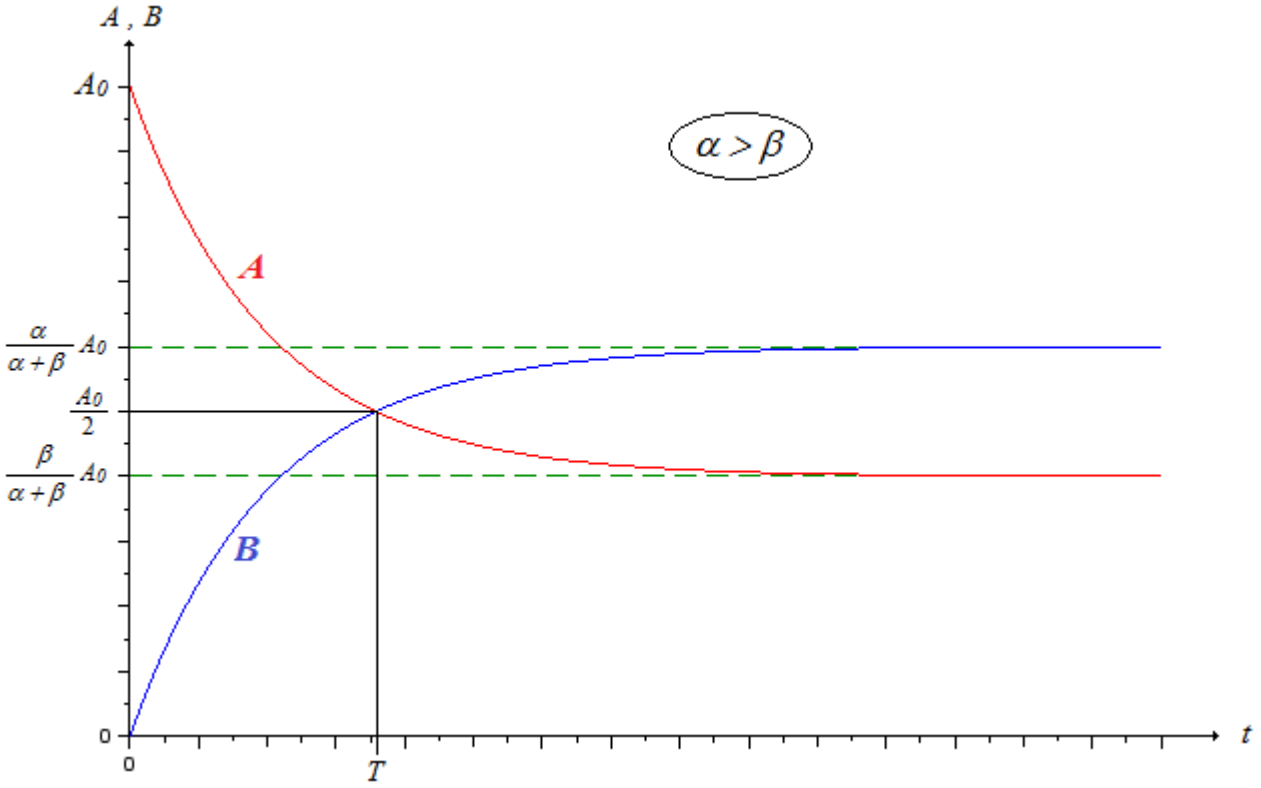
```
A0 = 10 ;
Alpha = 1 ;
Beta = 3/2 ;
tm = 3 ;
t = linspace(0,tm,300) ;
A = A0*(Beta/(Alpha + Beta))*(1 + (Alpha/Beta)*exp(-(Alpha + Beta)*t)) ;
B = A0*(Alpha/(Alpha + Beta))*(1 - exp(-(Alpha + Beta)*t)) ;
plot2d([0,tm],[A0*Alpha/(Alpha + Beta),A0*Alpha/(Alpha + Beta)])
e = gce() ; p = e.children ; p.line_style = 2 ; p.foreground = 13 ;
plot2d([0,tm],[A0*Beta/(Alpha + Beta),A0*Beta/(Alpha + Beta)])
e = gce() ; p = e.children ; p.line_style = 2 ; p.foreground = 13 ;
plot2d(t,A,style = 5)
plot2d(t,B,style = 2)
```



Si $\alpha << \beta$, les substances A et B restent pratiquement dans leur état initial au cours du temps : $A \approx A_0$ et $B \approx 0$.

2) $\alpha > \beta$

```
A0 = 10 ;
Alpha = 3/2 ;
Beta = 1 ;
tm = 3 ;
t = linspace(0,tm,300) ;
te = log((2*Alpha)/(Alpha - Beta))/(Alpha + Beta) ;
A = A0*(Beta/(Alpha + Beta))*(1 + (Alpha/Beta)*exp(-(Alpha + Beta)*t)) ;
B = A0*(Alpha/(Alpha + Beta))*(1 - exp(-(Alpha + Beta)*t)) ;
plot2d([0,tm],[A0*Alpha/(Alpha + Beta),A0*Alpha/(Alpha + Beta)])
e = gce() ; p = e.children ; p.line_style = 2 ; p.foreground = 13 ;
plot2d([0,tm],[A0*Beta/(Alpha + Beta),A0*Beta/(Alpha + Beta)])
e = gce() ; p = e.children ; p.line_style = 2 ; p.foreground = 13 ;
plot2d([0,te;te,te],[A0/2,0;A0/2,A0/2],style = [1,1])
plot2d(t,A,style = 5)
plot2d(t,B,style = 2)
```



En égalant les formules de A et de B, on établit que la courbe de A croise la courbe de B à l'instant $T = \dfrac{\ln\left(\dfrac{2\alpha}{\alpha-\beta}\right)}{\alpha+\beta}$

Si $\alpha >> \beta$, les substances A et B obéissent approximativement aux lois exponentielles suivantes :

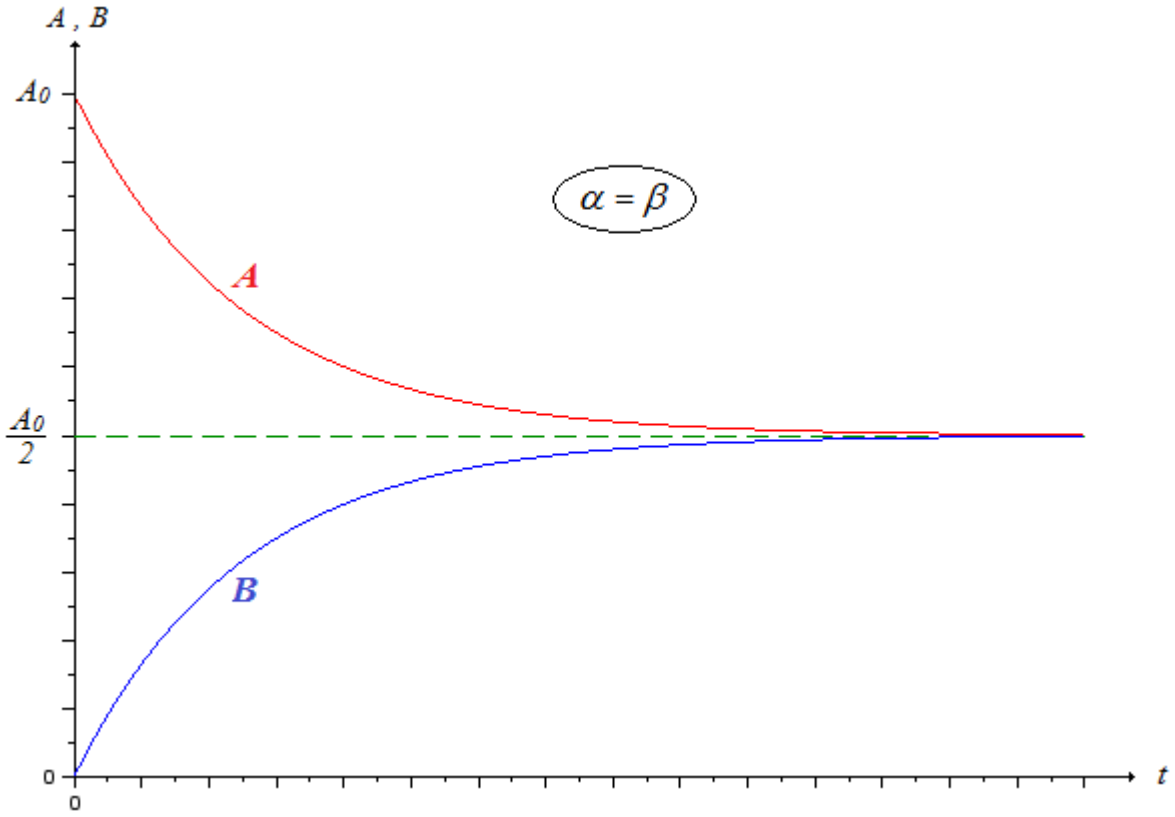
$$A \approx A_0 e^{-\alpha t}$$
$$B \approx A_0\left(1-e^{-\alpha t}\right)$$

En outre, sous cette même hypothèse, l'instant T pour lequel $A = B = \dfrac{A_0}{2}$ peut être estimé par :

$$T \approx \frac{\ln 2}{\alpha}$$

3) $\alpha = \beta$

```
A0 = 10 ;
Alpha = 1 ;
Beta = 1 ;
tm = 3 ;
t = linspace(0,tm,300) ;
A = A0*(Beta/(Alpha + Beta))*(1 + (Alpha/Beta)*exp(-(Alpha + Beta)*t)) ;
B = A0*(Alpha/(Alpha + Beta))*(1 - exp(-(Alpha + Beta)*t)) ;
plot2d([0,tm],[A0/2,A0/2])
e = gce() ; p = e.children ; p.line_style = 2 ; p.foreground = 13 ;
plot2d(t,A,style = 5)
plot2d(t,B,style = 2)
```



Donc :

- Si $\alpha < \beta$, la courbe de A est toujours au dessus de celle de B.
- Si $\alpha > \beta$, la courbe de A se situe d'abord au dessus de la courbe de B puis au dessous.
- Si $\alpha = \beta$, les 2 courbes tendent vers une limite commune $A = B = \frac{A_0}{2}$

Mathématique & Informatique

TRACÉ DES FONCTIONS PÉRIODIQUES

Soit une fonction f de période T.

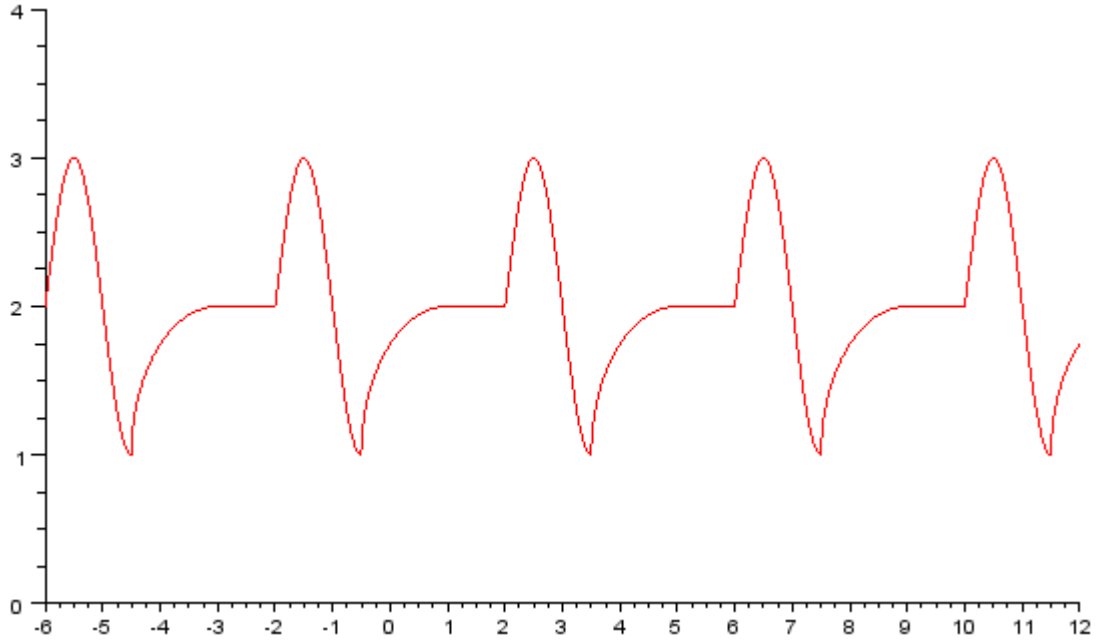
$$\forall\, X \in \left]-\infty\,,+\infty\right[\,,\ \exists\, x \in \left[\,x_0\,,\,x_0+T\,\right[\ :\qquad x = X - E\left(\frac{X-x_0}{T}\right)T \quad\Rightarrow\quad f(x) = f(X)$$

Prenons l'exemple de la fonction $y = f(x)$, de période 4, définie sur l'intervalle [1,5[par les équations :

- $1 \le x < 2 \ :\quad y = 2$
- $2 \le x < \frac{7}{2} \ :\quad y = 2 + \sin(\pi(x-2))$
- $\frac{7}{2} \le x < 5 \ :\quad y = 1 + \sqrt{1 - \frac{4}{9}(x-5)^2}$

Traçons avec Scilab la courbe représentative de la fonction périodique f sur l'intervalle [-6,12] :

```
T = 4 ;
x0 = 1 ; x1 = 5 ; X0 = -6 ; X1 = 12 ;
X = linspace(X0,X1,1000) ;
x = X - floor((X - x0)/T)*T ;
Y = bool2s(1 <= x & x < 2).*(2) + ...
    bool2s(2 <= x & x < 7/2).*(2 + sin(%pi*(x - 2))) + ...
    bool2s(7/2 <= x & x < 5).*(1 + sqrt(1 - 4/9*(x - 5)^2)) ;
a = gca() ; a.x_location = "origin" ; a.y_location = "left" ;
plot2d(X,Y,frameflag = 1,rect = [-6,0,12,4],nax = [3,19,3,5],style = 5)
```



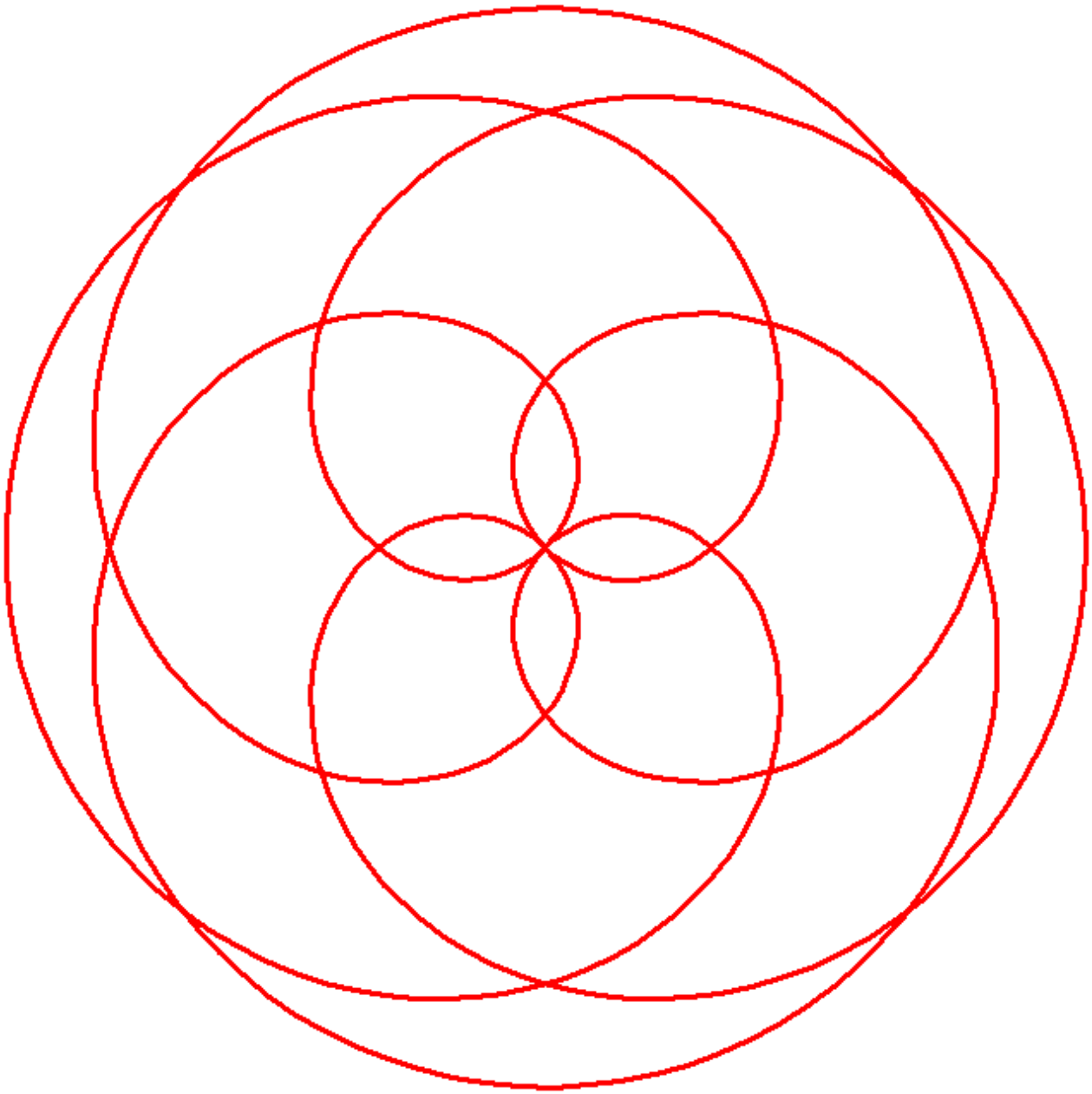
Explications

- L'instruction $x = X - floor((X - x0)/T)*T$ effectue le changement de variable $X \in \left]-\infty\,,+\infty\right[$ en $x \in \left[\,x_0\,,\,x_0+T\,\right[$ afin de se placer dans le domaine de définition de la fonction f de période T. On a : $Y = y$
- L'instruction *bool2s(test)* fournit la matrice entière constituée de *1* et de *0*, équivalente à la matrice booléenne dont les éléments sont des constantes prédéfinies *%t* (vrai) ou *%f* (faux) résultant du test arithmétique passé en paramètre.
- L'instruction *plot2d* avec l'option *frameflag = 1,rect = [x_{min} , y_{min} , x_{max} , y_{max}]* place la courbe dans un rectangle dont le sommet inférieur-gauche et le sommet supérieur-droit ont pour coordonnées respectives (x_{min} , y_{min}) et (x_{max} , y_{max}). L'option *nax = [n_x , N_x , n_y , N_y]* précise que l'axe des x possède N_x graduations principales avec n_x subdivisions entre 2 graduations (idem sur l'axe des y avec N_y et n_y). L'option *style = n_c* attribue le numéro de couleur n_c à la courbe.

Mathématique & Informatique

// Rosace

```
R = 1 ; p = 2 ; q = 5 ;
angle = linspace(0, 2*%pi*q, 500) ; rayon = R*cos(p/q*angle) ;
x = rayon.*cos(angle) ; y = rayon.*sin(angle) ;
xset("thickness", 3) ; plot2d(x, y, style = 5, frameflag = 4, axesflag = 0)
```

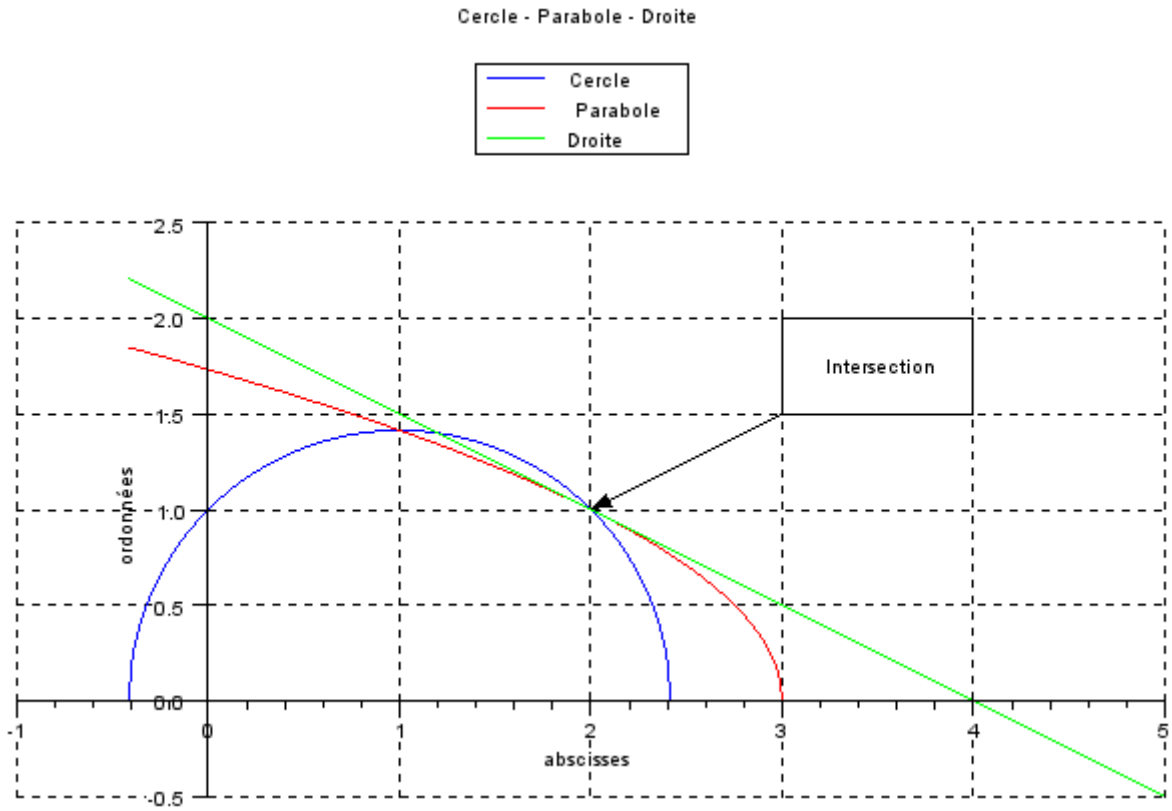


Pour n entier, r = R cos nθ (ou r = R sin nθ) donne une rosace à n feuilles si n est impair et une rosace à 2n feuilles si n est pair.

Mathématique & Informatique

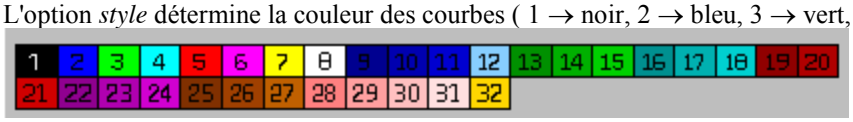
// Graphique légendé

```
x1 = linspace(1 - sqrt(2), 1 + sqrt(2), 300) ;
y1 = sqrt(1 - x1^2 + 2*x1) ;
x2 = linspace(1 - sqrt(2), 3, 300) ;
y2 = sqrt(3 - x2) ;
x3 = linspace(1 - sqrt(2), 5, 300) ;
y3 = 2 - x3/2 ;
a = gca() ; a.x_location = "origin" ; a.y_location = "origin" ;
plot2d([x1; x2; x3]', [y1; y2; y3]', style = [2, 5, 3], frameflag = 4) ;
xgrid(1) ; xtitle("Cercle - Parabole - Droite", "abscisses", "ordonnées") ;
xarrows([3,2], [1.5,1], 3) ; xrect(3, 2, 1, 0.5) ; xstringb(3, 1.5, "Intersection", 1, 0.5) ;
legends(["Cercle", "Parabole", "Droite"], [2, 5, 3]) ;
```



Remarques :

- L'option *style* détermine la couleur des courbes (1 → noir, 2 → bleu, 3 → vert, 5 → rouge, ...)

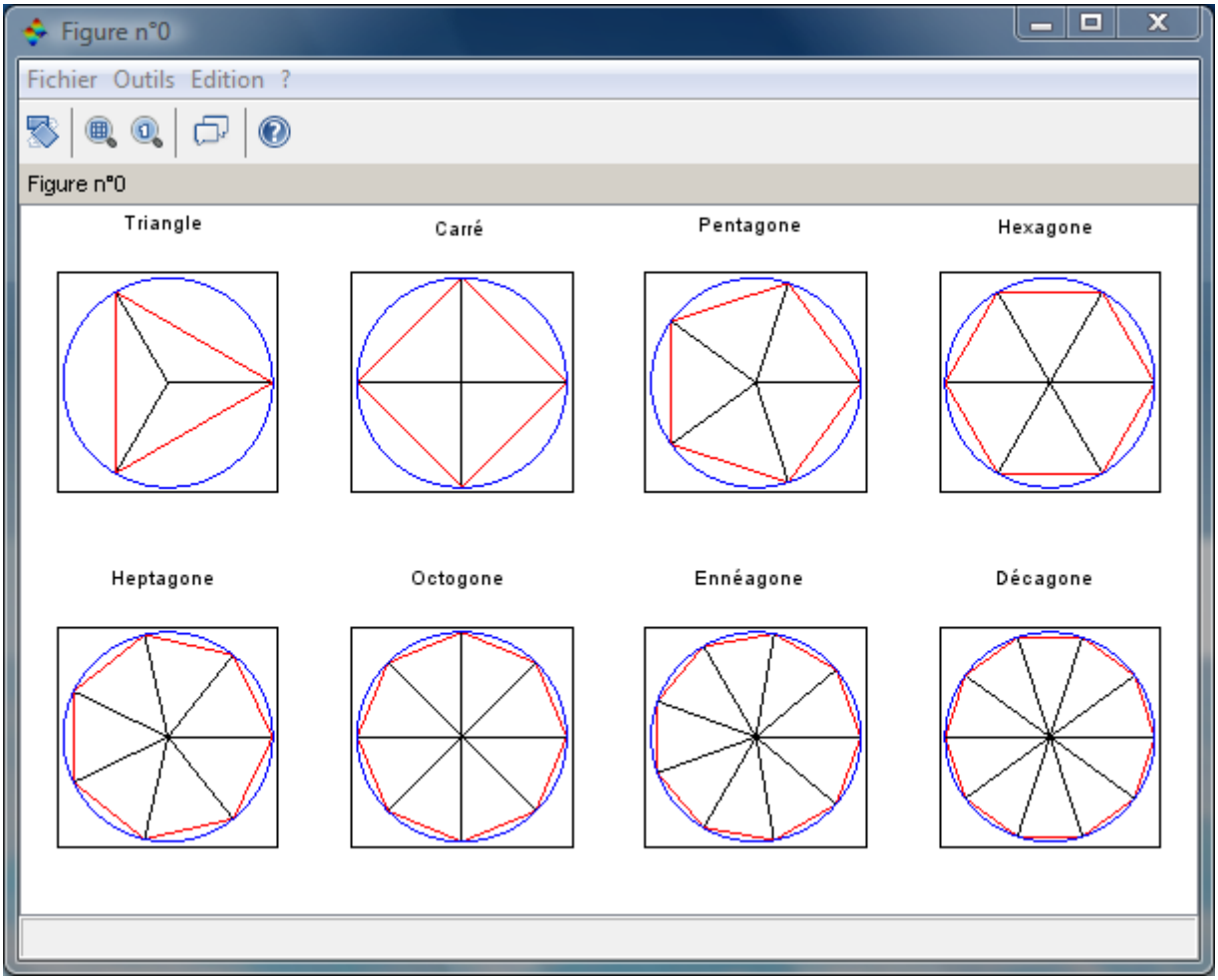


  Entrer la commande " – – > getcolor " et cliquer sur une couleur pour afficher ses caractéristiques.
- L'option *frameflag = 4* permet d'obtenir la même échelle sur les 2 axes.
- L'instruction *xgrid* dessine une grille en pointillés.
- L'instruction *xtitle* attribue un titre au graphique et un nom aux axes.
- Les instructions *xarrows*, *xrect*, et *xstringb* ajoutent une flèche et une boîte de texte.
- L'instruction *legends* insère une légende encadrée à l'endroit de son choix avec la souris.

Mathématique & Informatique

```
// Polygones réguliers
function polygone(n)
  // Cercle circonscrit
      R = 1.9 ; t = linspace(0, 2*%pi, 200) ;
      xc = R*cos(t) ; yc = R*sin(t) ;
  // Sommets
      xs = R*cos([0:n]*2*%pi/n) ; ys = R*sin([0:n]*2*%pi/n) ;
  // Rayons
      xr = [zeros(1,n) ; xs(1:n)] ; yr = [zeros(1,n) ; ys(1:n)] ;
  // Dessin
      plot2d(xr, yr, style = ones(1,n), frameflag = 4, axesflag = 2) ;
      plot2d(xs, ys, style = 5, frameflag = 4, axesflag = 2) ;
      plot2d(xc, yc, style = 2, frameflag = 4, axesflag = 2) ;
endfunction

Nom = ["Triangle", "Carré", "Pentagone", "Hexagone", "Heptagone", "Octogone", "Ennéagone", "Décagone"] ;
for n = 3:10
  subplot(2, 4, n-2) ; xtitle(Nom(n-2)) ; polygone(n) ;
end
```



Rappel : Les polygones à n = 7 et n = 9 côtés ne peuvent pas être tracés avec la règle et le compas (**Gauss-Wantzel**)

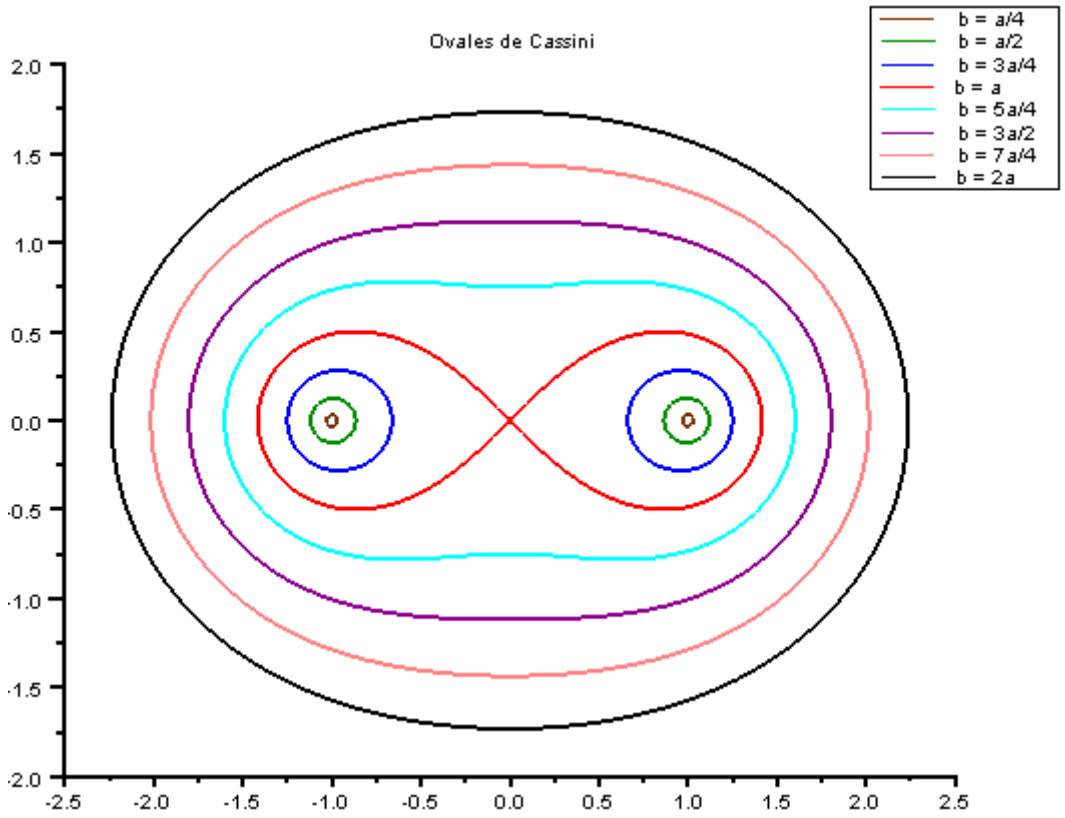
// **Ovales de Cassini**

```
function z = Cassini(x, y)
   z = ((x + a)^2 + y^2) * ((x - a)^2 + y^2)
endfunction

a = 1 ; b = [a/4   a/2   3*a/4   a   5*a/4   3*a/2   7*a/4   2*a] ;
xmax = sqrt(5)*a ; x = linspace(-xmax, +xmax, 300) ;
ymax = sqrt(3)*a ; y = linspace(-ymax, +ymax, 300) ;

xset("fpf", " ") ; xset("thickness", 2) ;
contour2d(x, y, Cassini, b^4, style = [25 13 2 5 4 22 28 1], frameflag = 4) ;

xtitle("Ovales de Cassini") ;
legends(["b = a/4" "b = a/2" "b = 3a/4" "b = a" "b = 5a/4" "b = 3a/2" "b = 7a/4" "b = 2a"], [25 13 2 5 4 22 28 1]) ;
```



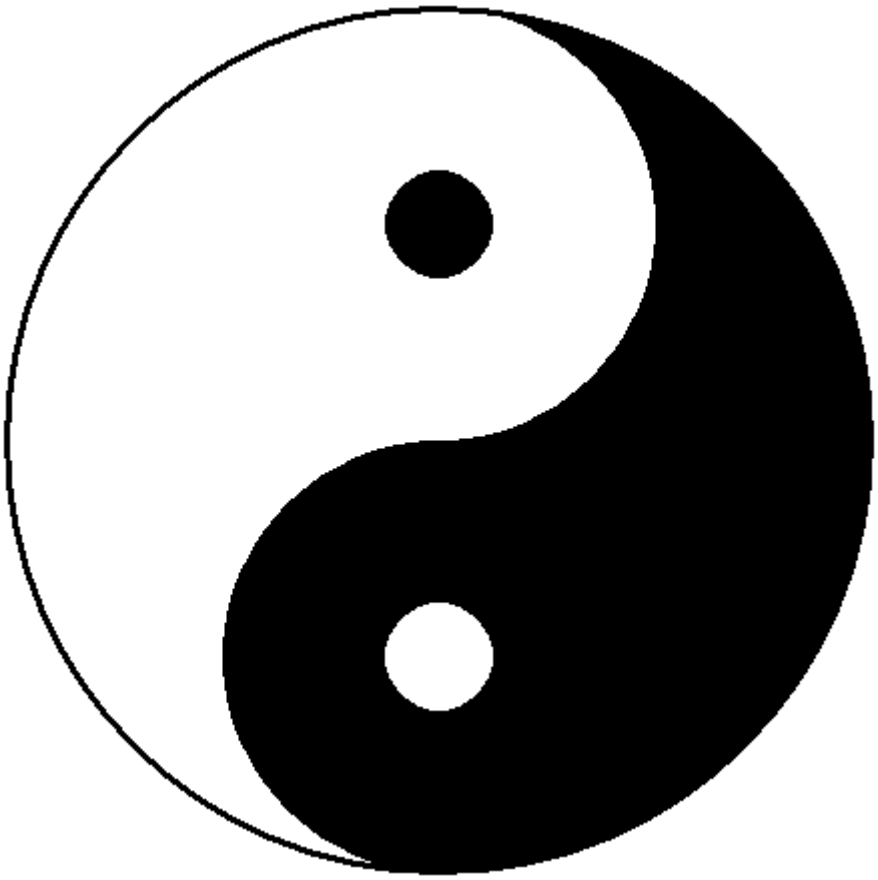
Remarques :

- L'instruction *contour2d*(x, y, f, C, options) trace les lignes de niveau de la fonction f, c'est-à-dire les courbes constituées par l'ensemble des points (x,y) du plan qui ont la même cote z = f(x,y) = C.
  En attribuant N valeurs à C, rassemblées dans le vecteur C = [C_1 , ... , C_N], on obtient N courbes.
  L'option frameflag = 4 impose une échelle isométrique afin de conserver les proportions.
- Le programme suivant :
  deff('z = f(x,y)' , 'z = équation en x et y') ; contour2d(xmin:p_x:xmax , ymin:p_y:ymax , f , [0,0] , frameflag = 4) dessine la courbe plane définie par l'équation cartésienne f(x,y) = 0, sans qu'il soit nécessaire d'exprimer y en fonction de x, ni de paramétrer x et y. Par exemple, pour représenter l'ovale de Cassini a = 1 et b = 2, on écrit :
  deff('z = f(x,y)', 'z=(x^2+y^2+1)^2-4*x^2-16') ; contour2d(-3:0.06:3, -2:0.04:2, f, [0,0], frameflag=4)

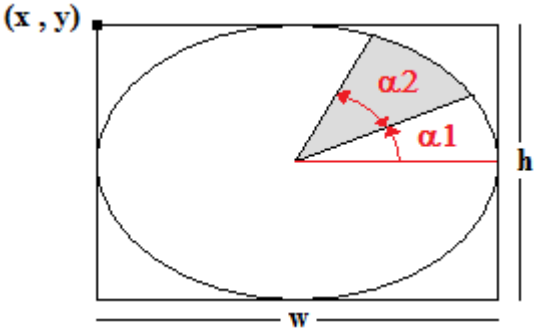
Mathématique & Informatique

// Yin - Yang

```
R = 1 ; noir = color(0, 0, 0) ; blanc = color(255, 255, 255) ;
plot2d(%inf, %inf, frameflag = 3, rect = [-R, -R, +R, +R], axesflag = 0)
xset("color", blanc) ; xfarc(-R, +R, 2*R, 2*R, 64*90, 64*180)
xset("color", noir)   ; xfarc(-R, +R, 2*R, 2*R, 64*270, 64*180)
xset("color", blanc) ; xfarc(-R/2, +R, R, R, 0, 64*360)
xset("color", noir)   ; xfarc(-R/2, 0, R, R, 0, 64*360)
xset("color", blanc) ; xfarc(-R/8, -R/2 + R/8, R/4, R/4, 0, 64*360)
xset("color", noir)   ; xfarc(-R/8, +R/2 + R/8, R/4, R/4, 0, 64*360)
xset("thickness", 3) ; xarc(-R, +R, 2*R, 2*R, 0, 64*360)
```



Remarque :



Les fonctions *xarc*(x,y,w,h,a_1,a_2) et *xfarc*(x,y,w,h,a_1,a_2) sont utilisées pour tracer un arc d'ellipse et remplir un secteur d'ellipse.
Formules en degrés reliant a_1 et a_2 aux angles α_1 et α_2 :

$$a_1 = 64 * \alpha_1$$
$$a_2 = 64 * \alpha_2$$

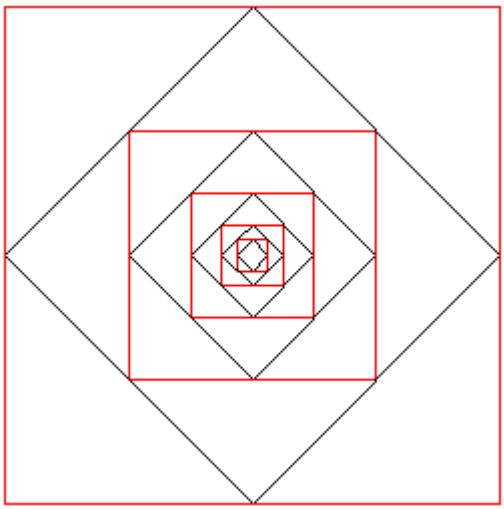
NB. Un cercle est une ellipse particulière où w = h

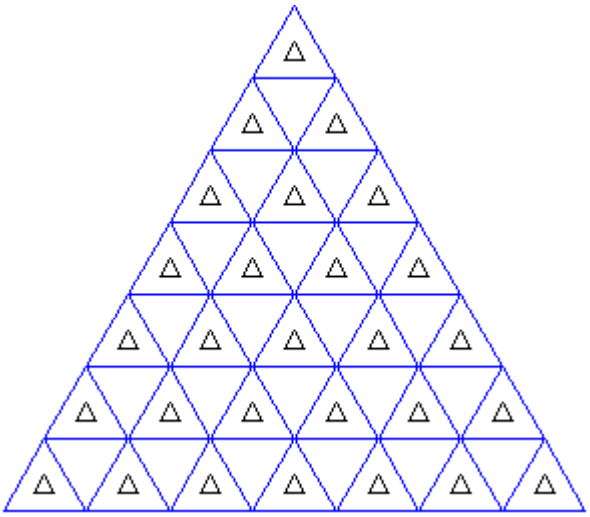
Mathématique & Informatique

```
// Motifs
function carre(x, y, c, n)
  if n > 0 then
     carre(x+c/4, y-c/4, c/2, n-1)
     plot2d([x+c/2,x+c,x+c/2,x,x+c/2], [y,y-c/2,y-c,y-c/2,y], style=1, frameflag=3, rect=dim, axesflag=0)
     plot2d([x,x+c,x+c,x,x], [y,y,y-c,y-c,y], style=5, frameflag=3, rect=dim, axesflag=0)
  end
endfunction

function triangle(x, y, h, n)
  if n > 0 then
     triangle(x-h/sqrt(3), y-h, h, n-1) , triangle(x+h/sqrt(3), y-h, h, n-1)
     plot2d([x,x-h/sqrt(3),x+h/sqrt(3),x], [y,y-h,y-h,y], style=2, frameflag=3, rect=dim, axesflag=0)
     plot2d(x, y-2*h/3, style=-6, frameflag=3, rect=dim, axesflag=0)
  end
endfunction

drawlater()
n = 5 ; c = 9 ; dim = c/2*[-1,-1,+1,+1] ; carre(-c/2, +c/2, c, n)
drawnow()
xclick() ; clf() ;
drawlater()
n = 7 ; h = 3 ; dim = n*h*[-1,0,+1,+1] ; triangle(0, n*h, h, n)
drawnow()
```





Remarques :

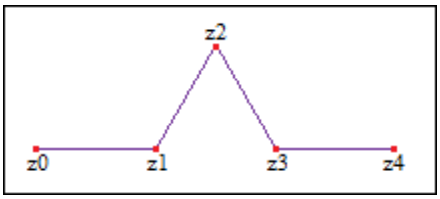
- Avec l'option *frameflag = 3* , *rect = [x_{min} , y_{min} , x_{max} , y_{max}]* on définit le rectangle qui contiendra le dessin.
- Avec l'instruction *drawlater()* le résultat d'un appel à plot2d n'est pas visualisé immédiatement, mais enregistré dans une mémoire tampon (buffer = {pixels}). Une fois terminé, le dessin est affiché à l'écran avec l'instruction *drawnow()*
- Avec les 2 instructions *xclick() ; clf()* on attend un clic de souris avant d'effacer le contenu de la fenêtre.
- La fonction plot2d utilisée avec l'option style = - k (≤ 0) ne trace pas des traits entre les points, mais remplace chaque point par un symbole représenté par l'entier k $\in$ {0,1,2, ... ,14} (0 → •, 1 → +, 2 → ×, ... , 14 → ✯).
  La taille i $\in$ {0, ... ,5} du symbole k peut être modifiée avec une instruction *xset*("mark size", i) ; par défaut i = 0.
  Ouvrir le dossier *Aide de Scilab > Graphiques > Segments properties* pour afficher la liste des symboles utilisés par Scilab ou utiliser l'éditeur graphique *ged*.
- La fonction récursive "triangle" dessine plusieurs fois les mêmes petits triangles. Son exécution est en 2^n , donc exponentielle en n, d'où sa lenteur pour les grandes valeurs de n. On peut construire un algorithme en n^2 qui optimise cette fonction (ENSIMAG - Algorithmique - Pierre **Berlioux**).

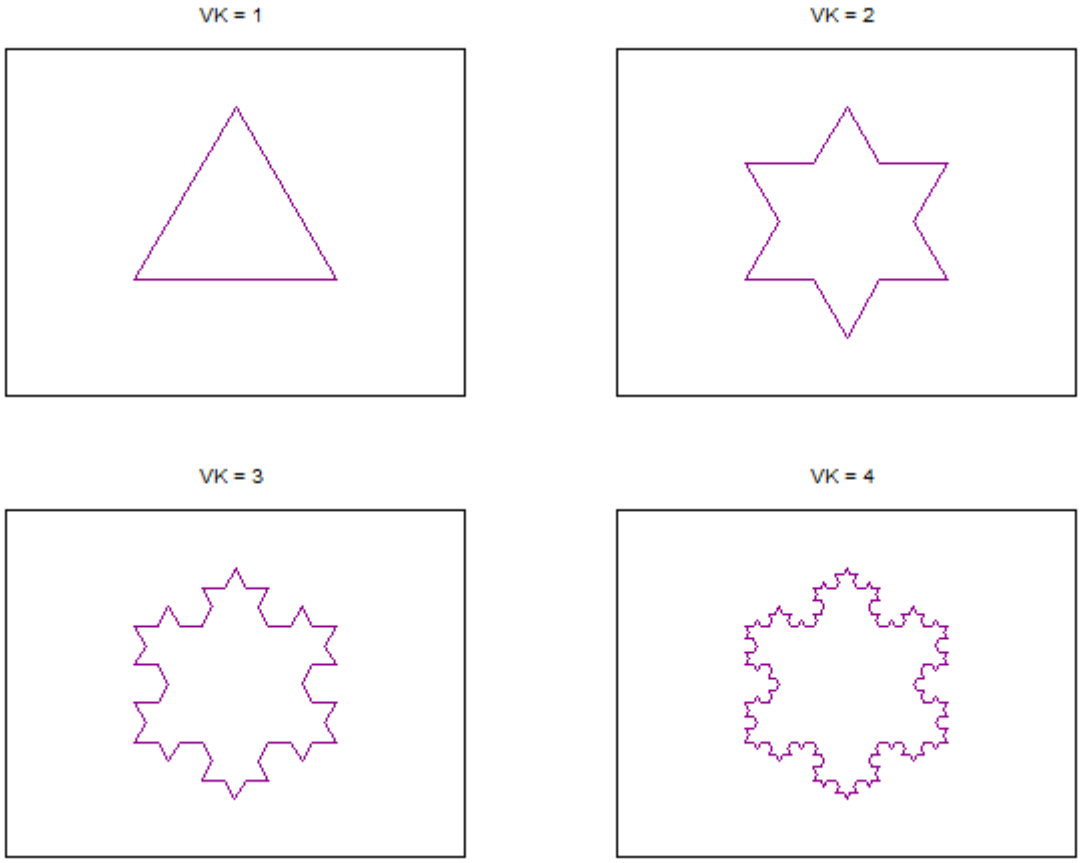
Mathématique & Informatique

// Flocon de Von Koch

```
function VonKoch(za, zb, VK)
  seg = 4^(VK-1) , z = zeros(1, seg+1) , z(1) = za , z($) = zb
  theta = exp(%i*%pi/3) , pas = seg
  while pas < > 1
    pas = pas/4
    for i = 1:4*pas:seg-3
      delta = (z(i+4*pas) - z(i))/3
      z(i+pas) = z(i) + delta
      z(i+2*pas) = z(i+pas) + delta*theta
      z(i+3*pas) = z(i+4*pas) - delta
    end
  end
  plot2d(real(z), imag(z), style = 22, frameflag = 3, rect = 1.5*[-1,-1,+1,+1], axesflag = 2)
endfunction

n = 3 ;
for VK = 1:4
  subplot(2, 2, VK) , xtitle("VK = " + string(VK))
  for k = 0:n-1 , VonKoch(exp(%i*(%pi/2 - 2*k*%pi/n)), exp(%i*(%pi/2 - 2*(k+1)*%pi/n)), VK) , end
end
```





Remarques :

- Les calculs se font dans $\mathbb{C}$. Le nombre complexe z(k) = x(k) + i y(k) est l'affixe du point P_k(x(k) , y(k)).
- Quand VK $\to \infty$, la courbe fermée obtenue, infiniment anguleuse, reste continue mais dérivable en aucun point. Son périmètre est infini et sa surface intérieure est finie ($\frac{8}{5}$ Triangle). C'est un des premiers exemples de fractale.

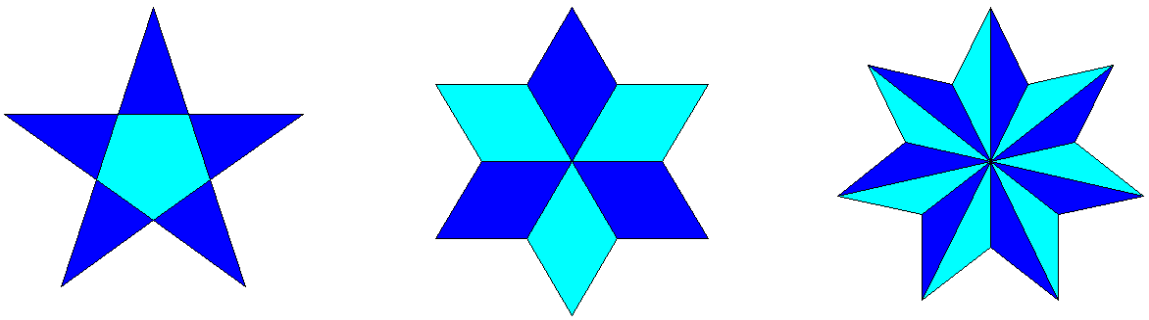
Mathématique & Informatique

```
// Etoiles
function E = etoile(b, R, r)
  E = zeros(2, 2*b)
  for k = 0:b-1
    E(:, 2*k+1) = R*[cos(%pi/2 + 2*k*%pi/b) ; sin(%pi/2 + 2*k*%pi/b)]
    E(:, 2*k+2) = r*[cos(%pi/2 + (2*k+1)*%pi/b) ; sin(%pi/2 + (2*k+1)*%pi/b)]
  end
endfunction

E = etoile(5, 1, cos(2*%pi/5)/cos(%pi/5)) ;
subplot(1, 3, 1) ; plot2d(%inf, %inf,   frameflag = 3, rect = [-1,-1,+1,+1], axesflag = 0) ;
xfpolys(E(1, :)', E(2, :)', 2) ;
xfpolys(E(1, 2:2:10)', E(2, 2:2:10)', 4) ;

E = etoile(6, 1, 1/sqrt(3)) ;
subplot(1, 3, 2) ; plot2d(%inf, %inf,   frameflag = 3, rect = [-1,-1,+1,+1], axesflag = 0) ;
xfpolys(E(1, :)', E(2, :)', 2) ;
xfpolys([zeros(1, 3) ; E(1, [2,6,10]) ; E(1, [3,7,11]) ; E(1, [4,8,12])], ...
        [zeros(1, 3) ; E(2, [2,6,10]) ; E(2, [3,7,11]) ; E(2, [4,8,12])], [4,4,4]) ;

E = etoile(7, 1, 1/(2*cos(%pi/7))) ;
subplot(1, 3, 3) ; plot2d(%inf, %inf,   frameflag = 3, rect = [-1,-1,+1,+1], axesflag = 0) ;
xfpolys(E(1, :)', E(2, :)', 2) ;
xfpolys([zeros(1, 7) ; E(1, 1:2:13) ; E(1, 2:2:14)], [zeros(1,7) ; E(2, 1:2:13) ; E(2, 2:2:14)], 4*ones(1, 7)) ;
```

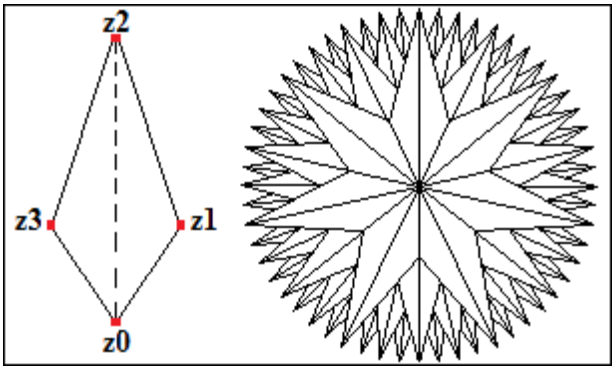


Remarques **:**

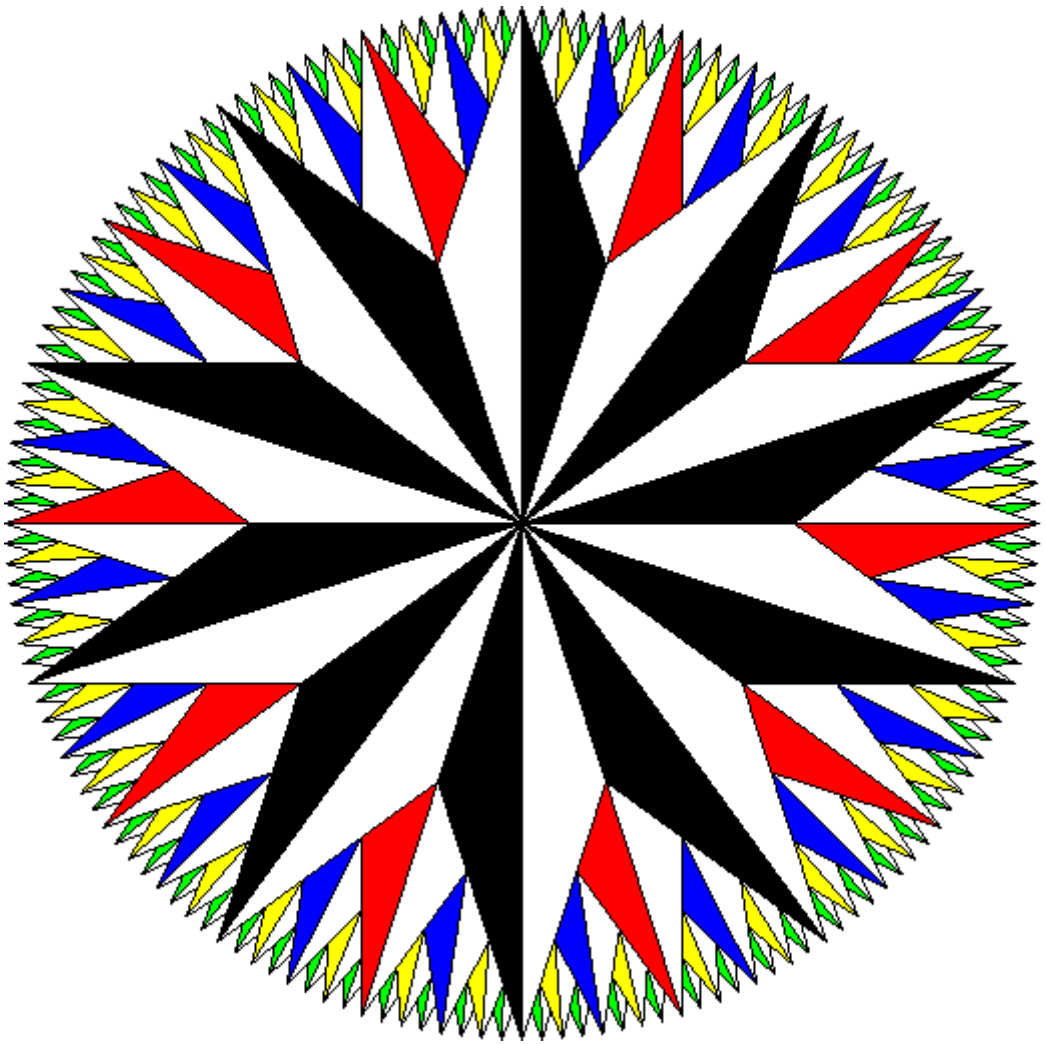
- L'instruction *subplot*(p, q, r) divise la fenêtre graphique en un tableau virtuel de p lignes et q colonnes, puis sélectionne la case numéro r = (i - 1)*q + j (numérotation de gauche à droite et de haut en bas).
- L'instruction *xfpolys*(x, y, [c_1, ... , c_N]) remplit N polygones avec les couleurs c_1 à c_N.
  Chaque colonne des matrices x et y correspond aux coordonnées des sommets d'un polygone.
- Les notations E(i, :) et E(:, j) désignent respectivement la ligne i et la colonne j de la matrice E.
- Scilab respecte la casse, donc les majuscules sont différenciées des minuscules (R ≠ r).
- Pour faire tourner l'étoile d'un angle theta, appliquer la fonction *rotate*(E, theta). L'angle theta est exprimé en radians et compté positivement dans le sens trigonométrique, c'est-à-dire inverse des aiguilles d'une montre (↺).

Mathématique & Informatique

```
// Fractale étoilée
function Star(b, R, r, n, C)
  z0 = 0 , z1 = r*exp(%i*(%pi/2 - %pi/b))
  z2 = R*%i , z3 = r*exp(%i*(%pi/2 + %pi/b))
  Z = [z0,z0;z1,z2;z2,z3]
  for p = n:-1:1
    for q = 1:min(p,2):2^(p-1)*b
      S = Z*exp(%i*q*%pi/2^(p-2)/b)
      xfpolys(real(S), imag(S), C(p,:))
    end
  end
endfunction
```




```
b = 10 ; R = 1 ; r = 1/(2*cos(%pi/10)) ; n = 5 ; C = [1,8;5,8;2,8;7,8;3,8] ;
a = gca() ; a.axes_visible = 'off' ; a.isoview = 'on' ; a.data_bounds = [-R,-R;+R,+R] ;
drawlater() ; Star(b, R, r, n, C) ; drawnow() ;
```



Quand $n \to \infty$, le contour de l'étoile est une courbe fermée de longueur finie inscrite dans le cercle de rayon R.

Son périmètre P tend vers $2\pi R\sqrt{1+\left[\frac{R/r - \cos(\pi/b)}{\sin(\pi/b)}\right]^2}$ et sa surface intérieure vers πR^2.

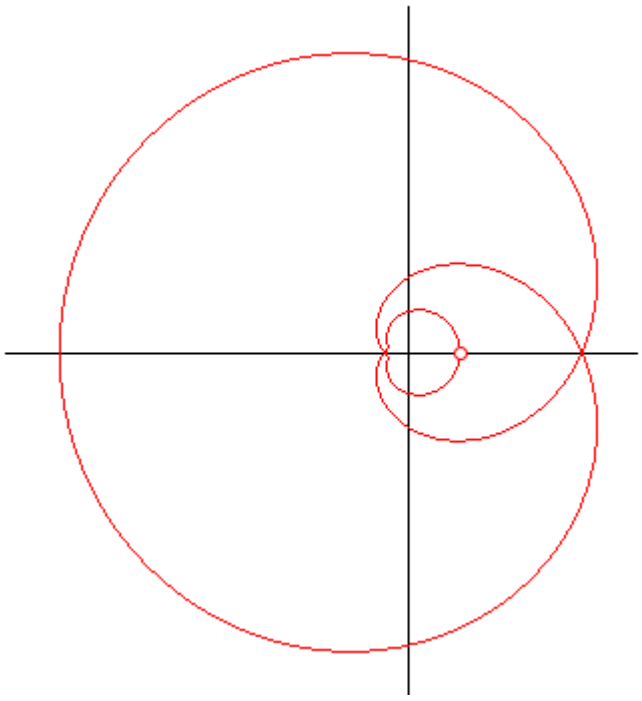
La fonction comet

Soit $P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ un polynôme à coefficients réels et soit z le nombre complexe défini par $z = Re^{i\theta}$. Lorsque θ varie de 0 à 2π, z parcourt le cercle de rayon R et $P(z)$ décrit une courbe plane représentable avec la fonction *comet*. On montre le résultat suivant : le nombre de fois où la courbe enveloppe l'origine des coordonnées est égal au nombre de racines de l'équation $P(x) = 0$ appartenant au disque de centre 0 et de rayon R.
Autrement dit, pour un rayon R donné, si la comète contourne k fois le centre du repère, alors le polynôme $P(x)$ possède exactement k racines (réelles ou complexes) de module inférieur à R.

Par exemple, prenons $P(x) = x^5 - 11x^4 + 101x^3 - 301x^2 + 860x - 650$ et $R = 6$.

```
x = poly(0,'x') ;
P = x^5 - 11*x^4 + 101*x^3 - 301*x^2 + 860*x - 650 ;
R = 6 ;
theta = linspace(0,2*%pi,300) ;
z = R*exp(%i*theta) ;
Z = horner(P,z) ;
a = gca() ;
a.x_location = "origin" ; a.y_location = "origin" ;
a.isoview = 'on' ; a.auto_ticks = 'off' ;
comet(real(Z),imag(Z),'colors',5)
```

On obtient la courbe ci-dessous :



On voit que le tracé enveloppe 3 fois l'origine.
Donc, parmi les 5 racines de $P(x)$, exactement 3 d'entre elles ont un module inférieur à 6.

Mathématique & Informatique

Graphisme en 3 dimensions :

On utilise la fonction ***param3d*** pour représenter une courbe dans l'espace :

param3d(x, y, z, options)

x est le vecteur ligne des abscisses : (1 , n) ⇒ x(i)
y est le vecteur ligne des ordonnées : (1 , n) ⇒ y(j)
z est le vecteur ligne des cotes : (1 , n) ⇒ z(k)

Généralement une courbe gauche est donnée par des équations paramétriques x(t), y(t), z(t).
La méthode est la même que celle utilisée pour plot2d. On écrit :

$t = linspace(t_{min} , t_{max} , n_t)$
$x = f_1 (t)$
$y = f_2 (t)$
$z = f_3 (t)$

La variante *param3d1* offre la possililité de dessiner plusieurs courbes en disposant les coordonnées des points en colonnes, et également d'employer des couleurs grâce à l'option list :

param3d1($[x_1 ; ... ; x_N]'$, $[y_1 ; ... ; y_N]'$, list($[z_1 ; ... ; z_N]'$, $[couleur_1 , ... , couleur_N]$) , options)

Pour utiliser cette expression il est nécessaire de discrétiser les N courbes avec le même nombre n de points.
Sinon il faut effectuer un appel à param3d1 pour chaque courbe.

Pour plus de détails, accéder à l'aide de Scilab : *F1* ⟶ *Graphiques* ⟶ *param3d*

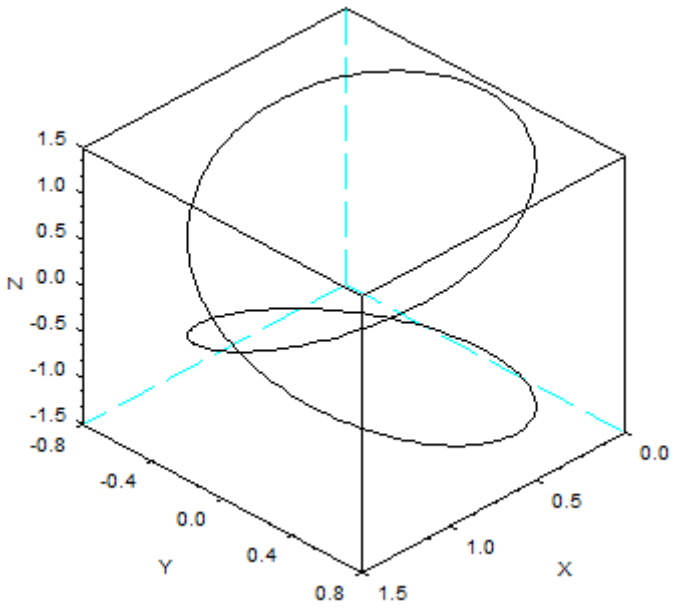
Remarques :

Comme pour plot2d, param3d relie les points éloignés d'une même courbe par des traits. Ainsi, l'instruction param3d($[a_1 , a_2]$, $[b_1 , b_2]$, $[c_1 , c_2]$) trace le segment de droite reliant le point $P_1(a_1, b_1, c_1)$ au point $P_2(a_2, b_2, c_2)$ avec la couleur noire, param3d1($[a_1 , a_2]'$, $[b_1 , b_2]'$, list($[c_1 , c_2]'$, k)) avec la couleur Scilab n° k.

Comme pour plot3d (voir ci-après), il est possible de préciser le point d'observation d'une courbe en attribuant des valeurs aux angles thêta et alpha.

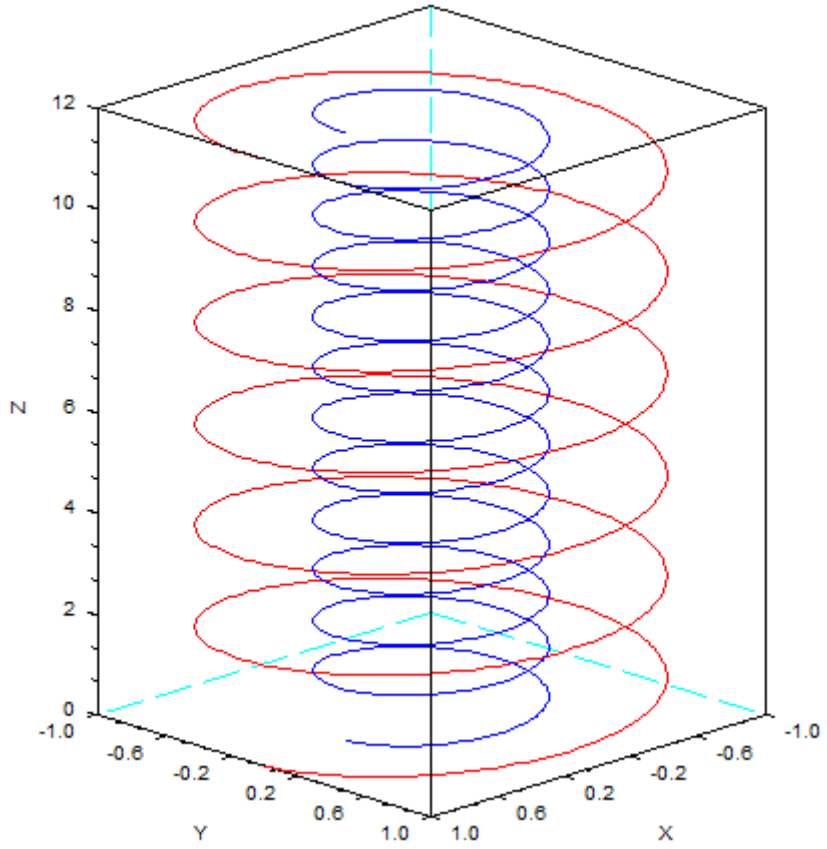
Exemples de programmes Scilab :

```
// Fenêtre de Viviani
r = 1.41 ;
t = linspace(0, 2*%pi, 300) ;
x = r*cos(t)^2 ;
y = r*cos(t).*sin(t) ;
z = r*sin(t) ;
param3d(x, y, z)
```

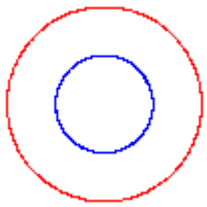


// Hélices

```
r1 = 1 ; p1 = 2 ; s1 = 6 ;
t1 = linspace(0, 2*%pi*s1, 1000) ;
x1 = r1*cos(t1) ;
y1 = r1*sin(t1) ;
z1 = p1*t1/(2*%pi) ;
r2 = 0.5 ; p2 = 1 ; s2 = 12 ;
t2 = linspace(0, 2*%pi*s2, 1000) ;
x2 = r2*cos(t2) ;
y2 = r2*sin(t2) ;
z2 = p2*t2/(2*%pi) ;
param3d1([x1; x2]', [y1; y2]', list([z1; z2]', [5, 2]))
```



En ajoutant les options theta = 0, alpha = 0, flag = [4, 0], on obtient une vue de dessus sans boîte ni graduations :



// Icosaèdre

```
a = 1 ; phi = (1 + sqrt(5))/2 ;

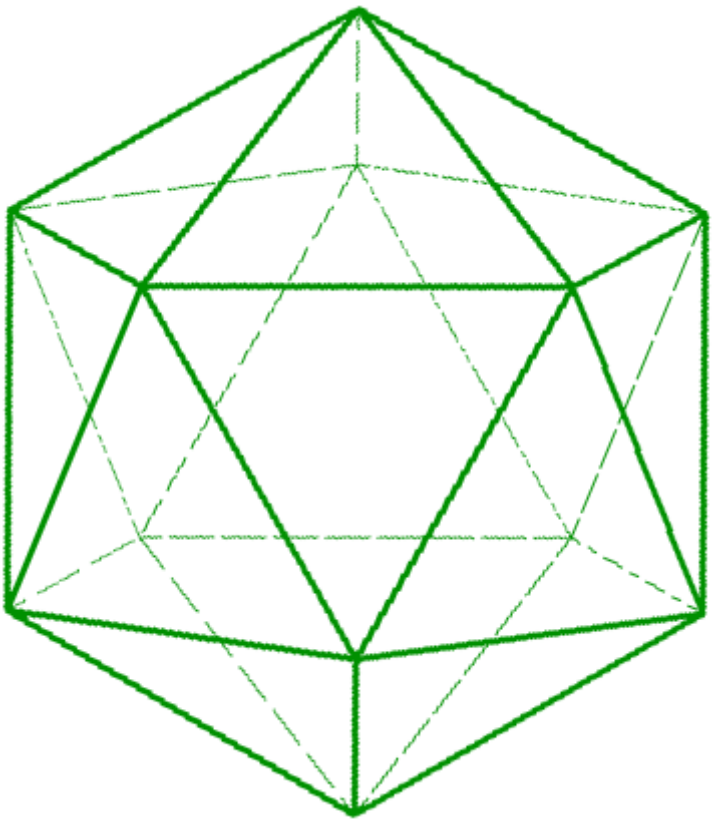
// Coordonnées des 12 Sommets   ==>   S(numéro, :) = [abscisse,ordonnée,cote]
S = [0,-a/2,a/2*phi;a/2,-a/2*phi,0;a/2*phi,0,a/2;0,a/2,a/2*phi;-a/2*phi,0,a/2;-a/2,-a/2*phi,0;
        a/2*phi,0,-a/2;a/2,a/2*phi,0;-a/2,a/2*phi,0;-a/2*phi,0,-a/2;0,-a/2,-a/2*phi;0,a/2,-a/2*phi] ;

// Coordonnées des 30 Arêtes   ==>   A(numéro, :) = [sommet1,sommet2]
A = [1,2;1,3;1,4;1,5;1,6;2,3;3,4;4,5;5,6;2,6;2,7;3,7;3,8;4,8;4,9; 5,9;5,10;
        6,10;6,11;2,11;7,8;8,9;9,10;10,11;7,11;7,12;8,12;9,12;10,12;11,12] ;

// Dessin
X = zeros(2, 30) ; Y = zeros(2, 30) ; Z = zeros(2, 30) ;
for i = 1:30
  arete = [S(A(i, 1), :) ; S(A(i, 2), :)] ;
  X(:, i) = arete(:, 1) ; Y(:, i) = arete(:, 2) ; Z(:, i) = arete(:, 3) ;
end

xset("line style", 1) ; xset("color", 13) ;   xset("thickness", 3) ;
continu = [1,2,3,4,6,7,8,11,12,13,14,15,16,21,22,26,27,28] ;
param3d1(X(:, continu), Y(:, continu), Z(:, continu), theta = 45, alpha = 55, flag = [6, 0]) ;

xset("line style", 4) ; xset("color", 13) ;   xset("thickness", 1) ;
pointille = [5,9,10,17,18,19,20,23,24,25,29,30] ;
param3d1(X(:, pointille), Y(:, pointille), Z(:, pointille), theta = 45, alpha = 55, flag = [6, 0]) ;
```



Cercle circonscrit

Soient 3 points non alignés : A(x_A,y_A,z_A), B(x_B,y_B,z_B), C(x_C,y_C,z_C).
On désire tracer le cercle passant par ces 3 points.

Désignons par a, b, c les côtés BC, AC, AB du triangle (A,B,C) :

$$a = \sqrt{(x_B - x_C)^2 + (y_B - y_C)^2 + (z_B - z_C)^2}$$
$$b = \sqrt{(x_A - x_C)^2 + (y_A - y_C)^2 + (z_A - z_C)^2}$$
$$c = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2}$$

Le rayon du cercle circonscrit au triangle (A,B,C) est :

$$R = \frac{a\,b\,c}{4\sqrt{p\,(p-a)(p-b)(p-c)}} \quad \text{avec} \quad p = \frac{a+b+c}{2}$$

Les coordonnées du centre du cercle circonscrit au triangle (A,B,C) sont :

$$x_0 = \frac{r\,x_A + s\,x_B + t\,x_C}{r+s+t}$$
$$y_0 = \frac{r\,y_A + s\,y_B + t\,y_C}{r+s+t} \qquad \text{avec} \quad \begin{cases} r = a^2(b^2 + c^2 - a^2) \\ s = b^2(a^2 + c^2 - b^2) \\ t = c^2(a^2 + b^2 - c^2) \end{cases}$$
$$z_0 = \frac{r\,z_A + s\,z_B + t\,z_C}{r+s+t}$$

L'équation cartésienne du plan passant par les 3 points A, B, C est :

$$u\,x + v\,y + w\,z = u\,x_A + v\,y_A + w\,z_A$$

$$\text{avec} \quad \begin{cases} u = b_1 c_2 - b_2 c_1 \\ v = a_2 c_1 - a_1 c_2 \\ w = a_1 b_2 - a_2 b_1 \end{cases} , \quad \begin{cases} a_1 = x_B - x_A \\ b_1 = y_B - y_A \\ c_1 = z_B - z_A \end{cases} , \quad \begin{cases} a_2 = x_C - x_A \\ b_2 = y_C - y_A \\ c_2 = z_C - z_A \end{cases}$$

Les coordonnées du triangle (A,B,C) sont relatives à un repère initial $(O, \vec{i}, \vec{j}, \vec{k})$ orthonormé.
Construisons un nouveau repère $(O', \vec{i}', \vec{j}', \vec{k}')$ tel que :

- O' est le centre du cercle circonscrit au triangle (A,B,C) : O'(x_0 , y_0 , z_0).
- $\vec{k}'$ est le vecteur unitaire colinéaire au vecteur $\vec{N}(u,v,w)$ normal au plan défini par les 3 points A, B, C.
  Les coordonnées de $\vec{k}' = \frac{\vec{N}}{\|\vec{N}\|}$ sont :

$$x_3 = \frac{u}{d_3}$$
$$y_3 = \frac{v}{d_3} \qquad \text{avec} \quad d_3 = \sqrt{u^2 + v^2 + w^2}$$
$$z_3 = \frac{w}{d_3}$$

- $\vec{i}'$ est le vecteur unitaire colinéaire au vecteur $\overrightarrow{O'A}$.

  Les coordonnées de $\vec{i}' = \dfrac{\overrightarrow{O'A}}{\left\| \overrightarrow{O'A} \right\|}$ sont :

$$x_1 = \frac{x_A - x_0}{d_1}$$
$$y_1 = \frac{y_A - y_0}{d_1} \qquad \text{avec} \qquad d_1 = \sqrt{(x_A - x_0)^2 + (y_A - y_0)^2 + (z_A - z_0)^2}$$
$$z_1 = \frac{z_A - z_0}{d_1}$$

- $\vec{j}'$ est le vecteur unitaire permettant d'obtenir un repère $\left(O', \vec{i}', \vec{j}', \vec{k}'\right)$ orthonormé direct.

  Les coordonnées de $\vec{j}' = \vec{k}' \wedge \vec{i}'$ sont :

$$x_2 = y_3 z_1 - y_1 z_3$$
$$y_2 = x_1 z_3 - x_3 z_1$$
$$z_2 = x_3 y_1 - x_1 y_3$$

Dans le repère $\left(O', \vec{i}', \vec{j}', \vec{k}'\right)$ le cercle circonscrit au triangle (A,B,C) est défini par les équations cartésiennes :

$$\begin{cases} x'^2 + y'^2 = R^2 \\ z' = 0 \end{cases}$$

En introduisant l'angle polaire θ :

$$\begin{cases} x' = R\cos\theta \\ y' = R\sin\theta \qquad \text{avec} \qquad \theta \in [0, 2\pi] \\ z' = 0 \end{cases}$$

Soient (*x'*,*y'*,*z'*) les coordonnées d'un point quelconque dans le repère $\left(O', \vec{i}', \vec{j}', \vec{k}'\right)$.
Les coordonnées (*x*,*y*,*z*) de ce même point dans le repère $\left(O, \vec{i}, \vec{j}, \vec{k}\right)$ sont données par :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$$

La matrice M des cosinus directeurs des nouveaux axes est orthogonale. On a $M^{-1} = {}^tM$, d'où :

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix}$$

Finalement, les équations paramétriques du cercle circonscrit au triangle (A,B,C) s'écrivent :

$$\begin{cases} x = x_0 + x_1 R\cos\theta + x_2 R\sin\theta \\ y = y_0 + y_1 R\cos\theta + y_2 R\sin\theta \\ z = z_0 + z_1 R\cos\theta + z_2 R\sin\theta \end{cases}$$

Programmation

Notons A, B, C les vecteurs de coordonnées de 3 points de l'espace. La fonction *cercle* ci-dessous retourne les vecteurs de coordonnées x, y, z de n points $P_i(x(i),y(i),z(i))_{1 \le i \le n}$ régulièrement espacés sur le cercle circonscrit au triangle (A, B, C).

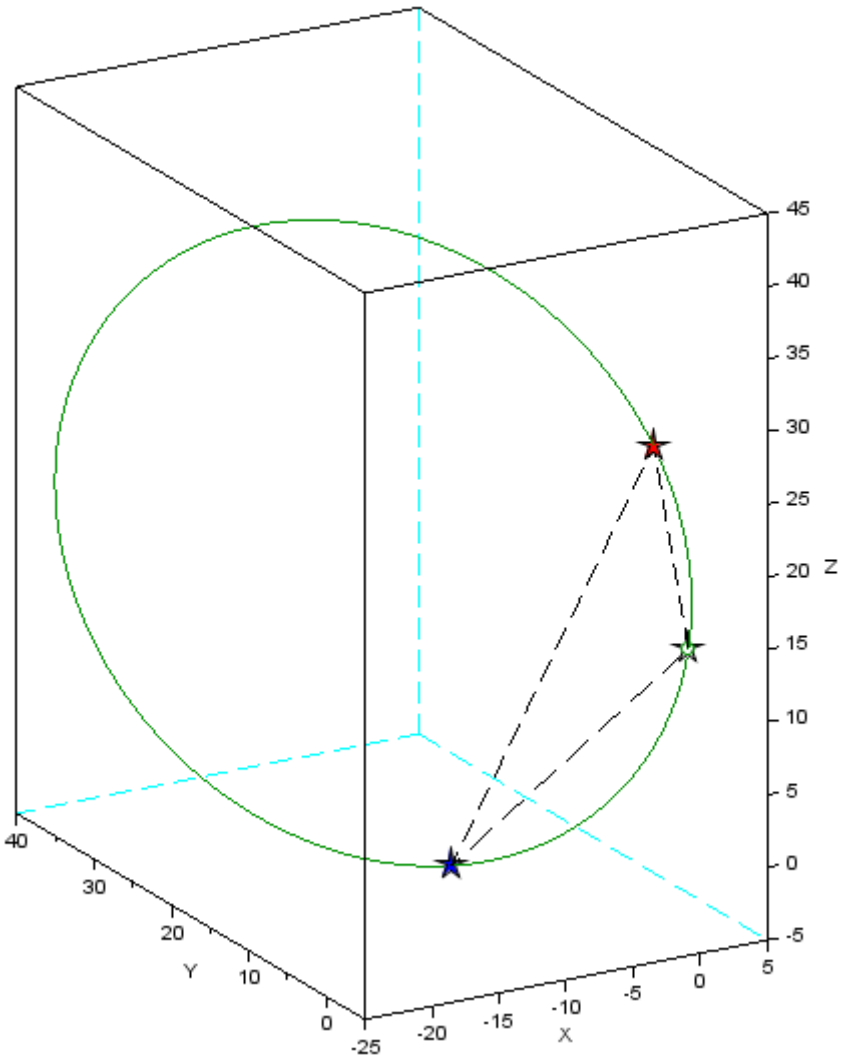
```
function [x,y,z] = cercle(A,B,C,n)
    // Coordonnées des 3 sommets du triangle
    xA = A(1) ; yA = A(2) ; zA = A(3)
    xB = B(1) ; yB = B(2) ; zB = B(3)
    xC = C(1) ; yC = C(2) ; zC = C(3)
    // Longueurs des côtés du triangle
    a = sqrt((xB - xC)^2 + (yB - yC)^2 + (zB - zC)^2)
    b = sqrt((xA - xC)^2 + (yA - yC)^2 + (zA - zC)^2)
    c = sqrt((xA - xB)^2 + (yA - yB)^2 + (zA - zB)^2)
    // Demi-périmètre et surface du triangle
    p = (a + b + c)/2
    S = sqrt(p*(p - a)*(p - b)*(p - c))
    if S == 0 then
        x = [] ; y = [] ; z = []
        return
    end
    // Rayon du cercle circonscrit
    R = (a*b*c)/(4*S)
    // Coordonnées du centre du cercle circonscrit
    r = a^2*(b^2 + c^2 - a^2)
    s = b^2*(a^2 + c^2 - b^2)
    t = c^2*(a^2 + b^2 - c^2)
    x0 = (r*xA + s*xB + t*xC)/(r + s + t)
    y0 = (r*yA + s*yB + t*yC)/(r + s + t)
    z0 = (r*zA + s*zB + t*zC)/(r + s + t)
    // Cosinus directeurs des nouveaux axes
    d1 = sqrt((xA - x0)^2 + (yA - y0)^2 + (zA - z0)^2)
    x1 = (xA - x0)/d1
    y1 = (yA - y0)/d1
    z1 = (zA - z0)/d1
    a1 = xB - xA
    b1 = yB - yA
    c1 = zB - zA
    a2 = xC - xA
    b2 = yC - yA
    c2 = zC - zA
    u = b1*c2 - b2*c1
    v = a2*c1 - a1*c2
    w = a1*b2 - a2*b1
    d3 = sqrt(u^2 + v^2 + w^2)
    x3 = u/d3
    y3 = v/d3
    z3 = w/d3
    x2 = y3*z1 - y1*z3
    y2 = x1*z3 - x3*z1
    z2 = x3*y1 - x1*y3
    // Coordonnées des points du cercle
    theta = linspace(0,2*%pi,n)
    xp = R*cos(theta)
    yp = R*sin(theta)
    x = x0 + x1*xp + x2*yp
    y = y0 + y1*xp + y2*yp
    z = z0 + z1*xp + z2*yp
endfunction
```

Mathématique & Informatique

Exemple

On prend : A(2,0,14), B(0,1,28), C(-7,15,-4).

```
A = [2,0,14] ; B = [0,1,28] ; C = [-7,15,-4] ;
// Fenêtre graphique
f = scf() ; f.figure_size = [650,800] ;
// Tracé du triangle
param3d([A(1),B(1),C(1),A(1)],[A(2),B(2),C(2),A(2)],[A(3),B(3),C(3),A(3)])
e = gce() ; e.line_style = 2 ;
// Sommets en forme d'étoile
param3d(A(1),A(2),A(3))
e = gce() ; e.mark_style = 14 ; e.mark_size = 4 ; e.mark_foreground = 1 ; e.mark_background = 8 ;
param3d(B(1),B(2),B(3))
e = gce() ; e.mark_style = 14 ; e.mark_size = 4 ; e.mark_foreground = 1 ; e.mark_background = 5 ;
param3d(C(1),C(2),C(3))
e = gce() ; e.mark_style = 14 ; e.mark_size = 4 ; e.mark_foreground = 1 ; e.mark_background = 2 ;
// Tracé du cercle circonscrit
[x,y,z] = cercle(A,B,C,300) ;
param3d(x,y,z)
e = gce() ; e.foreground = 13 ;
// Angle de vue
a = gca() ; a.isoview = 'on' ; a.rotation_angles = [430,-120] ;
```



Mathématique & Informatique

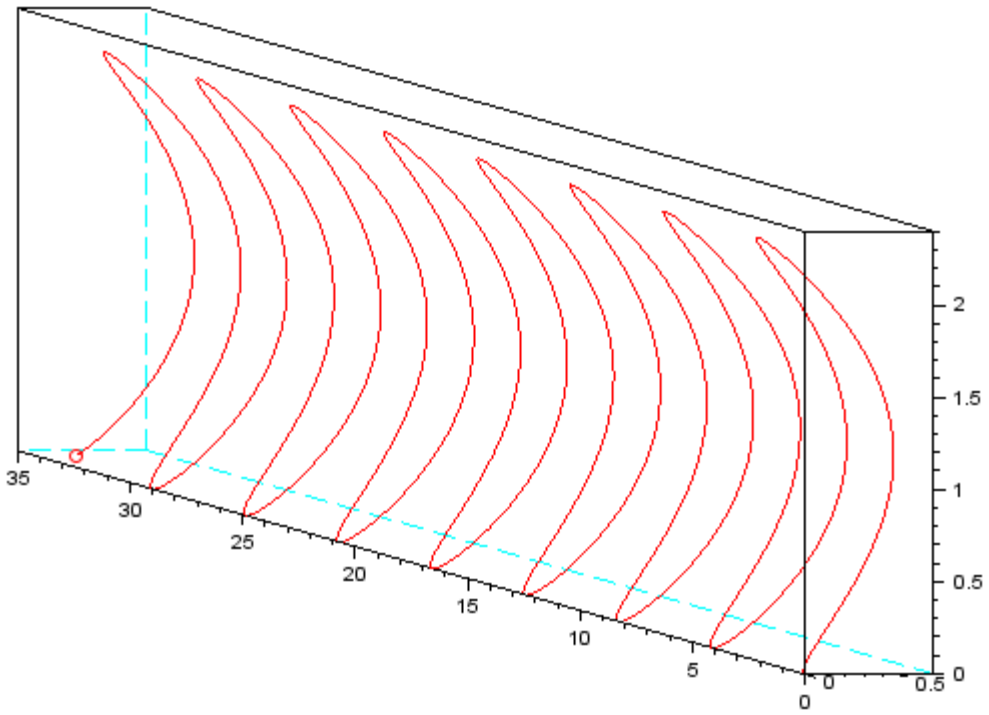
La fonction comet3d

Effectuons une représentation dynamique de la trajectoire définie par les équations horaires :

$$\begin{cases} x = \displaystyle\int_0^t f(u)\sin u\,du \\ y = \displaystyle\int_0^t f(u)\cos u\,du \\ z = \displaystyle\int_0^t f(u)\tan u\,du \end{cases} \qquad \text{où} \quad f(u) = \cos u\sqrt{1+\cos^2 u}$$

Après avoir discrétisé la variable *t* avec la fonction *linspace*, puis calculé les coordonnées *x*, *y*, *z* correspondantes avec la fonction *integrate*, on utilise la fonction *comet3d* pour afficher un mobile en forme de petit cercle parcourant la courbe.

```
t = linspace(0,50,1000) ;
x = integrate('cos(u)*sqrt(1 + cos(u)^2)*sin(u)','u',0,t) ;
y = integrate('cos(u)*sqrt(1 + cos(u)^2)*cos(u)','u',0,t) ;
z = integrate('cos(u)*sqrt(1 + cos(u)^2)*tan(u)','u',0,t) ;
a = gca() ; a.isoview = 'on' ; a.rotation_angles = [88,263] ;
comet3d(x,y,z,"colors",5)
```



Notons $\vec{v}$ le vecteur vitesse du mobile à l'instant *t*, le rayon de courbure *R* de la trajectoire est donné par :

$$R = \frac{\|\vec{v}\|^3}{\|\vec{v}\wedge\vec{v}'\|} = \frac{\left(x'^2+y'^2+z'^2\right)^{3/2}}{\sqrt{(y'z''-z'y'')^2+(z'x''-x'z'')^2+(x'y''-y'x'')^2}} \quad \text{avec} \quad \begin{cases} x' = f(t)\sin t \\ y' = f(t)\cos t \\ z' = f(t)\tan t \end{cases} \quad \text{et} \quad \begin{cases} x'' = f'(t)\sin t + f(t)\cos t \\ y'' = f'(t)\cos t - f(t)\sin t \\ z'' = f'(t)\tan t + f(t)(1+\tan^2 t) \end{cases}$$

Après simplifications, on trouve : *R* = 1 (rayon de courbure constant).

Mathématique & Informatique

Graphisme en 3 dimensions :

On utilise la fonction ***plot3d*** pour représenter une surface dans l'espace :

plot3d(x, y, z, options)
- x est le vecteur ligne des abscisses : $(1 , n_x) \Rightarrow x(i)$
- y est le vecteur ligne des ordonnées : $(1 , n_y) \Rightarrow y(j)$
- z est la matrice des cotes : $(n_x , n_y) \Rightarrow z(i , j) = f(x(i) , y(j))$

Suivant le cas, on prend la variante *fplot3d*, *fplot3d1*, *plot3d* ou *plot3d1* qui convient le mieux.

- **Cas 1 : la surface est donnée par une équation cartésienne z(x,y)**

  1) Définir les domaines de variation de l'abscisse x et de l'ordonnée y :
     $x = \text{linspace}(x_{min} , x_{max} , n_x)$
     $y = \text{linspace}(y_{min} , y_{max} , n_y)$
  2) Ecrire l'expression analytique de la cote z :
     $z = f(x , y)$
  3) Dessiner la surface : fplot3d(x, y, f, options)

- **Cas 2 : la surface est donnée par des équations paramétriques x(u,v), y(u,v), z(u,v)**

  1) Définir les domaines de variation des paramètres u et v :
     $u = \text{linspace}(u_{min} , u_{max} , n_u)$
     $v = \text{linspace}(v_{min} , v_{max} , n_v)$
  2) Ecrire les expressions analytiques de l'abscisse x, de l'ordonnée y et de la cote z des points :
     $[x, y, z] = f(u, v) = [f_1(u , v), f_2(u , v), f_3(u , v)]$
  3) Calculer les matrices de coordonnées xf, yf, zf des $N = (n_u - 1) \times (n_v - 1)$ facettes à 4 sommets :
     [xf, yf, zf] = eval3dp(f, u, v) où **eval3dp** est la fonction Scilab de calcul des facettes
  4) Dessiner la surface : plot3d(xf, yf, zf, options)

- **Cas 3 : la surface est donnée par un ensemble de points x, y, z**

  1) Définir les matrices de coordonnées des points x, y, z :
     x = vecteur (1,m) ; y = vecteur (1,n) ; z = matrice (m,n) telle que z(i,j) = cote de (x(i),y(j))
  2) Calculer les matrices de coordonnées xf, yf, zf des $N = (m - 1) \times (n - 1)$ facettes à 4 sommets :
     [xf, yf, zf] = genfac3d(x, y, z) où **genfac3d** est la fonction Scilab de calcul des facettes
  3) Dessiner la surface : plot3d(xf, yf, zf, options)

- **Cas 4 : la surface est donnée par un ensemble de faces polygonales**

  1) Définir la matrice S des n sommets et la matrice F des N faces :
     $S = [s_1; ... ; s_n]$ avec s_i = vecteur ligne des coordonnées du sommet i (abscisse, ordonnée, cote)
     $F = [f_1; ... ; f_N]$ avec f_i = vecteur ligne des numéros des sommets de la face i
  2) Calculer les matrices de coordonnées des faces xf, yf, zf :
     $xf = [xf_1, ... , xf_N]$ avec xf_i = vecteur colonne des abscisses des sommets de la face i
     $yf = [yf_1, ... , yf_N]$ avec yf_i = vecteur colonne des ordonnées des sommets de la face i
     $zf = [zf_1, ... , zf_N]$ avec zf_i = vecteur colonne des cotes des sommets de la face i
  3) Dessiner la surface : plot3d(xf, yf, zf, options)

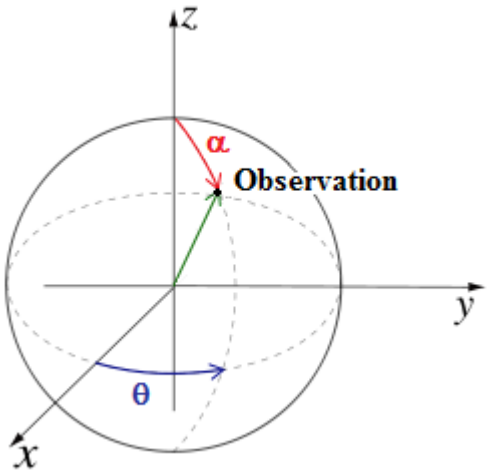
Remarques :

- Le cas 1 se ramène au cas 2 en prenant : $u = x$; $v = y$; $f_1 = Id$; $f_2 = Id$; $f_3 = f$.
- Dans le cas 4, les faces doivent avoir le même nombre de sommets. Sinon, il faut faire appel à plot3d pour chaque ensemble de faces ayant le même nombre de sommets, voire faire un appel à plot3d pour chaque face.
- Les faces (facettes) doivent être orientées positivement pour que leurs couleurs apparaissent. Par convention, il faut parcourir les sommets de chaque face (facette) dans le sens des aiguilles d'une montre (↻ vu de l'extérieur).

Mathématique & Informatique

Les options :

- L'option principale de la fonction plot3d est le point d'observation de la surface, défini par 2 angles de rotation en degrés : *theta* et *alpha* correspondant respectivement à la longitude et à la colatitude en coordonnées sphériques.
  L'appel de la fonction se fait par :
  plot3d(x, y, z, theta = θ, alpha = α)
  avec θ ∈ [0, 360°] et α ∈ [0, 180°]
  Par défaut : theta ≈ alpha ≈ 45°.



- L'option *flag* offre différentes possibilités de dessin :

  plot3d(x, y, z, flag = [n_1, n_2 , n_3])

  n_1 est un entier représentant la couleur de la surface
  n_2 est un entier représentant l'échelle utilisée
  n_3 est un entier représentant la boîte d'encadrement

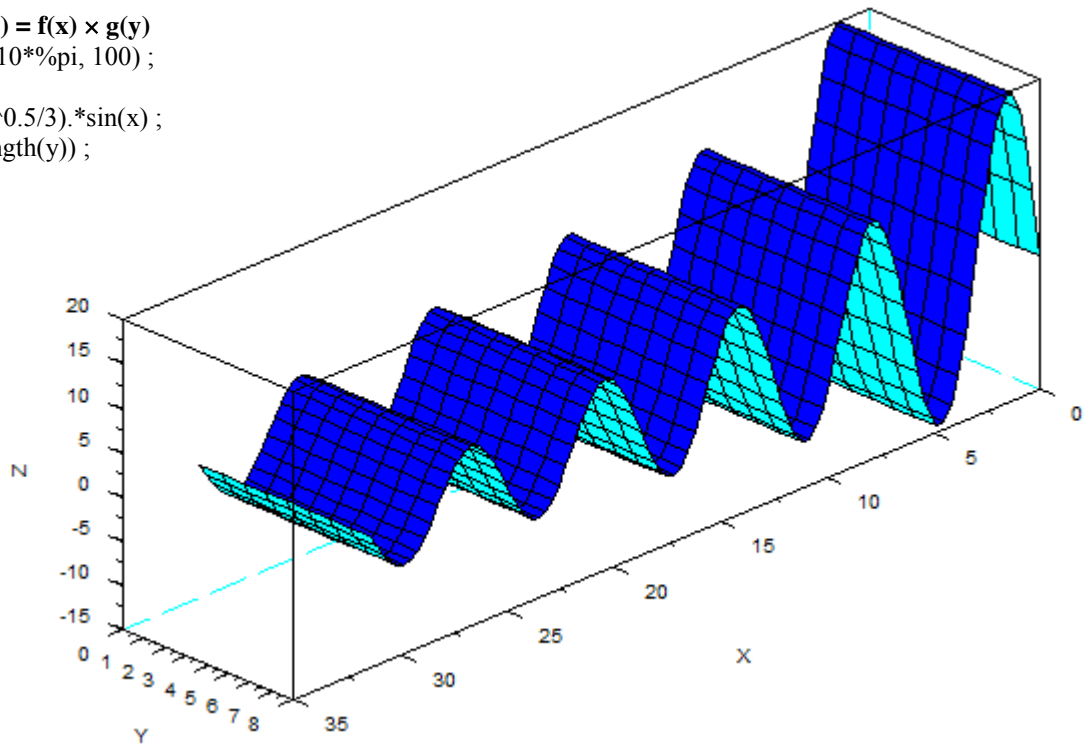
- L'option *list* permet d'attribuer une couleur à chaque facette :

  plot3d(xf, yf, list(zf, [couleur$_1$, ... , couleur$_N$])) où couleur$_i$ est un entier représentant la couleur de la facette i

Pour plus de détails, accéder à l'aide de Scilab : *F1* ⟶ *Graphiques* ⟶ *plot3d*

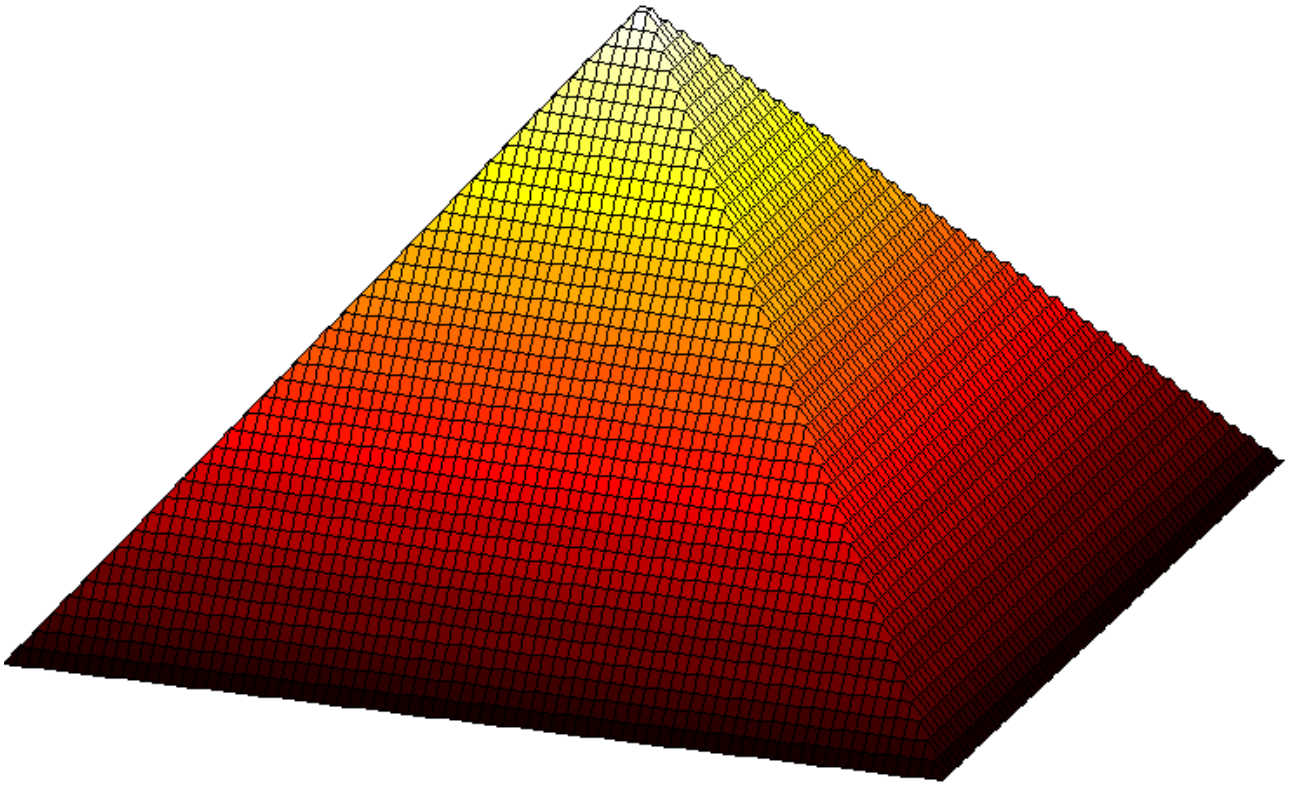
Exemples de programmes Scilab :

```
// Surface z(x,y) = f(x) × g(y)
x = linspace(0, 10*%pi, 100) ;
y = 0:8 ;
fx = 30*exp(-x^0.5/3).*sin(x) ;
gy = ones(1, length(y)) ;
z = fx' * gy ;
plot3d(x, y, z)
```





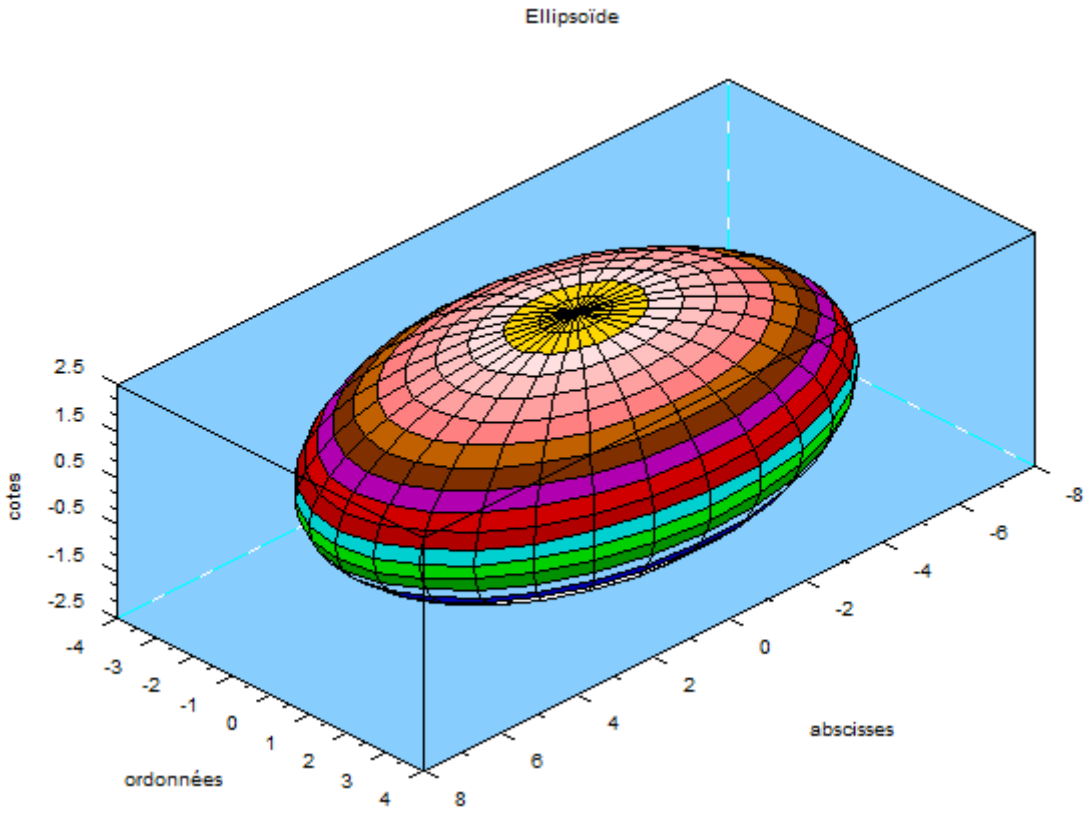
```
// Pyramide
// Equation
function z = pyramide(x, y)
z = h*(1 - max(abs([x/a, y/b])))
endfunction
// Discrétisation et dessin
a = 230/2 ; b = 230/2 ; h = 137 ;
xset("colormap", hotcolormap(32))
x = linspace(-a, +a, 80) ; y = linspace(-b, +b, 80) ;
fplot3d1(x, y, pyramide, theta = 20, alpha = 70, flag = [7, 6, 0])
```



Mathématique & Informatique

// Ellipsoïde

```
// Equations
function   [x, y, z] = ellipsoide(u, v)
   a = 6.28 ; b = 3.88 ; c = 2.40 ;
   x = a*sin(v).*cos(u) ;
   y = b*sin(v).*sin(u) ;
   z = c*cos(v) ;
endfunction
// Discrétisation et dessin
u = linspace(0, 2*%pi, 30) ;
v = linspace(0, %pi, 25) ;
[xf,   yf,   zf] = eval3dp(ellipsoide, u, v) ;
plot3d1(xf, yf, zf, theta = 45, alpha = 60, flag = [2, 6, 4]) ;
xtitle("Ellipsoïde", "abscisses", "ordonnées", "cotes") ;
xset("background", 12) ;
```



Remarques :

- La fonction eval3dp permet d'écrire les équations naturellement en remplaçant les * et / par des .* et ./
- La fonction plot3d1 est identique à plot3d avec en plus des couleurs différentes selon la valeur de z (si les couleurs n'apparaissent pas, il faut inverser les bornes de variation d'un des paramètres)
- L'option *flag* = [2, 6, 4] sert à afficher le maillage, utiliser une échelle isométrique, et graduer les axes.
- L'instruction *xset*("background", 12) remplit la boîte d'encadrement avec la couleur bleu clair.

Visualisation de l'ellipsoïde précédent dans un mouvement de rotation perpendiculaire à chacun des 3 axes x, y et z

```
// Equations de l'ellipsoïde
function [x, y, z]=ellipsoide(u, v)
a = 6.28 , b = 3.88 , c = 2.40
x = a*sin(v).*cos(u)
y = b*sin(v).*sin(u)
z = c*cos(v)
endfunction

// Discrétisation et calcul des facettes
u = linspace(0, 2*%pi, 30) ;
v = linspace(0, %pi, 25) ;
[xf,   yf,   zf] = eval3dp(ellipsoide, u, v) ;

// Fenêtre graphique
xset("wdim", 600, 400)

// Paramètres de la fonction plot3d1
theta = 0:10:360 ;
alpha = 0:10:360 ;
flag = [2, 1, 0] ;
ebox = [-6.28,+6.28,-3.88,+3.88,-2.4,+2.4] ;

// Vue en rotation autour de l'axe des x
for i = 1:length(alpha)
plot3d1(xf, yf, zf, theta = 90, alpha = alpha(i), flag = flag, ebox = ebox)
xtitle("ELLIPSOIDE :   thêta = 90 degrés , alpha = " + string(alpha(i)) + " degrés")
xpause(1e5)
end

// Vue en rotation autour de l'axe des y
for i = 1:length(alpha)
plot3d1(xf, yf, zf, theta = 0, alpha = alpha(i), flag = flag, ebox = ebox)
xtitle("ELLIPSOIDE :   thêta = 0 degrés , alpha = " + string(alpha(i)) + " degrés")
xpause(1e5)
end

// Vue en rotation autour de l'axe des z
for i = 1:length(theta)
plot3d1(xf, yf, zf, theta = theta(i), alpha = 90, flag = flag, ebox = ebox)
xtitle("ELLIPSOIDE :   alpha = 90 degrés , thêta = " + string(theta(i)) + " degrés")
xpause(1e5)
end
```

Rotation manuelle : faire tourner l'image dans l'espace en déplaçant la souris avec le bouton droit maintenu enfoncé

Pour connaître les angles de visualisation (colatitude alpha et longitude theta en degrés) d'une image fixe, cliquer sur :
Edition ⟶ *Propriétés des axes* ⟶ *Viewpoint*

Mathématique & Informatique

```
// Paraboloïde hyperbolique
// Equation
function z = paraboloide(x, y)
   a = 2 ; b = 0.4 ; z = x^2/a - y^2/b
endfunction

// Dessin
drawlater()

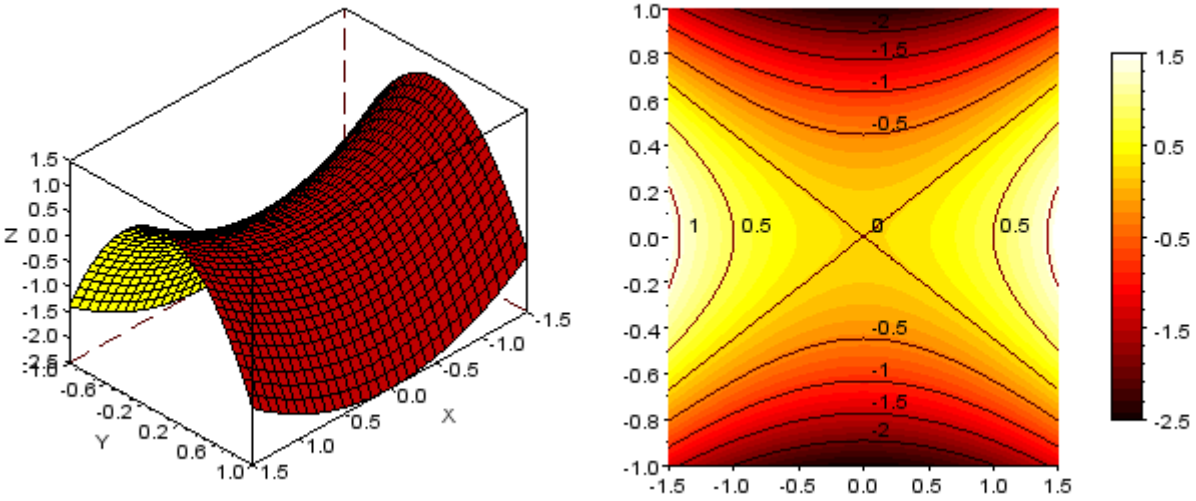
xset("colormap", hotcolormap(32)) ;

x = linspace(-1.5, +1.5, 31) ;
y = linspace(-1, +1, 41) ;

subplot(1, 2, 1) ; xset("hidden3d", 24) ;
fplot3d(x, y, paraboloide, flag = [9, 2, 4]) ;

subplot(1, 2, 2) ; colorbar(-2.5, +1.5) ;
Sfgrayplot(x, y, paraboloide, frameflag = 2) ;
contour2d(x, y, paraboloide, [-2:0.5:+1], frameflag = 2) ;

drawnow()
```



Remarques :

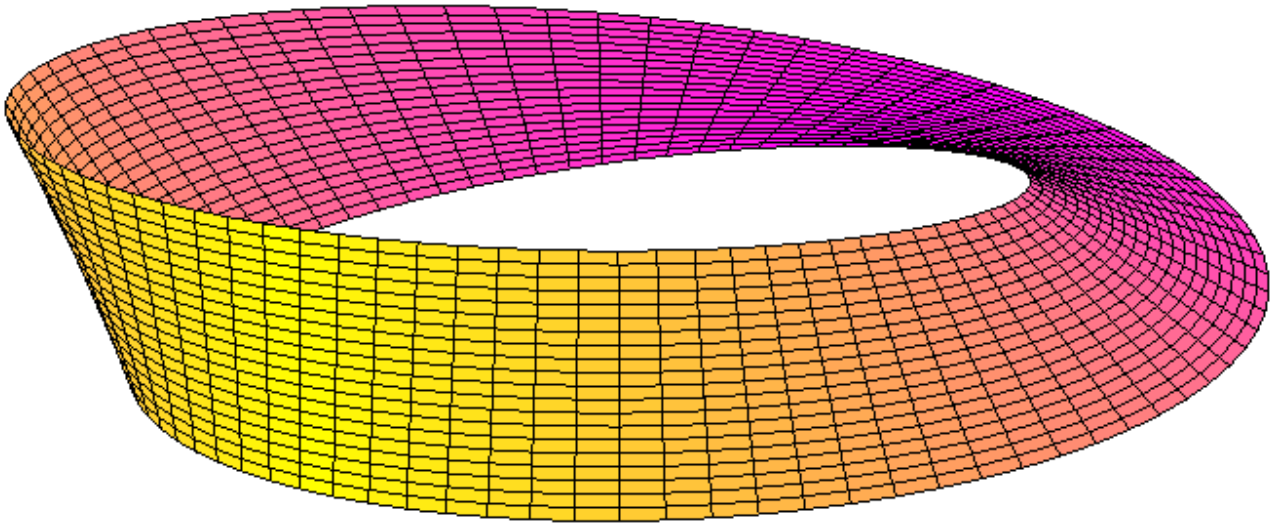
- Pour le 1er dessin : *fplot3d* effectue une représentation 3D de la surface définie par z = f(x,y), en utilisant la couleur n_1 (flag = [n_1, ...]) pour le dessus et la couleur n_2 pour le dessous (xset("hidden3d", n_2)).
  Les numéros de couleur sont pris dans la carte courante (Ici : "hotcolormap(32)").

- Pour le 2e dessin : *Sfgrayplot* effectue une représentation 2D de la surface définie par z = f(x,y), en utilisant des couleurs lissées en fonction de la cote z et de l'échelle fixée par l'instruction *colorbar* (colorbar(z_{min}, z_{max})).
  contour2d ajoute quelques courbes de niveau avec leurs valeurs (Ici : 0 ⇒ droites ; ≠ 0 ⇒ hyperboles).

Mathématique & Informatique

// Ruban de Möbius

```
function [x,y,z] = Mobius(u,v)
   x = (r + u.*cos(v/2)).*cos(v)
   y = (r + u.*cos(v/2)).*sin(v)
   z = u.*sin(v/2)
endfunction
r = 4 ; h = 1 ;
u = linspace(-h, +h, 21) ;
v = linspace(0, 2*%pi, 81) ;
[xf,yf,zf] = eval3dp(Mobius,u,v) ;
plot3d(xf, yf, list(zf,ones(1,1600)), alpha = -106, theta = -35, flag = [1,6,0])
f = gcf() ; f.color_map = springcolormap(800) ;
e = gce() ; e.hiddencolor = -1 ; e.color_flag = 3 ; e.data.color = [1:1:800,800:-1:1] ;
```



Remarque :

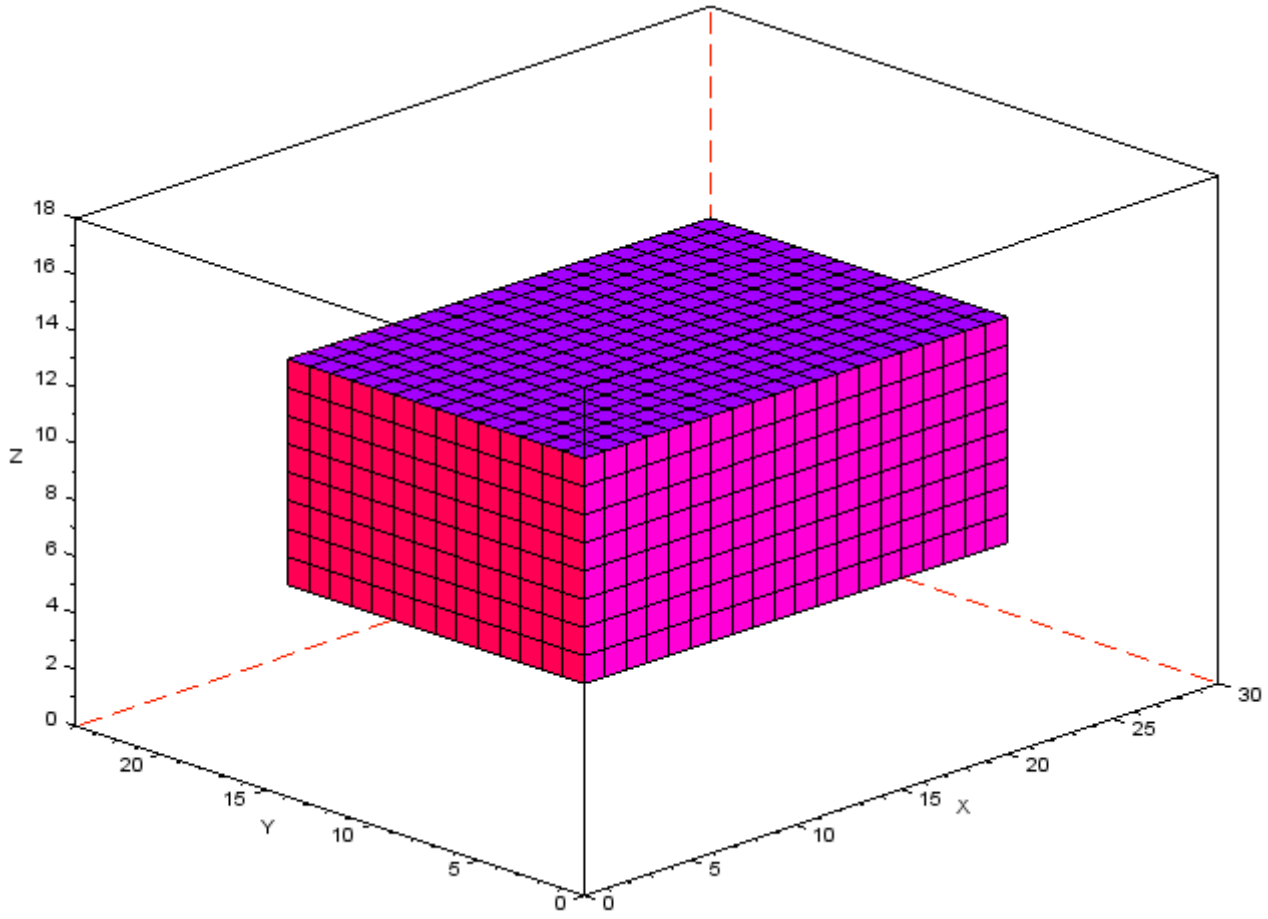
Le ruban de **Möbius** est un exemple classique de surface qui ne possède qu'une seule face (surface non orientable).

$$\begin{cases} x = \left(r + u\cos\dfrac{v}{2}\right)\cos v \\ y = \left(r + u\cos\dfrac{v}{2}\right)\sin v \\ z = u\sin\dfrac{v}{2} \end{cases} \qquad -h \le u \le +h \ , \ 0 \le v \le 2\pi$$

L'instruction *e.hiddencolor = -1* attribue la même couleur à la face "cachée" et à la face "visible" de la surface.

```
// Parallélépipède rectangle
function parallelepipede(p1,p2,p3,p4)
    function [X,Y,Z] = surface1(v,w)
        X = x*ones(1,j*k)
        Y = v
        Z = w
    endfunction
    function [X,Y,Z] = surface2(u,w)
        X = u
        Y = y*ones(1,i*k)
        Z = w
    endfunction
    function [X,Y,Z] = surface3(u,v)
        X = u
        Y = v
        Z = z*ones(1,i*j)
    endfunction
    x0 = p1(1) ; y0 = p1(2) ; z0 = p1(3)
    a = p2(1) ; b = p2(2) ; c = p2(3)
    i = p3(1) ; j = p3(2) ; k = p3(3)
    n1 = p4(1) ; n2 = p4(2) ; n3 = p4(3)
    n4 = p4(4) ; n5 = p4(5) ; n6 = p4(6)
    x = x0
    v = linspace(y0,y0 + b,j) ; w = linspace(z0,z0 + c,k)
    [xf,yf,zf] = eval3dp(surface1,v,w)
    plot3d(xf,yf,zf,flag = [n1,6,4])
    e = gce() ; e.hiddencolor = n1
    x = x0 + a
    v = linspace(y0,y0 + b,j) ; w = linspace(z0 + c,z0,k)
    [xf,yf,zf] = eval3dp(surface1,v,w)
    plot3d(xf,yf,zf,flag = [n2,6,0])
    e = gce() ; e.hiddencolor = n2
    y = y0
    u = linspace(x0,x0 + a,i) ; w = linspace(z0 + c,z0,k)
    [xf,yf,zf] = eval3dp(surface2,u,w)
    plot3d(xf,yf,zf,flag = [n3,6,4])
    e = gce() ; e.hiddencolor = n3
    y = y0 + b
    u = linspace(x0,x0 + a,i) ; w = linspace(z0,z0 + c,k)
    [xf,yf,zf] = eval3dp(surface2,u,w)
    plot3d(xf,yf,zf,flag = [n4,6,4])
    e = gce() ; e.hiddencolor = n4
    z = z0
    u = linspace(x0,x0 + a,i) ; v = linspace(y0,y0 + b,j)
    [xf,yf,zf] = eval3dp(surface3,u,v)
    plot3d(xf,yf,zf,flag = [n5,6,4])
    e = gce() ; e.hiddencolor = n5
    z = z0 + c
    u = linspace(x0,x0 + a,i) ; v = linspace(y0 + b,y0,j)
    [xf,yf,zf] = eval3dp(surface3,u,v)
    plot3d(xf,yf,zf,flag = [n6,6,4])
    e = gce() ; e.hiddencolor = n6
endfunction
```

```
f = scf() ;
f.figure_name = 'Parallélépipède' ; f.color_map = hsvcolormap(128) ;
X = getsystemmetrics('SM_CXSCREEN') ; Y = getsystemmetrics('SM_CYSCREEN') ;
f.figure_position = [0,0] ; f.figure_size = [X/2,3*Y/4] ;
p1 = [5,5,5] ; p2 = [20,14,8] ; p3 = [21,15,9] ; p4 = [122,122,111,111,100,100] ;
parallelepipede(p1,p2,p3,p4) ;
a = gca() ;
xmin = 0 ; xmax = 30 ; ymin = 0 ; ymax = 24 ; zmin = 0 ; zmax = 18 ;
a.data_bounds = [xmin,ymin,zmin;xmax,ymax,zmax] ;
a.isoview = 'on' ;
a.rotation_angles = [70.5,225] ;
```



La fonction *parallelepipede*(p1,p2,p3,p4) permet de dessiner un parallélépipède rectangle avec les paramètres suivants :

- p1 = [x0, y0, z0] donne respectivement l'abscisse, l'ordonnée et la cote du sommet de coordonnées minimales
- p2 = [a, b, c] donne les dimensions du parallélépipède selon l'axe des *x*, l'axe des *y*, l'axe des *z*
- p3 = [i, j, k] donne les nombres de points de discrétisation selon l'axe des *x*, l'axe des *y*, l'axe des *z*
- p4 = [n1, n2, n3, n4, n5, n6] donne les numéros de couleur des 6 faces

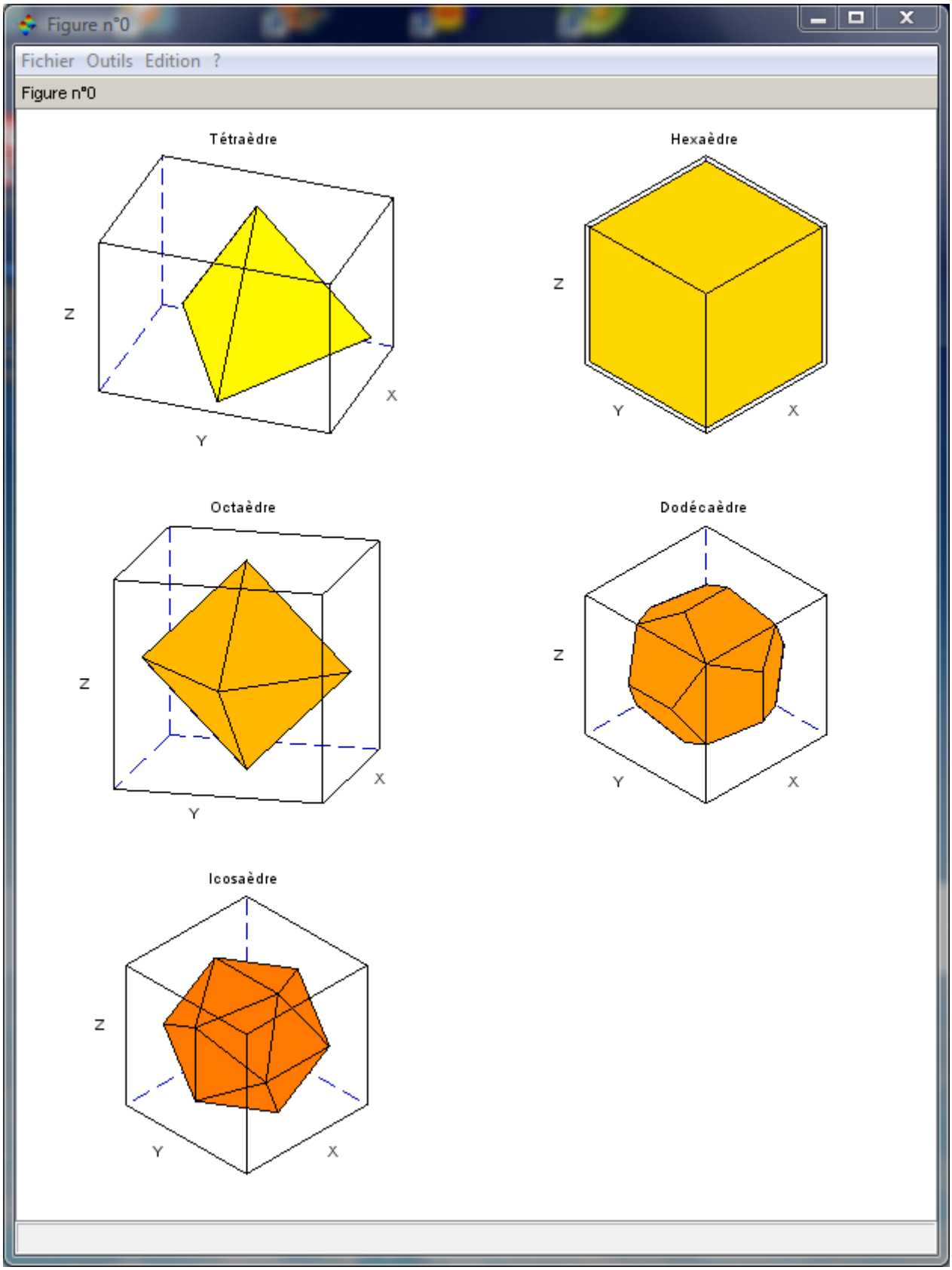
Remarque :

Les numéros de couleur sont dans l'ordre des plans d'équations x = x0, x = x0 + a, y = y0, y = y0 + b, z = z0, z = z0 + c.
Si l'on prend i = j = k = 2, les faces du parallélépipède rectangle ne sont pas subdivisées, seules les arêtes apparaissent.

```
// Polyèdres réguliers
function polyedre(n)
  // Coordonnées
  R = 1 ; phi = (1 + sqrt(5))/2 ;
  select n
    case 1
      nf = 4 ; na = 6 ; ns = 4 ; nsf = 3 ; a = 2*R/sqrt(3) ; param = [41,20,60] ;
      S = R/3*[0,0,3;2*sqrt(2),0,-1;-sqrt(2),sqrt(6),-1;-sqrt(2),-sqrt(6),-1] ;
      F = [1,3,2;1,4,3;1,2,4;2,3,4] ;
    case 2
      nf = 6 ; na = 12 ; ns = 8 ; nsf = 4 ; a = 2*R/sqrt(3) ; param = [43,45,55] ;
      S = R/sqrt(3)*[1,1,1;-1,1,1;-1,-1,1;1,-1,1;1,1,-1;-1,1,-1;-1,-1,-1;1,-1,-1] ;
      F = [1,4,3,2;1,2,6,5;2,3,7,6;3,4,8,7;4,1,5,8;5,6,7,8] ;
    case 3
      nf = 8 ; na = 12 ; ns = 6 ; nsf = 3 ; a = sqrt(2)*R ; param = [45,15,75] ;
      S = R*[1,0,0;0,1,0;-1,0,0;0,-1,0;0,0,1;0,0,-1] ;
      F = [5,2,1;5,3,2;5,4,3;5,1,4;6,1,2;6,2,3;6,3,4;6,4,1] ;
    case 4
      nf = 12 ; na = 30 ; ns = 20 ; nsf = 5 ; a = 2*R/(sqrt(3)*phi) ; param = [47,45,55] ;
      S = R/sqrt(3)*[1,1,1;1,-1,1;-1,1,1;-1,-1,1;1,1,-1;1,-1,-1;-1,1,-1;-1,-1,-1;
                     phi-1,0,phi;1-phi,0,phi;1-phi,0,-phi;phi-1,0,-phi;
                     0,phi,phi-1;0,phi,1-phi;0,-phi,1-phi;0,-phi,phi-1;
                     phi,phi-1,0;phi,1-phi,0;-phi,1-phi,0;-phi,phi-1,0] ;
      F = [1,17,18,2,9;1,13,14,5,17;1,9,10,3,13;9,2,16,4,10;17,5,12,6,18;13,3,20,7,14;
           8,15,6,12,11;8,11,7,20,19;8,19,4,16,15;15,16,2,18,6;11,12,5,14,7;19,20,3,10,4] ;
    case 5
      nf = 20 ; na = 30 ; ns = 12 ; nsf = 3 ; a = 2*R/sqrt(phi+2) ; param = [49,45,55] ;
      S = R/sqrt(phi+2)*[phi,0,1;-phi,0,1;-phi,0,-1;phi,0,-1;0,1,phi;0,1,-phi;
                         0,-1,-phi;0,-1,phi;1,phi,0;1,-phi,0;-1,-phi,0;-1,phi,0] ;
      F = [8,5,1;8,2,5;8,11,2;8,10,11;8,1,10;10,1,4;1,9,4;1,5,9;5,12,9;5,2,12;
           2,3,12;2,11,3;11,7,3;11,10,7;10,4,7;6,4,9;6,9,12;6,12,3;6,3,7;6,7,4] ;
  end
  // Dessin
  xf = zeros(nsf, nf) ; yf = zeros(nsf, nf) ; zf = zeros(nsf, nf) ; sf = zeros(nsf, 3) ;
  for i = 1:nf
    for j = 1:nsf
      sf(j, :) = S(F(i,j), :) ;
    end
    xf(:, i) = sf(:, 1) ; yf(:, i) = sf(:, 2) ; zf(:, i) = sf(:, 3) ;
  end
  plot3d(xf, yf, list(zf, param(1)*ones(1,nf)), theta = param(2), alpha = param(3), flag = [2,4,3]) ;
endfunction

xset("colormap", jetcolormap(64)) ;
Nom = ["Tétraèdre", "Hexaèdre", "Octaèdre", "Dodécaèdre", "Icosaèdre"] ;
for n = 1:5
  subplot(3, 2, n) ; xtitle(Nom(n)) ; polyedre(n) ;
end
```

Mathématique & Informatique



Mathématique & Informatique

Représentation spatiale du chlorure de sodium.

Le chlorure de sodium *NaCl* est un sel qui cristallise dans le système cubique : les ions Na^+ occupent les milieux des arêtes et le centre du cube, les ions Cl^- occupent les sommets et les centres des faces du cube.
Les mesures par diffraction de rayons *X* sur des cristaux de *NaCl* permettent de déterminer les rayons des ions Na^+ et Cl^- :

$$r_{Na^+} = 102\,\text{pm} \quad ; \quad r_{Cl^-} = 181\,\text{pm}$$

⇨ **http://www.elementschimiques.fr/?fr/proprietes/chimiques/rayon-ionique**

Graphiquement, les ions Na^+ et Cl^- seront représentés par des sphères tangentes de rayons respectifs r_{Na^+} et r_{Cl^-}.
On utilisera la couleur rouge pour le sodium et la couleur verte pour le chlore.

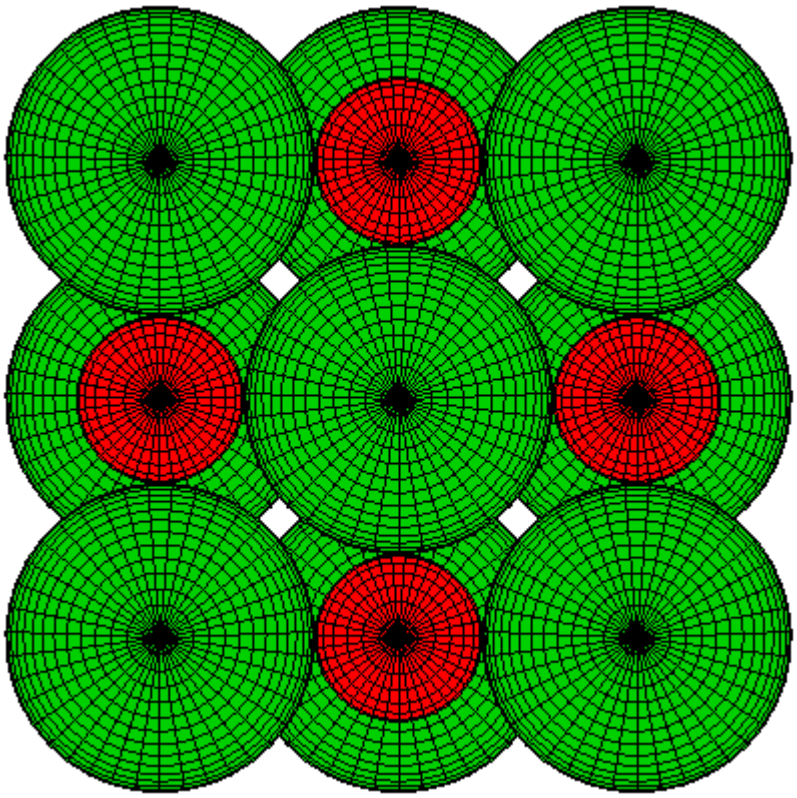
Le programme de dessin est le suivant :

```
function ion(centre, rayon, couleur)
    function [x, y, z]=sphere(u, v)
        x = centre(1) + rayon*sin(v).*cos(u)
        y = centre(2) + rayon*sin(v).*sin(u)
        z = centre(3) + rayon*cos(v)
    endfunction
    select rayon
    case rNa
        m = 40 ; n = 15
    case rCl
        m = 40 ; n = 30
    end
    u = linspace(0,2*%pi,m)
    v = linspace(0,%pi,n)
    [xf,yf,zf] = eval3dp(sphere,u,v)
    plot3d(xf,yf,zf,alpha = alpha,theta = theta,flag = [couleur,6,0])
endfunction

// Angles de vue 3d
alpha = 79 ;
theta = 121 ;
// Couleurs des ions
rouge = 5 ;
vert = 15 ;
// Rayons ioniques
rNa = 102 ;
rCl = 181 ;
// Arête du cube
a = 2*(rNa + rCl) ;
// Sommets du cube
S1 = [0,0,0] ;
S2 = [a,0,0] ;
S3 = [0,a,0] ;
S4 = [0,0,a] ;
S5 = [a,a,0] ;
S6 = [a,0,a] ;
S7 = [0,a,a] ;
S8 = [a,a,a] ;
```

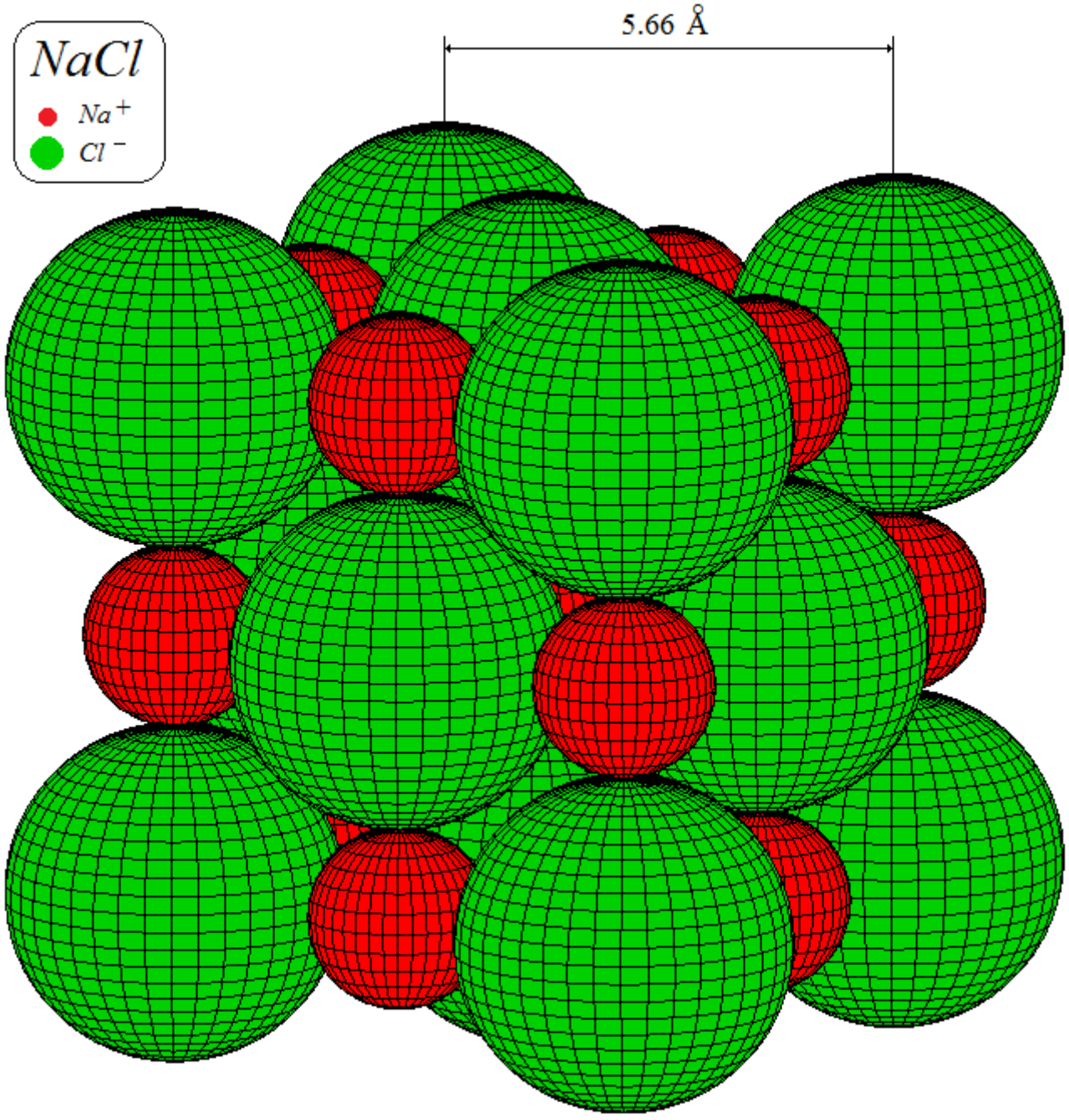
```
// Ions Na+
ion((S1 + S2)/2,rNa,rouge)
ion((S2 + S5)/2,rNa,rouge)
ion((S5 + S3)/2,rNa,rouge)
ion((S3 + S1)/2,rNa,rouge)
ion((S4 + S6)/2,rNa,rouge)
ion((S6 + S8)/2,rNa,rouge)
ion((S8 + S7)/2,rNa,rouge)
ion((S7 + S4)/2,rNa,rouge)
ion((S2 + S6)/2,rNa,rouge)
ion((S5 + S8)/2,rNa,rouge)
ion((S3 + S7)/2,rNa,rouge)
ion((S1 + S4)/2,rNa,rouge)
ion((S1 + S8)/2,rNa,rouge)
// Ions Cl-
ion(S1,rCl,vert)
ion(S2,rCl,vert)
ion(S3,rCl,vert)
ion(S4,rCl,vert)
ion(S5,rCl,vert)
ion(S6,rCl,vert)
ion(S7,rCl,vert)
ion(S8,rCl,vert)
ion((S1 + S5)/2,rCl,vert)
ion((S4 + S8)/2,rCl,vert)
ion((S2 + S8)/2,rCl,vert)
ion((S5 + S7)/2,rCl,vert)
ion((S3 + S4)/2,rCl,vert)
ion((S1 + S6)/2,rCl,vert)
```

Angles de vue 3d : $\alpha = 0°$ et $\theta = 0°$



Mathématique & Informatique

Angles de vue 3d : $\alpha = 79°$ et $\theta = 121°$



Ce motif, appelé maille, peut être reproduit à l'infini dans toutes les directions de l'espace.
Notons a l'arête de la maille : $a = 2\left(r_{Na^+} + r_{Cl^-}\right) = 566 \text{ pm} = 5.66 \text{ Å}$.

Cette valeur s'obtient expérimentalement par radiocristallographie en appliquant la loi de **Bragg** :

$$a = \frac{\lambda}{\sin\theta} \qquad \begin{cases} \lambda \text{ est la longueur d'onde des rayons } X \text{ réfléchis sur l'ensemble des plans réticulaires espacés de } d = a/2 \\ \theta \text{ est l'angle d'incidence minimal du faisceau de rayons } X \text{ donnant une intensité de réflexion maximale} \end{cases}$$

Par exemple, pour $\lambda = 1$ Å, on trouve $\theta = 10°\ 10'$.

On peut également calculer a d'après les caractéristiques physico-chimiques du chlorure de sodium :

$$a = \sqrt[3]{\frac{4M}{N_A \rho}} \qquad \begin{cases} M = 58.44 \text{ g est la masse molaire de } NaCl \\ \rho = 2.17 \text{ g/cm}^3 \text{ est la masse volumique de } NaCl \\ N_A = 6.022\ 10^{23} \text{ est le nombre d'}\textbf{Avogadro} \end{cases}$$

Estimons le nombre N de mailles contenues dans 1 cm^3 de cristal de chlorure de sodium : $N = (10^{-2} \times 10^{12})^3/566^3 = 5.51\ 10^{21}$
Donc, il y a plus de 5000 milliards de milliards de cellules ioniques identiques à celle représentée ci-dessus dans 1 cm^3 de $NaCl$!

Mathématique & Informatique

Représentation spatiale du diamant

Le diamant est un cristal de carbone pur, sa maille est un cube constitué de 18 atomes de carbone :

- 8 occupent les sommets
- 6 occupent les centres des faces
- 4 occupent les centres de tétraèdres interstitiels

Au sujet de la maille cubique à faces centrées et du choix des tétraèdres interstitiels (4 parmi 8 possibles), cliquer sur le lien
⇨ **http://fr.wikipedia.org/wiki/Cubique_%C3%A0_faces_centr%C3%A9es**

La maille cubique a pour arête a = 357 pm, les atomes de carbone ont pour rayon r_C = 77 pm. On a la relation : $r_C = a\sqrt{3}/8$.
Sur le dessin, les atomes de carbone seront représentés par des sphères de rayon $r_1 = r_C/2$ ($r_1 < r_C$ afin de mieux visualiser la structure de l'ensemble avec une échelle adaptée au picomètre). Trois couleurs différentes permettront de distinguer les trois lieux géométriques où se situent les sphères : couleur rouge pour les sphères qui se situent aux sommets du cube, couleur verte pour les sphères qui se situent aux centres des faces du cube, couleur bleue pour les sphères qui se situent aux centres des tétraèdres interstitiels. On ajoutera des liaisons entre les sphères des centres et des sommets des tétraèdres et entre les sphères des sommets du cube. Une liaison sera représentée par un cylindre de rayon $r_2 = r_C/20$, de couleur jaune s'il s'agit d'une liaison covalente entre un atome de carbone et l'un de ses 4 plus proches voisins (hybridation tétragonale sp^3 du carbone conduisant à 4 liaisons σ), de couleur blanche s'il s'agit d'une arête purement géométrique servant à délimiter le volume de la maille du cristal.

```
function atome(centre, rayon, couleur)
    function [x, y, z]=sphere(u, v)
        x = centre(1) + rayon*sin(v).*cos(u)
        y = centre(2) + rayon*sin(v).*sin(u)
        z = centre(3) + rayon*cos(v)
    endfunction
    m = 30 ; n = 20
    u = linspace(0,2*%pi,m)
    v = linspace(0,%pi,n)
    [xf,yf,zf] = eval3dp(sphere,u,v)
    plot3d(xf,yf,zf,flag = [couleur,6,0])
endfunction

function liaison(centre1, centre2, rayon, couleur)
    function [x, y, z]=cylindre(u, v)
        X = rayon*cos(v)
        Y = rayon*sin(v)
        Z = u
        x = A(1) + I(1)*X + J(1)*Y + K(1)*Z
        y = A(2) + I(2)*X + J(2)*Y + K(2)*Z
        z = A(3) + I(3)*X + J(3)*Y + K(3)*Z
    endfunction
    A = centre1
    B = centre2
    K = (B - A)/norm(B - A)
    if K(1) <> 0 then
        U = [K(2),- K(1),0]
    else
        U = [0,K(3),- K(2)]
    end
    I = U/norm(U)
    J = [I(3)*K(2) - I(2)*K(3),I(1)*K(3) - I(3)*K(1),I(2)*K(1) - I(1)*K(2)]
    h = norm(B - A)
    m = 50 ; n = 10
    u = linspace(0,h,m)
    v = linspace(0,2*%pi,n)
    [xf,yf,zf] = eval3dp(cylindre,u,v)
    plot3d(xf,yf,zf,flag = [couleur,6,0])
endfunction
```

```
// Fenêtre graphique
f1 = scf(1) ;

// Couleurs
couleur1 = color(255,0,0) ;
couleur2 = color(0,255,0) ;
couleur3 = color(0,0,255) ;
couleur4 = color(255,255,0) ;
couleur5 = color(255,255,255) ;

// Arête et rayons
a = 357 ;
rC = 77 ;
r1 = rC/2 ;
r2 = rC/20 ;

// Sommets
S1 = [0,0,0] ;
S2 = [a,0,0] ;
S3 = [0,a,0] ;
S4 = [0,0,a] ;
S5 = [a,a,0] ;
S6 = [a,0,a] ;
S7 = [0,a,a] ;
S8 = [a,a,a] ;

// Centres des faces
F1 = (S1 + S5)/2 ;
F2 = (S4 + S8)/2 ;
F3 = (S2 + S8)/2 ;
F4 = (S5 + S7)/2 ;
F5 = (S3 + S4)/2 ;
F6 = (S1 + S6)/2 ;

// Centres des tétraèdres
T1 = (S1 + F1 + F5 + F6)/4 ;
T2 = (S5 + F1 + F3 + F4)/4 ;
T3 = (S6 + F2 + F3 + F6)/4 ;
T4 = (S7 + F2 + F4 + F5)/4 ;

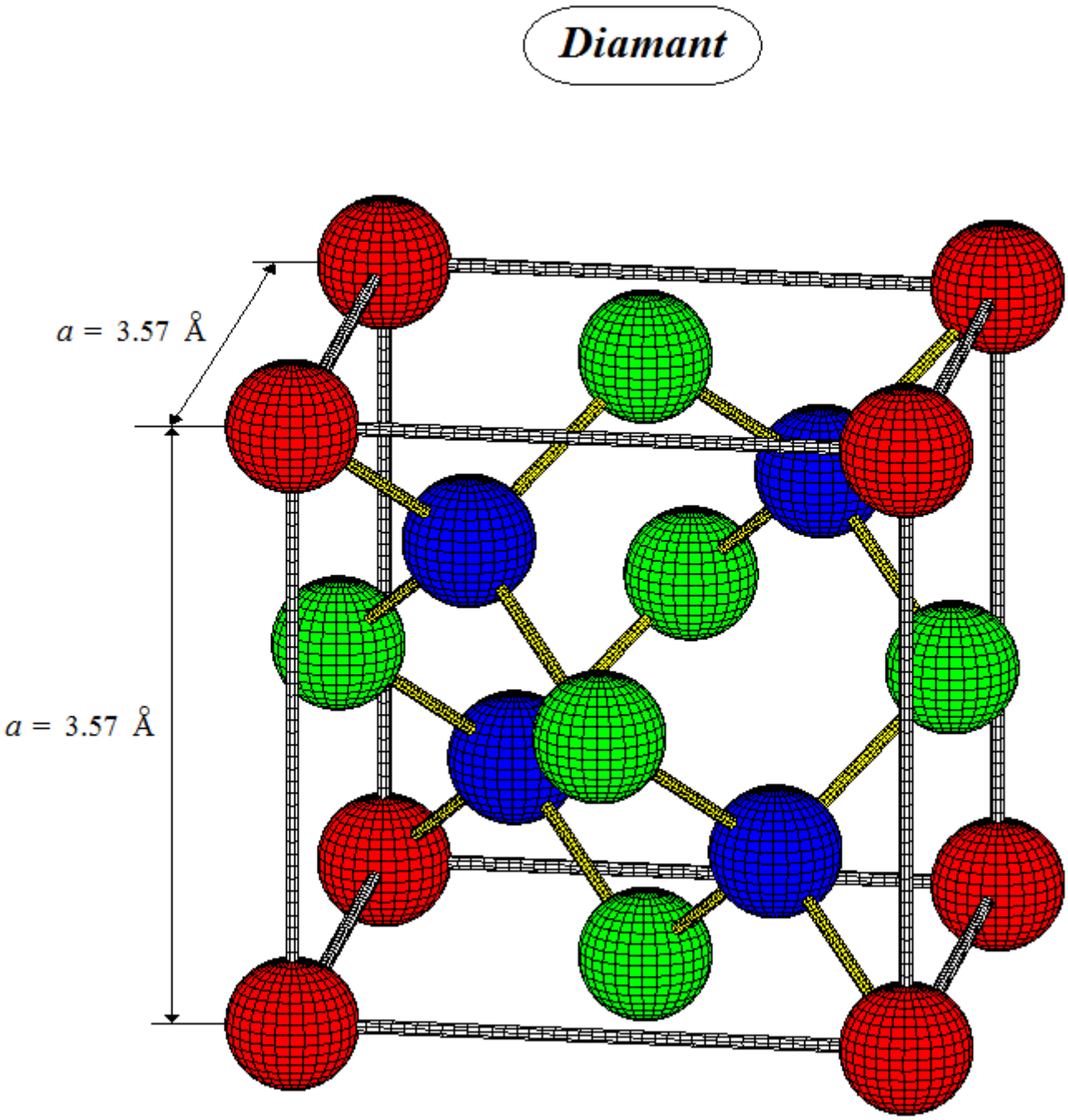
// Atomes
atome(S1,r1,couleur1) ;
atome(S2,r1,couleur1) ;
atome(S3,r1,couleur1) ;
atome(S4,r1,couleur1) ;
atome(S5,r1,couleur1) ;
atome(S6,r1,couleur1) ;
atome(S7,r1,couleur1) ;
atome(S8,r1,couleur1) ;
atome(F1,r1,couleur2) ;
atome(F2,r1,couleur2) ;
atome(F3,r1,couleur2) ;
atome(F4,r1,couleur2) ;
atome(F5,r1,couleur2) ;
atome(F6,r1,couleur2) ;
atome(T1,r1,couleur3) ;
atome(T2,r1,couleur3) ;
atome(T3,r1,couleur3) ;
atome(T4,r1,couleur3) ;
```

```
// Liaisons
liaison(T1,S1,r2,couleur4) ;
liaison(T1,F1,r2,couleur4) ;
liaison(T1,F5,r2,couleur4) ;
liaison(T1,F6,r2,couleur4) ;
liaison(T2,S5,r2,couleur4) ;
liaison(T2,F1,r2,couleur4) ;
liaison(T2,F3,r2,couleur4) ;
liaison(T2,F4,r2,couleur4) ;
liaison(T3,S6,r2,couleur4) ;
liaison(T3,F2,r2,couleur4) ;
liaison(T3,F3,r2,couleur4) ;
liaison(T3,F6,r2,couleur4) ;
liaison(T4,S7,r2,couleur4) ;
liaison(T4,F2,r2,couleur4) ;
liaison(T4,F4,r2,couleur4) ;
liaison(T4,F5,r2,couleur4) ;
liaison(S1,S2,r2,couleur5) ;
liaison(S2,S5,r2,couleur5) ;
liaison(S5,S3,r2,couleur5) ;
liaison(S3,S1,r2,couleur5) ;
liaison(S4,S6,r2,couleur5) ;
liaison(S6,S8,r2,couleur5) ;
liaison(S8,S7,r2,couleur5) ;
liaison(S7,S4,r2,couleur5) ;
liaison(S1,S4,r2,couleur5) ;
liaison(S2,S6,r2,couleur5) ;
liaison(S5,S8,r2,couleur5) ;
liaison(S3,S7,r2,couleur5) ;

// Angles de vue 3d
a1 = gca() ;
a1.rotation_angles = [74.5,188.5] ;
```

Mathématique & Informatique

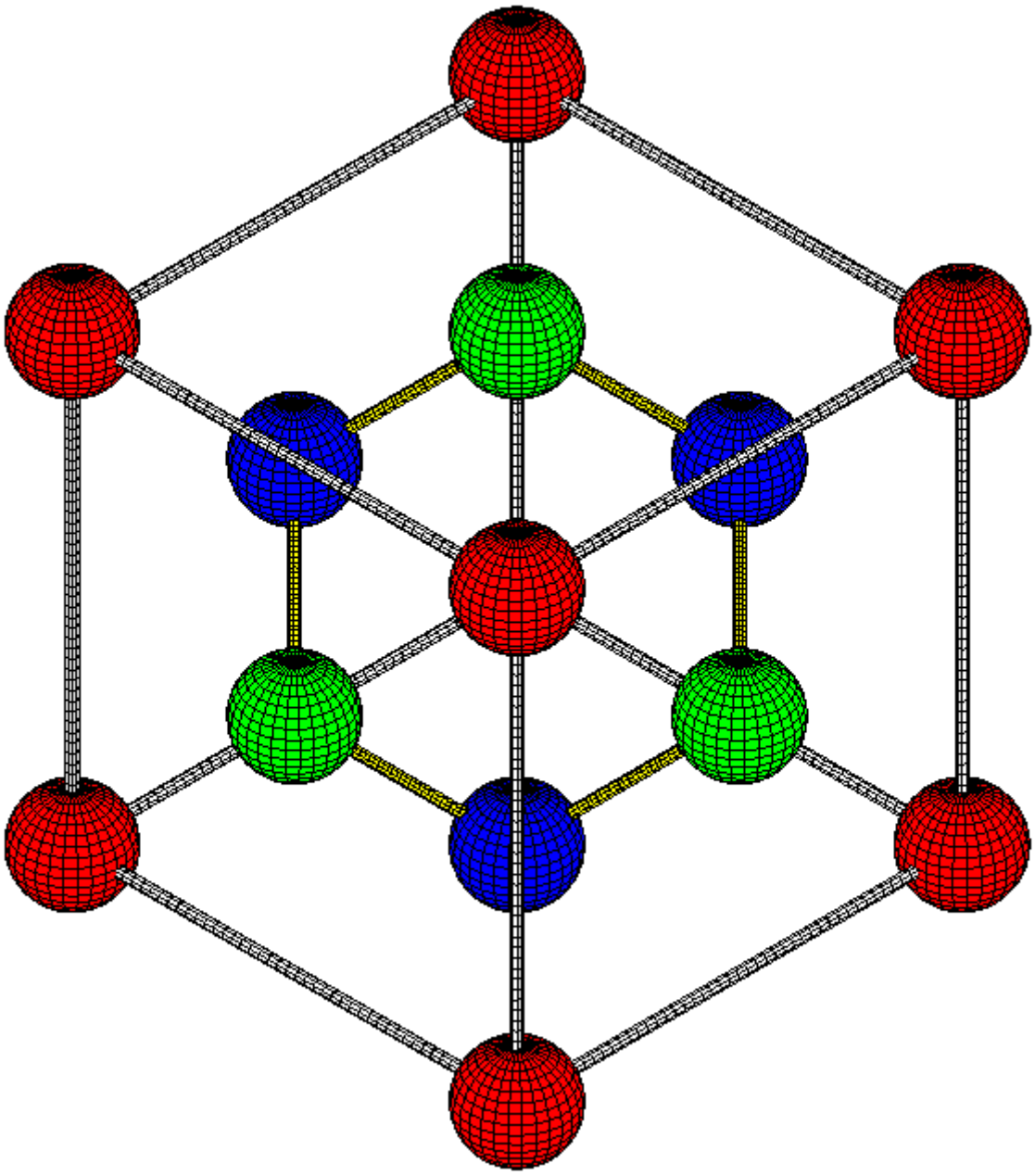
Angles de vue 3d : $\alpha = 74.5°$ et $\theta = 188.5°$



Mathématique & Informatique

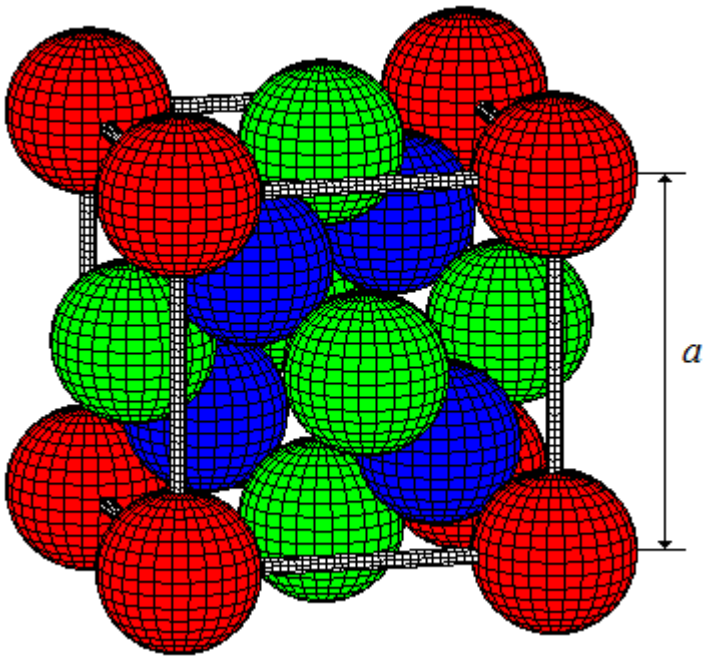
Angles de vue 3d : $\alpha = 54.75°$ et $\theta = 45°$

```
f2 = scf(2) ;
copy(f1) ;
a2 = gca() ;
a2.rotation_angles = [54.75,45] ;
```



Mathématique & Informatique

L'arête de la maille de diamant



La valeur a de l'arête de la maille de diamant peut s'obtenir directement à partir de mesures par diffraction de rayons X.
On peut aussi calculer a à partir des propriétés physico-chimiques du diamant.
Rappelons l'identité générale valable pour un cristal quelconque :

$$M = \frac{mN_A}{n}\rho V \qquad \begin{cases} M = \text{masse molaire du cristal} \\ \rho = \text{masse volumiquedu cristal} \\ V = \text{volumed'une maille} \\ n = \text{nombre d'entités par maille} \\ m = \text{nombre molaire d'entités} \\ N_A = \text{constante d'}\textbf{Avogadro} \end{cases}$$

Le nombre n d'entités (atomes, ions, molécules, …) contenues dans les limites géométriques du volume V de la maille est :

$$n = \sum_i \frac{p_i}{q_i} \qquad \begin{cases} p_i = \text{nombre d'entités appartenan t au même type } i \text{ de lieu géométrique dans la maille} \\ q_i = \text{nombre de mailles adjacentes auxquelles appartient une même entité de type } i \end{cases}$$

Dans le cas du cristal de diamant, la maille est constituée d'atomes appartenant à 3 lieux géométriques différents :

- type 1 (sommet du cube ⇒ sphère rouge) : $p_1 = 8$; $q_1 = 8$
- type 2 (centre d'une face ⇒ sphère verte) : $p_2 = 6$; $q_2 = 2$
- type 3 (centre d'un tétraèdre ⇒ sphère bleue) : $p_3 = 4$; $q_3 = 1$

D'où : $n = \frac{8}{8} + \frac{6}{2} + \frac{4}{1} = 8$

Le volume de la maille est celui d'un cube de côté a : $V = a^3$
La masse molaire du carbone (masse de N_A atomes de carbone) est : $M = 12$ g
Le nombre de moles d'atomes correspondant à la masse molaire M est : $m = 1$
La mesure de la masse volumique du diamant donne : $\rho = 3.517$ g/cm^3
On en déduit :

$$a = \sqrt[3]{\frac{nM}{mN_A\rho}} = \sqrt[3]{\frac{8\times 12\times 10^{-3}}{1\times 6.022\times 10^{23}\times 3.517\times 10^{3}}} = 3.57\ 10^{-10}\,\text{m} = 3.57\ \text{Å}$$

Mathématique & Informatique

Représentation spatiale du graphite

Le graphite est un cristal de carbone, sa maille est un prisme droit à base hexagonale constitué de 16 atomes de carbone. Les dimensions sont les suivantes :

- côté de la base hexagonale : c = 142 pm
- hauteur du prisme : h = 670 pm
- rayon de l'atome : r_C = 71 pm

On a la relation : $r_C = c/2$

Programme de représentation 3d de la maille :

```
// Fenêtre graphique
f1 = scf(1) ;

// Couleurs
couleur1 = color(255,0,0) ;
couleur2 = color(0,255,0) ;
couleur3 = color(0,0,255) ;
couleur4 = color(255,255,0) ;
couleur5 = color(255,255,255) ;

// Dimensions
c = 142 ;
h = 670 ;
rC = 71 ;
r1 = rC/2 ;
r2 = rC/10 ;

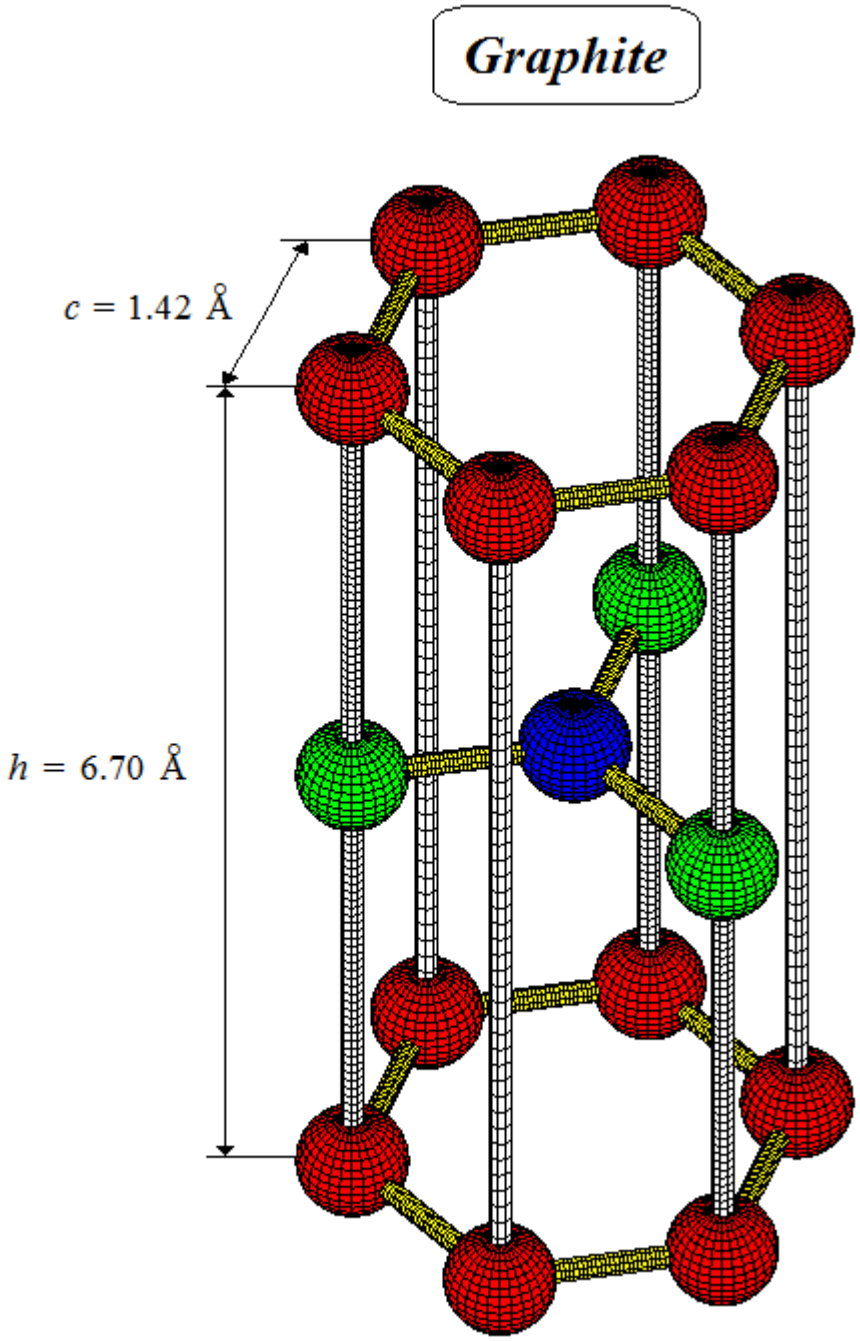
// Points
P01 = [0,0,0] ;
P02 = [c,0,0] ;
P03 = [-c/2,c*sqrt(3)/2,0] ;
P04 = [-c/2,-c*sqrt(3)/2,0] ;
P05 = [c,0,h/2] ;
P06 = [c/2,c*sqrt(3)/2,h/2] ;
P07 = [-c/2,c*sqrt(3)/2,h/2] ;
P08 = [-c,0,h/2] ;
P09 = [-c/2,-c*sqrt(3)/2,h/2] ;
P10 = [c/2,-c*sqrt(3)/2,h/2] ;
P11 = [c,0,-h/2] ;
P12 = [c/2,c*sqrt(3)/2,-h/2] ;
P13 = [-c/2,c*sqrt(3)/2,-h/2] ;
P14 = [-c,0,-h/2] ;
P15 = [-c/2,-c*sqrt(3)/2,-h/2] ;
P16 = [c/2,-c*sqrt(3)/2,-h/2] ;
```

```
// Atomes
atome(P01,r1,couleur3) ;
atome(P02,r1,couleur2) ;
atome(P03,r1,couleur2) ;
atome(P04,r1,couleur2) ;
atome(P05,r1,couleur1) ;
atome(P06,r1,couleur1) ;
atome(P07,r1,couleur1) ;
atome(P08,r1,couleur1) ;
atome(P09,r1,couleur1) ;
atome(P10,r1,couleur1) ;
atome(P11,r1,couleur1) ;
atome(P12,r1,couleur1) ;
atome(P13,r1,couleur1) ;
atome(P14,r1,couleur1) ;
atome(P15,r1,couleur1) ;
atome(P16,r1,couleur1) ;

// Liaisons
liaison(P01,P02,r2,couleur4) ;
liaison(P01,P03,r2,couleur4) ;
liaison(P01,P04,r2,couleur4) ;
liaison(P05,P06,r2,couleur4) ;
liaison(P06,P07,r2,couleur4) ;
liaison(P07,P08,r2,couleur4) ;
liaison(P08,P09,r2,couleur4) ;
liaison(P09,P10,r2,couleur4) ;
liaison(P10,P05,r2,couleur4) ;
liaison(P11,P12,r2,couleur4) ;
liaison(P12,P13,r2,couleur4) ;
liaison(P13,P14,r2,couleur4) ;
liaison(P14,P15,r2,couleur4) ;
liaison(P15,P16,r2,couleur4) ;
liaison(P16,P11,r2,couleur4) ;
liaison(P02,P05,r2,couleur5) ;
liaison(P02,P11,r2,couleur5) ;
liaison(P03,P07,r2,couleur5) ;
liaison(P03,P13,r2,couleur5) ;
liaison(P04,P09,r2,couleur5) ;
liaison(P04,P15,r2,couleur5) ;
liaison(P06,P12,r2,couleur5) ;
liaison(P08,P14,r2,couleur5) ;
liaison(P10,P16,r2,couleur5) ;
```

```
// Angles de vue 3d
a1 = gca() ;
a1.rotation_angles = [46.25,199.25]
```

Mathématique & Informatique



Propriétés

Hybridation sp^2 du carbone conduisant à 3 liaisons σ + 1 liaison π.

Nombre n d'atomes par maille : $n = \frac{12}{6} + \frac{3}{3} + \frac{1}{1} = 4$

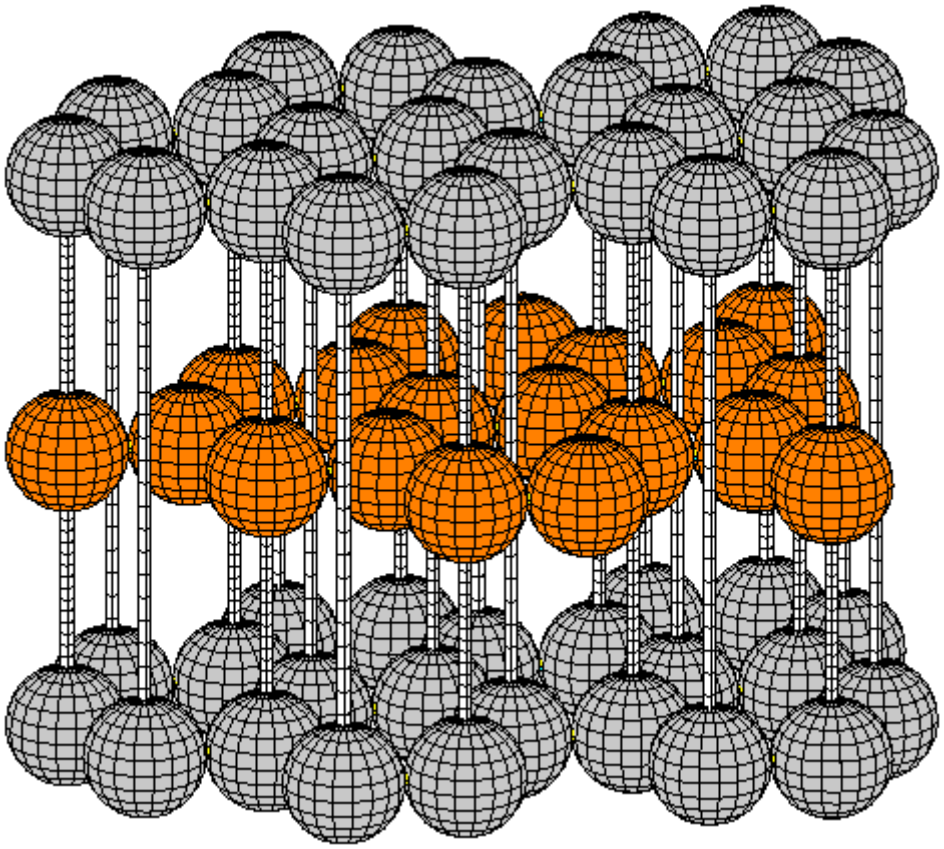
Volume V de la maille : $V = \frac{3\sqrt{3}}{2} c^2 h = 35.1\,\text{Å}^3$

Relation entre la masse molaire M et la masse volumique ρ : $M = \left(\frac{3\sqrt{3}}{8} N_A c^2 h \right) \rho$

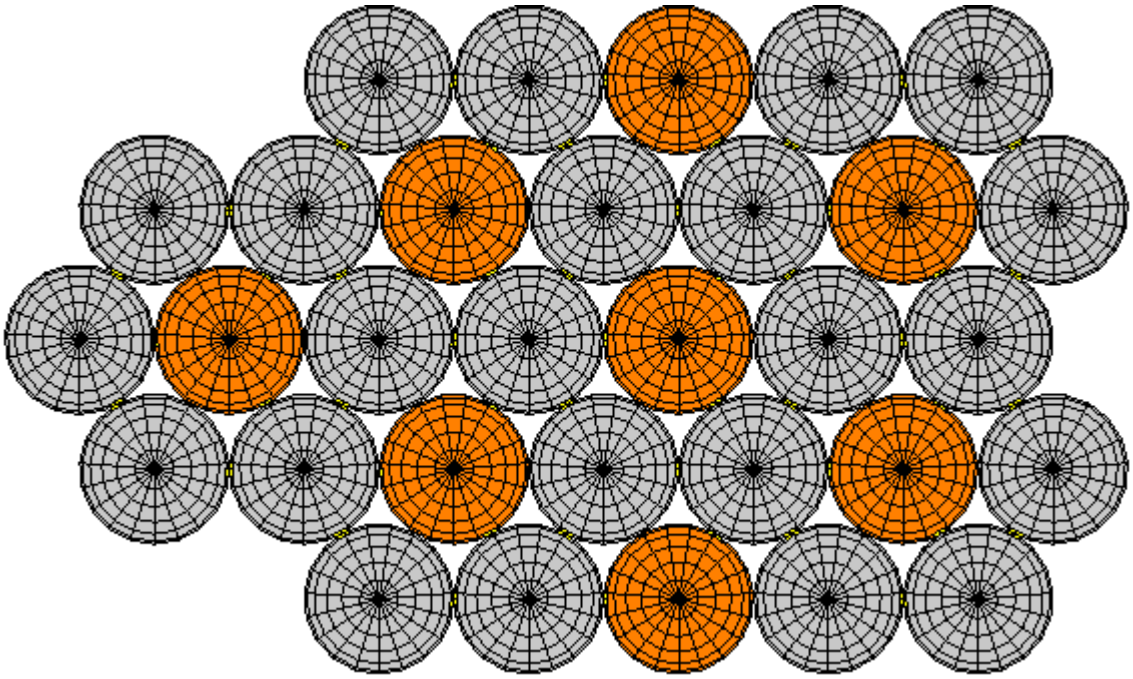
Mathématique & Informatique

Représentation 3d et 2d du décalage des feuillets (espacés de $h/2$ = 3.35 Å) dans la structure lamellaire du carbone graphite.

Rayon $r_I = r_C$; angles de vue : $\alpha = 70.5°$ et $\theta = -8.75°$



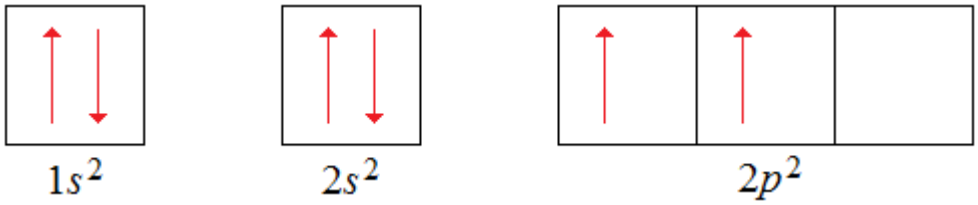
Rayon $r_I = r_C$; angles de vue : $\alpha = 0°$ et $\theta = 0°$



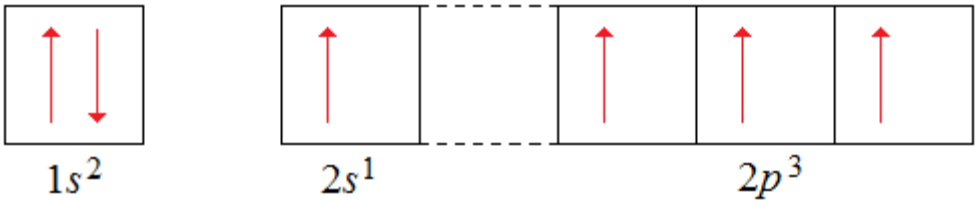
Mathématique & Informatique

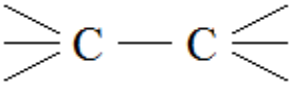
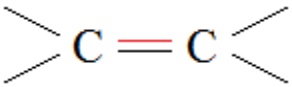
Rappel des principales caractéristiques des liaisons carbone-carbone

Structure électronique de l'atome de carbone dans son état fondamental (non hybridé) :



Structure électronique de l'atome de carbone hybridé :



	Hybridation	Liaisons	Angle	Géométrie
Liaison simple 	sp^3	7σ	109°	8 atomes dans deux tétraèdres
Liaison double 	sp^2	5σ + 1π	120°	6 atomes dans un même plan
Liaison triple 	sp	3σ + 2π	180°	4 atomes sur une même droite

Lorsqu'un atome de carbone est hybridé sp^3, il est situé au centre de gravité G d'un tétraèdre régulier (A,B,C,D) :

$\overrightarrow{GA} + \overrightarrow{GB} + \overrightarrow{GC} + \overrightarrow{GD} = \vec{0}$ avec $GA = GB = GC = GD = R$

Pour calculer l'angle α formé par deux demi-droites issues de G et passant par 2 sommets distincts de (A,B,C,D), on écrit :

$$\left(\overrightarrow{GA} + \overrightarrow{GB} + \overrightarrow{GC} + \overrightarrow{GD}\right)^2 = 0 \Rightarrow 4R^2 + 12R^2\cos\alpha = 0 \Rightarrow \cos\alpha = -\frac{1}{3} \Rightarrow \alpha = \text{Arccos}\left(-\frac{1}{3}\right) = 109° 28'$$

Mathématique & Informatique

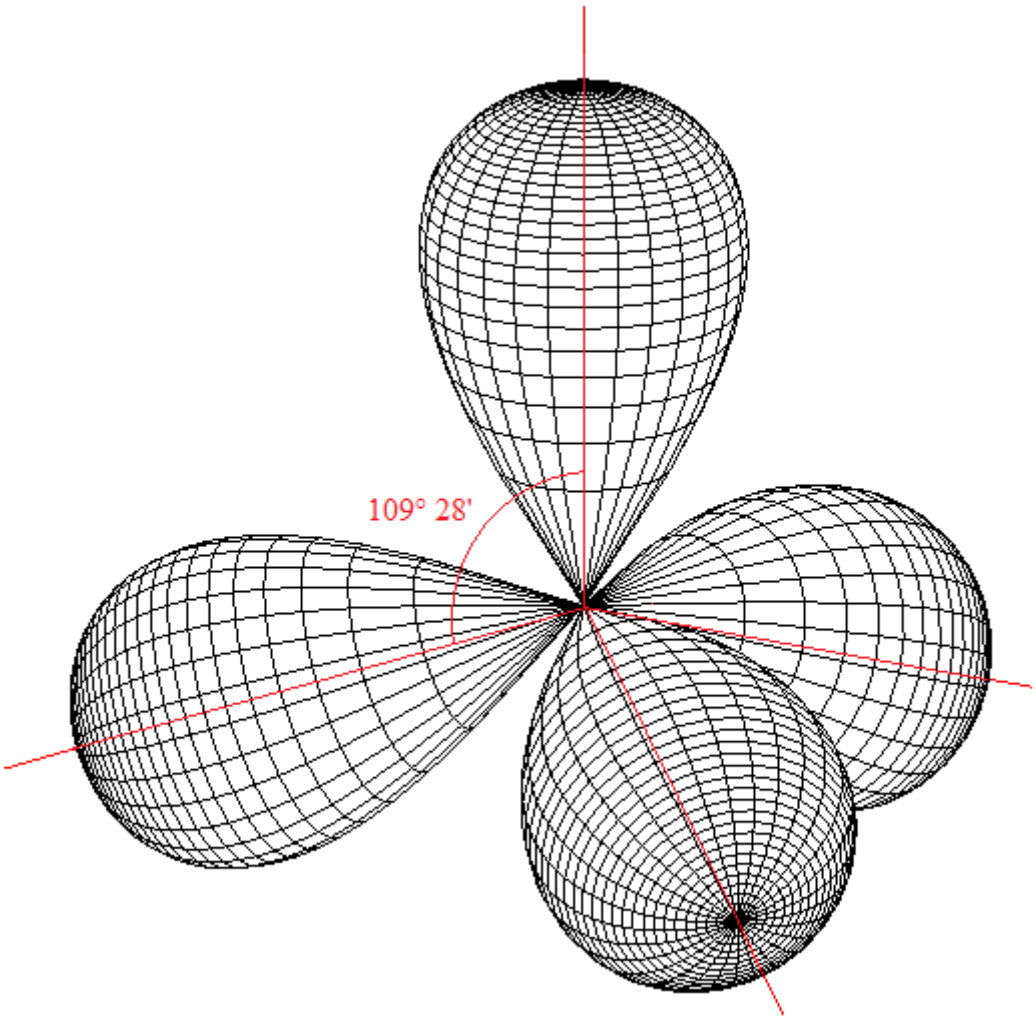
Représentation schématique des 4 orbitales piriformes hybridées sp^3, dirigées du centre du tétraèdre vers les sommets :

```
function orbitale(P,Q)
    function [x,y,z] = surface(u,v)
        r = R*sqrt(cos(k*v)./cos(v)^4)
        X = r.*sin(v).*cos(u)
        Y = r.*sin(v).*sin(u)
        Z = r.*cos(v)
        x = P(1) + I(1)*X + J(1)*Y + K(1)*Z
        y = P(2) + I(2)*X + J(2)*Y + K(2)*Z
        z = P(3) + I(3)*X + J(3)*Y + K(3)*Z
    endfunction
    k = 2.8 ; m = 30 ; n = 30
    R = norm(Q - P)
    K = (Q - P)/norm(Q - P)
    if K(1) <> 0 then, U = [K(2),- K(1),0], else, U = [0,K(3),- K(2)], end
    I = U/norm(U)
    J = [I(3)*K(2) - I(2)*K(3),I(1)*K(3) - I(3)*K(1),I(2)*K(1) - I(1)*K(2)]
    u = linspace(0,2*%pi,m)
    v = linspace(0,%pi/(2*k),n)
    [xf,yf,zf] = eval3dp(surface,u,v)
    plot3d(xf,yf,zf,alpha = alpha,theta = theta,flag = [couleur,6,0])
endfunction
```

```
// Coordonnées du tétraèdre
O = [0,0,0] ;
A = [0,0,1] ;
B = [2*sqrt(2)/3,0,-1/3] ;
C = [-sqrt(2)/3,sqrt(6)/3,-1/3] ;
D = [-sqrt(2)/3,-sqrt(6)/3,-1/3] ;

// Paramètres graphiques
alpha = 73.0 ; theta = 102.5 ;
couleur = color(255,255,255) ;

// Dessin des orbitales
orbitale(O,A)
orbitale(O,B)
orbitale(O,C)
orbitale(O,D)
```



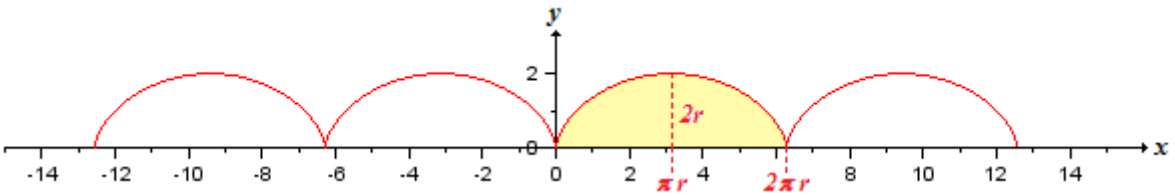
Mathématique & Informatique

La cycloïde

La cycloïde est la courbe décrite par un point fixe d'une roue de rayon *r* en déplacement rectiligne, d'où le nom de "roulette" donné par **Blaise Pascal**. Elle possède de nombreuses propriétés, notamment celle d'être égale à sa développée(*).

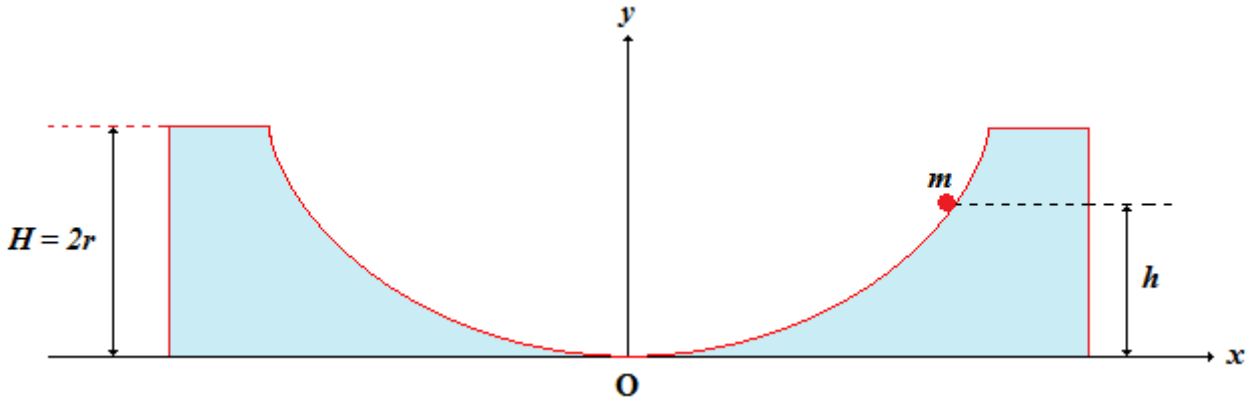
$$\begin{cases} x = r(t - \sin t) \\ y = r(1 - \cos t) \end{cases} \qquad t \in \mathbb{R}$$

```
f1 = scf(1) ;
r = 1 ;
t = linspace(-4*%pi,4*%pi,300) ;
x = r*(t - sin(t)) ;
y = r*(1 - cos(t)) ;
plot2d(x, y, frameflag = 4, style = 5)
a = gca() ; a.x_location = "origin" ; a.y_location = "origin" ;
```



- point de rebroussement $\det\left(d^2\overrightarrow{OM}/dt^2, d^3\overrightarrow{OM}/dt^3\right)_{t=2k\pi} = -r^2 \neq 0$
- longueur d'une arche $l = \int ds = \int_0^{2\pi} \sqrt{x'^2 + y'^2}\, dt = -4r\left[\cos\frac{t}{2}\right]_0^{2\pi} = 8r$
- surface sous une arche $S = \iint d\sigma = -\oint y\,dx = -\int_{2\pi}^{0} x'y\,dt = r^2\left[\frac{3}{2}t - 2\sin t + \frac{1}{4}\sin 2t\right]_0^{2\pi} = 3\pi r^2$

Une cycloïde inversée permet de concevoir des mouvements aux caractéristiques remarquables. Citons le problème de la brachistochrone résolu par **Jean** et **Jacques Bernoulli** et le principe du pendule isochrone de **Christian Huygens**.



Quelle que soit la hauteur *h* à laquelle on lâche la masse *m* (sans vitesse initiale), elle mettra toujours le même temps t_0 pour atteindre le point O, avec toutefois une vitesse v_h dépendant du point de départ. Aucune autre courbe, en partant de la même position, ne permet d'effectuer plus rapidement ce trajet. On montre que $t_0 = \pi\sqrt{\frac{r}{g}}$ où g = 9.81 m/s^2 est l'accélération de la pesanteur. Sans frottements, la période *T* du mouvement oscillatoire de la masse *m* est constante (indépendante de l'amplitude) :

$T = 4\pi\sqrt{\frac{r}{g}}$

(*) Développée d'une cycloïde

Les coordonnées $x_C(t)$ et $y_C(t)$ du centre de courbure sont données par :

$$\begin{cases} x_C = x - y' \dfrac{x'^2 + y'^2}{x'y'' - y'x''} \\ y_C = y + x' \dfrac{x'^2 + y'^2}{x'y'' - y'x''} \end{cases}$$

Appliquons à la cycloïde :

$$\begin{cases} x = r(t - \sin t) \\ y = r(1 - \cos t) \end{cases} \quad \begin{cases} x' = r(1 - \cos t) \\ y' = r \sin t \end{cases} \quad \begin{cases} x'' = r \sin t \\ y'' = r \cos t \end{cases} \quad \Rightarrow \quad \frac{x'^2 + y'^2}{x'y'' - y'x''} = -2$$

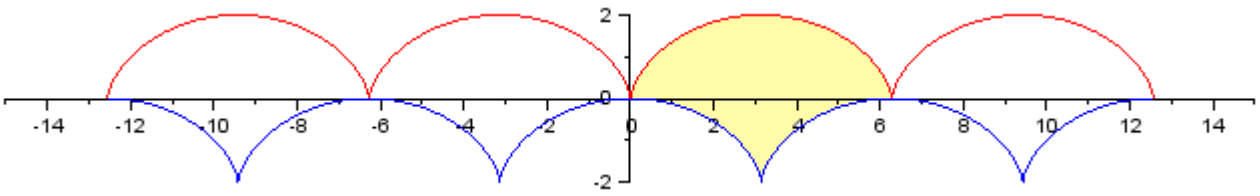
D'où :

$$\begin{cases} x_C(t) = r(t + \sin t) = x(t + \pi) - \pi r \\ y_C(t) = r(\cos t - 1) = y(t + \pi) - 2r \end{cases}$$

Donc, la développée d'une cycloïde est la même cycloïde translatée.

Effectuons une représentation graphique de la cycloïde (en rouge) et de sa développée (en bleu) :

```
r = 1 ;
t = linspace(-4*%pi,4*%pi,300) ;
x = r*(t - sin(t)) ;
y = r*(1 - cos(t)) ;
xc = r*(t + sin(t)) ;
yc = r*(cos(t) - 1) ;
plot2d(x, y, 5)
plot2d(xc, yc, 2)
a = gca() ;
a.isoview = 'on' ;
a.x_location = "origin" ;
a.y_location = "origin" ;
```



On voit que l'on peut recouvrir entièrement une surface plane avec des carreaux de faïence en forme d'arches de cycloïde.

Half-pipe cycloïdal

Notons H la hauteur et L la longueur de la surface en forme de cycloïde définie par les équations :

$$\begin{cases} x = u \\ y = \dfrac{H}{2}(v + \sin v) \\ z = \dfrac{H}{2}(1 - \cos v) \end{cases} \qquad u \in [0, L], v \in [-\pi, \pi]$$

L'aire de cette nappe paramétrée est $A = \iint_{[0,L]\times[-\pi,\pi]} \left\| \frac{\partial \overrightarrow{OM}}{\partial u} \wedge \frac{\partial \overrightarrow{OM}}{\partial v} \right\| du\,dv = H \int_0^L du \int_{-\pi}^{\pi} \cos\frac{v}{2}\, dv = 4HL$

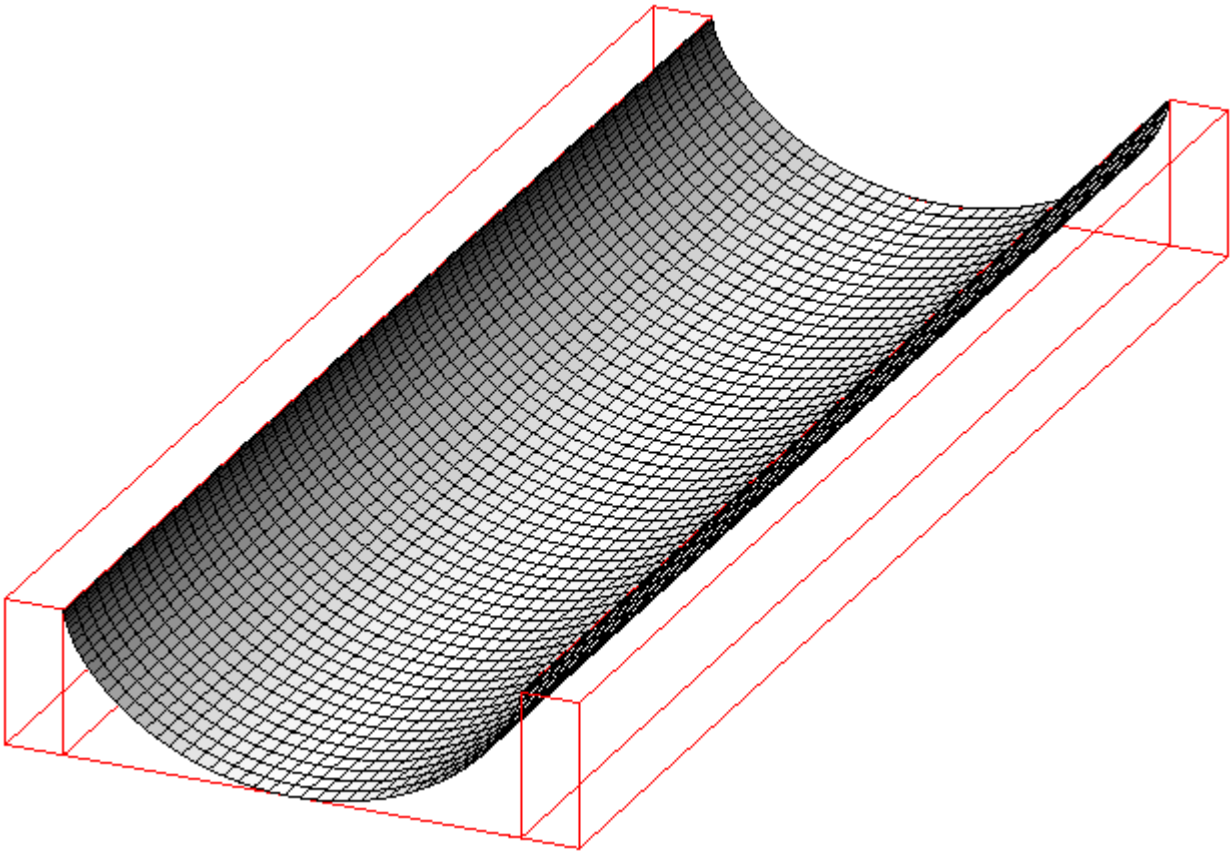
```
// Fenêtre
function cmap = colormap(c1,c2,n)
    A = linspace(c1(1),c2(1),n)'/255
    B = linspace(c1(2),c2(2),n)'/255
    C = linspace(c1(3),c2(3),n)'/255
    cmap = [A,B,C]
endfunction
f2 = scf(2) ;
f2.figure_name = 'Half-pipe cycloïdal' ;
f2.figure_size = [800,800] ;
f2.color_map = colormap([255,255,255],[50,50,50],64) ;

// Paramètres
H = 5 ; L = 50 ; l = 2 ;
i = 60 ; j = 60 ;
alpha = 66 ; theta = 24 ;
white = color(255,255,255) ;
red =   color(255,0,0) ;

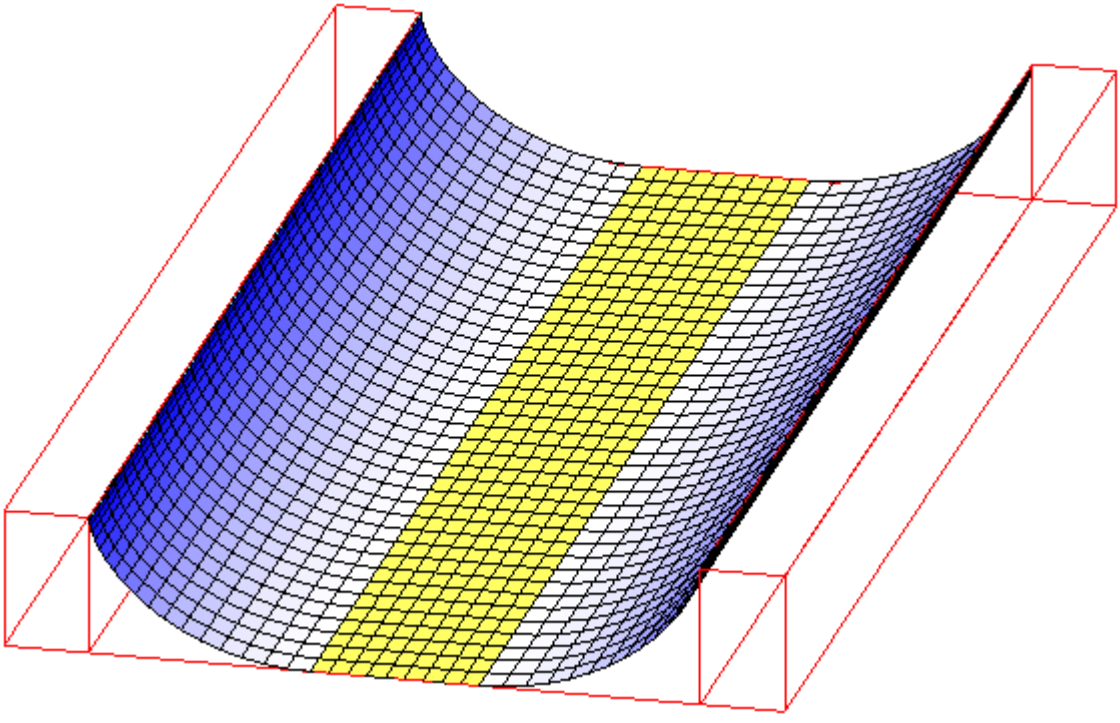
// Surface
function [x,y,z] = halfpipe(u,v)
  x = u
  y = H*(v + sin(v))/2
  z = H*(1 - cos(v))/2
endfunction
u = linspace(0, L, i) ;
v = linspace(%pi, -%pi, j) ;
[xf,yf,zf] = eval3dp(halfpipe,u,v) ;
plot3d1(xf, yf, zf, alpha = alpha, theta = theta, flag = [1,6,0])
e = gce() ; e.hiddencolor = white ;

// Cadre
a = -%pi*H/2 - l ; b = -%pi*H/2 ; c = %pi*H/2 ; d = %pi*H/2 + l ;
X = [0,0;0,0;L,L;L,L;0,0] ; Y = [a,b;d,c;d,c;a,b;a,b] ; Z = [0,0;0,0;0,0;0,0;0,0] ;
param3d1(X, Y, list(Z,[red,red]), alpha = alpha, theta = theta, flag = [6,0])
X = [0,0;0,0;L,L;L,L;0,0] ; Y = [a,c;b,d;b,d;a,c;a,c] ; Z = [H,H;H,H;H,H;H,H;H,H] ;
param3d1(X, Y, list(Z,[red,red]), alpha = alpha, theta = theta, flag = [6,0])
X = [0,0,L,L;0,0,L,L] ; Y = [a,d,d,a;a,d,d,a] ; Z = [0,0,0,0;H,H,H,H] ;
param3d1(X, Y, list(Z,[red,red,red,red]), alpha = alpha, theta = theta, flag = [6,0])
X = [0,0,L,L;0,0,L,L] ; Y = [b,c,c,b;b,c,c,b] ; Z = [0,0,0,0;H,H,H,H] ;
param3d1(X, Y, list(Z,[red,red,red,red]), alpha = alpha, theta = theta, flag = [6,0])
```

Half-pipe cycloïdal



Half-pipe cycloïdal avec partie centrale plane



On rencontre ce type de structure dans certains sports acrobatiques comme le ski freestyle, le skateboard, le BMX, …

Mathématique & Informatique

Développée d'une courbe plane

Par définition la développée d'une courbe plane est le lieu géométrique de ses centres de courbure, c'est-à-dire l'ensemble des points C tels que :

$\overrightarrow{MC} = R\vec{n}$ (R rayon de courbure algébrique et $\vec{n}$ normale unitaire au point M de la courbe).

Pour un arc paramétré quelconque $x(t)$ et $y(t)$, orienté dans le sens des t croissants, les coordonnées $x_C(t)$ et $y_C(t)$ du point C sont données par :

$$\overrightarrow{OC} = \overrightarrow{OM} + R\vec{n}$$

$$\begin{vmatrix} x_C \\ y_C \end{vmatrix} = \begin{vmatrix} x \\ y \end{vmatrix} + \frac{(x'^2 + y'^2)^{3/2}}{x'y'' - y'x''} \begin{vmatrix} -y'/\sqrt{x'^2 + y'^2} \\ x'/\sqrt{x'^2 + y'^2} \end{vmatrix}$$

$$\begin{cases} x_C = x - y' \dfrac{x'^2 + y'^2}{x'y'' - y'x''} \\ y_C = y + x' \dfrac{x'^2 + y'^2}{x'y'' - y'x''} \end{cases}$$

où :

- $x'(t)$ et $y'(t)$ sont les dérivées d'ordre 1 des fonctions $x(t)$ et $y(t)$
- $x''(t)$ et $y''(t)$ sont les dérivées d'ordre 2 des fonctions $x(t)$ et $y(t)$

Le but est d'écrire une fonction *evolute* qui trace la développée d'un arc paramétré $x(t)$, $y(t)$ pour $t_1 \le t \le t_2$.

Les fonctions $x(t)$, $y(t)$, $x'(t)$, $y'(t)$, $x''(t)$, $y''(t)$ seront fournies sous la forme de 6 chaînes de caractères (paramètres *x, y, xp, yp, xs, ys* de la fonction *evolute*). L'arc paramétré sera représenté en rouge, sa développée en bleu.

On écrira également une fonction *evolute2* où l'on fournit uniquement les fonctions $x(t)$ et $y(t)$ sous la forme de 2 chaînes de caractères (paramètres x et y). Dans ce cas les dérivées seront approximées.
Soit une fonction f et $f(a)$ sa valeur en a. On suppose que f est de classe C^4.
Considérons une petite variation h autour du point a, il existe 2 valeurs ξ_1 et ξ_2 dans $]a - h\ ,\ a + h[$ telles que :

$$f'(a) = \frac{f(a+h) - f(a-h)}{2h} - \frac{h^2}{6} f^{(3)}(\xi_1) = \frac{f(a+h) - f(a-h)}{2h} + O(h^2)$$

$$f''(a) = \frac{f(a+h) - 2f(a) + f(a-h)}{h^2} - \frac{h^2}{12} f^{(4)}(\xi_2) = \frac{f(a+h) - 2f(a) + f(a-h)}{h^2} + O(h^2)$$

Comme pour les dérivées centrales, l'application de la formule de **Taylor** à $f(a + kh)$ fournit les expressions des dérivées à droite :

$$f'(a) = \frac{-f(a+2h) + 4f(a+h) - 3f(a)}{2h} + O(h^2)$$

$$f''(a) = \frac{-f(a+3h) + 4f(a+2h) - 5f(a+h) + 2f(a)}{h^2} + O(h^2)$$

et celles des dérivées à gauche :

$$f'(a) = \frac{3f(a) - 4f(a-h) + f(a-2h)}{2h} + O(h^2)$$

$$f''(a) = \frac{2f(a) - 5f(a-h) + 4f(a-2h) - f(a-3h)}{h^2} + O(h^2)$$

Ces formules donnent une approximation de l'ordre de h^2 des dérivées première et seconde de la fonction f en a.

Développante d'une courbe plane

Par définition la développante d'une courbe plane est le lieu géométrique des points D tels que :

$\overrightarrow{MD} = -s\vec{t}$ (s abscisse curviligne et $\vec{t}$ tangente unitaire au point M de la courbe).

Pour un arc paramétré quelconque $x(t)$ et $y(t)$, orienté dans le sens des t croissants, les coordonnées $x_D(t)$ et $y_D(t)$ du point D sont données par :

$$\overrightarrow{OD} = \overrightarrow{OM} - s\vec{t}$$

$$\begin{vmatrix} x_D \\ y_D \end{vmatrix} = \begin{vmatrix} x \\ y \end{vmatrix} - \int_{t_0}^{t} \sqrt{x'^2 + y'^2}\, dt \begin{vmatrix} x'/\sqrt{x'^2 + y'^2} \\ y'/\sqrt{x'^2 + y'^2} \end{vmatrix}$$

$$\begin{cases} x_D = x - \dfrac{x'}{\sqrt{x'^2 + y'^2}} \displaystyle\int_{t_0}^{t} \sqrt{x'^2 + y'^2}\, dt \\ y_D = y - \dfrac{y'}{\sqrt{x'^2 + y'^2}} \displaystyle\int_{t_0}^{t} \sqrt{x'^2 + y'^2}\, dt \end{cases}$$

où :

- $x'(t)$ et $y'(t)$ sont les dérivées d'ordre 1 des fonctions $x(t)$ et $y(t)$
- t_0 est la valeur du paramètre t au point M_0 d'abscisse curviligne nulle : $s(t_0) = 0$

Le but est d'écrire fonction *involute* qui trace la développante d'un arc paramétré $x(t)$, $y(t)$ pour $t_1 \le t \le t_2$.

Le calcul de l'abscisse curviligne $s(t)$ s'effectuera avec la fonction Scilab *integrate* qui donne une approximation de $\int_a^b f(x)\,dx$ à partir de l'expression analytique de f.

Les fonctions $x(t)$, $y(t)$, $x'(t)$, $y'(t)$ seront fournies sous la forme de 4 chaînes de caractères (paramètres *x*, *y*, *xp*, *yp* de la fonction *involute*). L'arc paramétré sera représenté en rouge, sa développante en vert.

On écrira également une fonction *involute2* où l'on fournit uniquement les fonctions $x(t)$ et $y(t)$ sous la forme de 2 chaînes de caractères (paramètres *x* et *y*). Dans ce cas les dérivées seront approximées et l'abscisse curviligne sera approchée par la longueur algébrique d'une ligne polygonale issue de la discrétisation de la variable t :

$$s(t) = \pm \sum_{i=1}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} + \varepsilon$$

Note

La développante d'une courbe dépend du point M_0 pris comme origine des arcs alors que la développée d'une courbe est indépendante de ce point.

Mathématique & Informatique

Les fonctions de représentation graphique de la développée sont les suivantes :

```
function evolute(x,y,xp,yp,xs,ys,t1,t2)
    eps = 1e-2
    n = floor(abs(t2 - t1)/eps) + 2
    h = (t2 - t1)/(n - 1)
    X = zeros(1,n)
    Y = zeros(1,n)
    XP = zeros(1,n)
    YP = zeros(1,n)
    XS = zeros(1,n)
    YS = zeros(1,n)
    for i = 1:n
        t = t1 + (i - 1)*h
        X(i) = evstr(x)
        Y(i) = evstr(y)
        XP(i) = evstr(xp)
        YP(i) = evstr(yp)
        XS(i) = evstr(xs)
        YS(i) = evstr(ys)
    end
    K = (XP^2 + YP^2)./(XP.*YS - YP.*XS)
    XC = X - K.*YP
    YC = Y + K.*XP
    plot2d(X,Y,5)
    plot2d(XC,YC,2)
    a = gca()
    a.isoview = "on"
    a.x_location = "origin"
    a.y_location = "origin"
endfunction
```

```
function evolute2(x,y,t1,t2)
    eps = 1e-3
    n = floor(abs(t2 - t1)/eps) + 4
    h = (t2 - t1)/(n - 1)
    X = zeros(1,n)
    Y = zeros(1,n)
    XP = zeros(1,n)
    YP = zeros(1,n)
    XS = zeros(1,n)
    YS = zeros(1,n)
    for i = 1:n
        t = t1 + (i - 1)*h
        X(i) = evstr(x)
        Y(i) = evstr(y)
    end
    XP(1) = (- X(3) + 4*X(2) - 3*X(1))/(2*h)
    YP(1) = (- Y(3) + 4*Y(2) - 3*Y(1))/(2*h)
    XS(1) = (- X(4) + 4*X(3) - 5*X(2) + 2*X(1))/(h^2)
    YS(1) = (- Y(4) + 4*Y(3) - 5*Y(2) + 2*Y(1))/(h^2)
    for i = 2:n-1
        XP(i) = (X(i+1) - X(i-1))/(2*h)
        YP(i) = (Y(i+1) - Y(i-1))/(2*h)
        XS(i) = (X(i+1) - 2*X(i) + X(i-1))/(h^2)
        YS(i) = (Y(i+1) - 2*Y(i) + Y(i-1))/(h^2)
    end
    XP(n) = (3*X(n) - 4*X(n-1) + X(n-2))/(2*h)
    YP(n) = (3*Y(n) - 4*Y(n-1) + Y(n-2))/(2*h)
    XS(n) = (2*X(n) - 5*X(n-1) + 4*X(n-2) - X(n-3))/(h^2)
    YS(n) = (2*Y(n) - 5*Y(n-1) + 4*Y(n-2) - Y(n-3))/(h^2)
    K = (XP^2 + YP^2)./(XP.*YS - YP.*XS)
    XC = X - K.*YP
    YC = Y + K.*XP
    plot2d(X,Y,5)
    plot2d(XC,YC,2)
    a = gca()
    a.isoview = "on"
    a.x_location = "origin"
    a.y_location = "origin"
endfunction
```

Mathématique & Informatique

Les fonctions de représentation graphique de la développante sont les suivantes :

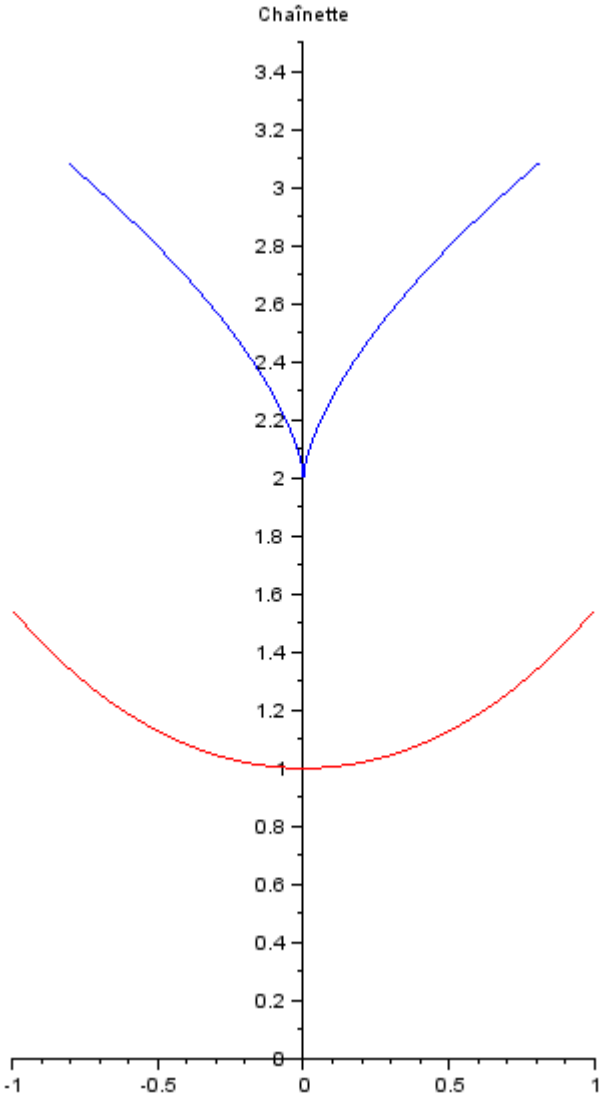
```
function involute(x,y,xp,yp,t0,t1,t2)
    eps = 1e-2
    n = floor(abs(t2 - t1)/eps) + 2
    h = (t2 - t1)/(n - 1)
    X = zeros(1,n)
    Y = zeros(1,n)
    XP = zeros(1,n)
    YP = zeros(1,n)
    S = zeros(1,n)
    for i = 1:n
        t = t1 + (i - 1)*h
        X(i) = evstr(x)
        Y(i) = evstr(y)
        XP(i) = evstr(xp)
        YP(i) = evstr(yp)
        f = 'sqrt((' + xp + ')^2 + (' + yp + ')^2)'
        S(i) = integrate(f,'t',t0,t)
    end
    XD = X - (XP./sqrt(XP^2 + YP^2)).*S
    YD = Y - (YP./sqrt(XP^2 + YP^2)).*S
    plot2d(X,Y,5)
    plot2d(XD,YD,13)
    a = gca()
    a.isoview = "on"
    a.x_location = "origin"
    a.y_location = "origin"
endfunction
```

```
function involute2(x,y,t0,t1,t2)
    function L = curve(a,b)
        L = 0
        eps = 1e-3
        n = floor(abs(b - a)/eps) + 2
        h = (b - a)/(n - 1)
        X = zeros(1,n)
        Y = zeros(1,n)
        for i = 1:n
            t = a + (i - 1)*h
            X(i) = evstr(x)
            Y(i) = evstr(y)
        end
        for i = 2:n
            L = L + sqrt((X(i) - X(i-1))^2 + (Y(i) - Y(i-1))^2)
        end
        if b < a then
            L = - L
        end
    endfunction
    eps = 1e-3
    n = floor(abs(t2 - t1)/eps) + 3
    h = (t2 - t1)/(n - 1)
    X = zeros(1,n)
    Y = zeros(1,n)
    XP = zeros(1,n)
    YP = zeros(1,n)
    S = zeros(1,n)
    for i = 1:n
        t = t1 + (i - 1)*h
        X(i) = evstr(x)
        Y(i) = evstr(y)
    end
    XP(1) = (- X(3) + 4*X(2) - 3*X(1))/(2*h)
    YP(1) = (- Y(3) + 4*Y(2) - 3*Y(1))/(2*h)
    S(1) = curve(t0,t1)
    for i = 2:n-1
        XP(i) = (X(i+1) - X(i-1))/(2*h)
        YP(i) = (Y(i+1) - Y(i-1))/(2*h)
        S(i) = S(i-1) + sqrt((X(i) - X(i-1))^2 + (Y(i) - Y(i-1))^2)
    end
    XP(n) = (3*X(n) - 4*X(n-1) + X(n-2))/(2*h)
    YP(n) = (3*Y(n) - 4*Y(n-1) + Y(n-2))/(2*h)
    S(n) = S(n-1) + sqrt((X(n) - X(n-1))^2 + (Y(n) - Y(n-1))^2)
    XD = X - (XP./sqrt(XP^2 + YP^2)).*S
    YD = Y - (YP./sqrt(XP^2 + YP^2)).*S
    plot2d(X,Y,5)
    plot2d(XD,YD,13)
    a = gca()
    a.isoview = "on"
    a.x_location = "origin"
    a.y_location = "origin"
endfunction
```

Exemples de développées

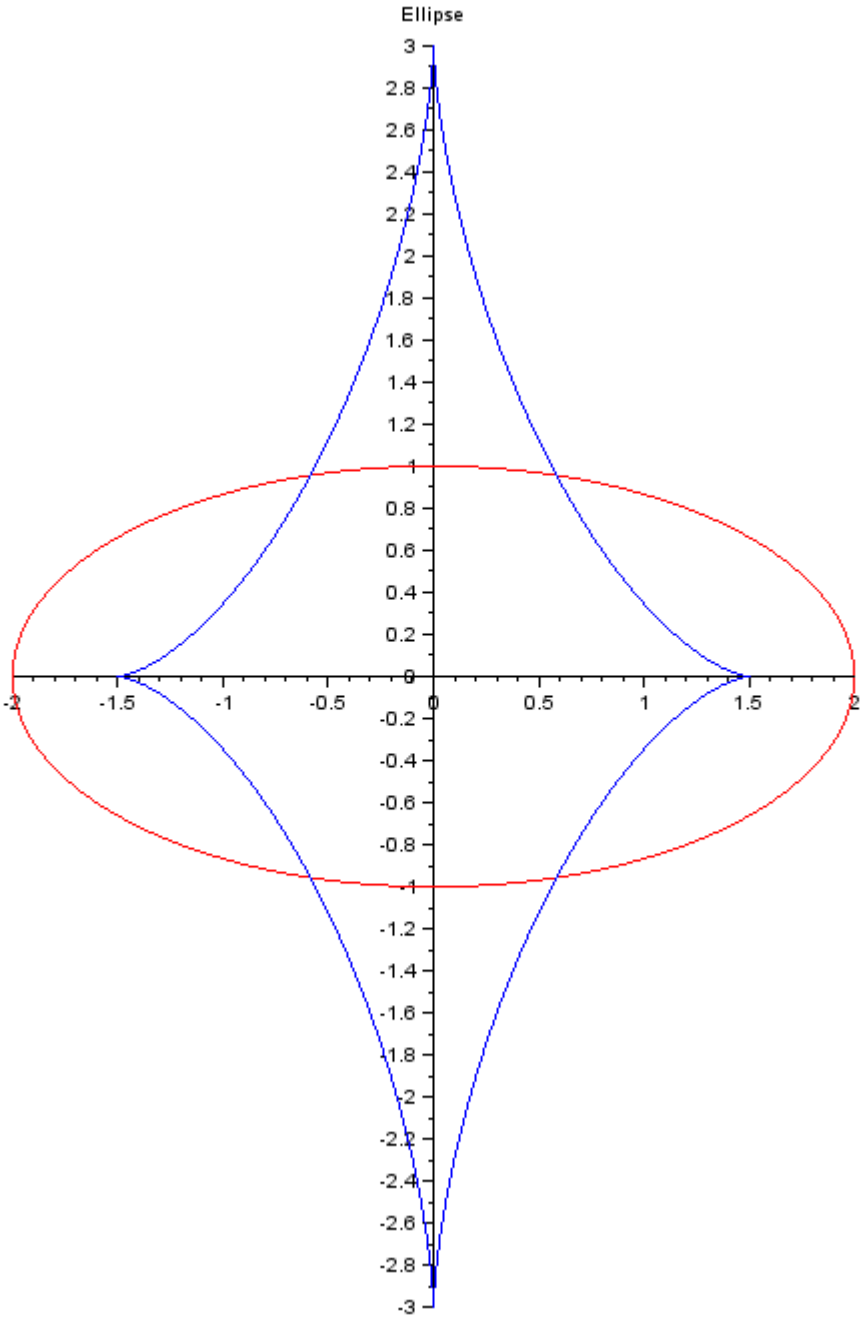
- Développée de chaînette : $y = ch(t)$, $-1 \leq t \leq 1$ (coordonnées cartésiennes)

```
scf(11) ;
xtitle('Chaînette') ;
evolute('t','cosh(t)','1','sinh(t)','0','cosh(t)',-1,1)
```



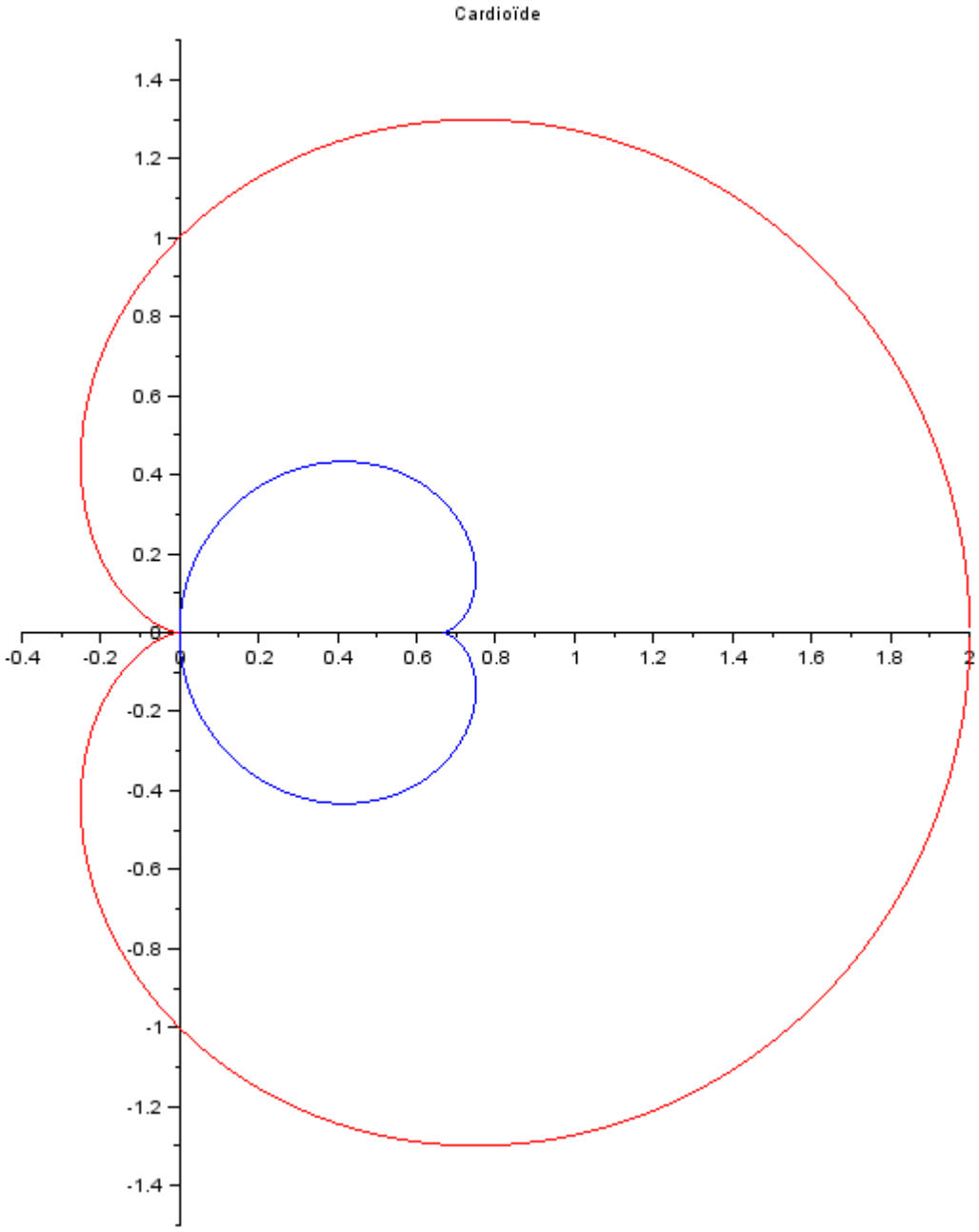
- Développée d'ellipse : $x = 2cos(t)$; $y = sin(t)$, $0 \leq t \leq 2\pi$ (coordonnées paramétriques)

```
scf(12) ;
xtitle('Ellipse') ;
evolute('2*cos(t)','sin(t)','-2*sin(t)','cos(t)','-2*cos(t)','-sin(t)',0,2*%pi)
```



- Développée de cardioïde : $r = 1 + cos(t)$, $0 \leq t \leq 2\pi$ (coordonnées polaires)

```
scf(13) ;
xtitle('Cardioïde') ;
x = '(1 + cos(t))*cos(t)' ;
y = '(1 + cos(t))*sin(t)' ;
xp = '- sin(t) - sin(2*t)' ;
yp = 'cos(t) + cos(2*t)' ;
xs = '- cos(t) - 2*cos(2*t)' ;
ys = '- sin(t) - 2*sin(2*t)' ;
evolute(x,y,xp,yp,xs,ys,0,2*%pi)
```

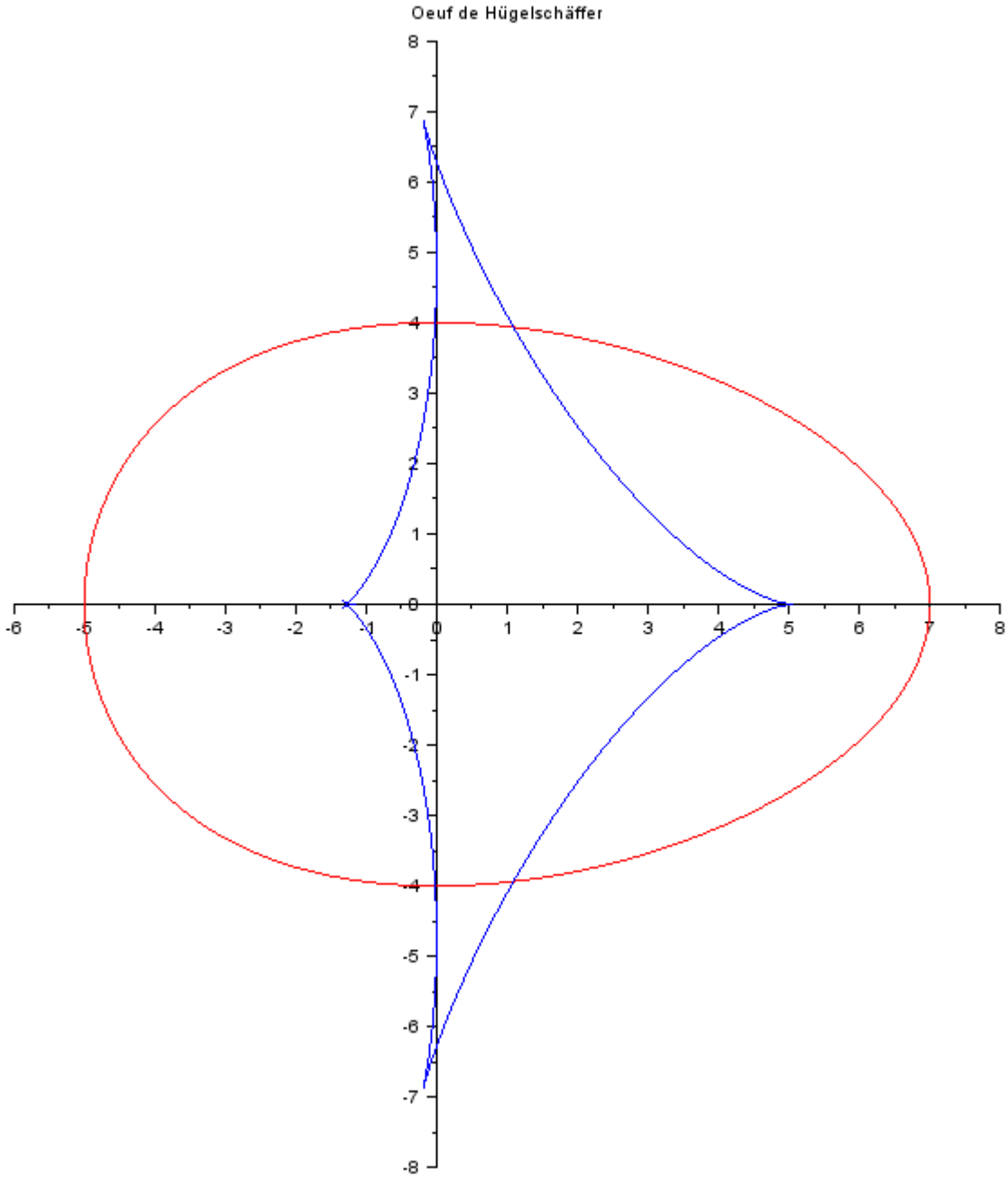


Mathématique & Informatique

- Développée de l'oeuf de **Hügelschäffer** : $x(t) = \left(\sqrt{36 - \sin^2(t)} + \cos(t)\right)\cos(t)$; $y(t) = 4\sin(t)$, $0 \le t \le 2\pi$

  ⇨ **http://www.mathcurve.com/courbes2d/oeuf/oeuf.shtml**

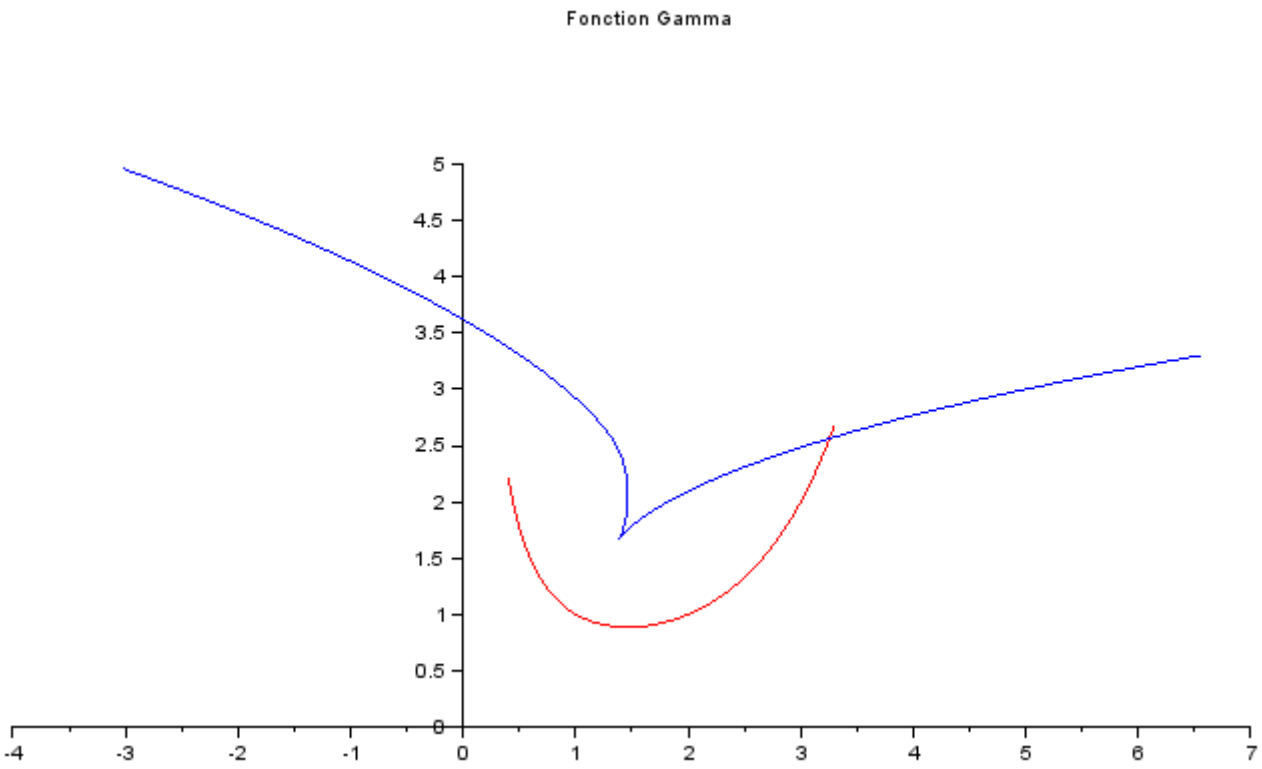
```
scf(14) ;
xtitle('Oeuf de Hügelschäffer') ;
evolute2('(sqrt(36 - sin(t)^2) + cos(t))*cos(t)','4*sin(t)',0,2*%pi)
```



Mathématique & Informatique

- Développée de la fonction Gamma : $\Gamma(t) = \int_0^{+\infty} u^{t-1} e^{-u}\, du$

```
scf(15) ;
xtitle('Fonction Gamma') ;
evolute2('t','gamma(t)',0.4,3.3)
```



Le centre de courbure au point $x = y = 1$ de la courbe de la fonction Gamma est le point minimum de sa développée.

On a : $\Gamma(1)=1, \Gamma'(1)=-\gamma, \Gamma''(1)=\gamma^2+\frac{\pi^2}{6}$. D'où :

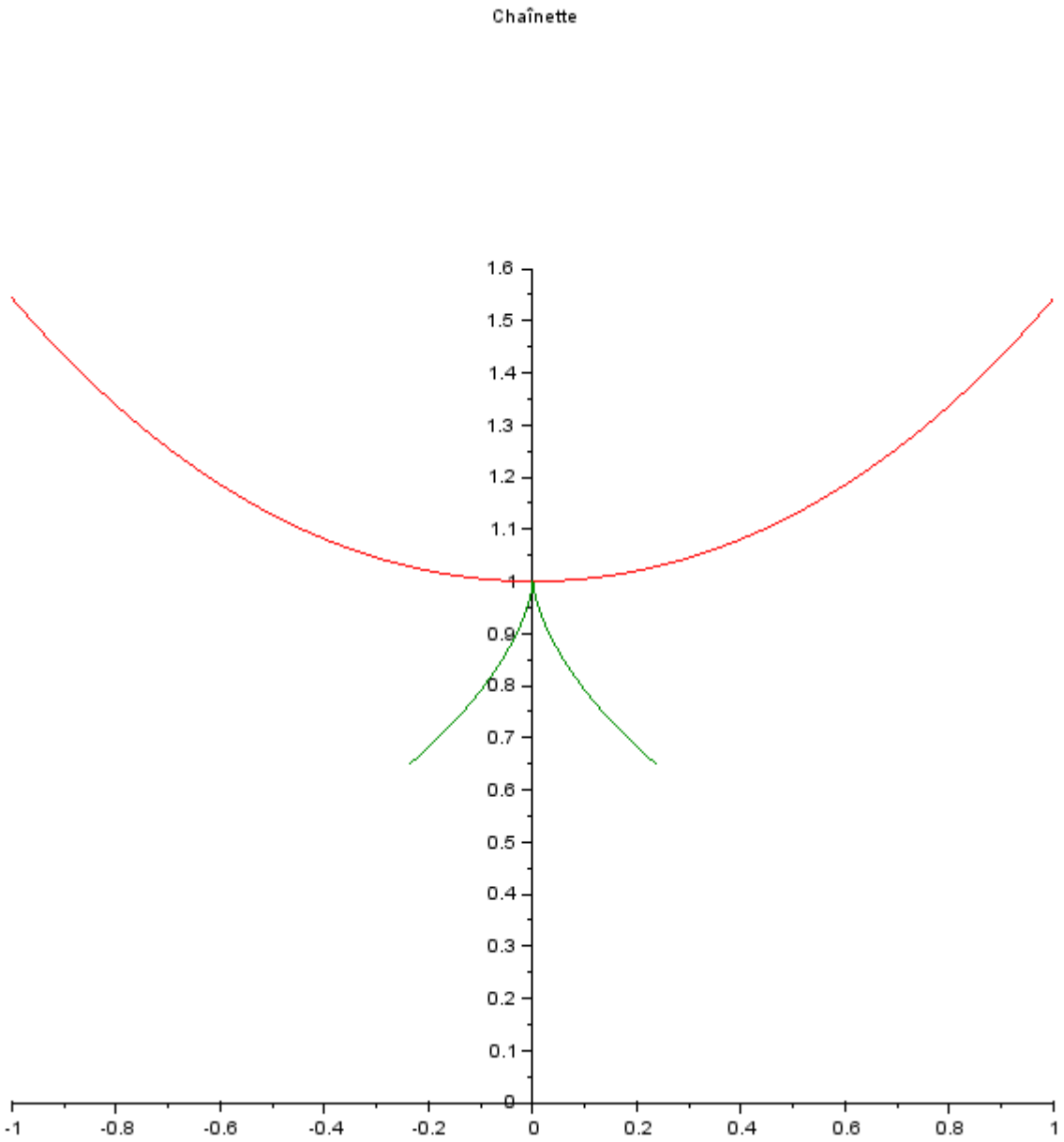
$$\begin{cases} x_C = 1 + \gamma\dfrac{\gamma^2+1}{\gamma^2+\frac{\pi^2}{6}} \approx 1.3890231 \\ y_C = 1 + \dfrac{\gamma^2+1}{\gamma^2+\frac{\pi^2}{6}} \approx 1.6739648 \end{cases}$$

(γ désigne la constante d'**Euler**)

Exemples de développantes

- Développante de chaînette

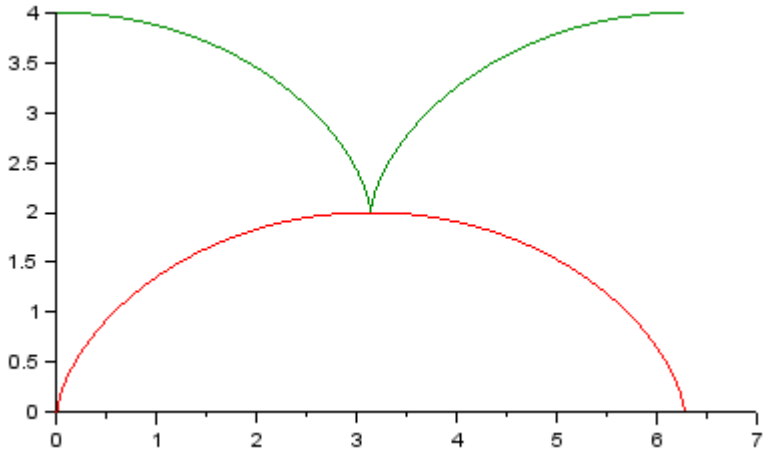
scf(21) ;
xtitle('Chaînette') ;
involute('t','cosh(t)','1','sinh(t)',0,-1,1)



- Développante de cycloïde

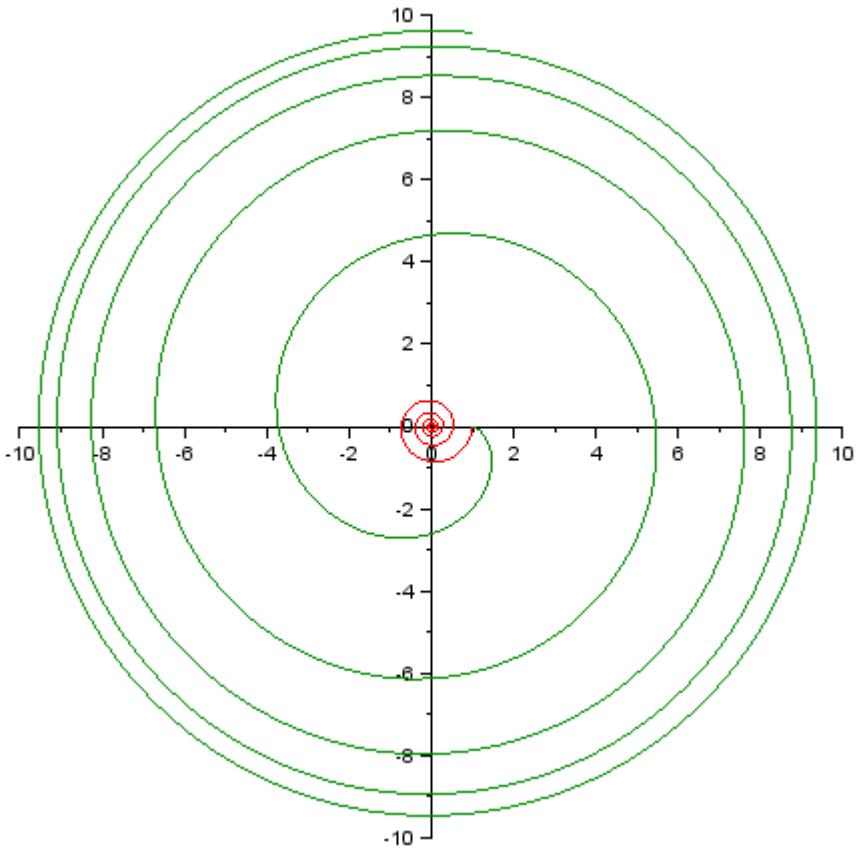
scf(22) ;
xtitle('Cycloïde') ;
involute('t - sin(t)','1 - cos(t)','1 - cos(t)','sin(t)',%pi,1e-3,2%pi - 1e-3)*

NB. On prend l'origine de l'abscisse curviligne au sommet de l'arche ($t_0 = \pi$) et on écarte les points singuliers ($t = 0$ et $t = 2\pi$).



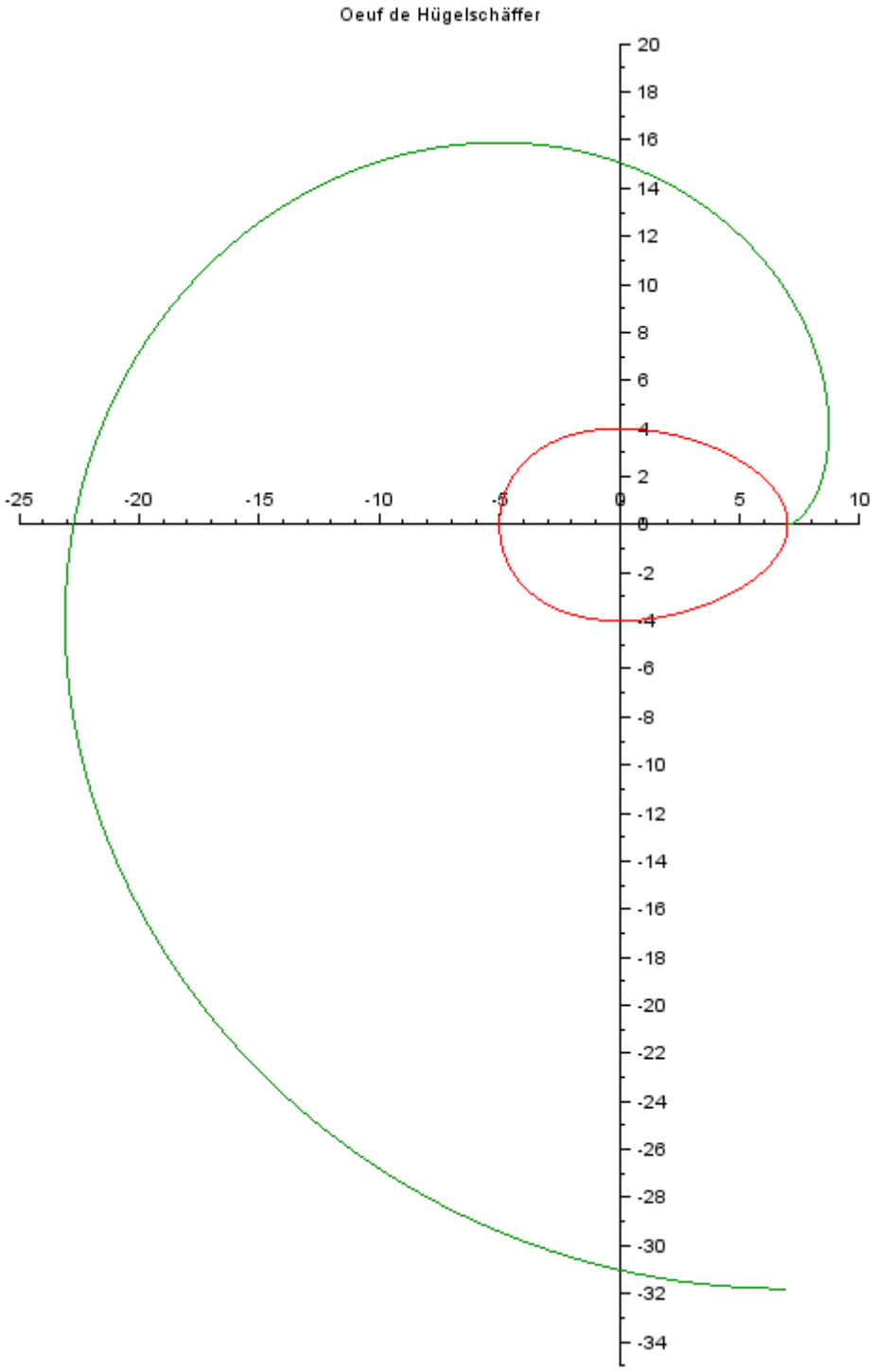
- Développante de spirale logarithmique

scf(23) ;
xtitle('Spirale') ;
*involute('exp(0.1*t)*cos(t)','exp(0.1*t)*sin(t)','exp(0.1*t)*(0.1*cos(t) - sin(t))','exp(0.1*t)*(0.1*sin(t) + cos(t))',0,-10*%pi,0)*



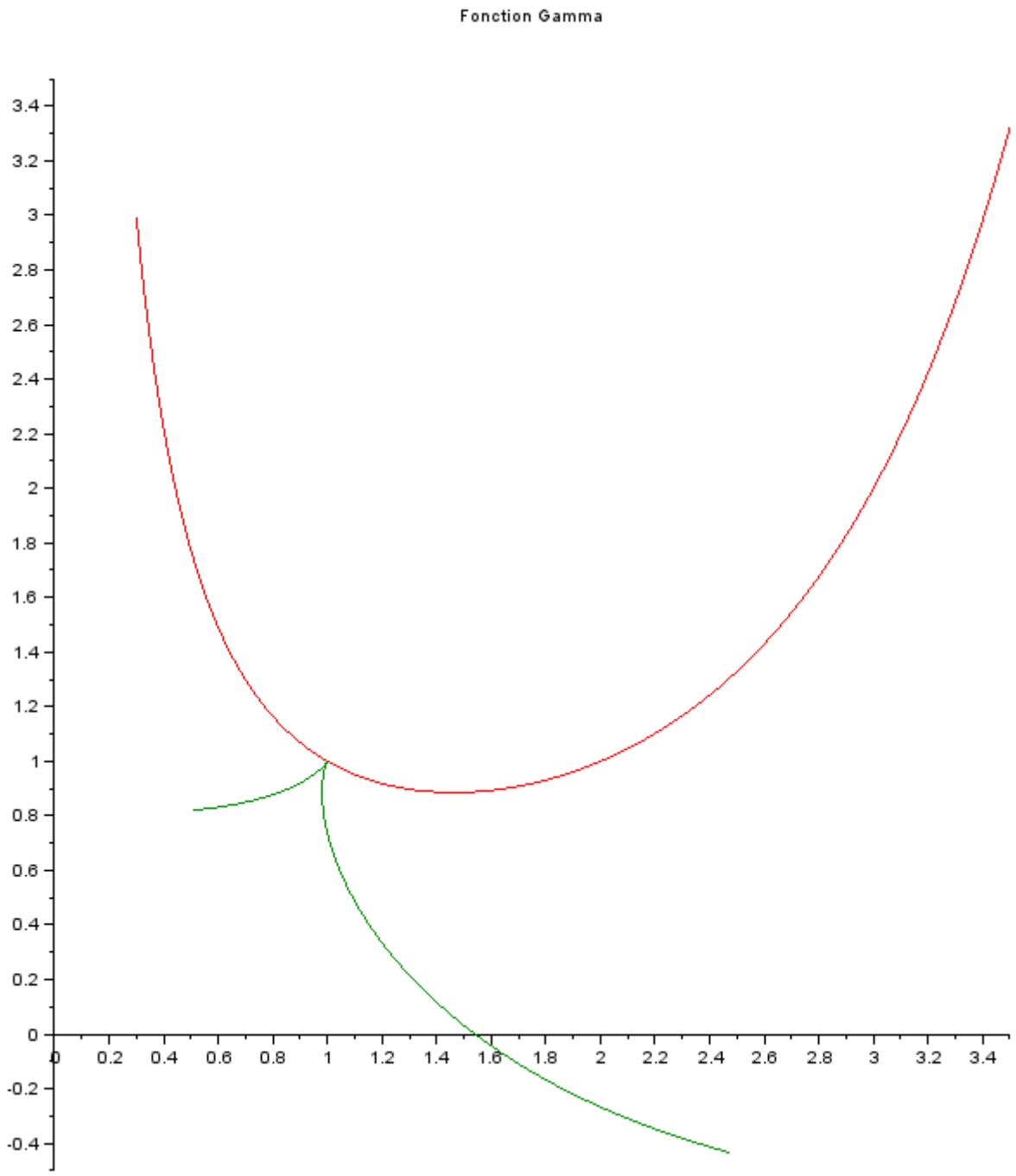
- Développante de l'oeuf de **Hügelschäffer**

scf(24) ;
xtitle('Oeuf de Hügelschäffer') ;
*involute2('(sqrt(36 - sin(t)^2) + cos(t))*cos(t)','4*sin(t)',0,2*%pi)*



- Développante de la fonction Gamma : $\Gamma(t) = \int_0^{+\infty} u^{t-1} e^{-u}\, du$

```
scf(25) ;
xtitle('Fonction Gamma') ;
involute2('t','gamma(t)',1,0.3,3.5)
```



Mathématique & Informatique

COMPLÉMENTS.

⇨ La syntaxe

Utilisation de la virgule, du point-virgule, et des points de suspension :

- La "**,**" correspond à un passage à la ligne. L'interpréteur Scilab affiche le résultat du calcul, s'il y a lieu.
- Le "**;**" correspond aussi à un passage à la ligne mais l'interpréteur Scilab n'affiche pas le résultat du calcul.
- Les "**...**" indiquent que le texte continue à la ligne suivante (le passage à la ligne n'est pas pris en compte).

L'affichage de la valeur des variables en cours d'exécution permet de déboguer facilement un programme.
Les calculs effectués par une fonction ne sont pas visualisés, contrairement à ceux du programme principal.
Pour simplifier, si l'on souhaite que Scilab n'affiche rien, il suffit de terminer chaque instruction par un point-virgule.

⇨ Les fonctions

function [résultats] = f (données)
|
|
|
endfunction

Les données (arguments d'entrée) d'une fonction f sont passées par valeur, c'est-à-dire qu'elles conservent leur valeur d'appel au retour de la fonction. Les résultats (arguments de sortie) sont des variables calculées par la fonction f.
Pour pouvoir modifier une donnée, il faut la faire figurer dans les résultats. Autrement dit, si la fonction f définie par *function* [res1, res2, ...] = f (don1, don2, ...) est appelée avec [d, r2, ...] = f (d, d2, ...) alors la donnée d est passée par variable (on dit aussi par nom, par adresse, ou par référence).

Exemple : function y = f(x) , y = 2 , x = 0 , endfunction
x = 1 ; y = f(x) ; x renvoie x = 1.
x = 1 ; x = f(x) ; x renvoie x = 2.

NB. Les données d'une fonction peuvent être de tout type, y compris une autre fonction.

⇨ La récursivité

L'écriture d'un programme récursif n'est souvent qu'une première étape pour trouver la solution d'un problème complexe.
Dans une deuxième étape, on utilise des méthodes d'élimination de la récursivité afin d'aboutir à un programme itératif plus performant en temps de calcul et en place mémoire (il faut éviter la saturation de la pile).
Le résultat est parfois peu compréhensible, mais il aurait été difficile à obtenir par une autre voie.
De plus, il est possible de prouver la validité du programme itératif, dans la mesure où on peut le faire pour le programme récursif dont il est issu (voir Algorithmique & Preuves de programmes).

⇨ Les entrées/sorties standards

Les E/S standards se font à la Console (fenêtre Scilab affectée du numéro -1) :

- La fonction *input* lit au clavier la valeur d'une variable x : x = input('x?')
- La fonction *disp* écrit à l'écran la valeur d'une variable x : disp(x)

NB. L'instruction disp(x_1, x_2, ... , x_n) affiche dans l'ordre x_n, ... , x_2, x_1 avec un saut de ligne entre chaque variable x_i.
On peut utiliser, au choix, la simple quote ' ou la double quote " pour délimiter une chaîne de caractères.
Pour insérer les caractères spéciaux ' ou " dans une chaîne, il faut les doubler. Attention toutefois à ne pas confondre la double quote (1 caractère) avec la juxtaposition de deux simples quotes (2 caractères).
Exemple : 'aujourd''hui' (' + ' + ') est équivalent à "aujourd"hui" (" + ' + ")

Mathématique & Informatique

⇨ L'échelle isométrique

Lorsque le paramètre "frameflag" (*plot2d*) ou "type" (*param3d*, *plot3d*) vaut 3 ou 4, on prend la même échelle de graduation sur les différents axes Ox, Oy, Oz du dessin. L'échelle est dite isométrique. Par contre, si ce paramètre vaut 0,1, 2, ou par défaut, alors l'échelle est non-isométrique et l'image peut être déformée. Exemple d'un cercle :

```
t = linspace(0,2*%pi,300) ; x = cos(t) ; y = sin(t) ;
subplot(1,2,1) ; plot2d(x, y)
subplot(1,2,2) ; plot2d(x, y, frameflag=4)
```

⇨ Les couleurs

Une couleur est définie par un entier de 1 à 32 (couleurs initialisées par Scilab) ou plus généralement par un mélange des 3 couleurs fondamentales : rouge, vert, bleu (Red, Green, Blue). La fonction *color*(R, G, B) , où R, G, B sont 3 entiers compris entre 0 et 255, fournit la palette des "couleurs vraies", soit 16 millions de couleurs codées sur 24 bits.
La liste des noms et des codes RGB des couleurs usuelles s'obtient en saisissant la commande d'aide "help color_list".
L'instruction id = color(R, G, B) ajoute une couleur à la carte courante ; le numéro de cette couleur est représenté par la constante nommée "id" (id = 33, 34, ...).
Exemple : orange = color(255, 165, 0) ; plot2d(0:1, 0:1, orange)
Autre possibilité : installer une nouvelle carte de n couleurs avec l'instruction *xset*("colormap", cmap) , où cmap est une matrice (n , 3) de valeurs comprises entre 0 et 1, représentant les intensités en rouge, vert, bleu des différentes couleurs.
On peut - soit créer une carte personnalisée (cmap = $[c_{11}, c_{12}, c_{13}$; ... ; $c_{n1}, c_{n2}, c_{n3}]$, $0 \le c_{ij} \le 1$) - soit utiliser une carte prédéfinie (cmap = hotcolormap(n) pour n couleurs "chaudes" du rouge au jaune ; cmap = graycolormap(n) pour n niveaux de gris du noir au blanc ; cmap = jetcolormap(n) pour n teintes variées du bleu au rouge).
Exemple : xset("colormap", jetcolormap(64)) ; getcolor

NB. Lorsqu'un programme a terminé son exécution, Scilab revient à la carte de couleurs par défaut (32 couleurs).

⇨ Les traits

La fonction *xset* permet d'effectuer des changements simples du contexte graphique de la fenêtre courante.
Elle est employée, entre autres, pour modifier l'aspect des traits d'un dessin en fonction de la valeur d'un entier n :
1. xset("line style", n) pour spécifier le type de trait (continu, pointillé, ...)
2. xset("color", n) pour choisir la couleur du trait (noir, bleu, ...)
3. xset("thickness", n) pour déterminer l'épaisseur du trait (fin, large, ...)

⇨ Les listes

L'instruction l = *list*(v_1 , ... ,v_n) crée une liste dynamique initialisée avec les valeurs v_i , analogue à un vecteur de dimension variable. La liste l se gère sans avoir recours aux pointeurs :
- Numérotation : $l(1) = v_1$, $l(2) = v_2$, ... , $l(i) = v_i$, ... , $l(n) = v_n$
- Premier élément : l(1) ; Dernier élément : l($)
- Nombre d'éléments de la liste : length(l)
- Insertion en tête de liste : l(0) = v ; insertion en queue de liste : l($+1) = v ; insertion en position i : l(i) = v
- Suppression de l'élément i : l(i) = null()
- Concaténation de 2 listes : l = lstcat(l1, l2)
- Création d'une liste vide : l = list()

En plus de la liste simple, on utilise également la liste typée : tl = *tlist*(['NOM', 'nom_1', ... , 'nom_n'], v_1 , ... , v_n)
Le vecteur de chaînes de caractères tl(1) sert à nommer les différents éléments de la liste : NOM est le nom de la liste et nom_i est le nom du i^e élément v_i. L'élément de valeur v_i est accessible par : tl(i+1) ou tl('nom_i') ou tl.nom_i ; il est modifiable avec l'instruction : tl(i+1) = v ou tl('nom_i') = v ou tl.nom_i = v.

NB. Les éléments d'une liste peuvent être de types différents (réels, matrices, chaînes de caractères, listes, ...).

Carte de couleurs personnalisée

La fonction *cmap = colormap(C,n)* ci-dessous permet de créer une carte de couleurs personnalisée.

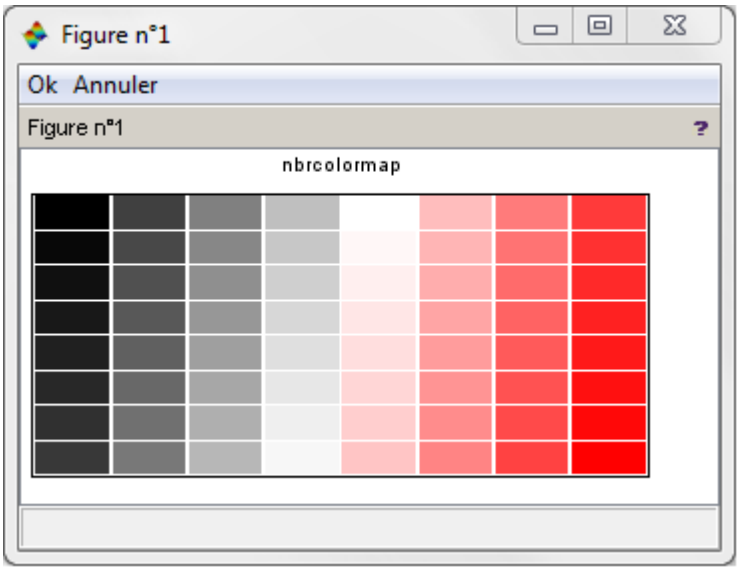
- Le paramètre d'entrée *C* est une liste de *m* triplets. Chaque élément $C(i) = [r_i,v_i,b_i]$, $1 \le i \le m$, caractérise une couleur. Les entiers r_i, v_i, b_i sont compris entre 0 et 255, ils correspondent aux intensités en Rouge, Vert, Bleu respectivement.
- Le paramètre d'entrée *n* est le nombre de couleurs de la carte ($n \ge m \ge 2$).
- Le paramètre de sortie *cmap* est une matrice (*n*,3) de *n* couleurs différentes dégradées de *C*(1) à *C*(2) … à *C*(*m*).

```
function cmap = colormap(C,n)
    function ccode = colorcode(c1,c2,k)
        R = linspace(c1(1),c2(1),k)'/255
        G = linspace(c1(2),c2(2),k)'/255
        B = linspace(c1(3),c2(3),k)'/255
        ccode = [R(1:k-1),G(1:k-1),B(1:k-1)]
    endfunction
    cmap = []
    m = length(C)
    if m < 2 | n < m then, return, end
    q = floor((n - m)/(m - 1))
    r = (n - m) - q*( m - 1)
    for i = 1:m-1
        if r <> 0 then
            cmap = [cmap ; colorcode(C(i),C(i+1),q+3)]
            r = r - 1
        else
            cmap = [cmap ; colorcode(C(i),C(i+1),q+2)]
        end
    end
    cmap = [cmap ; C(m)/255]
endfunction
```

Par exemple, pour créer une carte de 64 couleurs dégradées du noir au blanc au rouge, on écrit à la Console :

```
nbrcolormap = colormap(list([0,0,0],[255,255,255],[255,0,0]),64) ;
```

Pour visualiser cette carte à l'écran, on saisit les instructions *xset('colormap',nbrcolormap) ; getcolor* et on obtient :



Mathématique & Informatique

⇨ La partie entière

Soit x un nombre réel, la partie entière de x est un entier relatif.

Notation française : $\mathrm{E}(x)$ est l'entier relatif tel que $\mathrm{E}(x) \leq x < \mathrm{E}(x) + 1$

Notation anglo-saxonne : $\begin{cases} \lfloor x \rfloor \text{ est l'entier relatif tel que } \lfloor x \rfloor \leq x < \lfloor x \rfloor + 1 \\ \lceil x \rceil \text{ est l'entier relatif tel que } \lceil x \rceil - 1 < x \leq \lceil x \rceil \end{cases}$

$\mathrm{E}(x)$ est appelée partie entière de x.
$\lfloor x \rfloor$ est appelée partie entière inférieure de x.
$\lceil x \rceil$ est appelée partie entière supérieure de x.

On a : $\mathrm{E}(x) = \lfloor x \rfloor$

L'arrondi entier vers 0 de x (troncature) est noté : $[x] = \begin{cases} \lfloor x \rfloor \text{ si } x \geq 0 \\ \lceil x \rceil \text{ si } x < 0 \end{cases}$

On utilise aussi l'arrondi à l'entier le plus proche de x.

Les fonctions Scilab correspondantes sont :

- *floor*(x) pour calculer la partie entière inférieure de x. Par exemple : *floor*(4.5) = 4 et *floor*(- 4.5) = - 5
- *ceil*(x) pour calculer la partie entière supérieure de x. Par exemple : *ceil*(4.5) = 5 et *ceil*(- 4.5) = - 4
- *int*(x) pour calculer l'arrondi entier vers 0 de x. Par exemple : *int*(4.5) = 4 et *int*(- 4.5) = - 4
- *round*(x) pour calculer l'arrondi à l'entier le plus proche de x. Par exemple : *round*(4.5) = 5 et *round*(- 4.5) = - 5

⇨ La division entière

Pour tout couple $(a,b) \in \mathbb{N} \times \mathbb{N}^*$, il existe un unique couple $(q,r) \in \mathbb{N} \times \mathbb{N}$ tel que :

$$\begin{cases} a = bq + r \\ 0 \leq r < b \end{cases}$$

Les nombres a, b, q, r sont appelés respectivement dividende, diviseur, quotient, reste.

Notations usuelles : $\begin{cases} q = a \text{ div } b \\ r = a \text{ mod } b \end{cases}$

Instructions Scilab : $\begin{cases} q = floor(a / b) \\ r = modulo(a , b) \end{cases}$

⇨ Les polynômes

Définition d'un polynôme p de degré n en x : x = *poly*(0, 'x') ; p = $a_0 + a_1$*x + a_2*x^2 + ... + a_n*x^n
Son degré : *degree*(p). Son coefficient d'ordre k : *coeff*(p,k). Sa dérivée : *derivat*(p)
Sa valeur en un point x0 par le schéma de Horner : *horner*(p, x0)
Ses n racines dans $\mathbb{C}$: *roots*(p)
Division euclidienne de 2 polynômes a(x) et b(x) : [r,q] = *pdiv*(a,b) fournit q(x) et r(x) tels que a = b*q + r
Identité de Bézout : [pgcd,U] = *bezout*(a,b) calcule pgcd(a(x),b(x)) et u_1(x) , u_2(x) tels que u_1*a + u_2*b = pgcd(a,b)

Exemple : x = poly(0, 'x') ; [pgcd,U] = bezout(2 - 3*x^2 + x^4 , 6 - 2*x - 3*x^2 + x^3)
renvoie pgcd = $-2 + x^2$, u_1 = U(1,1) = 0.125 , u_2 = U(2,1) = - 0.375 - 0.125x

NB. bezout s'applique aux entiers codés sur 32 bits avec la fonction *int32* : [pgcd,U] = bezout(int32(n_1) , int32(n_2))

Mathématique & Informatique

Exemple d'utilisation des fonctions polynomiales Scilab

La dérivée n^e de tan(x) s'écrit comme un polynôme $P_n(X)$ de degré $n + 1$ en $X = \tan(x)$.
Ainsi :

$\tan(x) = X = P_0(X)$
$\tan'(x) = 1 + \tan^2(x) = 1 + X^2 = P_1(X)$
$\tan''(x) = 2\tan(x) + 2\tan^3(x) = 2X + 2X^3 = P_2(X)$
$\tan'''(x) = 2 + 8\tan^2(x) + 6\tan^4(x) = 2 + 8X^2 + 6X^4 = P_3(X)$
$\tan^{(4)}(x) = 16\tan(x) + 40\tan^3(x) + 24\tan^5(x) = 16X + 40X^3 + 24X^5 = P_4(X)$
$\tan^{(5)}(x) = 16 + 136\tan^2(x) + 240\tan^4(x) + 120\tan^6(x) = 16 + 136X^2 + 240X^4 + 120X^6 = P_5(X)$
…

On a la relation de récurrence :

$$\begin{cases} P_n(X) = (1+X^2)P'_{n-1}(X) \\ P_0(X) = X \end{cases}$$

Ecrire une fonction qui calcule $P_n(X)$ pour n donné.
En déduire le développement limité à l'ordre 11 au voisinage de 0 de tan(x).

La fonction de calcul de $P_n(X)$:

```
function P = PX(n)
    X = poly(0,'X')
    P = X
    for i = 1:n
        P = (1 + X^2)*derivat(P)
    end
endfunction
```

Le développement limité de tan(x) :

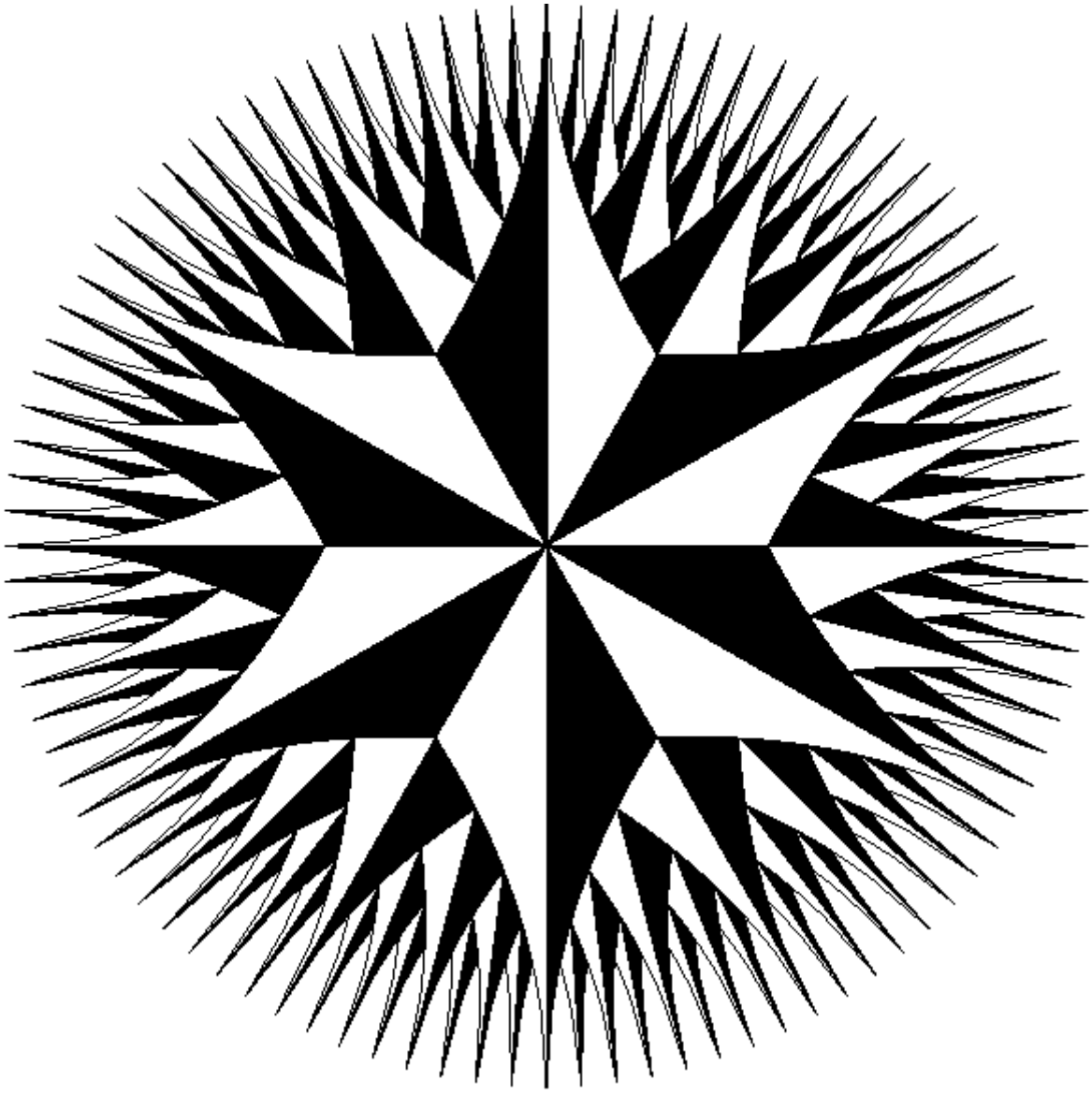
$$\tan(x) = \sum_{k=0}^{n} a_k x^k + o(x^n) \quad \text{avec } a_k = \frac{\tan^{(k)}(0)}{k!} = \frac{P_k(0)}{k!}$$

```
n = 11 ;
for k = 0:n
    n1 = coeff(PX(k),0) ;
    n2 = factorial(k) ;
    n3 = gcd(int32([n1,n2])) ;
    printf('\n a(%i) = %i/%i',k,n1/n3,n2/n3)
end
```

On trouve :

$$\tan(x) = x + \frac{1}{3}x^3 + \frac{2}{15}x^5 + \frac{17}{315}x^7 + \frac{62}{2835}x^9 + \frac{1382}{155925}x^{11} + o(x^{11})$$

Mathématique & Informatique



Mathématique & Informatique

Décomposition des fractions rationnelles en éléments simples sur $\mathbb{C}$

Etant donné deux polynômes $A(x)$ et $B(x)$ n'ayant aucun zéro commun, on définit la fraction rationnelle irréductible :

$$F(x) = \frac{A(x)}{B(x)} = E(x) + \frac{N(x)}{D(x)}$$

Le polynôme $E(x)$ est la partie entière de la fraction rationnelle $F(x)$, il est égal au quotient de la division euclidienne de $A(x)$ par $B(x)$: $A(x) = B(x)E(x) + R(x)$

NB Si deg A < deg B, alors $E(x) = 0$. Si deg A = deg B, alors $E(x) = C^{te}$.
Le numérateur $N(x)$ et le dénominateur $D(x)$ vérifient : deg N < deg D.

Supposons que le polynôme normalisé $D(x)$ de degré n ait p racines distinctes $r_1 , r_2 , \ldots , r_p$ ($r_i \in \mathbb{C}$) avec les ordres de multiplicité respectifs $m_1 , m_2 , \ldots , m_p$ ($m_i \in \mathbb{N}^*$), on a :

$$D(x) = (x-r_1)^{m_1}(x-r_2)^{m_2} \cdots (x-r_p)^{m_p} \quad \text{avec} \quad \sum_{i=1}^{p} m_i = n$$

On montre que la fraction rationnelle $\frac{N(x)}{D(x)}$ admet une décomposition (unique) en éléments simples sur $\mathbb{C}$, de la forme :

$$\frac{N(x)}{D(x)} = \frac{c_{m_1}^{(1)}}{(x-r_1)^{m_1}} + \frac{c_{m_1-1}^{(1)}}{(x-r_1)^{m_1-1}} + \cdots + \frac{c_1^{(1)}}{(x-r_1)} \qquad (\text{pôle } r_1)$$

$$+ \frac{c_{m_2}^{(2)}}{(x-r_2)^{m_2}} + \frac{c_{m_2-1}^{(2)}}{(x-r_2)^{m_2-1}} + \cdots + \frac{c_1^{(2)}}{(x-r_2)} \qquad (\text{pôle } r_2)$$

……

$$+ \frac{c_{m_p}^{(p)}}{(x-r_p)^{m_p}} + \frac{c_{m_p-1}^{(p)}}{(x-r_p)^{m_p-1}} + \cdots + \frac{c_1^{(p)}}{(x-r_p)} \qquad (\text{pôle } r_p)$$

Les $c_j^{(i)}$, $1 \le i \le p$ et $1 \le j \le m_i$, sont des nombres complexes.

Le coefficient $c_1^{(i)}$ est appelé résidu de F au pôle r_i , noté Res(F,r_i).

Soit $P_j^{(i)}(x) = \frac{D(x)}{(x-r_i)^j}$ le polynôme de degré $n - j$ associé à l'inconnue $c_j^{(i)}$:

$$P_j^{(i)}(x) = (x - r_i)^{m_i - j} \prod_{\substack{k=1 \\ k \neq i}}^{p} (x - r_k)^{m_k}$$

$P_j^{(i)}(x)$ est le polynôme "polaire" relatif au pôle r_i de multiplicité j dans la décomposition en éléments simples de la fraction rationnelle $\frac{N(x)}{D(x)}$. Il y a n polynômes $P_j^{(i)}(x)$ linéairement indépendants.

La décomposition en éléments simples précédente s'écrit :

$$\frac{N(x)}{D(x)} = \sum_{i=1}^{p}\sum_{j=1}^{m_i} \frac{c_j^{(i)}}{(x-r_i)^j} = \frac{\displaystyle\sum_{i=1}^{p}\sum_{j=1}^{m_i} c_j^{(i)}\, P_j^{(i)}(x)}{\displaystyle\prod_{i=1}^{p}(x-r_i)^{m_i}}$$

Identifions les numérateurs :

$$N(x) = \sum_{i=1}^{p}\sum_{j=1}^{m_i} c_j^{(i)}\, P_j^{(i)}(x)$$

Exprimons les polynômes $N(x)$ et $P_j^{(i)}(x)$, tous les deux de degré strictement inférieur à n :

$$N(x) = b_{n-1}\,x^{n-1} + b_{n-2}\,x^{n-2} + \cdots + b_k\,x^k + \cdots + b_1\,x + b_0$$

$$P_j^{(i)}(x) = a_{j,n-1}^{(i)}\,x^{n-1} + a_{j,n-2}^{(i)}\,x^{n-2} + \cdots + a_{j,k}^{(i)}\,x^k + \cdots + a_{j,1}^{(i)}\,x + a_{j,0}^{(i)}$$

NB Si $N(x)$ (resp. $P_j^{(i)}(x)$) est de degré $d < n-1$, alors $b_{n-1}, b_{n-2}, \cdots, b_{d+1}$ (resp. $a_{j,n-1}^{(i)}, a_{j,n-2}^{(i)}, \cdots, a_{j,d+1}^{(i)}$) sont nuls.

La relation à vérifier devient :

$$\sum_{i=1}^{p}\sum_{j=1}^{m_i} a_{j,k}^{(i)}\, c_j^{(i)} = b_k \quad \text{pour } k = 0, 1, 2, \ldots, n-1$$

On reconnait un système linéaire de n équations à n inconnues $c_j^{(i)}$.

Soit $M = \left[a_{j,k}^{(i)}\right]$ la matrice carrée d'ordre n dont les colonnes s'identifient aux composantes des polynômes $P_{m_1}^{(1)}(x)\, P_{m_1-1}^{(1)}(x) \cdots P_1^{(1)}(x) \cdots P_j^{(i)}(x) \cdots P_1^{(p)}(x)$ dans la base $\left\{x^{n-1}, x^{n-2}, \cdots, x, 1\right\}$.
Soit $b = [b_k]$ le vecteur colonne de dimension n égal aux composantes du polynôme $N(x)$ dans cette même base.
Les n inconnues $c_{m_1}^{(1)}\, c_{m_1-1}^{(1)} \cdots c_1^{(1)} \cdots c_j^{(i)} \cdots c_1^{(p)}$, regroupées dans le vecteur colonne $c = \left[c_j^{(i)}\right]$, sont les solutions du système linéaire :

$$\boxed{Mc = b}$$

On sait résoudre un tel système linéaire dans $\mathbb{C}$.
Les pages suivantes contiennent le programme Scilab correspondant à cette méthode.

Important

La fiabilité des résultats repose à la fois sur la précision des calculs de la fonction *roots* donnant les n racines d'un polynôme de degré n dans $\mathbb{C}$ (problème fondamental de l'algèbre numérique) et sur la précision des calculs de l'algorithme de résolution d'un système linéaire de n équations à n inconnues dans $\mathbb{C}$.
En pratique, on ne peut guère dépasser $n = 10$ et les racines doivent être suffisamment séparées, ce qui est généralement le cas.

```
function rfc(F)
    // Tri de nombres complexes dans l'ordre croissant
    function B = csort(A)
        function c = compare(z1,z2)
            if real(z1) < real(z2) then
                c = -1
            elseif real(z1) > real(z2) then
                c = 1
            else
                if imag(z1) < imag(z2) then
                    c = -1
                elseif imag(z1) > imag(z2) then
                    c = 1
                else
                    c = 0
                end
            end
        endfunction
        B = A
        n = length(B)
        i = 1
        while i < n
            j = i
            temp = B(i+1)
            loop = %t
            while loop
                loop = %f
                if compare(B(j),temp) == 1 then
                    B(j+1) = B(j)
                    j = j - 1
                    loop = (j >= 1)
                end
            end
            B(j+1) = temp
            i = i + 1
        end
    endfunction
    // Convertion d'un nombre complexe en une chaîne de caractères
    function str = cdisp(z)
        str = ''
        s1 = sign(real(z))
        s2 = sign(imag(z))
        if s1 == -1 then
            str = str + '- '
        end
        if s1 <> 0 | (s1 == 0 & s2 == 0) then
            str = str + string(abs(real(z)))
        end
        if s1 <> 0 & s2 <> 0 then
            str = str + ' '
        end
        if s2 == -1 then
            str = str + '- '
        end
        if s1 <> 0 & s2 == 1 then
            str = str + '+ '
        end
        if s2 <> 0 then
            str = str + string(abs(imag(z))) + 'i'
        end
    endfunction
```

```
// Contrôle de validité
if typeof(F) <> 'rational' then
    disp(F)
    printf('\n L''argument n''est pas une fraction rationnelle.')
    return
end
// Précision des calculs
eps1 = 1e-5     // racines fonction roots
eps2 = 1e-4     // séparation des racines
eps3 = 1e-5     // arrondi des coefficients
// Fraction rationnelle F(x) = A(x)/B(x)
x = poly(0,'x')
FI = simp(F)
A = varn(FI.num,'x')
B = varn(FI.den,'x')
// Partie entière de F(x)
[R,E] = pdiv(A,B)
// Normalisation de B(x)
lambda = coeff(B,degree(B))
N = R/lambda
D = B/lambda
// Racines du dénominateur D(x)
R = roots(D)
// Tri des racines réelles et imaginaires
R1 = []
R2 = []
for i = 1:length(R)
    if abs(imag(R(i))) < eps1 then
        R1 = [R1,real(R(i))]
    else
        R2 = [R2,R(i)]
    end
end
R1 = csort(R1)
R2 = csort(R2)
// Unicité des racines
r1 = []
for i = 1:length(R1)
    exist = %f
    for j = 1:i-1
        exist = abs(R1(j) - R1(i)) < eps2
        if exist then, break, end
    end
    if ~exist then
        r1 = [r1,R1(i)]
    end
end
r2 = []
for i = 1:length(R2)
    exist = %f
    for j = 1:i-1
        exist = abs(R2(j) - R2(i)) < eps2
        if exist then, break, end
    end
    if ~exist then
        r2 = [r2,R2(i)]
    end
end
r = [r1,r2]
```

```
// Ordres de multiplicité des racines
m1 = zeros(1,length(r1))
for p = 1:length(r1)
    for k = 1:length(R1)
        if abs(R1(k) - r1(p)) < eps2 then
            m1(p) = m1(p) + 1
        end
    end
end
m2 = zeros(1,length(r2))
for p = 1:length(r2)
    for k = 1:length(R2)
        if abs(R2(k) - r2(p)) < eps2 then
            m2(p) = m2(p) + 1
        end
    end
end
m = [m1,m2]
// Construction de la matrice M du système linéaire
n = degree(D)
M = zeros(n,n)
j = 0
for p = 1:length(r)
    P = 1
    for k = 1:length(r)
        if k <> p then
            P = P*(x - r(k))^m(k)
        end
    end
    for q = 1:m(p)
        if q > 1 then
            P = P*(x - r(p))
        end
        j = j + 1
        for i = 1:n
            M(i,j) = coeff(P,n - i)
        end
    end
end
// Construction du second membre b du système linéaire
b = zeros(n,1)
for i = 1:n
    b(i) = coeff(N,n - i)
end
// Résolution du système linéaire Mc = b
c = M\b
// Liste des éléments simples de F(x)
L = list()
i = 0
for p = 1:length(r)
    for q = m(p):-1:1
        i = i + 1
        L($+1) = list(c(i),r(p),q)
    end
end
```

```
    // Affichage du résultat
    disp(F)
    if max(abs(coeff(E))) > %eps then
        printf('\n Partie entière :\n')
        disp(E)
    end
    printf('\n Décomposition en éléments simples du type coeff/(' + varn(F.den) + ' - pôle)^multi dans C :\n')
    for i = 1:length(L)
        e = L(i)
        z1 = round(e(1)/eps3)*eps3
        z2 = round(e(2)/eps3)*eps3
        q = e(3)
        printf('\n     pôle = ' + cdisp(z2) + ' ; multiplicité = ' + string(q) + ' ; coefficient = ' + cdisp(z1))
    end
endfunction
```

Exemples

Décomposer en éléments simples sur $\mathbb{C}$ la fraction rationnelle $\dfrac{-13x^3 + 38x^2 + 173x + 252}{x^6 + 5x^5 - x^4 - 21x^3 + 4x^2 + 28x - 16}$

On écrit :

```
x = poly(0,'x') ;
F = (-13*x^3 + 38*x^2 + 173*x + 252)/(x^6 + 5*x^5 - x^4 - 21*x^3 + 4*x^2 + 28*x - 16) ;
rfc(F)
```

On obtient :

```
                          2     3
       252 + 173x + 38x - 13x
-----------------------------------
                 2     3   4    5   6
 - 16 + 28x + 4x - 21x - x + 5x + x
```

Décomposition en éléments simples du type coeff/(x - pôle)^multi dans C :

pôle = - 4 ; multiplicité = 1 ; coefficient = - 2
pôle = - 2 ; multiplicité = 2 ; coefficient = - 3.00001
pôle = - 2 ; multiplicité = 1 ; coefficient = 1
pôle = 1 ; multiplicité = 3 ; coefficient = 10.00001
pôle = 1 ; multiplicité = 2 ; coefficient = - 4
pôle = 1 ; multiplicité = 1 ; coefficient = 1

Soit :

$$\frac{-13x^3 + 38x^2 + 173x + 252}{x^6 + 5x^5 - x^4 - 21x^3 + 4x^2 + 28x - 16} = -\frac{2}{(x+4)} - \frac{3.00001}{(x+2)^2} + \frac{1}{(x+2)} + \frac{10.00001}{(x-1)^3} - \frac{4}{(x-1)^2} + \frac{1}{(x-1)}$$

La valeur exacte est :

$$\frac{-13x^3 + 38x^2 + 173x + 252}{x^6 + 5x^5 - x^4 - 21x^3 + 4x^2 + 28x - 16} = -\frac{2}{(x+4)} - \frac{3}{(x+2)^2} + \frac{1}{(x+2)} + \frac{10}{(x-1)^3} - \frac{4}{(x-1)^2} + \frac{1}{(x-1)}$$

Mathématique & Informatique

Décomposer en éléments simples sur $\mathbb{C}$ la fraction rationnelle $\dfrac{(3-2i)x-5+3i}{x^2+ix+2}$

On écrit :

```
x = poly(0,'x') ;
F = ((3 - 2*%i)*x - 5 + 3*%i)/(x^2 + %i*x + 2) ;
rfc(F)
```

On obtient :

```
(-5+i*3) + ( 3-i*2)x
--------------------
                   2
    2 + ix +  1x
```

Décomposition en éléments simples du type coeff/(x - pôle)^multi dans C :

pôle = - 2i ; multiplicité = 1 ; coefficient = 1 - 3i
pôle = 1i ; multiplicité = 1 ; coefficient = 2 + 1i

Soit :

$$\frac{(3-2i)x-5+3i}{x^2+ix+2}=\frac{1-3i}{(x+2i)}+\frac{2+i}{(x-i)}$$

La valeur exacte est :

$$\frac{(3-2i)x-5+3i}{x^2+ix+2}=\frac{1-3i}{(x+2i)}+\frac{2+i}{(x-i)}$$

Mathématique & Informatique

Décomposer en éléments simples sur $\mathbb{C}$ la fraction rationnelle $\dfrac{5x^6 - x^5 + 3x^4 - 5x^2 + 3x - 2}{2x^4 - 2}$

On écrit :

```
x = poly(0,'x') ;
F = (5*x^6 - x^5 + 3*x^4 - 5*x^2 + 3*x - 2)/(2*x^4 - 2) ;
rfc(F)
```

On obtient :

```
                2    4   5    6
- 2 + 3x - 5x + 3x - x + 5x
---------------------------
                  4
          - 2 + 2x
```

Partie entière :

```
                2
1.5 - 0.5x + 2.5x
```

Décomposition en éléments simples du type coeff/(x - pôle)^multi dans C :

pôle = - 1 ; multiplicité = 1 ; coefficient = 0.125
pôle = 1 ; multiplicité = 1 ; coefficient = 0.375
pôle = - 1i ; multiplicité = 1 ; coefficient = - 0.25 - 0.125i
pôle = 1i ; multiplicité = 1 ; coefficient = - 0.25 + 0.125i

Soit :

$$\frac{5x^6 - x^5 + 3x^4 - 5x^2 + 3x - 2}{2x^4 - 2} = 2.5x^2 - 0.5x + 1.5 + \frac{0.125}{(x+1)} + \frac{0.375}{(x-1)} - \frac{0.25 + 0.125i}{(x+i)} - \frac{0.25 - 0.125i}{(x-i)}$$

La valeur exacte est :

$$\frac{5x^6 - x^5 + 3x^4 - 5x^2 + 3x - 2}{2x^4 - 2} = \frac{5}{2}x^2 - \frac{1}{2}x + \frac{3}{2} + \frac{\frac{1}{8}}{(x+1)} + \frac{\frac{3}{8}}{(x-1)} - \frac{\frac{1}{4} + \frac{1}{8}i}{(x+i)} - \frac{\frac{1}{4} - \frac{1}{8}i}{(x-i)}$$

Mathématique & Informatique

Décomposer en éléments simples sur $\mathbb{C}$ la fraction rationnelle $\dfrac{1}{x^7 + 1}$

On écrit :

```
x = poly(0,'x') ;
F = 1/(x^7 + 1) ;
rfc(F)
```

On obtient :

```
  1
------
     7
1 + x
```

Décomposition en éléments simples du type coeff/(x - pôle)^multi dans C :

pôle = - 1 ; multiplicité = 1 ; coefficient = 0.14286
pôle = - 0.62349 - 0.78183i ; multiplicité = 1 ; coefficient = 0.08907 + 0.11169i
pôle = - 0.62349 + 0.78183i ; multiplicité = 1 ; coefficient = 0.08907 - 0.11169i
pôle = 0.22252 - 0.97493i ; multiplicité = 1 ; coefficient = - 0.03179 + 0.13928i
pôle = 0.22252 + 0.97493i ; multiplicité = 1 ; coefficient = - 0.03179 - 0.13928i
pôle = 0.90097 - 0.43388i ; multiplicité = 1 ; coefficient = - 0.12871 + 0.06198i
pôle = 0.90097 + 0.43388i ; multiplicité = 1 ; coefficient = - 0.12871 - 0.06198i

Soit :

$$\frac{1}{x^7 + 1} =$$

$$\frac{0.14286}{(x + 1)}$$

$$+ \frac{0.08907 + 0.11169i}{(x + 0.62349 + 0.78183i)} + \frac{0.08907 - 0.11169i}{(x + 0.62349 - 0.78183i)} - \frac{0.03179 - 0.13928i}{(x - 0.22252 + 0.97493i)}$$

$$- \frac{0.03179 + 0.13928i}{(x - 0.22252 - 0.97493i)} - \frac{0.12871 - 0.06198i}{(x - 0.90097 + 0.43388i)} - \frac{0.12871 + 0.06198i}{(x - 0.90097 - 0.43388i)}$$

La valeur exacte est :

$$\frac{1}{x^7 + 1} = \sum_{k=1}^{7} \frac{c_k}{x - r_k} \qquad \text{avec} \quad r_k = e^{i\frac{(2k-1)\pi}{7}}, \; c_k = -\frac{r_k}{7}$$

Mathématique & Informatique

Décomposer en éléments simples sur $\mathbb{C}$ la fraction rationnelle $\dfrac{t^3 - 2}{t^4(t^2 + t + 1)^2}$

On écrit :

```
t = poly(0,'t') ;
F = (t^3 - 2)/(t^4*(t^2 + t + 1)^2) ;
rfc(F)
```

On obtient :

```
                3
     - 2 + t
----------------------
  4     5     6     7   8
 t + 2t + 3t + 2t + t
```

Décomposition en éléments simples du type coeff/(t - pôle)^multi dans C :

pôle = 0 ; multiplicité = 4 ; coefficient = - 2
pôle = 0 ; multiplicité = 3 ; coefficient = 4
pôle = 0 ; multiplicité = 2 ; coefficient = - 2
pôle = 0 ; multiplicité = 1 ; coefficient = - 3
pôle = - 0.5 - 0.86603i ; multiplicité = 2 ; coefficient = - 0.16667 + 0.28868i
pôle = - 0.5 - 0.86603i ; multiplicité = 1 ; coefficient = 1.5 + 2.21318i
pôle = - 0.5 + 0.86603i ; multiplicité = 2 ; coefficient = - 0.16667 - 0.28868i
pôle = - 0.5 + 0.86603i ; multiplicité = 1 ; coefficient = 1.5 - 2.21318i

Soit :

$$\frac{t^3 - 2}{t^4(t^2 + t + 1)^2} =$$

$$-\frac{2}{t^4} + \frac{4}{t^3} - \frac{2}{t^2} - \frac{3}{t} - \frac{0.16667 - 0.28868i}{(t + 0.5 + 0.86603i)^2} + \frac{1.5 + 2.21318i}{(t + 0.5 + 0.86603i)} - \frac{0.16667 + 0.28868i}{(t + 0.5 - 0.86603i)^2} + \frac{1.5 - 2.21318i}{(t + 0.5 - 0.86603i)}$$

Posons $j = -\dfrac{1}{2} + \dfrac{\sqrt{3}}{2}i$ (racine cubique de l'unité), la valeur exacte est :

$$\frac{t^3 - 2}{t^4(t^2 + t + 1)^2} = -\frac{2}{t^4} + \frac{4}{t^3} - \frac{2}{t^2} - \frac{3}{t} + \frac{\frac{1}{3}j}{(t - j^2)^2} + \frac{\frac{3}{2} + \frac{23\sqrt{3}}{18}i}{(t - j^2)} + \frac{\frac{1}{3}j^2}{(t - j)^2} + \frac{\frac{3}{2} - \frac{23\sqrt{3}}{18}i}{(t - j)}$$

D'autres exemples à l'adresse internet ci-dessous

⇨ **http://www.iecn.u-nancy.fr/~deroton/enseignement/Calcul_et_math/TD4-fractions.pdf**

Décomposition des fractions rationnelles en éléments simples sur ℝ

Soit $F(x) = \dfrac{A(x)}{B(x)}$ une fraction rationnelle irréductible à coefficients réels.

On part de la décomposition de $F(x)$ sur ℂ pour aboutir à la décomposition de $F(x)$ sur ℝ. Dans ℂ, tous les éléments simples de la décomposition de $F(x)$ sont de la forme $\dfrac{c_k}{(x-r)^k}$, $1 \le k \le m$, où r est une racine d'ordre m de $B(x)$. Deux cas possibles :

1) <u>r est réelle</u>

Chaque terme $\dfrac{c_k}{(x-r)^k}$ est un élément simple de première espèce qui reste inchangé dans ℝ.

2) <u>r est complexe</u>

Comme $\bar{r}$ est également racine d'ordre m de $B(x)$, il existe m couples d'éléments $\dfrac{c_k}{(x-r)^k}$ et $\dfrac{\overline{c_k}}{(x-\bar{r})^k}$ $(1 \le k \le m)$.

L'algorithme de décomposition en éléments simples de deuxième espèce relatif aux racines r et $\bar{r}$ s'écrit :

$p = -2\,\mathrm{Re}(r)$
$q = |r|^2$
$Q(x) = 0$
<u>pour</u> $k = m, m-1, \ldots, 1$

- Calculer le polynôme $N(x) = 2\,\mathrm{Re}\left(\overline{c_k}\,(x-r)^k\right) + Q(x)$
- Calculer le quotient $Q(x)$ et le reste $R(x) = ax + b$ tels que : $N(x) = (x^2 + px + q)Q(x) + R(x)$
- L'élément simple de deuxième espèce d'ordre k est $\dfrac{ax+b}{(x^2+px+q)^k}$

<u>fin</u>

Finalement $F(x)$ se met sous la forme :

$$\left.\begin{aligned} F(x) = {} & \frac{A_\alpha}{(x-r_1)^\alpha} + \frac{A_{\alpha-1}}{(x-r_1)^{\alpha-1}} + \cdots + \frac{A_1}{(x-r_1)} \\ & + \frac{B_\beta}{(x-r_2)^\beta} + \frac{B_{\beta-1}}{(x-r_2)^{\beta-1}} + \cdots + \frac{B_1}{(x-r_2)} \\ & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \end{aligned}\right\} \text{ pôles réels } r_i$$

$$\left.\begin{aligned} & + \frac{M_\lambda x + N_\lambda}{(x^2+p_1x+q_1)^\lambda} + \frac{M_{\lambda-1}x + N_{\lambda-1}}{(x^2+p_1x+q_1)^{\lambda-1}} + \cdots + \frac{M_1x+N_1}{(x^2+p_1x+q_1)} \\ & + \frac{R_\mu x + S_\mu}{(x^2+p_2x+q_2)^\mu} + \frac{R_{\mu-1}x + S_{\mu-1}}{(x^2+p_2x+q_2)^{\mu-1}} + \cdots + \frac{R_1x+S_1}{(x^2+p_2x+q_2)} \\ & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \end{aligned}\right\} \text{ pôles complexes } r'_j \text{ et } \overline{r'_j}$$

où $A_\alpha, \ldots, A_1, B_\beta, \ldots, B_1, \ldots, M_\lambda, \ldots, M_1, N_\lambda, \ldots, N_1, R_\mu, \ldots, R_1, S_\mu, \ldots, S_1, \ldots$ sont des constantes réelles.

<u>Note</u> : $r'_j = \dfrac{-p_j + i\sqrt{4q_j - {p_j}^2}}{2}$ et $\overline{r'_j} = \dfrac{-p_j - i\sqrt{4q_j - {p_j}^2}}{2}$

```
function rfr(F)
    // Tri de nombres complexes dans l'ordre croissant
    function B = csort(A)
        function c = compare(z1,z2)
            if real(z1) < real(z2) then
                c = -1
            elseif real(z1) > real(z2) then
                c = 1
            else
                if imag(z1) < imag(z2) then
                    c = -1
                elseif imag(z1) > imag(z2) then
                    c = 1
                else
                    c = 0
                end
            end
        endfunction
        B = A
        n = length(B)
        i = 1
        while i < n
            j = i
            temp = B(i+1)
            loop = %t
            while loop
                loop = %f
                if compare(B(j),temp) == 1 then
                    B(j+1) = B(j)
                    j = j - 1
                    loop = (j >= 1)
                end
            end
            B(j+1) = temp
            i = i + 1
        end
    endfunction
    // Contrôle de validité
    if typeof(F) <> 'rational' then
        disp(F)
        printf('\n L''argument n''est pas une fraction rationnelle.')
        return
    end
    // Précision des calculs
    eps1 = 1e-5     // racines fonction roots
    eps2 = 1e-4     // séparation des racines
    eps3 = 1e-5     // arrondi des coefficients
    // Fraction rationnelle F(x) = A(x)/B(x)
    x = poly(0,'x')
    FI = simp(F)
    A = varn(FI.num,'x')
    B = varn(FI.den,'x')
    // Partie entière de F(x)
    [R,E] = pdiv(A,B)
    // Normalisation de B(x)
    lambda = coeff(B,degree(B))
    N = R/lambda
    D = B/lambda
```

```
// Racines du dénominateur D(x)
R = roots(D)
// Tri des racines réelles et imaginaires
R1 = []
R2 = []
for i = 1:length(R)
    if abs(imag(R(i))) < eps1 then
        R1 = [R1,real(R(i))]
    else
        R2 = [R2,R(i)]
    end
end
R1 = csort(R1)
R2 = csort(R2)
// Unicité des racines
r1 = []
for i = 1:length(R1)
    exist = %f
    for j = 1:i-1
        exist = abs(R1(j) - R1(i)) < eps2
        if exist then, break, end
    end
    if ~exist then
        r1 = [r1,R1(i)]
    end
end
r2 = []
for i = 1:length(R2)
    exist = %f
    for j = 1:i-1
        exist = abs(R2(j) - R2(i)) < eps2
        if exist then, break, end
    end
    if ~exist then
        r2 = [r2,R2(i)]
    end
end
r = [r1,r2]
// Ordres de multiplicité des racines
m1 = zeros(1,length(r1))
for p = 1:length(r1)
    for k = 1:length(R1)
        if abs(R1(k) - r1(p)) < eps2 then
            m1(p) = m1(p) + 1
        end
    end
end
m2 = zeros(1,length(r2))
for p = 1:length(r2)
    for k = 1:length(R2)
        if abs(R2(k) - r2(p)) < eps2 then
            m2(p) = m2(p) + 1
        end
    end
end
m = [m1,m2]
```

```
// Construction de la matrice M du système linéaire
n = degree(D)
M = zeros(n,n)
j = 0
for p = 1:length(r)
    P = 1
    for k = 1:length(r)
        if k <> p then
            P = P*(x - r(k))^m(k)
        end
    end
    for q = 1:m(p)
        if q > 1 then
            P = P*(x - r(p))
        end
        j = j + 1
        for i = 1:n
            M(i,j) = coeff(P,n - i)
        end
    end
end
// Construction du second membre b du système linéaire
b = zeros(n,1)
for i = 1:n
    b(i) = coeff(N,n - i)
end
// Résolution du système linéaire Mc = b
c = M\b
// Liste des éléments simples de F(x)
L = list()
z = []
i = 0
for p = 1:length(r)
    for q = m(p):-1:1
        i = i + 1
        z1 = c(i)
        z2 = r(p)
        if imag(z2) == 0 then
            L($+1) = list(real(z1),real(z2),q)
        elseif imag(z2) > 0 then
            if z2 <> z then
                z = z2
                Q = 0
                v = - 2*real(z2)
                w = abs(z2)^2
                D = x^2 + v*x + w
            end
            N = 2*real(conj(z1)*(x - z2)^q) + Q
            [R,Q] = pdiv(N,D)
            t = coeff(R,1)
            u = coeff(R,0)
            L($+1) = list(t,u,v,w,q)
        end
    end
end
```

```
    // Affichage du résultat
    disp(F)
    if max(abs(coeff(E))) > %eps then
        printf('\n Partie entière :\n')
        disp(E)
    end
    flag1 = %t
    flag2 = %t
    for i = 1:length(L)
        e = L(i)
        select length(e)
        case 3 then
            if flag1 then
                printf('\n Eléments simples de première espèce du type ')
                printf('c/(' + varn(F.den) + ' - r)^k dans R :\n')
                flag1 = %f
            end
            c = string(round(e(1)/eps3)*eps3)
            r = string(round(e(2)/eps3)*eps3)
            k = string(e(3))
            printf('\n c = ' + c + ' ; r = ' + r + ' ; k = ' + k)
        case 5 then
            if flag2 then
                printf('\n\n Eléments simples de deuxième espèce du type ')
                printf('(a' + varn(F.den) + ' + b)/(' + varn(F.den) + '^2 + p' + varn(F.den) + ' + q)^k dans R :\n')
                flag2 = %f
            end
            a = string(round(e(1)/eps3)*eps3)
            b = string(round(e(2)/eps3)*eps3)
            p = string(round(e(3)/eps3)*eps3)
            q = string(round(e(4)/eps3)*eps3)
            k = string(e(5))
            printf('\n a = ' + a + ' ; b = ' + b + ' ; p = ' + p + ' ; q = ' + q + ' ; k = ' + k)
        end
    end
endfunction
```

Mathématique & Informatique

Exemples

Décomposer en éléments simples sur $\mathbb{R}$ la fraction rationnelle $\dfrac{4}{x^7 + 3x^6 + 5x^5 + 7x^4 + 7x^3 + 5x^2 + 3x + 1}$

On écrit :

```
x = poly(0,'x') ;
F = 4/(x^7 + 3*x^6 + 5*x^5 + 7*x^4 + 7*x^3 + 5*x^2 + 3*x + 1) ;
rfr(F)
```

On obtient :

```
                    4
-----------------------------------
            2     3     4     5     6    7
1 + 3x + 5x + 7x + 7x + 5x + 3x + x
```

Eléments simples de première espèce du type c/(x - r)^k dans R :

c = 0.99998 ; r = -1.00001 ; k = 3
c = 1.99997 ; r = -1.00001 ; k = 2
c = 1.99998 ; r = -1.00001 ; k = 1

Eléments simples de deuxième espèce du type (ax + b)/(x^2 + px + q)^k dans R :

a = -1 ; b = -0.99998 ; p = 0 ; q = 1 ; k = 2
a = -1.99998 ; b = 0.00002 ; p = 0 ; q = 1 ; k = 1

Soit :

$$\frac{4}{x^7 + 3x^6 + 5x^5 + 7x^4 + 7x^3 + 5x^2 + 3x + 1} = \frac{0.99998}{(x+1.00001)^3} + \frac{1.99997}{(x+1.00001)^2} + \frac{1.99998}{(x+1.00001)} - \frac{x+0.99998}{(x^2+1)^2} - \frac{1.99998x - 0.00002}{(x^2+1)}$$

La valeur exacte est :

$$\frac{4}{x^7 + 3x^6 + 5x^5 + 7x^4 + 7x^3 + 5x^2 + 3x + 1} = \frac{1}{(x+1)^3} + \frac{2}{(x+1)^2} + \frac{2}{(x+1)} - \frac{x+1}{(x^2+1)^2} - \frac{2x}{(x^2+1)}$$

Mathématique & Informatique

Décomposer en éléments simples sur $\mathbb{R}$ la fraction rationnelle $\dfrac{x^7 + 3}{x^6 + 3x^5 + 9x^4 + 13x^3 + 18x^2 + 12x + 8}$

On écrit :

```
x = poly(0,'x') ;
F = (x^7 + 3)/(x^6 + 3*x^5 + 9*x^4 + 13*x^3 + 18*x^2 + 12*x + 8) ;
rfr(F)
```

On obtient :

```
                    7
                3 + x
---------------------------------
              2     3     4     5   6
8 + 12x + 18x + 13x + 9x + 3x + x
```

Partie entière :

- 3 + x

Eléments simples de deuxième espèce du type (ax + b)/(x^2 + px + q)^k dans R :

a = 7.00005 ; b = 12.9996 ; p = 1.00001 ; q = 1.99999 ; k = 3
a = -7.00027 ; b = -20.99959 ; p = 1.00001 ; q = 1.99999 ; k = 2
a = 0 ; b = 14 ; p = 1.00001 ; q = 1.99999 ; k = 1

Soit :

$$\frac{x^7 + 3}{x^6 + 3x^5 + 9x^4 + 13x^3 + 18x^2 + 12x + 8} =$$

$$x - 3 + \frac{7.00005x + 12.9996}{(x^2 + 1.00001x + 1.99999)^3} - \frac{7.00027x + 20.99959}{(x^2 + 1.00001x + 1.99999)^2} + \frac{14}{(x^2 + 1.00001x + 1.99999)}$$

La valeur exacte est :

$$\frac{x^7 + 3}{x^6 + 3x^5 + 9x^4 + 13x^3 + 18x^2 + 12x + 8} = x - 3 + \frac{7x + 13}{(x^2 + x + 2)^3} - \frac{7x + 21}{(x^2 + x + 2)^2} + \frac{14}{(x^2 + x + 2)}$$

Mathématique & Informatique

Décomposer en éléments simples sur $\mathbb{R}$ la fraction rationnelle $\dfrac{1}{x^7 + 1}$

On écrit :

```
x = poly(0,'x') ;
F = 1/(x^7 + 1) ;
rfr(F)
```

On obtient :

```
  1
------
     7
1 + x
```

Eléments simples de première espèce du type c/(x - r)^k dans R :

c = 0.14286 ; r = -1 ; k = 1

Eléments simples de deuxième espèce du type (ax + b)/(x^2 + px + q)^k dans R :

a = 0.17814 ; b = 0.28571 ; p = 1.24698 ; q = 1 ; k = 1
a = -0.06358 ; b = 0.28571 ; p = -0.44504 ; q = 1 ; k = 1
a = -0.25742 ; b = 0.28571 ; p = -1.80194 ; q = 1 ; k = 1

Soit :

$$\frac{1}{x^7 + 1} = \frac{0.14286}{(x + 1)} + \frac{0.17814x + 0.28571}{(x^2 + 1.24698x + 1)} - \frac{0.06358x - 0.28571}{(x^2 - 0.44504x + 1)} - \frac{0.25742x - 0.28571}{(x^2 - 1.80194x + 1)}$$

La valeur exacte est :

$$\frac{1}{x^7 + 1} = \frac{\frac{1}{7}}{(x+1)} + \sum_{k=1}^{3} \frac{-\frac{2}{7}\left(\cos\frac{(2k-1)\pi}{7}\right)x + \frac{2}{7}}{x^2 - 2\left(\cos\frac{(2k-1)\pi}{7}\right)x + 1}$$

Mathématique & Informatique

Décomposer en éléments simples sur $\mathbb{R}$ la fraction rationnelle $\dfrac{t^3 - 2}{t^4(t^2 + t + 1)^2}$

On écrit :

```
t = poly(0,'t') ;
F = (t^3 - 2)/(t^4*(t^2 + t + 1)^2) ;
rfr(F)
```

On obtient :

```
                3
     - 2 + t
--------------------
 4    5    6    7   8
t + 2t + 3t + 2t + t
```

Eléments simples de première espèce du type c/(t - r)^k dans R :

c = -2 ; r = 0 ; k = 4
c = 4 ; r = 0 ; k = 3
c = -2 ; r = 0 ; k = 2
c = -3 ; r = 0 ; k = 1

Eléments simples de deuxième espèce du type (at + b)/(t^2 + pt + q)^k dans R :

a = 1 ; b = 1 ; p = 1 ; q = 1 ; k = 2
a = 3 ; b = 5 ; p = 1 ; q = 1 ; k = 1

Soit :

$$\frac{t^3 - 2}{t^4(t^2 + t + 1)^2} = -\frac{2}{t^4} + \frac{4}{t^3} - \frac{2}{t^2} - \frac{3}{t} + \frac{t+1}{(t^2+t+1)^2} + \frac{3t+5}{(t^2+t+1)}$$

La valeur exacte est :

$$\frac{t^3 - 2}{t^4(t^2 + t + 1)^2} = -\frac{2}{t^4} + \frac{4}{t^3} - \frac{2}{t^2} - \frac{3}{t} + \frac{t+1}{(t^2+t+1)^2} + \frac{3t+5}{(t^2+t+1)}$$

Mathématique & Informatique

Décomposer la fraction rationnelle $\frac{x^3 - x - 1}{x^5 - 8x^4 + 46x^3 + 27x^2 - 171x + 147}$ en éléments simples sur $\mathbb{C}$ puis sur $\mathbb{R}$

On écrit :

```
x = poly(0,'x') ;
F = (x^3 - x - 1)/(x^5 - 8*x^4 + 46*x^3 + 27*x^2 - 171*x + 147) ;
rfc(F)
rfr(F)
```

On obtient :

```
                    3
          - 1 - x + x
------------------------------
                2     3    4   5
147 - 171x + 27x + 46x - 8x + x
```

Décomposition en éléments simples du type coeff/(x - pôle)^multi dans C :

pôle = - 2.1038 ; multiplicité = 1 ; coefficient = - 0.01137
pôle = 1.0519 - 0.56524i ; multiplicité = 1 ; coefficient = 0.0087 - 0.01182i
pôle = 1.0519 + 0.56524i ; multiplicité = 1 ; coefficient = 0.0087 + 0.01182i
pôle = 4 - 5.74456i ; multiplicité = 1 ; coefficient = - 0.00302 + 0.08663i
pôle = 4 + 5.74456i ; multiplicité = 1 ; coefficient = - 0.00302 - 0.08663i

Eléments simples de première espèce du type c/(x - r)^k dans R :

c = -0.01137 ; r = -2.1038 ; k = 1

Eléments simples de deuxième espèce du type (ax + b)/(x^2 + px + q)^k dans R :

a = 0.0174 ; b = -0.03166 ; p = -2.1038 ; q = 1.42599 ; k = 1
a = -0.00603 ; b = 1.0194 ; p = -8 ; q = 49 ; k = 1

Soit :

$$\frac{x^3 - x - 1}{x^5 - 8x^4 + 46x^3 + 27x^2 - 171x + 147}$$

$$= -\frac{0.01137}{(x + 2.1038)} + \frac{0.0087 - 0.01182i}{(x - 1.0519 + 0.56524i)} + \frac{0.0087 + 0.01182i}{(x - 1.0519 - 0.56524i)} - \frac{0.00302 - 0.08663i}{(x - 4 + 5.74456i)} - \frac{0.00302 + 0.08663i}{(x - 4 - 5.74456i)}$$

$$= -\frac{0.01137}{(x + 2.1038)} + \frac{0.0174x - 0.03166}{(x^2 - 2.1038x + 1.42599)} - \frac{0.00603x - 1.0194}{(x^2 - 8x + 49)}$$

Calculons directement (sans utiliser les fonctions *rfc* ou *rfr*) la valeur des coefficients de la décomposition en éléments simples de $F(x) = \frac{A(x)}{B(x)}$.

Le dénominateur se factorise : $B(x) = x^5 - 8x^4 + 46x^3 + 27x^2 - 171x + 147 = (x^3 - 3x + 3)(x^2 - 8x + 49)$

D'où : $F(x) = \frac{x^3 - x - 1}{(x^3 - 3x + 3)(x^2 - 8x + 49)}$

Mathématique & Informatique

L'équation du troisième degré $x^3 - 3x + 3 = 0$ admet une racine réelle et 2 racines complexes conjuguées :

$r_0 = u + v$; $r_1 = -\frac{1}{2}(u+v) - i\frac{\sqrt{3}}{2}(u-v)$; $\overline{r_1} = -\frac{1}{2}(u+v) + i\frac{\sqrt{3}}{2}(u-v)$ avec $u = -\sqrt[3]{\frac{3-\sqrt{5}}{2}}$ et $v = -\sqrt[3]{\frac{3+\sqrt{5}}{2}}$

Numériquement : $r_0 = -2.1038034$; $r_1 = 1.0519017 - 0.5652359i$

L'équation du deuxième degré $x^2 - 8x + 49 = 0$ admet 2 racines complexes conjuguées :

$r_2 = 4 - i\sqrt{33}$; $\overline{r_2} = 4 + i\sqrt{33}$

Numériquement : $r_2 = 4. - 5.7445626i$

Dans $\mathbb{C}$, on a la décomposition : $F(x) = \frac{c_0}{(x - r_0)} + \frac{c_1}{(x - r_1)} + \frac{\overline{c_1}}{(x - \overline{r_1})} + \frac{c_2}{(x - r_2)} + \frac{\overline{c_2}}{(x - \overline{r_2})}$

Les résidus relatifs aux pôles r_0 , r_1 , r_2 sont donnés par :

$c_0 = \frac{A(r_0)}{B'(r_0)}$; $c_1 = \frac{A(r_1)}{B'(r_1)}$; $c_2 = \frac{A(r_2)}{B'(r_2)}$ où $B'(x) = 5x^4 - 32x^3 + 138x^2 + 54x - 171$ est le polynôme dérivé de $B(x)$

Numériquement : $c_0 = -0.0113664$; $c_1 = 0.0086989 - 0.0118197i$; $c_2 = -0.0030157 + 0.0866275i$

Dans $\mathbb{R}$, on a la décomposition : $F(x) = \frac{c_0}{(x - r_0)} + \frac{a_1 x + b_1}{(x^2 + p_1 x + q_1)} + \frac{a_2 x + b_2}{(x^2 + p_2 x + q_2)}$

Avec :

$a_k = 2\mathrm{Re}(c_k)$; $b_k = -2\mathrm{Re}(\overline{c_k}\, r_k)$; $p_k = -2\mathrm{Re}(r_k)$; $q_k = |r_k|^2$ $\quad (k = 1, 2)$

Numériquement :

$a_1 = 0.0173978$; $b_1 = -0.0316627$; $p_1 = -2.1038034$; $q_1 = 1.4259888$

$a_2 = -0.0060314$; $b_2 = 1.0193997$; $p_2 = -8$; $q_2 = 49$

Programme de calcul des coefficients :

```
x = poly(0,'x') ;
A = x^3 - x - 1 ;
B = (x^3 - 3*x + 3)*(x^2 - 8*x + 49) ;
C = derivat(B) ;
u = - ((3 - sqrt(5))/2)^(1/3) ;
v = - ((3 + sqrt(5))/2)^(1/3) ;
r0 = u + v
r1 = - (u + v)/2 - %i*sqrt(3)/2*(u - v)
r2 = 4 - %i*sqrt(33)
c0 = horner(A,r0)/horner(C,r0)
c1 = horner(A,r1)/horner(C,r1)
c2 = horner(A,r2)/horner(C,r2)
a1 = 2*real(c1)
b1 = - 2*real(conj(c1)*r1)
p1 = - 2*real(r1)
q1 = abs(r1)^2
a2 = 2*real(c2)
b2 = - 2*real(conj(c2)*r2)
p2 = - 2*real(r2)
q2 = abs(r2)^2
```

Mathématique & Informatique

Décomposition des fractions rationnelles en éléments simples - Interface LaTeX

Avec l'éditeur mathématique LaTeX, une fraction $\frac{a}{b}$ est représentée par la chaîne de caractères "$\frac{a}{b}$".

Un élément simple du type $\frac{a_i x + b_i}{\left(x^2 + p_i x + q_i\right)^{k_i}}$ par "$\frac{a_ix+b_i}{\left(x^2+p_ix+q_i\right)^{k_i}}$".

Pour afficher une formule LaTeX à l'intérieur d'une fenêtre Scilab, on utilise la fonction *xstring* :

xstring(*x*,*y*,*str*) où *x* et *y* sont les coordonnées de la formule représentée par la chaîne de caractères *str*.

```
//-------------------------------------------------------------------------------
//       Décomposition des fractions rationnelles en éléments simples sur R
//-------------------------------------------------------------------------------


//-----------------------------------
//   Loïc Vrillon (Juillet 2014)
//-----------------------------------


function str=rf(F)
    // Tri de nombres complexes dans l'ordre croissant
    function B=csort(A)
        function c=compare(z1, z2)
            if real(z1) < real(z2) then
                c = -1
            elseif real(z1) > real(z2) then
                c = 1
            else
                if imag(z1) < imag(z2) then
                    c = -1
                elseif imag(z1) > imag(z2) then
                    c = 1
                else
                    c = 0
                end
            end
        endfunction
        B = A
        n = length(B)
        i = 1
        while i < n
            j = i
            temp = B(i+1)
            loop = %t
            while loop
                loop = %f
                if compare(B(j),temp) == 1 then
                    B(j+1) = B(j)
                    j = j - 1
                    loop = (j >= 1)
                end
            end
            B(j+1) = temp
            i = i + 1
        end
    endfunction
```

Mathématique & Informatique

```
// Conversion au format LaTeX
function pltx=P_LATEX(p)
    pltx = ''
    if p == [] then, return, end
    for i = degree(p):-1:0
        a = coeff(p,i)
        s = sign(a)
        stra = string(abs(a))
        if strrchr(stra,'D') <> '' then
            stra = '(' + stra + ')'
        end
        stri = string(i)
        select s
        case -1 then
            pltx = pltx + '-'
        case +1 then
            if pltx <> '' then
                pltx = pltx + '+'
            end
        end
        if s <> 0 then
            if i > 1 then
                if abs(a) <> 1 then
                    pltx = pltx + stra
                end
                pltx = pltx + 'x^{' + stri + '}'
            elseif i == 1 then
                if abs(a) <> 1 then
                    pltx = pltx + stra
                end
                pltx = pltx + 'x'
            else
                pltx = pltx + stra
            end
        elseif degree(p) == 0 then
            pltx = '0'
        end
    end
endfunction
function fltx=F_LATEX(pnum, pden, k)
    pltx1 = P_LATEX(pnum)
    pltx2 = P_LATEX(pden)
    if k > 1 & pltx2 <> 'x' then
        fltx = '\frac{' + pltx1 + '}{\left(' + pltx2 + '\right)^{' + string(k) + '}}'
    elseif k <> 1 then
        fltx = '\frac{' + pltx1 + '}{' + pltx2 + '^{' + string(k) + '}}'
    else
        fltx = '\frac{' + pltx1 + '}{' + pltx2 + '}'
    end
endfunction
// Précision des calculs
eps1 = 1e-3      // racines fonction roots
eps2 = 1e-3      // séparation des racines
eps3 = 1e-3      // arrondi des coefficients
// Fraction rationnelle F(x) = A(x)/B(x)
F = simp(F)
A = F.num
B = F.den
// Partie entière de F(x)
[R,E] = pdiv(A,B)
// Normalisation de B(x)
lambda = coeff(B,degree(B))
N = R/lambda
D = B/lambda
```

```
// Racines du dénominateur D(x)
R = roots(D)
// Tri des racines réelles et imaginaires
R1 = []
R2 = []
for i = 1:length(R)
    if abs(imag(R(i))) < eps1 then
        R1 = [R1,real(R(i))]
    else
        R2 = [R2,R(i)]
    end
end
R1 = csort(R1)
R2 = csort(R2)
// Unicité des racines
r1 = []
for i = 1:length(R1)
    exist = %f
    for j = 1:i-1
        exist = abs(R1(j) - R1(i)) < eps2
        if exist then, break, end
    end
    if ~exist then
        r1 = [r1,R1(i)]
    end
end
r2 = []
for i = 1:length(R2)
    exist = %f
    for j = 1:i-1
        exist = abs(R2(j) - R2(i)) < eps2
        if exist then, break, end
    end
    if ~exist then
        r2 = [r2,R2(i)]
    end
end
r = [r1,r2]
// Ordres de multiplicité des racines
m1 = zeros(1,length(r1))
for p = 1:length(r1)
    for k = 1:length(R1)
        if abs(R1(k) - r1(p)) < eps2 then
            m1(p) = m1(p) + 1
        end
    end
end
m2 = zeros(1,length(r2))
for p = 1:length(r2)
    for k = 1:length(R2)
        if abs(R2(k) - r2(p)) < eps2 then
            m2(p) = m2(p) + 1
        end
    end
end
m = [m1,m2]
```

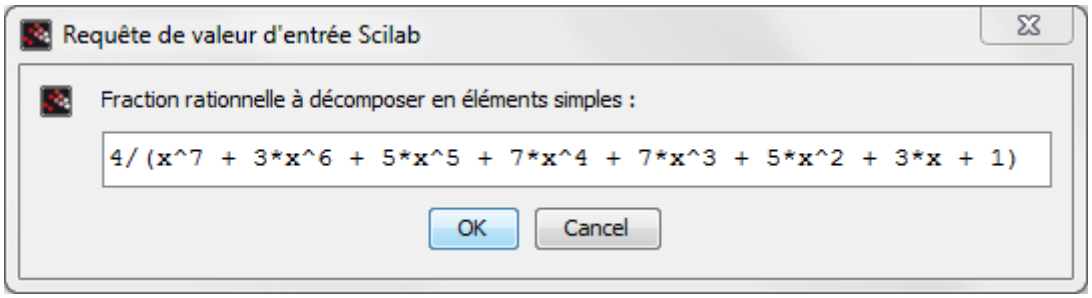
```
// Construction de la matrice M du système linéaire
n = degree(D)
M = zeros(n,n)
j = 0
for p = 1:length(r)
    P = 1
    for k = 1:length(r)
        if k <> p then
            P = P*(x - r(k))^m(k)
        end
    end
    for q = 1:m(p)
        if q > 1 then
            P = P*(x - r(p))
        end
        j = j + 1
        for i = 1:n
            M(i,j) = coeff(P,n - i)
        end
    end
end
// Construction du second membre b du système linéaire
b = zeros(n,1)
for i = 1:n
    b(i) = coeff(N,n - i)
end
// Résolution du système linéaire Mc = b
c = M\b
// Liste des éléments simples de F(x)
L = list()
z = []
i = 0
for p = 1:length(r)
    for q = m(p):-1:1
        i = i + 1
        z1 = c(i)
        z2 = r(p)
        if imag(z2) == 0 then
            L($+1) = list(real(z1),real(z2),q)
        elseif imag(z2) > 0 then
            if z2 <> z then
                z = z2
                Q = 0
                v = - 2*real(z2)
                w = abs(z2)^2
                D = x^2 + v*x + w
            end
            N = 2*real(conj(z1)*(x - z2)^q) + Q
            [R,Q] = pdiv(N,D)
            t = coeff(R,1)
            u = coeff(R,0)
            L($+1) = list(t,u,v,w,q)
        end
    end
end
```

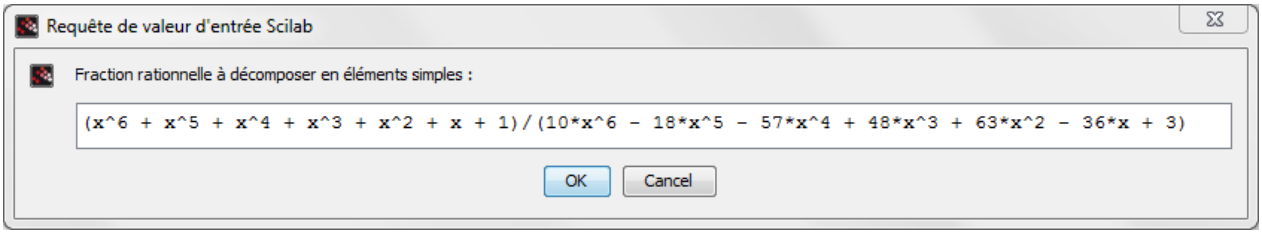
```
    // Construction de la formule LaTeX
    str = F_LATEX(A,B,1)
    if max(abs(coeff(E))) > %eps then
        E = P_LATEX(E)
    else
        E = ''
    end
    str = str + '=' + E
    for i = 1:length(L)
        e = L(i)
        select length(e)
        case 3 then
            c = round(e(1)/eps3)*eps3
            r = round(e(2)/eps3)*eps3
            k = e(3)
            s = sign(c)
            select s
            case -1 then
                s = '-'
                c = - c
            case +1 then
                s = '+'
            case 0 then
                s = ''
            end
            pnum = c
            pden = x - r
        case 5 then
            a = round(e(1)/eps3)*eps3
            b = round(e(2)/eps3)*eps3
            p = round(e(3)/eps3)*eps3
            q = round(e(4)/eps3)*eps3
            k = e(5)
            s1 = sign(a)
            s2 = sign(b)
            select s1
            case -1 then
                s = '-'
                a = - a
                b = - b
            case +1 then
                s = '+'
            case 0 then
                select s2
                case -1 then
                    s = '-'
                    b = - b
                case +1 then
                    s = '+'
                case 0 then
                    s = ''
                end
            end
            pnum = a*x + b
            pden = x^2 + p*x + q
        end
        if s == '-' then
            str = str + s + F_LATEX(pnum,pden,k)
        elseif s == '+' then
            if part(str,length(str)) <> '=' then
                str = str + s + F_LATEX(pnum,pden,k)
            else
                str = str + F_LATEX(pnum,pden,k)
            end
        end
    end
    str = "$" + str + "$"
endfunction
```

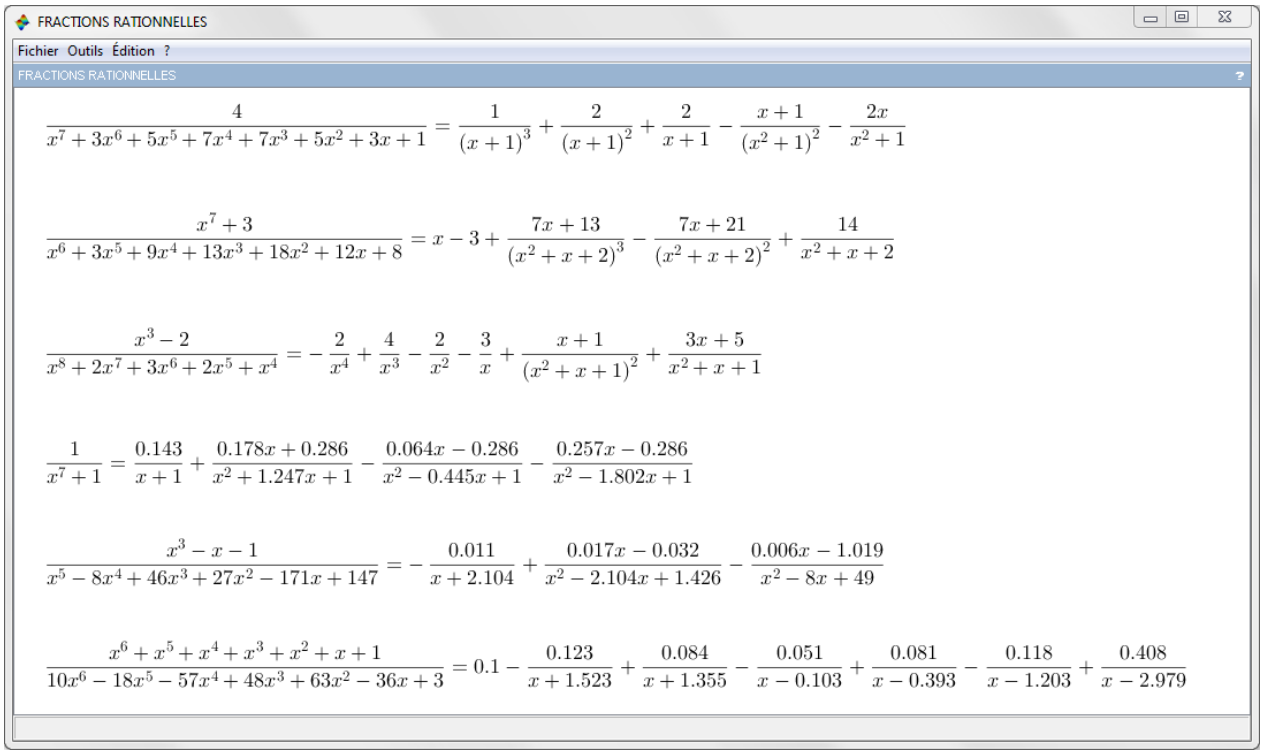
```
// Procédure interactive
function FRACTION()
    x = poly(0,'x')
    init = ''
    xstr = 0
    ystr = 0
    hstr = 0.5
    fontId = 2
    fontSz = 4
    X = getsystemmetrics("SM_CXSCREEN")
    Y = getsystemmetrics("SM_CYSCREEN")
    f = scf(1)
    f.figure_name = 'FRACTIONS RATIONNELLES'
    f.figure_position = [0,0]
    f.figure_size = [2*X/3,3*Y/4]
    toolbar(1,'off')
    a = gca()
    a.data_bounds = [0,-1;1,0]
    a.x_location = 'top'
    a.y_location = 'left'
    a.axes_visible = 'off'
    while %t
        resp = x_dialog('Fraction rationnelle à décomposer en éléments simples :',init)
        if resp == [] then, break, else, resp = strcat(resp,' '), end
        init = resp
        err = execstr('F = evstr(resp)','errcatch')
        check = (err == 0) & (typeof(F) == 'rational')
        if check then
            A = coeff(F.num)
            B = coeff(F.den)
            check = isreal(A) & isreal(B)
        end
        if check then
            str = rf(F)
            rect = xstringl(0,0,str,fontId,fontSz)
            h = rect(4)
            ystr = ystr - h
            xstring(xstr,ystr,str)
            e = gce()
            e.font_style = fontId
            e.font_size = fontSz
            ystr = ystr – h*hstr
            next = %t
        else
            messagebox('Saisir une fraction rationnelle réelle de la variable x, au format Scilab.',...
                       'Erreur','error','Correction','modal')
            next = %f
        end
        if next then, init = '', end
    end
endfunction


// Programme principal
FRACTION()
```

Mathématique & Informatique







$$\frac{4}{x^7+3x^6+5x^5+7x^4+7x^3+5x^2+3x+1} = \frac{1}{(x+1)^3} + \frac{2}{(x+1)^2} + \frac{2}{x+1} - \frac{x+1}{(x^2+1)^2} - \frac{2x}{x^2+1}$$

$$\frac{x^7+3}{x^6+3x^5+9x^4+13x^3+18x^2+12x+8} = x - 3 + \frac{7x+13}{(x^2+x+2)^3} - \frac{7x+21}{(x^2+x+2)^2} + \frac{14}{x^2+x+2}$$

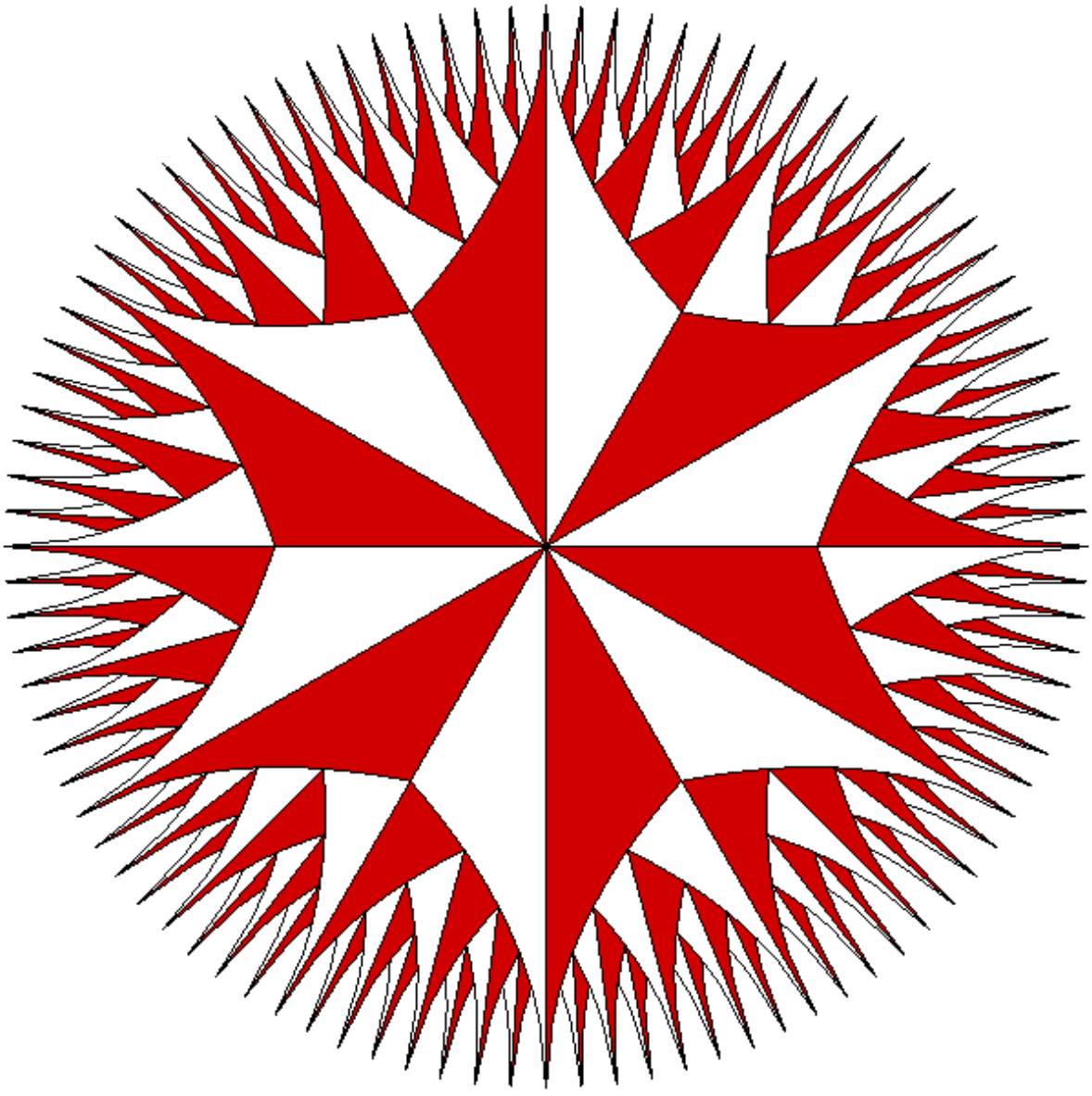
$$\frac{x^3-2}{x^8+2x^7+3x^6+2x^5+x^4} = -\frac{2}{x^4} + \frac{4}{x^3} - \frac{2}{x^2} - \frac{3}{x} + \frac{x+1}{(x^2+x+1)^2} + \frac{3x+5}{x^2+x+1}$$

$$\frac{1}{x^7+1} = \frac{0.143}{x+1} + \frac{0.178x+0.286}{x^2+1.247x+1} - \frac{0.064x-0.286}{x^2-0.445x+1} - \frac{0.257x-0.286}{x^2-1.802x+1}$$

$$\frac{x^3-x-1}{x^5-8x^4+46x^3+27x^2-171x+147} = -\frac{0.011}{x+2.104} + \frac{0.017x-0.032}{x^2-2.104x+1.426} - \frac{0.006x-1.019}{x^2-8x+49}$$

$$\frac{x^6+x^5+x^4+x^3+x^2+x+1}{10x^6-18x^5-57x^4+48x^3+63x^2-36x+3} = 0.1 - \frac{0.123}{x+1.523} + \frac{0.084}{x+1.355} - \frac{0.051}{x-0.103} + \frac{0.081}{x-0.393} - \frac{0.118}{x-1.203} + \frac{0.408}{x-2.979}$$

Mathématique & Informatique



Mathématique & Informatique

⇨ Les intégrales

Intégrale simple : $\int_D f(x)dx$

Fonction Scilab *integrate*.
Le domaine *D* d'intégration est un ensemble de segments.

Par exemple, calculer $\int_D x^2\,dx$ où *D* est le segment défini par les 2 points 0 et *a*.
Pour *a* = 1, on écrit :

```
a = 1 ;
I = integrate('x^2','x',0,a)
```

Et on obtient :

```
I = 0.3333333
```

Note La valeur exacte est $\int_D x^2\,dx = \frac{a^3}{3} = \frac{1}{3}$

Intégrale double : $\iint_D f(x,y)dxdy$

Fonction Scilab *int2d*.
Le domaine *D* d'intégration est un ensemble de triangles.

Par exemple, calculer $\iint_D (x^2+y^2)dxdy$ où *D* est le triangle défini par les 3 points (0,0), (*a*,0), (0,*b*).
Pour *a* = 1, *b* = 2 on écrit :

```
a = 1 ; b = 2 ;
X = [0;a;0] ; Y = [0;0;b] ;
deff('z = f(x,y)','z = x^2 + y^2') ;
[I,e] = int2d(X,Y,f)
```

Et on obtient :

```
I = 0.8333333
```

Note La valeur exacte est $\iint_D (x^2+y^2)dxdy = \frac{ab(a^2+b^2)}{12} = \frac{5}{6}$

Mathématique & Informatique

Intégrale triple : $\iiint_D f(x,y,z)\,dx\,dy\,dz$

Fonction Scilab *int3d*.
Le domaine *D* d'intégration est un ensemble de tétraèdres.

Par exemple, calculer $\iiint_D (x^2+y^2+z^2)\,dx\,dy\,dz$ où *D* est le tétraèdre défini par les 4 points (0,0,0), (*a*,0,0), (0,*b*,0), (0,0,*c*).
Pour $a = 1$, $b = 2$, $c = 3$ on écrit :

```
a = 1 ; b = 2 ; c = 3 ;
X = [0;a;0;0] ; Y = [0;0;b;0] ; Z = [0;0;0;c] ;
deff('d = f(v,n)','x = v(1) ; y = v(2) ; z = v(3) ; d = x^2 + y^2 + z^2') ;
[I,e] = int3d(X,Y,Z,f)
```

Et on obtient : I = 1.4

Note La valeur exacte est $\iiint_D (x^2+y^2+z^2)\,dx\,dy\,dz = \dfrac{abc\,(a^2+b^2+c^2)}{60} = \dfrac{7}{5}$

Intégrale de Cauchy : $\int_D f(z)\,dz$

Fonction Scilab *intc*.
Le domaine *D* d'intégration est un segment de droite.

Par exemple, calculer $\int_D (z+1)\,dz$ où *D* est le segment défini par les 2 points d'affixes $5-i$ et $(2\sqrt{2}+1)+(2\sqrt{2}-1)i$

On écrit : *deff('Z = f(z)','Z = z + 1') ; I = intc(5-%i,2*sqrt(2)+1+(2*sqrt(2)-1)*%i,f)*
Et on obtient : I = - 7.5147186 + 14.828427i
La valeur exacte est $(-16+6\sqrt{2})+(12+2\sqrt{2})i$

Fonction Scilab *intl*.
Le domaine *D* d'intégration est un arc de cercle.

Par exemple, calculer $\int_D (z+1)\,dz$ où *D* est l'arc de cercle de centre $1-i$, de rayon 4, d'angle compris entre 0 et $\pi/4$.

On écrit : *deff('Z = f(z)','Z = z + 1') ; I = intl(0,%pi/4,1-%i,4,f)*
Et on obtient : I = - 7.5147186 + 14.828427i
La valeur exacte est $(-16+6\sqrt{2})+(12+2\sqrt{2})i$

Note La fonction $f(z) = z + 1$ étant holomorphe sur $\mathbb{C}$, l'intégrale $\int_D (z+1)\,dz$ ne dépend que des extrémités $A = 5-i$ et $B = (2\sqrt{2}+1)+(2\sqrt{2}-1)i$ du chemin *D*, d'où la même valeur de I obtenue avec *intc* et *intl* dans les 2 exemples précédents.

Remarque

L'intégration numérique a pour avantage de ne pas nécessiter la recherche de primitives, ni même l'expression analytique de la fonction *f* dans le cas où les données $f(x_i)$ sont issues de mesures. Parmi les nombreuses méthodes, citons la méthode de Monte-Carlo qui permet d'évaluer une intégrale multiple à partir des valeurs prisent par la fonction *f* en des points aléatoires de son domaine d'intégration (approche probabiliste). Pour toutes les méthodes, on augmente le nombre de points de calcul afin d'obtenir, en principe, une plus grande précision des résultats.

Exemple d'utilisation de la fonction Scilab *integrate*

I = *integrate*(*f*,'*t*',*a*,X,precision)

integrate permet d'intégrer numériquement une fonction *f* d'une variable sur un ensemble de segments $[a,x_k]$.
f est donnée sous la forme d'une chaîne de caractères (entre quotes), '*t*' est le nom de la variable d'intégration,
a est une constante, $X = [x_1 , x_2 , \dots , x_n]$ et $I = [I_1 , I_2 , \dots , I_n]$ sont des vecteurs de *n* valeurs. On a :

$$I_k = \int_a^{x_k} f(t)\,dt \qquad (1 \le k \le n)$$

Par exemple, traçons le graphe de la fonction φ définie par : $\varphi(x) = \int_{1/x}^{x^2} \frac{dt}{\sqrt{1+t^3}}$ $\qquad x \in]-\infty , -1] \cup [0 , +\infty[$

Pour utiliser la fonction Scilab *integrate*, mettons φ(*x*) sous la forme : $\varphi(x) = \int_0^{x^2} f(t)\,dt - \int_0^{1/x} f(t)\,dt$

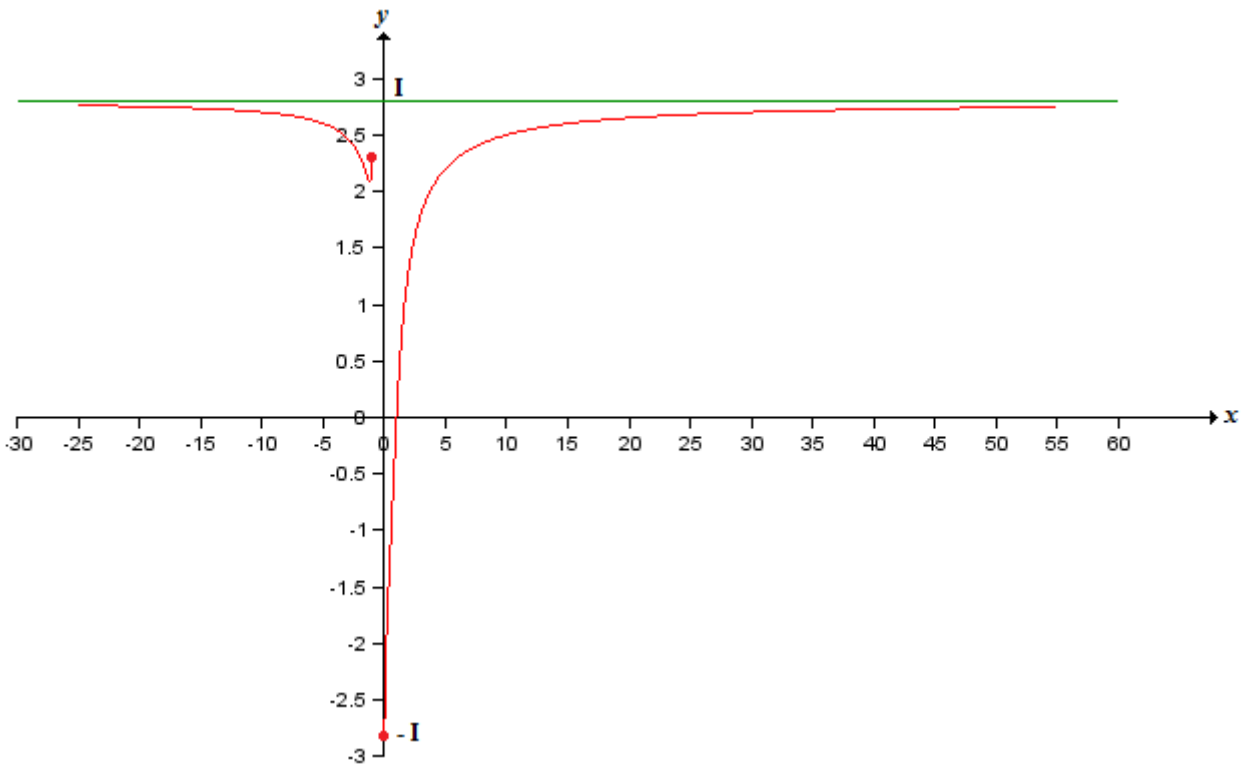
On séparera les intervalles de discrétisation de la variable *x* : $x \in [x_{min} , -1 - \varepsilon]$ et $x \in [\varepsilon , x_{max}]$.

On évaluera aussi l'intégrale $I = \int_0^{+\infty} f(t)\,dt$ afin de tracer l'asymptote horizontale $y = \lim_{x \to \pm\infty} \varphi(x) = I$.

Programme :

```
f = '1/sqrt(1 + t^3)' ;
x1 = linspace(-25,-1-1e-6,300) ;
y1 = integrate(f,'t',0,x1^2,1e-6) - integrate(f,'t',0,x1^(-1),1e-6) ;
x2 = linspace(1e-6,55,500) ;
y2 = integrate(f,'t',0,x2^2,1e-6) - integrate(f,'t',0,x2^(-1),1e-6) ;
I = integrate(f,'t',0,50e3) ;
a = gca() ; a.x_location = 'origin' ; a.y_location = 'origin' ;
plot2d(x1,y1,5) ; plot2d(x2,y2,5)
plot2d([-30,60],[I,I],13)
```

On obtient le graphe ci-dessous :



Exemple d'utilisation de la fonction Scilab *int2d*

[I,err] = *int2d*(X,Y,*f*,precision)

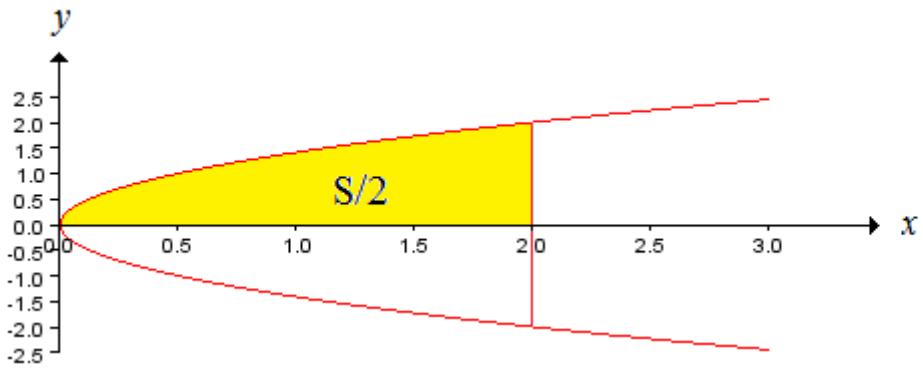
int2d permet d'intégrer numériquement une fonction *f* de 2 variables sur une surface plane S définie comme une somme de triangles dont les sommets ont pour matrices de coordonnées X, Y. Supposons que la surface S soit divisée en *n* triangles :

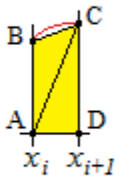
X est une matrice (3,*n*) : X(*i*,*j*) = abscisse du i^e sommet du j^e triangle constituant la surface S
Y est une matrice (3,*n*) : Y(*i*,*j*) = ordonnée du i^e sommet du j^e triangle constituant la surface S

Recherchons, par exemple, une valeur approchée de l'intégrale double :

$$I = \iint_S \frac{dxdy}{\left(1+\frac{x^2}{4}+\frac{y^2}{2}\right)^2} \qquad S = \left\{(x,y)\,\middle|\, 0 \le x \le 2\,,\, y^2 \le 2x\right\}$$

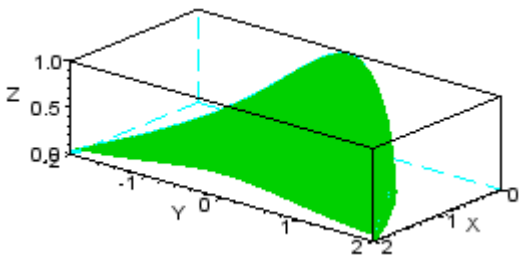
La surface S est le domaine fini délimité par la parabole $y = \pm\sqrt{2x}$ et la verticale $x = 2$.





Par symétrie, on restreindra le calcul avec *int2d* au domaine S/2 des $y \ge 0$, puis on multipliera le résultat obtenu par 2.
La surface S/2 ne peut pas être scindée en une somme exacte de triangles. En conséquence, on effectuera une approximation du domaine d'intégration en le divisant en 2(N – 1) petits triangles issus de la discrétisation de la variable *x* en N points sur [0,2].
Le programme de calcul de I est le suivant :

```
function z = f(x,y)
    z = 1/(1 + x^2/4 + y^2/2)^2
endfunction
N = 10000 ;
x = linspace(0,2,N) ;
X = [] ; Y = [] ;
for i = 1:N-1
    A = [x(i),0] ;
    B = [x(i),sqrt(2*x(i))] ;
    C = [x(i+1),sqrt(2*x(i+1))] ;
    D = [x(i+1),0] ;
    X = [X,[A(1);B(1);C(1)],[A(1);C(1);D(1)]] ;
    Y = [Y,[A(2);B(2);C(2)],[A(2);C(2);D(2)]] ;
end
[I,err] = int2d(X,Y,f,[1e-6,1,20000,2000000,1]) ;
I = 2*I ;
printf('\n I = %.6f , err = %e',I,err)
```



On trouve :

I = 2.141591 avec une erreur absolue de l'ordre de 10^{-6}

Mathématique & Informatique

Exemple d'utilisation de la fonction Scilab *int3d*

[I,err] = *int3d*(X,Y,Z,*F*,*nf*,precision)

int3d permet d'intégrer numériquement une ou plusieurs fonctions *F* de 3 variables à l'intérieur d'un même volume V défini comme une somme de tétraèdres dont les sommets ont pour matrices de coordonnées X, Y, Z.
A titre d'exemple, considérons un solide homogène en forme de pyramide régulière de hauteur *h* dont la base est un polygone à *n* côtés inscrit dans un cercle de rayon *R*. Le moment d'inertie de ce solide par rapport à l'axe passant par le centre O de sa base et son sommet S est $I = kMR^2$ où *M* est la masse du solide et *k* est un coefficient ne dépendant que de *n*. On montre que :

$$k = \frac{\iiint_V (x^2 + y^2)\,dx\,dy\,dz}{\iiint_V dx\,dy\,dz}$$

Le volume V d'intégration est la pyramide unité correspondant à $R = h = 1$. La quantité $x^2 + y^2$ est le carré de la distance d'un point quelconque de cette pyramide à l'axe O*z* du repère O*xyz* positionné au centre O de la base et passant par le sommet S.
Divisons le volume V en *n* tétraèdres (O, P_j, P_{j-1}, S), $1 \le j \le n$, dont les coordonnées des sommets sont :

$$O(0,0,0) \qquad P_j\left(\cos\frac{2j\pi}{n}, \sin\frac{2j\pi}{n}, 0\right) \qquad P_{j-1}\left(\cos\frac{2(j-1)\pi}{n}, \sin\frac{2(j-1)\pi}{n}, 0\right) \qquad S(0,0,1)$$

Définissons les 3 paramètres X, Y, Z de la fonction *int3d* :

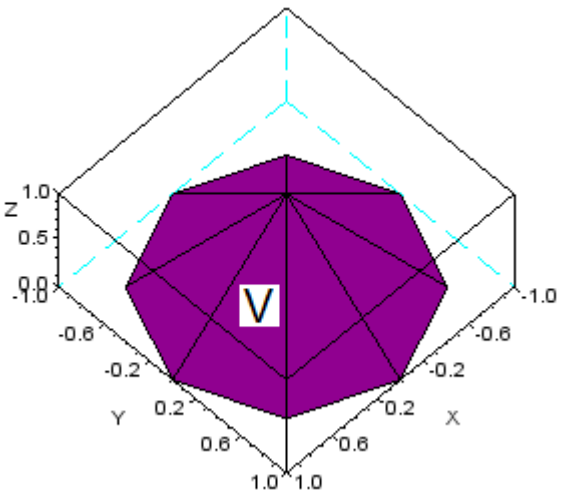
X est une matrice (4,*n*) : X(*i*,*j*) = abscisse du i^e sommet du j^e tétraèdre constituant la pyramide unité
Y est une matrice (4,*n*) : Y(*i*,*j*) = ordonnée du i^e sommet du j^e tétraèdre constituant la pyramide unité
Z est une matrice (4,*n*) : Z(*i*,*j*) = cote du i^e sommet du j^e tétraèdre constituant la pyramide unité

On a *nf* = 2 fonctions f_1 et f_2 à intégrer dans V :

$f_1(x,y,z) = 1$
$f_2(x,y,z) = x^2 + y^2$

Le programme de calcul de *k* est le suivant :

```
function f = F(xyz,nf)
    x = xyz(1) ; y = xyz(2) ; z = xyz(3)
    f(1) = 1
    f(2) = x^2 + y^2
endfunction
n = input('n?') ;
X = [] ; Y = [] ; Z = [] ;
for j = 1:n
    X = [X,[0;cos(2*j*%pi/n);cos(2*(j-1)*%pi/n);0]] ;
    Y = [Y,[0;sin(2*j*%pi/n);sin(2*(j-1)*%pi/n);0]] ;
    Z = [Z,[0;0;0;1]] ;
end
plot3d(X(2:4,:),Y(2:4,:),Z(2:4,:),flag = [24,6,4]) ;
[I,err] = int3d(X,Y,Z,F,2,[0,1000,1e-14,1e-13]) ;
k = I(2)/I(1) ;
printf('\n k = %.15f , err = %e',k,err)
```



On trouve, pour *n* = 8 (par exemple) :

k = 0.27071067811865 avec une erreur absolue de l'ordre de 10^{-14}

Remarque : quand n → + ∞, la pyramide tend vers un cône et $k \to \frac{3}{10}$

Exemple d'utilisation des fonctions Scilab *intc* et *intl*

I = *intc*(z_1,z_2,*f*)
I = *intl*(θ_1,θ_2,z_0,r,*f*)

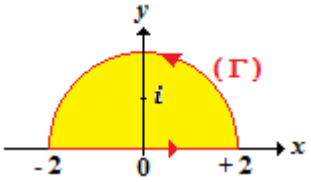
intc et *intl* permettent d'intégrer numériquement une fonction *f* de la variable complexe le long d'un chemin Γ :

$$I = \int_{\Gamma} f(z)\,dz$$

Pour *intc*, Γ est le segment de droite dont les extrémités ont pour affixes z_1 et z_2.
Pour *intl*, Γ est l'arc de cercle dont le centre a pour affixe z_0 , le rayon a pour valeur r, l'angle est compris entre θ_1 et θ_2 radians.
Dans le cas général I est un nombre complexe.

Par exemple, recherchons :

$$I = \int_{\Gamma} \frac{dz}{(z^2+1)^5}$$



Le contour fermé (Γ) est la réunion du diamètre (D) dont les extrémités sont les points de coordonnées (-2,0) et (2,0) avec le demi-cercle (C) de centre (0,0) et de rayon 2 : (Γ) = (D) ∪ (C)

Posons $f(z) = \frac{1}{(z^2+1)^5}$, on a : $I = \int_{\Gamma} f(z)dz = \int_{D} f(z)dz + \int_{C} f(z)dz$

Le calcul numérique de I s'écrit :

```
I = intc(-2,2,f) + intl(0,%pi,0,2,f)
```

Avec :

```
function Z = f(z)
    Z = 1/(z^2 + 1)^5
endfunction
```

Résultat :

I = 0.8590292 - 1.408D-18i

Remarque

La fonction *f*(*z*) admet 2 pôles : $z_1 = i$ et $z_2 = -i$. Par suite, *f* est holomorphe dans le domaine intérieur à (Γ) sauf au point z_1.

Le théorème des résidus s'applique : $\int_{\Gamma} f(z)dz = 2i\pi \,\mathrm{Res}(f, z_1)$

Le résidu de *f* en z_1 est le coefficient du terme en $\frac{1}{z-z_1}$ dans la décomposition en éléments simples sur $\mathbb{C}$ de la fraction rationnelle $\frac{1}{(z^2+1)^5}$:

$$\mathrm{Res}(f,i) = \frac{1}{4!}\left\{\frac{d^4}{dz^4}\left[\frac{(z-i)^5}{(z^2+1)^5}\right]\right\}_{z=i} = \frac{1}{4!}\left\{\frac{d^4}{dz^4}(z+i)^{-5}\right\}_{z=i} = \frac{(-5)(-6)(-7)(-8)(2i)^{-9}}{4!} = \frac{35}{256i}$$

D'où la valeur exacte de I : $I = 2i\pi \times \frac{35}{256i} = \frac{35\pi}{128}$

Mathématique & Informatique

Intégrale impropre d'une fraction rationnelle

Il s'agit d'effectuer l'approximation numérique de l'intégrale :

$$I = \int_{-\infty}^{+\infty} \frac{P(x)}{Q(x)}\,dx$$

où $P(x)$ et $Q(x)$ sont 2 polynômes à coefficients réels vérifiant les 2 conditions suivantes :

1) $Q(x)$ n'a aucune racine réelle
2) degré(P(x)) ≤ degré(Q(x)) – 2

Application : $$\int_{-\infty}^{+\infty} \frac{274x^9 + 411x^8 - 1233x^7 + 2x^2 + 3x - 9}{(x^2 - x + 7)^3\,(x^6 + x + 1)}\,dx = ?$$

Méthode 1

Par définition : $$\int_{-\infty}^{+\infty} \frac{P(x)}{Q(x)}dx = \lim_{\substack{a\to-\infty\\ b\to+\infty}} \int_a^b \frac{P(x)}{Q(x)}dx = \lim_{\substack{a\to-\infty\\ b\to+\infty}} \int_a^b f(x)\,dx$$

On utilise la fonction Scilab *integrate* avec deux grandes valeurs $a < 0$ et $b > 0$ en espérant l'approximation satisfaisante :

I = integrate(f, 'x', a, b)

Les conditions imposées à $P(x)$ et $Q(x)$ garantissent la convergence de l'intégrale, d'où : $$\int_{-\infty}^{+\infty} \frac{P(x)}{Q(x)}dx = \lim_{c\to+\infty} \int_{-c}^{c} f(x)\,dx$$

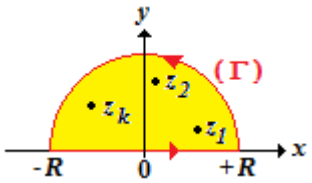
Donc, pour évaluer I, on peut prendre $a = -c$ et $b = c$ comme paramètres de la fonction *integrate* :

I = integrate(f, 'x', - c, c)

mais on ne connait pas, a priori, la valeur à donner à la constante $c > 0$ pour obtenir la précision souhaitée sur I.

Méthode 2

Introduisons l'intégrale de **Cauchy** $\displaystyle\int_\Gamma \frac{P(z)}{Q(z)}dz$



Posons $f(z) = \dfrac{P(z)}{Q(z)}$, on a : $$\int_\Gamma f(z)dz = \int_{-R}^{+R} f(x)dx + \int_C f(z)dz$$

où C est le demi-cercle de centre 0 et de rayon R parcouru dans le sens trigonométrique.

Le théorème des résidus s'écrit : $$\int_\Gamma f(z)dz = 2i\pi \sum_{k=1}^{p} \mathrm{Res}(f, z_k)$$ où les z_k sont les pôles de $f(z)$ à l'intérieur de Γ.

Si l'on prend R suffisamment grand pour que tous les pôles de la fraction $f(z)$, situés dans le demi-plan Im(z) > 0, soient à l'intérieur du demi-disque bordé par Γ, alors l'intégrale $\displaystyle\int_\Gamma f(z)dz$ ne dépend pas du choix de R puisqu'elle est égale à une somme (constante) de résidus.

Supposons que $Q(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$. On sait [*] que toute racine x_0 de $Q(x)$ vérifie en module :

$$|x_0| < 1 + \frac{\underset{i=0,\ldots,n}{\text{Max}}(|a_i|)}{|a_n|}$$

D'autre part, en faisant tendre R vers l'infini :

$$\int_\Gamma f(z)dz = \int_{-\infty}^{+\infty} f(x)dx + \lim_{R\to\infty}\int_C f(z)dz$$

Comme $\lim_{z\to\infty} zf(z) = 0$ puisque d°P + 1 ≤ d°Q – 1 < d°Q, on a : $\lim_{R\to\infty}\int_C f(z)dz = 0$ d'après un lemme de **Jordan**.

Finalement :

$$\int_{-\infty}^{+\infty}\frac{P(x)}{Q(x)}dx = \int_\Gamma \frac{P(z)}{Q(z)}dz \text{ où } \Gamma \text{ est le bord du demi-disque de rayon } R = 1 + \frac{\underset{i=0,\ldots,n}{\text{Max}}(|a_i|)}{|a_n|}$$

Note

Généralement on dispose d'une fonction qui permet de calculer les n racines complexes x_i (conjuguées 2 à 2) de $Q(x)$ et il est possible d'attribuer à R une valeur plus petite : $R = \underset{i=1,\ldots,n}{\text{Max}}\{|x_i|; Q(x_i)=0\} + \varepsilon$ avec $\varepsilon > 0$.

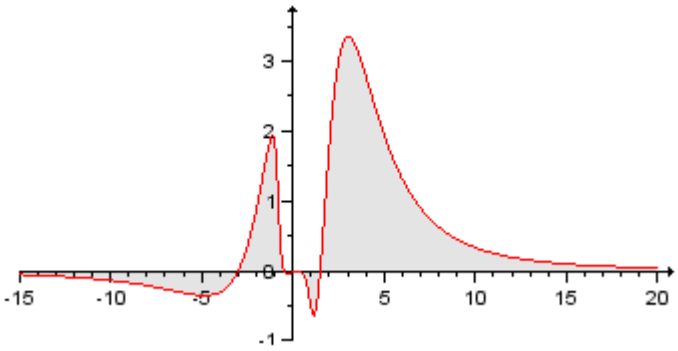
Programmation

Soient P et Q deux polynômes Scilab. La fonction de calcul de $I = \int_{-\infty}^{+\infty}\frac{P(x)}{Q(x)}dx$ est la suivante :

```
function I = integrale(P,Q)
    function Z = f(z)
        Z = horner(P,z)/horner(Q,z)
    endfunction
    I = []
    if find(imag(roots(Q)) == 0) <> [] then, return, end
    if degree(P) > degree(Q) - 2 then, return, end
    R = 1 + max(abs(coeff(Q)))/abs(coeff(Q,degree(Q)))
    I = real(intc(-R,R,f) + intl(0,%pi,0,R,f))
endfunction
```

Pour obtenir la valeur de l'intégrale particulière proposée, on écrit

```
x = poly(0,'x') ;
P = 274*x^9 + 411*x^8 - 1233*x^7 + 2*x^2 + 3*x - 9 ;
Q = ((x^2 - x + 7)^3).*(x^6 + x + 1) ;
I = integrale(P,Q)
```



Et on trouve :

$$\int_{-\infty}^{+\infty}\frac{274x^9 + 411x^8 - 1233x^7 + 2x^2 + 3x - 9}{(x^2 - x + 7)^3(x^6 + x + 1)}dx \approx 14.096192$$

Mathématique & Informatique

(*) Majorant et minorant des racines d'un polynôme à coefficients réels

$$A(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \qquad (a_i \in \mathbb{R}, a_n \neq 0, a_0 \neq 0)$$

Notons : $M = \operatorname{Max}(|a_0|, |a_1|, \cdots, |a_{n-1}|)$

Rappelons l'inégalité triangulaire valable dans $\mathbb{C}$: $\forall (z_1, z_2) \quad |z_1| - |z_2| \leq |z_1 + z_2| \leq |z_1| + |z_2|$

Exprimons $|A(x)|$:

$$\begin{aligned}
|A(x)| &= \left| a_n x^n + (a_{n-1}x^{n-1} + \cdots + a_1 x + a_0) \right| \\
&\geq \left| a_n x^n \right| - \left| a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \right| \\
&\geq \left| a_n x^n \right| - \left(\left| a_{n-1}x^{n-1} \right| + \cdots + |a_1 x| + |a_0| \right) \\
&\geq |a_n||x|^n - M\left(|x|^{n-1} + \cdots + |x| + 1 \right) \\
&= |a_n||x|^n - M\left(\frac{|x|^n - 1}{|x| - 1} \right) \\
&= \left(|a_n| - \frac{M}{|x| - 1} \right)|x|^n + \frac{M}{|x| - 1}
\end{aligned}$$

Si $|x| \geq 1 + \frac{M}{|a_n|}$ alors $|a_n| - \frac{M}{|x| - 1} \geq 0$ et $\frac{M}{|x| - 1} > 0$, d'où $|A(x)| > 0$ et par conséquent $A(x) \neq 0$.

Donc :

$$|x| \geq 1 + \frac{\operatorname{Max}(|a_0|, |a_1|, \cdots, |a_{n-1}|)}{|a_n|} \quad \Rightarrow \quad x \text{ n'est pas une racine de } A(x)$$

La contraposée s'écrit :

$$x \text{ est une racine de } A(x) \quad \Rightarrow \quad |x| < 1 + \frac{\operatorname{Max}(|a_0|, |a_1|, \cdots, |a_{n-1}|)}{|a_n|}$$

Posons $x = \frac{1}{y}$, on a :

$$A(x) = \frac{1}{y^n} B(y) \text{ où } B(y) = a_0 y^n + a_1 y^{n-1} + \cdots + a_{n-1} y + a_n$$

Appliquons le résultat précédent au polynôme $B(y)$:

$$y \text{ est une racine de } B(y) \quad \Rightarrow \quad |y| < 1 + \frac{\operatorname{Max}(|a_1|, |a_2|, \cdots, |a_n|)}{|a_0|}$$

On en déduit :

$$x \text{ est une racine de } A(x) \quad \Rightarrow \quad |x| > \frac{1}{1 + \dfrac{\operatorname{Max}(|a_1|, |a_2|, \cdots, |a_n|)}{|a_0|}}$$

En regroupant les inégalités :

$$x \text{ est une racine de } A(x) \quad \Rightarrow \quad \frac{1}{1 + \dfrac{\operatorname{Max}(|a_1|, |a_2|, \cdots, |a_n|)}{|a_0|}} < |x| < 1 + \frac{\operatorname{Max}(|a_0|, |a_1|, \cdots, |a_{n-1}|)}{|a_n|}$$

A fortiori :

$$x \text{ est une racine de } A(x) \quad \Rightarrow \quad \frac{1}{1 + \dfrac{\operatorname{Max}_{i=0,\ldots,n}(|a_i|)}{|a_0|}} < |x| < 1 + \frac{\operatorname{Max}_{i=0,\ldots,n}(|a_i|)}{|a_n|}$$

Mathématique & Informatique

⇨ Les nombres aléatoires

Les fonctions *rand* et *grand* fournissent des matrices de nombres aléatoires obéissant à différentes lois de probabilité.
En particulier, pour obtenir une matrice A(p,q) de nombres aléatoires uniformément répartis sur [0,1], on écrit :
A = rand(p, q, 'uniform') ou mieux A = grand(p, q, 'def') avec le générateur Mersenne Twister de période $2^{19937}- 1$

APPLICATION : Calcul des intégrales multiples par la **méthode de Monte-Carlo**

Le domaine d'intégration D d'une intégrale multiple de dimension m peut être ramené à un domaine Δ intérieur à un hypercube unité de dimension m en utilisant la transformation suivante :
$\forall\ k = 1, \dots, m : \quad a_k \le x_k \le b_k \quad \Rightarrow \quad x_k = a_k + (b_k - a_k)\,\xi_k \quad , \quad 0 \le \xi_k \le 1$

$$Ainsi : \quad I = \iint\!\dots\!\int_D f(x_1, x_2, \dots, x_m)\,dx_1\,dx_2 \dots dx_m = \iint\!\dots\!\int_\Delta F(\xi_1, \xi_2, \dots, \xi_m)\,d\xi_1\,d\xi_2 \dots d\xi_m \qquad (1)$$

$$où \quad F(\xi_1, \xi_2, \dots, \xi_m) = \left[\prod_{k=1}^{m}(b_k - a_k)\right] f(a_1+(b_1-a_1)\xi_1,\ a_2+(b_2-a_2)\xi_2,\ \dots,\ a_m+(b_m-a_m)\xi_m)$$

En générant N points aléatoires $P(\xi_1, \xi_2, \dots, \xi_m)$ à l'intérieur de l'hypercube unité, un nombre n d'entre eux sont situés dans Δ. Le produit de la valeur moyenne de F ($\approx \frac{1}{n} \Sigma\, F(\xi)$) par l'hypervolume $\Delta = Prob(\ P \in \Delta\ ) \approx n/N$, donne :

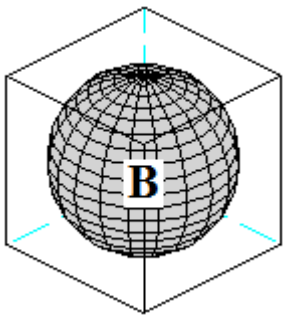
$$I \approx \frac{1}{N}\sum_{i=1}^{n} F(\xi_1^{(i)}, \xi_2^{(i)}, \dots, \xi_m^{(i)}) \qquad (2)$$

La formule d'approximation (2) est facile à calculer, ce qui n'est pas le cas, en général, de la formule (1).
Le nombre N doit être très grand pour que les lois statistiques se manifestent pleinement et que le résultat obtenu soit proche de la valeur exacte. La précision dépend également de la qualité du générateur de nombres pseudo-aléatoires.

Exemple 1

L'intégrale triple $I = \iiint_{B(0,1)} (x_1^2 + x_2^2 + x_3^2)^{1/2}\, dx_1\, dx_2\, dx_3$ dans la boule de centre 0 et de rayon 1 se calcule à l'aide des coordonnées sphériques. On trouve : $I = \pi = 3.1415926536$
Le changement de variable $\xi_i = \frac{1}{2}\, x_i + \frac{1}{2}$ conduit à intégrer dans la boule B de centre $c(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ et de rayon $r = \frac{1}{2}$
On obtient : $I = \iiint_{B(c,r)} 8\ [\ (2\xi_1 - 1)^2 + (2\xi_2 - 1)^2 + (2\xi_3 - 1)^2\ ]^{1/2}\, d\xi_1\, d\xi_2\, d\xi_3$
L'instruction P = grand(1, 3, 'def') génère 1 point aléatoire P à l'intérieur du cube unité. Pour vérifier si ce point appartient à la boule B, il suffit d'effectuer le test : $cP \le r \Leftrightarrow [P(1) - \frac{1}{2}]^2 + [P(2) - \frac{1}{2}]^2 + [P(3) - \frac{1}{2}]^2 \le \frac{1}{4}$
Programme Scilab :

```
N = input('N ?')
s = 0 ;
for i = 1:N
   P = grand(1, 3, 'def') ;
   if sum((P - 1/2)^2) <= 1/4 then
      s = s + 8*sqrt(sum((2*P - 1)^2)) ;
   end
end
I = s/N , e = abs(I/%pi - 1)*100
```



Résultats obtenus :

N	I	% d'erreur
1000	3.2879434	4.6584876
10 000	3.1708621	0.9316756
100 000	3.1413087	0.0090385
1 000 000	3.1446727	0.0980396
10 000 000	3.1415368	0.0017789

NB. Le programme ci-dessus n'est pas de type déterministe puisque chaque exécution fournit un résultat différent.
L'inégalité de **Tchebychev** assure la convergence en probabilité : $Prob(\ |\ I_N - I\ |\ <\ \varepsilon\ ) = 1$, quand $N \to \infty$

Exemple 2

Recherchons une approximation numérique de l'intégrale double I suivante :

$$\iint \frac{dx_1\, dx_2}{(1 + x_1^2/4 + x_2^2/2)^2} \qquad \text{dans le domaine : } 0 \le x_1 \le 2 \text{ et } x_2^2 \le 2\,x_1$$

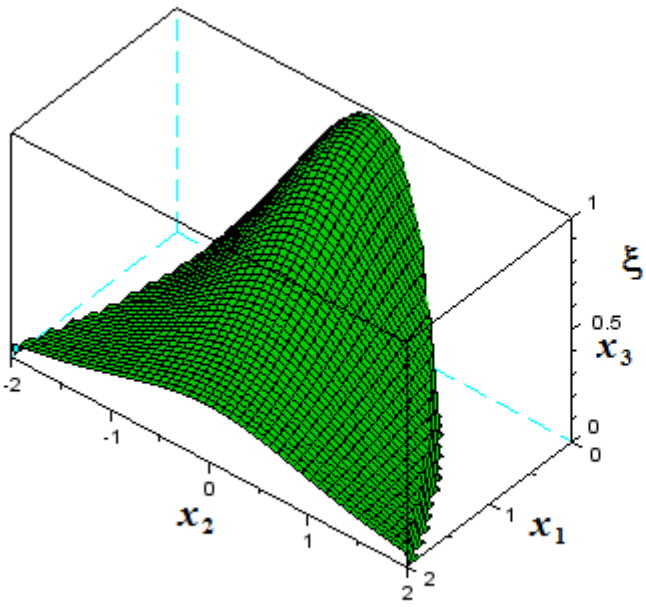
Le changement de variable $\xi_1 = \frac{1}{2}\,x_1$, $\xi_2 = \frac{1}{4}\,x_2 + \frac{1}{2}$ ramène le domaine d'intégration à l'intérieur du carré unité :

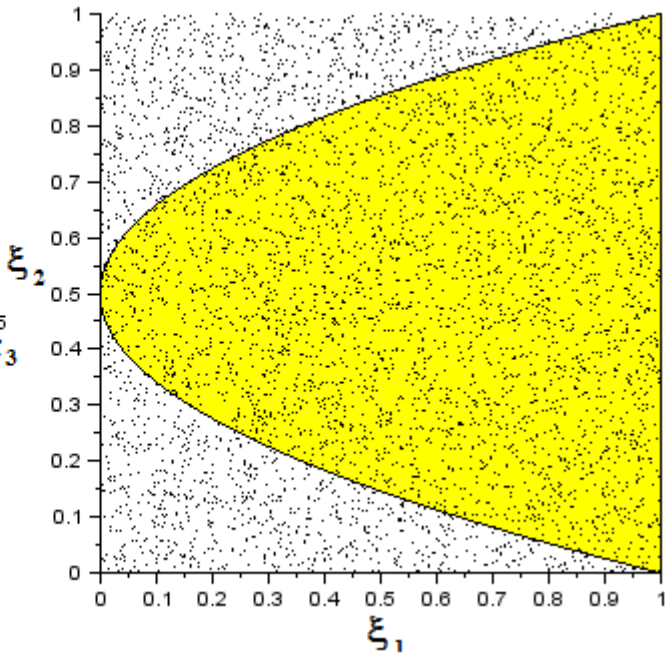
$$\iint \frac{8\, d\xi_1\, d\xi_2}{(3 + \xi_1^2 + 8\,\xi_2^2 - 8\,\xi_2)^2} \qquad \text{dans le domaine : } 0 \le \xi_1 \le 1 \text{ et } (2\xi_2 - 1)^2 \le \xi_1$$

Appliquons la méthode de Monte-Carlo limitée à N = 5000 points de calcul :

```
N = 5000 ;
s = 0 ;
for i = 1:N
   P = grand(1, 2, 'def') ;
   if (2*P(2) - 1)^2 <= P(1) then
      s = s + 8/(3 + P(1)^2 + 8*P(2)^2 - 8*P(2))^2 ;
   end
end
I = s/N
```

Nous obtenons : I ≈ **2.1479156.** La valeur exacte est : I = π - 1 = 2.1415927





En initialisant le générateur de nombres aléatoires Mersenne Twister avec l'instruction *grand('setsd', 0)*, on obtient toujours la même approximation de l'intégrale I, sinon elle varie d'une exécution à une autre.

Exemple 3

Dans le cas m = 1, l'intégrale simple sur [*a*,*b*] est transformée en intégrale simple sur le segment unité :

$$I = \int_a^b f(x)dx = (b-a)\int_0^1 f(a+(b-a)\xi)d\xi$$

Tous les points P(ξ) appartiennent à [0,1], donc n = N et la méthode de Monte-Carlo consiste à calculer la somme :

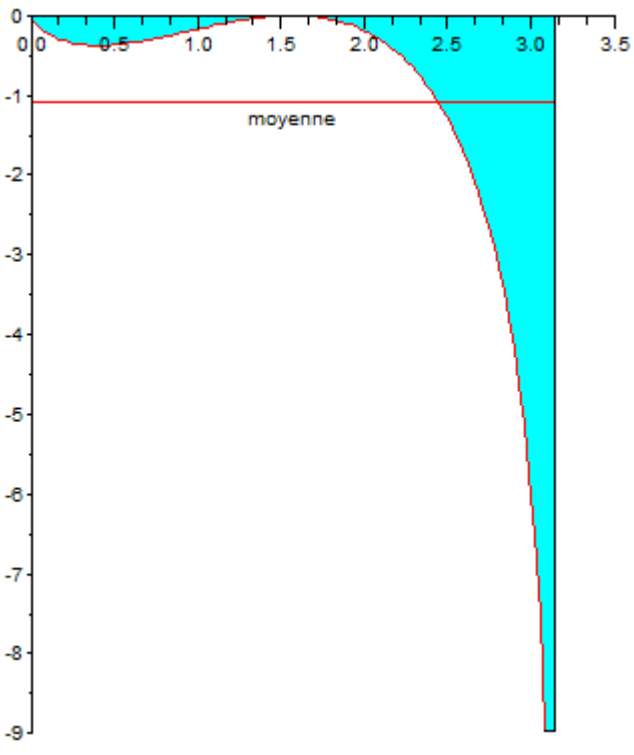
$$S = (b-a) \times \frac{\sum_{i=1}^{N} f(x_i)}{N}$$

Géométriquement, l'aire algébrique sous la courbe (notée I) est approximée par l'aire du rectangle (notée S) dont un côté vaut (*b*-*a*) et l'autre côté vaut $\Sigma f(x_i)/N$, c'est-à-dire une estimation de la valeur moyenne de la fonction *f* sur [*a*,*b*].
L'écart entre S et I sera faible (en probabilité) pour un nombre de points N suffisant.
Avec les ordinateurs actuels on peut aisément faire des calculs sur plusieurs millions de points en un temps raisonnable, ce qui rend la méthode facile à mettre en œuvre. Application :

$$\int_0^{\pi} x\,Log(\sin x)dx = -\frac{\pi^2}{2}Log\,2 = -3.4205442$$

Le programme :

```
N = input('N ?') ;
s = 0 ;
for i = 1:N
   P = grand(1, 1, 'def') ;
   if (P <> 0) & (P <> 1) then
      s = s + %pi*P*log(sin(%pi*P)) ;
   end
end
I = %pi*s/N
```



donne les résultats suivants :

N	I
10^3	- 3.3703960
10^4	- 3.4442343
10^5	- 3.4043024
10^6	- 3.4162092
10^7	- 3.4203205
10^8	- 3.4202922

On constate que la valeur calculée pour 10^8 points est moins précise que celle fournie pour 10^7 points, en raison du caractère aléatoire de la méthode et de l'accumulation des erreurs d'arrondi. Finalement, on a quand même obtenu une bonne approximation de cette intégrale impropre, en écartant les points singuliers P = 0 et P = 1 correspondant aux bornes x = 0 et x = π de l'intégrale. Le test P ≠1 peut être omis car la fonction "*grand*" ne retourne jamais la valeur 1.
Pour une intégrale simple, la formule des trapèzes ou la formule de **Simpson** sont plus efficaces.

Exemple 4

La boule de $\mathbb{R}^n$, de centre 0 et de rayon R, est l'ensemble des points de $\mathbb{R}^n$ vérifiant : $x_1^2 + x_2^2 + ... + x_n^2 \leq R^2$
Son volume est donné par :

$$V = \iint \dots \int dx_1\, dx_2 \dots dx_n = \frac{\pi^{n/2} R^n}{\Gamma(n/2+1)}$$

Avec le changement de variable $\xi_i = (1/2R)x_i + 1/2$ on obtient l'expression équivalente :

$$V = (2R)^n \iint \dots \int d\xi_1\, d\xi_2 \dots d\xi_n$$

Le nouveau domaine d'intégration est la boule de $\mathbb{R}^n$ de centre $(\frac{1}{2}, \frac{1}{2}, \dots, \frac{1}{2})$ et de rayon $\frac{1}{2}$
Prenons R = 1 et n ∈ {1, 2, ... , 8}, puis évaluons *V* par la méthode de Monte-Carlo avec N = 10^6 points de calcul :

```
N = 1D6 ;
R = 1 ;
for n = 1:8
  s = 0 ;
  for i = 1:N
    P = grand(1, n, 'def') ;
    if sum((P - 1/2)^2) <= 1/4 then, s = s + 1 ; end
  end
  V = (2*R)^n*s/N
  e = abs(V/((%pi^(n/2)*R^n)/gamma(n/2+1)) - 1)*100
end
```

Résultats de 4 exécutions successives du programme :

n	1	2	3	4	5	6	7	8
V	2.	3.141492	4.187288	4.936896	5.274272	5.170304	4.68096	4.074752
e %	2.220D-14	0.0032039	0.0358625	0.0424292	0.1991529	0.0501425	0.9271564	0.3951961

n	1	2	3	4	5	6	7	8
V	2.	3.14354	4.185064	4.934048	5.263936	5.140288	4.77312	4.073728
e %	2.220D-14	0.0619860	0.0889566	0.0152833	0.0027924	0.5306947	1.0234164	0.3699665

n	1	2	3	4	5	6	7	8
V	2.	3.14022	4.19812	4.953136	5.254976	5.15488	4.7168	4.063488
e %	2.220D-14	0.0436929	0.2227325	0.3715205	0.1674272	0.2483261	0.1686003	0.1176697

n	1	2	3	4	5	6	7	8
V	2.	3.140936	4.18788	4.939408	5.269472	5.1728	4.721024	4.033792
e %	2.220D-14	0.0209019	0.0217295	0.0933330	0.1079638	0.0984424	0.0791991	0.6139910

Conclusion
Certes, la méthode de Monte-Carlo n'est pas très précise (à moins de prendre un très grand nombre de points ce qui augmente le temps de calcul), mais elle permet d'évaluer une intégrale multiple quelconque en utilisant un algorithme simple et indépendant de la dimension de l'intégrale.
On améliore la méthode de Monte-Carlo en produisant des nombres aléatoires distribués non pas suivant une loi de probabilité uniforme, mais suivant une loi de probabilité qui tient compte de la fonction *f* à intégrer, plus exactement une loi qui produit davantage de nombres aléatoires dans les régions où f prend des valeurs élevées afin d'accroître la précision là où c'est nécessaire (possible en choisissant convenablement la densité de probabilité de la nouvelle loi).

⇨ LA METHODE DE MONTE-CARLO METROPOLIS - Echantillonnage suivant l'importance - Marc **Hayoun**
http://www.cpt.univ-mrs.fr/~crepieux/stock/MC_Metropolis.pdf

Mathématique & Informatique

LES PROGRAMMES

// Calcul d'une intégrale simple

```
N = input('N ?') ;
s = 0 ;
for i = 1:N
  P = grand(1, 1, 'def') ;
  if (P <> 0) & (P <> 1) then
    s = s + %pi*P*log(sin(%pi*P)) ;
  end
end
I = %pi*s/N

// Dessin

x = linspace(0.01, 3.087, 101) ; y = x.*log(sin(x)) ;
xfpolys([x,%pi,%pi,0]', [y,3.087*log(sin(3.087)),0,0]', 4)
a = gca() ; a.x_location = "origin" ; a.y_location = "origin" ;
plot2d(x, y, style = 5)
plot2d([0,%pi], [-%pi*log(2)/2,-%pi*log(2)/2], style = 5)
xstring(+1.30, -1.45, "moyenne")
```

// Calcul d'une intégrale triple

```
N = input('N ?') ;
s = 0 ;
for i = 1:N
  P = grand(1, 3, 'def') ;
  if sum((P - 1/2)^2) <= 1/4 then
    s = s + 8*sqrt(sum((2*P - 1)^2)) ;
  end
end
I = s/N , e = abs(I/%pi - 1)*100

// Dessin

function  [x, y, z] = sphere(u, v)
  x = sin(v).*cos(u) ;
  y = sin(v).*sin(u) ;
  z = cos(v) ;
endfunction
u = linspace(0, 2*%pi, 21) ;
v = linspace(0, %pi, 21) ;
[xf,  yf,  zf] = eval3dp(sphere, u, v) ;
gris = color('light grey') ;
plot3d(xf, yf, zf, theta = 45, alpha = 60, flag = [gris, 4, 3])
```

// Calcul d'une intégrale double

```
grand('setsd', 0)
N = 5000 ;
s = 0 ;
for i = 1:N
  P = grand(1, 2, 'def') ;
  if (2*P(2) - 1)^2 <= P(1) then
    s = s + 8/(3 + P(1)^2 + 8*P(2)^2 - 8*P(2))^2 ;
  end
end
I = s/N

// Dessin1

subplot(1, 2, 1)
drawlater()
m = 50 ; n = 50 ;
x = linspace(0, 2, m) ; y = linspace(-2, +2, n) ;
z = zeros(m, n) ; mask = zeros(m, n) ;
for i = 1:m
  for j = 1:n
    if y(j)^2 <= 2*x(i) then
      mask(i,j) = 1 ;
      z(i,j) = 1/(1 + x(i)^2/4 + y(j)^2/2)^2 ;
    end
  end
end
[xf,yf,zf] = genfac3d(x, y, z, mask) ;
plot3d(xf, yf, zf, theta = 40, alpha = 70, flag = [15,2,4])
xtitle("", "x1", "x2", "x3")
drawnow()

// Dessin2

subplot(1, 2, 2)
drawlater()
y = 0:0.01:1 ; x = (2*y - 1)^2 ;
xfpolys(x', y', 7)
grand('setsd', 0)
for i = 1:N
  P = grand(1, 2, 'def') ;
  plot2d(P(1), P(2), style = 0)
end
xtitle("", "Ksi1", "Ksi2")
drawnow()
```

// Calcul d'une intégrale multiple

```
N = 1D6 ;
R = 1 ;
for n = 1:8
  s = 0 ;
  for i = 1:N
    P = grand(1, n, 'def') ;
    if sum((P - 1/2)^2) <= 1/4 then, s = s + 1 ; end
  end
  V = (2*R)^n*s/N
  e = abs(V/((%pi^(n/2)*R^n)/gamma(n/2+1)) - 1)*100
end
```

Mathématique & Informatique

⇨ Les équations

- La fonction *fsolve* est utilisée pour résoudre des systèmes d'équations non linéaires : f(x) = 0
- La fonction *ode* est utilisée pour résoudre des systèmes d'équations différentielles ordinaires : dy/dt = f(t,y)
- La fonction *linpro* est utilisée pour résoudre des problèmes de programmation linéaire : cx = z(Min), Ax ≤ b

(Consulter la documentation "Optimisation et Simulation" en tapant help fsolve, help ode, ou help linpro)

Exemple "fsolve" :

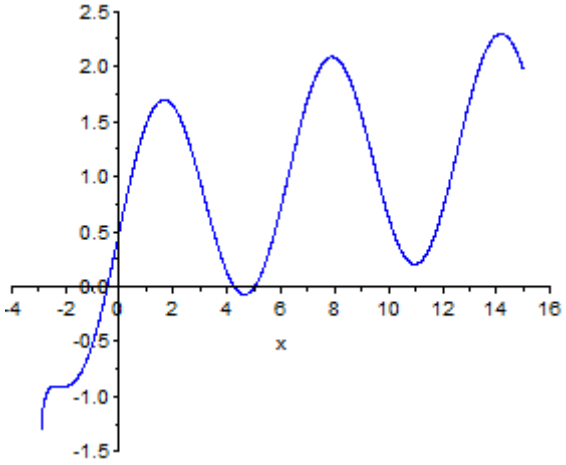
Recherchons les solutions de l'équation suivante :

$$\sin(x) + \log_9 (x+3) = 0$$

Le programme :

deff('y = f(x)', 'y = sin(x) + log(x+3)/log(9)') ;
x0 = 0 ; x1 = fsolve(x0, f)
x0 = 4 ; x2 = fsolve(x0, f)
x0 = 5 ; x3 = fsolve(x0, f)

donne les racines :
x1 = - 0.4417337 ; x2 = 4.2675854 ; x3 = 5.0350787



NB.
Sur le dessin les racines réelles de f(x) = 0 sont les points de rencontre de la courbe avec l'axe des x.
En pratique le graphique permet d'obtenir une estimation grossière de la racine (valeur x0) ou de la séparer, c'est-à-dire de définir un intervalle [a,b] où elle est la seule. Il ne reste plus alors qu'à appliquer un algorithme qui converge vers la solution (méthode par dichotomie, méthode des tangentes, ...).

Exemple "linpro" :

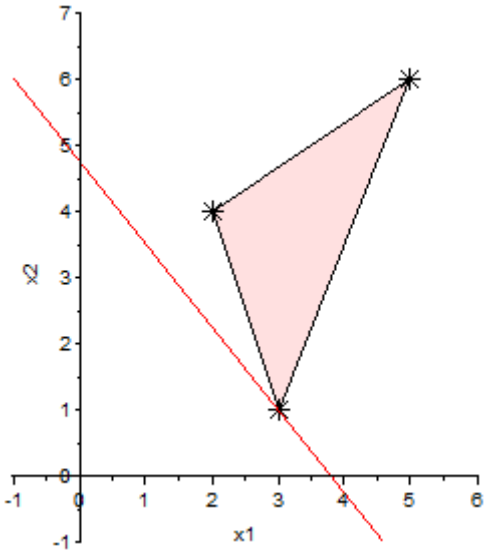
Il s'agit de minimiser une fonction z sous un ensemble de contraintes :

$$\begin{cases} 5x_1 + 4x_2 = z(Min) \\ 5x_1 - 2x_2 \le 13 \\ -2x_1 + 3x_2 \le 8 \\ -3x_1 - x_2 \le -10 \end{cases}$$

Le programme :

A = [5,-2;-2,3;-3,-1] ; b = [13;8;-10] ; c = [5;4] ;
[x, lagrange, z] = linpro(c, A, b)

donne l'optimum : z = 19. ; x(1) = 3. ; x(2) = 1.



NB.
Pour calculer le maximum cx = z(Max), il suffit de changer le signe de la fonction linéaire en écrivant -cx = z(Min), puis de prendre -z comme optimum. Dans l'exemple ci-dessus on écrit c = [-5;-4] et on obtient z = -49. , x(1) = 5. , x(2) = 6. ; d'où l'optimum 49. Sur le dessin, la droite en rouge se déplace vers le sommet supérieur du triangle.
Au-delà de n = 3 variables, on ne peut plus représenter graphiquement un programme linéaire. La solution s'obtient en appliquant des algorithmes comme le célèbre "simplexe" de **Dantzig** qui consiste en fait à rechercher l'optimum en parcourant les sommets du polyèdre convexe de $\mathbb{R}^n$ défini par les inégalités $\Sigma a_{ij}x_j \le b_i$ (i = 1, ... , m ; j = 1, ... , n).

Mathématique & Informatique

La chute libre dans l'air

Lors de la chute libre dans l'air d'un corps de masse m, les forces appliquées sont le poids $\vec{p}$ de module $p = mg$ ($g = 9.81$ m/s^2 est l'accélération de la pesanteur) et la résistance de l'air $\vec{f}$, opposée au mouvement, de module $f = kv^2$ (k est le coefficient de frottement, v est la vitesse). Le principe fondamental de la dynamique $\vec{p} + \vec{f} = m\vec{a}$ conduit à l'équation différentielle vérifiée par la vitesse du corps :

$$\frac{dv}{dt} + \frac{k}{m} v^2 - g = 0$$

Cette équation différentielle non linéaire du premier ordre, de type équation de **Riccati**, peut être intégrée.

Notons v_l la solution particulière constante et positive, on a : $v_l = \sqrt{\frac{mg}{k}}$

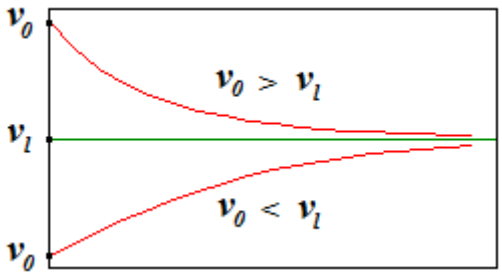
Posons $v = v_l + \frac{1}{w}$ pour se ramener à l'équation différentielle linéaire : $\frac{dw}{dt} - \frac{2g}{v_l} w - \frac{g}{{v_l}^2} = 0$

La solution est de la forme : $w = \lambda e^{\frac{2g}{v_l}t} - \frac{1}{2v_l}$ où λ est une constante qui dépend des conditions initiales.

Revenons à la variable v : $v = v_l + \dfrac{1}{\lambda e^{\frac{2g}{v_l}t} - \frac{1}{2v_l}}$

Soit v_0 la vitesse initiale du corps (v à $t = 0$), on obtient $\lambda = \frac{1}{2v_l}\left(\frac{v_0 + v_l}{v_0 - v_l}\right)$, valeur que l'on reporte dans l'expression de v :

$$v = v_l \left[\frac{\left(\frac{v_l + v_0}{v_l - v_0}\right) e^{\frac{2g}{v_l}t} - 1}{\left(\frac{v_l + v_0}{v_l - v_0}\right) e^{\frac{2g}{v_l}t} + 1} \right]$$



Distinguons les 3 cas possibles :

$$\begin{cases} v_0 < v_l & \Rightarrow & v = v_l \operatorname{th}\left(\frac{g}{v_l}t + \operatorname{Argth}\left(\frac{v_0}{v_l}\right)\right) \\ v_0 = v_l & \Rightarrow & v = v_l \\ v_0 > v_l & \Rightarrow & v = v_l \coth\left(\frac{g}{v_l}t + \operatorname{Argcoth}\left(\frac{v_0}{v_l}\right)\right) \end{cases}$$

Dans tous les cas $v \xrightarrow{t \to +\infty} v_l$ puisque $\operatorname{th} z \xrightarrow{z \to +\infty} 1$ et $\coth z \xrightarrow{z \to +\infty} 1$. Donc v_l est une vitesse limite.

Connaissant la vitesse v, on en déduit l'accélération $a = \frac{dv}{dt}$ et le déplacement $x = \int v\,dt$:

$$\begin{cases} v_0 < v_l & \Rightarrow & a = \dfrac{g}{\operatorname{ch}^2\left(\frac{g}{v_l}t + \operatorname{Argth}\left(\frac{v_0}{v_l}\right)\right)} \\ v_0 = v_l & \Rightarrow & a = 0 \\ v_0 > v_l & \Rightarrow & a = \dfrac{-g}{\operatorname{sh}^2\left(\frac{g}{v_l}t + \operatorname{Argcoth}\left(\frac{v_0}{v_l}\right)\right)} \end{cases} \quad \text{et} \quad \begin{cases} v_0 < v_l & \Rightarrow & x = \frac{{v_l}^2}{g} \ln\left[\dfrac{\operatorname{ch}\left(\frac{g}{v_l}t + \operatorname{Argth}\left(\frac{v_0}{v_l}\right)\right)}{\operatorname{ch}\left(\operatorname{Argth}\left(\frac{v_0}{v_l}\right)\right)}\right] + x_0 \\ v_0 = v_l & \Rightarrow & x = v_l t + x_0 \\ v_0 > v_l & \Rightarrow & x = \frac{{v_l}^2}{g} \ln\left[\dfrac{\operatorname{sh}\left(\frac{g}{v_l}t + \operatorname{Argcoth}\left(\frac{v_0}{v_l}\right)\right)}{\operatorname{sh}\left(\operatorname{Argcoth}\left(\frac{v_0}{v_l}\right)\right)}\right] + x_0 \end{cases}$$

Mathématique & Informatique

Application numérique

Un parachute de masse totale 100 kg s'ouvre alors que sa vitesse est de 200 km/h.
Quelle sera sa nouvelle vitesse v au bout de 2 s ? (On prendra $k = 10$ U.S.I.)

Le programme :

```
deff('yp = f(t,y)', 'yp = - (k/m)*y^2 + g') ;
m = 100 ; k = 10 ; g = 9.81 ;
v0 = 200*1D3/3600 ; t0 = 0 ; t = 2 ;
v = ode(v0, t0, t, f) ;
v = v*3600/1D3 ;
disp(v)
```

donne la vitesse : $v = 36.61542$ km/h

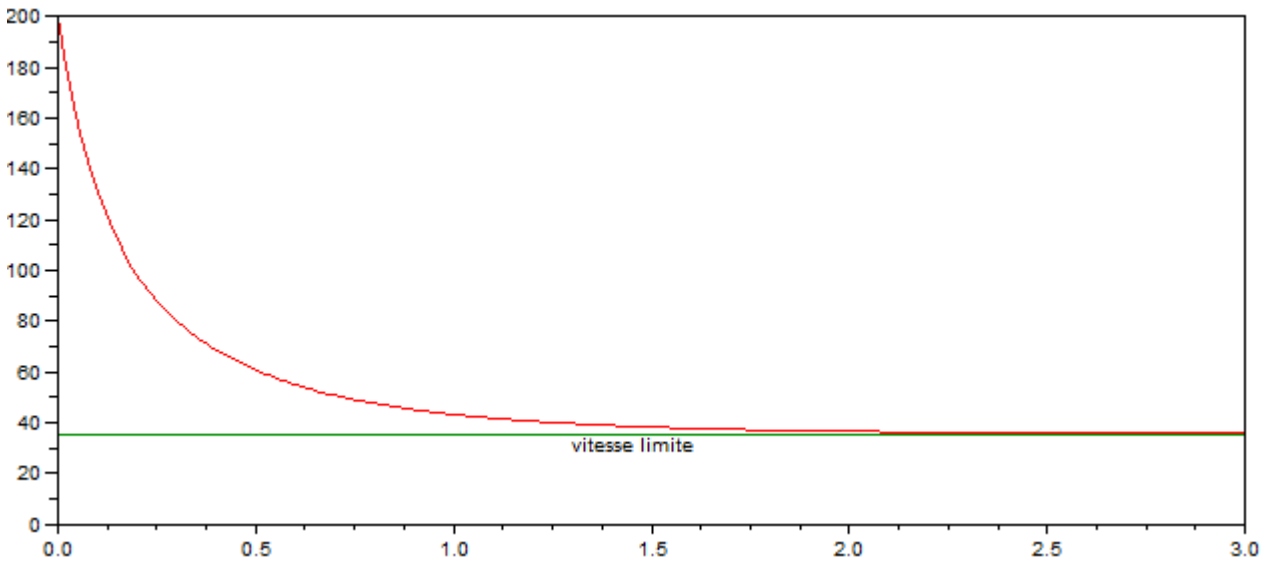
On retrouve cette valeur en utilisant la formule avec les fonctions hyperboliques dans le cas $v_0 > v_l$:

```
m = 100 ; k = 10 ; g = 9.81 ;
v0 = 200*1D3/3600 ; t = 2 ;
vl = sqrt(m*g/k) ;
v = vl*coth(g/vl*t + acoth(v0/vl)) ;
vl = vl*3600/1D3 ;
v = v*3600/1D3 ;
disp(vl)
disp(v)
```

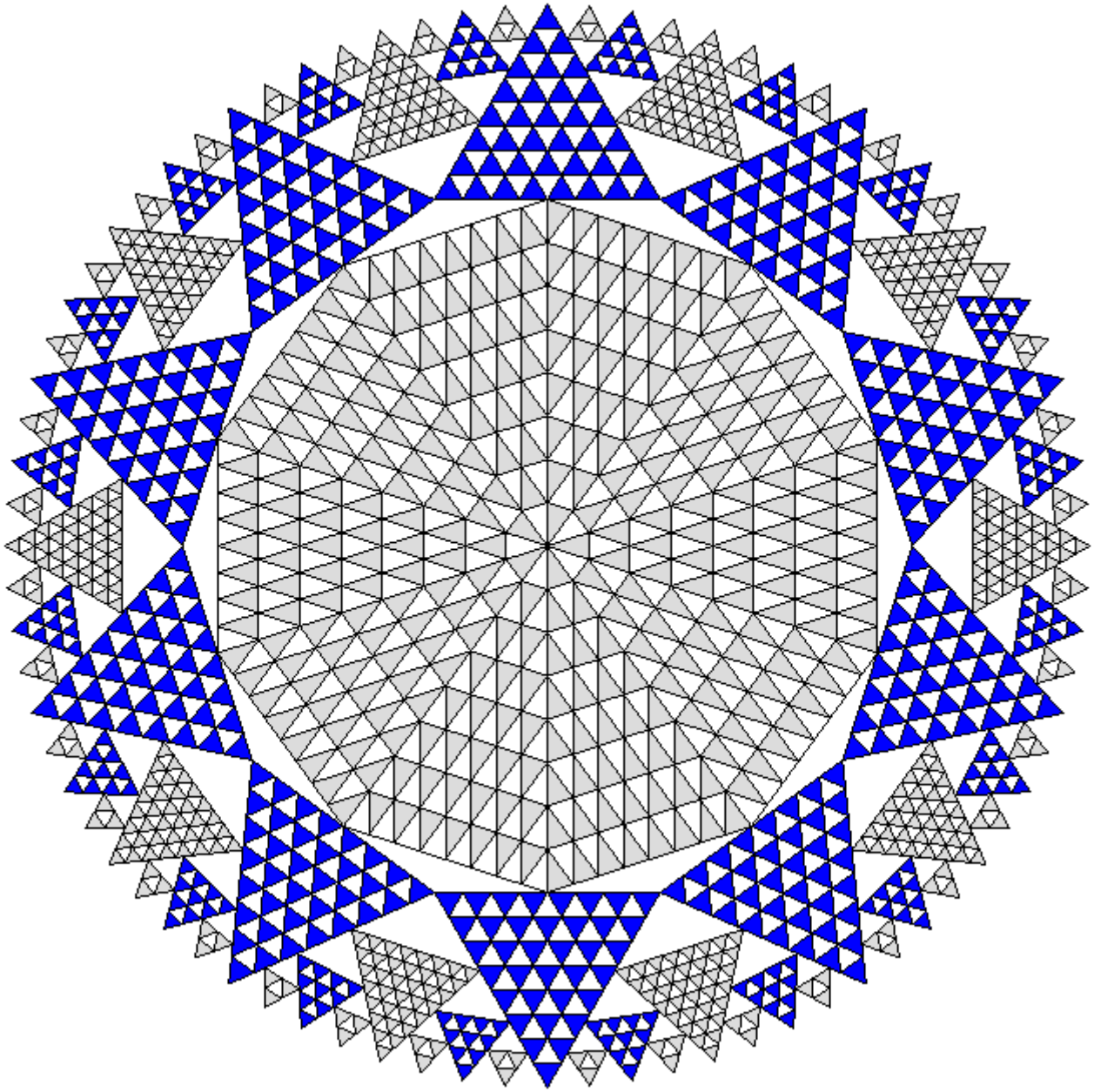
Soit : $v = 36.615422$ km/h
Le parachute atteint donc rapidement sa vitesse limite théorique $v_l = 35.65636$ km/h

Traçons la courbe de variation de la vitesse en fonction du temps (secondes en abscisse ; kilomètres-heure en ordonnée) :

```
deff('yp = f(t,y)', 'yp = - (k/m)*y^2 + g') ;
m = 100 ; k = 10 ; g = 9.81 ;
v0 = 200*1D3/3600 ; t0 = 0 ; t = 0:0.01:3 ;
vl = sqrt(m*g/k) ;
v = ode(v0, t0, t, f) ;
plot2d(t, 3600/1D3*v, 5, frameflag = 1, rect = [0,0,3,200])
plot2d([0,3], 3600/1D3*[vl,vl], 13, frameflag = 1, rect = [0,0,3,200])
xstring(+1.30, +26.00, "vitesse limite")
xtitle("", "t", "V")
```



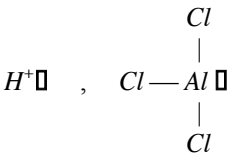
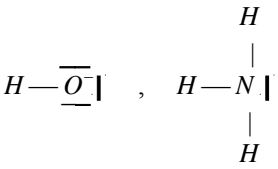
Mathématique & Informatique



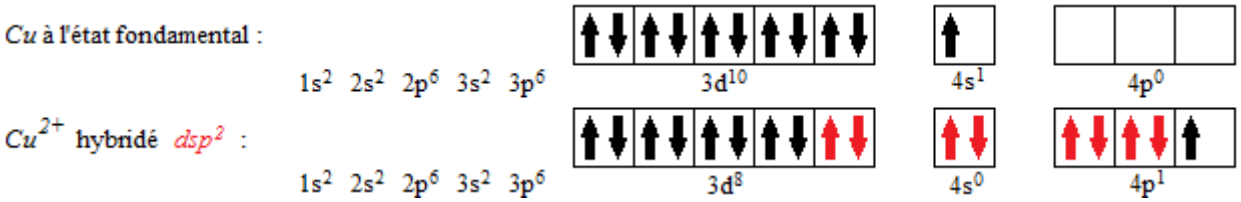
Mathématique & Informatique

Rappel succinct des définitions des acides et des bases en chimie

Essentiellement 3 théories sont utilisées : celle d'**Arrhenius**, celle de **Brønsted**, celle de **Lewis**.
Il est usuel de les comparer sous la forme d'un tableau :

	Arrhenius	**Brønsted**	**Lewis**
Acide	cède H^+ : $AH \rightarrow A^- + H^+$	cède H^+ : $AH \rightleftarrows A^- + H^+$	capte un doublet électronique : A□
Base	cède OH^- : $BOH \rightarrow B^+ + OH^-$	capte H^+ : $B + H^+ \rightleftarrows BH^+$	cède un doublet électronique : B▮
Neutralisation	$AH + BOH \rightarrow A^- + B^+ + H_2O$	$AH + B \rightleftarrows A^- + BH^+$	$A \longleftarrow B$ (liaison de coordination)
Remarque	Cette définition ne permet pas d'expliquer le caractère basique de certains composés comme l'ammoniac NH_3.	A^- est la base conjuguée de l'acide AH ; BH^+ est l'acide conjugué de la base B ; AH/A^- et BH^+/B sont des couples acide/base conjugués.	□ correspond à une case quantique vide (□ = 0 e^-) ▮ correspond à une case quantique pleine (⇅ = 2 e^-)
Exemples	• HCl est un acide d'**Arrhenius** : $HCl \rightarrow H^+ + Cl^-$ • $NaOH$ est une base d'**Arrhenius** : $NaOH \rightarrow Na^+ + OH^-$	• CH_3COOH est un acide de **Brønsted** : $CH_3COOH \rightleftarrows CH_3COO^- + H^+$ • NH_3 est une base de **Brønsted** : $NH_3 + H^+ \rightleftarrows NH_4^+$	• H^+ et $AlCl_3$ sont des acides de **Lewis** :  • OH^- et NH_3 sont des bases de **Lewis** : 

En pratique, les définitions d'**Arrhenius** et de **Brønsted** conduisent à des équations qui permettent d'effectuer le calcul du pH et du pK en milieu ionique. La définition de **Lewis**, plus générale, sert à décrire des mécanismes chimiques : réaction électrophile ou nucléophile, formation des complexes, … Par exemple, la formation du complexe $Cu(NH_3)_4^{2+}$ peut être interprétée comme une réaction de neutralisation acide-base de **Lewis** dans laquelle le cation Cu^{2+} présente un caractère acide avec 4 lacunes électroniques.



(les cases remplies par des doublets en rouge correspondent aux 4 liaisons de coordination avec 4 atomes d'azote N)

Mathématique & Informatique

pH et polyacidité

Dans une solution aqueuse se trouvent mélangés un triacide faible AH_3 de molarité a et une base forte BOH de molarité b.
Quelle est la valeur du pH de la solution ?

Les espèces chimiques dissoutes sont AH_3 , AH_2^- , AH^{2-} , A^{3-} , B^+ , OH^- , H^+
On a les équations suivantes liant les différentes concentrations :

- Equilibres

$$\left[H^+\right]\left[OH^-\right] = K_0 \quad \text{(autoprotolyse de l'eau : } H_2O \rightleftarrows H^+ + OH^-\text{)}$$

$$\frac{\left[H^+\right]\left[AH_2^-\right]}{\left[AH_3\right]} = K_1 \quad \text{(première autoprotolyse de l'acide : } AH_3 \rightleftarrows H^+ + AH_2^-\text{)}$$

$$\frac{\left[H^+\right]\left[AH^{2-}\right]}{\left[AH_2^-\right]} = K_2 \quad \text{(deuxième autoprotolyse de l'acide : } AH_2^- \rightleftarrows H^+ + AH^{2-}\text{)}$$

$$\frac{\left[H^+\right]\left[A^{3-}\right]}{\left[AH^{2-}\right]} = K_3 \quad \text{(troisième autoprotolyse de l'acide : } AH^{2-} \rightleftarrows H^+ + A^{3-}\text{)}$$

- Bilans

$$\left[AH_3\right] + \left[AH_2^-\right] + \left[AH^{2-}\right] + \left[A^{3-}\right] = a$$

$$\left[B^+\right] = b$$

- Electroneutralité

$$\left[B^+\right] + \left[H^+\right] = \left[AH_2^-\right] + 2\left[AH^{2-}\right] + 3\left[A^{3-}\right] + \left[OH^-\right]$$

Le produit ionique de l'eau est une constante K_0 (sans unité) qui dépend de la température et de la pression.
On suppose que les 3 constantes d'acidité K_1 ($pK_1 = -\log_{10}K_1$), K_2 ($pK_2 = -\log_{10}K_2$), K_3 ($pK_3 = -\log_{10}K_3$) sont connues.
Les données a et b étant fixées, on déduit des 7 relations précédentes l'équation algébrique de degré 5 vérifiée par $[H^+]$:

$$\left[H^+\right]^5 + (K_1 + b)\left[H^+\right]^4 - (K_0 - K_1K_2 + aK_1 - bK_1)\left[H^+\right]^3 - (K_0K_1 - K_1K_2K_3 + 2aK_1K_2 - bK_1K_2)\left[H^+\right]^2$$
$$- (K_0K_1K_2 + 3aK_1K_2K_3 - bK_1K_2K_3)\left[H^+\right] - K_0K_1K_2K_3 = 0$$

La résolution numérique de cette équation permet de calculer le pH (potentiel Hydrogène) :

$$pH = -\log_{10}\left[H^+\right]$$

Remarque 1
Dans le cas d'un diacide faible AH_2 , faire $K_3 = 0$ et ramener l'équation en $[H^+]$ au degré 4.
Dans le cas d'un monoacide faible AH, faire $K_2 = K_3 = 0$ et ramener l'équation en $[H^+]$ au degré 3.

Remarque 2
Dans une solution aqueuse, les protons H^+ ne sont pas libres mais associés à une ou plusieurs molécules d'eau pour former un ion H_3O^+ à structure tétraédrique ou plus généralement un ion solvaté de type $H(H_2O)_n^+$ à structure complexe. Bien que le nombre n de molécules d'eau porteuses en liaison avec un proton ne soit pas précisément déterminé, il est courant de remplacer l'ion H^+ par l'ion H_3O^+ que l'on considère être la forme prépondérante. En fait, comme l'espèce réagissante est le proton H^+ d'après la théorie de **Brønsted-Lowry**, cela ne change rien. Donc, dans un but de simplification, on conservera la notation H^+ dans les équations.

Remarque 3
Convention d'écriture : pX désigne le cologarithme décimal de la molarité d'une espèce X ($pX = \text{colog}_{10}[X] = -\log_{10}[X]$) et pK le cologarithme décimal d'une constante d'équilibre K ($pK = \text{colog}_{10}K = -\log_{10}K$).

Application numérique

Le produit ionique de l'eau est $K_0 = 10^{-14}$ (à 25 °C sous 1 atm).
Les pK de l'acide phosphorique H_3PO_4 sont :

$pK_1 = 2.15$
$pK_2 = 7.20$
$pK_3 = 12.42$

La neutralisation s'effectue avec de la soude $NaOH$.
Concernant les concentrations molaires, on prend $a = 0.1$ mol/l pour H_3PO_4 et $b = 0.2$ mol/l pour $NaOH$.
L'équation en $[H^+]$ est de type polynomial, ses racines se calculent avec la fonction Scilab *roots* :

```
K0 = 10^(-14) ;
K1 = 10^(-2.15) ;
K2 = 10^(-7.20) ;
K3 = 10^(-12.42) ;
a = 0.1 ;
b = 0.2 ;
H = poly(0,'H') ;
P = H^5 + (K1 + b)*H^4 - (K0 - K1*K2 + a*K1 - b*K1)*H^3 ...
    - (K0*K1 - K1*K2*K3 + 2*a*K1*K2 - b*K1*K2)*H^2 ...
    - (K0*K1*K2 + 3*a*K1*K2*K3 - b*K1*K2*K3)*H - K0*K1*K2*K3 ;
R = roots(P) ;
pH = - log10(R) ;
disp(pH)
```

On obtient :

$pH1 = 0.6912172 - 1.3643764i$
$pH2 = 2.4587825 - 1.3643764i$
$pH3 = 9.7590704$
$pH4 = 9.7595175 - 1.3643764i$
$pH5 = 13.101412 - 1.3643764i$

La seule valeur réelle du pH, correspondant à une racine réelle $[H^+] > 0$, est $pH3$. Par conséquent :

$pH = 9.7590704 \approx 9.76$ (solution basique)

Note

La fonction logarithme décimal s'applique à des nombres complexes non nuls. On a :

$$\log_{10}(z) = \log_{10}(a+ib) = \log_{10}(re^{i\theta}) = \log_{10}(r) + i\frac{\theta}{\log_e(10)} \quad \text{avec } (a,b) \neq (0,0),\ r \in]0,+\infty[,\ \theta \in]-\pi,\pi]$$

Par exemple :

$\log_{10}(10) = 1$
$\log_{10}(10i) = 1 + i\pi/(2\log_e 10)$
$\log_{10}(-10) = 1 + i\pi/(\log_e 10)$
$\log_{10}(-10i) = 1 - i\pi/(2\log_e 10)$

```
log10(10) = 1.
log10(10*%i) = 1. + 0.6821882i
log10(-10) = 1. + 1.3643764i
log10(-10*%i) = 1. - 0.6821882i
```

Le nombre complexe $\log_{10}(z)$ a une partie imaginaire nulle, $\theta = 0$, si et seulement si $a > 0$ et $b = 0$, c'est-à-dire si et seulement si z est un réel strictement positif.

Mathématique & Informatique

Traçons la courbe de titrage $pH = f(x)$ où $x = \dfrac{b}{a}$ est la proportion de soude par rapport à l'acide phosphorique.

L'équation obtenue précédemment est de degré 5 en $[H^+]$, mais elle est de degré 1 en x, on a :

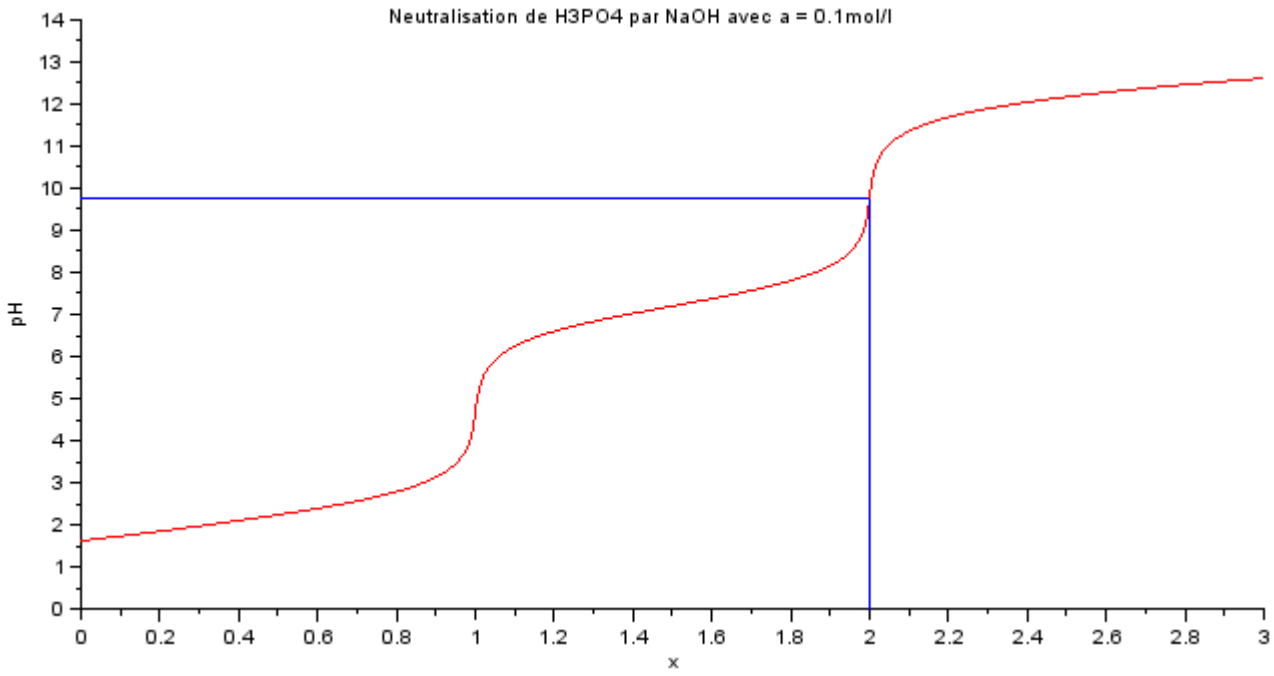
$$x = \frac{-[H^+]^5 - K_1[H^+]^4 + (K_0 - K_1K_2 + aK_1)[H^+]^3 + (K_0K_1 - K_1K_2K_3 + 2aK_1K_2)[H^+]^2 + (K_0K_1K_2 + 3aK_1K_2K_3)[H^+] + K_0K_1K_2K_3}{a[H^+]\{[H^+]^3 + K_1[H^+]^2 + K_1K_2[H^+] + K_1K_2K_3\}}$$

La courbe théorique se construit point par point en choisissant une valeur du pH puis en calculant la valeur de x correspondante.

```
function pHx=f(x)
    H = poly(0,'H')
    P = H^5 + (K1 + a*x)*H^4 - (K0 - K1*K2 + a*K1 - a*x*K1)*H^3 ...
        - (K0*K1 - K1*K2*K3 + 2*a*K1*K2 - a*x*K1*K2)*H^2 ...
        - (K0*K1*K2 + 3*a*K1*K2*K3 - a*x*K1*K2*K3)*H - K0*K1*K2*K3
    pH = - log10(roots(P))
    pHx = pH(find(imag(pH) == 0))
endfunction

function xpH=g(pH)
    H = 10^(-pH)
    xpH = (-H^5 - K1*H^4 + (K0 - K1*K2 + a*K1)*H^3 + (K0*K1 - K1*K2*K3 + 2*a*K1*K2)*H^2 ...
           + (K0*K1*K2 + 3*a*K1*K2*K3)*H + K0*K1*K2*K3)/(a*H*(H^3 + K1*H^2 + K1*K2*H + K1*K2*K3))
endfunction

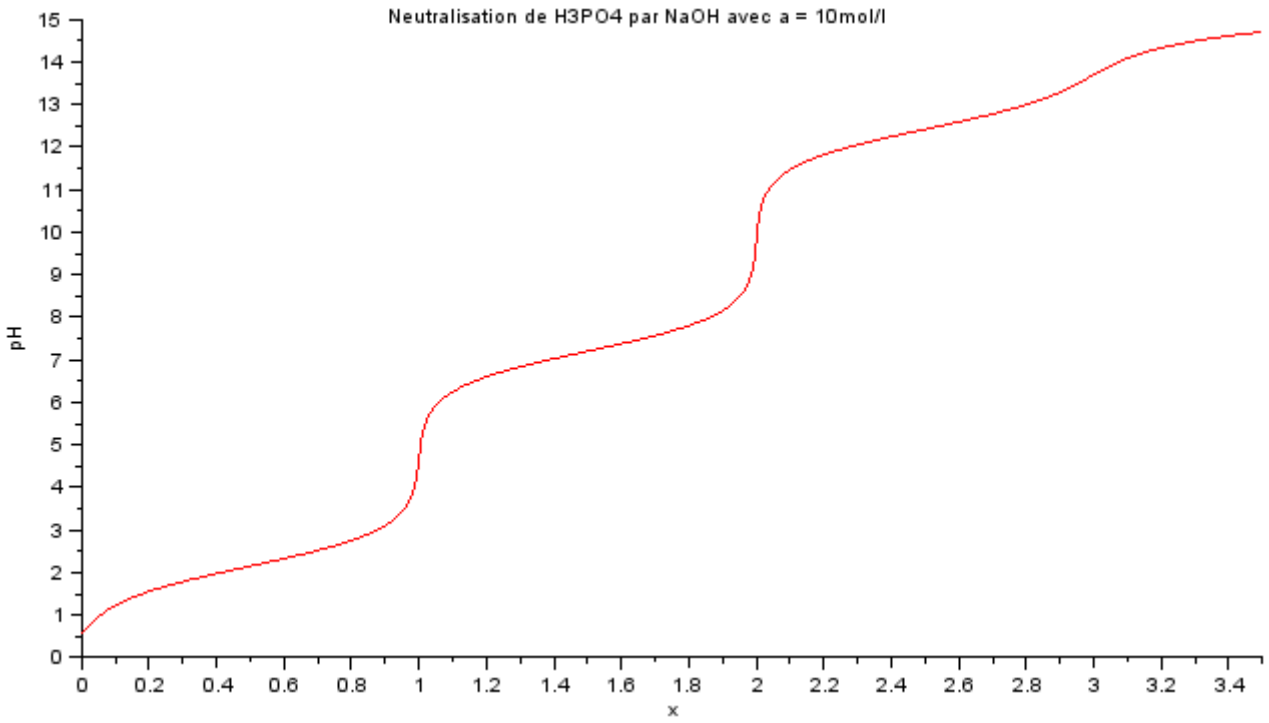
pK1 = 2.15 ; pK2 = 7.20 ; pK3 = 12.42 ;
K0 = 10^(-14) ; K1 = 10^(-pK1) ; K2 = 10^(-pK2) ; K3 = 10^(-pK3) ;
a = 0.1 ; x1 = 0 ; x2 = 3 ; pH1 = f(x1) ; pH2 = f(x2) ; n = 300 ;
pH = linspace(pH1,pH2,n) ;
x = zeros(1,n) ;
for i = 1:n
    x(i) = g(pH(i)) ;
end
xtitle("Neutralisation de H3PO4 par NaOH avec a = " + string(a) + "mol/l", "x", "pH")
plot2d(x,pH,frameflag = 1,rect = [x1,0,x2,14],style = 5)
x0 = 2 ; pH0 = f(x0) ; plot2d([0,x0;x0,x0],[pH0,0;pH0,pH0],[2,2])
```



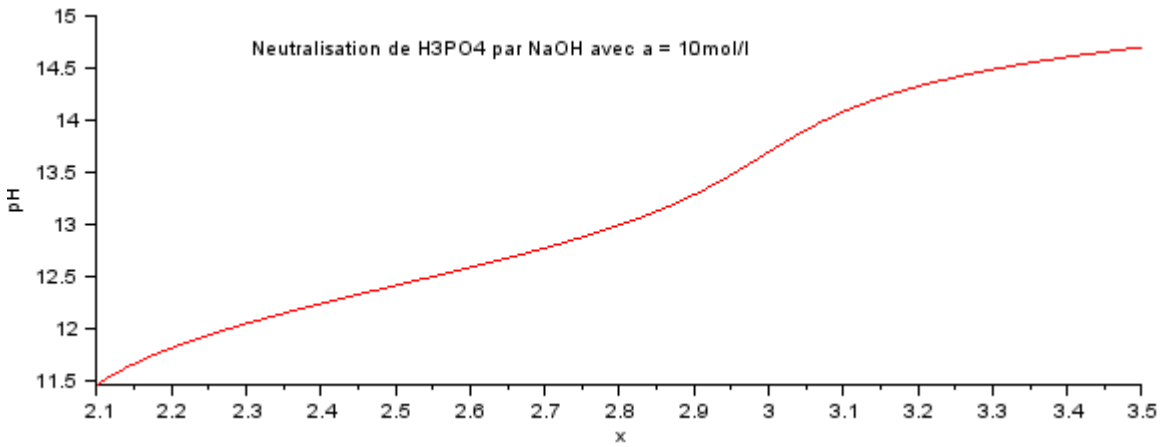
La courbe est conforme à celle obtenue expérimentalement : dans une solution d'acide phosphorique, on verse petit à petit de la soude concentrée (valeur x) en mesurant à chaque fois le pH (valeur y) puis on positionne le point (x,y) à l'aide d'un logiciel graphique. Dans cette expérience, on fait en sorte que l'ajout de soude s'effectue sans variation significative du volume de la solution, ce qui a pour effet de maintenir presque constante la concentration molaire a de l'acide phosphorique. En pratique, pour remplir cette condition, $NaOH$ dans la burette doit être concentrée par rapport à H_3PO_4 dans le bécher. Ecrivons les équations de dissociation de d'acide phosphorique :

$H_3PO_4 \rightleftarrows H^+ + H_2PO_4^-$ (première acidité : formation de l'ion dihydrogénophosphate $H_2PO_4^-$)
$H_2PO_4^- \rightleftarrows H^+ + HPO_4^{2-}$ (deuxième acidité : formation de l'ion hydrogénophosphate HPO_4^{2-})
$HPO_4^{2-} \rightleftarrows H^+ + PO_4^{3-}$ (troisième acidité : formation de l'ion orthophosphate PO_4^{3-})

La troisième acidité étant très faible, elle génère peu de protons H^+ (il est plus difficile d'ôter un proton chargé positivement à HPO_4^{2-} qu'à $H_2PO_4^-$ qui possède une seule charge –, ou à H_3PO_4 qui est neutre) et par conséquent elle n'est pas visible à l'échelle du tracé.
Si l'on souhaite néanmoins mettre en évidence cette troisième acidité en faisant apparaître le point d'inflexion correspondant sur la courbe, il faut augmenter la concentration en acide phosphorique. Par exemple, prenons a = 10 mol/l d'acide phosphorique (équivalent à 980g de H_3PO_4 pur dissous par litre de solution) et notons x le nombre de moles de soude ajoutées par mole d'acide. Considérons, en première approximation, que les équations restent valables pour de fortes concentrations (on ne fait pas intervenir la notion d'activité[*]). La simulation numérique donne la courbe ci-dessous :



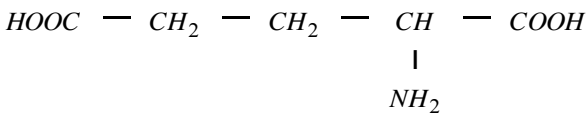
Zoomons sur le point d'inflexion (changement de sens de la courbure) aux alentours de pH = 13 :



Mathématique & Informatique

pH et isoélectricité

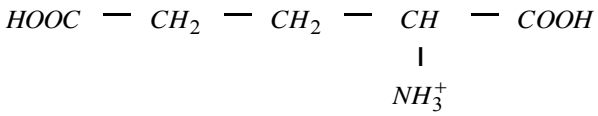
On considère une solution aqueuse d'acide glutamique de formule semi-développée :

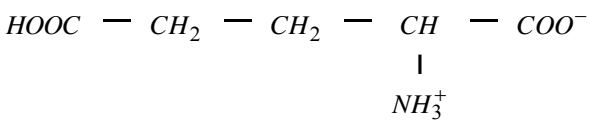


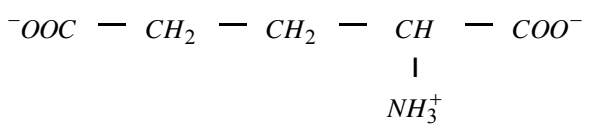
Calculer le *pH* isoélectrique (*pHi*) de cet acide aminé.
Quelle est la valeur du *pH* d'une solution d'acide glutamique de concentration $c = 0.05$ mol/l ?

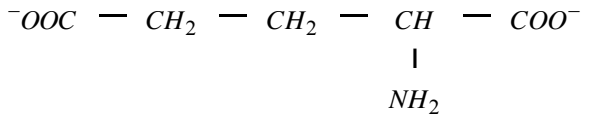
La molécule d'acide glutamique, comme tous les acides aminés, est un ampholyte : elle se comporte tantôt comme un acide en cédant un proton H^+ sur l'un de ses groupements carboxyle $COOH$, tantôt comme une base en captant un proton H^+ sur son groupement amine NH_2.

En solution aqueuse apparaissent les espèces suivantes :


notée symboliquement A^+


notée symboliquement B


notée symboliquement C^-


notée symboliquement D^{2-}

On est en présence de 3 couples acide/base (A^+/B, B/C^-, C^-/D^{2-}) caractérisés par 3 constantes d'équilibre (K_1, K_2, K_3). L'espèce A^+ est un triacide de **Brønsted**. L'espèce B, électriquement neutre, est un zwitterion (amphion) qui correspond à une structure particulière de la molécule d'acide glutamique $C_5H_9O_4N$ (transfert d'un proton H^+ du groupe carboxyle sur le groupe amine). L'espèce C^-, comme l'espèce B, est un ampholyte. L'espèce D^{2-} est une tribase, elle peut capter 3 ions H^+.

Rappelons que toute dissociation acide et base se ramène au même type de réaction d'équilibre :

$$acide \rightleftarrows H^+ + base \quad \text{avec} \quad K = \frac{[H^+][base]}{[acide]}$$

Les réactions qui interviennent sont la réaction d'ionisation de l'eau et les 3 réactions de protolyse de l'acide glutamique :

$$\begin{cases} H_2O \rightleftarrows H^+ + OH^- \\ A^+ \rightleftarrows H^+ + B \\ B \rightleftarrows H^+ + C^- \\ C^- \rightleftarrows H^+ + D^{2-} \end{cases}$$

Le *pH* isoélectrique est par définition la valeur du *pH* pour laquelle il y a égalité entre l'ensemble des charges positives et l'ensemble des charges négatives portées par les différentes formes ionisées de la molécule d'acide glutamique.
Le calcul du *pHi* revient donc à trouver $[H^+]$ avec la condition : $[A^+] = [C^-] + 2[D^{2-}]$

Les équations dont on dispose :

$$\begin{cases} \text{---------- } \textit{Équilibres} \\ [H^+][OH^-] = K_0 \\ \dfrac{[H^+][B\,]}{[A^+]} = K_1 \\ \dfrac{[H^+][C^-]}{[B\,]} = K_2 \\ \dfrac{[H^+][D^{2-}]}{[C^-]} = K_3 \\ \text{---------- } \textit{Bilan} \\ [A^+]+[B\,]+[C^-]+[D^{2-}] = c \\ \text{---------- } \textit{Électroneutralité} \\ [A^+]+[H^+] = [C^-]+2[D^{2-}]+[OH^-] \\ \text{---------- } \textit{Isoélectricité} \\ [A^+] = [C^-]+2[D^{2-}] \end{cases}$$

A partir des 3 équations définissant les constantes d'équilibre K_1, K_2, K_3 exprimons $[B\,]$, $[C^-]$ et $[D^{2-}]$ en fonction de $[A^+]$:

$$[A^+] = \frac{[H^+]}{K_1}[B\,] = \frac{[H^+]^2}{K_1K_2}[C^-] = \frac{[H^+]^3}{K_1K_2K_3}[D^{2-}] \;\Rightarrow\; [B\,] = \frac{K_1}{[H^+]}[A^+], [C^-] = \frac{K_1K_2}{[H^+]^2}[A^+] \text{ et } [D^{2-}] = \frac{K_1K_2K_3}{[H^+]^3}[A^+]$$

En reportant dans la condition d'isoélectricité, on obtient l'équation vérifiée par la concentration en ions H^+ à $pH = pHi$:

$$[H^+]^3 - K_1K_2[H^+] - 2K_1K_2K_3 = 0$$

Application numérique

Données : $\begin{cases} pK_1 = 2.19 \\ pK_2 = 4.25 \\ pK_3 = 9.67 \end{cases}$

Programme de calcul du *pHi* :

```
K1 = 10^(-2.19) ;
K2 = 10^(-4.25) ;
K3 = 10^(-9.67) ;
H = poly(0,'H') ;
P = H^3 - K1*K2*H - 2*K1*K2*K3 ;
R = roots(P) ;
logR = - log10(R) ;
pHi = logR(find(imag(logR) == 0)) ;
disp(pHi)
```

Résultat :

$pHi = 3.22$

Recherchons l'équation vérifiée par $[H^+]$ pour une concentration molaire quelconque c en acide glutamique.
En utilisant la première équation "*Equilibres*", on a immédiatement :

$$[OH^-] = \frac{K_0}{[H^+]}$$

En utilisant l'équation "*Bilan*", on obtient :

$$[A^+] = \left(\frac{[H^+]^3}{[H^+]^3 + K_1[H^+]^2 + K_1K_2[H^+] + K_1K_2K_3} \right) c$$

D'où, d'après une relation précédente :

$$[B\ ] = \left(\frac{K_1[H^+]^2}{[H^+]^3 + K_1[H^+]^2 + K_1K_2[H^+] + K_1K_2K_3} \right) c$$

$$[C^-] = \left(\frac{K_1K_2[H^+]}{[H^+]^3 + K_1[H^+]^2 + K_1K_2[H^+] + K_1K_2K_3} \right) c$$

$$[D^{2-}] = \left(\frac{K_1K_2K_3}{[H^+]^3 + K_1[H^+]^2 + K_1K_2[H^+] + K_1K_2K_3} \right) c$$

En utilisant l'équation "*Electroneutralité*" et les expressions de $[A^+]$, $[C^-]$, $[D^{2-}]$, $[OH^-]$ en fonction de $[H^+]$,
on aboutit finalement à :

$$[H^+]^5 + (K_1 + c)[H^+]^4 - (K_0 - K_1K_2)[H^+]^3 - (K_0K_1 - K_1K_2K_3 + cK_1K_2)[H^+]^2 - (K_0K_1K_2 + 2cK_1K_2K_3)[H^+] - K_0K_1K_2K_3 = 0$$

Cette équation polynomiale se résoud avec la fonction *roots* de Scilab.
Connaissant $[H^+]$, le *pH* est donné par la formule : $pH = -\log_{10}[H^+]$

Application numérique

Par exemple, pour c = 0.05 mol/l, on écrit :

```
K0 = 10^(-14) ;
K1 = 10^(-2.19) ;
K2 = 10^(-4.25) ;
K3 = 10^(-9.67) ;
c = 0.05 ;
H = poly(0,'H') ;
P = H^5 + (K1 + c)*H^4 - (K0 - K1*K2)*H^3 - (K0*K1 - K1*K2*K3 + c*K1*K2)*H^2 ...
    - (K0*K1*K2 + 2*c*K1*K2*K3)*H - K0*K1*K2*K3 ;
R = roots(P) ;
logR = - log10(R) ;
pH = logR(find(imag(logR) == 0)) ;
disp(pH)
```

On trouve :

pH = 3.25

Le *pH* de l'acide glutamique dissous dans l'eau pure est pratiquement égal au *pHi* de la molécule.

Mathématique & Informatique

En ajoutant à la solution aqueuse de $C_5H_9O_4N$ un acide fort, par exemple HCl ($HCl \rightarrow H^+ + Cl^-$), ou une base forte, par exemple $NaOH$ ($NaOH \rightarrow Na^+ + OH^-$), on modifie son pH.

Les équations d'équilibres demeurent inchangées, par contre l'équation d'électroneutralité est différente :

$$\left[A^+\right]+\left[H^+\right]=\left[C^-\right]+2\left[D^{2-}\right]+\left[OH^-\right]+\left[Cl^-\right] \quad \text{avec} \quad \left[Cl^-\right]=c_{HCl} \quad \text{(molarité de l'acide chlorhydrique ajouté)}$$

$$\left[A^+\right]+\left[H^+\right]+\left[Na^+\right]=\left[C^-\right]+2\left[D^{2-}\right]+\left[OH^-\right] \quad \text{avec} \quad \left[Na^+\right]=c_{NaOH} \quad \text{(molarité de la soude ajoutée)}$$

Par suite, l'équation polynomiale de degré 5 en $[H^+]$ se met sous la forme :

$$\left[H^+\right]^5 + (K_1 + c - x)\left[H^+\right]^4 - (K_0 - K_1K_2 + xK_1)\left[H^+\right]^3 - (K_0K_1 - K_1K_2K_3 + cK_1K_2 + xK_1K_2)\left[H^+\right]^2$$
$$- (K_0K_1K_2 + 2cK_1K_2K_3 + xK_1K_2K_3)\left[H^+\right] - K_0K_1K_2K_3 = 0$$

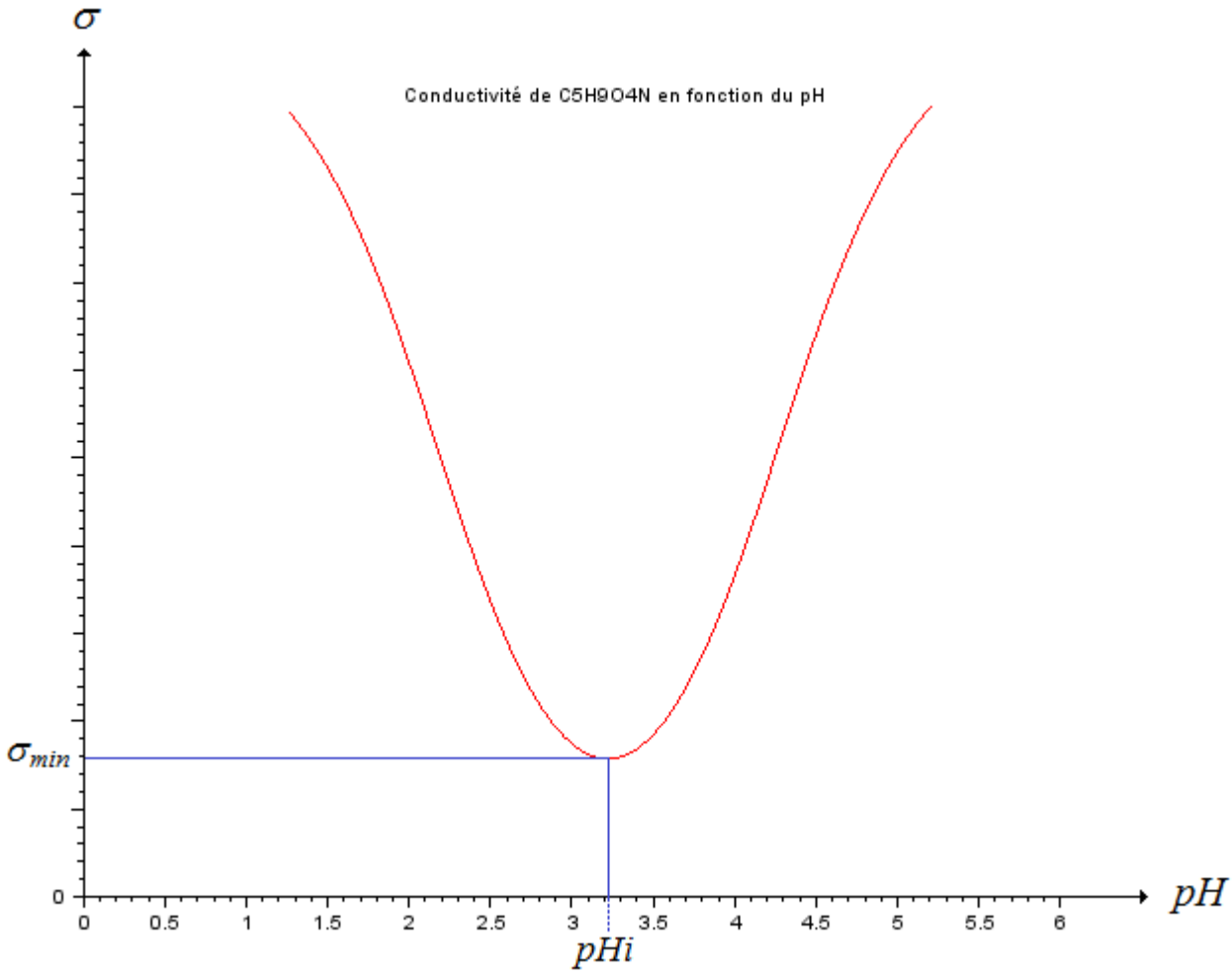
où $x = \begin{cases} +c_{HCl} \\ -c_{NaOH} \end{cases}$

Soit σ la conductivité électrique propre de l'acide glutamique en solution, c'est-à-dire la conductivité due aux seuls ions A^+, C^- et D^{2-} (on fait abstraction des autres ions en solution). En admettant que les conductivités molaires ioniques $\lambda_{A(+)}$, $\lambda_{C(-)}$, $\lambda_{D(2-)}$ de ces 3 ions sont pratiquement identiques et égales à une constante λ, la valeur de σ est proportionnelle à la somme $[A^+] + [C^-] + 2[D^{2-}]$. On a :

$$\sigma = \lambda\left(\left[A^+\right]+\left[C^-\right]+2\left[D^{2-}\right]\right) = \left(\frac{\left[H^+\right]^3 + K_1K_2\left[H^+\right] + 2K_1K_2K_3}{\left[H^+\right]^3 + K_1\left[H^+\right]^2 + K_1K_2\left[H^+\right] + K_1K_2K_3}\right)\lambda c$$

Traçons la courbe $\sigma = f(pH)$ lorsque le pH varie en fonction de la molarité x en acide fort ou en base forte :

```
K0 = 10^(-14) ;
K1 = 10^(-2.19) ;
K2 = 10^(-4.25) ;
K3 = 10^(-9.67) ;
c = 0.05 ;
xmin = -0.045 ;
xmax = 0.1 ;
n = 300 ;
x = linspace(xmin,xmax,n) ;
pH = zeros(1,n) ;
sigma = zeros(1,n) ;
for i = 1:n
    H = poly(0,'H') ;
    P = H^5 + (K1 + c - x(i))*H^4 - (K0 - K1*K2 + x(i)*K1)*H^3 ...
      - (K0*K1 - K1*K2*K3 + c*K1*K2 + x(i)*K1*K2)*H^2 ...
      - (K0*K1*K2 + 2*c*K1*K2*K3 + x(i)*K1*K2*K3)*H - K0*K1*K2*K3 ;
    R = roots(P) ;
    cH = R(find(real(R) > 0 & imag(R) == 0)) ;
    pH(i) = - log10(cH) ;
    sigma(i) = ((cH^3 + K1*K2*cH + 2*K1*K2*K3)/(cH^3 + K1*cH^2 + K1*K2*cH + K1*K2*K3))*c ;
end
xtitle("Conductivité de C5H9O4N en fonction du pH")
plot2d(pH,sigma,frameflag = 1,rect = [0,0,6,max(real(sigma))],style = 5)
```



En fait, on a représenté $\Sigma = [A^+] + [C^-] + 2[D^{2-}] = g(pH)$ et non $\sigma = f(pH)$, mais comme les grandeurs Σ et σ ne diffèrent que par une constante multiplicative λ, l'allure des 2 courbes est la même, en particulier la valeur du pH au point minimum est identique. Ce minimum $\sigma = \sigma_{min}$ correspond approximativement au point isoélectrique $pH = pHi = 3.22$ calculé précédemment.

Remarque

Pour obtenir la courbe donnant la conductivité électrique σ de l'intégralité de la solution en fonction du pH, l'expression de σ doit faire intervenir tous les ions présents dans la solution.

Si l'ion X_i de charge z_i (en valeur absolue) a pour conductivité molaire ionique λ_i, la loi de **Kohlrausch** s'écrit :

$$\sigma = \sum_i \lambda_i z_i [X_i] \qquad \text{σ est en siemens par mètre (S/m)}$$

Les constantes λ_i sont très faibles, de l'ordre de 10^{-8} S.m^2/mol (concentrations $[X_i]$ en mol/m^3).

Dans le cas de concentrations $[X_i]$ élevées, il faudra également tenir compte pour chaque ion X_i de son coefficient d'activité γ_i qui dépend de la force ionique I de la solution (voir théorie de **Debye**) :

$$I = \frac{1}{2}\sum_i z_i^2 [X_i] \qquad I \text{ est en moles par litre (mol/l)}$$

De manière expérimentale, la mesure de la conductivité σ (en µS/cm) s'effectue avec un conductimètre. La méthode consiste à plonger dans la solution ionique une sonde constituée de 2 plaques de platine de surface S équidistantes de l, puis à appliquer entre ces 2 conducteurs une tension alternative de valeur efficace U_e ; un courant électrique d'intensité efficace I_e traverse la solution ionique. Notons R la résistance du tube de courant de section rectangulaire S et de longueur l et utilisons la loi d'**Ohm** pour établir l'expression de la conductivité σ en fonction de S, l, U_e, I_e :

$$U_e = RI_e = \frac{1}{\sigma}\frac{l}{S}I_e \Rightarrow \sigma = \frac{l}{S}\frac{I_e}{U_e}$$ (une solution étalon dont la conductivité σ_0 est connue permet de calculer le rapport l/S)

D'autre part, la mesure du pH s'effectue avec un pH-mètre.

On en déduit le tracé $\sigma = f(pH)$, valable pour une température fixée (en °C) de la solution ionique.

Mathématique & Informatique

pH et tampon

En solution aqueuse, considérons un système tampon *AH*/*ANa* constitué d'un acide faible *AH* de molarité c_1 et de son sel de sodium *ANa* de molarité c_2. On ajoute un acide fort, par exemple *HCl*, de molarité c_3.
Quelle est la valeur du *pH* ?

---

1) Les réactions chimiques

$$\begin{cases} AH \rightleftarrows H^+ + A^- \\ ANa \rightarrow Na^+ + A^- \\ HCl \rightarrow H^+ + Cl^- \\ H_2O \rightleftarrows H^+ + OH^- \end{cases}$$

2) Les équations algébriques

$$\begin{cases} \text{---------- } \textit{Équilibres} \\ \dfrac{[H^+][A^-]}{[AH]} = K_a \\ [H^+][OH^-] = K_e \\ \text{---------- } \textit{Bilans} \\ [Na^+] = c_2 \\ [Cl^-] = c_3 \\ [AH] + [A^-] = c_1 + c_2 \\ \text{---------- } \textit{Électroneutralité} \\ [Na^+] + [H^+] = [A^-] + [Cl^-] + [OH^-] \end{cases}$$

$$\Rightarrow \quad [H^+]^3 + (K_a + c_2 - c_3)[H^+]^2 - (K_e + c_1 K_a + c_3 K_a)[H^+] - K_a K_e = 0$$

Application numérique

AH = CH_3COOH (acide acétique de pK_a = 4.76) et *ANa* = CH_3COONa (acétate de sodium).
Les concentrations en acide acétique, acétate de sodium et acide chlorhydrique sont notées respectivement c_1 , c_2 et c_3.

Programme de calcul du *pH* :

```
Ka = 10^(-4.76) ;
Ke = 10^(-14) ;
c1 = 0.2 ;
c2 = 0.2 ;
c3 = 0.02 ;
H = poly(0,'H') ;
P = H^3 + (Ka + c2 - c3)*H^2 - (Ke + c1*Ka + c3*Ka)*H - Ka*Ke ;
R = roots(P) ;
logR = - log10(R) ;
pH = logR(find(imag(logR) == 0)) ;
disp(pH)
```

Pour $c_1 = c_2$ = 0.2 mol/l et c_3 = 0 mol/l, on trouve *pH* = 4.76
Pour $c_1 = c_2$ = 0.2 mol/l et c_3 = 0.02 mol/l, on trouve *pH* = 4.67

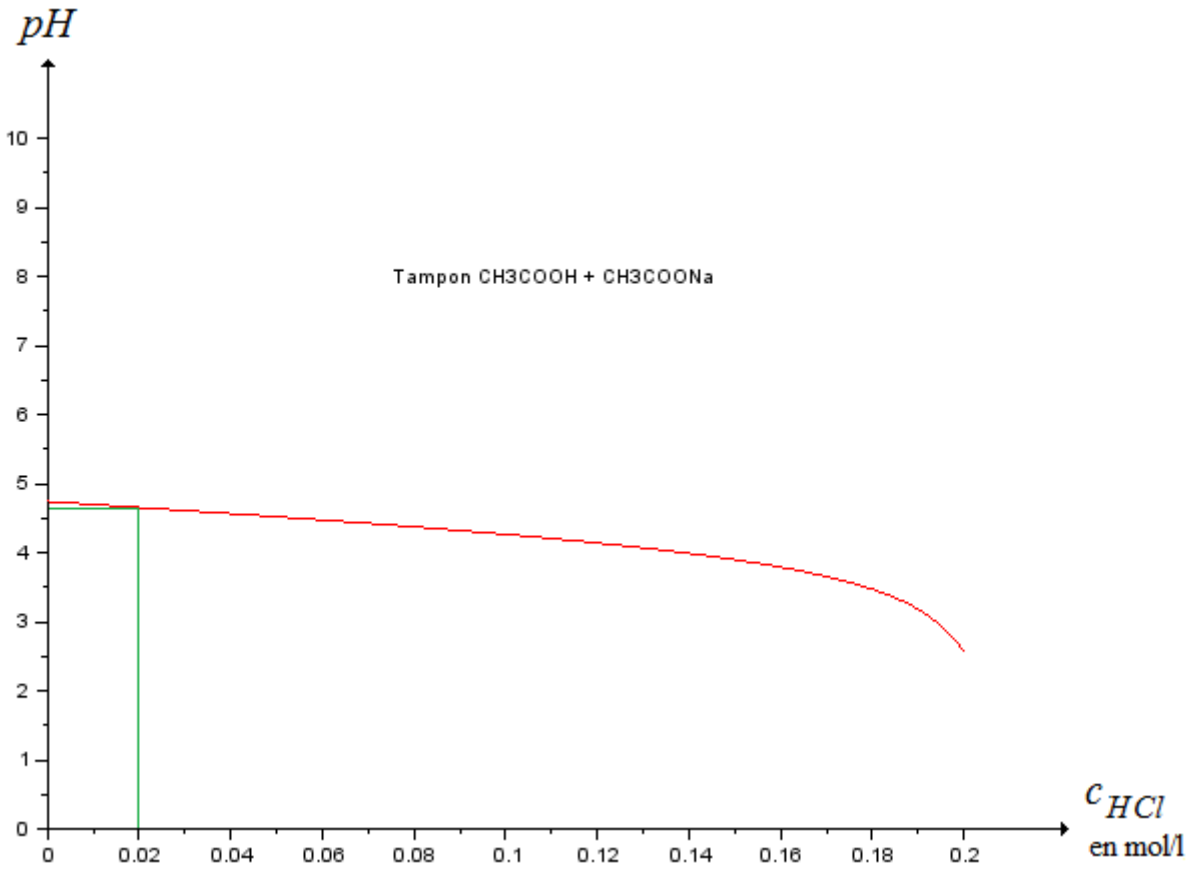
Pour $c_1 = c_2$ = 0 mol/l et c_3 = 0 mol/l, on trouve *pH* = 7
Pour $c_1 = c_2$ = 0 mol/l et c_3 = 0.02 mol/l, on trouve *pH* = 1.70

L'ajout d'acide (*HCl*) fait peu varier le *pH* de la solution ($CH_3COOH + CH_3COONa + H_2O$) : le *pH* passe de 4.76 à 4.67, soit une variation de seulement 0.09 unités. Si la même quantité d'acide avait été ajoutée à de l'eau pure, le *pH* serait passé de 7 à 1.70, soit une variation de 5.3 unités.

Mathématique & Informatique

Traçons la courbe $pH = f(c_{HCl})$ exprimant le pH de la solution aqueuse d'acide acétique et d'acétate de sodium, de même molarité $c_1 = c_2 = 0.2$ mol/l, additionnée d'acide chlorhydrique de concentration c_3 variant de 0 à 0.2 mol/l.

```
Ka = 10^(-4.74) ;
Ke = 10^(-14) ;
c1 = 0.2 ;
c2 = 0.2 ;
cmin = 0 ;
cmax = 0.2 ;
n = 300 ;
c3 = linspace(cmin,cmax,n) ;
pH = zeros(1,n) ;
for i = 1:n
    H = poly(0,'H') ;
    P = H^3 + (Ka + c2 - c3(i))*H^2 - (Ke + c1*Ka + c3(i)*Ka)*H - Ka*Ke ;
    R = roots(P) ;
    logR = - log10(R) ;
    pH(i) = logR(find(imag(logR) == 0)) ;
end
xtitle("Tampon CH3COOH + CH3COONa")
plot2d(c3,pH,frameflag = 1,rect = [cmin,0,cmax,10],style = 5)
```



Remarque

La molécule de sel CH_3COONa n'existe pas en tant que telle, en réalité on est en présence d'ions CH_3COO^- et Na^+.
La notation CH_3COONa est donc une commodité d'écriture qu'il serait plus exact de remplacer par $CH_3COO^- Na^+$.
De même, il n'y a pas de "molécule" de chlorure de sodium $NaCl$, il s'agit en fait d'un édifice cristallin constitué d'un grand nombre d'ions Na^+ et Cl^- régulièrement organisés dans l'espace. D'une manière générale, dans les équations et les calculs associés, pour représenter un système ionique (A^{a-},B^{b+}) on adoptera la formule stœchiométrique $A_\alpha B_\beta$ avec $\alpha a + \beta b = 0$.

Mathématique & Informatique

pH et solubilité

Soit une solution aqueuse de chlorure d'ammonium NH_4Cl, de molarité c, saturée en hydroxyde de magnésium $Mg(OH)_2$.
Quelle est la valeur du pH ?

---

1) Les réactions chimiques

$$\begin{cases} NH_4Cl \rightarrow NH_4^+ + Cl^- \\ NH_4^+ \rightleftarrows H^+ + NH_3 \\ Mg(OH)_2 \rightleftarrows Mg^{2+} + 2OH^- \\ H_2O \rightleftarrows H^+ + OH^- \end{cases}$$

2) Les équations algébriques

$$\begin{cases} \text{---------- } \textit{Équilibres} \\ \dfrac{[H^+][NH_3]}{[NH_4^+]} = K_a \\ [Mg^{2+}][OH^-]^2 = K_s \\ [H^+][OH^-] = K_e \\ \text{---------- } \textit{Bilans} \\ [Cl^-] = c \\ [NH_4^+] + [NH_3] = c \\ \text{---------- } \textit{Électroneutralité} \\ [NH_4^+] + [H^+] + 2[Mg^{2+}] = [Cl^-] + [OH^-] \end{cases}$$

$$\Rightarrow \quad 2K_s[H^+]^4 + (K_e^2 + 2K_aK_s)[H^+]^3 + K_aK_e^2[H^+]^2 - (K_e^3 + cK_aK_e^2)[H^+] - K_aK_e^3 = 0$$

Application numérique

Données : $\begin{cases} K_a = 10^{-9.25} & \text{(constante d'acidité du couple ammonium/ammoniac)} \\ K_s = 10^{-10.82} & \text{(produit de solubilité de l'hydroxyde de magnésium)} \\ K_e = 10^{-14} & \text{(produit ionique de l'eau)} \\ c = 0.1\,\text{mol/l} & \text{(molarité en chlorure d'ammonium)} \end{cases}$

```
Ka = 10^(-9.25) ;
Ks = 10^(-10.82) ;
Ke = 10^(-14) ;
c = 0.1 ;
H = poly(0,'H') ;
P = 2*Ks*H^4 + (Ke^2 + 2*Ka*Ks)*H^3 + Ka*Ke^2*H^2 - (Ke^3 + c*Ka*Ke^2)*H - Ka*Ke^3 ;
R = roots(P) ;
logR = - log10(R) ;
pH = logR(find(imag(logR) == 0)) ;
disp(pH)
```

Résultat : $pH = 9.36$

Le pH influe sur la solubilité s de l'hydroxyde de magnésium : $s = [Mg^{2+}] = K_s / (K_e\, 10^{pH})^2$

Ainsi, s ↗ quand pH ↘ : plus le milieu est acide, plus la magnésie est soluble.

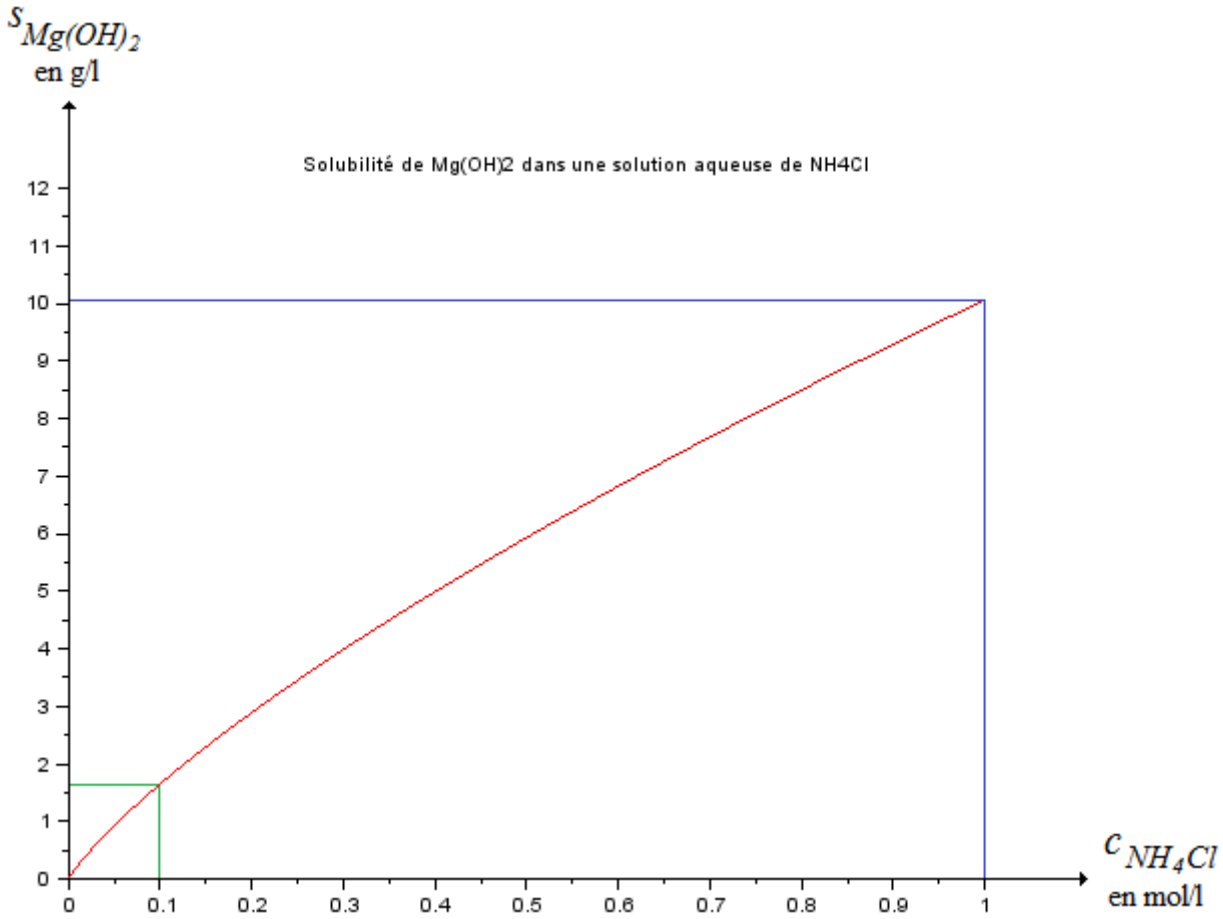
A l'inverse, s ↘ quand pH ↗ : plus le milieu est basique, moins la magnésie est soluble.

En particulier, pour c = 0.1 mol/l, le pH vaut 9.36 (basique) et s est égale à 0.028 mol/l = 1.65 g/l (peu soluble).

Mathématique & Informatique

Traçons la courbe $s = f(c)$ exprimant la solubilité s (en g/l) de l'hydroxyde de magnésium en fonction de la concentration c (en mol/l) du chlorure d'ammonium dans la solution aqueuse. La masse molaire de $Mg(OH)_2$ est M = 58.3197 g.

```
Ka = 10^(-9.25) ;
Ks = 10^(-10.82) ;
Ke = 10^(-14) ;
M = 58.3197 ;
cmin = 0 ;
cmax = 1 ;
n = 300 ;
c = linspace(cmin,cmax,n) ;
s = zeros(1,n) ;
for i = 1:n
    H = poly(0,'H') ;
    P = 2*Ks*H^4 + (Ke^2 + 2*Ka*Ks)*H^3 + Ka*Ke^2*H^2 - (Ke^3 + c(i)*Ka*Ke^2)*H - Ka*Ke^3 ;
    R = roots(P) ;
    cH = R(find(real(R) > 0 & imag(R) == 0)) ;
    s(i) = Ks*(cH/Ke)^2*M ;
end
xtitle("Solubilité de Mg(OH)2 dans une solution aqueuse de NH4Cl")
plot2d(c,s,style = 5)
```



La solubilité de $Mg(OH)_2$ dans l'eau pure ($c_{NH4Cl} = 0$) est très faible : le calcul donne $s_{Mg(OH)2} = 0.009$ g/l.
En ajoutant 1 mole de NH_4Cl par litre de solution ($c_{NH4Cl} = 1$) cette solubilité est multipliée par 1000 : $s_{Mg(OH)2} = 10.06$ g/l.

Mathématique & Informatique

pH et complexation

Quel est le *pH* d'une solution aqueuse de liqueur de **Schweizer**(*) de molarité c_1 en $Cu(NH_3)_4^{2+}$, additionnée d'acide chlorhydrique *HCl* de molarité c_2 ?

(*) appelée aussi "liqueur céleste" en raison de sa couleur bleue.

L'acide chlorhydrique détruit la liqueur de **Schweizer** : le complexe $Cu(NH_3)_4^{2+}$ se dissocie et il y a libération d'ions Cu^{2+} et de molécules d'ammoniac NH_3 qui vont capter des protons H^+ pour former des ions ammonium NH_4^+.

1) Les réactions chimiques

$$\begin{cases} HCl \rightarrow H^+ + Cl^- \\ Cu(NH_3)_4^{2+} \rightleftarrows Cu^{2+} + 4NH_3 \\ NH_4^+ \rightleftarrows H^+ + NH_3 \\ H_2O \rightleftarrows H^+ + OH^- \end{cases}$$

2) Les équations algébriques

$$\begin{cases} \text{---------- } \textit{Équilibres} \\ \dfrac{[Cu^{2+}][NH_3]^4}{[Cu(NH_3)_4^{2+}]} = K_d \\ \dfrac{[H^+][NH_3]}{[NH_4^+]} = K_a \\ [H^+][OH^-] = K_e \\ \text{---------- } \textit{Bilans} \\ [Cl^-] = c_2 \\ [Cu(NH_3)_4^{2+}] + [Cu^{2+}] = c_1 \\ [NH_3] + [NH_4^+] = 4[Cu^{2+}] \\ \text{---------- } \textit{Électroneutralité} \\ [NH_4^+] + [H^+] + 2[Cu^{2+}] + 2[Cu(NH_3)_4^{2+}] = [Cl^-] + [OH^-] \end{cases}$$

3) L'équation du *pH*

Pour déterminer le *pH*, il faut rechercher la concentration en ions H^+ et appliquer la formule de définition :

$$pH = -\log_{10}[H^+]$$

On a successivement :

- $[Cl^-] = c_2$
- $[OH^-] = \dfrac{K_e}{[H^+]}$
- $[NH_4^+] + [H^+] + 2c_1 = c_2 + \dfrac{K_e}{[H^+]} \Rightarrow [NH_4^+] = \dfrac{-[H^+]^2 - (2c_1 - c_2)[H^+] + K_e}{[H^+]}$

- $\left[NH_3\right]=\dfrac{K_a\left[NH_4^+\right]}{\left[H^+\right]}=\dfrac{-K_a\left[H^+\right]^2-(2c_1-c_2)K_a\left[H^+\right]+K_aK_e}{\left[H^+\right]^2}$
- $\left[Cu^{2+}\right]\left[NH_3\right]^4=K_d\left[Cu(NH_3)_4^{2+}\right]$

  $\dfrac{1}{4}\left(\left[NH_3\right]+\left[NH_4^+\right]\right)\left[NH_3\right]^4=K_d\left\{c_1-\dfrac{1}{4}\left(\left[NH_3\right]+\left[NH_4^+\right]\right)\right\}$

  $\dfrac{1}{4}\left(\left[NH_3\right]+\left[NH_4^+\right]\right)\left(\left[NH_3\right]^4+K_d\right)=c_1K_d$

Dans cette dernière relation, remplaçons $\left[NH_3\right]$ et $\left[NH_4^+\right]$ par leurs expressions en fonction de $\left[H^+\right]$:

$$\frac{1}{4}\left\{\left(\frac{-K_a\left[H^+\right]^2-(2c_1-c_2)K_a\left[H^+\right]+K_aK_e}{\left[H^+\right]^2}\right)+\left(\frac{-\left[H^+\right]^2-(2c_1-c_2)\left[H^+\right]+K_e}{\left[H^+\right]}\right)\right\}\left\{\left(\frac{-K_a\left[H^+\right]^2-(2c_1-c_2)K_a\left[H^+\right]+K_aK_e}{\left[H^+\right]^2}\right)^4+K_d\right\}=c_1K_d$$

Introduisons le polynôme $P=\left[H^+\right]^2+(2c_1-c_2)\left[H^+\right]-K_e$:

$$\frac{1}{4}\left\{\left(\frac{-K_aP}{\left[H^+\right]^2}\right)+\left(\frac{-P}{\left[H^+\right]}\right)\right\}\left\{\left(\frac{-K_aP}{\left[H^+\right]^2}\right)^4+K_d\right\}=c_1K_d$$

L'équation s'écrit :

$$\left(\left[H^+\right]+K_a\right)P\left(K_a^4P^4+K_d\left[H^+\right]^8\right)+4c_1K_d\left[H^+\right]^{10}=0$$

Soit, de manière équivalente :

$$\boxed{\left(\left[H^+\right]+K_a\right)\left(\left[H^+\right]^2+(2c_1-c_2)\left[H^+\right]-K_e\right)\left(K_a^4\left(\left[H^+\right]^2+(2c_1-c_2)\left[H^+\right]-K_e\right)^4+K_d\left[H^+\right]^8\right)+4c_1K_d\left[H^+\right]^{10}=0}$$

Il s'agit d'une équation polynomiale (E) de degré 11 en $\left[H^+\right]$: $\left(K_a^4+K_d\right)\left[H^+\right]^{11}+\cdots+\lambda_k\left[H^+\right]^k+\cdots-K_a^5K_e^5=0$
Elle possède 11 racines dans $\mathbb{C}$. Le *pH* est égal au cologarithme décimal de la racine réelle strictement positive.
Cette racine existe toujours puisque le produit des coefficients extrêmes de (E) est négatif : $-\left(K_a^4+K_d\right)K_a^5K_e^5<0$.
Précisons qu'il n'est pas nécessaire de développer l'équation (E) afin d'exprimer les coefficients λ_k des termes en $\left[H^+\right]^k$.
En effet, comme le logiciel *Scilab* permet d'effectuer les opérations habituelles d'addition, de multiplication, de puissance sur les polynômes, le calcul des racines de l'équation (E) sera réalisé en appliquant la fonction *roots* à la forme abrégée $\left(\left[H^+\right]+K_a\right)P\left(K_a^4P^4+K_d\left[H^+\right]^8\right)+4c_1K_d\left[H^+\right]^{10}$ après avoir défini le polynôme $P=\left[H^+\right]^2+(2c_1-c_2)\left[H^+\right]-K_e$.

Remarque

Un ion complexe est noté $[AB_n]^q$: *A* est le cation métallique central, *B* est le ligand, *n* est la coordinence et *q* est la charge.
Par exemple, l'ion tétraamminecuivre (II) est symbolisé par $[Cu(NH_3)_4]^{2+}$. Ici, nous n'utiliserons pas cette écriture qui présente une ambiguïté avec la notation $[\cdots]$ habituellement réservée à la molarité, c'est-à-dire à la concentration molaire exprimée en moles de soluté par litre de solution (mol/l). Un ion complexe sera noté AB_n^q et sa molarité $\left[AB_n^q\right]$.
En particulier, l'ion tétraamminecuivre (II) sera symbolisé par $Cu(NH_3)_4^{2+}$ et sa molarité par $\left[Cu(NH_3)_4^{2+}\right]$.

Application numérique

Données : $\begin{cases} K_d = 10^{-12.59} & \text{(constante de dissociation de l'ion tétraamminecuivre (II))} \\ K_a = 10^{-9.25} & \text{(constante d'acidité de l'ion ammonium)} \\ K_e = 10^{-14} & \text{(produit ionique de l'eau)} \\ c_1 = 0.1\,\text{mol/l} & \text{(concentration molaire initiale en } Cu(NH_3)_4^{2+}\text{)} \\ c_2 = 0.5\,\text{mol/l} & \text{(concentration molaire initiale en } HCl\text{)} \end{cases}$

Programme de calcul du pH :

```
Kd = 10^(-12.59) ;
Ka = 10^(-9.25) ;
Ke = 10^(-14) ;
c1 = 0.1 ;
c2 = 0.5 ;
H = poly(0,'H') ;
P = H^2 + (2*c1 - c2)*H - Ke ;
E = (H + Ka)*P*(Ka^4*P^4 + Kd*H^8) + 4*c1*Kd*H^10 ;
R = roots(E) ;
logR = - log10(R) ;
pH = logR(find(imag(logR) == 0)) ;
disp(pH)
```

Résultat :

$pH = 6.51$

Recherchons la proportion d'ions complexes $Cu(NH_3)_4^{2+}$ qui ont été dissociés.

Notons α le coefficient de dissociation relatif aux ions tétraamminecuivre (II) : $\alpha = \dfrac{nombre\ de\ moles\ dissociées}{nombre\ de\ moles\ initiales}$

La réaction $Cu(NH_3)_4^{2+} \rightleftarrows Cu^{2+} + 4NH_3$ donne le tableau suivant :

	$\left[Cu(NH_3)_4^{2+}\right]$	$\left[Cu^{2+}\right]$
Etat initial	c_1	0
Etat d'équilibre	$(1-\alpha)c_1$	αc_1

D'où, à l'état d'équilibre : $r = \dfrac{\left[Cu(NH_3)_4^{2+}\right]}{\left[Cu^{2+}\right]} = \dfrac{1-\alpha}{\alpha} \Rightarrow \alpha = \dfrac{1}{1+r}$

On a :

$$\left[H^+\right] = 10^{-pH} = 3\,10^{-7}\,\text{mol/l}$$

$$\left[NH_3\right] = \frac{-K_a\left[H^+\right]^2 - (2c_1 - c_2)K_a\left[H^+\right] + K_aK_e}{\left[H^+\right]^2} = 5.4\,10^{-4}\,\text{mol/l}$$

$$\frac{\left[Cu(NH_3)_4^{2+}\right]}{\left[Cu^{2+}\right]} = \frac{\left[NH_3\right]^4}{K_d} = 0.331$$

Donc, à $pH = 6.51$, $\dfrac{1}{1+0.331} \approx \dfrac{3}{4}$ de l'ensemble des ions $Cu(NH_3)_4^{2+}$ de la solution aqueuse initiale ont été dissociés.

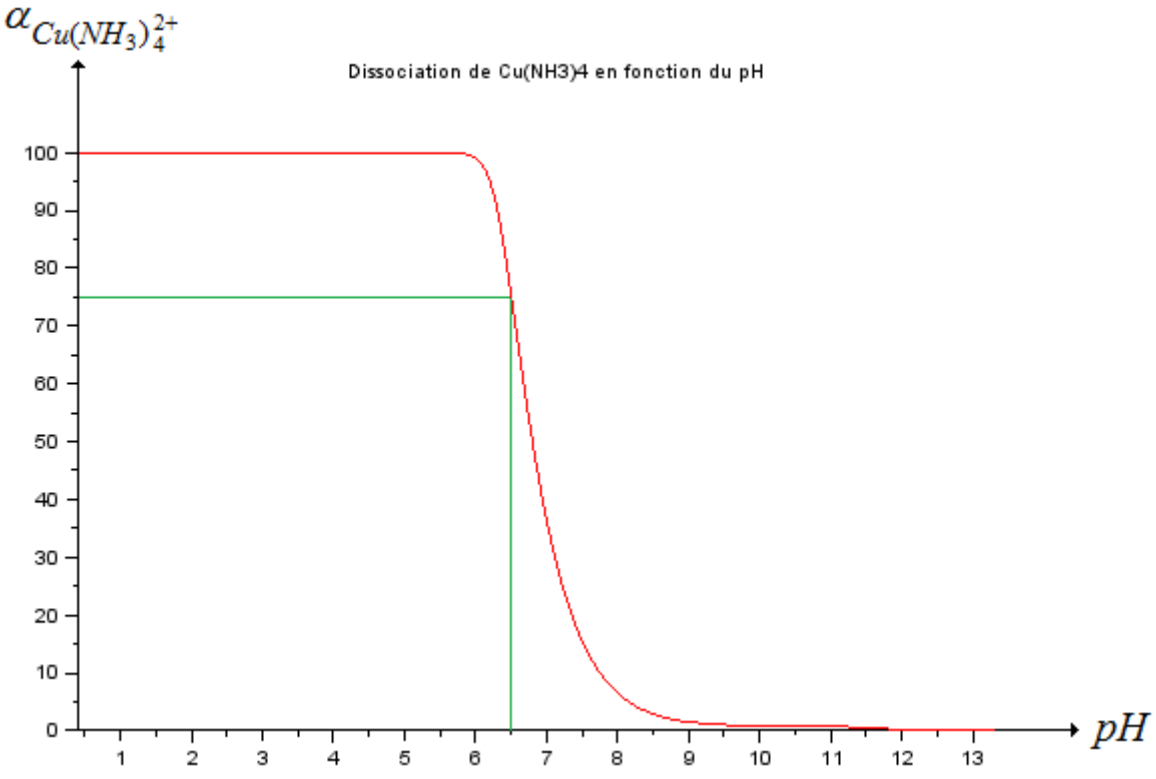
Si l'on acidifie davantage la solution, c'est-à-dire si l'on augmente la concentration en acide chlorhydrique, pratiquement tous les cations tétraamminecuivre (II) vont disparaître. Par exemple, avec $c_2 = 0.6$ mol/l, on obtient $pH = 4.82$ et on en déduit que plus de 99.99 % des ions complexes $Cu(NH_3)_4^{2+}$ ont disparu.

Notons c_i , $1 \leq i \leq 4$, les concentrations molaires suivantes :

$c_1 = c_{Cu(NH_3)_4^{2+}}$
$c_2 = c_{HCl}$
$c_3 = c_{H^+}$
$c_4 = c_{NH_3}$

Traçons la courbe donnant le pourcentage α de dissociation des ions tétraamminecuivre (II) en fonction du *pH* de la solution :

```
Kd = 10^(-12.59) ;
Ka = 10^(-9.25) ;
Ke = 10^(-14) ;
c1 = 0.1 ;
cmin = 0 ;
cmax = 1 ;
n = 300 ;
c2 = linspace(cmin,cmax,n) ;
pH = zeros(1,n) ;
alpha = zeros(1,n) ;
for i = 1:n
    H = poly(0,'H') ;
    P = H^2 + (2*c1 - c2(i))*H - Ke ;
    E = (H + Ka)*P*(Ka^4*P^4 + Kd*H^8) + 4*c1*Kd*H^10 ;
    R = roots(E) ;
    c3 = R(find(real(R) > 0 & imag(R) == 0)) ;
    c4 = (-Ka*c3^2 - (2*c1 - c2(i))*Ka*c3 + Ka*Ke)/c3^2 ;
    r = c4^4/Kd ;
    pH(i) = - log10(c3) ;
    alpha(i) = 100/(1 + r) ;
end
xtitle("Dissociation de Cu(NH3)4 en fonction du pH")
plot2d(pH,alpha,frameflag = 1,rect = [pH(n),0,pH(1),100],style = 5)
```



Mathématique & Informatique

(*) Activité chimique

Les calculs précédents ne s'appliquent théoriquement qu'à une solution ionique idéale, c'est-à-dire infiniment diluée.
Pour une solution ionique réelle, en raison des interactions entre les différentes espèces chimiques, il est nécessaire d'apporter une correction aux valeurs des concentrations. Ainsi, dans l'expression de la constante d'équilibre *K* (loi d'action et de masse) et dans la formule du *pH*, on remplace la concentration $\left[X^{z\pm}\right]$ d'un ion *X* de charge *z* (positif ou négatif) par l'activité $a\left(X^{z\pm}\right)$.
On pose :

$$a\left(X^{z\pm}\right)=\gamma_{X^{z\pm}}\left[X^{z\pm}\right]$$

où $\gamma_{X^{z\pm}}$ est le coefficient d'activité (sans dimension) vérifiant : $\gamma_{X^{z\pm}}\to 1$ quand $\left[X^{z\pm}\right]\to 0$

Exemples

- Le *pH* d'une solution est défini par $pH=-\log_{10} a\left(H^+\right)$
- La constante thermodynamique *K* régissant l'équilibre $AH \rightleftarrows H^+ + A^-$ s'écrit $K=\dfrac{a\left(H^+\right)a\left(A^-\right)}{a(AH)}=\dfrac{\left[H^+\right]\left[A^-\right]}{[AH]}\times\dfrac{\gamma_{H^+}\gamma_{A^-}}{\gamma_{AH}}$
- L'activité de l'eau (solvant) est par convention $a(H_2O)=1$ ($[H_2O]=\dfrac{1000}{18}\approx 55.55$ mol/l fait partie de la constante *K*)

On appelle force ionique d'une solution, la demi-somme des produits de la molarité de chaque ion présent par le carré de sa charge :

$$I=\frac{1}{2}\sum_{X^{z\pm}}\left[X^{z\pm}\right]z^2$$

La grandeur *I* s'exprime en mol/l (ou M), elle caractérise l'atmosphère ionique de la solution.
Comme le coefficient d'activité $\gamma_{X^{z\pm}}$ n'est pas mesurable pour une espèce chimique seule, on utilise un coefficient d'activité moyen noté $\gamma_{\pm}$ qui se déduit de la force ionique *I*. On écrit :

$$a\left(X^{z\pm}\right)=\gamma_{\pm}\left[X^{z\pm}\right]$$

Les principales expressions permettant le calcul du coefficient $\gamma_{\pm}$ pour un ion ($z \neq 0$) sont :

- $\log_{10}\gamma_{\pm}=-0.5z^2\sqrt{I}$ pour $I < 0.005$ mol/l (formule de **Debye** et **Hückel**)
- $\log_{10}\gamma_{\pm}=-0.5z^2\dfrac{\sqrt{I}}{1+\sqrt{I}}$ pour $I < 0.1$ mol/l (formule de **Güntelberg**)
- $\log_{10}\gamma_{\pm}=-0.51z^2\left(\dfrac{\sqrt{I}}{1+\sqrt{I}}-0.3I\right)$ pour $I < 0.5$ mol/l (formule de **Davies**)

D'autres formules, plus complexes, ont été développées pour des solutions plus concentrées (modèle de **Pitzer**).

Dans le cas d'une molécule non dissociée ($z = 0$), le coefficient $\gamma_{\pm}$ est proche de 1. Plus précisément, on prend :

- $\log_{10}\gamma_{\pm}=0.11\,I$ pour une molécule polaire
- $\log_{10}\gamma_{\pm}=0$ pour une molécule apolaire

Plus d'explications au sujet des coefficients d'activité dans le document *Équilibres chimiques et électrochimiques en solution aqueuse* (extraits) accessible en suivant le lien ⇨ **http://electrochimie.minatec.grenoble-inp.fr/Cours.pdf**

Application numérique

Calculons le *pH* d'une solution décimolaire d'acide chlorhydrique.

Les ions présents dans la solution aqueuse sont H^+ , Cl^- et OH^-.
Dans une solution d'acide chlorhydrique 0.1M, les molarités des ions H^+ et OH^- issus de $H_2O \rightleftarrows H^+ + OH^-$ (faible dissociation) sont négligeables par rapport aux molarités des ions H^+ et Cl^- issus de $HCl \rightarrow H^+ + Cl^-$ (acide fort, entièrement dissocié). Par suite, on ne tient pas compte de l'ionisation de l'eau et on a :

$$\left[H^+\right] \approx \left[Cl^-\right] = 0.1 \text{ mol/l}$$

La force ionique de la solution est :

$$I = \frac{1}{2}\left\{\left[H^+\right] + \left[Cl^-\right] + \left[OH^-\right]\right\} \approx \frac{1}{2}\{0.1 + 0.1\} = 0.1 \text{ mol/l}$$

Comme $I < 0.5$ mol/l, la formule de **Davies** est applicable. Le coefficient d'activité moyen $\gamma_\pm$ de l'ion H^+ vérifie :

$$\log_{10}\gamma_\pm = -0.51 z^2 \left(\frac{\sqrt{I}}{1+\sqrt{I}} - 0.3I\right)$$

On en déduit sa valeur :

$$\gamma_\pm = 10^{-0.51\left(\frac{\sqrt{0.1}}{1+\sqrt{0.1}} - 0.3\times 0.1\right)} = 0.78$$

D'où :

$$pH = -\log_{10} a\left(H^+\right) = -\log_{10}\left\{\gamma_\pm \left[H^+\right]\right\} \approx -\log_{10}\{0.78 \times 0.1\} = 1.11$$

La formule simplifiée (sans faire intervenir le coefficient d'activité) donne :

$$pH = -\log_{10}\left[H^+\right] = -\log_{10}\{0.1\} = 1$$

Les valeurs expérimentales du *pH* pour des concentrations molaires croissantes en *HCl* sont les suivantes :

Concentration molaire en *HCl*	Mesure du *pH*
0.00050	3.31
0.0100	2.04
0.100	1.10
0.40	0.52

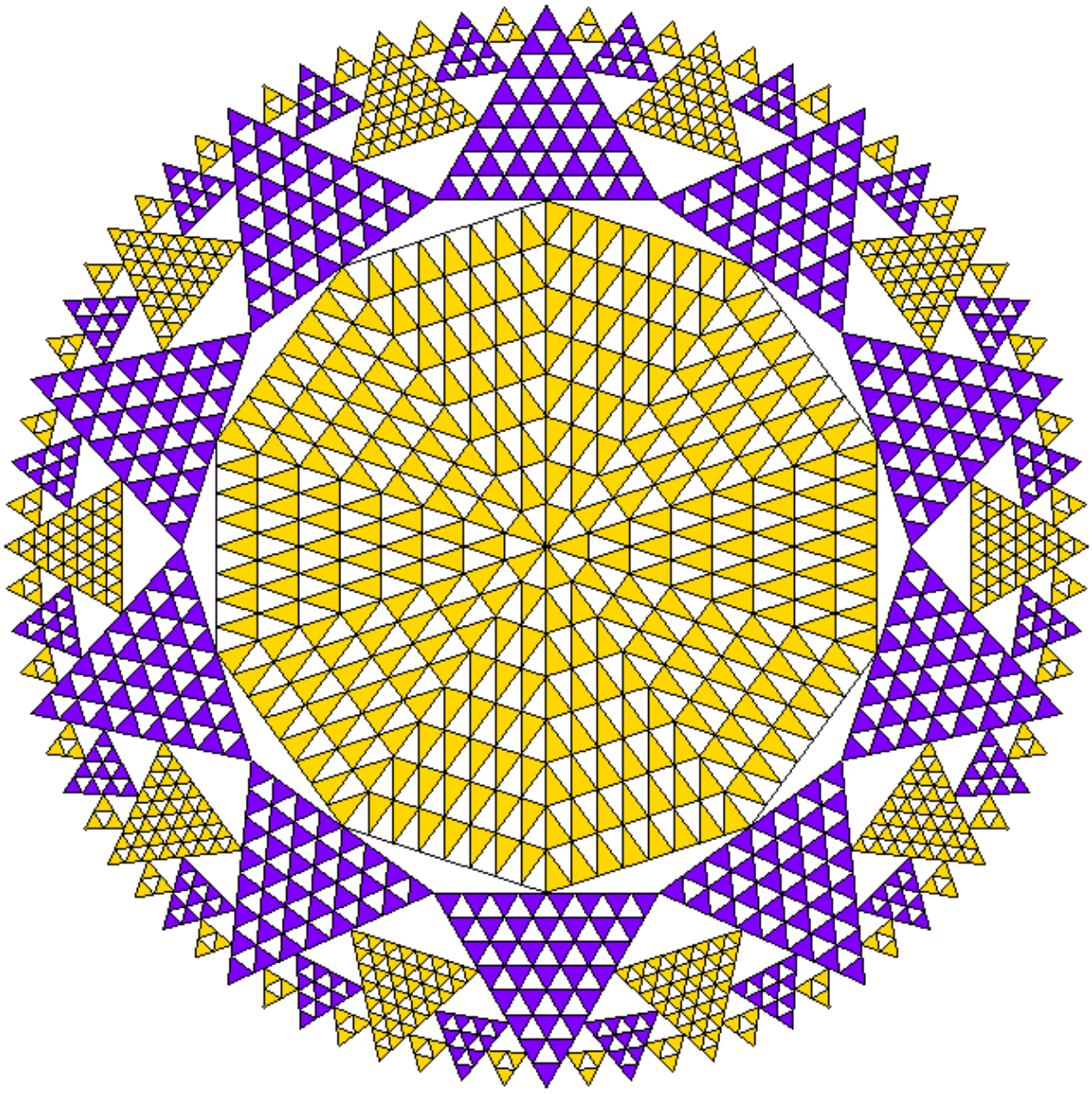
D'après ce tableau, la mesure du *pH* d'une solution d'acide chlorhydrique 0.1M est de 1.10.
L'expression $pH = -\log_{10}\left[H^+\right] = 1.00$ présente à un écart d'environ 10% avec la valeur expérimentale.
L'expression $pH = -\log_{10} a\left(H^+\right) = 1.11$ présente à un écart d'environ 1% avec la valeur expérimentale.

Regroupons les résultats obtenus pour différentes molarités en acide chlorhydrique :

Molarité en *HCl*	*pH* mesuré	*pH* calculé sans $\gamma_\pm$	*pH* calculé avec $\gamma_\pm$
0.00050	3.31	3.30	3.31
0.0100	2.04	2.00	2.04
0.100	1.10	1.00	1.11
0.40	0.52	0.40	0.53

Donc, l'utilisation du coefficient d'activité est d'autant plus justifiée que la solution est plus concentrée.

Mathématique & Informatique



RÉSOLUTION NUMÉRIQUE DES SYSTÈMES D'ÉQUATIONS (fonction *fsolve*)

On recherche $x \in \mathbb{R}^n$ tel que : $f(x) = 0$. La fonction f : $\mathbb{R}^n \rightarrow \mathbb{R}^n$ est connue.
L'algorithme consiste à construire une suite $x^{(p)}$ qui converge vers x, en partant d'une valeur initiale $x^{(0)} = x_0$
Le système d'équations :

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0 \\ f_2(x_1, x_2, \ldots, x_n) = 0 \\ \vdots \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) = 0 \end{cases} \qquad \text{avec } x_1^{(0)} = c_1, x_2^{(0)} = c_2, \ldots, x_n^{(0)} = c_n$$

se programme de la manière suivante :

```
function y = f(x)
    y(1) = f1 (x(1), x(2), .... , x(n))
    y(2) = f2 (x(1), x(2), .... , x(n))
    /
    /
    y(n) = fn (x(1), x(2), .... , x(n))
endfunction
x0 = [c1 ; c2 ; ... ; cn] ;
x = fsolve(x0, f) ;
```

On obtient un vecteur colonne x de dimension n tel que $f_i(x(1), x(2), \ldots, x(n)) = 0$, $\forall i = 1, \ldots, n$

Remarques

- Pour mettre un système d'équations sous la forme $n \times n$ ci-dessus, il convient parfois d'ajouter des inconnues $x_k = 0$
- En général, la valeur x_0 ne doit pas être trop éloignée de la solution exacte pour que l'algorithme converge.
- La résolution nécessite le calcul de la matrice jacobienne $J = (\partial f_i / \partial x_j)_{1 \le i, j \le n}$

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

On peut fournir explicitement cette matrice (n^2 dérivées) pour éviter son approximation et améliorer la précision.

Cas particulier

Un système linéaire $Ax = b$ de n équations à n inconnues est équivalent à :

$$f_i(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} a_{ij} x_j - b_i = 0 \qquad (i = 1, \ldots, n)$$

On définit la fonction $f(x) = Ax - b$ ainsi que la matrice jacobienne $J(x) = A$ et on prend $x_0 = 0$ (vecteur nul) :

```
A = [a11, ... , a1n ; ... ; an1, ... , ann] ;
b = [b1 ; ... ; bn] ;
deff('y = f(x)','y = A*x - b')
deff('J = jac(x)','J = A')
x0 = zeros(n,1) ;
x = fsolve(x0, f, jac) ;
```

Mathématique & Informatique

Géométrie dans l'espace-temps

L'introduction d'une dimension supplémentaire, le temps *t*, dans l'espace ordinaire (*x*,*y*,*z*) à 3 dimensions permet de créer un "espace-temps" (*x*,*y*,*z*,*t*) à 4 dimensions. Un tel espace possède de nombreuses propriétés géométriques, certaines sont la généralisation de théorèmes connus (le théorème élémentaire de **Pythagore** s'applique dans un hyperespace) mais d'autres sont sans équivalent dans un espace de dimension inférieure. Par exemple, on montre que deux quadriques (surfaces) quelconques d'équations cartésiennes $a_1x^2 + b_1y^2 + c_1z^2 + d_1t^2 = 1$ et $a_2x^2 + b_2y^2 + c_2z^2 + d_2t^2 = 1$ dans un espace à 4 dimensions ont toujours un ensemble de droites communes ce qui n'est pas le cas dans un espace à 1, 2 ou 3 dimensions.
Effectuons une application numérique de cette propriété :
soit les quadriques $x^2 + 11y^2 - 15z^2 + 23t^2 = 1$ et $4x^2 - 19y^2 + 17z^2 + t^2 = 1$; donner l'équation cartésienne d'une droite, passant par l'origine des temps et dirigée selon un vecteur unitaire, commune à ces deux surfaces.

Solution

Rappelons qu'une droite Δ dans un espace à 4 dimensions est définie par un point $M_0(x_0, y_0, z_0, t_0)$ et par un vecteur directeur $\vec{v}(x_1, y_1, z_1, t_1)$. L'équation vectorielle de Δ s'écrit : $\overrightarrow{M_0M} = \lambda\vec{v}$ où $\lambda \in \mathbb{R}$.

On en déduit les équations paramétriques de Δ : $\begin{cases} x = x_0 + \lambda x_1 \\ y = y_0 + \lambda y_1 \\ z = z_0 + \lambda z_1 \\ t = t_0 + \lambda t_1 \end{cases}$

et l'équation cartésienne de Δ : $\dfrac{x - x_0}{x_1} = \dfrac{y - y_0}{y_1} = \dfrac{z - z_0}{z_1} = \dfrac{t - t_0}{t_1}$

avec la convention que si un dénominateur est nul, le numérateur correspondant est nul lui aussi.

Donc, quel que soit le paramètre λ, il s'agit de vérifier le système d'équations suivant :

$$\begin{cases} a_1x^2 + b_1y^2 + c_1z^2 + d_1t^2 = 1 & (1) \\ a_2x^2 + b_2y^2 + c_2z^2 + d_2t^2 = 1 & (2) \\ x = x_0 + \lambda x_1 & (3) \\ y = y_0 + \lambda y_1 & (4) \\ z = z_0 + \lambda z_1 & (5) \\ t = t_0 + \lambda t_1 & (6) \end{cases}$$

Reportons les équations (3), (4), (5), (6) dans les équations (1) et (2) puis identifions les termes en λ, on obtient :

$$\begin{cases} a_1x_0^2 + b_1y_0^2 + c_1z_0^2 + d_1t_0^2 = 1 \\ a_2x_0^2 + b_2y_0^2 + c_2z_0^2 + d_2t_0^2 = 1 \\ a_1x_0x_1 + b_1y_0y_1 + c_1z_0z_1 + d_1t_0t_1 = 0 \\ a_2x_0x_1 + b_2y_0y_1 + c_2z_0z_1 + d_2t_0t_1 = 0 \\ a_1x_1^2 + b_1y_1^2 + c_1z_1^2 + d_1t_1^2 = 0 \\ a_2x_1^2 + b_2y_1^2 + c_2z_1^2 + d_2t_1^2 = 0 \end{cases}$$

où $a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2$ sont les données et $x_0, y_0, z_0, t_0, x_1, y_1, z_1, t_1$ sont les inconnues.

Par hypothèse, on impose 2 conditions supplémentaires :

- $t_0 = 0$ (Δ passe par l'origine des temps)
- $x_1^2 + y_1^2 + z_1^2 + t_1^2 = 1$ ($\vec{v}$ est unitaire : $\|\vec{v}\| = 1$)

Introduisons de nouvelles inconnues : $X_1 = x_1^2,\ X_2 = y_1^2,\ X_3 = z_1^2,\ X_4 = t_1^2,\ X_5 = x_0 x_1,\ X_6 = y_0 y_1,\ X_7 = z_0 z_1$.
On est ramené à résoudre un système non linéaire de 7 équations à 7 inconnues :

$$F(X) = 0 \quad \Leftrightarrow \quad \begin{cases} a_1 X_5^2 / X_1 + b_1 X_6^2 / X_2 + c_1 X_7^2 / X_3 - 1 = 0 & (1') \\ a_2 X_5^2 / X_1 + b_2 X_6^2 / X_2 + c_2 X_7^2 / X_3 - 1 = 0 & (2') \\ a_1 X_5 + b_1 X_6 + c_1 X_7 = 0 & (3') \\ a_2 X_5 + b_2 X_6 + c_2 X_7 = 0 & (4') \\ a_1 X_1 + b_1 X_2 + c_1 X_3 + d_1 X_4 = 0 & (5') \\ a_2 X_1 + b_2 X_2 + c_2 X_3 + d_2 X_4 = 0 & (6') \\ X_1 + X_2 + X_3 + X_4 - 1 = 0 & (7') \end{cases}$$

La matrice jacobienne des dérivées partielles est :

$$J(X) = \begin{pmatrix} -a_1 X_5^2 / X_1^2 & -b_1 X_6^2 / X_2^2 & -c_1 X_7^2 / X_3^2 & 0 & 2a_1 X_5 / X_1 & 2b_1 X_6 / X_2 & 2c_1 X_7 / X_3 \\ -a_2 X_5^2 / X_1^2 & -b_2 X_6^2 / X_2^2 & -c_2 X_7^2 / X_3^2 & 0 & 2a_2 X_5 / X_1 & 2b_2 X_6 / X_2 & 2c_2 X_7 / X_3 \\ 0 & 0 & 0 & 0 & a_1 & b_1 & c_1 \\ 0 & 0 & 0 & 0 & a_2 & b_2 & c_2 \\ a_1 & b_1 & c_1 & d_1 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & d_2 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Méthode algorithmique

On utilise la fonction *fsolve* pour trouver une solution numérique ($X_1, X_2, X_3, X_4, X_5, X_6, X_7$) lorsque les coefficients $a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2$ sont fournis. On en déduit facilement la valeur des inconnues $x_0, y_0, z_0, t_0, x_1, y_1, z_1, t_1$:

$x_1 = \sqrt{X_1},\ y_1 = \sqrt{X_2},\ z_1 = \sqrt{X_3},\ t_1 = \sqrt{X_4},\ x_0 = X_5 / x_1,\ y_0 = X_6 / y_1,\ z_0 = X_7 / z_1,\ t_0 = 0$

Concernant les paramètres d'appel de la fonction *fsolve*, le vecteur *X0* des valeurs initiales attribuées aux variables $X_1 = x_1^2$, $X_2 = y_1^2$, $X_3 = z_1^2$, $X_4 = t_1^2$, $X_5 = x_0 x_1$, $X_6 = y_0 y_1$, $X_7 = z_0 z_1$ doit être suffisamment proche d'une solution du système d'équations.
On connait un encadrement des variables X_1, X_2, X_3, X_4 d'une solution quelconque :

$x_1^2 + y_1^2 + z_1^2 + t_1^2 = 1 \quad \Rightarrow \quad |x_1| \le 1, |y_1| \le 1, |z_1| \le 1, |t_1| \le 1 \quad \Rightarrow \quad 0 \le X_1 \le 1, 0 \le X_2 \le 1, 0 \le X_3 \le 1, 0 \le X_4 \le 1$

Pour $a_1 = 1$, $b_1 = 11$, $c_1 = -15$, $d_1 = 23$, $a_2 = 4$, $b_2 = -19$, $c_2 = 17$, $d_2 = 1$ on vérifie que le vecteur $X0 = (1,1,1,1,1,1,1)$ entraîne la convergence de l'algorithme avec une erreur relative $tol < 10^{-13}$ entre la solution estimée et la solution exacte.

Méthode algébrique

On peut résoudre algébriquement le système d'équations $F(X) = 0$. Exposons la méthode sans détailler les calculs.
Les équations (5'), (6'), (7') sont linéaires et permettent d'exprimer les inconnues X_2, X_3, X_4 en fonction de X_1 sous la forme $X_2 = \alpha_2 X_1 + \beta_2$, $X_3 = \alpha_3 X_1 + \beta_3$, $X_4 = \alpha_4 X_1 + \beta_4$. De même, les équations (3') et (4') sont linéaires et permettent d'exprimer les inconnues X_6 et X_7 en fonction de X_5 sous la forme $X_6 = \gamma_6 X_5$, $X_7 = \gamma_7 X_5$. Par suite, les équations (1') et (2') sont du type $P_1(X_1)X_5^2 - Q(X_1) = 0$ et $P_2(X_1)X_5^2 - Q(X_1) = 0$ où P_1 et P_2 sont deux polynômes de degré 2 en X_1 et Q est un polynôme de degré 3 en X_1. La condition nécessaire et suffisante pour que ces deux équations du second degré en X_5 aient une racine commune conduit à une équation du second degré en X_1 : $AX_1^2 + BX_1 + C = 0$ avec

$$A = (a_1 - a_2)\alpha_2\alpha_3 + (b_1 - b_2)\alpha_3\gamma_6^2 + (c_1 - c_2)\alpha_2\gamma_7^2$$
$$B = (a_1 - a_2)(\alpha_2\beta_3 + \alpha_3\beta_2) + (b_1 - b_2)\beta_3\gamma_6^2 + (c_1 - c_2)\beta_2\gamma_7^2$$
$$C = (a_1 - a_2)\beta_2\beta_3$$

De la racine $X_1 = \left(-B + \sqrt{B^2 - 4AC}\right)/(2A)$, on déduit $X_2, X_3, X_4, X_5, X_6, X_7$ puis $x_0, y_0, z_0, t_0, x_1, y_1, z_1, t_1$.

Remarque

Pour résoudre le système d'équations $F(X) = 0$ dans le cas général avec la méthode algébrique, il faut tenir compte de tous les cas particuliers en fonction des valeurs des coefficients $a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2$. L'avantage de la méthode algorithmique est de ne pas en tenir compte, mais la difficulté réside alors dans la détermination du vecteur initial *X0*.

Mathématique & Informatique

Programme Scilab de résolution algorithmique du système d'équations

```
function Y=F(X)
    Y(1) = a1*X(5)^2/X(1) + b1*X(6)^2/X(2) + c1*X(7)^2/X(3) - 1
    Y(2) = a2*X(5)^2/X(1) + b2*X(6)^2/X(2) + c2*X(7)^2/X(3) - 1
    Y(3) = a1*X(5) + b1*X(6) + c1*X(7)
    Y(4) = a2*X(5) + b2*X(6) + c2*X(7)
    Y(5) = a1*X(1) + b1*X(2) + c1*X(3) + d1*X(4)
    Y(6) = a2*X(1) + b2*X(2) + c2*X(3) + d2*X(4)
    Y(7) = X(1) + X(2) + X(3) + X(4) - 1
endfunction
function M=J(X)
    M(1,1) = - a1*X(5)^2/X(1)^2
    M(1,2) = - b1*X(6)^2/X(2)^2
    M(1,3) = - c1*X(7)^2/X(3)^2
    M(1,4) = 0
    M(1,5) = 2*a1*X(5)/X(1)
    M(1,6) = 2*b1*X(6)/X(2)
    M(1,7) = 2*c1*X(7)/X(3)
    M(2,1) = - a2*X(5)^2/X(1)^2
    M(2,2) = - b2*X(6)^2/X(2)^2
    M(2,3) = - c2*X(7)^2/X(3)^2
    M(2,4) = 0
    M(2,5) = 2*a2*X(5)/X(1)
    M(2,6) = 2*b2*X(6)/X(2)
    M(2,7) = 2*c2*X(7)/X(3)
    M(3,1) = 0
    M(3,2) = 0
    M(3,3) = 0
    M(3,4) = 0
    M(3,5) = a1
    M(3,6) = b1
    M(3,7) = c1
    M(4,1) = 0
    M(4,2) = 0
    M(4,3) = 0
    M(4,4) = 0
    M(4,5) = a2
    M(4,6) = b2
    M(4,7) = c2
    M(5,1) = a1
    M(5,2) = b1
    M(5,3) = c1
    M(5,4) = d1
    M(5,5) = 0
    M(5,6) = 0
    M(5,7) = 0
    M(6,1) = a2
    M(6,2) = b2
    M(6,3) = c2
    M(6,4) = d2
    M(6,5) = 0
    M(6,6) = 0
    M(6,7) = 0
    M(7,1) = 1
    M(7,2) = 1
    M(7,3) = 1
    M(7,4) = 1
    M(7,5) = 0
    M(7,6) = 0
    M(7,7) = 0
endfunction
```

```
a1 = 1 ;
b1 = 11 ;
c1 = - 15 ;
d1 = 23 ;
a2 = 4 ;
b2 = - 19 ;
c2 = 17 ;
d2 = 1 ;
tol = 1D-13 ;
X0 = [1;1;1;1;1;1;1] ;
[X,v,info] = fsolve(X0,F,J,tol) ;
x1 = sqrt(X(1)) ;
y1 = sqrt(X(2)) ;
z1 = sqrt(X(3)) ;
t1 = sqrt(X(4)) ;
x0 = X(5)/x1 ;
y0 = X(6)/y1 ;
z0 = X(7)/z1 ;
t0 = 0 ;
printf('\n x0 = %.13f , x1 = %.13f',x0,x1)
printf('\n y0 = %.13f , y1 = %.13f',y0,y1)
printf('\n z0 = %.13f , z1 = %.13f',z0,z1)
printf('\n t0 = %.13f , t1 = %.13f',t0,t1)
```

Résultat de l'exécution du programme :

```
x0 = 0.9047515482329 , x1 = 0.4806520454446
y0 = 0.5627039587627 , y1 = 0.6072182400404
z0 = 0.4691530355374 , z1 = 0.5958817339868
t0 = 0.0000000000000 , t1 = 0.2125666466618
```

Par conséquent, les deux surfaces $x^2 + 11y^2 - 15z^2 + 23t^2 = 1$ et $4x^2 - 19y^2 + 17z^2 + t^2 = 1$ possèdent en commun la droite d'équation cartésienne :

$$\frac{x - 0.9047515482329}{0.4806520454446} = \frac{y - 0.5627039587627}{0.6072182400404} = \frac{z - 0.4691530355374}{0.5958817339868} = \frac{t}{0.2125666466618}$$

Mathématique & Informatique

Programme Scilab de résolution algébrique du système d'équations

```
a1 = 1 ;
b1 = 11 ;
c1 = - 15 ;
d1 = 23 ;
a2 = 4 ;
b2 = - 19 ;
c2 = 17 ;
d2 = 1 ;
M1 = inv([b1,c1,d1;b2,c2,d2;1,1,1]) ;
M2 = inv([b1,c1;b2,c2]) ;
alpha2 = M1(1,:)*[-a1;-a2;-1] ;
alpha3 = M1(2,:)*[-a1;-a2;-1] ;
alpha4 = M1(3,:)*[-a1;-a2;-1] ;
beta2 = M1(1,:)*[0;0;1] ;
beta3 = M1(2,:)*[0;0;1] ;
beta4 = M1(3,:)*[0;0;1] ;
gamma6 = M2(1,:)*[-a1;-a2] ;
gamma7 = M2(2,:)*[-a1;-a2] ;
A = (a1 - a2)*alpha2*alpha3 + (b1 - b2)*alpha3*gamma6^2 + (c1 - c2)*alpha2*gamma7^2 ;
B = (a1 - a2)*(alpha2*beta3 + alpha3*beta2) + (b1 - b2)*beta3*gamma6^2 + (c1 - c2)*beta2*gamma7^2 ;
C = (a1 - a2)*beta2*beta3 ;
X1 = (- B + sqrt(B^2 - 4*A*C))/(2*A) ;
X2 = alpha2*X1 + beta2 ;
X3 = alpha3*X1 + beta3 ;
X4 = alpha4*X1 + beta4 ;
a = a1*alpha2*alpha3 + b1*alpha3*gamma6^2 + c1*alpha2*gamma7^2 ;
b = a1*(alpha2*beta3 + alpha3*beta2) + b1*beta3*gamma6^2 + c1*beta2*gamma7^2 ;
c = a1*beta2*beta3 ;
P = a*X1^2 + b*X1 + c ;
Q = alpha2*alpha3*X1^3 + (alpha2*beta3 + alpha3*beta2)*X1^2 + beta2*beta3*X1 ;
X5 = sqrt(Q/P) ;
X6 = gamma6*X5 ;
X7 = gamma7*X5 ;
x1 = sqrt(X1) ;
y1 = sqrt(X2) ;
z1 = sqrt(X3) ;
t1 = sqrt(X4) ;
x0 = X5/x1 ;
y0 = X6/y1 ;
z0 = X7/z1 ;
t0 = 0 ;
printf('\n x0 = %.13f , x1 = %.13f',x0,x1)
printf('\n y0 = %.13f , y1 = %.13f',y0,y1)
printf('\n z0 = %.13f , z1 = %.13f',z0,z1)
printf('\n t0 = %.13f , t1 = %.13f',t0,t1)
```

Résultat de l'exécution du programme :

$x0 = 0.9047515482329$, $x1 = 0.4806520454446$
$y0 = 0.5627039587627$, $y1 = 0.6072182400404$
$z0 = 0.4691530355374$, $z1 = 0.5958817339868$
$t0 = 0.0000000000000$, $t1 = 0.2125666466618$

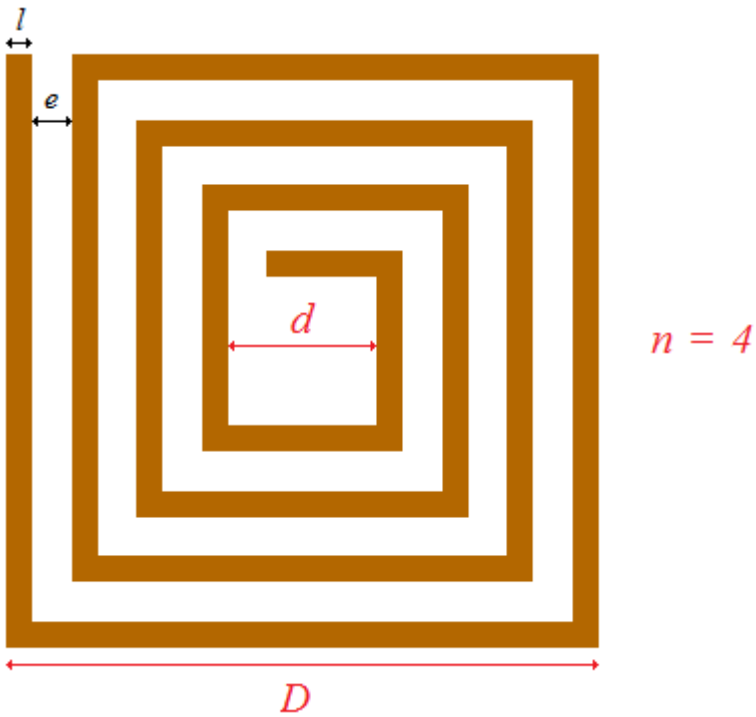
Par conséquent, les deux surfaces $x^2 + 11y^2 - 15z^2 + 23t^2 = 1$ et $4x^2 - 19y^2 + 17z^2 + t^2 = 1$ possèdent en commun la droite d'équation cartésienne :

$$\frac{x - 0.9047515482329}{0.4806520454446} = \frac{y - 0.5627039587627}{0.6072182400404} = \frac{z - 0.4691530355374}{0.5958817339868} = \frac{t}{0.2125666466618}$$

Mathématique & Informatique

Réalisation graphique d'une inductance imprimée

Il est possible de réaliser une inductance directement sur le circuit imprimé d'une carte électronique.
La technique des bobinages imprimés permet de réaliser des selfs de très faible épaisseur (couche de cuivre de 35 µm), faciles à reproduire, robustes et peu coûteuses. L'inconvénient, en pratique, est la valeur limitée de l'inductance *L* entre quelques dizaines de nH et quelques µH. Ajoutons qu'un enroulement de ce type présente une capacité répartie *C* de l'ordre de quelques pF dont il faut généralement tenir compte dans l'étude des circuits HF.



Notations :

- *n* est le nombre de spires carrées de l'inductance
- *D* est la longueur du côté extérieur de la plus grande spire
- *d* est la longueur du côté intérieur de la plus petite spire
- *l* est la largeur des spires
- *e* est l'écartement entre les spires

Lien entre les paramètres géométriques : $D = d + 2nl + 2(n - 1)e$

La formule de **Bryan** donne la valeur de l'inductance *L* :

$$L \approx \frac{3}{5} n^{5/3} (D+d) \ln\left(4 \frac{D+d}{D-d}\right)$$

Les dimensions *D* et *d* sont en millimètres, *L* est en nanohenrys.
La formule de **Bryan** est une relation empirique dérivée de celle de **Nagaoka**. Le calcul de *L* ne fournit qu'une valeur approximative de l'inductance réellement obtenue.
D'autres expressions analytiques (plus précises) de l'inductance *L* ont été établies. Citons la formule de **Terman** :

$$L \approx a_1 \frac{D^3}{(l+e)^2} \left[1 - \left(\frac{d}{D}\right)^2\right] \left[1 - \left(\frac{d}{D}\right)\right] \left[a_2 \left(\frac{D-d}{D+d}\right) + a_3 - \ln\left(\frac{D-d}{D+d}\right)\right]$$

Avec $a_1 = 0.1$, $a_2 = 0.2235$, $a_3 = 0.726$; *D*, *d*, *l*, *e* en millimètres, *L* en nanohenrys.

⇨ https://ori-nuxeo.univ-lille1.fr/nuxeo/site/esupversions/1be0afae-6454-4f5c-abb2-2aa4a0628b56

Mathématique & Informatique

Problème

L'encombrement maximal D_m réservé à une inductance imprimée sur une carte électronique est supposé connu, la largeur l des spires est fixée ainsi que la dimension d de la première spire.
Ecrire une fonction Scilab qui dessine une spirale carrée d'inductance L.

Solution 1 (formule de **Bryan**)

1) Calcul de n pour L, D_m, d donnés :

$$n = \left\lceil \left(\frac{L}{\frac{3}{5}(D_m + d)\ln\left(4\frac{D_m + d}{D_m - d}\right)} \right)^{\frac{3}{5}} \right\rceil$$

$\lceil x \rceil$ désigne la partie entière supérieure de x, $\ln(x)$ est le logarithme népérien de x.

2) Calcul de $D \le D_m$ correspondant à la valeur de n obtenue en 1) :

$$f(D) = L - \frac{3}{5} n^{5/3} (D + d) \ln\left(4\frac{D + d}{D - d}\right) = 0$$

L'équation d'inconnue D est résolue avec la fonction *fsolve* en partant de la valeur initiale D_m.

3) Calcul de e connaissant n, D, d, l :

$$e = \frac{D - d - 2nl}{2(n - 1)}$$

On en déduit la fonction Scilab :

```
function [n, D, e]=Bryan(L, Dm, d, l)
    function turn(c)
        if c > d then, xrects([D/2 - c/2 + e ; D/2 + c/2 ; l ; e] , copper), end
        xrects([D/2 - c/2 + e ; D/2 + c/2 + l ; c + l - e ; l] , copper)
        xrects([D/2 + c/2 ; D/2 + c/2 ; l ; c + l] , copper)
        xrects([D/2 - c/2 - l ; D/2 - c/2 ; c + l ; l] , copper)
        xrects([D/2 - c/2 - l ; D/2 + c/2 + l ; l ; c + l] , copper)
    endfunction
    copper = color(179,103,0)
    n = ceil((L/((3/5)*(Dm + d)*log(4*(Dm + d)/(Dm - d))))^(3/5))
    deff('y = f(D)','y = L - (3/5)*n^(5/3)*(D + d)*log(4*(D + d)/(D - d))')
    D = fsolve(Dm,f)
    e = (D - d - 2*n*l)/(2*(n - 1))
    c = d
    for k = 1:n
        turn(c)
        c = c + 2*(l + e)
    end
    plot2d(%inf,%inf,frameflag = 3,rect = [0,0,Dm,Dm])
    a = gca() ; a.x_location = 'bottom' ; a.y_location = 'left'
endfunction
```

Mathématique & Informatique

Application

Dans un récepteur VHF, pour sélectionner la fréquence intermédiaire de 10.7 Mhz, on utilise un circuit accordé constitué d'une inductance L en parallèle avec un condensateur ajustable C. On souhaite réaliser l'inductance sous la forme d'une spirale carrée imprimée répondant aux caractéristiques suivantes :

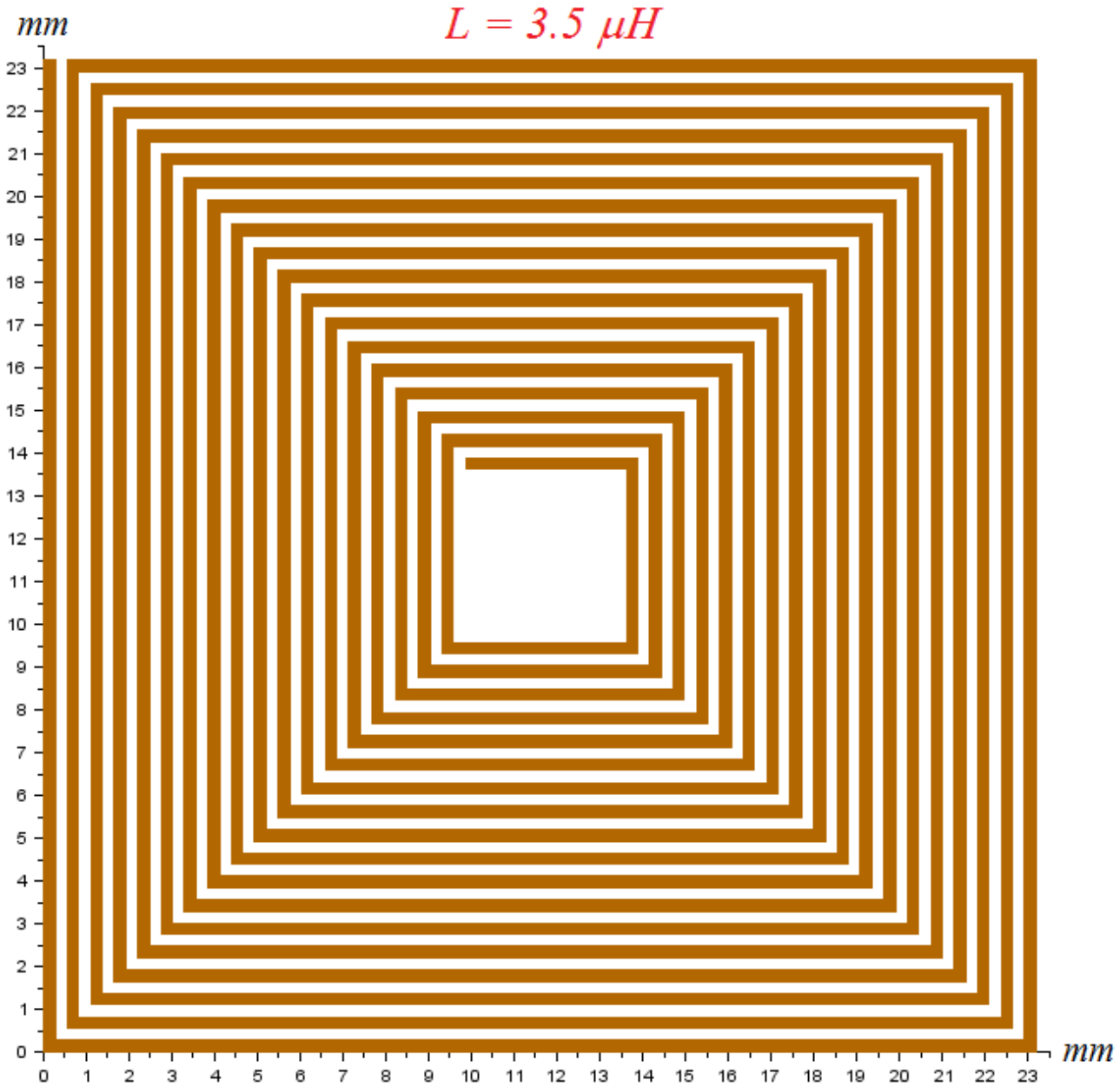
- $L = 3.5$ µH
- $D_m = 25$ mm
- $d = 4$ mm
- $l = 0.3$ mm

Le schéma du tracé s'obtient avec l'instruction :

```
[n,D,e] = Bryan(3500,25,4,0.3)
```

Résultat :

$n = 18$
$D = 23.202262$ mm
$e = 0.2471254$ mm



Solution 2 (formule de **Terman**)

1) Calcul de e pour L, D_m, d, l donnés :

$$e = \sqrt{a_1 \frac{{D_m}^3}{L}\left[1-\left(\frac{d}{D_m}\right)^2\right]\left[1-\left(\frac{d}{D_m}\right)\right]\left[a_2\left(\frac{D_m-d}{D_m+d}\right)+a_3-\ln\left(\frac{D_m-d}{D_m+d}\right)\right]} - l$$

2) Calcul de n connaissant D_m, d, l, e :

$$n = \left\lceil \frac{D_m - d + 2e}{2(l+e)} \right\rceil$$

3) Calcul de $D \le D_m$ correspondant à la valeur de n obtenue en 2) :

$$f(D) = L - a_1 \frac{D^3}{\left(l + \frac{D-d-2nl}{2(n-1)}\right)^2}\left[1-\left(\frac{d}{D}\right)^2\right]\left[1-\left(\frac{d}{D}\right)\right]\left[a_2\left(\frac{D-d}{D+d}\right)+a_3-\ln\left(\frac{D-d}{D+d}\right)\right] = 0$$

4) Calcul du nouvel écartement e avec n, D, d, l déterminés :

$$e = \frac{D-d-2nl}{2(n-1)}$$

On en déduit la fonction Scilab :

```
function [n, D, e]=Terman(L, Dm, d, l)
    function turn(c)
        if c > d then, xrects([D/2 - c/2 + e ; D/2 + c/2 ; l ; e] , copper), end
        xrects([D/2 - c/2 + e ; D/2 + c/2 + l ; c + l - e ; l] , copper)
        xrects([D/2 + c/2 ; D/2 + c/2 ; l ; c + l] , copper)
        xrects([D/2 - c/2 - l ; D/2 - c/2 ; c + l ; l] , copper)
        xrects([D/2 - c/2 - l ; D/2 + c/2 + l ; l ; c + l] , copper)
    endfunction
    copper = color(179,103,0)
    e = sqrt(0.1*(Dm^3/L)*(1 - (d/Dm)^2)*(1 - (d/Dm))*(0.2235*((Dm - d)/(Dm + d)) + 0.726 - log((Dm - d)/(Dm + d)))) - l
    n = ceil((Dm - d + 2*e)/(2*(l + e)))
    deff('y = f(D)','y = L - 0.1*(D^3/((l + (D - d - 2*n*l)/(2*(n - 1)))^2))*(1 - (d/D)^2)*(1 - (d/D))* ...
                    (0.2235*((D - d)/(D + d)) + 0.726 - log((D - d)/(D + d)))')
    D = fsolve(Dm,f)
    e = (D - d - 2*n*l)/(2*(n - 1))
    c = d
    for k = 1:n
        turn(c)
        c = c + 2*(l + e)
    end
    plot2d(%inf,%inf,frameflag = 3,rect = [0,0,Dm,Dm])
    a = gca() ; a.x_location = 'bottom' ; a.y_location = 'left'
endfunction
```

Mathématique & Informatique

Application

Dans un récepteur VHF, pour sélectionner la fréquence intermédiaire de 10.7 Mhz, on utilise un circuit accordé constitué d'une inductance L en parallèle avec un condensateur ajustable C. On souhaite réaliser l'inductance sous la forme d'une spirale carrée imprimée répondant aux caractéristiques suivantes :

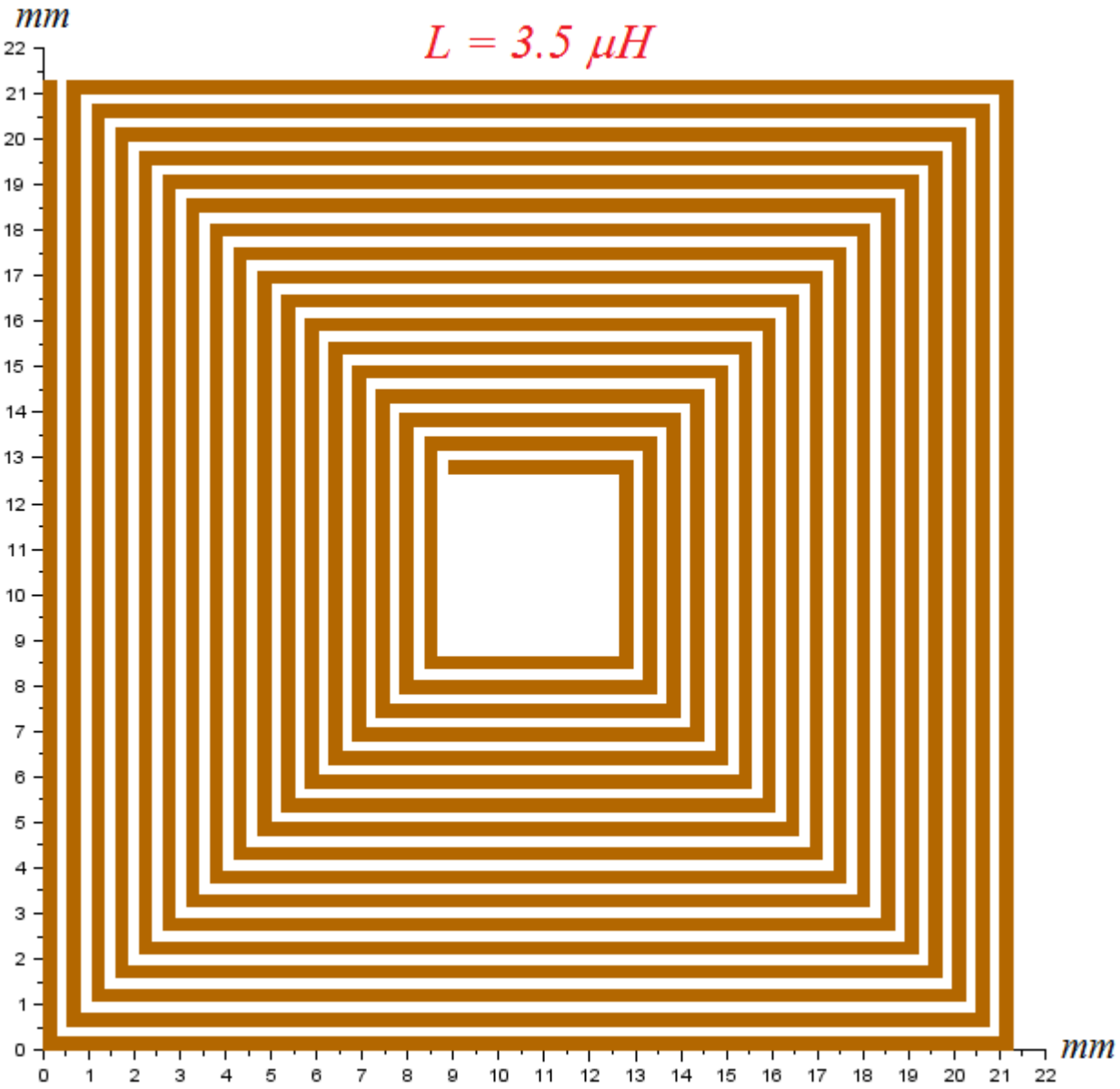
- $L = 3.5$ µH
- $D_m = 25$ mm
- $d = 4$ mm
- $l = 0.3$ mm

Le schéma du tracé s'obtient avec l'instruction :

```
[n,D,e] = Terman(3500,25,4,0.3)
```

Résultat :

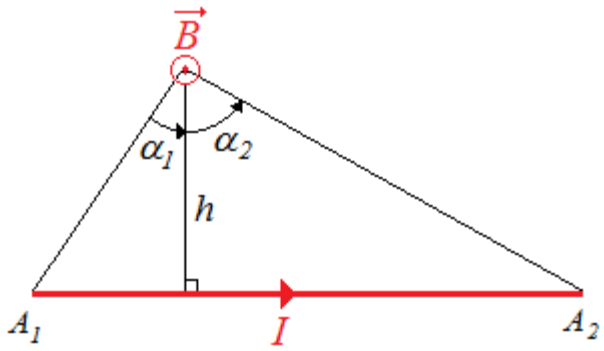
$n = 17$
$D = 21.298814$ mm
$e = 0.2218379$ mm



Mathématique & Informatique

Calcul littéral et calcul numérique de l'inductance d'une spire imprimée

1) Champ d'induction magnétique $\vec{B}$ créé par un segment linéaire parcouru par un courant I



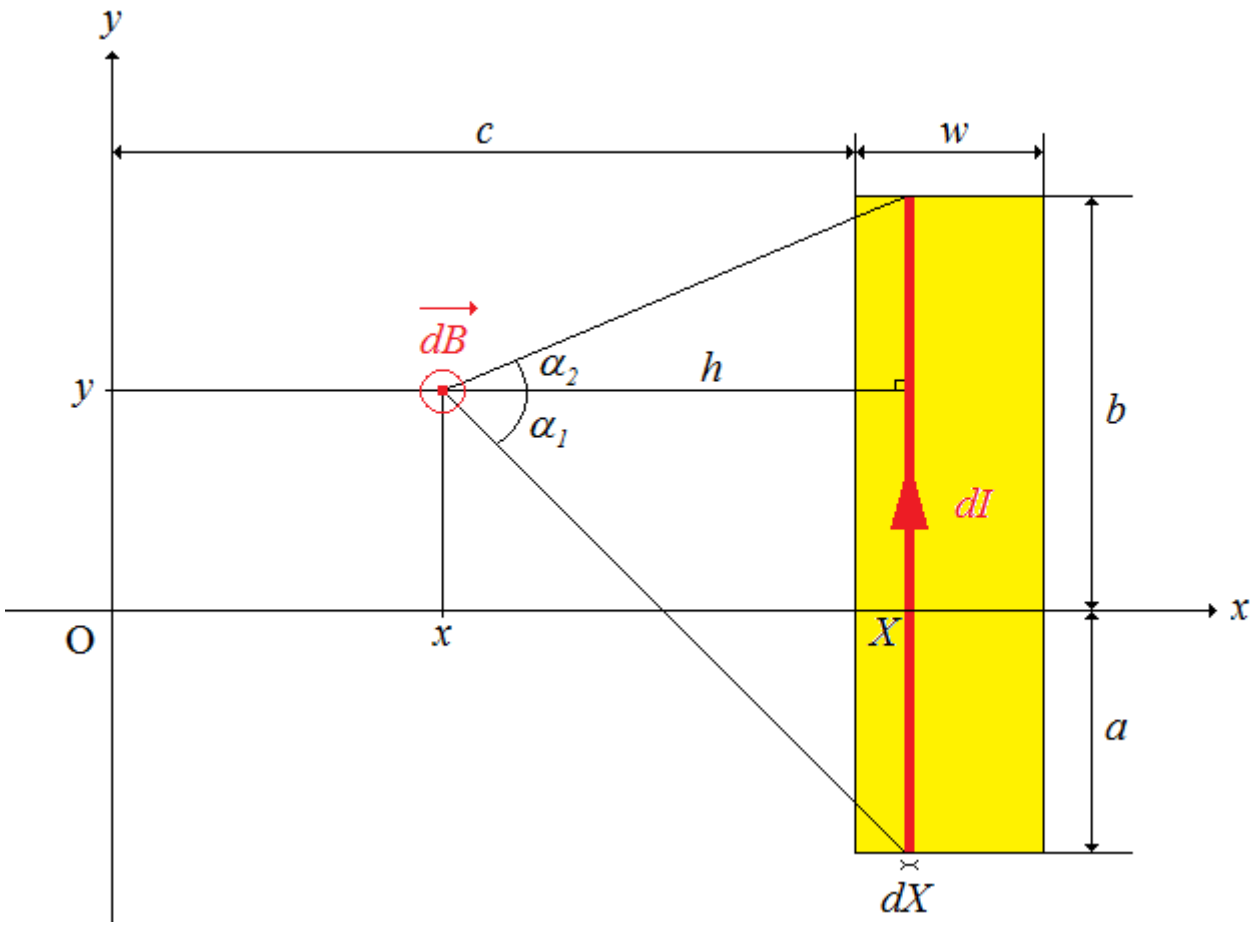
Il s'agit d'une application classique de la loi de **Biot** et **Savart** : $\overrightarrow{dB} = \frac{\mu_0}{4\pi}\frac{I}{r^2}\overrightarrow{dl}\wedge\frac{\vec{r}}{r}$

On trouve, en module : $B = \frac{\mu_0 I}{4\pi h}(\sin\alpha_1 + \sin\alpha_2)$

Pour un conducteur de longueur infinie : $\alpha_1 = \alpha_2 = \frac{\pi}{2} \Rightarrow B = \frac{\mu_0 I}{2\pi h}$

2) Champ d'induction magnétique $\vec{B}$ créé par un ruban rectangulaire très mince parcouru par un courant I

Effectuons une représentation géométrique du problème :



On suppose : $-c \le x \le c$ et $-a \le y \le b$

Le courant dI circulant dans la bande de largeur dX est donné par :

$$dI = \frac{I}{w}dX \quad \text{où} \quad \frac{I}{w} = j \quad \text{est la densité linéique de courant.}$$

Le champ élémentaire $\overrightarrow{dB}$ créé par dI en M(x,y) a pour module :

$$dB = \frac{\mu_0}{4\pi} \times \frac{\sin\alpha_1 + \sin\alpha_2}{h} \times dI$$

Du schéma, on déduit les relations suivantes :

$$h = X - x$$
$$\sin\alpha_1 = \frac{a + y}{\sqrt{(X - x)^2 + (a + y)^2}}$$
$$\sin\alpha_2 = \frac{b - y}{\sqrt{(X - x)^2 + (b - y)^2}}$$

Le champ total $\vec{B}$ créé par le courant I en un point M(x,y) a pour module :

$$B = \int dB = \frac{\mu_0 I}{4\pi w}\int_c^{c+w}\left(\frac{a + y}{\sqrt{(X - x)^2 + (a + y)^2}} + \frac{b - y}{\sqrt{(X - x)^2 + (b - y)^2}}\right)\frac{dX}{X - x}$$

Posons $u = \frac{a + y}{X - x}$ et $v = \frac{b - y}{X - x}$:

$$B = -\frac{\mu_0 I}{4\pi w}\left(\int_{\frac{a+y}{c-x}}^{\frac{a+y}{c+w-x}}\frac{du}{\sqrt{u^2 + 1}} + \int_{\frac{b-y}{c-x}}^{\frac{b-y}{c+w-x}}\frac{dv}{\sqrt{v^2 + 1}}\right) = -\frac{\mu_0 I}{4\pi w}\left([\text{Argsh}(u)]_{\frac{a+y}{c-x}}^{\frac{a+y}{c+w-x}} + [\text{Argsh}(v)]_{\frac{b-y}{c-x}}^{\frac{b-y}{c+w-x}}\right)$$

$$= \frac{\mu_0 I}{4\pi w}\left[\text{Argsh}\left(\frac{a + y}{c - x}\right) + \text{Argsh}\left(\frac{b - y}{c - x}\right) - \text{Argsh}\left(\frac{a + y}{c + w - x}\right) - \text{Argsh}\left(\frac{b - y}{c + w - x}\right)\right]$$

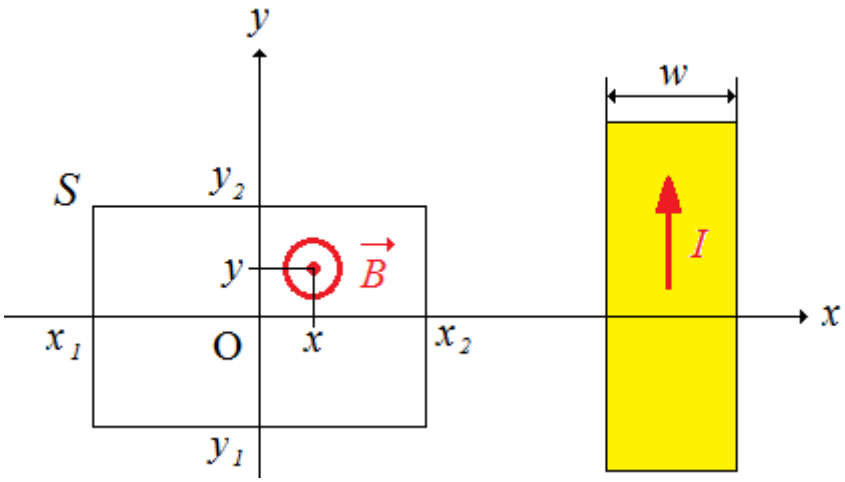
Avec la relation $\text{Argsh}(t) = \ln\left(t + \sqrt{t^2 + 1}\right)$, l'expression analytique du champ d'induction magnétique $B(x,y)$ s'écrit :

$$B(x, y) = \frac{\mu_0 I}{4\pi w}\ln\left(\frac{\left[\left(\frac{a + y}{c - x}\right) + \sqrt{\left(\frac{a + y}{c - x}\right)^2 + 1}\right]\left[\left(\frac{b - y}{c - x}\right) + \sqrt{\left(\frac{b - y}{c - x}\right)^2 + 1}\right]}{\left[\left(\frac{a + y}{c + w - x}\right) + \sqrt{\left(\frac{a + y}{c + w - x}\right)^2 + 1}\right]\left[\left(\frac{b - y}{c + w - x}\right) + \sqrt{\left(\frac{b - y}{c + w - x}\right)^2 + 1}\right]}\right)$$

Remarquons qu'au contact du conducteur ($x = c \Leftrightarrow h = 0$), la valeur de $B(x,y)$ tend théoriquement vers l'infini …

Pour un ruban de longueur infinie ($a = b = +\infty$), on obtient : $B = \frac{\mu_0 I}{2\pi w}\ln\left(1 + \frac{w}{c - x}\right)$

3) Flux de $\vec{B}$ à travers une surface rectangulaire S située dans le plan xOy



$$\phi = \iint_S \vec{B}.\overrightarrow{dS} = \iint_{[x_1,x_2]\times[y_1,y_2]} B(x,y)\,dx\,dy = \int_{x_1}^{x_2} dx \int_{y_1}^{y_2} B(x,y)\,dy$$

Intégration en y

En prenant l'expression de $B(x,y)$ avec les 4 fonctions argument sinus hyperbolique, $\int_{y_1}^{y_2} B(x,y)\,dy$ se décompose en 4 intégrales de la forme $\kappa \int_{y_1}^{y_2} \mathrm{Argsh}(f\,y+g)\,dy$ où f et g sont 2 fonctions de x. Le changement de variable $t = fy + g$ transforme ces intégrales en $h\int_{t_1}^{t_2} \mathrm{Argsh}(t)\,dt = h\left[t\,\mathrm{Argsh}(t) - \sqrt{t^2+1}\right]_{t_1}^{t_2} = F(x)$.

Intégration en x

La fonction $F(x)$ se décompose en une somme 2 de fonctions de la forme $\lambda\,\mathrm{Argsh}\left(\frac{\alpha}{x+\beta}\right)$ plus 2 fonctions de la forme $\mu\sqrt{\left(\frac{x+\beta}{\alpha}\right)^2+1}$ où $\alpha, \beta, \lambda, \mu$ sont des constantes. Le changement de variable $u = \frac{x+\beta}{\alpha}$ transforme l'intégrale $\int_{x_1}^{x_2} F(x)\,dx$ en 2 intégrales du type $\zeta\int_{u_1}^{u_2} \mathrm{Argsh}\left(\frac{1}{u}\right)du = \zeta\left[\mathrm{Argsh}(u) + u\,\mathrm{Argsh}\left(\frac{1}{u}\right)\right]_{u_1}^{u_2}$ plus 2 intégrales du type $\xi\int_{u_1}^{u_2}\sqrt{u^2+1}\,du = \xi\left[\frac{1}{2}\mathrm{Argsh}(u) + \frac{1}{2}u\sqrt{u^2+1}\right]_{u_1}^{u_2}$.

Finalement, on aboutit à 16 termes pour $\int_{x_1}^{x_2} F(x)\,dx$ et par conséquent à 64 termes pour $\iint_{[x_1,x_2]\times[y_1,y_2]} B(x,y)\,dx\,dy$.

Après simplifications, l'expression développée du flux d'induction magnétique $\phi(x_1 , y_1 , x_2 , y_2)$ à travers la surface S est:

$$\phi(x_1 , y_1 , x_2 , y_2) = \frac{\mu_0 I}{4\pi w}(P + Q + R + S)$$

$$\begin{aligned}
P = & -\frac{(a+y_1)^2}{2}\operatorname{Argsh}\left(\frac{c-x_1}{a+y_1}\right) - \frac{(a+y_2)^2}{2}\operatorname{Argsh}\left(\frac{c-x_2}{a+y_2}\right) \\
& + \frac{(a+y_2)^2}{2}\operatorname{Argsh}\left(\frac{c-x_1}{a+y_2}\right) + \frac{(a+y_1)^2}{2}\operatorname{Argsh}\left(\frac{c-x_2}{a+y_1}\right) \\
& -(c-x_1)(a+y_1)\operatorname{Argsh}\left(\frac{a+y_1}{c-x_1}\right) - (c-x_2)(a+y_2)\operatorname{Argsh}\left(\frac{a+y_2}{c-x_2}\right) \\
& + (c-x_1)(a+y_2)\operatorname{Argsh}\left(\frac{a+y_2}{c-x_1}\right) + (c-x_2)(a+y_1)\operatorname{Argsh}\left(\frac{a+y_1}{c-x_2}\right) \\
& -\frac{(c-x_1)}{2}\sqrt{(c-x_1)^2+(a+y_2)^2} - \frac{(c-x_2)}{2}\sqrt{(c-x_2)^2+(a+y_1)^2} \\
& + \frac{(c-x_1)}{2}\sqrt{(c-x_1)^2+(a+y_1)^2} + \frac{(c-x_2)}{2}\sqrt{(c-x_2)^2+(a+y_2)^2}
\end{aligned}$$

$$\begin{aligned}
Q = & +\frac{(b-y_1)^2}{2}\operatorname{Argsh}\left(\frac{c-x_1}{b-y_1}\right) + \frac{(b-y_2)^2}{2}\operatorname{Argsh}\left(\frac{c-x_2}{b-y_2}\right) \\
& - \frac{(b-y_2)^2}{2}\operatorname{Argsh}\left(\frac{c-x_1}{b-y_2}\right) - \frac{(b-y_1)^2}{2}\operatorname{Argsh}\left(\frac{c-x_2}{b-y_1}\right) \\
& +(c-x_1)(b-y_1)\operatorname{Argsh}\left(\frac{b-y_1}{c-x_1}\right) + (c-x_2)(b-y_2)\operatorname{Argsh}\left(\frac{b-y_2}{c-x_2}\right) \\
& - (c-x_1)(b-y_2)\operatorname{Argsh}\left(\frac{b-y_2}{c-x_1}\right) - (c-x_2)(b-y_1)\operatorname{Argsh}\left(\frac{b-y_1}{c-x_2}\right) \\
& + \frac{(c-x_1)}{2}\sqrt{(c-x_1)^2+(b-y_2)^2} + \frac{(c-x_2)}{2}\sqrt{(c-x_2)^2+(b-y_1)^2} \\
& - \frac{(c-x_1)}{2}\sqrt{(c-x_1)^2+(b-y_1)^2} - \frac{(c-x_2)}{2}\sqrt{(c-x_2)^2+(b-y_2)^2}
\end{aligned}$$

$$\begin{aligned}
R = & +\frac{(a+y_1)^2}{2}\operatorname{Argsh}\left(\frac{c+w-x_1}{a+y_1}\right) + \frac{(a+y_2)^2}{2}\operatorname{Argsh}\left(\frac{c+w-x_2}{a+y_2}\right) \\
& - \frac{(a+y_2)^2}{2}\operatorname{Argsh}\left(\frac{c+w-x_1}{a+y_2}\right) - \frac{(a+y_1)^2}{2}\operatorname{Argsh}\left(\frac{c+w-x_2}{a+y_1}\right) \\
& +(c+w-x_1)(a+y_1)\operatorname{Argsh}\left(\frac{a+y_1}{c+w-x_1}\right) + (c+w-x_2)(a+y_2)\operatorname{Argsh}\left(\frac{a+y_2}{c+w-x_2}\right) \\
& - (c+w-x_1)(a+y_2)\operatorname{Argsh}\left(\frac{a+y_2}{c+w-x_1}\right) - (c+w-x_2)(a+y_1)\operatorname{Argsh}\left(\frac{a+y_1}{c+w-x_2}\right) \\
& + \frac{(c+w-x_1)}{2}\sqrt{(c+w-x_1)^2+(a+y_2)^2} + \frac{(c+w-x_2)}{2}\sqrt{(c+w-x_2)^2+(a+y_1)^2} \\
& - \frac{(c+w-x_1)}{2}\sqrt{(c+w-x_1)^2+(a+y_1)^2} - \frac{(c+w-x_2)}{2}\sqrt{(c+w-x_2)^2+(a+y_2)^2}
\end{aligned}$$

$$S = -\frac{(b-y_1)^2}{2}\operatorname{Argsh}\left(\frac{c+w-x_1}{b-y_1}\right) - \frac{(b-y_2)^2}{2}\operatorname{Argsh}\left(\frac{c+w-x_2}{b-y_2}\right)$$
$$+ \frac{(b-y_2)^2}{2}\operatorname{Argsh}\left(\frac{c+w-x_1}{b-y_2}\right) + \frac{(b-y_1)^2}{2}\operatorname{Argsh}\left(\frac{c+w-x_2}{b-y_1}\right)$$
$$-(c+w-x_1)(b-y_1)\operatorname{Argsh}\left(\frac{b-y_1}{c+w-x_1}\right) - (c+w-x_2)(b-y_2)\operatorname{Argsh}\left(\frac{b-y_2}{c+w-x_2}\right)$$
$$+ (c+w-x_1)(b-y_2)\operatorname{Argsh}\left(\frac{b-y_2}{c+w-x_1}\right) + (c+w-x_2)(b-y_1)\operatorname{Argsh}\left(\frac{b-y_1}{c+w-x_2}\right)$$
$$-\frac{(c+w-x_1)}{2}\sqrt{(c-x_1)^2+(b-y_2)^2} - \frac{(c+w-x_2)}{2}\sqrt{(c+w-x_2)^2+(b-y_1)^2}$$
$$+ \frac{(c+w-x_1)}{2}\sqrt{(c+w-x_1)^2+(b-y_1)^2} + \frac{(c+w-x_2)}{2}\sqrt{(c+w-x_2)^2+(b-y_2)^2}$$

Remarque 1

Par hypothèse la surface S est un rectangle $[x_1, x_2] \times [y_1, y_2]$ vérifiant :

- $c \le x_1 < x_2 \le c$
- $a \le y_1 < y_2 \le b$

Pour les valeurs limites, lorsque $x_2 = c$ ou $y_1 = -a$ ou $y_2 = b$, les termes de $\phi(x_1, y_1, x_2, y_2)$ qui se présentent sous la forme indéterminée $0 \times \operatorname{Argsh}(\infty) = 0 \times \infty$ seront pris égaux à 0 puisque :

$$\lim_{u\to 0^+} \alpha u \operatorname{Argsh}\left(\frac{\beta}{u}\right) = \lim_{v\to 0^+} \gamma v \operatorname{Argsh}\left(\frac{1}{v}\right) = \lim_{v\to 0^+} \gamma v \ln\left(\frac{1+\sqrt{v^2+1}}{v}\right) = \lim_{v\to 0^+} \gamma v \ln\left(\frac{2}{v}\right) = \lim_{w\to +\infty} \delta \frac{\ln(w)}{w} = \delta \times 0 = 0$$

Remarque 2

Dans le cas d'un ruban de longueur infinie ($a = b = +\infty$), on a :

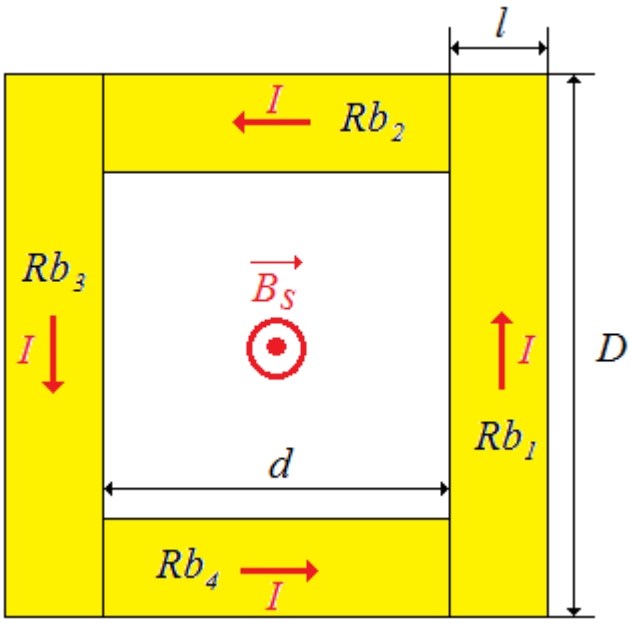
$$\phi(x_1, y_1, x_2, y_2) = \int_{x_1}^{x_2} dx \int_{y_1}^{y_2} \frac{\mu_0 I}{2\pi w} \ln\left(1 + \frac{w}{c-x}\right) dy = \frac{\mu_0 I}{2\pi w}(y_2 - y_1) \int_{x_1}^{x_2} \ln\left(1 + \frac{w}{c-x}\right) dx$$
$$= \frac{\mu_0 I}{2\pi}(y_2 - y_1) \int_{\frac{w}{c-x_1}}^{\frac{w}{c-x_2}} \frac{\ln(1+u)}{u^2} du = \frac{\mu_0 I}{2\pi}(y_2 - y_1)\left[\ln(u) - \left(1 + \frac{1}{u}\right)\ln(1+u)\right]_{\frac{w}{c-x_1}}^{\frac{w}{c-x_2}}$$
$$= \frac{\mu_0 I}{2\pi}(y_2 - y_1)\left[\ln\left(\frac{c-x_1}{c-x_2}\right) + \left(1 + \frac{c-x_1}{w}\right)\ln\left(1 + \frac{w}{c-x_1}\right) - \left(1 + \frac{c-x_2}{w}\right)\ln\left(1 + \frac{w}{c-x_2}\right)\right]$$

Quand $x_2 \to c$, $\phi \to \frac{\mu_0 I}{2\pi}(y_2 - y_1)\left[\ln\left(\frac{c-x_1}{w}\right) + \left(1 + \frac{c-x_1}{w}\right)\ln\left(1 + \frac{w}{c-x_1}\right)\right]$

Quand $w \to 0$, $\phi \to \frac{\mu_0 I}{2\pi}(y_2 - y_1)\ln\left(\frac{c-x_1}{c-x_2}\right)$

4) Champ et flux d'induction magnétique à l'intérieur d'une spire carrée extraplate parcourue par un courant I

On suppose que la spire a la forme ci-dessous (réunion de 4 rubans rectangulaires Rb_1 , Rb_2 , Rb_3 , Rb_4) :



Dimensions associées aux différents rubans :

	a	b	c	w
Rb_1	$D/2$	$D/2$	$d/2$	l
Rb_2	$d/2$	$d/2$	$d/2$	l
Rb_3	$D/2$	$D/2$	$d/2$	l
Rb_4	$d/2$	$d/2$	$d/2$	l

Chaque ruban Rb_k créé au point de coordonnées (x,y) un champ d'induction magnétique noté $\overrightarrow{B_k}(x,y)$.

Le champ $\overrightarrow{B_1}(x,y)$ correspond au champ $\vec{B}(x,y)$ étudié précédemment.

Les champs $\overrightarrow{B_2}(x,y)$, $\overrightarrow{B_3}(x,y)$, $\overrightarrow{B_4}(x,y)$ s'obtiennent à partir de $\vec{B}(x,y)$ par rotations successives d'un angle égal à $\frac{\pi}{2}$ dans le plan xOy. Ainsi, en module :

$$B_1(x,y) = B(x,y)$$
$$B_2(x,y) = B(y,-x)$$
$$B_3(x,y) = B(-x,-y)$$
$$B_4(x,y) = B(-y,x)$$

Le champ d'induction magnétique $\vec{B}_s$ créé par la spire parcourue par un courant I est : $\vec{B}_s = \vec{B}_1 + \vec{B}_2 + \vec{B}_3 + \vec{B}_4$

En module, comme les champs $\overrightarrow{B_k}$ sont colinéaires et de même sens : $B_s = B_1 + B_2 + B_3 + B_4$

Le flux d'induction magnétique ϕ_s à travers la surface intérieure de la spire est : $\phi_s = \iint_S \overrightarrow{B_s} \,.\, \overrightarrow{dS} = \iint_{[-d/2\,,\,d/2]^2} B_s(x,y)\,dx\,dy$

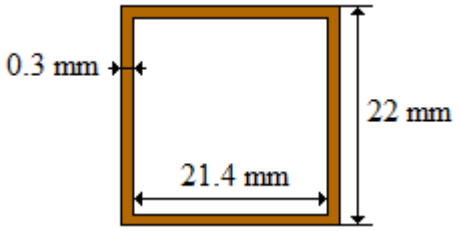
L'inductance de la spire est : $L_s = \frac{\phi_s}{I}$

5) Calcul numérique de ϕ_s

Le flux ϕ_s est défini par une intégrale double impropre (valeur infinie de $B_s(x,y)$ sur le bord de la surface S) délicate à évaluer numériquement avec précision. On peut néanmoins estimer ϕ_s en intégrant sur une surface S' dont le contour s'approche à ε près de la surface théorique S. On utilise la fonction Scilab *int2d* ou on applique une formule de cubature de type **Simpson**(*) adaptée à l'intégration sur une surface rectangulaire.
Noter qu'il est possible d'établir l'expression exacte de ϕ_s en fonction de D et d en prenant les valeurs limites des primitives de $B_s(x,y)$ aux bornes des intervalles d'intégration. Le calcul de ϕ_s (et donc de L_s) s'effectue alors avec une grande précision.

6) Exemple

Soit une spire carrée imprimée de dimensions $D = 22$ mm, $d = 21.4$ mm, $l = 0.3$ mm parcourue par un courant $I = 100$ mA.



- Pour représenter le module du champ d'induction magnétique $\overrightarrow{B_s}(x,y)$ apparaissant à l'intérieur de la spire, on écrit :

```
function zs=Bs(x, y)
    function z=B(x, y)
        p = (a + y)/(c - x) + sqrt(((a + y)/(c - x))^2 + 1)
        q = (b - y)/(c - x) + sqrt(((b - y)/(c - x))^2 + 1)
        r = (a + y)/(c + w - x) + sqrt(((a + y)/(c + w - x))^2 + 1)
        s = (b - y)/(c + w - x) + sqrt(((b - y)/(c + w - x))^2 + 1)
        z = (K*I/w)*log((p*q)/(r*s))
    endfunction
    a = D/2 ; b = D/2 ; c = d/2 ; w = l
    z1 = B(x,y)
    a = d/2 ; b = d/2 ; c = d/2 ; w = l
    z2 = B(y,-x)
    a = D/2 ; b = D/2 ; c = d/2 ; w = l
    z3 = B(-x,-y)
    a = d/2 ; b = d/2 ; c = d/2 ; w = l
    z4 = B(-y,x)
    zs = z1 + z2 + z3 + z4
endfunction

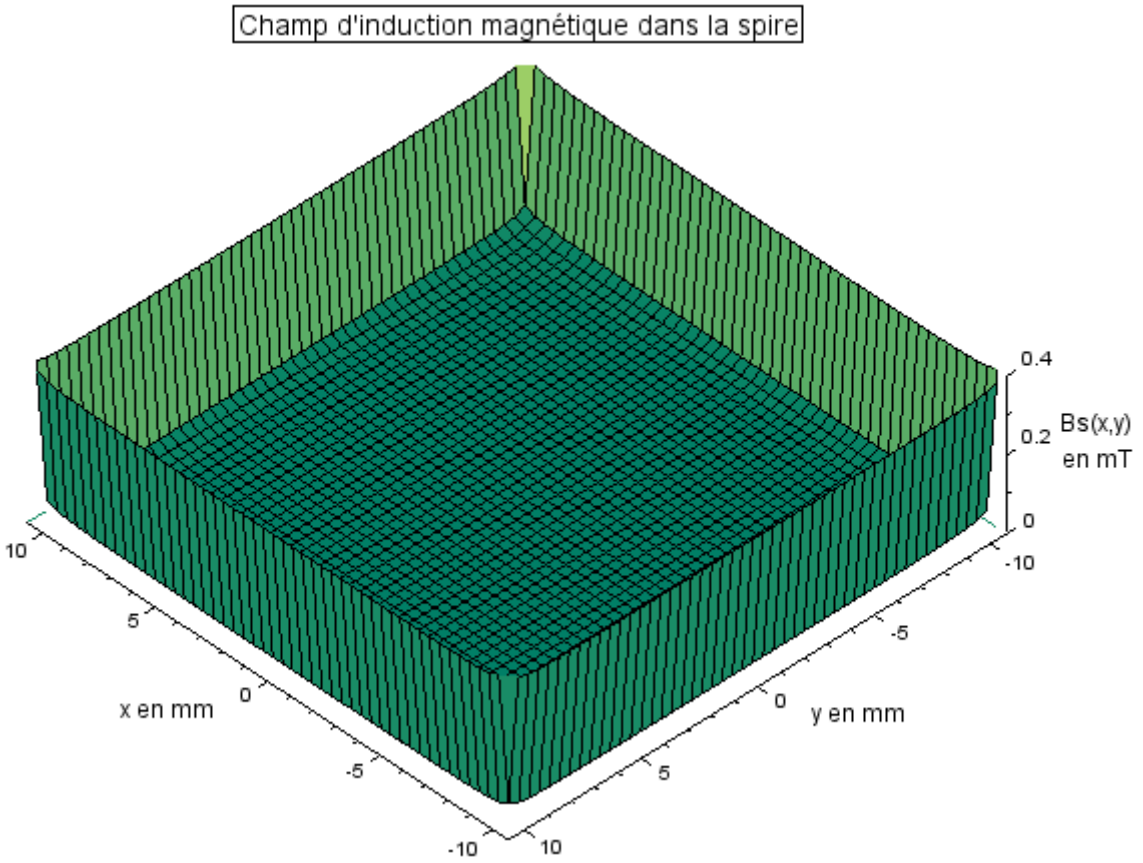
// Données
D = 22 ;
d = 21.4 ;
l = 0.3 ;
I = 0.1 ;

// Constantes
K = 0.1 ;
eps = 1e-3 ;

// Surface
x1 = - d/2 + eps ;
x2 = d/2 - eps ;
y1 = - d/2 + eps ;
y2 = d/2 - eps ;
```

```
// Champ magnétique
x = linspace(x1,x2,50) ;
y = linspace(y1,y2,50) ;
fplot3d1(x,y,Bs,alpha = 87,theta = 136,flag = [1,2,2])
f = gcf() ;
f.color_map = summercolormap(32) ;
f.figure_position = [0,0] ;
f.figure_size = [1000,700] ;
a = gca() ;
a.axes_visible = "on" ;
a.tight_limits = "on" ; a.data_bounds = [x1,y1,0;x2,y2,0.4] ;
a.x_label.visible = "on" ; a.x_label.text = "x en mm" ; a.x_label.font_size = 2 ;
a.y_label.visible = "on" ; a.y_label.text = "y en mm" ; a.y_label.font_size = 2 ;
a.z_label.visible = "on" ; a.z_label.text = "Bs(x,y) en mT" ; a.z_label.font_size = 2 ;
a.title.text = "Champ d"induction magnétique dans la spire" ;
a.title.font_size = 3 ; a.title.fill_mode = "on" ;
// Valeur de Bs(0,0)
Bs0 = Bs(0,0) ;
printf('\n Bs0 = %.6f mT \n',Bs0)
```

NB Avec K = 0.1, toutes les dimensions sont en millimètres (mm), I est en ampères (A), B_s est en milliteslas (mT).



Remarque

Le champ magnétique est minimal au centre de la spire, il vaut : $B_{s_0} = B_s(0,0) = 0.005214$ mT = 5.214 µT

On retrouve ce résultat en considérant que la spire est un conducteur carré filiforme (sans largeur) de côté $c = (D+d)/2$ parcouru par un courant I. Dans ce cas, on montre que la valeur du champ magnétique au centre du carré est donnée par :

$$B_0 = \frac{2\sqrt{2}\mu_0 I}{\pi c} = \frac{2\sqrt{2}\times 4\pi\times 10^{-7}\times 0.1}{\pi\times 21.7\times 10^{-3}} = 5.214\times 10^{-6}\ \text{T} = 5.214\,\mu\text{T}$$

- Pour calculer le flux d'induction ϕ_s et l'inductance L_s de la spire, on écrit :

```
// Flux magnétique
[Phis,e] = int2d([x1,x2;x2,x2;x1,x1],[y1,y1;y1,y2;y2,y2],Bs,[1e-1,1,100,9000,1]) ;
printf('\n Phis = %.3f nWb \n',Phis)

// Inductance
Ls = Phis/I ;
printf('\n Ls = %.3f nH \n',Ls)
```

NB ϕ_s est en nanowebers (nWb), L_s est en nanohenrys (nH).

Résultat :

$\phi_s = 7.848$ nWb

$L_s = 78.48$ nH

Comparons la valeur L_s avec celle donnée par la formule de **Wheeler** pour une inductance imprimée de n spires carrées :

$$L \approx \frac{9\pi n^2 (D+d)^2}{4(15D-7d)}$$

où D et d sont en millimètres (mm), L est en nanohenrys (nH).

Ici :

$n = 1$
$D = 22$
$d = 21.4$

D'où :

$$L \approx \frac{9\pi \times 1^2 \times (22+21.4)^2}{4(15\times 22 - 7\times 21.4)} = 73.89 \text{ nH}$$

Mathématique & Informatique

(*) Intégrale double : méthode d'approximation de type **Simpson**

$$I = \iint_{[a,A]\times[b,B]} f(x,y)\,dx\,dy = ?$$

1) Choisir les pas de discrétisation

$$h = \frac{A-a}{2n}$$
$$k = \frac{B-b}{2m}$$

2) Calculer les coordonnées des points du maillage

$$x_i = a + (i-1)h \qquad i = 1, 2, \ldots, 2n+1$$
$$y_j = b + (j-1)k \qquad j = 1, 2, \ldots, 2m+1$$

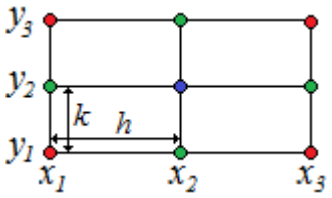
3) Evaluer l'intégrale avec les cotes $f_{ij} = f(x_i, y_j)$

$$I \approx \frac{hk}{9} \sum_{i=1}^{n} \sum_{j=1}^{m} \begin{bmatrix} f_{2i-1,\,2j-1} + f_{2i-1,\,2j+1} + f_{2i+1,\,2j-1} + f_{2i+1,\,2j+1} \\ +4\left(f_{2i-1,\,2j} + f_{2i,\,2j-1} + f_{2i,\,2j+1} + f_{2i+1,\,2j}\right) \\ +16 f_{2i,\,2j} \end{bmatrix}$$

Plus généralement, pour évaluer $I = \iint_D f(x,y)\,dx\,dy$ où le domaine D est curviligne, on construit un rectangle $R = [a,A] \times [b,B]$ qui contient D ($D \subset R$), de sorte que : $I = \iint_{[a,A]\times[b,B]} g(x,y)\,dx\,dy$ avec $g(x,y) = \begin{cases} f(x,y) \text{ si } (x,y) \in D \\ 0 \text{ sinon} \end{cases}$

Le plus souvent, on prend : $a = \text{Min}(x \in D)$, $A = \text{Max}(x \in D)$, $b = \text{Min}(y \in D)$, $B = \text{Max}(y \in D)$.

Preuve



Notons R_1 le premier petit rectangle élémentaire à 9 points :

$$I_1 = \iint_{R_1} f(x,y)\,dx\,dy = \int_{x_1}^{x_3} \left(\int_{y_1}^{y_3} f(x,y)\,dy \right) dx$$

Approximons l'intégrale intérieure avec la formule de **Simpson** : $\int_{y_1}^{y_3} f(x,y)\,dy \approx \frac{k}{3}\left[f(x,y_1) + 4f(x,y_2) + f(x,y_3)\right]$

D'où : $I_1 \approx \frac{k}{3}\left[\int_{x_1}^{x_3} f(x,y_1)\,dx + 4\int_{x_1}^{x_3} f(x,y_2)\,dx + \int_{x_1}^{x_3} f(x,y_3)\,dx\right]$

Appliquons une nouvelle fois la formule de **Simpson** :

$$I_1 \approx \frac{hk}{9}\{[f(x_1,y_1) + 4f(x_2,y_1) + f(x_3,y_1)] + 4[f(x_1,y_2) + 4f(x_2,y_2) + f(x_3,y_2)] + [f(x_1,y_3) + 4f(x_2,y_3) + f(x_3,y_3)]\}$$
$$= \frac{hk}{9}\{[f(x_1,y_1) + f(x_1,y_3) + f(x_3,y_1) + f(x_3,y_3)] + 4[f(x_1,y_2) + f(x_2,y_1) + f(x_2,y_3) + f(x_3,y_2)] + 16 f(x_2,y_2)\}$$

On étend cette formule de cubature aux $N = n \times m$ rectangles élémentaires constituant $R = [a,A] \times [b,B]$: $I = I_1 + I_2 + \cdots + I_N$.

Mathématique & Informatique

Exemple 1

$$I = \iint_{[0,\sqrt{3}]\times[0,1]} \frac{dx\,dy}{\left(x^2+y^2+2\right)^{3/2}}$$

```
function z = f(x,y)
    z = 1/(x^2 + y^2 + 2)^(3/2)
endfunction

// Rectangle
a = 0 ;
A = sqrt(3) ;
b = 0 ;
B = 1 ;
// Maillage
n = 20 ;
m = 20 ;
h = (A - a)/(2*n) ;
k = (B - b)/(2*m) ;
x = linspace(a,A,2*n + 1) ;
y = linspace(b,B,2*m + 1) ;
// Integration
I = 0 ;
for i = 1:n
    for j = 1:m
        I = I + f(x(2*i-1),y(2*j-1)) + f(x(2*i-1),y(2*j+1)) + f(x(2*i+1),y(2*j-1)) + f(x(2*i+1),y(2*j+1)) ...
            + 4*(f(x(2*i-1),y(2*j)) + f(x(2*i),y(2*j-1)) + f(x(2*i),y(2*j+1)) + f(x(2*i+1),y(2*j))) ...
            + 16*f(x(2*i),y(2*j)) ;
    end
end
I = ((h*k)/9)*I ;
// Résultat
disp(I,'Valeur approchée :')
```

On trouve :

$I \approx 0.3278484$

La valeur exacte est $I = \frac{\pi}{4\sqrt{2}} - \frac{1}{\sqrt{2}}\operatorname{Arcsin}\left(\frac{1}{2}\sqrt{\frac{2}{5}}\right) = 0.327848368$

Exemple 2

$$I = \iint_D (x^4 + y^4)\, dx\, dy \quad \text{où} \quad D = \left\{ (x,y) \,\middle|\, \frac{x^2}{4} + y^2 \le 1 \right\}$$

Le domaine d'intégration *D* est l'intérieur d'une ellipse de demi grand axe égal à 2 et de demi petit axe égal à 1.
On a : $D \subset [-2,2] \times [-1,1]$, d'où :

$$I = \iint_{[-2,2]\times[-1,1]} g(x,y)\, dx\, dy \quad \text{avec} \quad g(x,y) = \begin{cases} x^4 + y^4 & \text{si } \frac{x^2}{4} + y^2 \le 1 \\ 0 & \text{sinon} \end{cases}$$

```
function z = g(x,y)
    if x^2/4 + y^2 <= 1 then
        z = x^4 + y^4
    else
        z = 0
    end
endfunction

// Rectangle
a = -2 ;
A = 2 ;
b = -1 ;
B = 1 ;
// Maillage
n = 800 ;
m = 400 ;
h = (A - a)/(2*n) ;
k = (B - b)/(2*m) ;
x = linspace(a,A,2*n + 1) ;
y = linspace(b,B,2*m + 1) ;
// Integration
I = 0 ;
for i = 1:n
    for j = 1:m
        I = I + g(x(2*i-1),y(2*j-1)) + g(x(2*i-1),y(2*j+1)) + g(x(2*i+1),y(2*j-1)) + g(x(2*i+1),y(2*j+1)) ...
              + 4*(g(x(2*i-1),y(2*j)) + g(x(2*i),y(2*j-1)) + g(x(2*i),y(2*j+1)) + g(x(2*i+1),y(2*j))) ...
              + 16*g(x(2*i),y(2*j)) ;
    end
end
I = ((h*k)/9)*I ;
// Résultat
disp(I,'Valeur approchée :')
```

On trouve :

$I \approx 13.353177$

La valeur exacte est $I = \frac{17\pi}{4} = 13.35176878$

Exemple 3

$$I = \frac{1}{2}\iint_{[0,1]\times[0,\pi/2]} \frac{dk\,d\theta}{\sqrt{1-k^2\sin^2\theta}}$$

L'intégrale est impropre pour la borne $k = 1$ et $\theta = \pi/2$. En ce point où la fonction elliptique f est infinie, on prend pour valeur de $f(1,\pi/2)$ la cote attribuée à un point du maillage proche de $(1,\pi/2)$, par exemple $f(1 - h,\pi/2 - k)$.

```
function z = f(x,y)
    if x == 1 & y == %pi/2 then
        x = x - h
        y = y - k
    end
    z = (1/2)*(1/sqrt(1 - x^2*sin(y)^2))
endfunction

// Rectangle
a = 0 ;
A = 1 ;
b = 0 ;
B = %pi/2 ;
// Maillage
n = 200 ;
m = 200 ;
h = (A - a)/(2*n) ;
k = (B - b)/(2*m) ;
x = linspace(a,A,2*n + 1) ;
y = linspace(b,B,2*m + 1) ;
// Integration
I = 0 ;
for i = 1:n
    for j = 1:m
        I = I + f(x(2*i-1),y(2*j-1)) + f(x(2*i-1),y(2*j+1)) + f(x(2*i+1),y(2*j-1)) + f(x(2*i+1),y(2*j+1)) ...
              + 4*(f(x(2*i-1),y(2*j)) + f(x(2*i),y(2*j-1)) + f(x(2*i),y(2*j+1)) + f(x(2*i+1),y(2*j))) ...
              + 16*f(x(2*i),y(2*j)) ;
    end
end
I = ((h*k)/9)*I ;
// Résultat
disp(I,'Valeur approchée :')
```

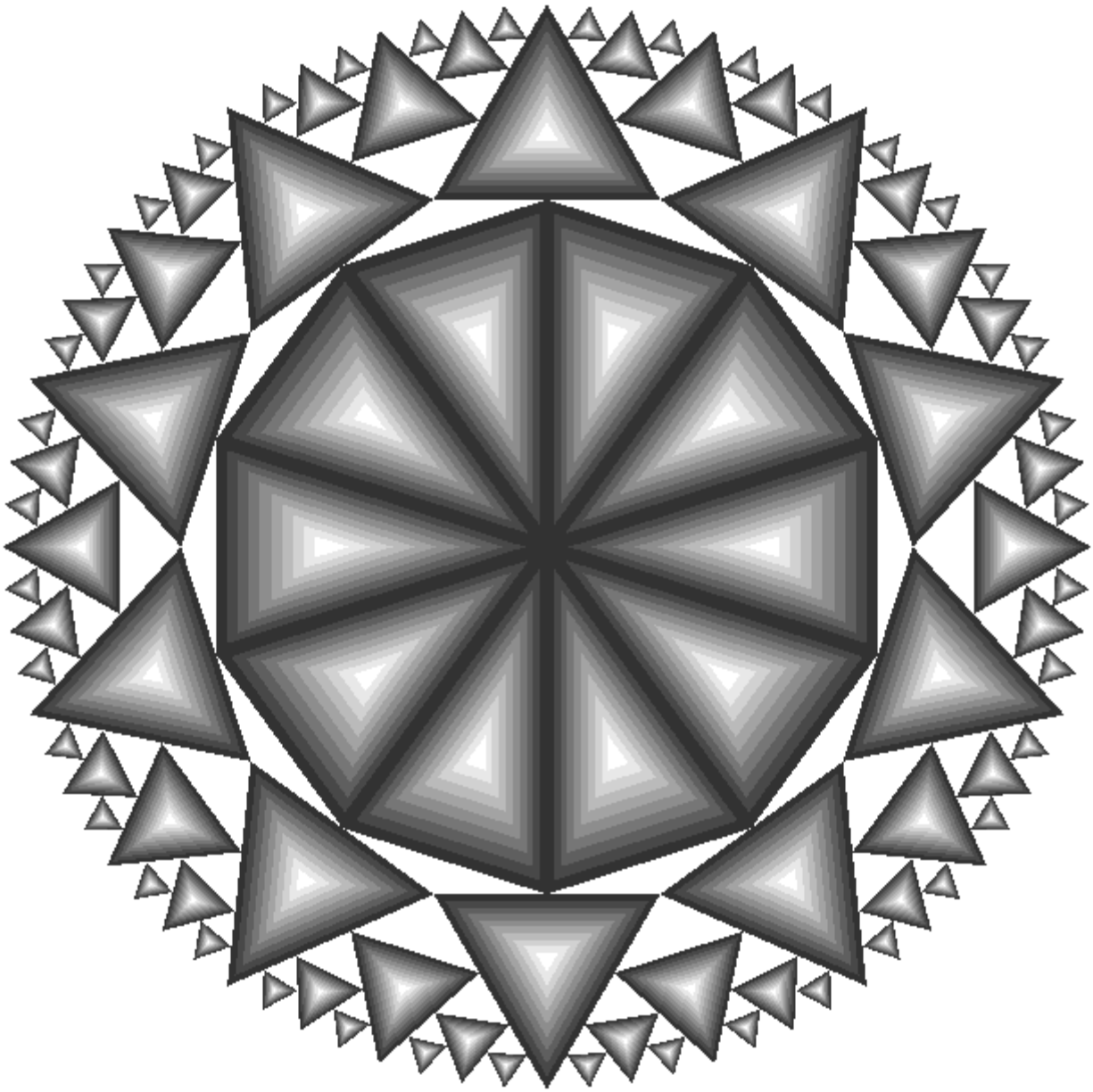
On trouve :

$$I \approx 0.9166984$$

La valeur exacte est $I = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)^2} = \frac{1}{1^2} - \frac{1}{3^2} + \frac{1}{5^2} - \cdots = 0.915965594\cdots$ (constante de **Catalan**)

Avec la formule $I = \iint_{[0,1]^2} \frac{dx\,dy}{1+x^2y^2}$ et la même discrétisation, on obtient une meilleure approximation : $I \approx 0.9159656$

Mathématique & Informatique



Mathématique & Informatique

Le problème de la chèvre (**Henri Poincaré**)

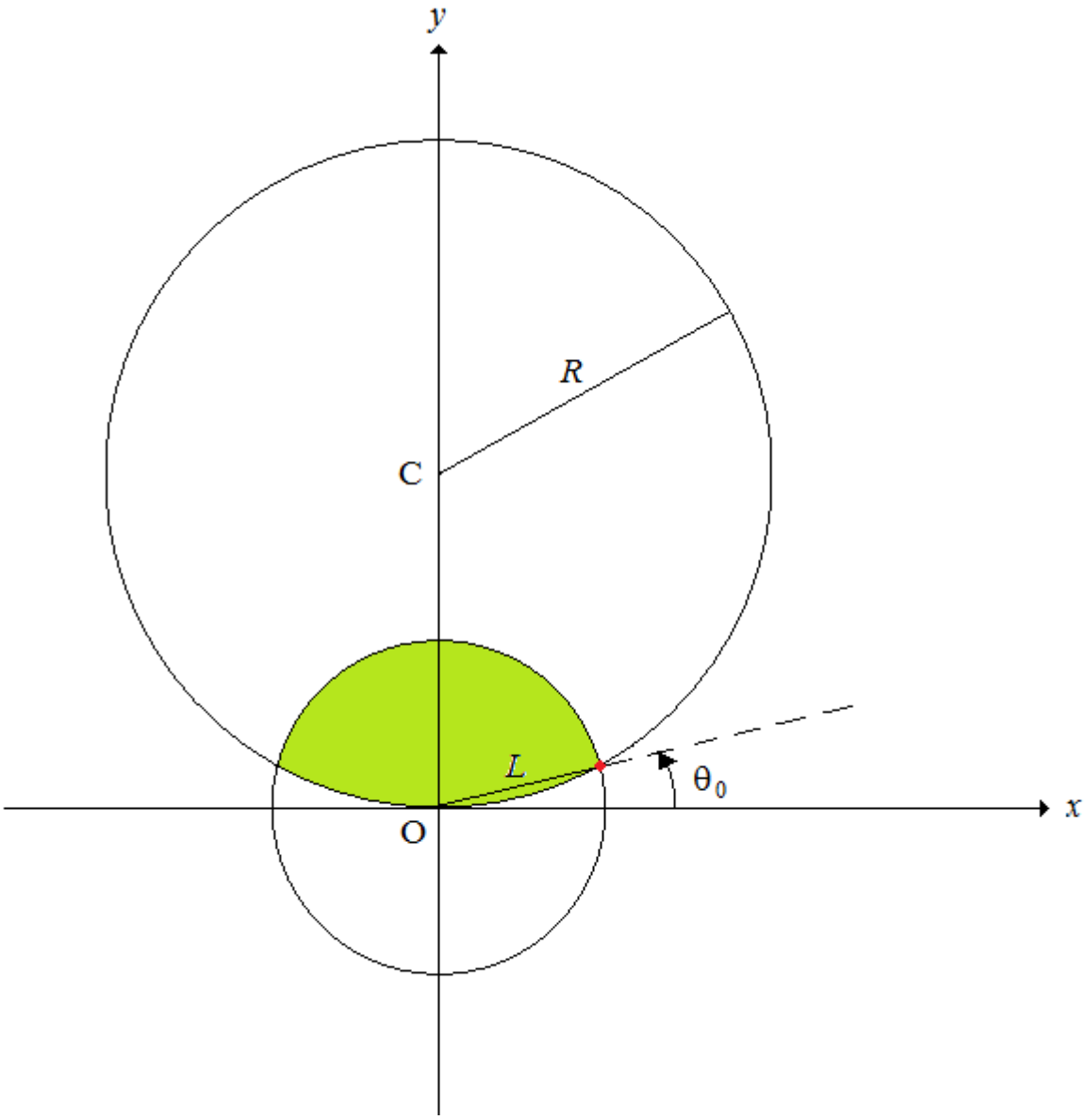
Une chèvre est attachée avec une corde reliée à un poteau situé en un point de la circonférence d'une prairie circulaire de rayon *R*. Pour *R* = 100 m, on demande de calculer :

1) La surface *S* que la chèvre peut brouter si la corde a une longueur égale à la moitié du rayon de la prairie

2) La longueur *L* à donner à la corde pour que la chèvre puisse brouter la moitié de la surface de la prairie

Solution

- Premièrement, on effectue une représentation géométrique du problème



O = origine du repère cartésien (position du poteau)
R = rayon du cercle centré en C (rayon de la prairie)
L = rayon du cercle centré en O (longueur de la corde)
S = intersection des 2 disques (surface de pâture de la chèvre)

- Deuxièmement, on exprime S en fonction de R et L

En coordonnées polaires (r,θ), le cercle de rayon L centré en O a pour équation : $r = L$

En coordonnées polaires (r,θ), le cercle de rayon R centré en C a pour équation : $r = 2R\cos(\frac{\pi}{2} - \theta) = 2R\sin\theta$

Soit θ_0 l'angle polaire du point d'intersection des 2 cercles : $L = 2R\sin\theta_0 \Rightarrow \theta_0 = \text{Arc sin}\left(\frac{L}{2R}\right)$

Puisque Oy est un axe de symétrie, on réduit le domaine de variation de θ à $\left[0,\frac{\pi}{2}\right] = [0,\theta_0] \cup \left[\theta_0,\frac{\pi}{2}\right]$, et on a :

$$\frac{S}{2} = \iint dx\,dy = \iint r\,dr\,d\theta = \int_0^{\theta_0} d\theta \int_0^{2R\sin\theta} r\,dr + \int_{\theta_0}^{\frac{\pi}{2}} d\theta \int_0^{L} r\,dr = 2R^2 \int_0^{\theta_0} \sin^2\theta\,d\theta + \frac{1}{2}L^2 \int_{\theta_0}^{\frac{\pi}{2}} d\theta$$

La primitive de $\sin^2\theta$ est $\frac{1}{2}(\theta - \sin\theta\cos\theta)$, d'où : $\frac{S}{2} = R^2(\theta_0 - \sin\theta_0\cos\theta_0) + \frac{1}{2}L^2\left(\frac{\pi}{2} - \theta_0\right)$

Comme $\cos\theta_0 = \sqrt{1 - \sin^2\theta_0}$ dans le premier quadrant, on déduit :

$$S = \frac{1}{2}\pi L^2 - \frac{L}{2}\sqrt{4R^2 - L^2} + (2R^2 - L^2)\,\text{Arc sin}\left(\frac{L}{2R}\right)$$

La relation $\text{Arc sin}\,x + \text{Arc cos}\,x = \frac{\pi}{2}$ permet de transformer l'expression précédente :

$$S = \pi R^2 - \frac{L}{2}\sqrt{4R^2 - L^2} - (2R^2 - L^2)\,\text{Arc cos}\left(\frac{L}{2R}\right)$$

- Troisièmement, on fait appel au logiciel Scilab pour obtenir les valeurs numériques recherchées

<u>Question 1</u>

Il s'agit de calculer S connaissant R et L.

Dans le cas particulier où $L = \frac{R}{2}$, on obtient : $S = \left(\pi - \frac{\sqrt{15}}{8} - \frac{7}{4}\ \text{Arc cos}\,\frac{1}{4}\right)R^2$

La solution est analytique, on utilise les fonctions prédéfinies *sqrt* et *acos*

```
R = 100 ;
S = (%pi - sqrt(15)/8 - (7/4)*acos(1/4))*R^2
```

On trouve : $S = 3507.6661\ \text{m}^2$

<u>Question 2</u>

Il s'agit de calculer L connaissant R et S.

Dans le cas particulier où $S = \frac{\pi R^2}{2}$, on obtient : $\pi R^2 - L\sqrt{4R^2 - L^2} - 2(2R^2 - L^2)\,\text{Arc cos}\left(\frac{L}{2R}\right) = 0$

La solution est algorithmique, on utilise la fonction prédéfinie *fsolve*

```
R = 100 ;
deff('y = f(L)','y = %pi*R^2 - L*sqrt(4*R^2 - L^2) - 2*(2*R^2 - L^2)*acos(L/(2*R))') ;
L = fsolve(R,f)
```

On trouve : $L = 115.87285\ \text{m}$

Extension du problème à l'espace

Remplaçons les cercles par des sphères : une première sphère de rayon R centrée en C et une deuxième sphère de rayon L centrée en O. L'intersection des 2 boules correspondantes est le volume V engendré par la rotation de l'aire $A = \frac{S}{2}$ autour de l'axe Oy. Le théorème de **Guldin** s'applique : $V = 2\pi x_G A$ où x_G est la distance du centre de gravité de l'aire A à l'axe Oy.
Comme $x_G = \frac{1}{A}\iint x\,dx\,dy$, on a :

$$V = 2\pi\iint x\,dx\,dy = 2\pi\iint r^2\cos\theta\,dr\,d\theta = 2\pi\left\{\int_0^{\theta_0} d\theta\int_0^{2R\sin\theta} r^2\cos\theta\,dr + \int_{\theta_0}^{\frac{\pi}{2}} d\theta\int_0^{L} r^2\cos\theta\,dr\right\}$$

$$= 2\pi\left\{\frac{8R^3}{3}\int_0^{\theta_0}\sin^3\theta\cos\theta\,d\theta + \frac{L^3}{3}\int_{\theta_0}^{\frac{\pi}{2}}\cos\theta\,d\theta\right\} = 2\pi\left\{\frac{8R^3}{3}\left[\frac{\sin^4\theta}{4}\right]_0^{\theta_0} + \frac{L^3}{3}\left[\sin\theta\right]_{\theta_0}^{\frac{\pi}{2}}\right\}$$

$$= \frac{4\pi R^3}{3}\left(\frac{L}{2R}\right)^4 + \frac{2\pi L^3}{3}\left(1 - \frac{L}{2R}\right)$$

Après simplification :

$$V = \left(\frac{2}{3} - \frac{L}{4R}\right)\pi L^3$$

Premier cas, si L est égal à la moitié du rayon R de la sphère centrée en C, on trouve :

$$V = \frac{13}{192}\,\pi R^3$$

Par exemple, pour R = 100 m : V = 212712.0026 m^3
Deuxième cas, si l'on veut que V soit égal à la moitié du volume de la sphère de rayon R centrée en C, il faut que :

$$\left(\frac{2}{3} - \frac{L}{4R}\right)\pi L^3 = \frac{1}{2}\left(\frac{4}{3}\pi R^3\right)$$

c'est-à-dire :

$$3L^4 - 8RL^3 + 8R^4 = 0$$

Le rayon R étant fixé, cette condition est une équation polynomiale de degré 4 en L. Il est possible d'exprimer L en fonction de R avec la méthode de **Ferrari** (*). Toutefois, le plus simple est d'utiliser la fonction *roots* de Scilab.
Pour R = 100 m, le programme s'écrit :

```
R = 100 ;
L = poly(0,'L') ;
P = 3*L^4 - 8*R*L^3 + 8*R^4 ;
roots(P)
```

Les 4 racines dans $\mathbb{C}$ du polynôme $P(L) = 3L^4 - 8RL^3 + 8R^4$ sont approximativement :

```
 249.49646
 122.85449
- 52.842139 + 76.860846i
- 52.842139 - 76.860846i
```

Seule la valeur réelle vérifiant $0 \le L \le 2R$ est à retenir, donc L = 122.85449 m
Remarquons que ce résultat est différent de celui obtenu dans le plan en résolvant l'équation :

$$\pi R^2 - L\sqrt{4R^2 - L^2} - 2(2R^2 - L^2)\,\mathrm{Arc}\cos\left(\frac{L}{2R}\right) = 0$$

(*) Résolution de l'équation du quatrième degré $3L^4 - 8RL^3 + 8R^4 = 0$ avec la méthode de **Ferrari**

Posons $L = kR$, reportons dans $3L^4 - 8RL^3 + 8R^4 = 0$ et divisons par R^4. Il faut trouver k tel que : $3k^4 - 8k^3 + 8 = 0$
On met cette équation sous la forme $(k^2 + \alpha k + \beta)(k^2 + \gamma k + \delta) = 0$ pour se ramener à 2 équations du 2e degré.

Faisons apparaître un premier carré : $3k^4 - 8k^3 + 8 = 0 \Rightarrow k^4 - \frac{8}{3}k^3 + \frac{8}{3} = 0 \Rightarrow \left(k^2 - \frac{4}{3}k\right)^2 - \frac{16}{9}k^2 + \frac{8}{3} = 0$

Introduisons la variable auxiliaire x : $\left(k^2 - \frac{4}{3}k + x\right)^2 - \left[\left(2x + \frac{16}{9}\right)k^2 - \frac{8}{3}xk + \left(x^2 - \frac{8}{3}\right)\right] = 0$

Choisissons x de sorte que l'expression entre crochets soit un deuxième carré du type $(uk + v)^2$, ce qui impose que l'on ait la relation $\left(2x + \frac{16}{9}\right)\left(x^2 - \frac{8}{3}\right) = \left(-\frac{4}{3}x\right)^2$ équivalente à $x^3 - \frac{8}{3}x - \frac{64}{27} = 0$

Cette équation du troisième degré en x a pour discriminant $\Delta = 4\left(-\frac{8}{3}\right)^3 + 27\left(-\frac{64}{27}\right)^2 = \frac{2048}{27} > 0$
Par suite, il existe une unique racine réelle x donnée par la formule de **Cardan**(**) :

$$x = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} \quad \text{avec } p = -\frac{8}{3},\ q = -\frac{64}{27}$$

Tous les calculs faits : $x = \frac{2}{3}\left(\sqrt[3]{4 + 2\sqrt{2}} + \sqrt[3]{4 - 2\sqrt{2}}\right) \approx 1.967582$

Il reste à rechercher k tel que : $\left(k^2 - \frac{4}{3}k + x\right)^2 - (uk + v)^2 = 0$ avec $u^2 = 2x + \frac{16}{9}$, $v^2 = x^2 - \frac{8}{3}$, $uv < 0$

Factorisons la différence des 2 carrés : $\left(k^2 - \left(\frac{4}{3} + u\right)k + (x - v)\right)\left(k^2 - \left(\frac{4}{3} - u\right)k + (x + v)\right) = 0$

On obtient 2 équations du 2e degré : $k^2 - \left(\frac{4}{3} + u\right)k + (x - v) = 0$ et $k^2 - \left(\frac{4}{3} - u\right)k + (x + v) = 0$

Soit : $k^2 - ak + b = 0$ avec $a = \frac{4}{3} \pm u$, $b = x \mp v$ et $u = \pm\sqrt{2x + \frac{16}{9}}$, $v = \mp\sqrt{x^2 - \frac{8}{3}}$

En convenant que $\sqrt{-1} = i$, $\lambda = \pm 1$ et $\mu = \pm 1$, on aboutit à la solution algébrique :

$$k = \frac{a + \mu\sqrt{a^2 - 4b}}{2} \quad \text{avec } a = \frac{4 + \lambda\sqrt{18x + 16}}{3},\ b = \frac{3x + \lambda\sqrt{9x^2 - 24}}{3}$$

D'où le programme Scilab :

```
x = (2/3)*((4 + 2*sqrt(2))^(1/3) + (4 - 2*sqrt(2))^(1/3)) ;
for lambda = [+1,-1]
    a = (4 + lambda*sqrt(18*x + 16))/3 ;
    b = (3*x + lambda*sqrt(9*x^2 - 24))/3 ;
    for mu = [+1,-1]
        k = (a + mu*sqrt(a^2 - 4*b))/2 ;
        disp(lambda), disp(mu), disp(k)
    end
end
```

L'exécution de ces instructions fournit le résultat suivant :

$\lambda = +1$, $\mu = +1$, $k = 2.4949646$
$\lambda = +1$, $\mu = -1$, $k = 1.2285449$
$\lambda = -1$, $\mu = +1$, $k = -0.5284214 + 0.7686085i$
$\lambda = -1$, $\mu = -1$, $k = -0.5284214 - 0.7686085i$

La seule racine réelle k qui satisfait la double inégalité $0 \le k \le 2$ (équivalente à $0 \le L \le 2R$) est $k \approx 1.2285449$
Donc, pour $R = 100$ m par exemple : $L = k \times 100 \approx 122.85449$ m
Cette valeur est identique à celle obtenue en recherchant les zéros du polynôme $3L^4 - 8RL^3 + 8R^4$ avec la fonction *roots*.

En résumé, l'équation $3L^4 - 8RL^3 + 8R^4 = 0$ admet une unique solution L comprise entre 0 et $2R$:

$$L = kR \text{ avec } k = \frac{a - \sqrt{a^2 - 4b}}{2},\ a = \frac{4 + \sqrt{18x + 16}}{3},\ b = \frac{3x + \sqrt{9x^2 - 24}}{3},\ x = \frac{2}{3}\left(\sqrt[3]{4 + 2\sqrt{2}} + \sqrt[3]{4 - 2\sqrt{2}}\right)$$

(**) Complément sur la formule de **Cardan**

En fait, on devrait attribuer cette formule à **Tartaglia**. Pour se rappeler l'histoire de l'équation du troisième degré et la querelle qui opposa, au XVI[e] siècle, les algébristes italiens **Tartaglia**, **Cardan** et **Ferrari**, cliquer sur le lien internet **http://ljk.imag.fr/membres/Bernard.Ycart/mel/pf/node22.html**

L'équation à résoudre est :

$$x^3 + px + q = 0$$

Effectuons le changement de variable $x = X + \frac{m}{X}$ dans lequel X est l'inconnue et m désigne un paramètre :

$$\left(X + \frac{m}{X}\right)^3 + p\left(X + \frac{m}{X}\right) + q = 0$$

Après développement de cette expression et multiplication par X^3, on obtient :

$$X^6 + (3m + p)X^4 + qX^3 + m(3m + p)X^2 + m^3 = 0$$

Prenons $m = -\frac{p}{3}$ et abaissons le degré en posant $Y = X^3$:

$$Y^2 + qY - \frac{p^3}{27} = 0$$

Lorsque le discriminant $\Delta = q^2 + 4\frac{p^3}{27}$ de cette dernière équation est strictement positif, on a 2 racines réelles distinctes :

$$Y_1 = -\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3} \quad \text{et} \quad Y_2 = -\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}$$

D'où :

$$X_1 = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} \quad \text{et} \quad X_2 = \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}$$

Donc :

$$x_1 = X_1 - \frac{p}{3X_1} \quad \text{et} \quad x_2 = X_2 - \frac{p}{3X_2}$$

Exprimons le produit des racines de l'équation $Y^2 + qY - \frac{p^3}{27} = 0$ en fonction de ses coefficients :

$$Y_1 Y_2 = -\frac{p^3}{27} \Rightarrow X_1 X_2 = -\frac{p}{3} \Rightarrow X_1 = -\frac{p}{3X_2} \quad \text{et} \quad X_2 = -\frac{p}{3X_1} \Rightarrow x_1 = x_2 = X_1 + X_2$$

Finalement, l'équation $x^3 + px + q = 0$ admet une unique racine réelle si $4p^3 + 27q^2$ est strictement positif :

$$x = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}$$

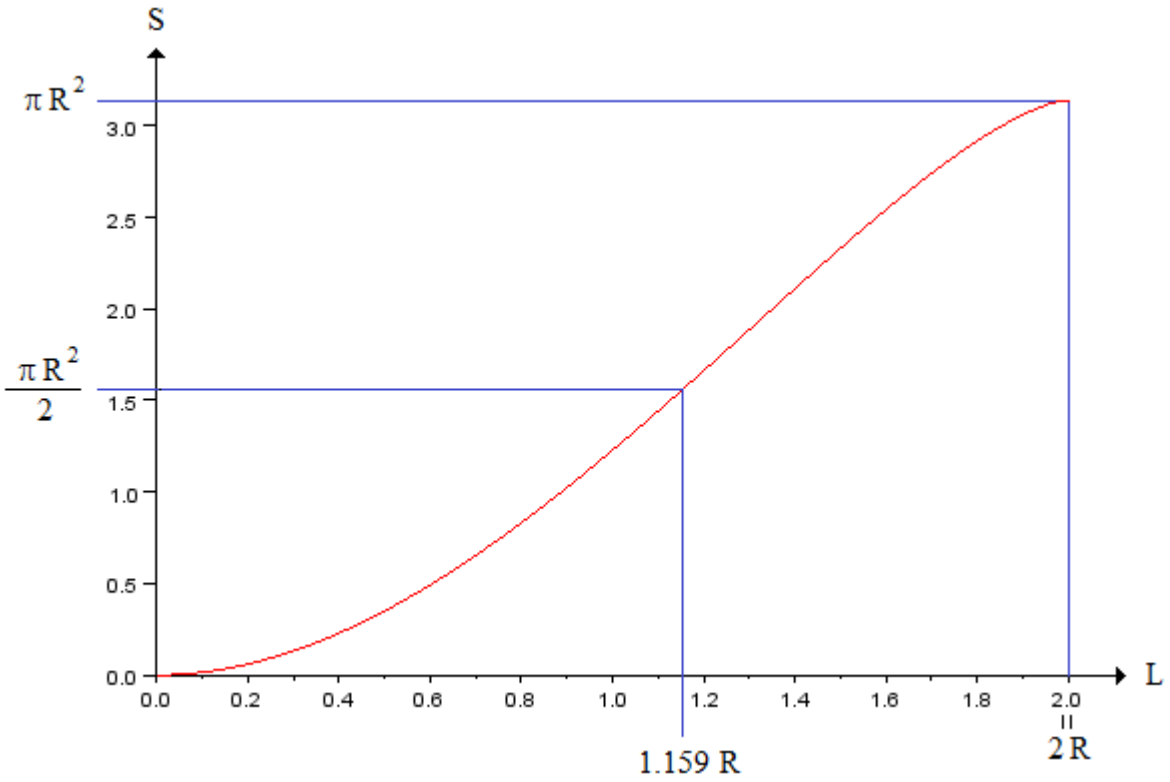
Note

L'équation du 3[e] degré $ay^3 + by^2 + cy + d = 0$ se ramène à la forme $x^3 + px + q = 0$ en posant $y = x - \frac{b}{3a}$

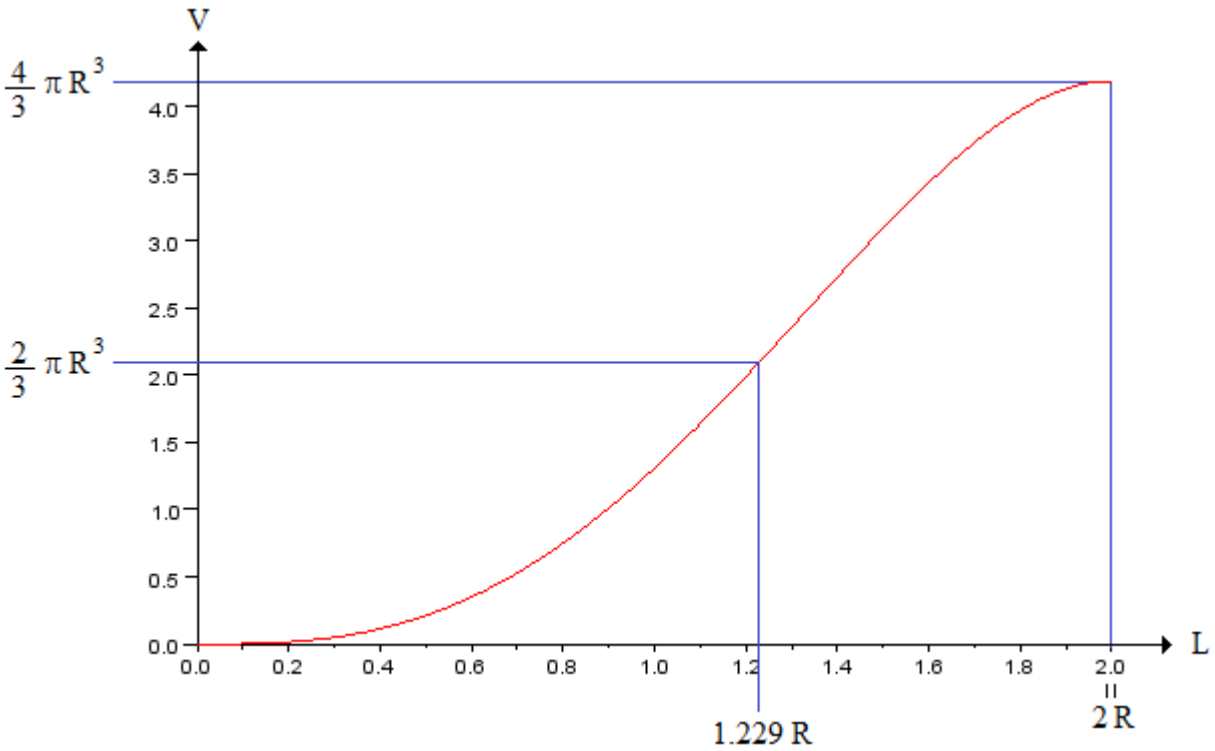
Mathématique & Informatique

Graphiques

- Dans le plan : $S = \pi R^2 - \frac{L}{2}\sqrt{4R^2 - L^2} - (2R^2 - L^2)\,\text{Arc}\cos\left(\frac{L}{2R}\right)$



- Dans l'espace : $V = \left(\frac{2}{3} - \frac{L}{4R}\right)\pi L^3$



Question annexe

Existe-t-il une valeur de L, différente de 0 et $2R$, pour laquelle le rapport de la surface S à celle du disque de rayon R est égal au rapport du volume V à celui de la boule de rayon R ?

Réponse

On recherche $L, 0 < L < 2R$, tel que : $\dfrac{S}{\pi R^2} = \dfrac{V}{\frac{4}{3}\pi R^3}$

Compte tenu des expressions de S et V établies précédemment, il s'agit de trouver L vérifiant :

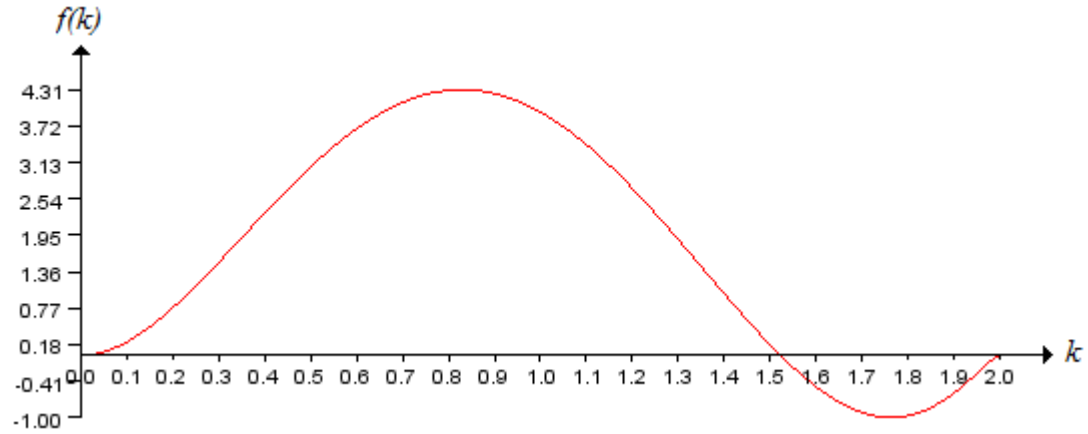
$$\frac{4}{3}\pi R^3\left\{\pi R^2 - \frac{L}{2}\sqrt{4R^2 - L^2} - (2R^2 - L^2)\,\text{Arc}\cos\left(\frac{L}{2R}\right)\right\} = \pi R^2\left\{\left(\frac{2}{3} - \frac{L}{4R}\right)\pi L^3\right\}$$

Posons $L = kR$, cette condition se met sous la forme simplifiée :

$$f(k) = 3\pi k^4 - 8\pi k^3 - 8k\sqrt{4 - k^2} - 16(2 - k^2)\text{Arc}\cos\left(\frac{k}{2}\right) + 16\pi = 0$$

Traçons la courbe de $f(k)$ pour $k \in [0,2]$:

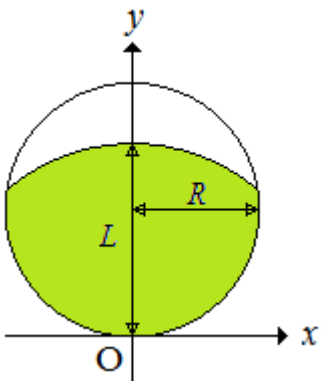
```
k = linspace(0,2,300) ;
f = 3*%pi*k^4 - 8*%pi*k^3 - 8*k.*sqrt(4 - k^2) - 16*(2 - k^2).*acos(k/2) + 16*%pi ;
a = gca() ; a.x_location = "origin" ; a.y_location = "origin" ;
plot2d(k,f,frameflag=2,nax=[0,21,0,10],style=5)
```



Le graphique suggère que l'équation $f(k) = 0$ admet une solution unique dans l'intervalle $]0,2[$ et que $k = 1.5$ est une approximation grossière de la racine, approximation qui servira de valeur initiale pour l'algorithme de convergence utilisé par la fonction Scilab *fsolve*. Les 2 instructions suivantes :

```
deff('y = f(k)','y = 3*%pi*k^4 - 8*%pi*k^3 - 8*k*sqrt(4 - k^2) - 16*(2 - k^2)*acos(k/2) + 16*%pi') ;
k = fsolve(1.5,f)
```

donnent la valeur numérique $k = 1.5227962$, d'où $L = 1.5227962R$ et $S/\pi R^2 = V/\frac{4}{3}\pi R^3 = (8 - 3k)k^3/16 = 0.7573619$

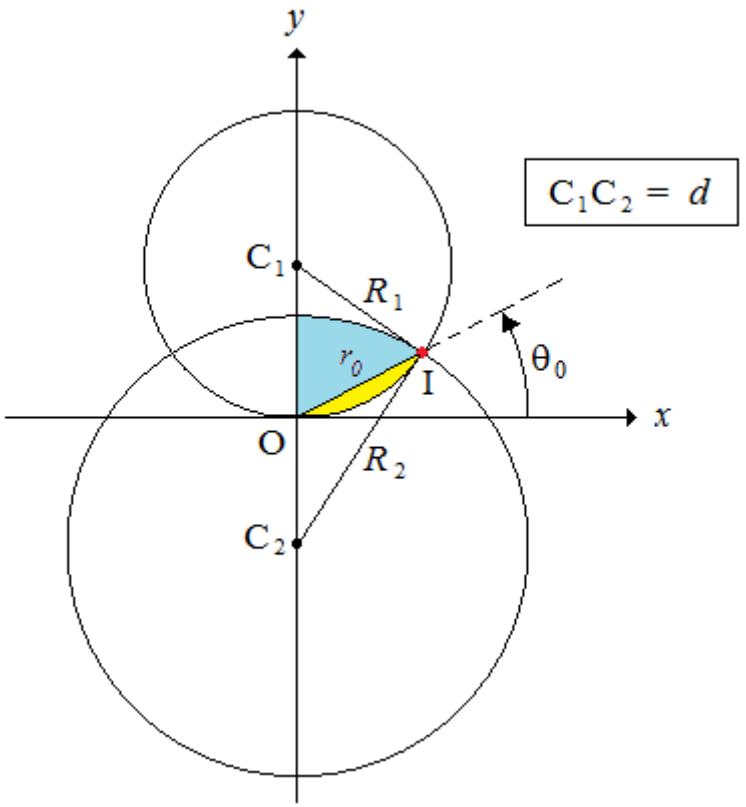


$L = 1.5227962R$

Mathématique & Informatique

Problème général

Etudions le cas de 2 cercles ou de 2 sphères de rayons R_1, R_2 et de centres C_1, C_2 distants de d.

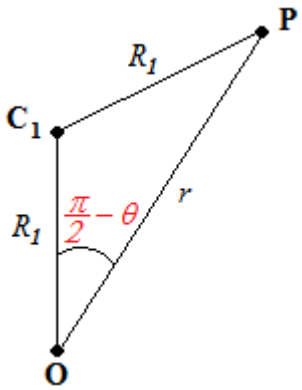


Notons S l'intersection des 2 disques et V l'intersection des 2 boules.
Remarquons que pour $R_2 \leq |d - R_1|$ ou $R_2 \geq d + R_1$ le calcul de S et V s'effectue sans difficulté : la surface S (le volume V) est égale à l'ensemble vide ou au point O ou au disque (à la boule) de rayon R_1 ou au disque (à la boule) de rayon R_2.
Dans ce qui suit on suppose : $|d - R_1| < R_2 < d + R_1$
Trois cas sont possibles : $d > R_1$ (C_2 est un point de l'axe des $y < 0$ à l'extérieur du 1er cercle, cas du schéma ci-dessus) ou $d = R_1$ (C_2 est confondu avec l'origine O du repère sur le 1er cercle, voir le problème de la chèvre) ou $d < R_1$ (C_2 est un point du segment $]O,C_1[$ à l'intérieur du 1er cercle).
On souhaite exprimer S et V en fonction de R_1, R_2, d.

Solution

Utilisons les coordonnées polaires (r,θ) dans le premier quadrant du repère Oxy.

1er cercle



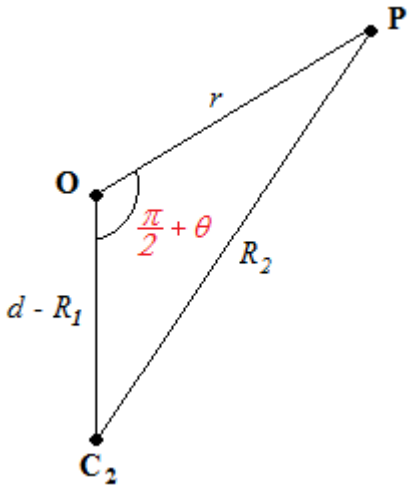
Appliquons le théorème d'**Al-Kashi** au triangle (O,C_1,P) :

$$R_1^2 = r^2 + R_1^2 - 2rR_1\cos\left(\frac{\pi}{2} - \theta\right)$$

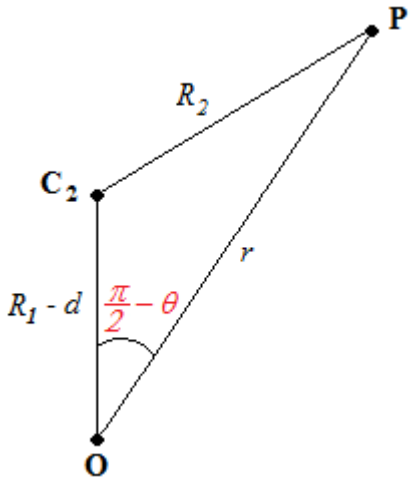
On déduit que le 1er cercle de rayon R_1 a pour équation polaire : $r = 2R_1\sin\theta$

2e cercle

$d \geq R_1$



$d \leq R_1$



$$R_2^2 = r^2 + (d - R_1)^2 - 2r(d - R_1)\cos\left(\frac{\pi}{2} + \theta\right) \qquad R_2^2 = r^2 + (R_1 - d)^2 - 2r(R_1 - d)\cos\left(\frac{\pi}{2} - \theta\right)$$

On déduit que le 2e cercle de rayon R_2 a pour équation polaire : $r = \sqrt{R_2^2 - (d - R_1)^2\cos^2\theta} - (d - R_1)\sin\theta$

L'angle polaire θ_0 du point I situé à l'intersection des 2 cercles vérifie :

$$2R_1\sin\theta_0 = \sqrt{R_2^2 - (d - R_1)^2\cos^2\theta_0} - (d - R_1)\sin\theta_0$$

On tire de cette relation la valeur de θ_0 : $\theta_0 = \text{Arc sin}\sqrt{\frac{R_2^2 - (d - R_1)^2}{4dR_1}}$

Divisons l'aire $A = \frac{S}{2}$ en deux parties (colorées en jaune et en bleu sur le schéma) :

- 1re partie (en jaune) : θ varie de 0 à θ_0 et r varie de 0 à $2R_1\sin\theta$ pour θ fixé
- 2e partie (en bleu) : θ varie de θ_0 à $\frac{\pi}{2}$ et r varie de 0 à $\sqrt{R_2^2 - (d - R_1)^2\cos^2\theta} - (d - R_1)\sin\theta$ pour θ fixé

La surface S et le volume V sont donnés par les formules :

$$S = 2\iint_A r\,dr\,d\theta$$

$$V = 2\pi\iint_A r^2\cos\theta\,dr\,d\theta \qquad \text{(d'après le théorème de } \textbf{Guldin}\text{)}$$

Le théorème de **Fubini** permet de ramener ces intégrales doubles à une succession d'intégrales simples, donc à la recherche de primitives. Le résultat est présenté sous la forme d'intégrales itérées :

$$S = 2\left\{\int_0^{\theta_0} d\theta \int_0^{2R_1\sin\theta} r\,dr + \int_{\theta_0}^{\frac{\pi}{2}} d\theta \int_0^{\sqrt{R_2^2 - (d - R_1)^2\cos^2\theta} - (d - R_1)\sin\theta} r\,dr\right\}$$

$$V = 2\pi\left\{\int_0^{\theta_0} d\theta \int_0^{2R_1\sin\theta} r^2\cos\theta\,dr + \int_{\theta_0}^{\frac{\pi}{2}} d\theta \int_0^{\sqrt{R_2^2 - (d - R_1)^2\cos^2\theta} - (d - R_1)\sin\theta} r^2\cos\theta\,dr\right\}$$

Intégrons d'abord en r :

$$S = \int_0^{\theta_0} (2R_1\sin\theta)^2 d\theta + \int_{\theta_0}^{\frac{\pi}{2}} \left(\sqrt{R_2^2 - (d - R_1)^2\cos^2\theta} - (d - R_1)\sin\theta \right)^2 d\theta$$

$$V = \frac{2}{3}\pi \left\{ \int_0^{\theta_0} (2R_1\sin\theta)^3 \cos\theta\, d\theta + \int_{\theta_0}^{\frac{\pi}{2}} \left(\sqrt{R_2^2 - (d - R_1)^2\cos^2\theta} - (d - R_1)\sin\theta \right)^3 \cos\theta\, d\theta \right\}$$

Ce qui s'écrit après développement :

$$S = \left\{ \begin{array}{l} 4R_1^2 \int_0^{\theta_0} \sin^2\theta\, d\theta + 2(d - R_1)^2 \int_{\theta_0}^{\frac{\pi}{2}} \sin^2\theta\, d\theta + \left(R_2^2 - (d - R_1)^2\right) \int_{\theta_0}^{\frac{\pi}{2}} d\theta \\ - 2(d - R_1) \int_{\theta_0}^{\frac{\pi}{2}} \sqrt{R_2^2 - (d - R_1)^2\cos^2\theta}\, \sin\theta\, d\theta \end{array} \right\}$$

$$V = \frac{2}{3}\pi \left\{ \begin{array}{l} 8R_1^3 \int_0^{\theta_0} \sin^3\theta\, \cos\theta\, d\theta - 4(d - R_1)^3 \int_{\theta_0}^{\frac{\pi}{2}} \sin^3\theta\, \cos\theta\, d\theta - 3(d - R_1)\left(R_2^2 - (d - R_1)^2\right) \int_{\theta_0}^{\frac{\pi}{2}} \sin\theta\, \cos\theta\, d\theta \\ + \left(R_2^2 - (d - R_1)^2\right) \int_{\theta_0}^{\frac{\pi}{2}} \sqrt{\left(R_2^2 - (d - R_1)^2\right) + (d - R_1)^2\sin^2\theta}\, \cos\theta\, d\theta \\ + 4(d - R_1)^2 \int_{\theta_0}^{\frac{\pi}{2}} \sin^2\theta\, \sqrt{\left(R_2^2 - (d - R_1)^2\right) + (d - R_1)^2\sin^2\theta}\, \cos\theta\, d\theta \end{array} \right\}$$

Ensuite, pour intégrer en θ, d'une part on utilise les primitives usuelles

$$\int d\theta = \theta \ , \ \int \sin^2\theta\, d\theta = \frac{1}{2}(\theta - \sin\theta\cos\theta) \ , \ \int \sin^n\theta\cos\theta\, d\theta = \frac{\sin^{n+1}\theta}{n+1} \quad (n = 1,\ n = 3)$$

d'autre part on effectue les changements de variable $x = (d - R_1)\cos\theta$ avec $a = R_2$ dans S et $x = (d - R_1)\sin\theta$ avec $a = \sqrt{R_2^2 - (d - R_1)^2}$ dans V, de façon à faire apparaître les intégrales suivantes [*] :

$$I_1 = \int \sqrt{a^2 - x^2}\, dx = \frac{x\sqrt{a^2 - x^2}}{2} + \frac{a^2}{2}\text{Arc}\sin\frac{x}{a}$$

$$I_2 = \int \sqrt{a^2 + x^2}\, dx = \frac{x\sqrt{a^2 + x^2}}{2} + \frac{a^2}{2}\ln\left(x + \sqrt{a^2 + x^2}\right)$$

$$I_3 = \int x^2\sqrt{a^2 + x^2}\, dx = \frac{x(a^2 + 2x^2)\sqrt{a^2 + x^2}}{8} - \frac{a^4}{8}\ln\left(x + \sqrt{a^2 + x^2}\right)$$

[*] Méthode de calcul de I_1, I_2, I_3 : pour I_1 poser $x = a\sin t$, pour I_2 poser $x = a\,\text{sh}\,t$, pour I_3 intégrer par parties $a^2I_2 + I_3$

D'où :

$$S = \left\{ \begin{array}{l} 2R_1^2[\theta - \sin\theta\cos\theta]_0^{\theta_0} + (d - R_1)^2[\theta - \sin\theta\cos\theta]_{\theta_0}^{\frac{\pi}{2}} + \left(R_2^2 - (d - R_1)^2\right)[\theta]_{\theta_0}^{\frac{\pi}{2}} \\ + \left[x\sqrt{R_2^2 - x^2} + R_2^2 \operatorname{Arc}\sin\frac{x}{R_2} \right]_{(d - R_1)\cos\theta_0}^{0} \end{array} \right\}$$

$$V = \frac{2}{3}\pi \left\{ \begin{array}{l} 2R_1^3\left[\sin^4\theta\right]_0^{\theta_0} - (d - R_1)^3\left[\sin^4\theta\right]_{\theta_0}^{\frac{\pi}{2}} - \frac{3}{2}(d - R_1)\left(R_2^2 - (d - R_1)^2\right)\left[\sin^2\theta\right]_{\theta_0}^{\frac{\pi}{2}} \\ + \frac{R_2^2 - (d - R_1)^2}{2(d - R_1)}\left[x\sqrt{R_2^2 - (d - R_1)^2 + x^2} + \left(R_2^2 - (d - R_1)^2\right)\ln\left(x + \sqrt{R_2^2 - (d - R_1)^2 + x^2} \right) \right]_{(d - R_1)\sin\theta_0}^{(d - R_1)} \\ + \frac{1}{2(d - R_1)}\left[x\left(R_2^2 - (d - R_1)^2 + 2x^2\right)\sqrt{R_2^2 - (d - R_1)^2 + x^2} - \left(R_2^2 - (d - R_1)^2\right)^2 \ln\left(x + \sqrt{R_2^2 - (d - R_1)^2 + x^2} \right) \right]_{(d - R_1)\sin\theta_0}^{(d - R_1)} \end{array} \right\}$$

Donc :

$$S = \left\{ \begin{array}{l} 2R_1^2(\theta_0 - \sin\theta_0\cos\theta_0) + (d - R_1)^2\left(\frac{\pi}{2} - \theta_0 + \sin\theta_0\cos\theta_0\right) + \left(R_2^2 - (d - R_1)^2\right)\left(\frac{\pi}{2} - \theta_0\right) \\ - (d - R_1)\cos\theta_0\sqrt{R_2^2 - (d - R_1)^2\cos^2\theta_0} - R_2^2 \operatorname{Arc}\sin\frac{(d - R_1)\cos\theta_0}{R_2} \end{array} \right\}$$

$$V = \frac{2}{3}\pi \left\{ \begin{array}{l} 2R_1^3\sin^4\theta_0 - (d - R_1)^3\left(1 - \sin^4\theta_0\right) - \frac{3}{2}(d - R_1)\left(R_2^2 - (d - R_1)^2\right)\left(1 - \sin^2\theta_0\right) \\ + \frac{R_2^2 - (d - R_1)^2}{2(d - R_1)}\left[\begin{array}{l} R_2(d - R_1) + \left(R_2^2 - (d - R_1)^2\right)\ln(d - R_1 + R_2) \\ - (d - R_1)\sin\theta_0\sqrt{R_2^2 - (d - R_1)^2 + (d - R_1)^2\sin^2\theta_0} \\ - \left(R_2^2 - (d - R_1)^2\right)\ln\left((d - R_1)\sin\theta_0 + \sqrt{R_2^2 - (d - R_1)^2 + (d - R_1)^2\sin^2\theta_0} \right) \end{array} \right] \\ + \frac{1}{2(d - R_1)}\left[\begin{array}{l} R_2(d - R_1)\left(R_2^2 + (d - R_1)^2\right) - \left(R_2^2 - (d - R_1)^2\right)^2\ln(d - R_1 + R_2) \\ - (d - R_1)\sin\theta_0\left(R_2^2 - (d - R_1)^2 + 2(d - R_1)^2\sin^2\theta_0\right)\sqrt{R_2^2 - (d - R_1)^2 + (d - R_1)^2\sin^2\theta_0} \\ + \left(R_2^2 - (d - R_1)^2\right)^2\ln\left((d - R_1)\sin\theta_0 + \sqrt{R_2^2 - (d - R_1)^2 + (d - R_1)^2\sin^2\theta_0} \right) \end{array} \right] \end{array} \right\}$$

Simplifions :

$$S = \left\{ \begin{array}{l} 2R_1^2(\theta_0 - \sin\theta_0\cos\theta_0) + (d - R_1)^2\sin\theta_0\cos\theta_0 + R_2^2\left(\frac{\pi}{2} - \theta_0\right) \\ - (d - R_1)\cos\theta_0\sqrt{R_2^2 - (d - R_1)^2\cos^2\theta_0} - R_2^2 \operatorname{Arc}\sin\frac{(d - R_1)\cos\theta_0}{R_2} \end{array} \right\}$$

$$V = \frac{2}{3}\pi \left\{ \begin{array}{l} 2R_1^3\sin^4\theta_0 - (d - R_1)^3\left(1 - \sin^4\theta_0\right) - \frac{3}{2}(d - R_1)\left(R_2^2 - (d - R_1)^2\right)\cos^2\theta_0 \\ + R_2^3 - \sin\theta_0\left(R_2^2 - (d - R_1)^2\cos^2\theta_0\right)\sqrt{R_2^2 - (d - R_1)^2\cos^2\theta_0} \end{array} \right\}$$

Regroupons les termes en θ_0 :

$$S = \left\{ \frac{1}{2}\pi R_2^2 + \left(2R_1^2 - R_2^2\right)\theta_0 - \left(2R_1^2 - (d - R_1)^2\right)\sin\theta_0 \cos\theta_0 - (d - R_1)\cos\theta_0 \sqrt{R_2^2 - (d - R_1)^2\cos^2\theta_0} - R_2^2 \operatorname{Arc\,sin} \frac{(d - R_1)\cos\theta_0}{R_2} \right\}$$

$$V = \frac{2}{3}\pi \left\{ R_2^3 - (d - R_1)^3 + \left(2R_1^3 + (d - R_1)^3\right)\sin^4\theta_0 - \frac{3}{2}(d - R_1)\left(R_2^2 - (d - R_1)^2\right)\cos^2\theta_0 - \sin\theta_0 \left(R_2^2 - (d - R_1)^2\cos^2\theta_0\right)\sqrt{R_2^2 - (d - R_1)^2\cos^2\theta_0} \right\}$$

Il reste à remplacer θ_0, $\sin\theta_0$, $\cos\theta_0$ par leurs valeurs :

$$\theta_0 = \operatorname{Arc\,sin}\sqrt{\frac{R_2^2 - (d - R_1)^2}{4dR_1}}, \quad \sin\theta_0 = \sqrt{\frac{R_2^2 - (d - R_1)^2}{4dR_1}}, \quad \cos\theta_0 = \sqrt{\frac{(d + R_1)^2 - R_2^2}{4dR_1}}$$

On obtient :

$$S = \left\{ \begin{array}{l} \frac{1}{2}\pi R_2^2 + \left(2R_1^2 - R_2^2\right)\operatorname{Arc\,sin}\sqrt{\frac{R_2^2 - (d - R_1)^2}{4dR_1}} - R_2^2 \operatorname{Arc\,sin}\left(\left(\frac{d - R_1}{R_2}\right)\sqrt{\frac{(d + R_1)^2 - R_2^2}{4dR_1}}\right) \\ - \left(\frac{2R_1^2 - (d - R_1)^2}{4dR_1}\right)\sqrt{\left(R_2^2 - (d - R_1)^2\right)\left((d + R_1)^2 - R_2^2\right)} - \left(\frac{d - R_1}{4dR_1}\right)\sqrt{(d + R_1)^2 - R_2^2}\sqrt{4dR_1R_2^2 - (d - R_1)^2\left((d + R_1)^2 - R_2^2\right)} \end{array} \right\}$$

$$V = \frac{2}{3}\pi \left\{ \begin{array}{l} R_2^3 - (d - R_1)^3 + \left(2R_1^3 + (d - R_1)^3\right)\left(\frac{R_2^2 - (d - R_1)^2}{4dR_1}\right)^2 - \frac{3}{2}(d - R_1)\left(R_2^2 - (d - R_1)^2\right)\left(\frac{(d + R_1)^2 - R_2^2}{4dR_1}\right) \\ - \left(\frac{4dR_1R_2^2 - (d - R_1)^2\left((d + R_1)^2 - R_2^2\right)}{(4dR_1)^2}\right)\sqrt{R_2^2 - (d - R_1)^2}\sqrt{4dR_1R_2^2 - (d - R_1)^2\left((d + R_1)^2 - R_2^2\right)} \end{array} \right\}$$

Soit, après simplification :

$$S = \frac{1}{2}\pi R_2^2 + \left(2R_1^2 - R_2^2\right)\operatorname{Arc\,sin}\sqrt{\frac{R_2^2 - (d - R_1)^2}{4dR_1}} - R_2^2 \operatorname{Arc\,sin}\left(\frac{d - R_1}{R_2}\sqrt{\frac{(d + R_1)^2 - R_2^2}{4dR_1}}\right) - \frac{1}{2}\sqrt{\left(R_2^2 - (d - R_1)^2\right)\left((d + R_1)^2 - R_2^2\right)}$$

$$V = \frac{\pi}{12d}\left\{ 8d\left(R_2^3 - (d - R_1)^3\right) + 3\left(R_2^2 - (d - R_1)^2\right)\left((d + R_1)^2 - R_2^2 - 4d^2\right) \right\}$$

On améliore l'expression de S en enlevant les racines carrées dans les fonctions Arc sin.
Pour cela, on met en facteur R_1^2 et R_2^2 puis on utilise les égalités suivantes :

$$\operatorname{Arc\,sin} u = \frac{1}{2}\operatorname{Arc\,cos}(1 - 2u^2)$$

$$\operatorname{Arc\,sin} u + \operatorname{Arc\,sin} v = \operatorname{Arc\,sin}\left(u\sqrt{1 - v^2} + v\sqrt{1 - u^2}\right)$$

$$\operatorname{Arc\,sin} u + \operatorname{Arc\,cos} u = \frac{\pi}{2}$$

Finalement, les expressions de la surface S et du volume V en fonction des 2 rayons R_1, R_2 et de la distance d sont :

$$S = R_1^2 \operatorname{Arc\,cos}\left(\frac{R_1^2 - R_2^2 + d^2}{2dR_1}\right) + R_2^2 \operatorname{Arc\,cos}\left(\frac{R_2^2 - R_1^2 + d^2}{2dR_2}\right) - \frac{1}{2}\sqrt{(R_1 + R_2 - d)(R_1 + R_2 + d)(R_1 - R_2 + d)(R_2 - R_1 + d)}$$

$$V = \frac{\pi}{12d}(R_1 + R_2 - d)^2\left[(R_1 + R_2 + d)^2 - 4(R_1^2 - R_1R_2 + R_2^2)\right]$$

Notons que par construction géométrique, R_1 et R_2 jouent des rôles symétriques dans S et V :

$S = D(C_1,R_1) \cap D(C_2,R_2) = D(C_2,R_2) \cap D(C_1,R_1)$ et $V = B(C_1,R_1) \cap B(C_2,R_2) = B(C_2,R_2) \cap B(C_1,R_1)$

Par suite, on peut intervertir les variables R_1 et R_2 dans les fonctions S et V :

$S(R_1, R_2, d) = S(R_2, R_1, d)$ et $V(R_1, R_2, d) = V(R_2, R_1, d)$

Il est facile de vérifier cette propriété avec les expressions de S et V ci-dessus.

Programme

Ecrivons une fonction Scilab qui calcule S et V à partir des paramètres R_1, R_2, d :

```
function [S,V] = F(R1,R2,d)
    S = R1^2*acos((R1^2 - R2^2 + d^2)/(2*d*R1)) + R2^2*acos((R2^2 - R1^2 + d^2)/(2*d*R2)) ...
        - (1/2)*sqrt((R1 + R2 - d)*(R1 + R2 + d)*(R1 - R2 + d)*(R2 - R1 + d))
    V = (%pi/(12*d))*((R1 + R2 - d)^2)*((R1 + R2 + d)^2 - 4*(R1^2 - R1*R2 + R2^2))
endfunction
```

Par exemple :

- pour R_1 = 10 m, R_2 = 20 m, d = 15 m, l'instruction *[S,V] = F(10,20,15)* donne S = 239.25499 m^2, V = 3239.7674 m^3
- pour R_1 = 10 m, R_2 = 15 m, d = 10 m, l'instruction *[S,V] = F(10,15,10)* donne S = 233.01195 m^2, V = 3092.5053 m^3
- pour R_1 = 10 m, R_2 = 10 m, d = 5 m, l'instruction *[S,V] = F(10,10,5)* donne S = 215.21092 m^2, V = 2650.7188 m^3

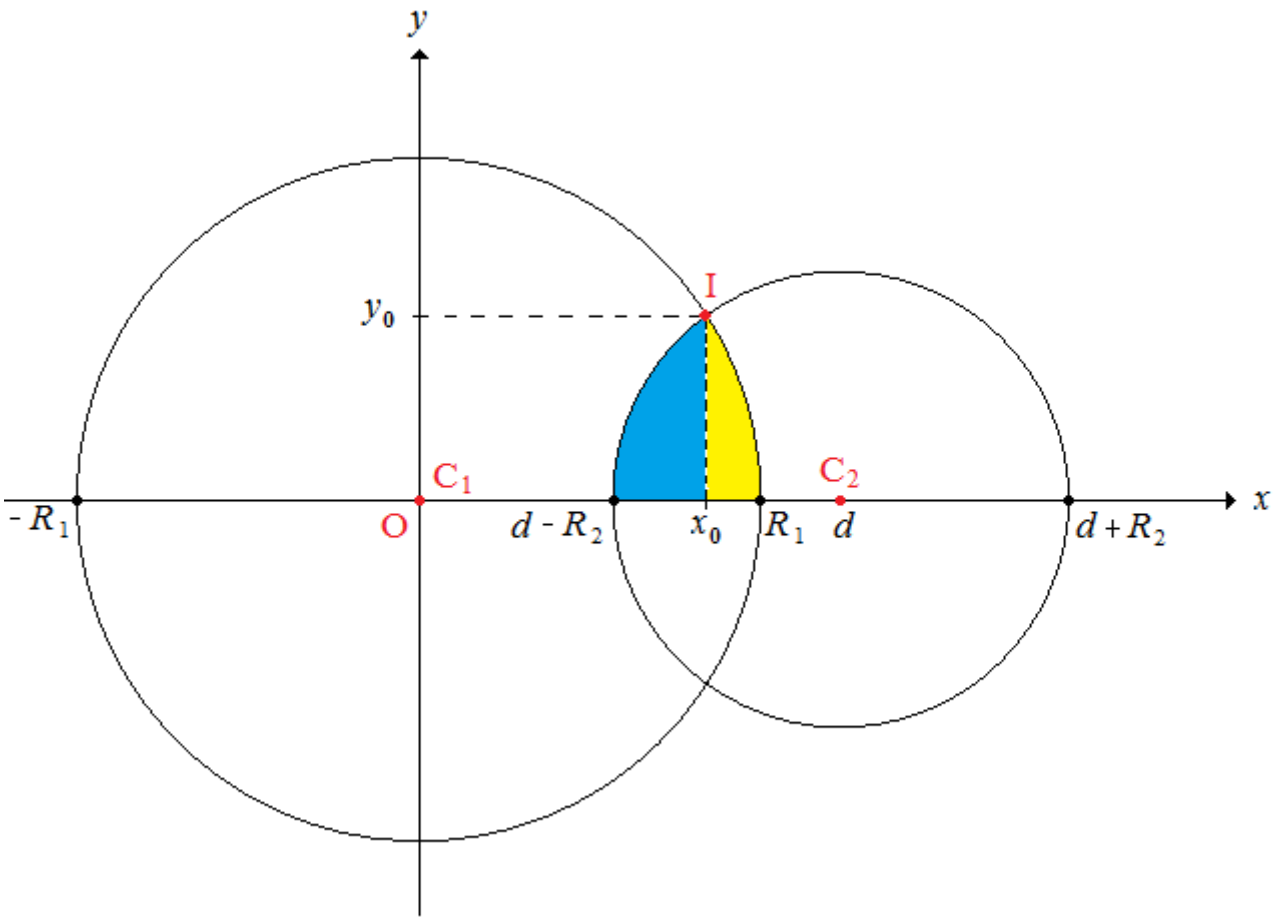
Remarque

L'emploi des coordonnées polaires est judicieux dans le cas particulier $d = R_1$, mais il s'avère compliqué à mettre en œuvre dans le cas général. Mieux vaut alors revenir aux classiques coordonnées cartésiennes.

Mathématique & Informatique

Problème général (deuxième version, plus simple)

Soient 2 cercles ou 2 sphères de rayons R_1, R_2 et de centres C_1, C_2 distants de d.



On suppose : $|d - R_1| < R_2 < d + R_1$
Désignons par S l'intersection des 2 disques et par V l'intersection des 2 boules.
Exprimer S et V en fonction de R_1, R_2, d.

Solution

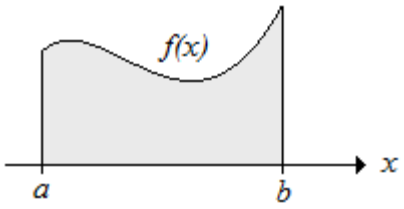
Utilisons les coordonnées cartésiennes (x,y) dans le premier quadrant du repère Oxy.

Equation du 1er cercle : $x^2 + y^2 = R_1^2$
Equation du 2e cercle : $(x-d)^2 + y^2 = R_2^2$

L'abscisse x_0 du point d'intersection des 2 cercles est donnée par :

$$R_1^2 - x_0^2 = R_2^2 - (x_0 - d)^2 \quad \Rightarrow \quad x_0 = \frac{d^2 + R_1^2 - R_2^2}{2d}$$

Rappelons deux formules fondamentales du calcul intégral :



- L'aire située au dessus de l'axe des x et délimitée par la courbe $y = f(x)$, la verticale $x = a$, la verticale $x = b$, est :

$$S = \int_a^b f(x)dx$$

- Le volume engendré par la rotation de cette aire autour de l'axe des x est :

$$V = \pi \int_a^b f^2(x)dx$$

Appliquons ces résultats à la surface S et au volume V du problème :

$$S = 2\left\{\int_{d-R_2}^{x_0} \sqrt{R_2^2 - (x-d)^2}\, dx + \int_{x_0}^{R_1} \sqrt{R_1^2 - x^2}\, dx\right\}$$

$$V = \pi\left\{\int_{d-R_2}^{x_0} \left(R_2^2 - (x-d)^2\right)dx + \int_{x_0}^{R_1} \left(R_1^2 - x^2\right)dx\right\}$$

Sachant que :

$$\int \sqrt{a^2 - x^2}\, dx = \frac{x\sqrt{a^2 - x^2}}{2} + \frac{a^2}{2}\operatorname{Arc\,sin}\frac{x}{a} + C \qquad (\text{poser } x = a\sin t)$$

Il vient :

$$S = \left[X\sqrt{R_2^2 - X^2} + R_2^2 \operatorname{Arc\,sin}\frac{X}{R_2}\right]_{-R_2}^{x_0 - d} + \left[x\sqrt{R_1^2 - x^2} + R_1^2 \operatorname{Arc\,sin}\frac{x}{R_1}\right]_{x_0}^{R_1}$$

$$V = \pi\left\{\left[R_2^2 X - \frac{X^3}{3}\right]_{-R_2}^{x_0 - d} + \left[R_1^2 x - \frac{x^3}{3}\right]_{x_0}^{R_1}\right\}$$

On développe ces 2 expressions comme des polynômes en d et on utilise la relation $\operatorname{Arc\,sin} x + \operatorname{Arc\,cos} x = \frac{\pi}{2}$ pour aboutir à :

$$S = R_1^2 \operatorname{Arc\,cos}\left(\frac{d^2 + R_1^2 - R_2^2}{2R_1 d}\right) + R_2^2 \operatorname{Arc\,cos}\left(\frac{d^2 - R_1^2 + R_2^2}{2R_2 d}\right) - \frac{1}{2}\sqrt{-d^4 + 2(R_1^2 + R_2^2)d^2 - (R_1^2 - R_2^2)^2}$$

$$V = \frac{\pi}{12d}\left(d^4 - 6(R_1^2 + R_2^2)d^2 + 8(R_1^3 + R_2^3)d - 3(R_1^2 - R_2^2)^2\right)$$

Mathématique & Informatique

Programme

Ecrivons une fonction Scilab qui calcule S et V à partir des paramètres R_1, R_2, d :

```
function [S,V] = F(R1,R2,d)
    S = R1^2*acos((d^2 + R1^2 - R2^2)/(2*R1*d)) + R2^2*acos((d^2 - R1^2 + R2^2)/(2*R2*d)) ...
        - (1/2)*sqrt(- d^4 + 2*(R1^2 + R2^2)*d^2 - (R1^2 - R2^2)^2)
    V = (%pi/(12*d))*(d^4 - 6*(R1^2 + R2^2)*d^2 + 8*(R1^3 + R2^3)*d - 3*(R1^2 - R2^2)^2)
endfunction
```

Par exemple, pour $R_1 = 4.5$ m, $R_2 = 3.0$ m, $d = 5.6$ m, l'instruction *[S,V] = F(4.5,3.0,5.6)* donne $S = 6.31$ m^2, $V = 18.33$ m^3

Application numérique

Une première sphère a un rayon R_1 connu. Une deuxième sphère de rayon R_2 inconnu intersecte la première sphère. Les centres des 2 sphères sont séparés d'une distance d inconnue. On suppose que l'on peut mesurer avec précision le volume V de l'intersection des 2 boules ainsi que la surface S de la projection plane de cette intersection.
Calculer R_2 et d pour : $R_1 = 10.00$ m, $S = 7.89$ m^2, $V = 34.56$ m^3

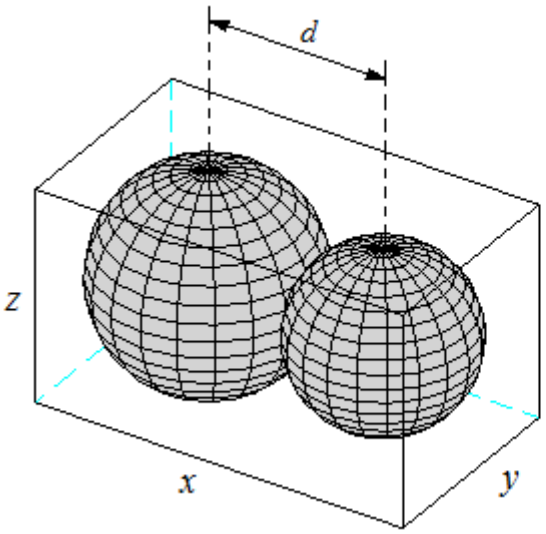
R_2 et d vérifient un système non linéaire de 2 équations à 2 inconnues :

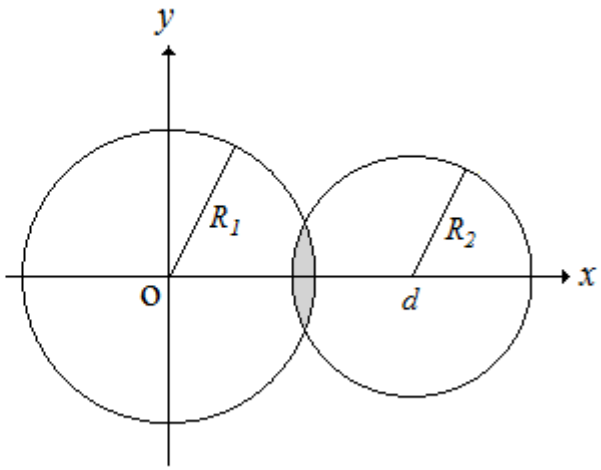
$$\begin{cases} R_1^2 \operatorname{Arc\,cos}\left(\dfrac{d^2 + R_1^2 - R_2^2}{2R_1 d}\right) + R_2^2 \operatorname{Arc\,cos}\left(\dfrac{d^2 - R_1^2 + R_2^2}{2R_2 d}\right) - \dfrac{1}{2}\sqrt{-d^4 + 2(R_1^2 + R_2^2)d^2 - (R_1^2 - R_2^2)^2} - S = 0 \\ \dfrac{\pi}{12d}\left(d^4 - 6(R_1^2 + R_2^2)d^2 + 8(R_1^3 + R_2^3)d - 3(R_1^2 - R_2^2)^2\right) - V = 0 \end{cases}$$

Pour trouver R_2 et d, on utilise la fonction Scilab *fsolve* en partant de la valeur initiale $(R_2,d) = (R_1,R_1)$

```
function y = f(x)
    y(1) = R1^2*acos((x(2)^2 + R1^2 - x(1)^2)/(2*R1*x(2))) + x(1)^2*acos((x(2)^2 - R1^2 + x(1)^2)/(2*x(1)*x(2))) ...
           - (1/2)*sqrt(- x(2)^4 + 2*(R1^2 + x(1)^2)*x(2)^2 - (R1^2 - x(1)^2)^2) - S
    y(2) = (%pi/(12*x(2)))*(x(2)^4 - 6*(R1^2 + x(1)^2)*x(2)^2 + 8*(R1^3 + x(1)^3)*x(2) - 3*(R1^2 - x(1)^2)^2) - V
endfunction

R1 = 10.00 ;
S = 7.89 ;
V = 34.56 ;
x0 = [R1;R1] ;
x = fsolve(x0,f) ;
R2 = x(1) ;
d = x(2) ;
printf('\n R2 = %f',R2)
printf('\n d = %f',d)
[S,V] = F(R1,R2,d) ;
printf('\n S = %f',S)
printf('\n V = %f',V)
```

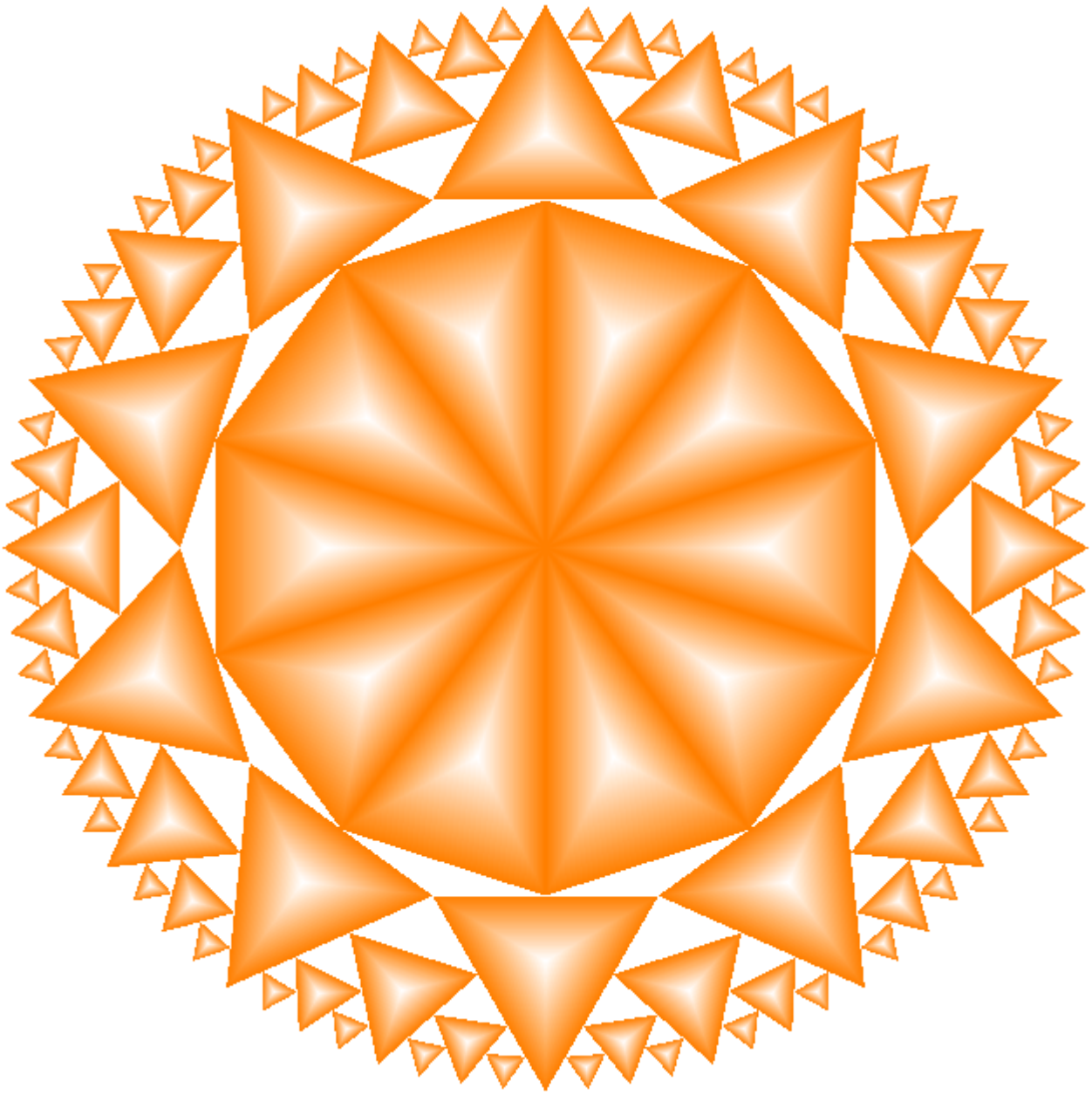




Résultat :

$R_2 = 8.202400$ m
$d = 16.615742$ m

Mathématique & Informatique

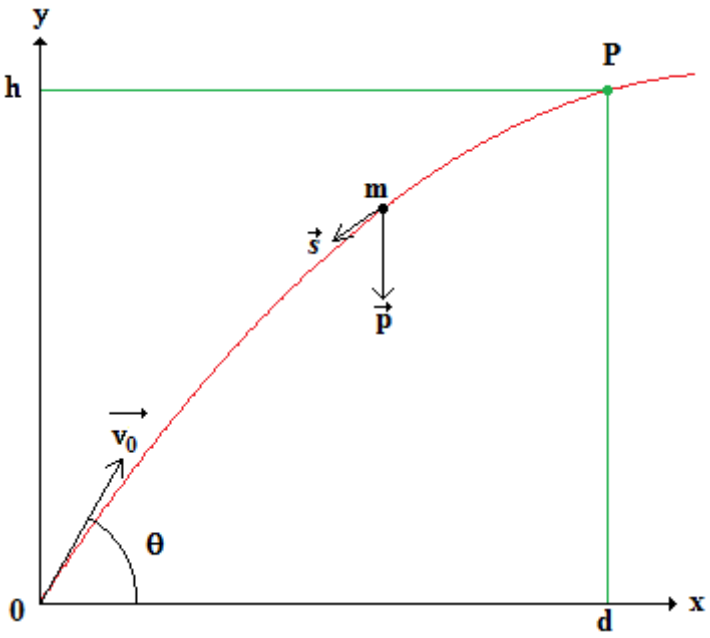


Mathématique & Informatique

BALISTIQUE

On lance un projectile de masse **m** avec une vitesse $\vec{v_0}$ faisant un angle **θ** avec l'horizontale.
La vitesse à l'instant t est $\vec{v}(v_x , v_y)$. On suppose que les seules forces qui s'appliquent sont le poids $\vec{p}(0 , - mg)$ où g est l'accélération de la pesanteur, et une force supplémentaire $\vec{s}(- kv_x^2 , - kv_y^2)$ où k est une constante.

Dans le même plan, un objet effectue un mouvement défini par les équations horaires X(t), Y(t).
Quelle valeur doit on donner à θ pour que les 2 trajectoires se rencontrent ?



Appliquons le Principe Fondamental de la Dynamique :

$$\vec{p} + \vec{s} = m\vec{a}$$

En projetant sur les axes *x* et *y* :

$$\begin{cases} 0 - kv_x^2 = m\dfrac{d^2x}{dt^2} \\ -mg - kv_y^2 = m\dfrac{d^2y}{dt^2} \end{cases}$$

Comme $v_x = \dfrac{dx}{dt}$ et $v_y = \dfrac{dy}{dt}$, le système différentiel s'écrit :

$$\begin{cases} \dfrac{dv_x}{dt} + \dfrac{k}{m}v_x^2 = 0 \\ \dfrac{dv_y}{dt} + \dfrac{k}{m}v_y^2 + g = 0 \end{cases}$$

avec pour conditions initiales à $t = 0$: $\quad x = 0$ et $v_x = v_0 \cos\theta$
$y = 0$ et $v_y = v_0 \sin\theta$

Mathématique & Informatique

Utilisons les résultats suivants :

- L'équation de **Riccati** $\frac{du}{dt} + \lambda u^2 = 0$ admet pour solution $u = \frac{1}{\lambda t + Cte}$
- L'équation de **Riccati** $\frac{du}{dt} + \lambda u^2 + \mu = 0$ admet pour solution $u = \sqrt{\frac{\mu}{\lambda}} \operatorname{cotan}\left(\sqrt{\lambda\mu}\, t + Cte\right)$

En particulier, pour $\lambda = \frac{k}{m}$, $\mu = g$, et avec les conditions initiales $v_x(0) = v_0 \cos\theta$, $v_y(0) = v_0 \sin\theta$, on obtient :

$$\begin{cases} v_x = \dfrac{v_0 \cos\theta}{\left(\dfrac{k}{m} v_0 \cos\theta\right) t + 1} \\ v_y = \sqrt{\dfrac{mg}{k}} \operatorname{cotan}\left(\sqrt{\dfrac{kg}{m}}\, t + \operatorname{Arc\,cotan}\left(\sqrt{\dfrac{k}{mg}} v_0 \sin\theta\right)\right) \end{cases}$$

Des intégrales $\int \frac{du}{u} = \operatorname{Log} u$, $\int \operatorname{cotan} u\, du = \operatorname{Log}(\sin u)$, et des conditions initiales $x(0) = 0$, $y(0) = 0$, on déduit :

$$\begin{cases} x = \displaystyle\int v_x\, dt = \frac{m}{k} \operatorname{Log}\left[1 + \left(\frac{k}{m} v_0 \cos\theta\right) t\right] \\ y = \displaystyle\int v_y\, dt = \frac{m}{k} \operatorname{Log}\left[\cos\left(\sqrt{\frac{kg}{m}}\, t\right) + \sqrt{\frac{k}{mg}} v_0 \sin\theta \sin\left(\sqrt{\frac{kg}{m}}\, t\right)\right] \end{cases}$$

Le problème consiste à rechercher θ et t tels que $x = X(t)$ et $y = Y(t)$, c'est-à-dire à résoudre :

$$\begin{cases} \left(\dfrac{k}{m} v_0 \cos\theta\right) t + 1 - e^{\frac{k}{m} X(t)} = 0 \\ \cos\left(\sqrt{\dfrac{kg}{m}}\, t\right) + \sqrt{\dfrac{k}{mg}} v_0 \sin\theta \sin\left(\sqrt{\dfrac{kg}{m}}\, t\right) - e^{\frac{k}{m} Y(t)} = 0 \end{cases}$$

Il s'agit d'un système de 2 équations non linéaires à 2 inconnues θ et t

On le résoud avec la fonction *fsolve* en partant des valeurs initiales $\theta_0 = \frac{\pi}{4}$ (angle médian) et $t_0 = \frac{\sqrt{X(0)^2 + Y(0)^2}}{v_0}$

Application numérique : $m = 1$ kg ; $k = 0.005$ USI ; $g = 9.81$ m/s^2 ; $v_0 = 40$ m/s.

Cas 1

L'objet est immobile au point P(d,h). Donc : $X(t) = d$ et $Y(t) = h$. On prend : $d = 101$ m, $h = 9$ m.

Cas 2

L'objet se déplace en ligne droite avec une vitesse v_1 qui fait un angle α avec l'horizontale. A $t = 0$, il est en (D,H).
Donc : $X(t) = (v_1 \cos\alpha)t + D$ et $Y(t) = (v_1 \sin\alpha)t + H$. On prend : $v_1 = 27$ m/s, $\alpha = 0.15$ rad, $D = 0$ m, $H = 20$ m.

Cas 3

L'objet effectue un mouvement hyperbolique d'équations : $X(t) = 41(2 - \operatorname{ch}(t))$ et $Y(t) = 15\operatorname{sh}(t)$

Ecrivons le programme Scilab en notant $x(1)$ la variable θ et $x(2)$ la variable t :

```
function [X,Y] = F1(t)
  X = 101 ; Y = 9
endfunction
function [X,Y] = F2(t)
   X = 27*cos(0.15)*t ; Y = 27*sin(0.15)*t + 20
endfunction
function [X,Y] = F3(t)
   X = 41*(2 - cosh(t)) ; Y = 15*sinh(t)
endfunction

function y = f1(x)
  [X,Y] = F1(x(2))
  y(1) = k/m*v0*cos(x(1))*x(2) + 1 - exp(k/m*X)
  y(2) = cos(sqrt(k*g/m)*x(2)) + sqrt(k/(m*g))*v0*sin(x(1))*sin(sqrt(k*g/m)*x(2)) - exp(k/m*Y)
endfunction
function y = f2(x)
  [X,Y] = F2(x(2))
  y(1) = k/m*v0*cos(x(1))*x(2) + 1 - exp(k/m*X)
  y(2) = cos(sqrt(k*g/m)*x(2)) + sqrt(k/(m*g))*v0*sin(x(1))*sin(sqrt(k*g/m)*x(2)) - exp(k/m*Y)
endfunction
function y = f3(x)
  [X,Y] = F3(x(2))
  y(1) = k/m*v0*cos(x(1))*x(2) + 1 - exp(k/m*X)
  y(2) = cos(sqrt(k*g/m)*x(2)) + sqrt(k/(m*g))*v0*sin(x(1))*sin(sqrt(k*g/m)*x(2)) - exp(k/m*Y)
endfunction

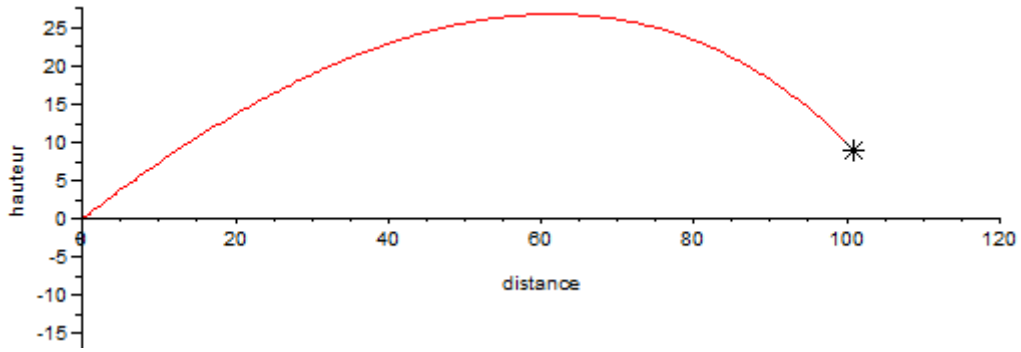
function trajectoire(x, F)
  drawlater()
  xtitle("", "distance", "hauteur")
  T = linspace(0, x(2), 100)
  xt = m/k*log(1 + k/m*v0*cos(x(1))*T)
  yt = m/k*log(cos(sqrt(k*g/m)*T) + sqrt(k/(m*g))*v0*sin(x(1))*sin(sqrt(k*g/m)*T))
  Xt = zeros(1, length(T)) ; Yt = zeros(1, length(T))
  for k = 1:length(T) , [X,Y] = F(T(k)) , Xt(k) = X , Yt(k) = Y , end
  a = min([xt,Xt]) ;   b = min([yt,Yt]) ; c = max([xt,Xt]) ; d =   max([yt,Yt])
  for k = 1:length(T)-1
    plot2d([Xt(k),Xt(k+1)], [Yt(k),Yt(k+1)], 2, frameflag=3, rect=[a,b,c,d], axesflag=1)
    plot2d([xt(k),xt(k+1)], [yt(k),yt(k+1)], 5, frameflag=3, rect=[a,b,c,d], axesflag=1)
    drawnow() ; xpause(100)
  end
  plot2d(xt($), yt($), -10, frameflag=3, rect=[a,b,c,d], axesflag=1) ;
  drawnow()
  xpause(2e6)
endfunction

m = 1 ; k = 0.005 ; g = 9.81 ; v0 = 40 ;
theta0 = %pi/4 ; [X0,Y0] = F1(0) ; t0 = norm([X0,Y0])/v0 ;
x1 = fsolve([theta0;t0], f1) ; theta1 = x1(1) , t1 = x1(2) , trajectoire(x1, F1)
xpause(1E6) ; clf()
theta0 = %pi/4 ; [X0,Y0] = F2(0) ; t0 = norm([X0,Y0])/v0 ;
x2 = fsolve([theta0;t0], f2) ; theta2 = x2(1) , t2 = x2(2) , trajectoire(x2, F2)
xpause(1E6) ; clf()
theta0 = %pi/4 ; [X0,Y0] = F3(0) ; t0 = norm([X0,Y0])/v0 ;
x3 = fsolve([theta0;t0], f3) ; theta3 = x3(1) , t3 = x3(2) , trajectoire(x3, F3)
xpause(1E6) ; clf()
xdel()
```

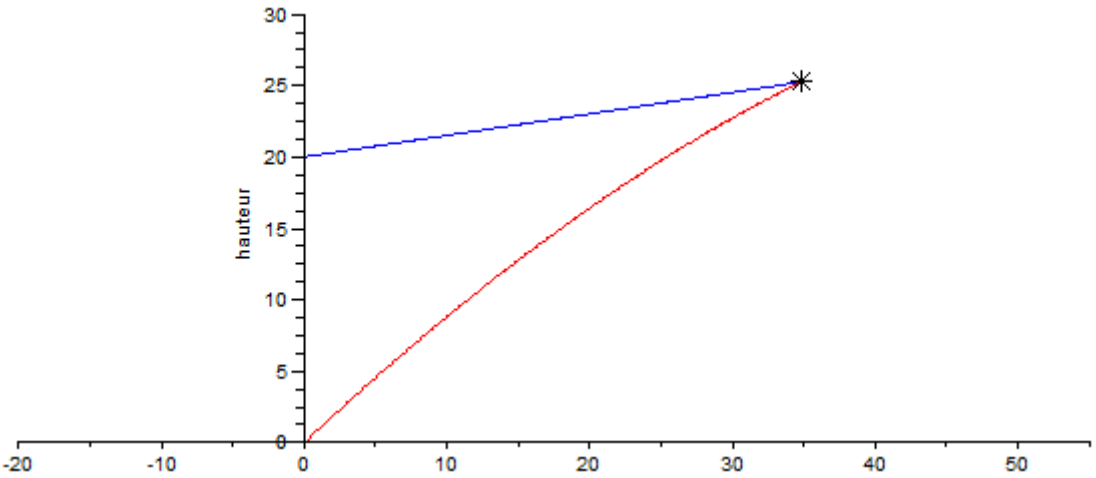
Mathématique & Informatique

TRAJECTOIRES

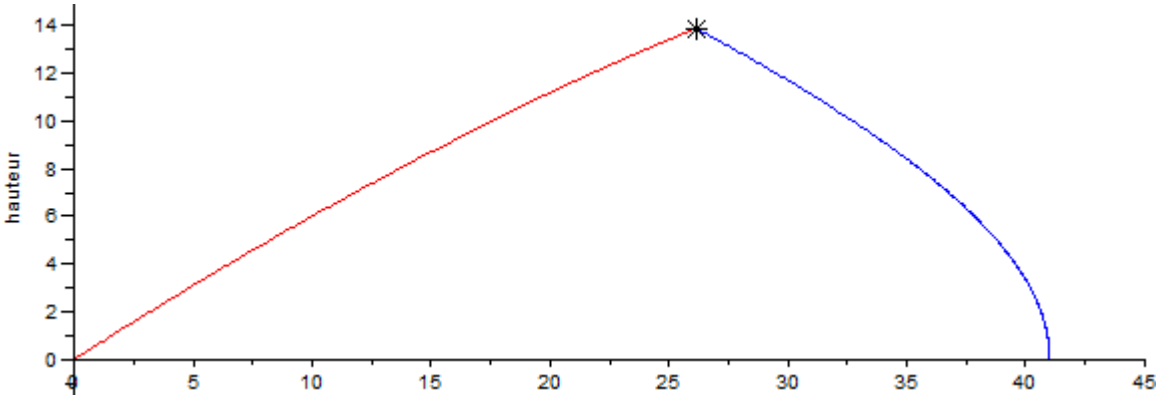
Cas 1 On obtient : theta1 = 0.6605400 rad (37.85 °) et t1 = 4.1599211 s.



Cas 2 On obtient : theta2 = 0.7538881 rad (43.19 °) et t2 = 1.3040743 s.

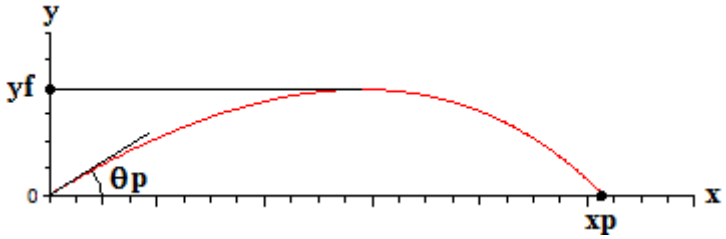


Cas 3 On obtient : theta3 = 0.5686619 rad (32.58 °) et t3 = 0.8279906 s.



Mathématique & Informatique

PORTÉE & FLÈCHE.



On recherche la valeur θ_p qui donne x_p maximale.
La portée x_p correspond à y = 0, c'est-à-dire à l'équation : $\cos[(kg/m)^{1/2}\, t] + (k/(mg))^{1/2}\, v_0 \sin\theta \sin[(kg/m)^{1/2}\, t] = 1$.
On a : $\sin\theta = \{1 - \cos[(kg/m)^{1/2}\, t]\} / \{(k/(mg))^{1/2}\, v_0 \sin[(kg/m)^{1/2}\, t]\} = \varphi(t)$.
D'où : $x(t) = (m/k) \operatorname{Log}[1 + (k/m)\, v_0\, t\, (1 - \varphi(t)^2)^{1/2}]$.
L'abscisse x_p est atteinte en un temps t_p vérifiant $x'(t_p) = 0$. L'équation $x'(t) = 0$ est équivalente à $1 - \varphi^2 - \varphi' \varphi\, t = 0$.
Elle se résoud avec *fsolve* en choisissant pour valeur initiale t_0 , le temps mis par l'objet pour retomber sur le sol avec un angle de lancement $\theta = \pi/4$. On obtient t_p , d'où $\varphi_p = \varphi(t_p)$, puis $\theta_p = \text{Arc sin}(\varphi_p)$ et $x_p = x(\theta_p, t_p)$.
La flèche y_f est l'altitude maximale de la trajectoire.
Elle correspond à $y'(t_f) = v_y(t_f) = 0$, d'où $t_f = (m/(kg))^{1/2}\{\pi/2 - \text{Arc tan}[(mg/k)^{1/2}/(v_0 \sin\theta_p)]\}$, et $y_f = y(\theta_p, t_f)$.

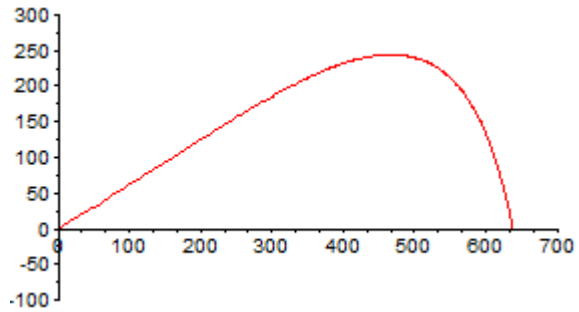
Application numérique : $m = 9.5\ 10^{-3}$ kg ; $k = 3.27\ 10^{-5}$ USI ; $g = 9.81$ m/s^2 ; $v_0 = 220$ m/s.
⇨ http://serge-etienne.pagesperso-orange.fr/textes/Balistique%20trajectoire%20et%20projectile.pdf
Programme :

```
function z = portee(t)
  a = 1 - cos(sqrt(k*g/m)*t) ; da = sqrt(k*g/m)*sin(sqrt(k*g/m)*t)
  b = sqrt(k/(m*g))*v0*sin(sqrt(k*g/m)*t) ; db = k/m*v0*cos(sqrt(k*g/m)*t)
  z = b^3 - a^2*b - a*(da*b - a*db)*t
endfunction

m = 9.5D-3 ; k = 3.27D-5 ; g = 9.81 ; v0 = 220 ;
if sqrt(k/(m*g))*v0*sin(%pi/4) < 1 then
  t0 = sqrt(m/(k*g))*asin((2*sqrt(k*m*g)*v0*sin(%pi/4))/(k*v0^2*sin(%pi/4)^2 + m*g)) ;
else
  t0 = sqrt(m/(k*g))*(%pi - asin((2*sqrt(k*m*g)*v0*sin(%pi/4))/(k*v0^2*sin(%pi/4)^2 + m*g))) ;
end
tp = fsolve(t0, portee)
phip = (1 - cos(sqrt(k*g/m)*tp))/(sqrt(k/(m*g))*v0*sin(sqrt(k*g/m)*tp)) ;
thetap = asin(phip)
xp = m/k*log(1 + k/m*v0*cos(thetap)*tp)
tf = sqrt(m/(k*g))*(%pi/2 - atan(sqrt(m*g/k)/(v0*sin(thetap)))) ;
yf = m/k*log(cos(sqrt(k*g/m)*tf) + sqrt(k/(m*g))*v0*sin(thetap)*sin(sqrt(k*g/m)*tf))

// Dessin
T = linspace(0, tp, 300) ;
x = m/k*log(1 + k/m*v0*cos(thetap)*T) ;
y = m/k*log(cos(sqrt(k*g/m)*T) + sqrt(k/(m*g))*v0*sin(thetap)*sin(sqrt(k*g/m)*T)) ;
a = gca() ; a.x_location = "origin" ; a.y_location = "origin" ;
plot2d(x, y, 5, frameflag = 4)
```

Résultats :



$t_p = 12.238963$ s
$\theta_p = 0.5318621$ rad (30.47 °)
$x_p = 637.94519$ m
$y_f = 244.09523$ m

En négligeant la force supplémentaire :
$t_p = \sqrt{2} v_0/g = 31.71$ s
$\theta_p = \pi/4$ rad (45°)
$x_p = v_0^2/g = 4933.74$ m
$y_f = v_0^2/(4g) = 1233.43$ m

Mathématique & Informatique

Transformation du système de 2 équations balistiques en une seule équation balistique

On a vu que l'application du Principe Fondamental de la Dynamique conduit à un système non linéaire de 2 équations à 2 inconnues, θ et t, qui peut être résolu directement avec *fsolve*.
Une autre méthode consiste à exprimer cosθ et sinθ en fonction de t.
On obtient immédiatement :

$$\cos\theta \;=\; \frac{e^{\frac{k}{m}X(t)} - 1}{\frac{k}{m}v_0\,t} \;=\; c(t)$$

$$\sin\theta \;=\; \frac{e^{\frac{k}{m}Y(t)} - \cos\left(\sqrt{\frac{kg}{m}}\,t\right)}{\sqrt{\frac{k}{mg}}\,v_0\,\sin\left(\sqrt{\frac{kg}{m}}\,t\right)} \;=\; s(t)$$

D'où :

$$\cos^2\theta + \sin^2\theta \;=\; c^2(t) + s^2(t) \;=\; 1 \quad\Rightarrow\quad c^2(t) + s^2(t) - 1 \;=\; 0 \quad\Rightarrow\quad \boldsymbol{f(t) \;=\; 0}$$

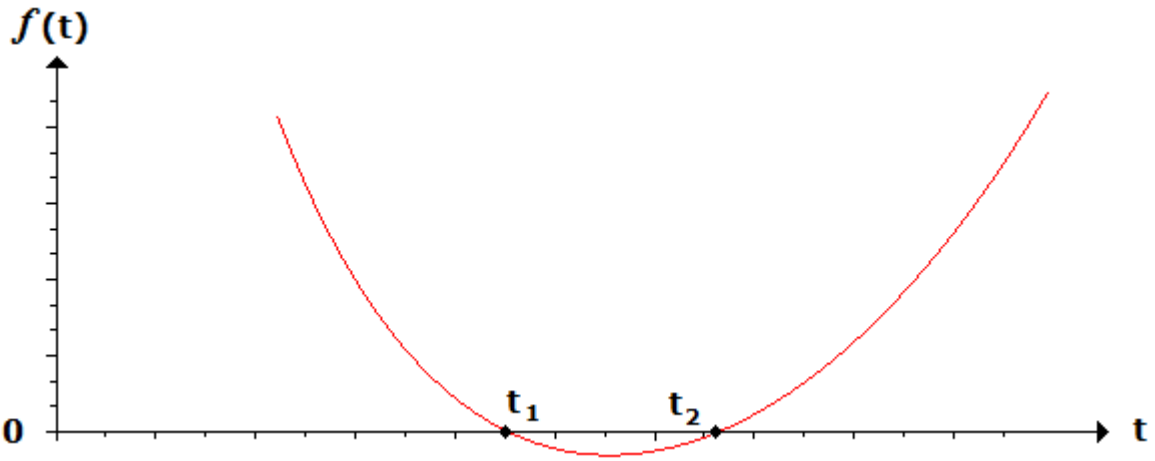
Donc, la résolution du système différentiel initial :

$$\begin{cases} \dfrac{d^2x}{dt^2} + \dfrac{k}{m}\left(\dfrac{dx}{dt}\right)^2 = 0 \\ \dfrac{d^2y}{dt^2} + \dfrac{k}{m}\left(\dfrac{dy}{dt}\right)^2 + g = 0 \end{cases}$$

conduit à appliquer *fsolve* à une unique équation de la variable t :

$$f(t) \;=\; \left[\frac{e^{\frac{k}{m}X(t)} - 1}{\frac{k}{m}v_0\,t}\right]^2 + \left[\frac{e^{\frac{k}{m}Y(t)} - \cos\left(\sqrt{\frac{kg}{m}}\,t\right)}{\sqrt{\frac{k}{mg}}\,v_0\,\sin\left(\sqrt{\frac{kg}{m}}\,t\right)}\right]^2 - 1 \;=\; 0$$

La courbe *f*(t) présente généralement l'allure suivante :



L'équation *f*(t) = 0 possède 2 solutions t_1 et t_2
La première solution t_1 est obtenue avec *fsolve* en partant de $t_0 = \varepsilon$ (ε petit mais non nul)
La deuxième solution t_2 est obtenue avec *fsolve* en partant de $t_0 = \alpha * t_1$ (α à déterminer de manière empirique)
Connaissant une valeur de t qui annule *f*, on déduit la valeur de θ correspondante :

$$\tan\theta = \frac{s(t)}{c(t)} \quad \Rightarrow \quad \theta = Arc\tan\left(\frac{s(t)}{c(t)}\right)$$

La fonction Scilab *atan* calcule θ dans l'intervalle [-π/2,+π/2].
L'angle θ est déterminé modulo π : on remplace θ par θ + π lorsque la fonction *atan* retourne une valeur négative.

Une cible fixe

Dans ce cas, le plus simple, les coordonnées de l'objet sont constantes. On a : $X(t) = Cte = d$ et $Y(t) = Cte = h$
L'équation à résoudre s'écrit :

$$\left[\frac{e^{\frac{kd}{m}} - 1}{\frac{k}{m} v_0 t}\right]^2 + \left[\frac{e^{\frac{kh}{m}} - \cos\left(\sqrt{\frac{kg}{m}}\, t\right)}{\sqrt{\frac{k}{mg}}\, v_0 \sin\left(\sqrt{\frac{kg}{m}}\, t\right)}\right]^2 - 1 = 0$$

La première solution t_1 de l'équation *f*(t) = 0, c'est-à-dire la plus petite valeur de t qui annule *f*, correspond à un tir "tendu" et la seconde solution $t_2 > t_1$ à un tir "lobé". Voir le **Cas 1** ci-après.

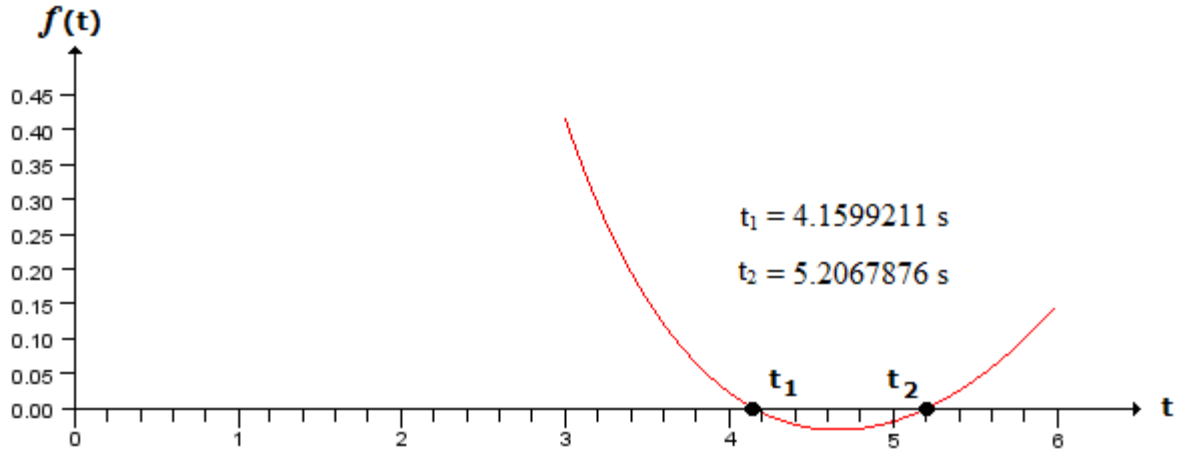
Lorsque la force supplémentaire est faible, la constante *k* est très petite.
Les équivalents : $e^u \sim 1 + u$, $\sin u \sim u$, $\cos u \sim 1 - \frac{1}{2}u^2$ quand $u \to 0$ permettent d'obtenir une équation simplifiée dont on peut exprimer les racines (équation bicarrée).
On trouve :

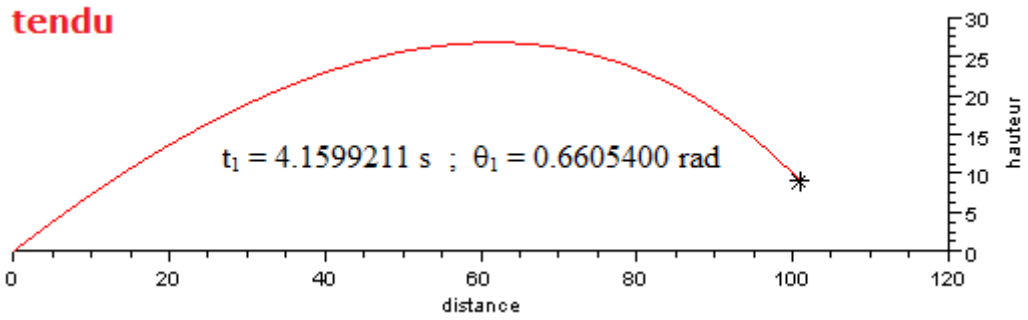
$$g^2t^4 + 4(gh - v_0^2)t^2 + 4(d^2 + h^2) = 0 \quad \Rightarrow \quad t = \frac{\sqrt{2(v_0^2 - gh) \pm 2\sqrt{v_0^4 - 2ghv_0^2 - g^2d^2}}}{g}$$

Application numérique

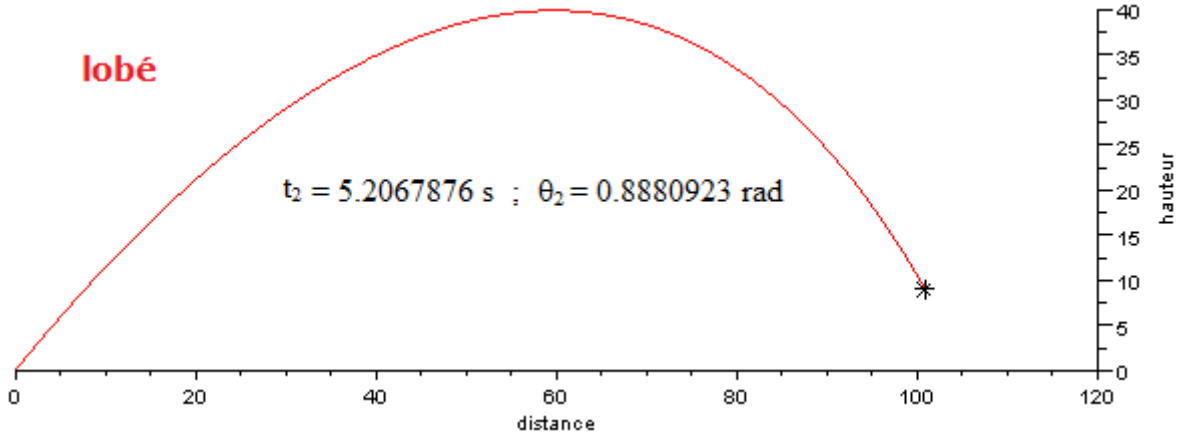
On reprend les données choisies pour **Cas 1**, **Cas 2**, **Cas 3** et on écrit les nouveaux programmes.
Les résultats sont identiques à ceux trouvés précédemment.

Dans le **Cas 1**, l'objet est au repos et les courbes sont les suivantes :



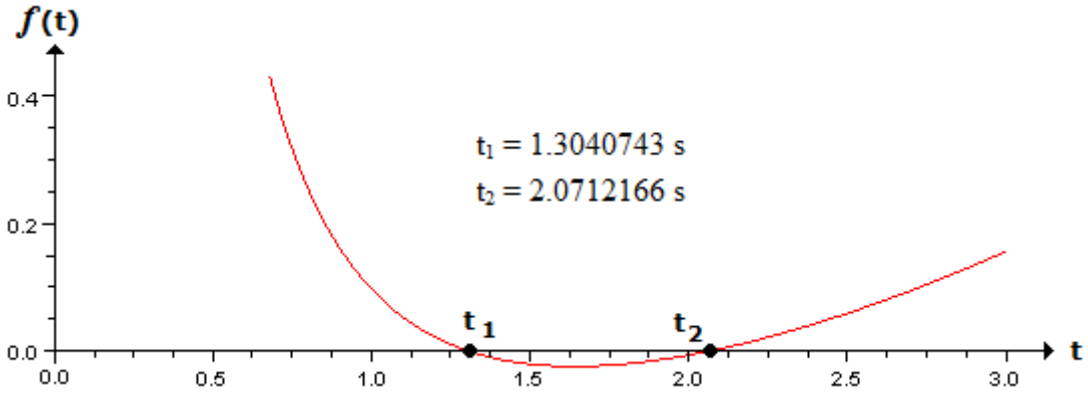


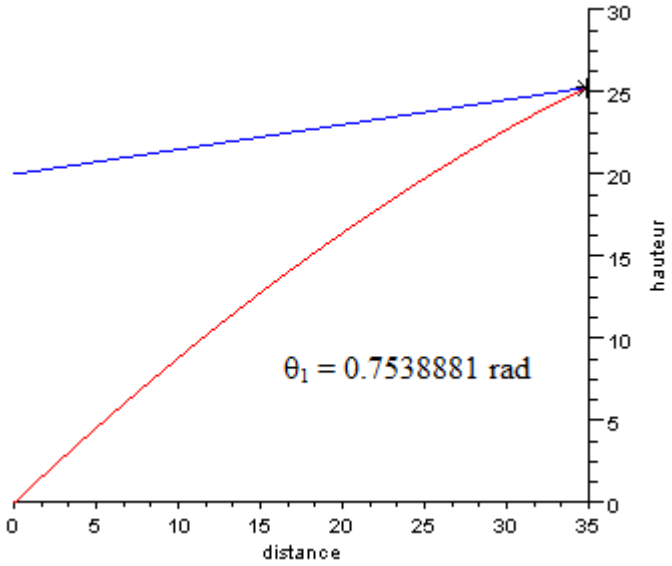
Le tir tendu est celui qui atteint le plus rapidement la cible.

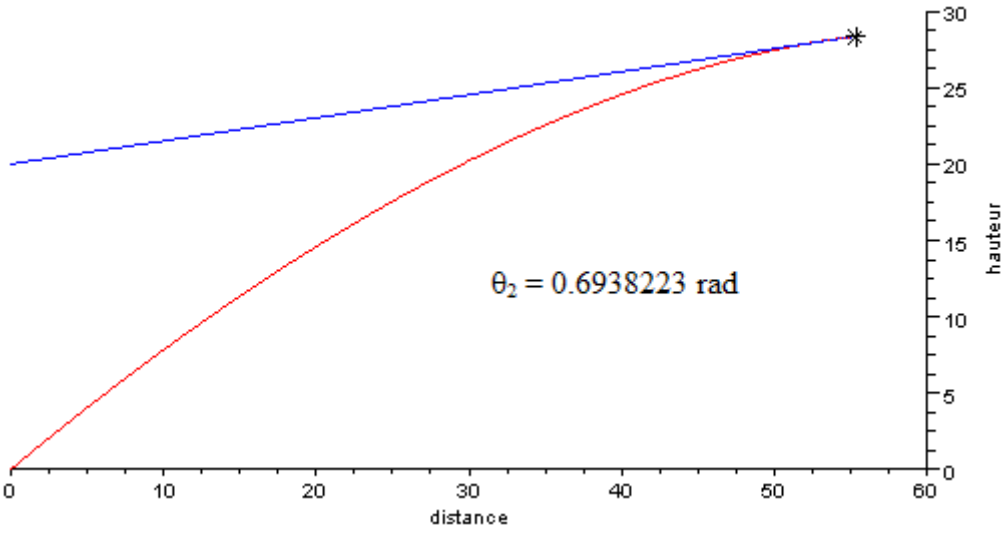


Le tir lobé permet de contourner un obstacle.

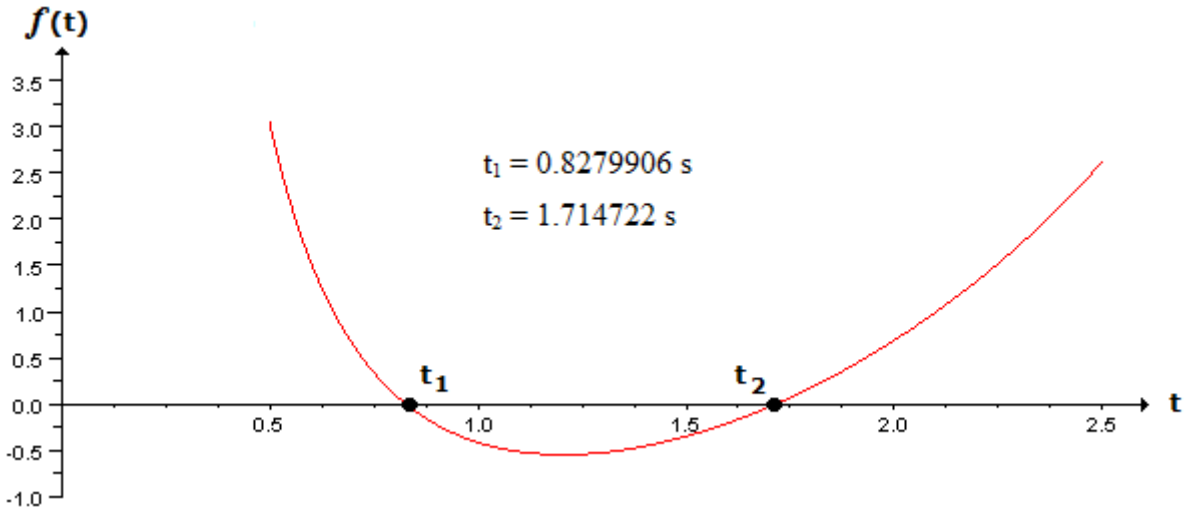
Dans le **Cas 2**, les 2 angles de tir sont proches et les courbes obtenues se ressemblent.
La trajectoire de l'objet est représentée en bleu et la trajectoire du projectile en rouge :

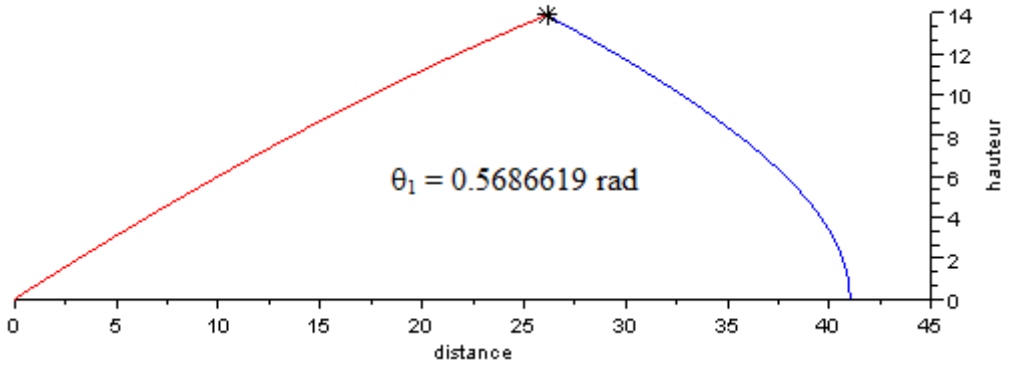


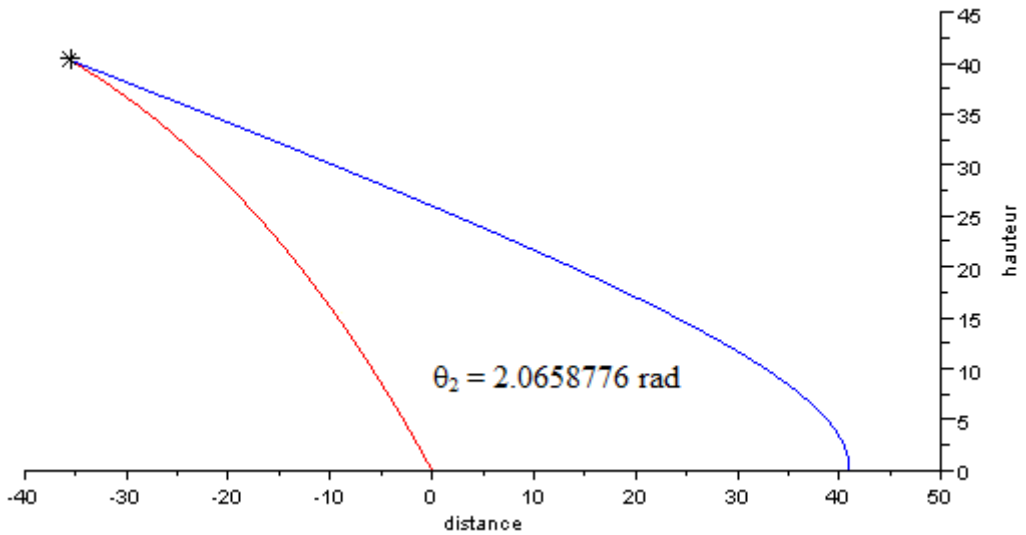




Dans le **Cas 3**, on peut atteindre l'objet mobile avant qu'il ait franchi la verticale du point de tir, c'est-à-dire avec un angle θ inférieur à 90° (1re trajectoire) ou après, c'est-à-dire avec un angle θ supérieur à 90°(2e trajectoire) :





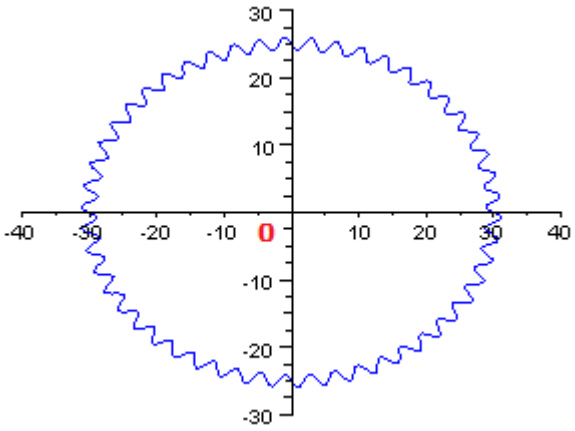


Mathématique & Informatique

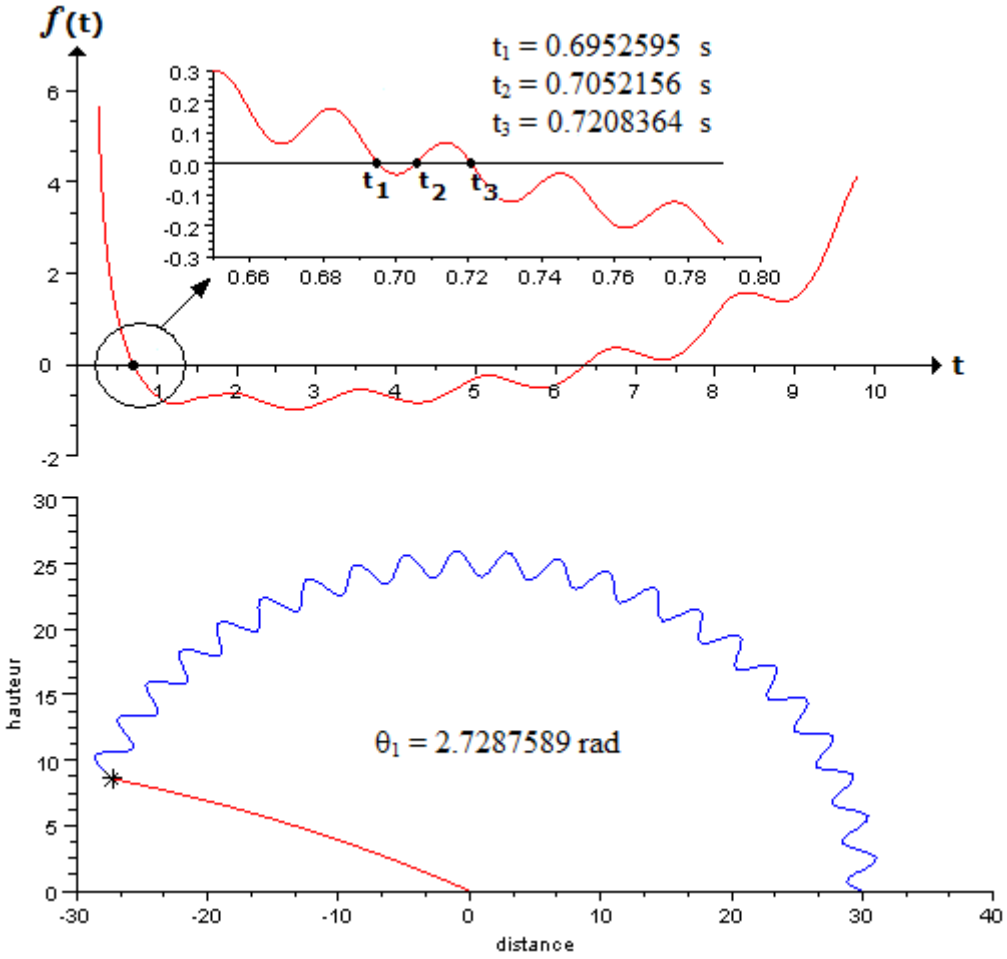
Dans le **Cas 4**, on considère que l'objet décrit un mouvement doublement périodique représenté par une trajectoire elliptique sinusoïdale de demi-grands axes a et b, de pulsation ω, de coefficient d'amplitude λ, avec n périodes autour du point fixe de tir situé en 0. Les équations horaires du mouvement sont :

$$X(t) = a[1 - \lambda\sin(n\omega t)]\cos(\omega t)$$
$$Y(t) = b[1 - \lambda\sin(n\omega t)]\sin(\omega t)$$

On recherche une valeur de θ qui permet au projectile d'entrer en collision avec l'objet.
Si les données sont quelconques, le problème n'admet pas toujours de solution.
Supposons par exemple que : a = 30, b = 25, ω = 4, λ = 0.04, n = 50. Le mobile parcourt la courbe ci-dessous :



La trajectoire que doit suivre le projectile pour atteindre l'objet se déduit de la résolution de l'équation $f(t) = 0$



Mathématique & Informatique

Programme

```
function [c,s,a] = trigonometrie(F, t)
  [X,Y] = F(t)
  c = (exp(k/m*X) - 1)/(k/m*v0*t)
  s = (exp(k/m*Y) - cos(sqrt(k*g/m)*t))/(sqrt(k/(m*g))*v0*sin(sqrt(k*g/m)*t))
  if c == 0 then, a = %pi/2, else, a = atan(s/c) ; if a < 0 then, a = a + %pi, end, end
endfunction


function trajectoire(F, t, theta)
  drawlater()
  T = linspace(0, t, 200)
  xt = m/k*log(1 + k/m*v0*cos(theta)*T)
  yt = m/k*log(cos(sqrt(k*g/m)*T) + sqrt(k/(m*g))*v0*sin(theta)*sin(sqrt(k*g/m)*T))
  Xt = zeros(1, length(T)) ; Yt = zeros(1, length(T))
  for k = 1:length(T), [X,Y] = F(T(k)), Xt(k) = X, Yt(k) = Y, end
  a = min([xt,Xt]) ;   b = min([yt,Yt]) ; c = max([xt,Xt]) ; d =   max([yt,Yt])
  plot2d(%inf, %inf, frameflag=3, rect=[a,b,c,d], axesflag=3)
  xtitle("", "distance", "hauteur")
  for k = 1:length(T)-1
    plot2d([Xt(k),Xt(k+1)], [Yt(k),Yt(k+1)], 2)
    plot2d([xt(k),xt(k+1)], [yt(k),yt(k+1)], 5)
    drawnow()
    xpause(1)
  end
  plot2d(xt($), yt($), -10)
  drawnow()
  xpause(3e6)
endfunction

function y = f1(t)
  [c,s,a] = trigonometrie(F1, t)
  y = c*c + s*s - 1
endfunction
function y = f2(t)
  [c,s,a] = trigonometrie(F2, t)
  y = c*c + s*s - 1
endfunction
function y = f3(t)
  [c,s,a] = trigonometrie(F3, t)
  y = c*c + s*s - 1
endfunction
function y = f4(t)
  [c,s,a] = trigonometrie(F4, t)
  y = c*c + s*s - 1
endfunction
```

```
function [X,Y] = F1(t)
   X = 101 ; Y = 9
endfunction
function [X,Y] = F2(t)
    X = 27*cos(0.15)*t ; Y = 27*sin(0.15)*t + 20
endfunction
function [X,Y] = F3(t)
    X = 41*(2 - cosh(t)) ; Y = 15*sinh(t)
endfunction
function [X,Y] = F4(t)
   X = 30*(1 - 0.04*sin(200*t))*cos(4*t)
   Y = 25*(1 - 0.04*sin(200*t))*sin(4*t)
endfunction

m = 1 ; k = 0.005 ; g = 9.81 ; v0 = 40 ;
//-------------------------------------------------------------
t1_1 = fsolve(%eps, f1)
[c1_1,s1_1,theta1_1] = trigonometrie(F1, t1_1)
trajectoire(F1, t1_1, theta1_1)
clf()
t2_1 = fsolve(2*t1_1, f1)
[c2_1,s2_1,theta2_1] = trigonometrie(F1, t2_1)
trajectoire(F1, t2_1, theta2_1)
clf()
//-------------------------------------------------------------
t1_2 = fsolve(%eps, f2)
[c1_2,s1_2,theta1_2] = trigonometrie(F2, t1_2)
trajectoire(F2, t1_2, theta1_2)
clf()
t2_2 = fsolve(2*t1_2, f2)
[c2_2,s2_2,theta2_2] = trigonometrie(F2, t2_2)
trajectoire(F2, t2_2, theta2_2)
clf()
//-------------------------------------------------------------
t1_3 = fsolve(%eps, f3)
[c1_3,s1_3,theta1_3] = trigonometrie(F3, t1_3)
trajectoire(F3, t1_3, theta1_3)
clf()
t2_3 = fsolve(2*t1_3, f3)
[c2_3,s2_3,theta2_3] = trigonometrie(F3, t2_3)
trajectoire(F3, t2_3, theta2_3)
clf()
//-------------------------------------------------------------
t1_4 = fsolve(%eps, f4)
[c1_4,s1_4,theta1_4] = trigonometrie(F4, t1_4)
trajectoire(F4, t1_4, theta1_4)
clf()
//-------------------------------------------------------------
xdel()
```

Mathématique & Informatique

A propos de la force supplémentaire

La force supplémentaire $\vec{s}$ ralentit le mouvement en phase ascendante mais l'accélère en phase descendante, elle dépend d'une constante k. La vitesse est représentée par $\vec{v}(v_x , v_y , v_z)$ et la force supplémentaire par $\vec{s}(-kv_x^2 , -kv_y^2 , -kv_z^2)$.
Le vecteur $\vec{s}$ n'est pas colinéaire à $\vec{v}$ et on a $s < k v^2$ où $v = (v_x^2 + v_y^2 + v_z^2)^{1/2}$ est le module de la vitesse.
La force $\vec{s}$ constitue néanmoins une bonne approximation de la force $\vec{r}$ de résistance de l'air lorsque la vitesse $\vec{v}$ a toutes ses coordonnées positives dans le repère Oxyz ($v_x \geq 0$, $v_y \geq 0$, $v_z \geq 0$).
L'expression de x, y, z en fonction du temps t se déduit du système différentiel régissant le mouvement du projectile.
Les problèmes sont résolus avec la fonction Scilab *fsolve*.

Sur la résistance de l'air et le coefficient de frottement

La résistance de l'air $\vec{r}$ est une force qui s'oppose toujours au mouvement, elle dépend d'un coefficient de frottement k.
On prend habituellement $\vec{r} = - k v \vec{v}$, c'est-à-dire $\vec{r}(-kvv_x , -kvv_y , -kvv_z)$, donc $r = k v^2$ en module.
Le vecteur $\vec{r}$ est colinéaire à $\vec{v}$ et demeure tangent à la trajectoire à chaque instant.
Le système différentiel obtenu n'a pas de solution analytique en x, y, z.
La résolution numérique s'effectue avec la fonction Scilab *ode*.

Le coefficient de frottement k est une constante qui s'exprime en unités du système international (USI).
Dimension de k : $r = k v^2 \Rightarrow k = r / v^2 \Rightarrow [k] = M L T^{-2} / (L T^{-1})^2 = M L^{-1}$, donc k est en kg/m.
Plus précisément, k est donné par la formule : $\mathbf{k = \frac{1}{2} C \rho S}$

- C = coefficient de traînée = constante sans dimension qui dépend de la forme du projectile

  par exemple :



  C = 0.4 **C = 0.04**

- ρ = masse volumique de l'air = 1.293 kg/m^3 dans les conditions normales (0°C et 1 atm)

- S = maître-couple = surface de la projection du volume sur un plan perpendiculaire au déplacement
  (par exemple : $S = \pi R^2$ si le projectile est une sphère de rayon R)

D'une manière générale, une force de frottement fluide s'exprime par la formule $\vec{f} = - k \varphi(v) \vec{v}/v$ où k est une constante (coefficient de frottement), $\varphi(v)$ une fonction du module de la vitesse, et $\vec{v}/v$ le vecteur unitaire donnant la direction de la vitesse. Pour les vitesses faibles (moins de 1 m/s) on prend $\varphi(v) = v$, pour les vitesses moyennes $\varphi(v) = v^2$ (jusqu'à environ 800 km/h), pour les vitesses élevées $\varphi(v) = v^4$.
On utilise aussi des expressions qui différencient les coefficients suivant 3 axes, on a alors :
$\vec{f} = \vec{f}_x + \vec{f}_y + \vec{f}_z$ avec $\vec{f}_x = k_x \varphi(v) \vec{X}$, $\vec{f}_y = k_y \varphi(v) \vec{Y}$, $\vec{f}_z = k_z \varphi(v) \vec{Z}$ dans un repère orthogonal $(\vec{X} , \vec{Y} , \vec{Z})$
($\vec{f}_x$ est la force de traînée, $\vec{f}_y$ est la force de dérive, $\vec{f}_z$ est la force de portance ⇨ voir Aérodynamisme)

Vers un modèle plus précis

En toute rigueur on ne peut pas considérer que le repère terrestre qui sert à étudier le mouvement d'un corps est fixe.
Il faut tenir compte de la rotation de la Terre et introduire 2 forces annexes "fictives" afin de pouvoir appliquer le PFD dans un repère mobile : la force d'entraînement $\vec{f}_e$ (force centrifuge si le corps est lié à la Terre, ce qui n'est pas le cas ici puisqu'il se déplace librement dans l'espace) et la force complémentaire $\vec{f}_c$ (force de **Coriolis** si le corps possède une vitesse non nulle). En outre, il est nécessaire d'ajouter d'autres forces "réelles" qui s'exercent sur la masse m, comme la force du vent $\vec{f}_v$, la force de **Magnus** $\vec{f}_m$, la poussée d'**Archimède** $\vec{f}_a$, ...
Finalement, le Principe Fondamental de la Dynamique s'écrit :

$$\vec{p} + \vec{r} + \vec{f_e} + \vec{f_c} + \vec{f_v} + \vec{f_m} + \vec{f_a} + \ldots = m\vec{a}$$

Le système différentiel résultant n'admet pas de solution explicite, et il doit être résolu numériquement.
Pour un corps de masse m et de volume V se déplaçant à la vitesse v, on a : $f_e < m \Omega^2 R$, $f_c < 2 m \Omega v$, $f_a = \rho V g$
avec $\Omega = 7.29\ 10^{-5}$ rad/s la vitesse angulaire de rotation de la Terre, $R = 6.4\ 10^6$ m le rayon terrestre, $\rho = 1.293$ kg/m^3 la masse volumique de l'air, et $g = 9.81$ m/s^2 l'accélération de la pesanteur.
En pratique f_e , f_c , f_v , f_m , f_a sont faibles par rapport à la force poids $p = mg$. Par contre, si les distances ou les durées mises en jeu sont grandes, alors le déplacement du repère terrestre ne peut pas être négligé.

RÉSOLUTION NUMÉRIQUE DES SYSTÈMES DIFFÉRENTIELS (fonction *ode*)

On recherche $y(t) \in \mathbb{R}^n$ tel que : $y' = f(t, y)$
La variable $t \in \mathbb{R}$ désigne habituellement le temps. La fonction f : $\mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ est connue.
Afin de déterminer entièrement la solution y, on ajoute la condition initiale : $y(t_0) = y_0$ (valeur de y à $t = t_0$)

Le problème se présente sous la forme d'un système différentiel :

$$\begin{cases} y_1' = f_1(t, y_1, y_2, \dots, y_n) \\ y_2' = f_2(t, y_1, y_2, \dots, y_n) \\ \vdots \\ \vdots \\ y_n' = f_n(t, y_1, y_2, \dots, y_n) \end{cases} \qquad \text{avec} \quad y_1(t_0) = c_1,\ y_2(t_0) = c_2,\ \dots,\ y_n(t_0) = c_n$$

Pour calculer les valeurs prises par les fonctions $y_1, y_2, \dots, y_n$ entre les instants t_0 et t_1, on écrit le programme :

```
function yp = f(t, y)
    yp(1) = f1 (t, y(1), y(2), .... , y(n))
    yp(2) = f2 (t, y(1), y(2), .... , y(n))
    /
    /
    yp(n) = fn (t, y(1), y(2), .... , y(n))
endfunction
y0 = [c1 ; c2 ; ... ; cn] ;
t = linspace(t0, t1, m) ;
y = ode(y0, t0, t, f) ;
```

On obtient un vecteur ligne t de dimension m et une matrice y de dimension (n, m) telle que $y(i, j)$ est une approximation de la valeur prise par la fonction y_i à l'instant $t(j) = t_0 + (j-1)\left(\frac{t_1 - t_0}{m-1}\right)$, $1 \le j \le m$

Cas particulier

Une équation différentielle du second ordre $y'' = f(t, y, y')$ est équivalente à un système différentiel.
Il suffit de poser $y_1 = y$ et $y_2 = y'$, pour obtenir :

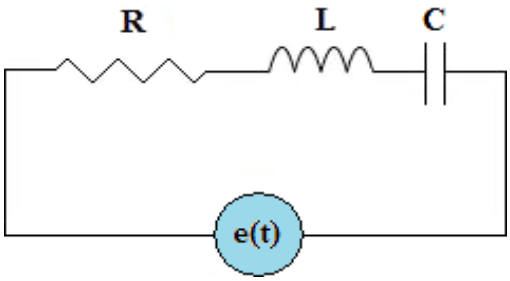
$$\begin{cases} y'_1 = y_2 \\ y'_2 = f(t, y_1, y_2) \end{cases}$$

Plus généralement, en posant $y_1 = y$, $y_2 = y'$, ... , $y_n = y^{(n-1)}$, l'équation différentielle $y^{(n)} = f(t, y, y', \dots, y^{(n-1)})$ s'écrit :

$$\begin{cases} y'_1 = y_2 \\ y'_2 = y_3 \\ \vdots \\ \vdots \\ y'_{n-1} = y_n \\ y'_n = f(t, y_1, y_2, \dots, y_n) \end{cases}$$

A la fin du programme, les valeurs de $y(t)$, $t_0 \le t \le t_1$, sont contenues dans le vecteur ligne $y(1, :)$

Exemple 1 : un problème d'électricité



Un circuit R, L, C série est alimenté par un générateur de tension triangulaire symétrique $e(t)$, d'amplitude A et de période T.
On prend :
R = 1 kΩ , L = 10 mH , C = 10 nF
A = 10 V , T = 100 ms.

Tracer les courbes donnant les différentes tensions aux bornes des éléments du circuit : $e(t)$, $u_R(t)$, $u_L(t)$, $u_C(t)$?

Soit $i(t)$ l'intensité du courant qui traverse le circuit à l'instant t.
L'équation différentielle vérifiée par i s'obtient en additionnant les tensions le long de la maille :

$$u_R + u_L + u_C = e \quad \Leftrightarrow \quad Ri + L\frac{di}{dt} + \frac{1}{C}\int idt = e$$

Dérivons pour faire disparaître l'intégrale :

$$L\frac{d^2i}{dt^2} + R\frac{di}{dt} + \frac{1}{C}i = \frac{de}{dt}$$

C'est-à-dire : $i'' = \left(\frac{1}{L}\right)e' - \left(\frac{1}{LC}\right)i - \left(\frac{R}{L}\right)i'$

Posons $i_1 = i$ et $i_2 = i'$, le système différentiel équivalent s'écrit :

$$\begin{cases} i_1' = i_2 \\ i_2' = \left(\frac{1}{L}\right)e' - \left(\frac{1}{LC}\right)i_1 - \left(\frac{R}{L}\right)i_2 \end{cases}$$

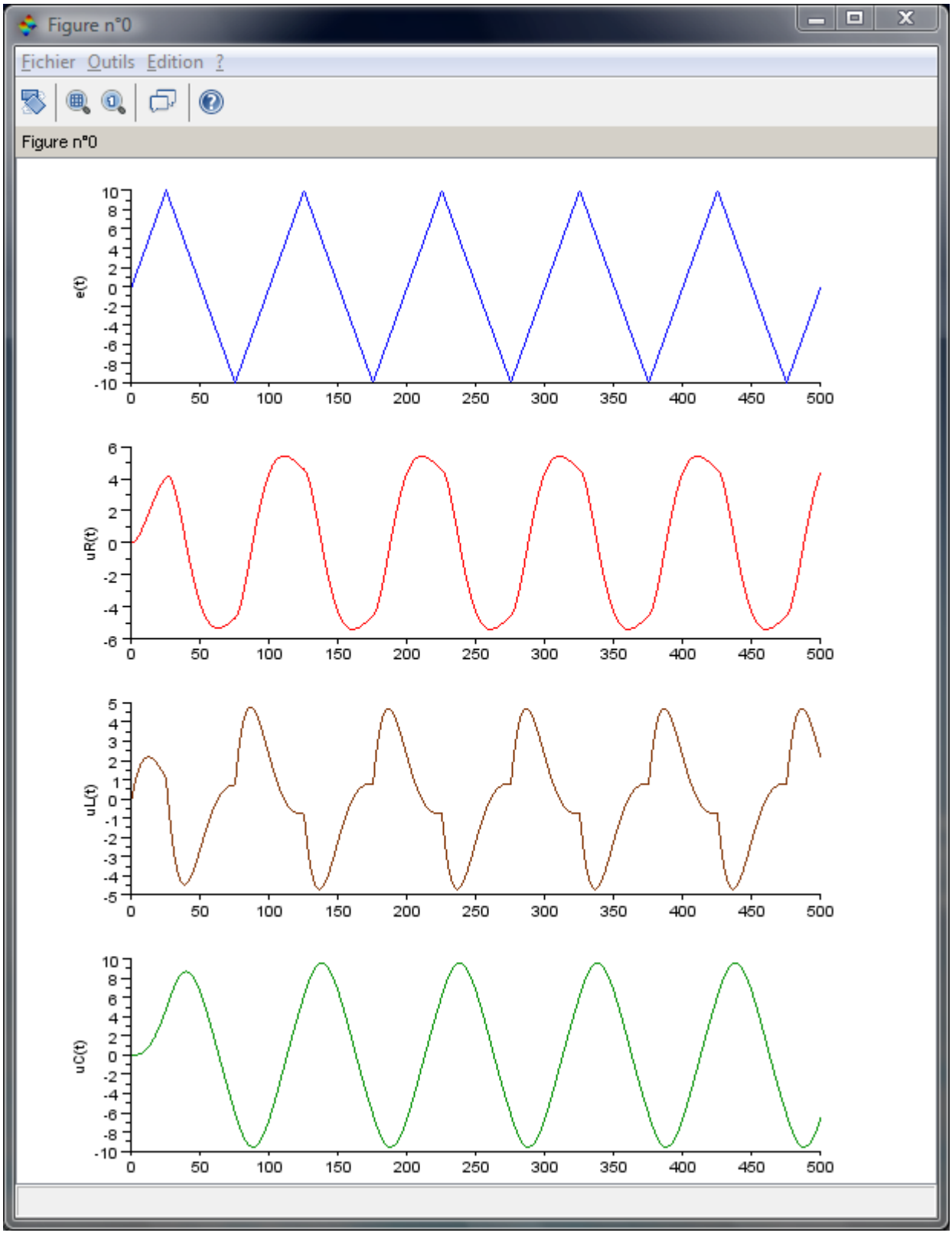
En choisissant $e(0) = 0$, le courant i et la tension $L\frac{di}{dt}$ aux bornes de la self sont nuls à l'instant initial.
Donc, à $t = 0$: $i = 0 \Rightarrow i_1(0) = 0$ et $L\frac{di}{dt} = 0 \Rightarrow i_2(0) = 0$
Le programme Scilab est le suivant :

```
R = 1D3 ; L = 10D-3 ; C = 10D-9 ; A = 10 ; T = 100D-6 ;
t0 = 0 ; t1 = 500D-6 ; n = 1000 ; t = linspace(t0, t1, n) ;
tp = t - floor(t/T)*T ;
e = bool2s(tp < T/4).*(4*A/T*tp) + bool2s(tp >= T/4 & tp < 3*T/4).*(-4*A/T*tp + 2*A) + bool2s(tp >= 3*T/4).*(4*A/T*tp - 4*A) ;

function ip = f(t, i)
  tp = t - floor(t/T)*T
  if (tp < T/4)|(tp >= 3*T/4) then , ep = 4*A/T , else , ep = -4*A/T , end
  ip(1) = i(2)
  ip(2) = ep/L - i(1)/(L*C) - R*i(2)/L
endfunction

i0 = [0;0] ; i = ode(i0, t0, t, f) ;
uR = R*i(1, :) ; uL = L*i(2, :) ; uC = e - uR - uL ;
subplot(4,1,1) ; plot2d(t*1D6, e, style = 2) ; xtitle("", "", "e(t)")
subplot(4,1,2) ; plot2d(t*1D6, uR, style = 5) ; xtitle("", "", "uR(t)")
subplot(4,1,3) ; plot2d(t*1D6, uL, style = 25) ; xtitle("", "", "uL(t)")
subplot(4,1,4) ; plot2d(t*1D6, uC, style = 13) ; xtitle("", "", "uC(t)")
```

Courbes obtenues sur 5 périodes (abscisse = temps en microsecondes , ordonnée = tension en volts) :

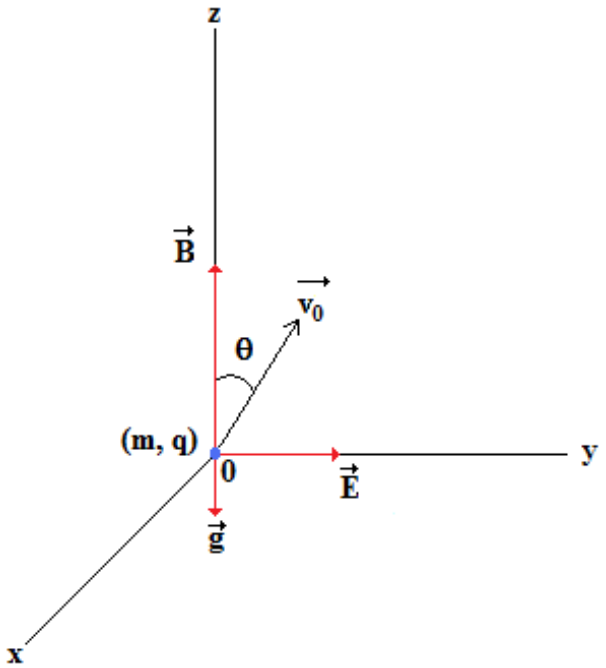


On peut vérifier expérimentalement ce résultat avec un oscilloscope.

Exemple 2 : un problème d'électromagnétisme

Une particule de masse m et de charge q, animée d'une vitesse initiale $\overrightarrow{v_0}$, est soumise au champ électromagnétique :

$$\left(\overrightarrow{E}, \overrightarrow{B}\right) = \left(\overrightarrow{E_0}, \overrightarrow{B_0} cos\omega t\right)$$



$m = 0.9\ 10^{-30}$ kg
$q = -1.6\ 10^{-19}$ C
$g = 9.81$ m/s^2
$v_0 = 10^7$ m/s
$E_0 = 10^3$ V/m
$B_0 = 10^{-3}$ T
$\omega = 10^7$ rad/s

Considérons les 6 cas suivants :

	E	**B**	**v_0**	**θ**
Cas 1	Oui	Non	Oui	π/2
Cas 2	Oui	Non	Oui	0
Cas 3	Non	Oui	Oui	π/2
Cas 4	Oui	Oui	Non	0
Cas 5	Non	Oui	Oui	π/3
Cas 6	Oui	Oui	Oui	π/3

Dans chaque cas, quelle est la trajectoire de la particule entre les instants $t_0 = 0$ et $t_1 = 10^{-7}$ s ?

Mathématique & Informatique

La particule est soumise à 3 champs : le champ de la pesanteur $\vec{g}$, le champ électrique $\vec{E}$ et le champ magnétique $\vec{B}$.
Les forces qui s'exercent sur elle sont :

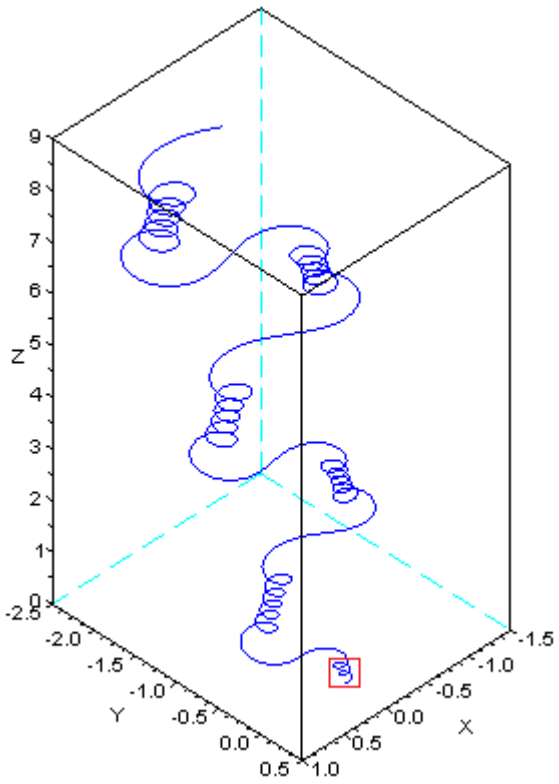
1) la force poids $\vec{f}_p = m\vec{g}$
2) la force électromagnétique $\vec{f}_e = q\,(\vec{E} + \vec{v} \wedge \vec{B})$ (force de **Lorentz**)

Le Principe Fondamental de la Dynamique s'écrit : $\vec{f}_p + \vec{f}_e = m\vec{a}$
Coordonnées cartésiennes des différents vecteurs : $\vec{g}(0,0,-g)$, $\vec{E}(0,E,0)$, $\vec{B}(0,0,B)$, $\vec{v}(\dot{x}\,,\dot{y}\,,\dot{z})$, $\vec{a}(\ddot{x}\,,\ddot{y}\,,\ddot{z})$
On déduit le système différentiel :

$$\begin{cases} \ddot{x} - \dfrac{qB}{m}\dot{y} = 0 \\ \ddot{y} + \dfrac{qB}{m}\dot{x} - \dfrac{qE}{m} = 0 \\ \ddot{z} + g = 0 \end{cases}$$

avec les conditions initiales à t = 0 : $x_0 = 0$, $y_0 = 0$, $z_0 = 0$, $\dot{x}_0 = 0$, $\dot{y}_0 = v_0 \sin\theta$, $\dot{z}_0 = v_0 \cos\theta$
Posons : $y_1 = x(t)$, $y_2 = y(t)$, $y_3 = z(t)$, $y_4 = \dot{x}(t)$, $y_5 = \dot{y}(t)$, $y_6 = \dot{z}(t)$
On obtient le système différentiel équivalent :

$$\begin{cases} \dot{y}_1 = y_4 \\ \dot{y}_2 = y_5 \\ \dot{y}_3 = y_6 \\ \dot{y}_4 = \left(\dfrac{qB_0}{m}\cos\omega t\right) y_5 \\ \dot{y}_5 = -\left(\dfrac{qB_0}{m}\cos\omega t\right) y_4 + \dfrac{qE_0}{m} \\ \dot{y}_6 = -g \end{cases}$$



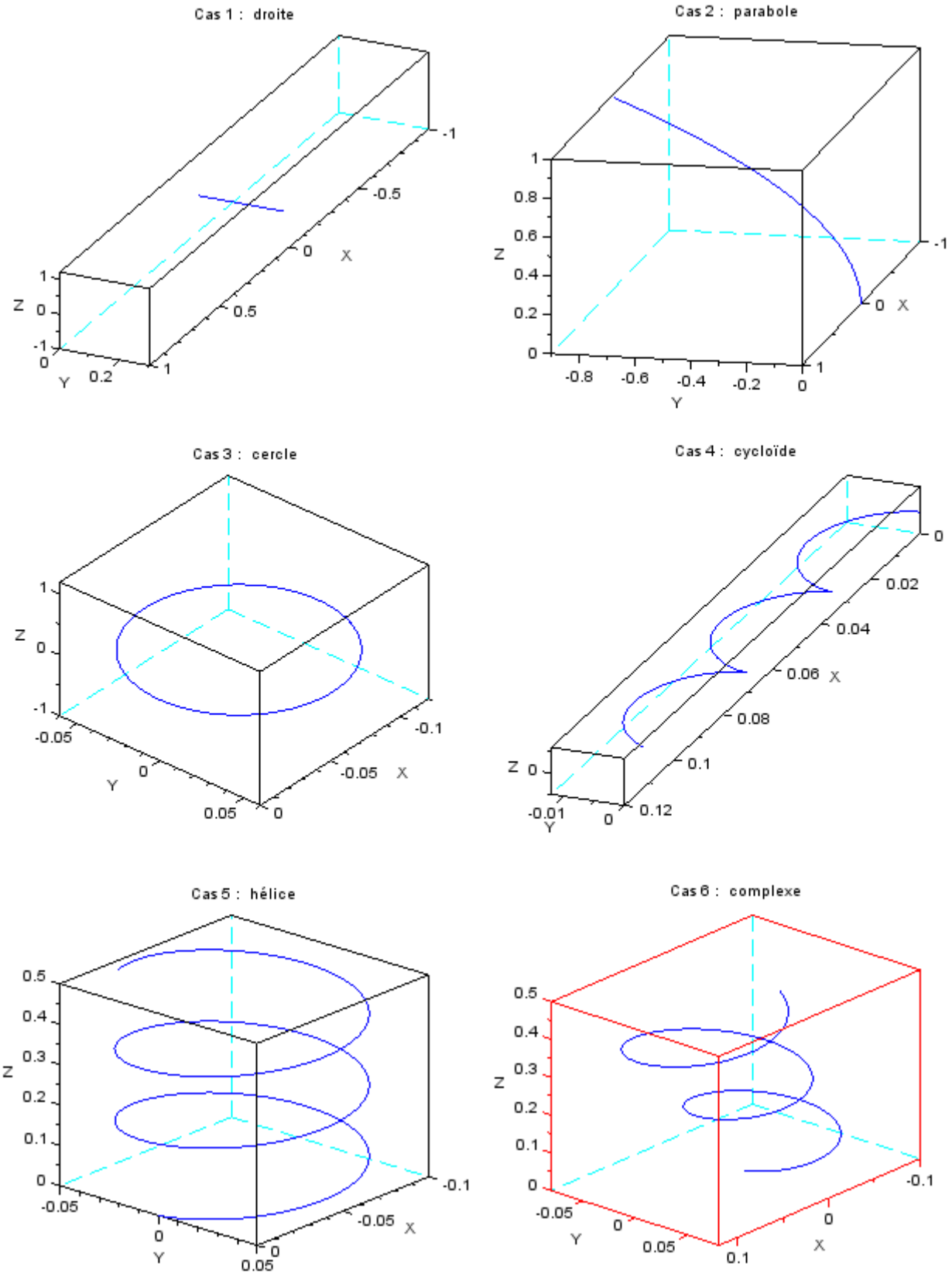
D'où le programme Scilab :

```
function trajectoire(m, q, g, E0, B0, v0, theta, omega, t0, t1, a1, a2)
  function yp = f(t, y)
    yp(1) = y(4)
    yp(2) = y(5)
    yp(3) = y(6)
    yp(4) = q*B0*cos(omega*t)/m*y(5)
    yp(5) = -q*B0*cos(omega*t)/m*y(4) + q*E0/m
    yp(6) = -g
  endfunction
  y0 = [0;0;0;0;v0*sin(theta);v0*cos(theta)]
  t = linspace(t0, t1, 1000)
  y = ode(y0, t0, t, f)
  z = max(y(3,:),ones(1,size(y,2))*%eps)
  param3d1(y(1,:)', y(2,:)', list(z', 2), a1, a2)
endfunction
```

```
subplot(3,2,1) ; xtitle("Cas 1 :  droite","","") ; trajectoire(0.9D-30, -1.6D-19, 9.81, 1D3, 0, 1D7, %pi/2, 0, 0, 1D-7, 25, 15)
subplot(3,2,2) ; xtitle("Cas 2 :  parabole","","") ; trajectoire(0.9D-30, -1.6D-19, 9.81, 1D3, 0, 1D7, 0, 0, 0, 1D-7, 12, 72)
subplot(3,2,3) ; xtitle("Cas 3 :  cercle","","") ; trajectoire(0.9D-30, -1.6D-19, 9.81, 0, 1D-3, 1D7, %pi/2, 0, 0, 1D-7, 40, 3)
subplot(3,2,4) ; xtitle("Cas 4 :  cycloïde","","") ; trajectoire(0.9D-30, -1.6D-19, 9.81, 1D3, 1D-3, 0, 0, 0, 0, 1D-7, 22, 0.5)
subplot(3,2,5) ; xtitle("Cas 5 :  hélice","","") ; trajectoire(0.9D-30, -1.6D-19, 9.81, 0, 1D-3, 1D7, %pi/3, 0, 0, 1D-7, 41, 24)
subplot(3,2,6) ; xtitle("Cas 6 :  complexe","","") ; trajectoire(0.9D-30, -1.6D-19, 9.81, 1D3, 1D-3, 1D7, %pi/3, 1D7, 0, 1D-7, 41, 36)
xset("color", 5)
```

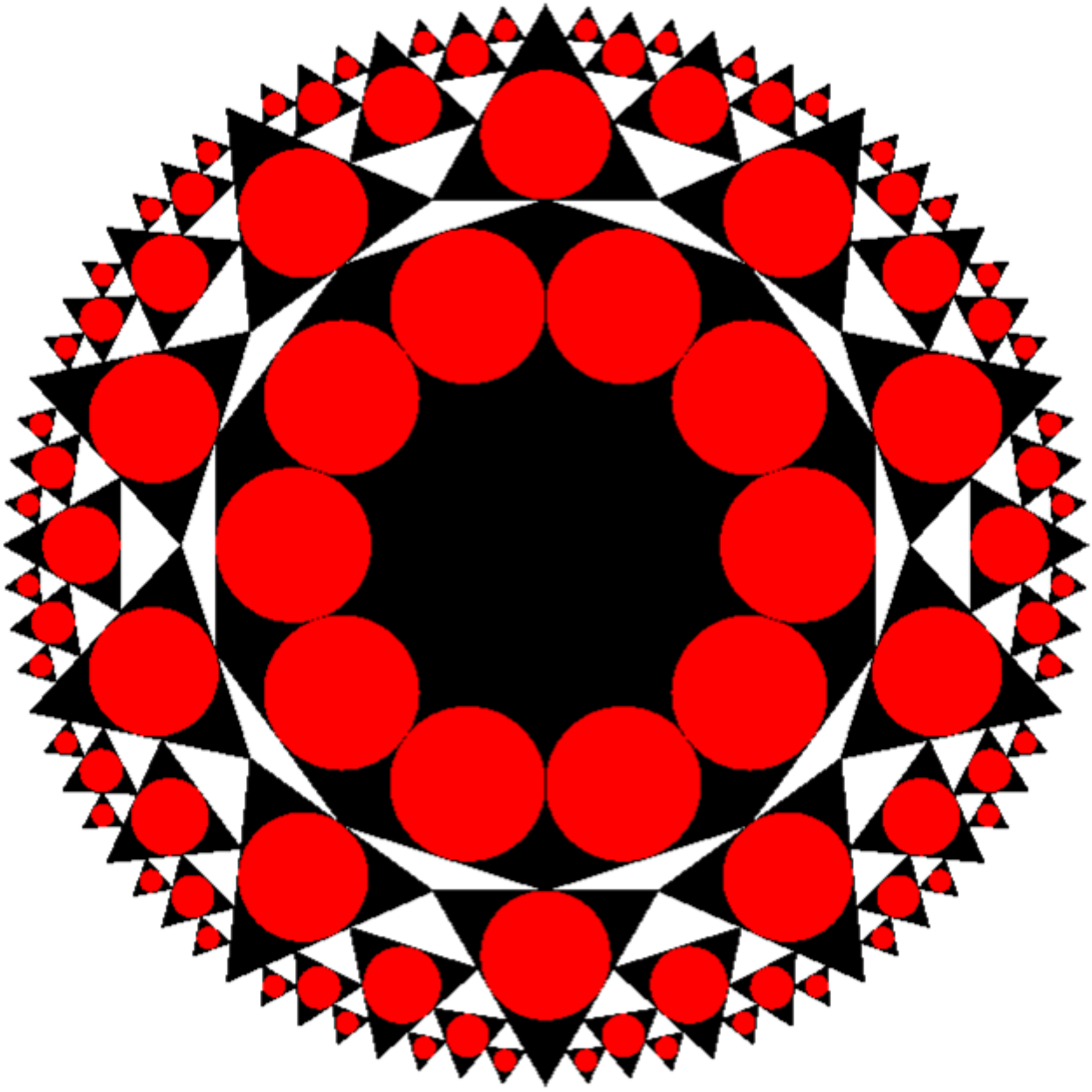
Mathématique & Informatique

Trajectoires obtenues :



Remarque : Dans le cas 6, la trajectoire de la particule encadrée en rouge n'est qu'une partie de son mouvement général représenté page précédente.

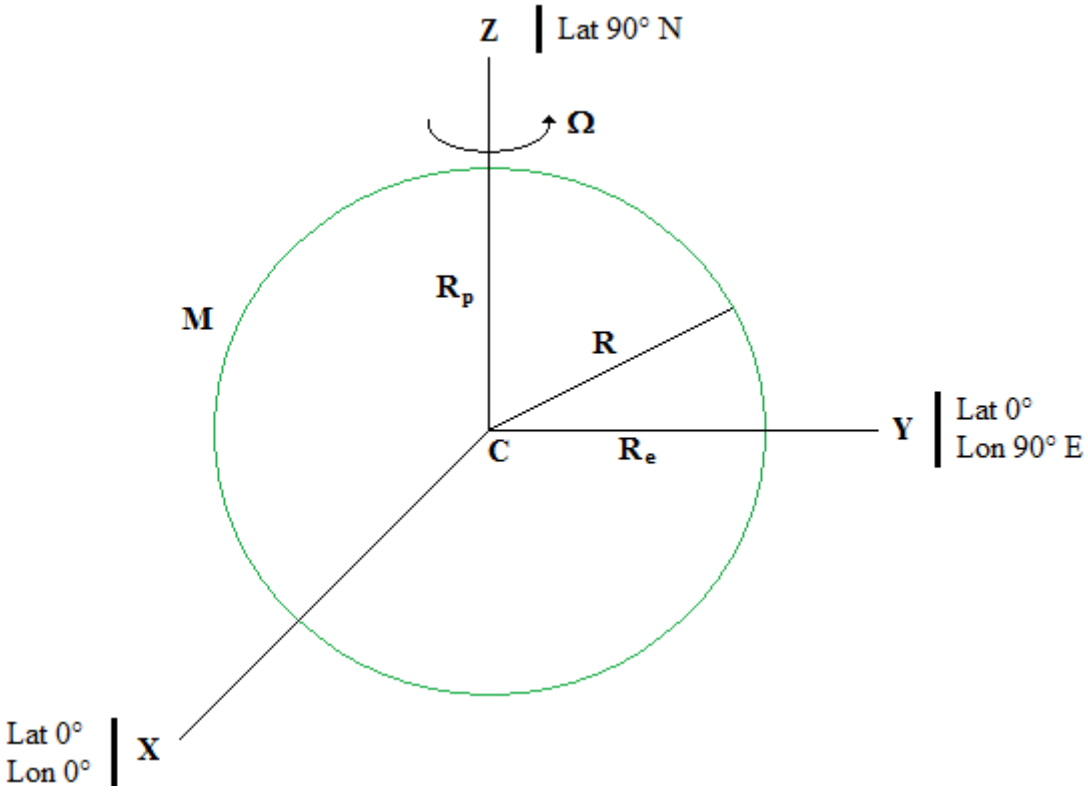
Mathématique & Informatique



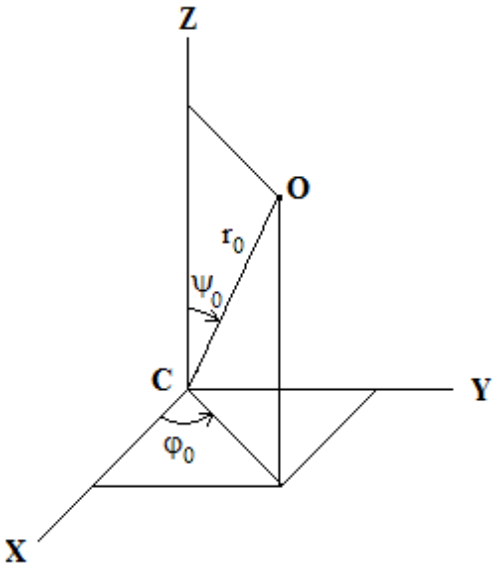
Mathématique & Informatique

LA FORCE DE CORIOLIS (Gaspard Coriolis, Paris 1792 - id. 1843)

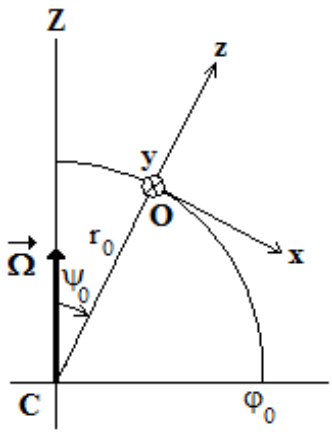
On considère que la Terre est un ellipsoïde de masse M, de demi-grands axes R_e (rayon équatorial) et R_p (rayon polaire), qui tourne à la vitesse angulaire constante Ω par rapport à un repère géocentrique absolu (galiléen) dont les axes sont dirigés vers des étoiles lointaines supposées "fixes".
Prenons un premier repère relatif CXYZ – lié à la Terre, c'est-à-dire tournant avec elle – où C est le centre de la Terre, Z l'axe passant par le pôle Nord, X l'axe passant par le point de latitude 0° / longitude 0°, Y l'axe passant par le point de latitude 0° / longitude 90° Est.



Prenons un deuxième repère relatif Oxyz – lié au précédent, c'est-à-dire effectuant le même mouvement de rotation – où O est un point de rayon r_0, de longitude φ_0, et de colatitude ψ_0, z l'axe du vecteur $\overrightarrow{CO}$, x l'axe orthogonal à z dans le plan méridien de longitude φ_0, y l'axe tel que $\vec{x} \wedge \vec{y} = \vec{z}$.



$r_0 \in [0, +\infty[$
$\varphi_0 \in [0, 2\pi[$
$\psi_0 \in [0, \pi]$



Mathématique & Informatique

Un projectile de masse m est lancé du point P_0 (x_0 , y_0 , z_0) avec la vitesse $\overrightarrow{v_0}$ ($\dot{x}_0$, $\dot{y}_0$, $\dot{z}_0$).
Il est soumis à 3 forces :

1) La force de gravitation : $\overrightarrow{f_g} = -G\frac{Mm}{r^3}\vec{r}$ où r est la distance du centre de la Terre au projectile
2) La force de résistance de l'air : $\overrightarrow{f_r} = -kv\vec{v}$ où v est la vitesse du projectile
3) La force de **Coriolis** : $\overrightarrow{f_c} = 2m\vec{v} \wedge \vec{\Omega}$ où Ω est la vitesse angulaire de rotation de la Terre

A partir du Principe Fondamental de la Dynamique $\vec{f}_g + \vec{f}_r + \vec{f}_c = m\vec{a}$, on établit que les coordonnées x(t), y(t), z(t) du projectile dans le repère Oxyz vérifient le système différentiel :

$$\begin{cases} \ddot{x} = -GM\dfrac{x}{[x^2+y^2+(z+r_0)^2]^{3/2}} - \dfrac{k}{m}\dot{x}\sqrt{\dot{x}^2+\dot{y}^2+\dot{z}^2} + 2\Omega\dot{y}\cos\psi_0 \\ \ddot{y} = -GM\dfrac{y}{[x^2+y^2+(z+r_0)^2]^{3/2}} - \dfrac{k}{m}\dot{y}\sqrt{\dot{x}^2+\dot{y}^2+\dot{z}^2} - 2\Omega(\dot{x}\cos\psi_0 + \dot{z}\sin\psi_0) \\ \ddot{z} = -GM\dfrac{z+r_0}{[x^2+y^2+(z+r_0)^2]^{3/2}} - \dfrac{k}{m}\dot{z}\sqrt{\dot{x}^2+\dot{y}^2+\dot{z}^2} + 2\Omega\dot{y}\sin\psi_0 \end{cases}$$

Deux rotations, une première d'angle φ_0 et une seconde d'angle ψ_0 , puis une translation de vecteur $\overrightarrow{CO}$, donnent les coordonnées X(t), Y(t), Z(t) du projectile dans le repère CXYZ :

$$\begin{cases} X = r_0\sin\psi_0\cos\varphi_0 + x\cos\psi_0\cos\varphi_0 - y\sin\varphi_0 + z\sin\psi_0\cos\varphi_0 \\ Y = r_0\sin\psi_0\sin\varphi_0 + x\cos\psi_0\sin\varphi_0 + y\cos\varphi_0 + z\sin\psi_0\sin\varphi_0 \\ Z = r_0\cos\psi_0 - x\sin\psi_0 + z\cos\psi_0 \end{cases}$$

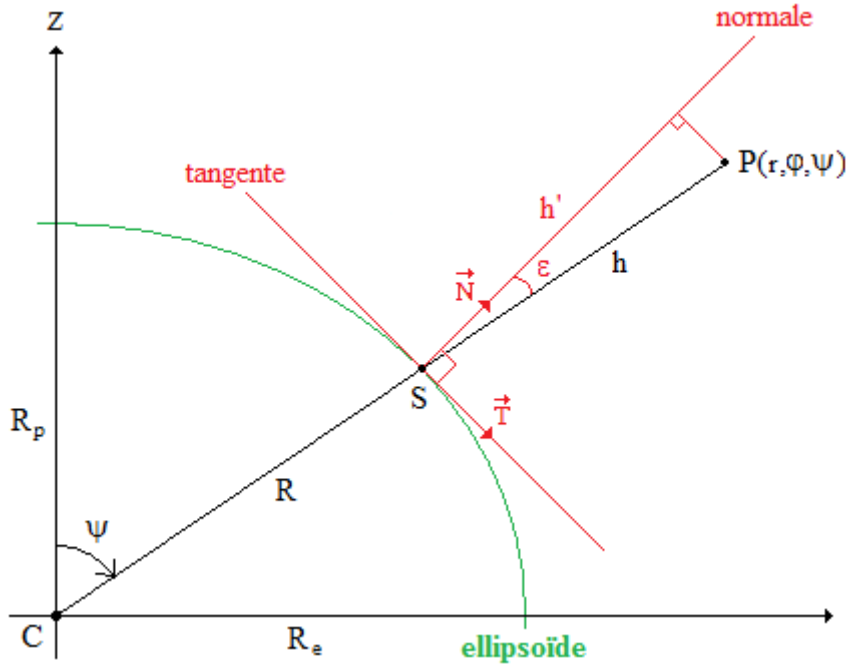
Remarque

La notation "point" sur une variable symbolise la dérivation par rapport au temps : • ⇔ d/dt , •• ⇔ d^2/dt^2
Dans les équations on peut remplacer la colatitude $\psi_0 \in [0,\pi]$ par la latitude $\lambda_0 \in [-\pi/2,+\pi/2]$ en utilisant la relation $\lambda_0 = \pi/2 - \psi_0$. Il suffit de changer les $\cos\psi_0$ en $\sin\lambda_0$ et les $\sin\psi_0$ en $\cos\lambda_0$.
On dispose de fonctions de conversion qui permettent de passer des coordonnées rectangulaires aux coordonnées sphériques en radians et aux latitudes/longitudes en degrés.

Données terrestres :
Rayon équatorial : $R_e = 6378137$ m
Rayon polaire : $R_p = 6356752.3142$ m
Masse de la Terre : $M = 5.9736\ 10^{24}$ kg
Vitesse angulaire de rotation : $\Omega = 7.292115\ 10^{-5}$ rad/s
Constante de la gravitation universelle : $G = 6.67428\ 10^{-11}$ $N.m^2/kg$

Mathématique & Informatique

ALTITUDE



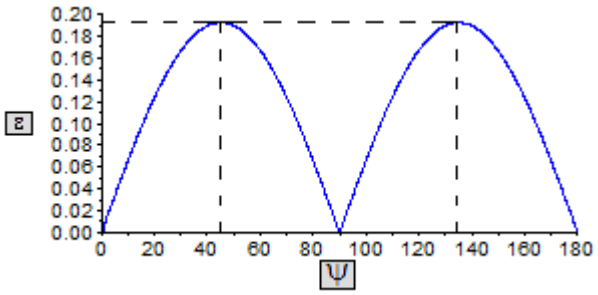
On appellera "altitude" du point P(r, φ, ψ) la valeur h = r - R, où r = CP représente la distance du point P de coordonnées (rsinψcosφ, rsinψsinφ, rcosψ) au centre de la Terre et R = CS désigne le rayon terrestre passant par le point S de coordonnées (Rsinψcosφ, Rsinψsinφ, Rcosψ) aligné avec C et P.
La normale en S a pour direction le vecteur $\vec{N}$ de coordonnées (sinψcosφ, sinψsinφ, $(R_e/R_p)^2$cosφ) situé dans le même plan méridien que P puisque le produit mixte $[\vec{N}\ ,\ \overrightarrow{CP}\ ,\ \vec{Z}]$ est nul.
La tangente en S a pour direction le vecteur $\vec{T}$ de coordonnées (cosψcosφ, cosψsinφ, - $(R_p/R_e)^2$sinψ) orthogonal à $\vec{N}$.
L'altitude h' = h cos ε mesurée sur la normale à l'ellipsoïde de référence, nécessite le calcul de la correction cos ε.
Les relations :

$$X_S^2/R_e^2 + Y_S^2/R_e^2 + Z_S^2/R_p^2 = 1$$
$$\vec{N} \cdot \overrightarrow{CS} = \| \vec{N} \| \| \overrightarrow{CS} \| \cos \varepsilon$$

donnent :

$$R = \frac{R_e R_p}{\sqrt{R_e^2 \cos^2\psi + R_p^2 \sin^2\psi}}$$

$$\cos\varepsilon = \frac{R_e^2 \cos^2\psi + R_p^2 \sin^2\psi}{\sqrt{R_e^4 \cos^2\psi + R_p^4 \sin^2\psi}}$$

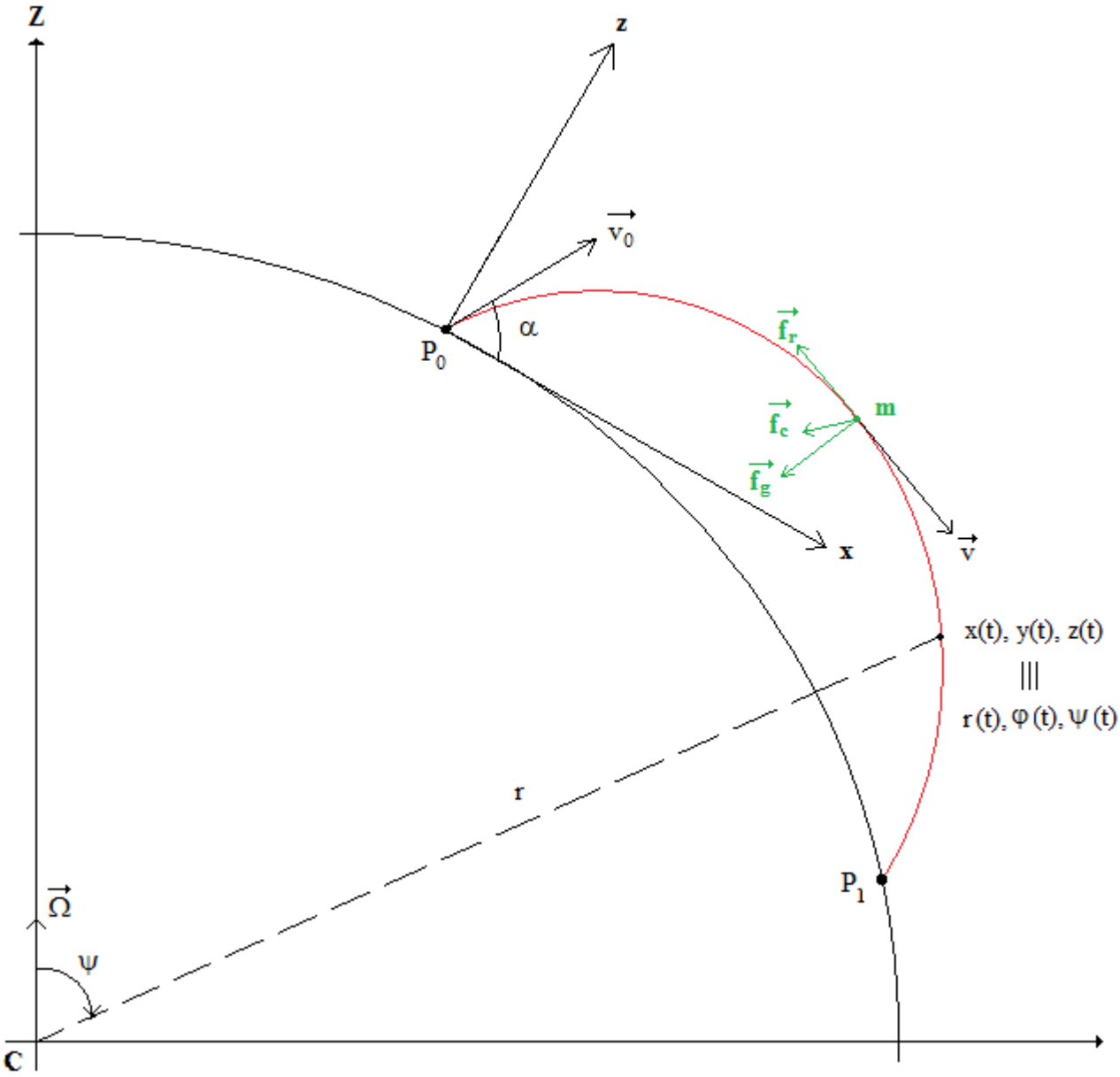


L'angle ε varie en fonction de la colatitude ψ. La valeur maximale de ε = f (ψ) est atteinte lorsque f '(ψ) = 0, soit :
ψ = Arc cos[R_p / (R_e^2 + R_p^2)$^{1/2}$] ou ψ = π - Arc cos[R_p / (R_e^2 + R_p^2)$^{1/2}$], c'est-à-dire pour ψ = 45° ou 135° environ.
D'où :
ε_{max} = Arc cos[2R_eR_p / (R_e^2 + R_p^2)] ≡ 0.1924°
$[(h - h')/h]_{max}$ = (R_e - R_p)^2 / (R_e^2 + R_p^2) = 5.64 10^{-6}
Par exemple pour des altitudes inférieures à 10 km l'écart entre h et h' ne dépasse pas 5.64 cm.

NB.
On a toujours h' ≠ h sauf aux pôles et à l'équateur (ψ ∈ {0°, 90°, 180°}).
Dans le cas où la Terre est considérée comme sphérique : $R_e = R_p = R$ et h' = h.

Mathématique & Informatique

Etudions le cas particulier suivant :
Dans un lieu "O" de latitude "lat0", longitude "lon0", altitude "alt0", un projectile situé au point P_0 (a, b, c) est lancé avec la vitesse $\overrightarrow{v_0}$ ($v_0 \cos\alpha$, 0, $v_0 \sin\alpha$).
Les conditions initiales du mouvement sont donc : $x_0 = a$, $y_0 = b$, $z_0 = c$, $\dot{x}_0 = v_0 \cos\alpha$, $\dot{y}_0 = 0$, $\dot{z}_0 = v_0 \sin\alpha$



Au temps t_0 le projectile est en P_0 , au temps t_1 il est en P_1. Trajectoire ?

Attention

Ne pas confondre (r_0 , φ_0 , ψ_0) = coordonnées sphériques de O dans CXYZ, avec (r(0) , φ(0) , ψ(0)) = coordonnées sphériques de P à t = 0 dans CXYZ. On a : (r_0 , φ_0 , ψ_0) = (r(0) , φ(0) , ψ(0)) uniquement si P est en O à l'instant initial.
En programmation Scilab les vecteurs sont indicés à partir de 1, d'où les notations :

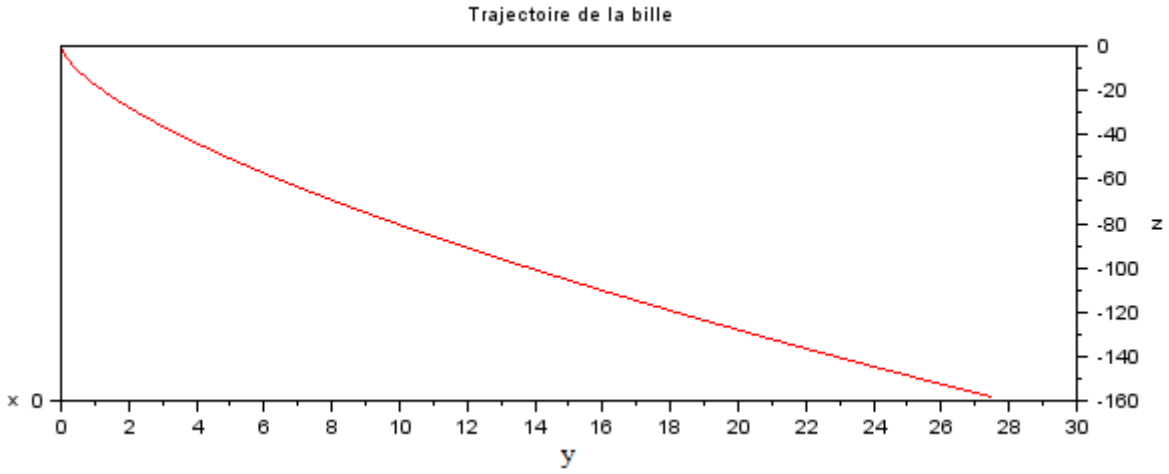
[r0 , phi0 , psi0]	≡	(r_0 , φ_0 , ψ_0)
[r(1) , phi(1) , psi(1)]	≡	(r(0) , φ(0) , ψ(0))
[r , phi , psi]	≡	(r(t) , φ(t) , ψ(t))

Mathématique & Informatique

Premier exemple : Expérience de **Reich** à Freiberg en Allemagne (1833).

Une bille est lachée sans vitesse initiale dans un puits de mine de 158 m de profondeur. On constate qu'elle ne tombe pas exactement à la verticale mais qu'elle est déviée vers l'Est de 28 mm. On prend :
lat0 = 50° 54' 43" N ; lon0 = 13° 20' 34" E ; alt0 = 398 m ;
$m = 4.75\ 10^{-2}$ kg ; $k = 8.12\ 10^{-5}$ kg/m ;
$x_0 = 0$ m ; $y_0 = 0$ m ; $z_0 = 0$ m ; $v_0 = 0$ m/s ; $\alpha = 0$ rad ;

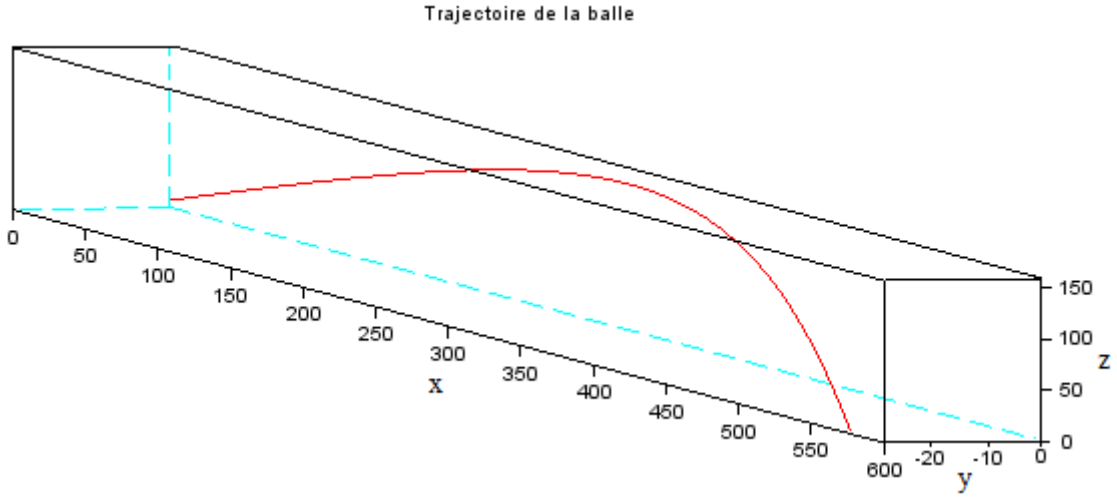
Trajectoire de la bille :



Deuxième exemple : Tir d'une balle de pistolet (Remington calibre .45).

Une balle de pistolet est tirée en direction du Sud avec un angle de 30° par rapport à l'horizontale. La portée est d'environ 570 m avec une déviation vers l'Ouest de 25 cm. On prend :
lat0 = 47° 24' 15" N ; lon0 = 0° 58' 48" E ; alt0 = 90 m ;
$m = 9.5\ 10^{-3}$ kg ; $k = 3.27\ 10^{-5}$ kg/m ;
$x_0 = 0$ m ; $y_0 = 0$ m ; $z_0 = 7$ m ; $v_0 = 220$ m/s ; $\alpha = \pi/6$ rad ;

Trajectoire de la balle :

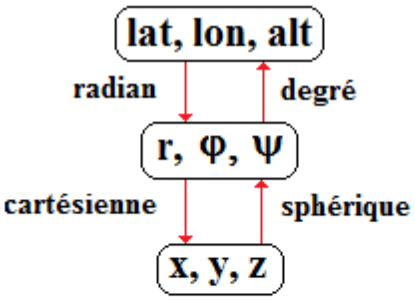


Conclusion

La force de **Coriolis** ne dévie pas toujours vers l'Est dans l'hémisphère Nord, tout dépend de l'orientation du vecteur vitesse. Dans l'exemple du tir ci-dessus, elle dévie vers l'Ouest !
En fait, la force de **Coriolis** n'est pas une force au sens propre du terme, elle n'agit pas directement sur un objet comme le ferait une force d'interaction. Il s'agit d'un concept mathématique qui permet d'effectuer le calcul d'un mouvement dans un repère relatif.

Mathématique & Informatique

CONVERSIONS



lat = latitude en ° ' " Nord/Sud
lon = longitude en ° ' " Est/Ouest
alt = altitude en mètres

---

r = rayon en mètres
φ = longitude en radians
Ψ = colatitude en radians

---

x = abscisse en mètres
y = ordonnée en mètres
z = cote en mètres

// **Cartésienne**

```
function [x,y,z] = cartesienne(r,phi,psi)
  x = r*sin(psi)*cos(phi)
  y = r*sin(psi)*sin(phi)
  z = r*cos(psi)
endfunction
```

// **Sphérique**

```
function [r,phi,psi] = spherique(x,y,z)
  if (z == 0) then
    psi = %pi/2
  end
  if (z > 0) then
    psi = atan(sqrt(x^2 + y^2)/z)
  end
  if (z < 0) then
    psi = %pi + atan(sqrt(x^2 + y^2)/z)
  end
  if (x == 0) & (y == 0) then
    phi = 0
  end
  if (x == 0) & (y > 0) then
    phi = %pi/2
  end
  if (x == 0) & (y < 0) then
    phi = 3*%pi/2
  end
  if (x > 0) & (y >= 0) then
    phi = atan(y/x)
  end
  if (x > 0) & (y < 0) then
    phi = 2*%pi + atan(y/x)
  end
  if (x < 0) & (y >= 0) then
    phi = %pi + atan(y/x)
  end
  if (x < 0) & (y < 0) then
    phi = %pi + atan(y/x)
  end
  r = sqrt(x^2 + y^2 + z^2)
endfunction
```

// **Radian**

```
function [r,phi,psi] = radian(lat,lon,alt)
  if (lat(4) == "N") then
    psi = %pi/2 - (lat(1) + lat(2)/60 + lat(3)/3600)*%pi/180
  end
  if (lat(4) == "S") then
    psi = %pi/2 + (lat(1) + lat(2)/60 + lat(3)/3600)*%pi/180
  end
  if (lon(4) == "E") then
    phi = (lon(1) + lon(2)/60 + lon(3)/3600)*%pi/180
  end
  if (lon(4) == "O") then
    phi = 2*%pi - (lon(1) + lon(2)/60 + lon(3)/3600)*%pi/180
  end
  r = alt + Re*Rp/sqrt((Re*cos(psi))^2 + (Rp*sin(psi))^2)
endfunction
```

// **Degré**

```
function [lat,lon,alt] = degre(r,phi,psi)
  lat = list() , lon = list()
  if (psi <= %pi/2) then
    angle = (%pi/2 - psi)*180/%pi
    lat(1) = floor(angle)
    lat(2) = floor((angle - lat(1))*60)
    lat(3) = (angle - lat(1) - lat(2)/60)*3600
    lat(4) = "N"
  end
  if (psi > %pi/2) then
    angle = (psi - %pi/2)*180/%pi
    lat(1) = floor(angle)
    lat(2) = floor((angle - lat(1))*60)
    lat(3) = (angle - lat(1) - lat(2)/60)*3600
    lat(4) = "S"
  end
  if (phi <= %pi) then
    angle = phi*180/%pi
    lon(1) = floor(angle)
    lon(2) = floor((angle - lon(1))*60)
    lon(3) = (angle - lon(1) - lon(2)/60)*3600
    lon(4) = "E"
  end
  if (phi > %pi) then
    angle = (2*%pi - phi)*180/%pi
    lon(1) = floor(angle)
    lon(2) = floor((angle - lon(1))*60)
    lon(3) = (angle - lon(1) - lon(2)/60)*3600
    lon(4) = "O"
  end
  alt = r - Re*Rp/sqrt((Re*cos(psi))^2 + (Rp*sin(psi))^2)
endfunction
```

Mathématique & Informatique

PROGRAMMES

```
function up = f(t,u)
  up(1) = u(4)
  up(2) = u(5)
  up(3) = u(6)
  up(4) = -G*M*u(1)/(u(1)^2 + u(2)^2 + (u(3) + r0)^2)^(3/2) - k/m*u(4)*sqrt(u(4)^2 + u(5)^2 + u(6)^2) + 2*OMEGA*u(5)*cos(psi0)
  up(5) = -G*M*u(2)/(u(1)^2 + u(2)^2 + (u(3) + r0)^2)^(3/2) - k/m*u(5)*sqrt(u(4)^2 + u(5)^2 + u(6)^2) - 2*OMEGA*(u(4)*cos(psi0) + u(6)*sin(psi0))
  up(6) = -G*M*(u(3) + r0)/(u(1)^2 + u(2)^2 + (u(3) + r0)^2)^(3/2) - k/m*u(6)*sqrt(u(4)^2 + u(5)^2 + u(6)^2) + 2*OMEGA*u(5)*sin(psi0)
endfunction

function [r,phi,psi] = geocentrique(x,y,z)
  r = zeros(n, 1) , phi = zeros(n, 1) , psi = zeros(n, 1)
  for k = 1:n
    X = r0*sin(psi0)*cos(phi0) + x(k)*cos(psi0)*cos(phi0) - y(k)*sin(phi0) + z(k)*sin(psi0)*cos(phi0)
    Y = r0*sin(psi0)*sin(phi0) + x(k)*cos(psi0)*sin(phi0) + y(k)*cos(phi0) + z(k)*sin(psi0)*sin(phi0)
    Z = r0*cos(psi0) - x(k)*sin(psi0) + z(k)*cos(psi0)
    [r(k),phi(k),psi(k)] = spherique(X, Y, Z)
  end
endfunction

// ----    Programme 1 :   expérience de Reich   ----

Re = 6378137 ; Rp = 6356752.3142 ; M = 5.9736D24 ; OMEGA = 7.292115D-5 ; G = 6.67428D-11 ;
m = 4.75D-2 ; k = 8.12D-5 ; v0 = 0 ; alpha = 0 ;
lat0 = list(50,54,43,'N') ; lon0 = list(13,20,34,'E') ; alt0 = 398 ;
[r0,phi0,psi0] = radian(lat0,lon0,alt0) ;
x0 = 0 ; y0 = 0 ; z0 = 0 ; xp0 = v0*cos(alpha) ; yp0 = 0 ; zp0 = v0*sin(alpha) ;

u0 = [x0;y0;z0;xp0;yp0;zp0] ; n = 500 ; t = linspace(0, 5.934, n) ;
u = ode(u0,0,t,f) ;
x = u(1,:)' ; y = u(2,:)' ; z =   u(3,:)' ;
[r,phi,psi] = geocentrique(x, y, z) ;
R = Re*Rp ./ sqrt((Re*cos(psi))^2 + (Rp*sin(psi))^2) ;

Variation_en_x = x($) - x(1) , Variation_en_y = y($) - y(1) , Variation_en_z = z($) - z(1)
[lat1,lon1,alt1] = degre(r($),phi($),psi($))

// ----    Programme 2 :   balle de pistolet   ----

Re = 6378137 ; Rp = 6356752.3142 ; M = 5.9736D24 ; OMEGA = 7.292115D-5 ; G = 6.67428D-11 ;
m = 9.5D-3 ; k = 3.27D-5 ; v0 = 220 ; alpha = %pi/6 ;
lat0 = list(47,24,15,'N') ; lon0 = list(0,58,48,'E') ; alt0 = 90 ;
[r0,phi0,psi0] = radian(lat0,lon0,alt0) ;
x0 = 0 ; y0 = 0 ; z0 = 7 ; xp0 = v0*cos(alpha) ; yp0 = 0 ; zp0 = v0*sin(alpha) ;

u0 = [x0;y0;z0;xp0;yp0;zp0] ; n = 500 ; t = linspace(0, 10.904, n) ;
u = ode(u0,0,t,f) ;
x = u(1,:)' ; y = u(2,:)' ; z =   u(3,:)' ;
[r,phi,psi] = geocentrique(x, y, z) ;
R = Re*Rp ./ sqrt((Re*cos(psi))^2 + (Rp*sin(psi))^2) ;

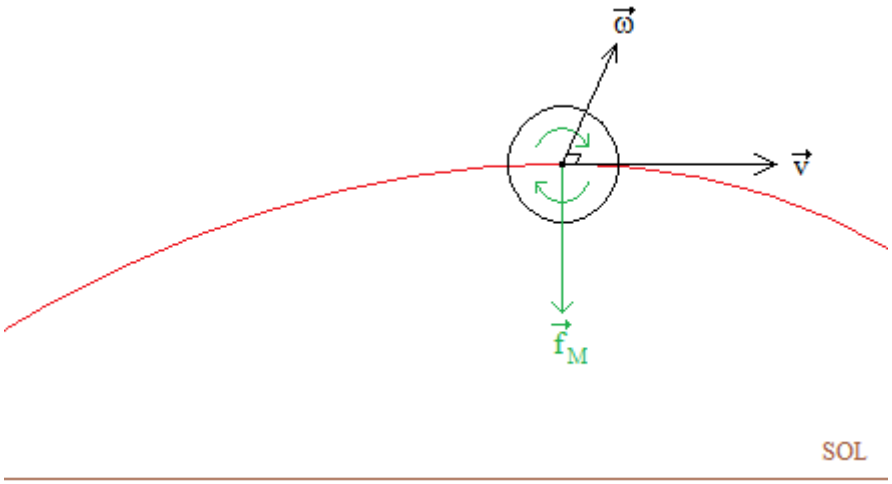
Variation_en_x = x($) - x(1) , Variation_en_y = y($) - y(1) , Variation_en_z = z($) - z(1)
[lat1,lon1,alt1] = degre(r($),phi($),psi($))
```

```
// ----    Graphiques   1   ----
subplot(2,1,1)
a = gca() ; a.x_location = "origin" ; a.y_location = "origin" ;
plot2d(t, r - R, 2)
plot2d([0,t($)], [alt0 - 158.42,alt0 - 158.42], 13)
xtitle("Altitude en fonction du temps","t","h")
subplot(2,1,2)
param3d1(x*1000, y*1000, list(z, 5), theta = 0)
xtitle("Trajectoire de la bille","x","y","z")
```

```
// ----    Graphiques   2   ----
subplot(2,1,1)
a = gca() ; a.x_location = "origin" ; a.y_location = "origin" ;
plot2d(t, r - R, 2)
plot2d([0,t($)], [alt0,alt0], 13)
xtitle("Altitude en fonction du temps","t","h")
subplot(2,1,2)
param3d1(x, y*100, list(z, 5), theta = 315)
xtitle("Trajectoire de la balle","x","y","z")
```

Mathématique & Informatique

Effet MAGNUS



Lorsque le projectlile est animé d'un mouvement de rotation sur lui-même, caractérisé par un vecteur $\vec{\omega}$ qui donne à la fois l'axe de rotation et le module de la vitesse angulaire de rotation, il s'exerce sur lui une force supplémentaire appelée force de **Magnus**. Elle s'exprime par :

$$\vec{f}_M = k_M \, v^2 \, \frac{\vec{\omega}}{\omega} \wedge \frac{\vec{v}}{v}$$

- Le module de $\vec{f}_M$ varie avec le carré de la vitesse v^2.
  ($k_M = \frac{1}{2} C_M \rho S$ est un coefficient de frottement qui dépend du projectile ; il est obtenu expérimentalement)
- La direction de $\vec{f}_M$ est donnée par le produit vectoriel des vecteurs unitaires $\vec{\omega}/\omega$ et $\vec{v}/v$.

Pour tenir compte de cette nouvelle force, on ajoute la quantité $\vec{f}_M / m$ dans le second membre du système différentiel obtenu précédemment. Soit :

$$\begin{cases} \ddot{x} = \ldots + f_{Mx}/m \\ \ddot{y} = \ldots + f_{My}/m \\ \ddot{z} = \ldots + f_{Mz}/m \end{cases}$$

NB.
Sur le schéma, la force de **Magnus** tend à ramener le projectile au sol.
Si l'axe de rotation et la vitesse ont le même support ou un support proche ($\sin(\vec{\omega},\vec{v}) = \varepsilon$) alors $\vec{f}_M$ est négligeable.

SATELLITE

Une fusée place un satellite de masse m en un point O de l'espace en lui fournissant une vitesse initiale $\overrightarrow{v_0}$.
Pour effectuer le calcul de la trajectoire du satellite par rapport à la Terre, on utilise des repères absolus (liés aux étoiles) qui coïncident à t = 0 avec les repères CXYZ et Oxyz définis précédemment.
Tout revient à considérer que ces 2 repères sont fixes dans le temps.
Supposons pour simplifier que la seule force existante soit la force de gravitation terrestre.
Dans ces conditions, le Principe Fondamental de la Dynamique exprimé dans CXYZ, se réduit à :

$$\vec{f} = m\vec{a} \quad \Leftrightarrow \quad \begin{cases} \ddot{X} = -GM\dfrac{X}{\left[X^2+Y^2+Z^2\right]^{3/2}} \\ \ddot{Y} = -GM\dfrac{Y}{\left[X^2+Y^2+Z^2\right]^{3/2}} \\ \ddot{Z} = -GM\dfrac{Z}{\left[X^2+Y^2+Z^2\right]^{3/2}} \end{cases}$$

Le point O est donné par une latitude, une longitude et une altitude. D'où, après un appel à la fonction *radian* :

$$\begin{cases} X_0 = r_0 \sin\psi_0 \cos\varphi_0 \\ Y_0 = r_0 \sin\psi_0 \sin\varphi_0 \\ Z_0 = r_0 \cos\psi_0 \end{cases}$$

Soient α_0 , β_0 , γ_0 les angles que fait le vecteur $\overrightarrow{v_0}$ avec les axes Ox, Oy, Oz.
Les coordonnées de $\overrightarrow{v_0}$ dans le repère Oxyz sont données par ses cosinus directeurs :

$$\begin{cases} \dot{x}_0 = v_0 \cos\alpha_0 \\ \dot{y}_0 = v_0 \cos\beta_0 \\ \dot{z}_0 = v_0 \cos\gamma_0 \end{cases} \qquad \text{avec :} \quad \cos^2\alpha_0 + \cos^2\beta_0 + \cos^2\gamma_0 = 1$$

Les coordonnées de $\overrightarrow{v_0}$ dans le repère CXYZ s'obtiennent à l'aide des formules de changement de repère :

$$\begin{cases} \dot{X}_0 = \dot{x}_0 \cos\psi_0 \cos\varphi_0 - \dot{y}_0 \sin\varphi_0 + \dot{z}_0 \sin\psi_0 \cos\varphi_0 \\ \dot{Y}_0 = \dot{x}_0 \cos\psi_0 \sin\varphi_0 + \dot{y}_0 \cos\varphi_0 + \dot{z}_0 \sin\psi_0 \sin\varphi_0 \\ \dot{Z}_0 = -\dot{x}_0 \sin\psi_0 + \dot{z}_0 \cos\psi_0 \end{cases}$$

Mathématique & Informatique

Appelons $\vec{r}$ (X, Y, Z) le vecteur donnant la position du satellite à l'instant t dans le repère CXYZ :

$\vec{r} = \overrightarrow{CP}$ (à t = 0, le satellite est en O : $\overrightarrow{r_0} = \overrightarrow{C0}$)

L'accélération $\vec{a} = -(GM/r^3)\,\vec{r}$ est colinéaire à $\vec{r}$, par conséquent :

$d(\vec{r}\wedge\vec{v})/dt = d\vec{r}/dt\wedge\vec{v} + \vec{r}\wedge d\vec{v}/dt = \vec{v}\wedge\vec{v} + \vec{r}\wedge\vec{a} = \vec{0} \Rightarrow \vec{r}\wedge\vec{v} = \overrightarrow{Cte} \Rightarrow$ le mouvement est plan.

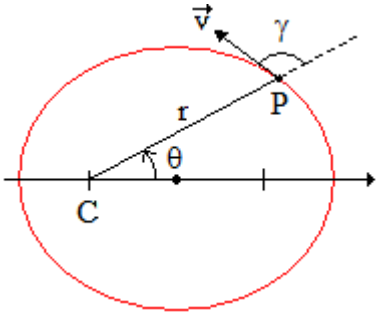
Plus précisément, en écrivant les équations en coordonnées polaires dans le plan $(\overrightarrow{r_0}, \overrightarrow{v_0})$, on établit que la trajectoire du satellite est une conique admettant le centre de la Terre pour foyer :

$$r = \frac{p}{1 - e\cos\theta}$$

r = rayon polaire , θ = angle polaire

$p = (r_0 v_0 \sin\gamma_0)^2 / GM$

$e^2 = 1 + (r_0 v_0 \sin\gamma_0/GM)^2 (v_0^2 - 2GM/r_0)$



Pour déterminer le type de trajectoire, la valeur de v_0 est comparée à la vitesse de libération $v_L = (2GM/r_0)^{1/2}$:

- $v_0 < v_L$: ellipse
- $v_0 = v_L$: parabole
- $v_0 > v_L$: hyperbole

Dans le cas particulier où le mouvement est circulaire uniforme on a $\gamma_0 = \pi/2$ et l'accélération est égale à v_0^2/r_0.

D'où : $GMm/r_0^2 = mv_0^2/r_0$. On déduit la vitesse circulaire $v_C = (GM/r_0)^{1/2}$

La vitesse minimale de satellisation pour ne pas entrer en collision avec la Terre de rayon r = R est :

$v_S = [(R/r_0)(1 - R/r_0) / (\sin^2\gamma_0 - (R/r_0)^2)]^{1/2}\, v_L$ avec $v_S < v_L$ (si $v_S \geq v_L$ le satellite ne peut pas être mis en orbite).

En prenant $\gamma_0 = \pi/2$ on a $v_S = v_L/(1 + r_0/R)^{1/2} < v_C$

En résumé, si la vitesse v_0 est trop faible ($v_0 < v_S$) le satellite retombe sur la Terre, si elle est trop forte ($v_0 > v_L$) il s'échappe dans le cosmos ...

On montre que la période de révolution d'un satellite sur une orbite elliptique est : $T = 2\pi GM / [v_L^2 - v_0^2]^{3/2}$, durée indépendante de l'angle γ_0 que fait la vitesse initiale avec le rayon vecteur.

Un satellite qui décrit une orbite circulaire autour de la Terre en T_0 = 1 jour est dit géosynchrone (le satellite semble immobile dans le ciel pour un observateur terrestre). On peut facilement calculer son altitude de révolution :

$T_0 = 2\pi r_0/v_0$ avec $v_0 = (GM/r_0)^{1/2} \Rightarrow h = r_0 - R = (GMT_0^2/4\pi^2)^{1/3} - R \approx 36\,000$ km

Point d'observation.

On se place sur l'axe dirigé par le vecteur $\vec{A} = \overrightarrow{r_0}\wedge\overrightarrow{v_0}$ orthogonal au plan de la trajectoire.

Les coordonnées de $\vec{A}$ dans Oxyz sont : $A_x = r_0 v_0 \cos\beta_0$; $A_y = -r_0 v_0 \cos\alpha_0$; $A_z = 0$

Le module de $\vec{A}$ est : $A = r_0 v_0 |\sin\gamma_0|$ = constante des aires

Dans CXYZ on a : $A_X = Y_0\dot{Z}_0 - Z_0\dot{Y}_0$; $A_Y = Z_0\dot{X}_0 - X_0\dot{Z}_0$; $A_Z = X_0\dot{Y}_0 - Y_0\dot{X}_0$

Ces valeurs correspondent à une longitude L et à une colatitude l calculées avec la fonction de conversion *spherique* :

[A, L, l] = *spherique* (A_X, A_Y, A_Z)

D'où les angles "theta" et "alpha" en degrés utilisés par les fonctions graphiques *plot3d* et *param3d* :

theta = L × 180/π et alpha = l × 180/π

Exemple 1.

Satellite en orbite à 1000 km d'altitude, passant au dessus de Paris de coordonnées géographiques :

Latitude = 48° 51' 24" Nord ; *Longitude* = 2° 21' 7" Est

Sur les graphiques la Terre et le satellite (point surdimensionné en noir) sont représentés à t = 0.

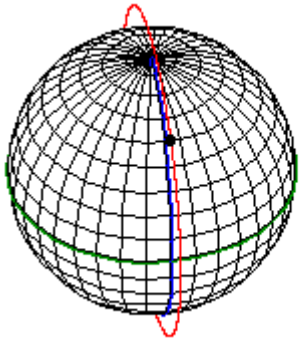
Le méridien de Greenwich (longitude 0°) est tracé en bleu, l'équateur (latitude 0° ⇔ colatitude 90°) est en vert.

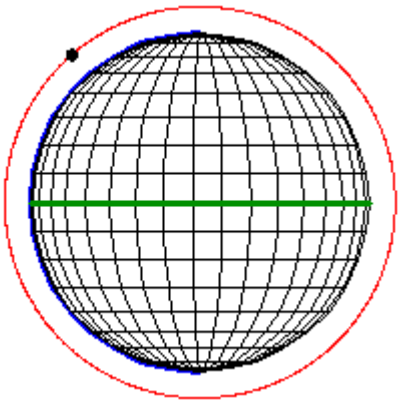
La trajectoire du satellite est dessinée en rouge.

Pour prendre en compte la rotation de la Terre et connaître la position du satellite dans les repères mobiles CXYZ et Oxyz, c'est-à-dire pour calculer la latitude, la longitude et l'altitude du satellite à chaque instant, il faut modifier les équations et faire intervenir la force de **Coriolis** (voir calcul de la balistique).

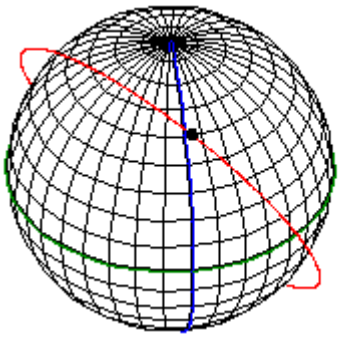
Mathématique & Informatique

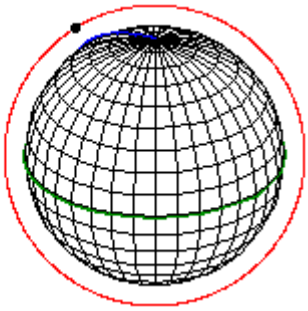
✶ *Cas 1 :* $\mathbf{v_0} = \mathbf{v_C}$ $(\alpha_0 = 0\ ,\ \beta_0 = \pi/2\ ,\ \gamma_0 = \pi/2)$; *orbite circulaire* $T = 1\ h\ 44\ mn\ 50\ s$



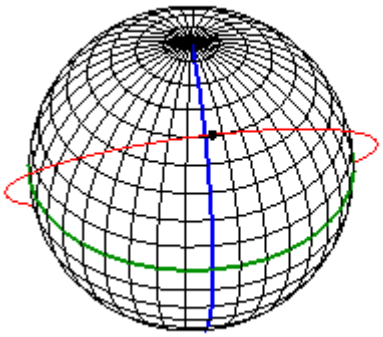


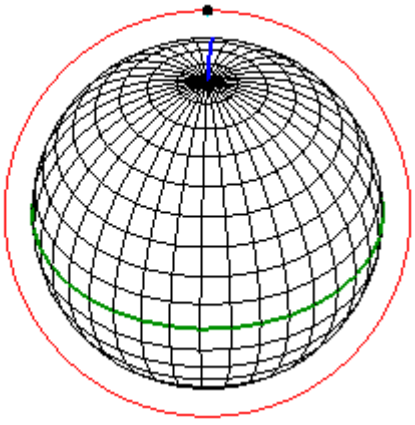
✶ *Cas 2 :* $\mathbf{v_0} = \mathbf{v_C}$ $(\alpha_0 = \pi/4\ ,\ \beta_0 = \pi/4\ ,\ \gamma_0 = \pi/2)$; *orbite circulaire* $T = 1\ h\ 44\ mn\ 50\ s$



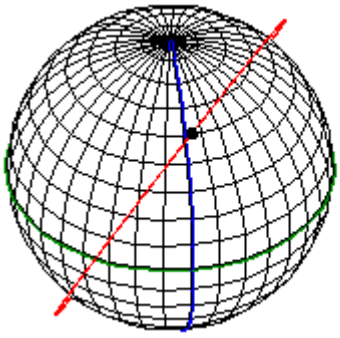


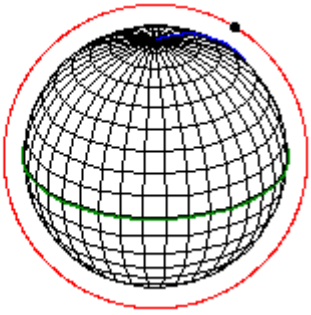
✶ *Cas 3 :* $\mathbf{v_0} = \mathbf{v_C}$ $(\alpha_0 = \pi/2\ ,\ \beta_0 = 0\ ,\ \gamma_0 = \pi/2)$; *orbite circulaire* $T = 1\ h\ 44\ mn\ 50\ s$



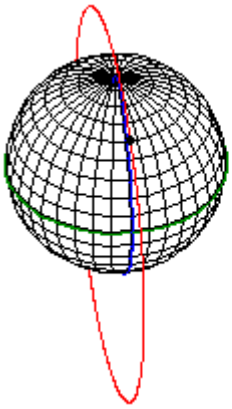


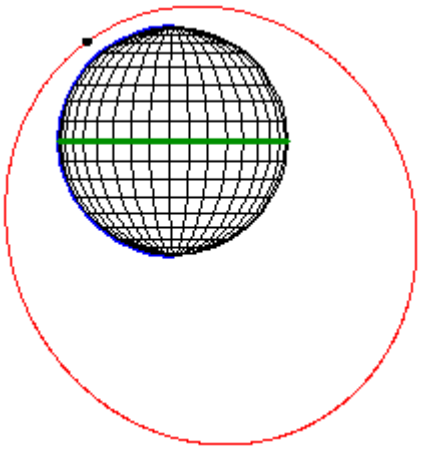
✶ *Cas 4 :* $\mathbf{v_0} = \mathbf{v_C}$ $(\alpha_0 = 3\pi/4\ ,\ \beta_0 = -\ \pi/4\ ,\ \gamma_0 = \pi/2)$; *orbite circulaire* $T = 1\ h\ 44\ mn\ 50\ s$



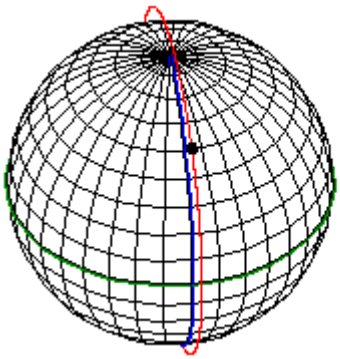


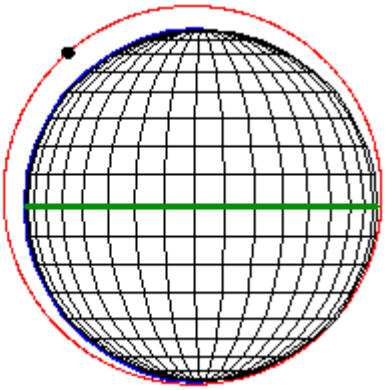
✶ *Cas 5 :* $\mathbf{v_0 < v_L}$ $(\alpha_0 = \pi/36\ ,\ \beta_0 = \pi/2\ ,\ \gamma_0 = 17\pi/36)$; *orbite elliptique*



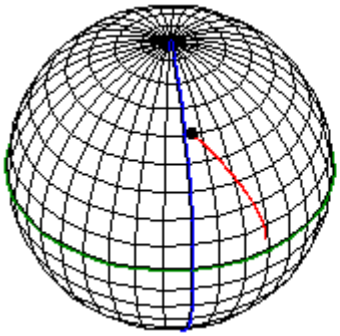


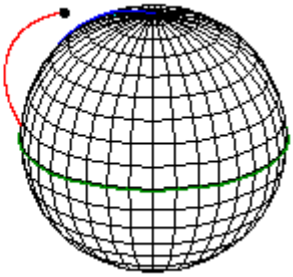
✶ *Cas 6 :* $\mathbf{v_0 = v_S}$ $(\alpha_0 = 0\ ,\ \beta_0 = \pi/2\ ,\ \gamma_0 = \pi/2)$; *limite de satellisation*



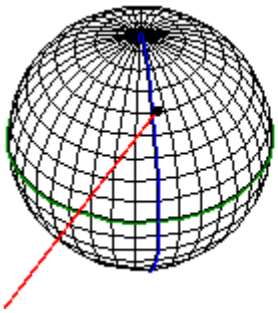


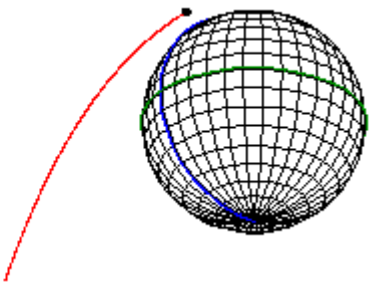
✶ *Cas 7 :* $\mathbf{v_0 < v_L}$ $(\alpha_0 = \pi/4\ ,\ \beta_0 = \pi/3\ ,\ \gamma_0 = \pi/3)$; *collision avec la Terre*





✶ *Cas 8 :* $\mathbf{v_0 > v_L}$ $(\alpha_0 = -\ \pi/4\ ,\ \beta_0 = 3\pi/4\ ,\ \gamma_0 = \pi/2)$; *libération hyperbolique*





Mathématique & Informatique

FONCTION SATELLITE.

```
function satellite()

  //  Calculs
  function Up = f(t,U)
    Up(1) = U(4)
    Up(2) = U(5)
    Up(3) = U(6)
    Up(4) = -GM*U(1)/(U(1)^2 + U(2)^2 + U(3)^2)^(3/2)
    Up(5) = -GM*U(2)/(U(1)^2 + U(2)^2 + U(3)^2)^(3/2)
    Up(6) = -GM*U(3)/(U(1)^2 + U(2)^2 + U(3)^2)^(3/2)
  endfunction
  X0 = r0*sin(psi0)*cos(phi0)
  Y0 = r0*sin(psi0)*sin(phi0)
  Z0 = r0*cos(psi0)
  Xp0 = v0*cos(alpha0)*cos(psi0)*cos(phi0) - v0*cos(beta0)*sin(phi0) + v0*cos(gamma0)*sin(psi0)*cos(phi0)
  Yp0 = v0*cos(alpha0)*cos(psi0)*sin(phi0) + v0*cos(beta0)*cos(phi0) + v0*cos(gamma0)*sin(psi0)*sin(phi0)
  Zp0 = - v0*cos(alpha0)*sin(psi0) + v0*cos(gamma0)*cos(psi0)
  U0 = [X0;Y0;Z0;Xp0;Yp0;Zp0]
  n = 1000 , t = linspace(0, T, n)
  U = ode(U0,0,t,f)
  X = U(1,:)' , Y = U(2,:)' , Z =  U(3,:)'

  //  Graphiques
  function dessin(angle1,angle2)
      // Terre
    function  [TeX, TeY, TeZ] = ellipsoide(u, v)
      TeX = Re*sin(v).*cos(u) , TeY = Re*sin(v).*sin(u) , TeZ = Rp*cos(v)
    endfunction
    u = linspace(0, 2*%pi, 37) , v = linspace(0, %pi, 19)
    [TeXf,  TeYf,  TeZf] = eval3dp(ellipsoide, u, v)
    plot3d(TeXf, TeYf, TeZf, theta = angle1, alpha = angle2, flag = [8,6,0])
      // Equateur
    u = linspace(0, 2*%pi, 37)'
    EqX = Re*sin(u) , EqY = Re*cos(u) , EqZ = 0*u
    xset("thickness", 2)
    param3d1(EqX, EqY, list(EqZ,13), theta = angle1, alpha = angle2, flag = [6,0])
    xset("thickness", 1)
      // Méridien de Greenwich
    v = linspace(0, %pi, 19)'
    GrX = Re*sin(v) , GrY = 0*v , GrZ = Rp*cos(v)
    xset("thickness", 2)
    param3d1(GrX, GrY, list(GrZ,2), theta = angle1, alpha = angle2, flag = [6,0])
    xset("thickness", 1)
      // Satellite
    function  [SatX, SatY, SatZ] = sphere(u, v)
      r = 2D5
      SatX = X0 + r*sin(v).*cos(u) , SatY = Y0 + r*sin(v).*sin(u) , SatZ = Z0 + r*cos(v)
    endfunction
    u = linspace(0, 2*%pi, 37) , v = linspace(0, %pi, 19)
    [SatXf,  SatYf,  SatZf] = eval3dp(sphere, u, v)
    plot3d(SatXf, SatYf, SatZf, theta = angle1, alpha = angle2, flag = [-1,6,0])
      // Trajectoire du Satellite
    param3d1(X, Y, list(Z, 5), theta = angle1, alpha = angle2, flag = [6,0])
  endfunction
  AX = Y0*Zp0 - Z0*Yp0 , AY = Z0*Xp0 - X0*Zp0 , AZ = X0*Yp0 - Y0*Xp0
  [A,L,l] = spherique(AX,AY,AZ)
  drawlater()
  subplot(1,2,1) , dessin(phi0*180/%pi - 10,psi0*180/%pi + 10)
  subplot(1,2,2) , dessin(L*180/%pi,l*180/%pi)
  drawnow()

endfunction
```

Mathématique & Informatique

PROGRAMME PRINCIPAL.

```
Re = 6378137 ; Rp = 6356752.3142 ; GM = 3.986004418D+14 ; OMEGA = 7.292115D-5 ;

lat0 = list(48,51,24,'N') ; lon0 = list(2,21,7,'E') ; alt0 = 1000D3 ;
[r0,phi0,psi0] = radian(lat0,lon0,alt0) ;
alpha0 = 0 ; beta0 = %pi/2 ; gamma0 = %pi/2 ;
v0 = sqrt(GM/r0) ;
T = 2*%pi*r0/v0
satellite()

xclick() ; clf()
lat0 = list(48,51,24,'N') ; lon0 = list(2,21,7,'E') ; alt0 = 1000D3 ;
[r0,phi0,psi0] = radian(lat0,lon0,alt0) ;
alpha0 = %pi/4 ; beta0 = %pi/4 ; gamma0 = %pi/2 ;
v0 = sqrt(GM/r0) ;
T = 2*%pi*r0/v0
satellite()

xclick() ; clf()
lat0 = list(48,51,24,'N') ; lon0 = list(2,21,7,'E') ; alt0 = 1000D3 ;
[r0,phi0,psi0] = radian(lat0,lon0,alt0) ;
alpha0 = %pi/2 ; beta0 = 0 ; gamma0 = %pi/2 ;
v0 = sqrt(GM/r0) ;
T = 2*%pi*r0/v0
satellite()

xclick() ; clf()
lat0 = list(48,51,24,'N') ; lon0 = list(2,21,7,'E') ; alt0 = 1000D3 ;
[r0,phi0,psi0] = radian(lat0,lon0,alt0) ;
alpha0 = 3*%pi/4 ; beta0 = -%pi/4 ; gamma0 = %pi/2 ;
v0 = sqrt(GM/r0) ;
T = 2*%pi*r0/v0
satellite()

xclick() ; clf()
lat0 = list(48,51,24,'N') ; lon0 = list(2,21,7,'E') ; alt0 = 1000D3 ;
[r0,phi0,psi0] = radian(lat0,lon0,alt0) ;
alpha0 = %pi/18 ; beta0 = %pi/2 ; gamma0 = 17*%pi/36 ;
v0 = 1.2*sqrt(GM/r0) ;
T = 2*%pi*GM/((2*GM/r0) - v0^2)^(3/2)
satellite()

xclick() ; clf()
lat0 = list(48,51,24,'N') ; lon0 = list(2,21,7,'E') ; alt0 = 1000D3 ;
[r0,phi0,psi0] = radian(lat0,lon0,alt0) ;
alpha0 = 0 ; beta0 = %pi/2 ; gamma0 = %pi/2 ;
v0 = sqrt((2*r0*Re - 2*Re^2)/(r0^2*sin(gamma0)^2 - Re^2))*sqrt(GM/r0) ;
T = 2*%pi*GM/((2*GM/r0) - v0^2)^(3/2)
satellite()

xclick() ; clf()
lat0 = list(48,51,24,'N') ; lon0 = list(2,21,7,'E') ; alt0 = 1000D3 ;
[r0,phi0,psi0] = radian(lat0,lon0,alt0) ;
alpha0 = %pi/4 ; beta0 = %pi/3 ; gamma0 = %pi/3 ;
v0 = 0.7*sqrt(GM/r0) ;
T = 2000
satellite()

xclick() ; clf()
lat0 = list(48,51,24,'N') ; lon0 = list(2,21,7,'E') ; alt0 = 1000D3 ;
[r0,phi0,psi0] = radian(lat0,lon0,alt0) ;
alpha0 = -%pi/4 ; beta0 = 3*%pi/4 ; gamma0 = %pi/2 ;
v0 = 1.5*sqrt(GM/r0) ;
T = 2000
satellite()
```

Exemple 2.

Le problème :
A l'instant initial le satellite est en O avec une vitesse $\overrightarrow{v_0}$.
Comment choisir $\overrightarrow{v_0}$ pour qu'il retombe sur la Terre en un point P_1 de longitude φ_1 et de colatitude ψ_1 données ?
Une solution :
On prend $\overrightarrow{v_0}$ orthogonale au rayon vecteur $\overrightarrow{r_0}$ ($\gamma_0 = \pi/2$) de sorte que le satellite soit à l'apogée de sa trajectoire elliptique à l'instant initial (le grand axe de l'ellipse est dirigé selon le vecteur $\overrightarrow{r_0}$).

1) Condition pour que P_1 appartienne au plan de la trajectoire.

On doit avoir : $[\ \overrightarrow{r_1}\ ,\ \overrightarrow{r_0}\ ,\ \overrightarrow{v_0}\ ] = 0$

$\overrightarrow{r_1}$ ($X_1 = r_1 \sin\psi_1 \cos\varphi_1$, $Y_1 = r_1 \sin\psi_1 \sin\varphi_1$, $Z_1 = r_1 \cos\psi_1$)
$\overrightarrow{r_0}$ ($X_0 = r_0 \sin\psi_0 \cos\varphi_0$, $Y_0 = r_0 \sin\psi_0 \sin\varphi_0$, $Z_0 = r_0 \cos\psi_0$)
$\overrightarrow{v_0}$ ($X_0^{\bullet} = v_0 \cos\alpha_0 \cos\psi_0 \cos\varphi_0 - v_0 \sin\alpha_0 \sin\varphi_0$, $Y_0^{\bullet} = v_0 \cos\alpha_0 \cos\psi_0 \sin\varphi_0 + v_0 \sin\alpha_0 \cos\varphi_0$, $Z_0^{\bullet} = - v_0 \cos\alpha_0 \sin\psi_0$)

On obtient une équation d'inconnue α_0 du type :
$a\cos\alpha_0 + b\sin\alpha_0 = 0 \quad \Rightarrow \quad \alpha_0 = \text{Arc tan}\ (\ \ldots\ )$
Tous les calculs faits :

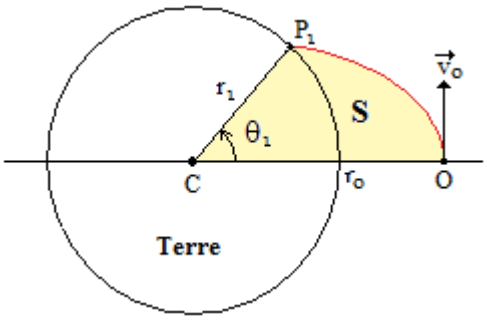
$$\alpha_0 \ = \ Arc\,tan\left[\frac{\sin\psi_1 \sin(\varphi_0 - \varphi_1)}{\sin\psi_0 \cos\psi_1 - \cos\psi_0 \sin\psi_1 \cos(\varphi_0 - \varphi_1)}\right]$$

Pour que l'angle $(\overrightarrow{r_1}, \overrightarrow{v_0})$ soit inférieur à $\pi/2$, on changera au besoin α_0 en $\alpha_0 + \pi$, c'est-à-dire $\overrightarrow{v_0}$ en $-\overrightarrow{v_0}$.
Les angles directeurs de $\overrightarrow{v_0}$ sont : α_0 , $\beta_0 = \pi/2 - \alpha_0$, $\gamma_0 = \pi/2$.

2) Condition pour que P_1 appartienne à la trajectoire elliptique.

On doit avoir : $r_1 = p / (1 - e\cos\theta_1)$

$p = (r_0 v_0)^2/GM$ → p = paramètre focal
$e^2 = 1 + (r_0 v_0/GM)^2 (v_0^2 - 2GM/r_0)$ → e = excentricité
$\cos\theta_1 = \overrightarrow{r_1} \cdot \overrightarrow{r_0} / \| \overrightarrow{r_1} \| \| \overrightarrow{r_0} \|$ → θ = angle polaire



On obtient une équation bicarrée d'inconnue v_0 :
$a (v_0^2)^2 + b (v_0^2) + c = 0 \quad \Rightarrow \quad v_0 = (\ \ldots\ )^{1/2}$
La seule racine qui convient est :

$$v_0 \ = \ \sqrt{\frac{1-\cos\theta_1}{\frac{r_0}{r_1} - \cos\theta_1}}\ \sqrt{\frac{GM}{r_0}}$$

v_0 est inférieure à la vitesse circulaire $v_c = (GM/r_0)^{1/2}$

3) Condition pour tenir compte de la rotation de la Terre.

Dans un mouvement à force centrale, la vitesse aréolaire est constante :
$dS/dt = A/2 \quad \Rightarrow \quad t = 2\,S/A \quad$ avec $\quad A =$ constante des aires $= r_0\, v_0$
L'aire S d'un secteur d'ellipse est une fonction de l'angle polaire θ :
$S = \frac{1}{2} \int r^2\, d\theta \quad$ avec $\quad r = p / (1 - e\cos\theta)$
L'intégrale se calcule en la ramenant à une fraction rationnelle en $\tan(\theta/2) = ((1 - \cos\theta)/(1 + \cos\theta))^{1/2}$
On trouve :

$$S = \frac{p^2}{(1-e^2)^{3/2}}\left[Arc\,tan\,\tau + \frac{e\tau}{1+\tau^2}\right] \quad avec\ \tau = \sqrt{\frac{1+e}{1-e}}\ \sqrt{\frac{1-\cos\theta}{1+\cos\theta}}$$

La valeur $\theta = \pi$ correspond à l'aire d'une demi-ellipse : $S = [p^2 / (1 - e^2)^{3/2}]\ \pi/2$

D'autre part, dans CXYZ considéré comme fixe, le point $P_1(r_1 , \varphi_1 , \psi_1)$ de la surface terrestre se déplace à colatitude constante ψ_1 et à rayon constant $(r_1 \sin\psi_1)$ autour de l'axe CZ. Sa longitude φ varie en fonction de la vitesse angulaire de rotation de la Terre : $\varphi = \varphi_1 + \Omega\, t$.

- Le satellite met le temps T_s pour atteindre un point de longitude φ.
  En remplaçant la constante φ_1 par la variable φ, on a :
  $\cos\theta = \sin\psi_0\sin\psi_1\cos(\varphi_0 - \varphi) + \cos\psi_0\cos\psi_1 = f_1(\varphi)$
  La condition $r_1 = p / (1 - e\cos\theta)$ fournit la vitesse v_0 :
  $v_0 = [(1 - \cos\theta) / (r_0/r_1 - \cos\theta)]^{1/2}\ (GM/r_0)^{1/2} = f_2(\varphi)$
  D'où :
  $S = f_3(\varphi)$
  Donc :
  $T_s = 2\, S/A = f(\varphi)$

- La Terre met le temps T_t pour atteindre un point de longitude φ.
  $T_t = (\varphi - \varphi_1) / \Omega$

Si $T_s = T_t = T$ alors le satellite et la Terre se retrouvent simultanément au même point $P_2(r_2 , \varphi_2 , \psi_2)$.
Comme $r_2 = r_1$ et $\psi_2 = \psi_1$, il suffit de déterminer l'angle φ_2 .
Il faut vérifier : $f(\varphi) = (\varphi - \varphi_1) / \Omega \quad \Leftrightarrow \quad \varphi - \varphi_1 - \Omega\, f(\varphi) = 0$

On obtient finalement une équation en φ : **$F(\varphi) = 0$**

La résolution s'effectue avec la fonction Scilab *fsolve* en prenant pour valeur initiale $\varphi = \varphi_1$.
Le résultat modulo 2π donne l'angle φ_2.

Sur les graphiques, le point en violet correspond à P_1 (t = 0) et le point en rouge à P_2 (t = T).

Cas limite : $\cos\theta = \pm 1$.

Dans le cas où $\cos\theta \approx \pm 1$ on donnera à $\cos\theta$ la valeur $\pm (1 - eps)$, avec eps = 1D -11 pour une erreur d'angle inférieure à 1".

Distance parcourue par le satellite.

De t = 0 à t = T, le satellite parcourt la distance D donnée par l'intégrale en coordonnées polaires :

$D = \int (r^2 + r'^2)^{1/2}\, d\theta$

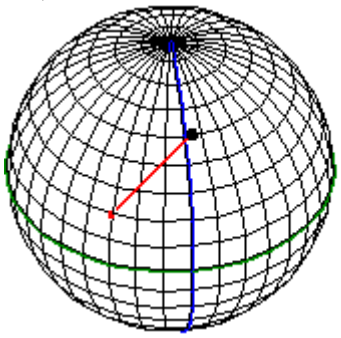
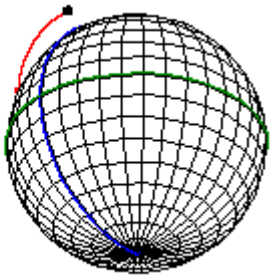
La variable θ varie de $\theta_1 = 0$ à $\theta_2 = \theta_T$ tel que : $\cos\theta_T = \sin\psi_0\sin\psi_1\cos(\varphi_0 - \varphi_2) + \cos\psi_0\cos\psi_1$
On sait que le calcul de la longueur d'un arc d'ellipse conduit à une intégrale elliptique incomplète qui ne possède pas de primitive exprimable avec les fonctions usuelles.
On approximera donc D avec la fonction Scilab *integrate*.
Dérivons le rayon r par rapport à l'angle θ :
$r = p / (1 - e\cos\theta) \quad \Rightarrow \quad r' = - p\, e \sin\theta / (1 - e\cos\theta)^2$
Il s'ensuit :

$$D = p \int_0^{\theta_T} \frac{\sqrt{1 - 2e\cos\theta + e^2}}{(1 - e\cos\theta)^2}\, d\theta$$

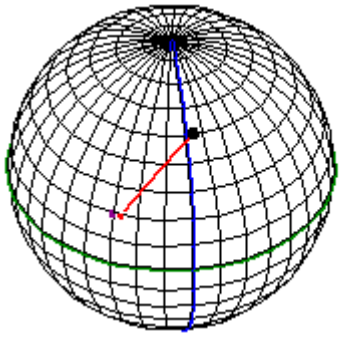
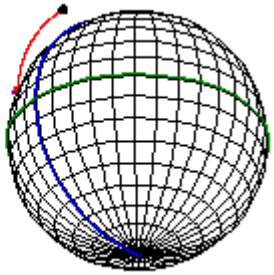
Mathématique & Informatique

Reprenons l'exemple du satellite situé à 1000 km d'altitude au dessus de Paris à l'instant initial (t =0).

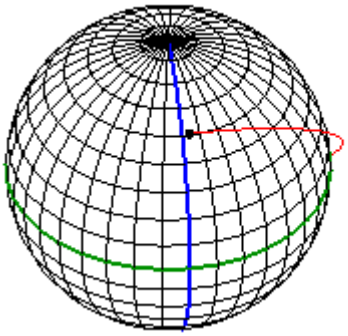
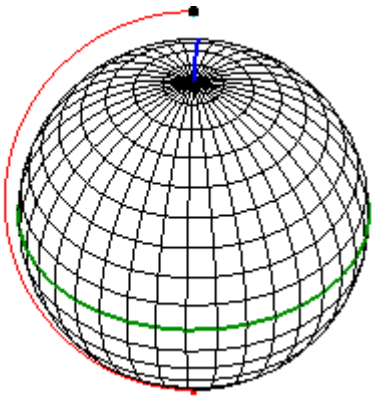
✶ *Cas 1 :* **P_1 = (20° 0' 0'' N , 30° 0' 0'' O)** $\Rightarrow$ v_0 = 5633 m/s , α_0 = - 0.932866 rad , D = 4872649 m
(Terre immobile)

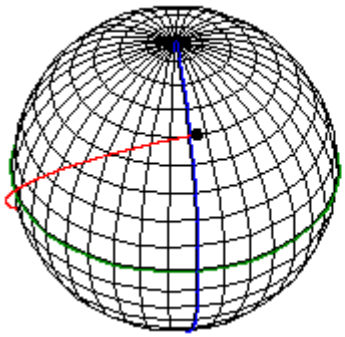
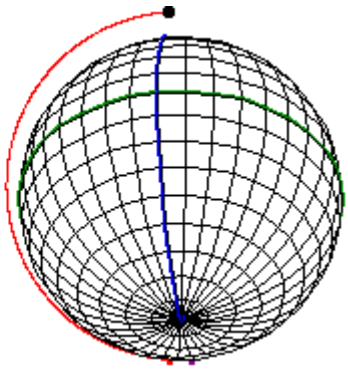
✶ *Cas 2 :* **P_1 = (20° 0' 0'' N , 30° 0' 0'' O)** $\Rightarrow$ v_0 = 5532 m/s , α_0 = - 0.859813 rad , D = 4673185 m
(Terre mobile)

✶ *Cas 3 :* **P_1 = (48° 51' 24'' S , 177° 38' 53'' O)** $\Rightarrow$ v_0 = 7083 m/s , α_0 = 1.57080 rad , D = 21541468 m
(Terre immobile)

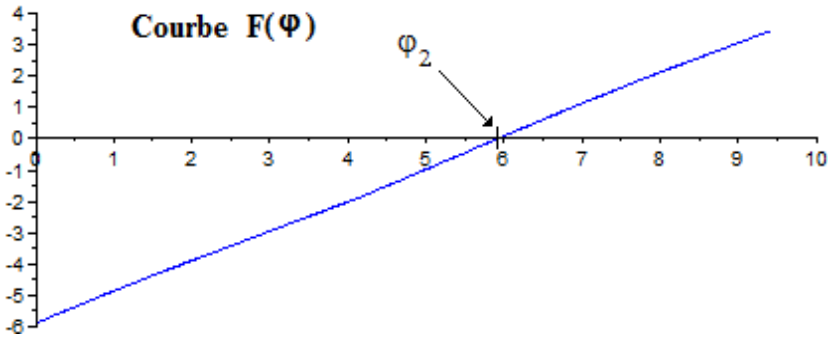
✶ *Cas 4 :* **P_1 = (48° 51' 24'' S , 177° 38' 53'' O)** $\Rightarrow$ v_0 = 7082 m/s , α_0 = - 1.495778 rad , D = 20703249 m
(Terre mobile)

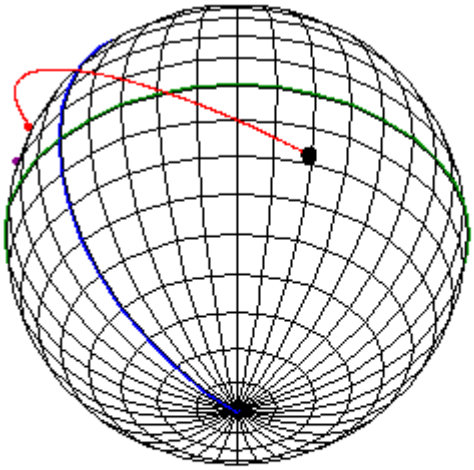



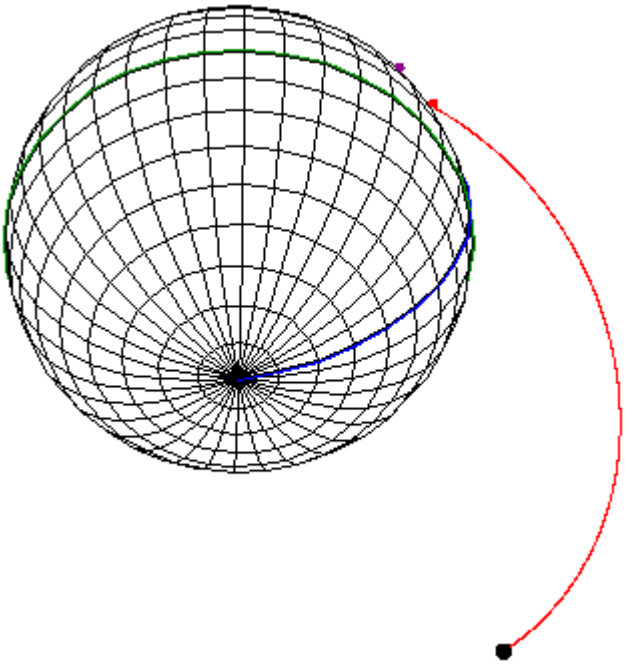
Mathématique & Informatique

✶ **P_0 = (30° S , 60° E , 7000 km) ; P_1 = (20° N , 35° O , 10 km)**

⇒ v_0 = 3,812 km/s, α_0 = 4.332 rad, D = 17 926 km

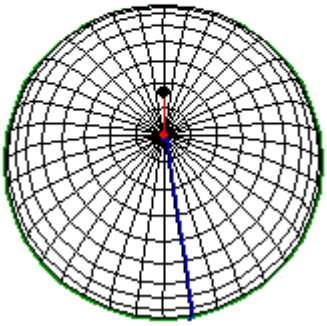


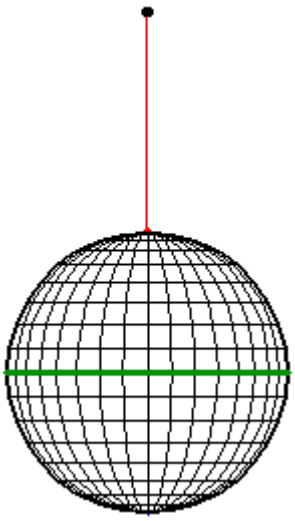




✶ **P_0 = (90° N , 10 000 km) ; P_1 = (90° N , 0 km)**

⇒ v_0 = 0 km/s , α_0 = 1.570 rad , D = 10 000 km





Mathématique & Informatique

PROGRAMME 1 *(Terre immobile)*

```
// ----     Programme Principal   ----

Re = 6378137 ; Rp = 6356752.3142 ; GM = 3.986004418D+14 ; OMEGA = 7.292115D-5 ;

lat0 = list(48,51,24,'N') ; lon0 = list(2,21,7,'E') ; alt0 = 1000D3 ;
[r0,phi0,psi0] = radian(lat0,lon0,alt0) ;
[X0,Y0,Z0] = cartesienne(r0,phi0,psi0) ;

lat1 = list(20,0,0,'N') ; lon1 = list(30,0,0,'O') ; alt1 = 0 ;
[r1,phi1,psi1] = radian(lat1,lon1,alt1) ;
[X1,Y1,Z1] = cartesienne(r1,phi1,psi1) ;

a = sin(psi1)*sin(phi0 - phi1) ;
b = sin(psi0)*cos(psi1) - cos(psi0)*sin(psi1)*cos(phi0 - phi1) ;
if abs(b) > %eps then, alpha0 = atan(a/b) ; else, alpha0 = %pi/2 ; end
X1u = sin(psi1)*cos(phi1) ; Y1u = sin(psi1)*sin(phi1) ; Z1u = cos(psi1) ;
Xp0u = cos(alpha0)*cos(psi0)*cos(phi0) - sin(alpha0)*sin(phi0) ;
Yp0u = cos(alpha0)*cos(psi0)*sin(phi0) + sin(alpha0)*cos(phi0) ;
Zp0u = - cos(alpha0)*sin(psi0) ;
if (X1u*Xp0u + Y1u*Yp0u + Z1u*Zp0u) < 0 then, alpha0 = alpha0 + %pi ; end
beta0 = %pi/2 - alpha0 ; gamma0 = %pi/2 ;

eps = 1D-11 ;
cos_theta = sin(psi0)*sin(psi1)*cos(phi0 - phi1) + cos(psi0)*cos(psi1) ;
if abs(cos_theta - 1) <= eps then, cos_theta = 1 - eps ; end
if abs(cos_theta + 1) <= eps then, cos_theta = -1 + eps ; end
v0 = sqrt((1 - cos_theta)/(r0/r1 - cos_theta))*sqrt(GM/r0) ;
p = (r0*v0)^2/(GM) ;
e = sqrt(1 + (r0*v0/GM)^2*(v0^2-2*GM/r0)) ;
tau = sqrt((1+e)/(1-e))*sqrt((1-cos_theta)/(1+cos_theta)) ;
T = 2*(p^2/(1-e^2)^(3/2))*(atan(tau) + e*tau/(1+tau^2))/(r0*v0) ;

if cos_theta < 1 - 1D-8 then
     D = p*integrate('sqrt(1-2*e*cos(theta)+e^2)/(1-e*cos(theta))^2', 'theta', 0, acos(cos_theta)) ;
else
   D = alt0 - alt1 ;
end

v0, alpha0, T, D
satellite()
```

PROGRAMME 2 *(Terre mobile)*

```
// ----     Trajectoire elliptique   ----

function [cos_theta,v0,p,e,T] = parametre(phi)
  eps = 1D-11
  cos_theta = sin(psi0)*sin(psi1)*cos(phi0 - phi) + cos(psi0)*cos(psi1)
  if abs(cos_theta - 1) <= eps then, cos_theta = 1 - eps , end
  if abs(cos_theta + 1) <= eps then, cos_theta = -1 + eps , end
  v0 = sqrt((1 - cos_theta)/(r0/r1 - cos_theta))*sqrt(GM/r0)
  p = (r0*v0)^2/(GM)
  e = sqrt(1 + (r0*v0/(GM))^2*(v0^2-2*GM/r0))
  tau = sqrt((1+e)/(1-e))*sqrt((1-cos_theta)/(1+cos_theta))
  T = 2*(p^2/(1-e^2)^(3/2))*(atan(tau) + e*tau/(1+tau^2))/(r0*v0) ;
endfunction

function graphique(phi)
  THETA = linspace(0,3*%pi,500) , F = zeros(1,500)
  for k = 1:500
    [cos_theta,v0,p,e,T] = parametre(THETA(k))
    F(k) = THETA(k) - phi1 - OMEGA*T
  end
  a = gca() ; a.x_location = "origin" ; a.y_location = "origin"
  plot2d(THETA, F, 2) , plot2d(phi, 0, -1)
  xtitle("Courbe F(phi)") , xclick() , clf()
endfunction

function F = trajectoire(phi)
  [cos_theta,v0,p,e,T] = parametre(phi)
  F = phi - phi1 - OMEGA*T
endfunction

// ----     Programme Principal   ----

Re = 6378137 ; Rp = 6356752.3142 ; GM = 3.986004418D+14 ; OMEGA = 7.292115D-5 ;

lat0 = list(48,51,24,'N') ; lon0 = list(2,21,7,'E') ; alt0 = 1000D3 ;
[r0,phi0,psi0] = radian(lat0,lon0,alt0) ;
[X0,Y0,Z0] = cartesienne(r0,phi0,psi0) ;

lat1 = list(20,0,0,'N') ; lon1 = list(30,0,0,'O') ; alt1 = 0 ;
[r1,phi1,psi1] = radian(lat1,lon1,alt1) ;
[X1,Y1,Z1] = cartesienne(r1,phi1,psi1) ;

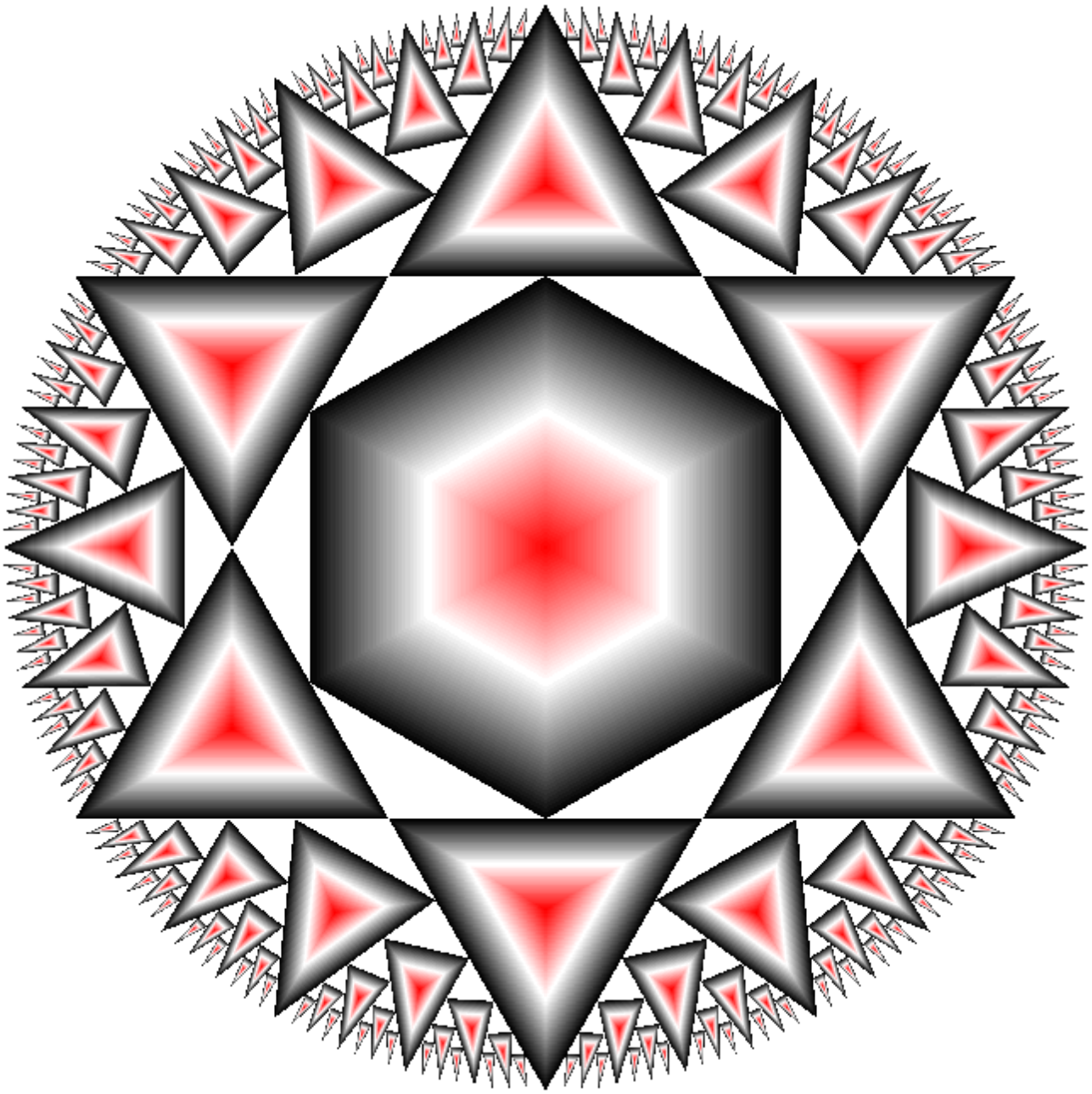
phi = fsolve(phi1, trajectoire) ;
r2 = r1 ; phi2 = phi - floor(phi/(2*%pi))*(2*%pi) ; psi2 = psi1 ;
[X2,Y2,Z2] = cartesienne(r2,phi2,psi2) ;
[cos_theta,v0,p,e,T] = parametre(phi2) ;
graphique(phi2) ;

a = sin(psi2)*sin(phi0 - phi2) ;
b = sin(psi0)*cos(psi2) - cos(psi0)*sin(psi2)*cos(phi0 - phi2) ;
if abs(b) > %eps then, alpha0 = atan(a/b) ; else, alpha0 = %pi/2 ; end
X2u = sin(psi2)*cos(phi2) ; Y2u = sin(psi2)*sin(phi2) ; Z2u = cos(psi2) ;
Xp0u = cos(alpha0)*cos(psi0)*cos(phi0) - sin(alpha0)*sin(phi0) ;
Yp0u = cos(alpha0)*cos(psi0)*sin(phi0) + sin(alpha0)*cos(phi0) ;
Zp0u = - cos(alpha0)*sin(psi0) ;
if (X2u*Xp0u + Y2u*Yp0u + Z2u*Zp0u) < 0 then, alpha0 = alpha0 + %pi ; end
beta0 = %pi/2 - alpha0 ; gamma0 = %pi/2 ;

if cos_theta < 1 - 1D-8 then
  D = p*integrate('sqrt(1-2*e*cos(theta)+e^2)/(1-e*cos(theta))^2', 'theta', 0, acos(cos_theta)) ;
else
  D = alt0 - alt1 ;
end

v0, alpha0, T, D
satellite()
```

Mathématique & Informatique



Mathématique & Informatique

⇨ Les graphes

Un graphe est schématisé par un ensemble de points reliés entre eux par des traits.
Selon la terminologie propre aux graphes, les points sont appelés sommets (ou nœuds) et les traits sont appelés :

- arêtes si on peut aller indifféremment du sommet n_1 au sommet n_2 ou du sommet n_2 au sommet n_1.

  n_1 ○———————○ n_2 (graphe non-orienté)

- arcs si on peut aller du sommet n_1 (initial ou queue) au sommet n_2 (terminal ou tête) dans le sens d'une flèche.

  n_1 ○——>——○ n_2 (graphe orienté)

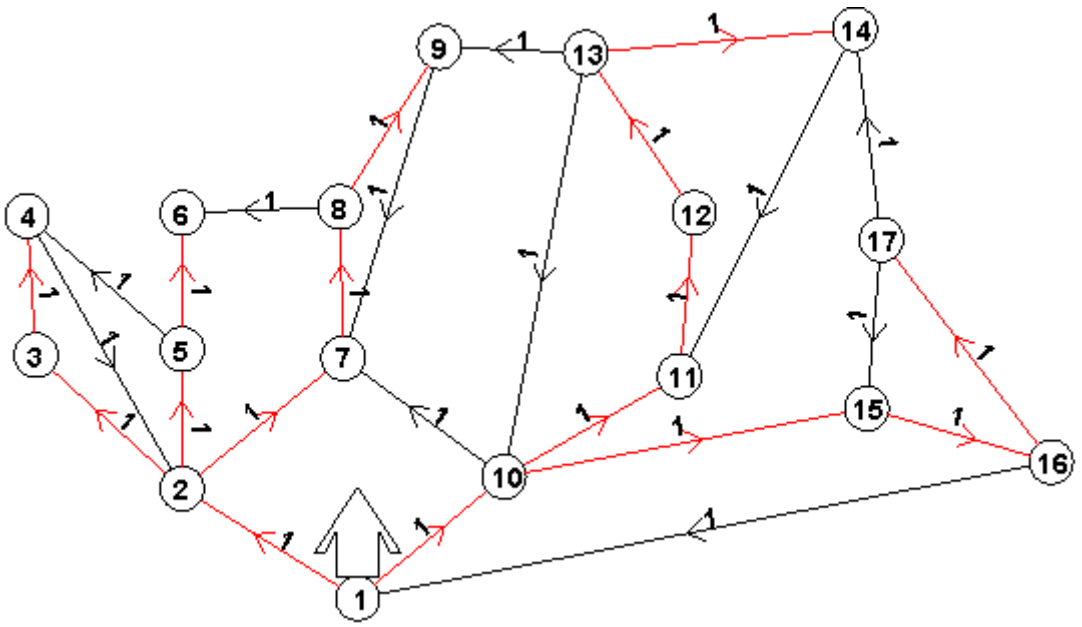
Chaque arc (arête) est souvent affecté d'un nombre p qui représente son "poids". Cette propriété correspond par exemple à une distance, un coût, une contrainte ...

n_1 ○——(p)——>——○ n_2 (n_1 ○——(p)——○ n_2)

Dans Scilab un graphe est une structure à base de listes. Les 2 principales composantes d'un graphe sont les 2 vecteurs lignes nommés "tail" et "head" qui contiennent les numéros de sommet de la queue et de la tête de chaque arc (arête). On ajoute éventuellement un ou plusieurs champs "nodes" et "edges" pour caractériser les propriétés des sommets et des arcs (arêtes). Entrer la commande " – – > help graph_data_structure" pour afficher la liste des éléments constitutifs d'un graphe. Scilab dispose de fonctions permettant de créer un graphe (*make_graph*), de le dessiner (*show_graph*) et de résoudre les problèmes classiques qui lui sont liés (le plus court chemin ≡ *shortest_path* ; l'arbre de poids minimum ≡ *min_weight_tree* ; le flot contraint de coût linéaire minimum ≡ *min_lcost_cflow* ; le voyageur de commerce ≡ *salesman*). Consulter la documentation "Metanet" via le Navigateur d'aide.

Exemple : Recherche de l'arbre de poids minimum (voir algorithmes de **Prim** et de **Kruskal**).

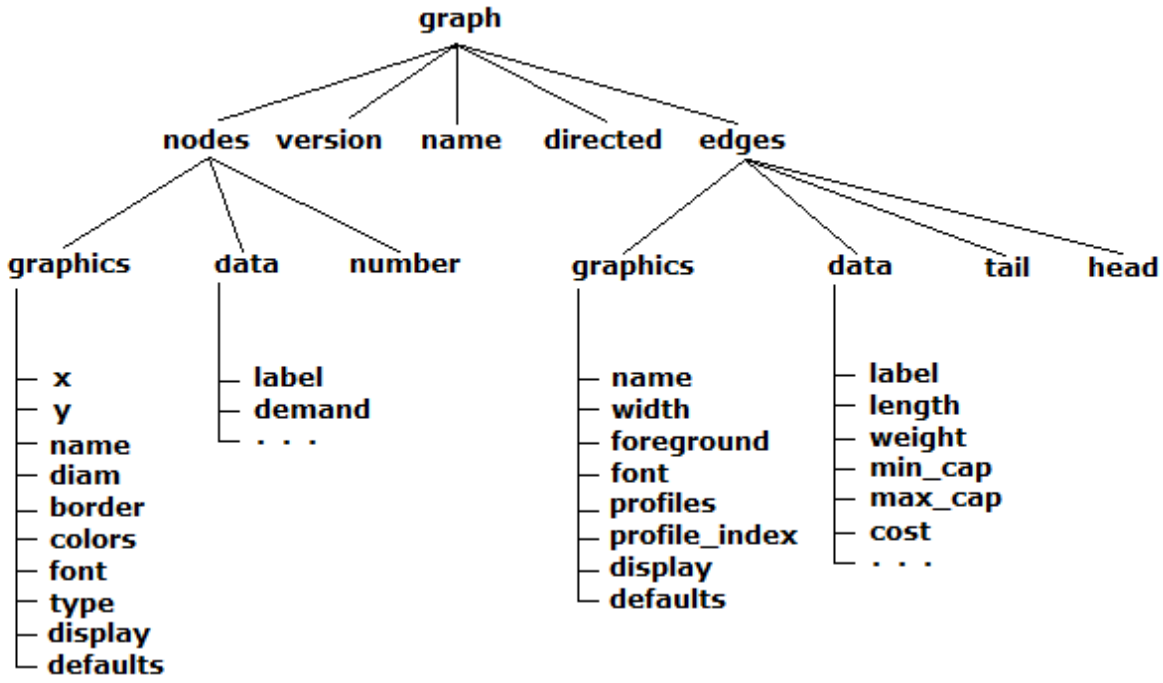
Il s'agit de construire un graphe partiel sans cycle reliant tous les sommets d'un graphe connexe et tel que la somme des poids des arcs soit minimale. Ce problème a de nombreuses applications pratiques. Illustration :



On fait quelquefois "un petit dessin" pour résoudre un problème de graphe. Dans les cas simples, on peut entrevoir la solution ; mais si le graphe possède des centaines de sommets et d'arêtes, le recours à un algorithme est nécessaire.
La difficulté réside également dans le temps d'exécution de l'algorithme, la recherche d'une solution pouvant conduire à une explosion combinatoire.

Mathématique & Informatique

A partir de la version 5 de Scilab, les graphes ont la structure arborescente suivante :



Pour plus de détails consulter la documentation *metanet* fournie par Scilab :

- sur les différents champs constituant un graphe
  graph_data_structure
  nodes_data_structure
  edges_data_structure
  ngraphic_data_structure
  egraphic_data_structure

- sur les fonctions permettant d'ajouter des champs à un graphe
  add_node_data
  add_edge_data

- sur les fonctions permettant de visualiser les champs d'un graphe
  nodedatafields
  edgedatafields

- sur les fonctions permettant de créer et de dessiner un graphe
  make_graph
  show_graph

Les données présentes dans la partie "data" d'un graphe sont utilisées par les fonctions de calcul :

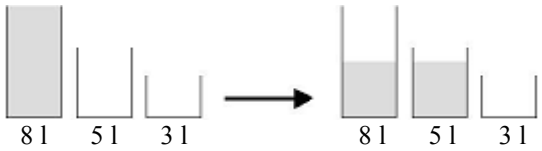
- length ⇒ *shortest_path, salesman*
- weight ⇒ *min_weight_tree*
- max_cap ⇒ *max_cap_path*
- min_cap + max_cap ⇒ *max_flow*
- max_cap + cost ⇒ *min_lcost_cflow*

Mathématique & Informatique

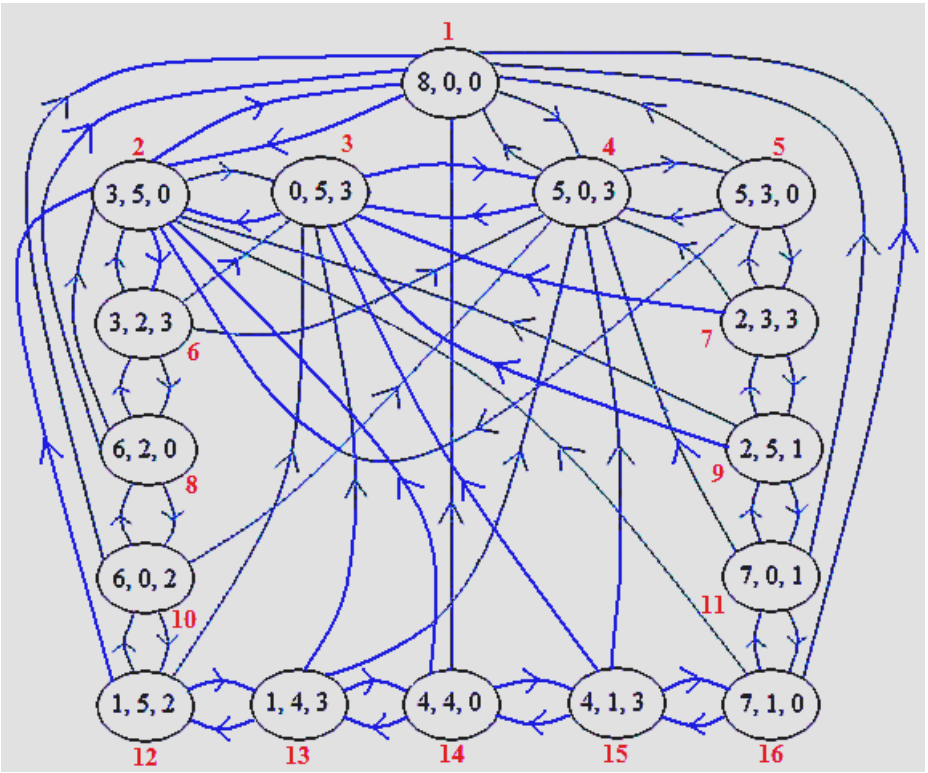
EXEMPLE 1

⇨ Eléments de théorie des graphes (ENSIMAG - Optimisation combinatoire - Michel **Sakarovitch**)

On dispose de 3 récipients pouvant contenir respectivement 8, 5 et 3 litres. Au départ, le récipient de 8 litres est plein et les 2 autres sont vides.
Quel est le nombre minimal de transvasements à effectuer (sans perte de liquide) pour diviser en 2 parties égales le volume initial, c'est-à-dire pour obtenir 4 litres dans chacun des 2 premiers récipients ?



Ce problème peut être modélisé par un graphe où chaque sommet représente l'état des 3 récipients et chaque arc symbolise une opération de transvasement. On obtient un graphe orienté possédant 16 sommets et 58 arcs :



Pour trouver le plus court chemin du sommet (8, 0, 0) au sommet (4, 4, 0), attribuons aux sommets les numéros indiqués en rouge sur le schéma et utilisons la fonction Scilab *shortest_path* :

```
tail = [1 1 2 2 2 3 3 4 4 4 5 5 5 5 6 6 6 6 7 7 7 7 8 8 8 8 9 9 9 9 10 10 10 10 ...
        11 11 11 11 12 12 12 12 13 13 13 13 14 14 14 14 15 15 15 15 16 16 16 16] ;
head = [2 4 1 3 6 2 4 1 3 5 1 2 4 7 2 3 4 8 3 4 5 9 1 2 6 10 2 3 7 11 1 4 8 12 ...
          1 4 9 16 2 3 10 13 3 4 12 14 1 2 13 15 3 4 14 16 1 2 11 15] ;
graph = make_graph('Example 1', 1, 16, tail, head) ;
[p, lp] = shortest_path(1, 14, graph, 'arc') ;
arc = [graph.edges.tail(p) ; graph.edges.head(p)]
```

Les colonnes de la matrice "arc" sont les couples de sommets définissant les arcs de la solution :

1.	2.	6.	8.	10.	12.	13.
2.	6.	8.	10.	12.	13.	14.

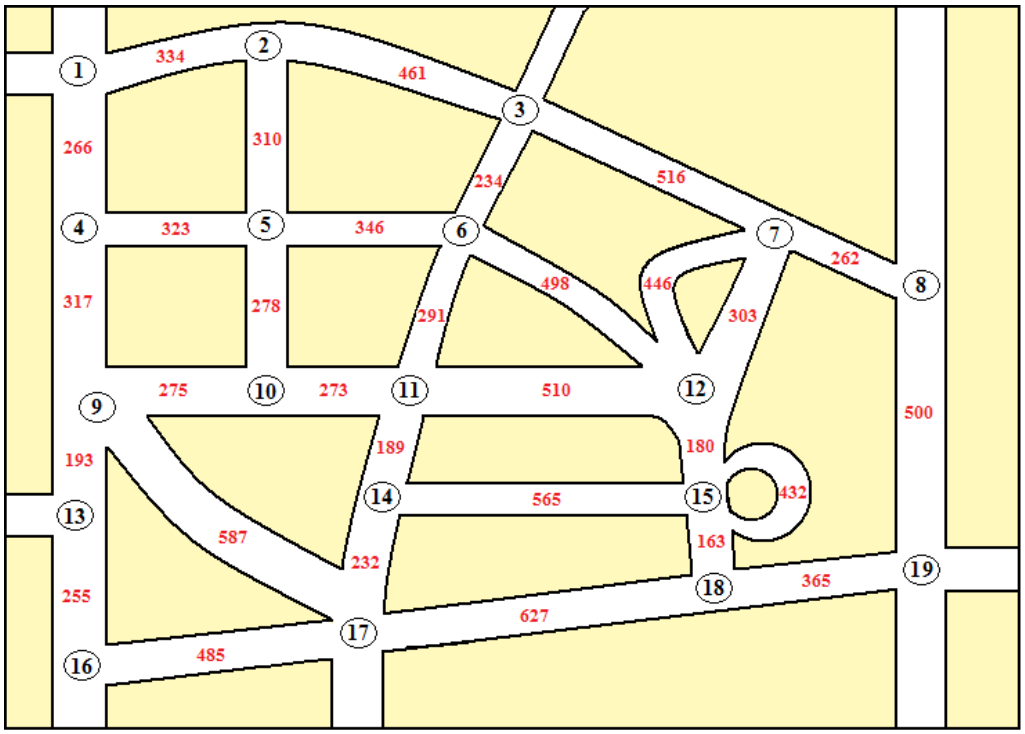
Résultat équivalent aux 7 transvasements successifs suivants :

(8, 0, 0) → (3, 5, 0) → (3, 2, 3) → (6, 2, 0) → (6, 0, 2) → (1, 5, 2) → (1, 4, 3) → (4, 4, 0)

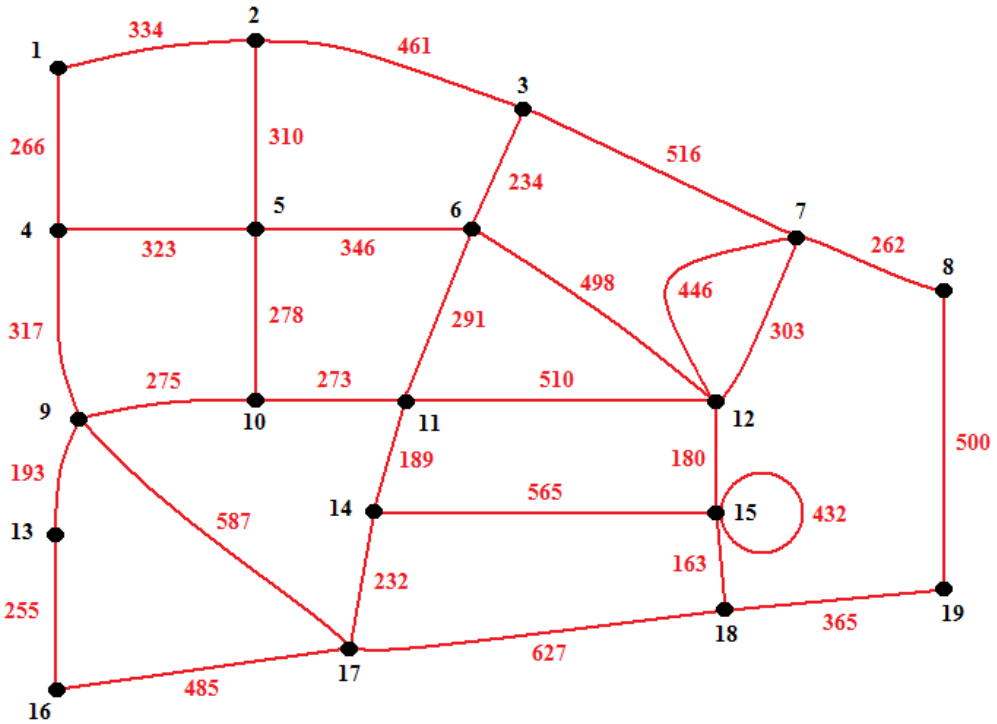
Mathématique & Informatique

EXEMPLE 2

Dans un quartier de centre ville, on souhaite réaliser un réseau de canalisations entérrées qui relieront entre eux tous les carrefours, numérotés de 1 à 19, en empruntant obligatoirement les voies publiques de circulation existantes.
Les distances (en mètres) entre les carrefours figurent en rouge sur le plan ci-dessous.
On impose de minimiser la longueur totale des tranchées à creuser afin de réduire le coût des travaux.



Associons au plan un graphe non-orienté à 19 sommets et 31 arêtes dont les poids sont les distances entre les carrefours.
On recherche l'arbre couvrant de poids minimum.



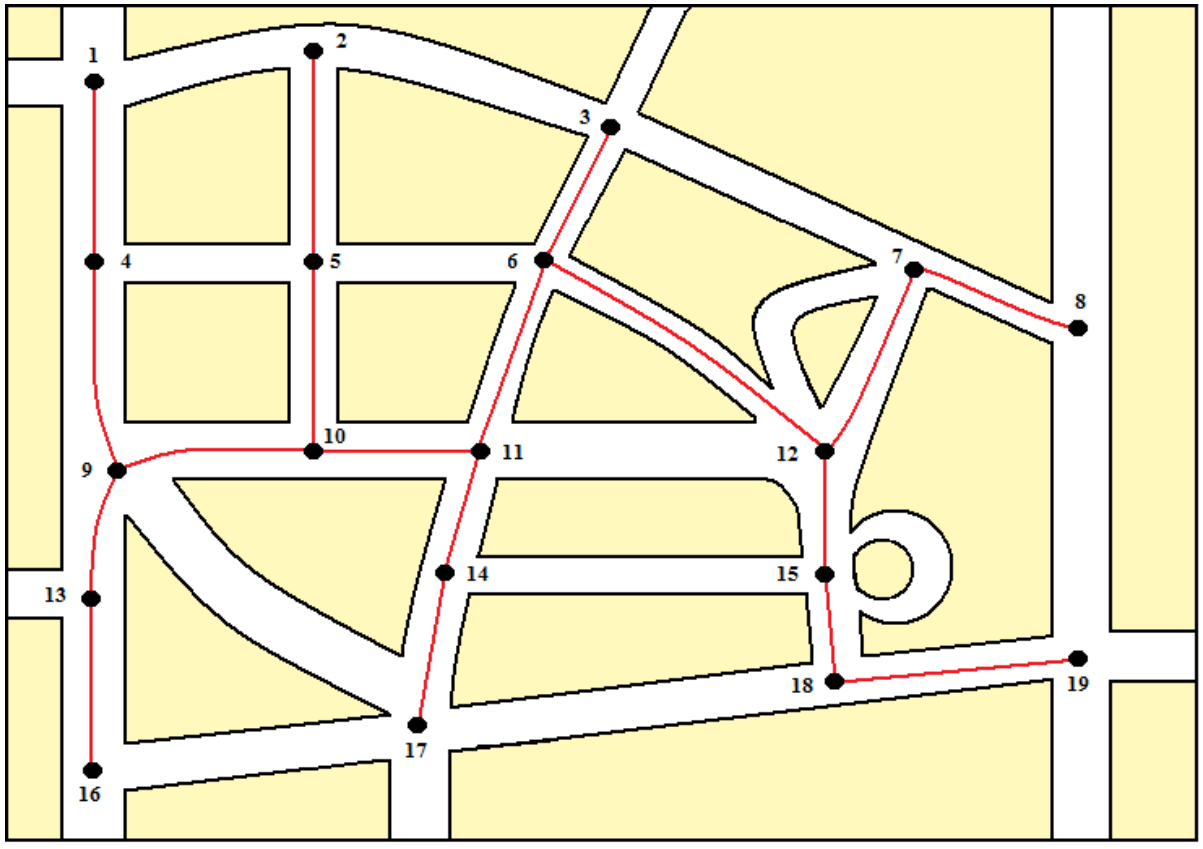
Utilisons la fonction Scilab *min_weight_tree* dans le programme suivant :

```
tail = [1 1 2 2 3 3 4 4 5 5 6 6 7 7 7 8 9 9 9 10 11 11 12 13 14 14 15 15 16 17 18] ;
head = [2 4 3 5 6 7 5 9 6 10 11 12 8 12 12 19 10 13 17 11 12 14 15 16 15 17 15 18 17 18 19] ;
g = make_graph('Example 2', 0, 19, tail, head) ;
w = [334 266 461 310 234 516 323 317 346 278 291 498 262 303 446 ...
     500 275 193 587 273 510 189 180 255 565 232 432 163 485 627 365] ;
g = add_edge_data(g,'weight',w) ;
t = min_weight_tree(g) ;
edge = [g.edges.tail(t) ; g.edges.head(t) ; g.edges.data.weight(t)]
weight = sum(edge(3, :))
```

Les colonnes de la matrice "edge" indiquent les numéros des carrefours entre lesquels il faudra creuser les tranchées :

```
  2.   3.   1.   5.   6.   7.   7.   4.   9.  10.   6.   9.  11.  12.  13.  14.  15.  18.
  5.   6.   4.  10.  11.  12.   8.   9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19.
310. 234. 266. 278. 291. 303. 262. 317. 275. 273. 498. 193. 189. 180. 255. 232. 163. 365.
```

On déduit le plan des travaux à réaliser :



La longueur totale des tranchées (variable "weight") est de 4 884 mètres.

Remarque :
Pour trouver l'arbre couvrant de poids maximum, on écrit : Max f(x) = - Min [- f(x)] ⇒ Max $\sum p_i$ = - Min $\sum(- p_i)$
Il suffit donc d'utiliser la fonction *min_weight_tree* en changeant le signe des poids et le signe du résultat.
On obtient une nouvelle matrice "edge" :

```
  1.   2.   4.   5.   6.   3.   8.   4.   5.  11.   7.  13.  14.  14.  16.   9.  17.  18.
  2.   3.   5.   6.  12.   7.  19.   9.  10.  12.  12.  16.  17.  15.  17.  17.  18.  19.
334. 461. 323. 346. 498. 516. 500. 317. 278. 510. 446. 255. 232. 565. 485. 587. 627. 365.
```

Le poids de l'arbre correspondant est de 7 645 mètres.

EXEMPLE 3

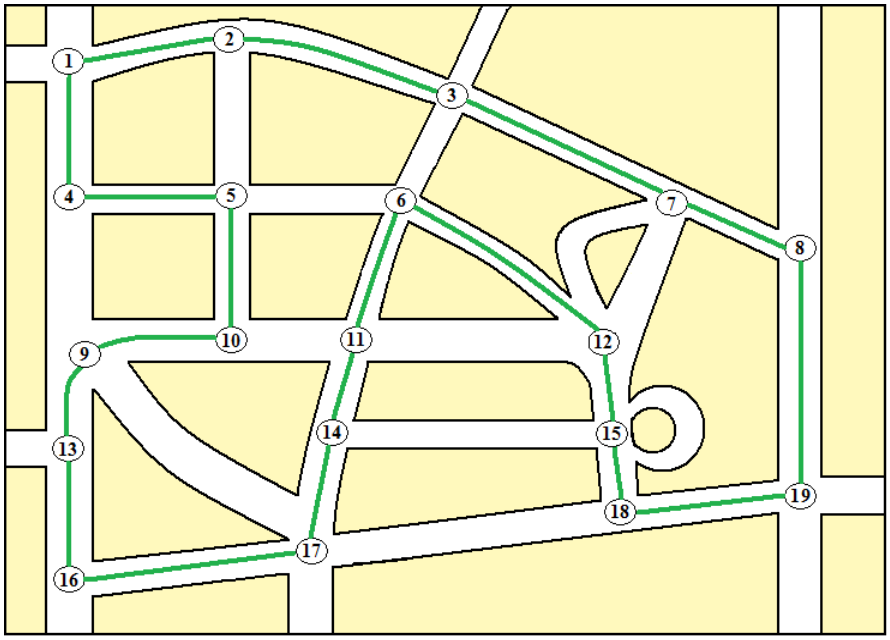
Reprenons l'exemple précédent du centre ville.
Un promeneur se demande s'il existe un itinéraire circulaire (c'est-à-dire le ramenant à son point de départ) qui lui permette de passer une fois et une seule par tous les carrefours en effectuant le trajet le plus court possible.
Pour résoudre ce problème, dit du "voyageur de commerce", on fait appel à la fonction Scilab *salesman* :

```
t0 = [1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 9 9 9 10 11 11 12 13 14 14 15 16 17 18] ;
h0 = [2 4 3 5 6 7 5 9 6 10 11 12 8 12 19 10 13 17 11 12 14 15 16 15 17 18 17 18 19] ;
l0 = [334 266 461 310 234 516 323 317 346 278 291 498 262 303  ...
      500 275 193 587 273 510 189 180 255 565 232 163 485 627 365] ;
t1 = [t0 h0] ; h1 = [h0 t0] ; l1 = [l0 l0] ;
g1 = make_graph('Example 3', 1, 19, t1, h1) ;
g1 = add_edge_data(g1, 'length', l1) ;
c = salesman(g1) ;
arc = [g1.edges.tail(c) ; g1.edges.head(c) ; g1.edges.data.length(c)]
distance = sum(arc(3, :))
```

On obtient la matrice "arc" :

```
  1.   4.   5.  10.   9.  13.  16.  17.  14.  11.   6.  12.  15.  18.  19.   8.   7.   3.   2.
  4.   5.  10.   9.  13.  16.  17.  14.  11.   6.  12.  15.  18.  19.   8.   7.   3.   2.   1.
266. 323. 278. 275. 193. 255. 485. 232. 189. 291. 498. 180. 163. 365. 500. 262. 516. 461. 334.
```

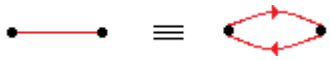
Ses colonnes correspondent au circuit :



La distance parcourue par le promeneur est de 6 066 mètres.

Remarque :
La fonction *salesman* ne s'applique qu'à un graphe simple orienté. C'est pourquoi :

- Si plusieurs arêtes (ou arcs de même orientation) relient 2 sommets, on ne garde que l'arête (l'arc) de longueur minimale pour définir les vecteurs "tail" et "head". On élimine aussi les boucles sur les sommets.
  Il est possible d'écrire une fonction *graph_simp_min* qui effectue automatiquement cette transformation.
  Dans notre exemple, on supprime les arêtes (7, 12, 446) et (15, 15, 432).
- On construit un graphe orienté g1, à partir du graphe non-orienté g, en transformant les arêtes en arcs :



FONCTION ***graph_simp_min***

Cette fonction crée un graphe simple de longueur minimum "gsm", à partir d'un graphe quelconque "g".

gsm = graph_simp_min(g, oriented)

L'algorithme de criblage est le suivant : toute boucle sur un sommet de "g" est enlevée et seul l'arc (l'arête) de longueur minimale reliant 2 sommets est conservée.
Le graphe "gsm" est orienté si le paramètre "oriented" est égal à 1 et non-orienté s'il est égal à 0.

Programme Scilab :

```
function gsm = graph_simp_min(g, oriented)
  t = g.edges.tail , h = g.edges.head , l = g.edges.data.length
  n = length(t) , choice = ones(1,n)
  for i = 1:n
    if t(i) == h(i) then
      choice(i) = 0
    end
  end
  for i = 1:n
    if choice(i) == 1 then
      for j = 1:n
        if choice(j) == 1 then
          bool1 = (j <> i) & (l(j) >= l(i))
          bool2 = (t(j) == t(i)) & (h(j) == h(i))
          bool3 = (oriented == 0) & (t(j) == h(i)) & (h(j) == t(i))
          if bool1 & (bool2 | bool3) then
            choice(j) = 0
          end
        end
      end
    end
  end
  nsm = sum(choice) , tsm = zeros(1,nsm) , hsm = zeros(1,nsm) , lsm = zeros(1,nsm)
  j = 0
  for i = 1:n
    if choice(i) == 1 then
      j = j + 1
      tsm(j) = t(i) , hsm(j) = h(i) , lsm(j) = l(i)
    end
  end
  gsm = make_graph(g.name, oriented, node_number(g), tsm, hsm)
  gsm = add_edge_data(gsm, 'length', lsm)
endfunction
```

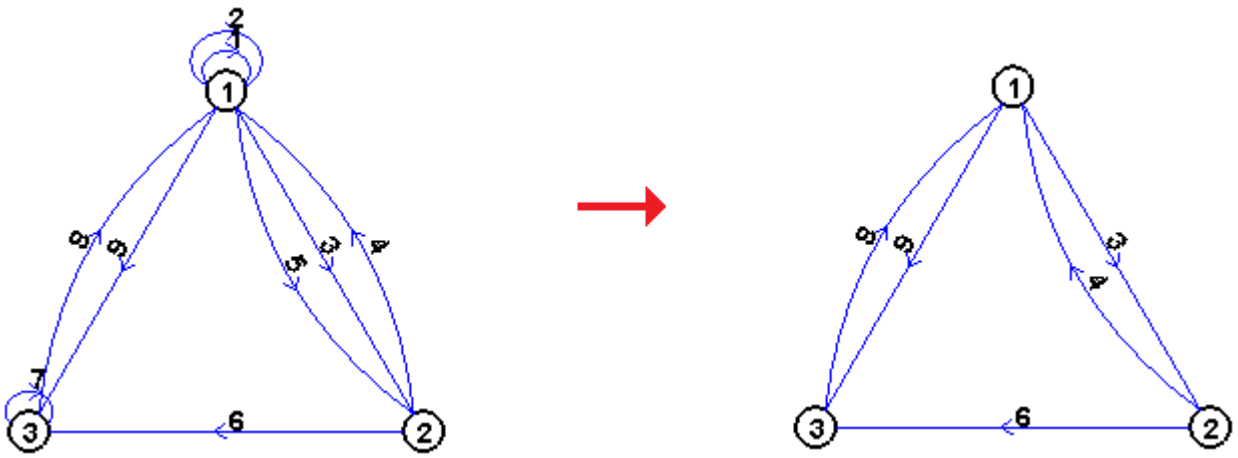
NB
Cette fonction permet d'utiliser les fonctions Scilab *shortest_path* et *salesman* avec un plus petit nombre d'arcs, donc avec un nombre réduit de possibilités à envisager dans la recherche de la solution.

Mathématique & Informatique

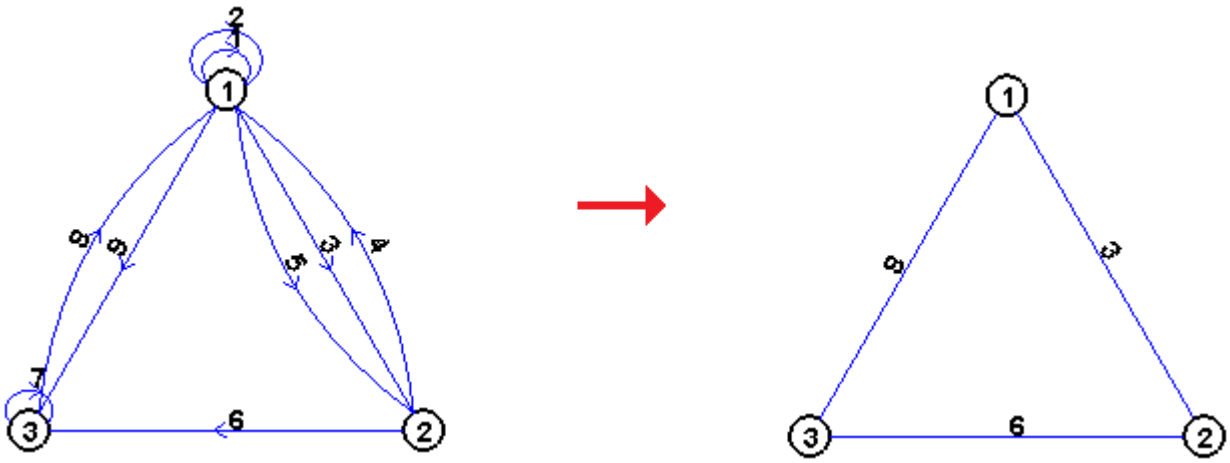
Exemple :

Soit le graphe définit par les vecteurs : t = [1 1 1 1 1 2 2 3 3] , h = [1 1 2 2 3 1 3 1 3] , l = [1 2 3 5 9 4 6 8 7]
Appliquons *graph_simp_min* :

- Cas orienté

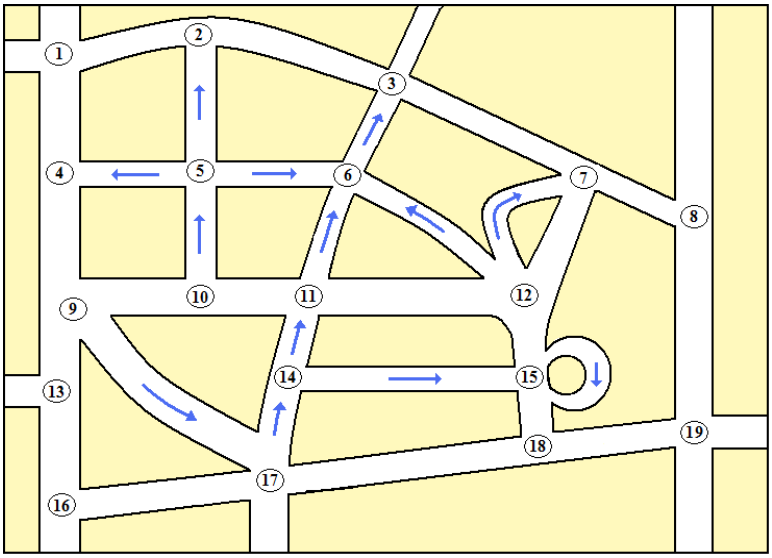


- Cas non-orienté



EXEMPLE 4

La circulation automobile en centre ville est réglementée par des sens interdits. Sur le plan, une flèche bleue indique le sens de circulation et l'absence de flèche signifie que l'on peut circuler dans les 2 sens.



Le graphe correspondant est orienté : 2 sommets sont reliés par 1 arc i ●—▶—● j si la circulation se fait uniquement de i vers j, ou par 2 arcs i ●—▶—● j et i ●—◀—● j si la circulation se fait de i vers j et de j vers i.
Vérifier que le centre ville est "circulable", autrement dit que l'on peut toujours aller d'un carrefour à un autre.
Quels sont les 2 carrefours les plus éloignés ?
Pour effectuer son circuit, le promeneur de l'exemple 3 décide de prendre un vélo et de respecter les sens interdits.
Quel doit être son parcours ?

```
tail = [1 1 2 2 3 3 4 4 5 5 5 6 7 7 7 8 8 9 9 9 9 10 10 10 11 11 11 ...
        12 12 12 12 13 13 14 14 15 15 16 16 17 17 17 18 18 18 19 19] ;
head = [2 4 1 3 2 7 1 9 2 4 6 3 3 8 12 7 19 4 10 13 17 5 9 11 6 10 12 ...
         6 7 11 15 9 16 11 15 12 18 13 17 14 16 18 15 17 19 8 18] ;
g2 = make_graph('Example 4', 1, 19, tail, head) ;
l = [334 266 334 461 461 516 266 317 310 323 346 234 516 262 303 262 500 317 275 193 587 278 275 273 ...
     291 273 510 498 303 510 180 193 255 189 565 180 163 255 485 232 485 627 163 627 365 500 365] ;
g2 = add_edge_data(g2, 'length', l) ;
// ------------------------------------------------------------------------------------------------------------
connexity = %t ; max_lp = 0 ;
for i = 1:node_number(g2)
  for j = 1:node_number(g2)
    if i <> j then
      [p, lp] = shortest_path(i, j, g2, 'length') ;
      connexity = connexity & (p <> []) ;
      if lp > max_lp then
        max_i = i ; max_j = j ; max_p = p ; max_lp = lp ;
      end
    end
  end
end
connexity , max_i , max_j , max_lp
path = [g2.edges.tail(max_p) ; g2.edges.head(max_p) ; g2.edges.data.length(max_p)]
// ------------------------------------------------------------------------------------------------------------
c = salesman(g2) ;
cycle = [g2.edges.tail(c) ; g2.edges.head(c) ; g2.edges.data.length(c)]
distance = sum(cycle(3, :))
```

Résultats :

connexity = T
max_i = 6.
max_j = 14.
max_lp = 2255.
path =

```
     6.   3.   7.  12.  15.  18.  17.
     3.   7.  12.  15.  18.  17.  14.
   234. 516. 303. 180. 163. 627. 232.
```

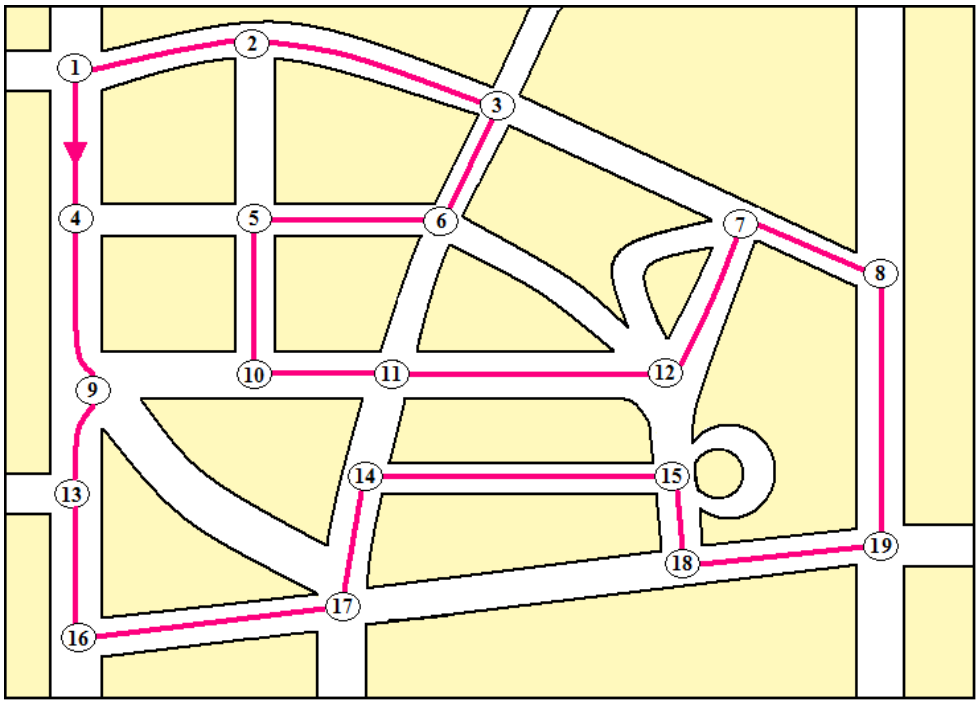
cycle =

```
     1.   4.   9.  13.  16.  17.  14.  15.  18.  19.   8.   7.  12.  11.  10.   5.   6.   3.   2.
     4.   9.  13.  16.  17.  14.  15.  18.  19.   8.   7.  12.  11.  10.   5.   6.   3.   2.   1.
   266. 317. 193. 255. 485. 232. 565. 163. 365. 500. 262. 303. 510. 273. 278. 346. 234. 461. 334.
```

distance = 6342.

Conclusion :

Le booléen "connexity" est vrai (T = true), donc le graphe est fortement connexe et le centre ville est "circulable".
On peut obtenir directement cette propriété avec l'instruction *[nc, ncomp] = strong_connex(g2)* qui renvoie une série de 1 prouvant que le graphe ne possède qu'une seule composante fortement connexe.
Les 2 carrefours les plus éloignés sont le n° 6 et le n° 14 ; le trajet minimal pour aller du carrefour n° 6 au carrefour n° 14 est de 2 255 mètres, il emprunte les carrefours successifs : $6 \to 3 \to 7 \to 12 \to 15 \to 18 \to 17 \to 14$
Le cycliste qui veut effectuer la boucle de longueur minimale doit traverser les 19 carrefours dans l'ordre suivant :
$1 \to 4 \to 9 \to 13 \to 16 \to 17 \to 14 \to 15 \to 18 \to 19 \to 8 \to 7 \to 12 \to 11 \to 10 \to 5 \to 6 \to 3 \to 2 \to 1$
La distance parcourue est de 6 342 mètres.



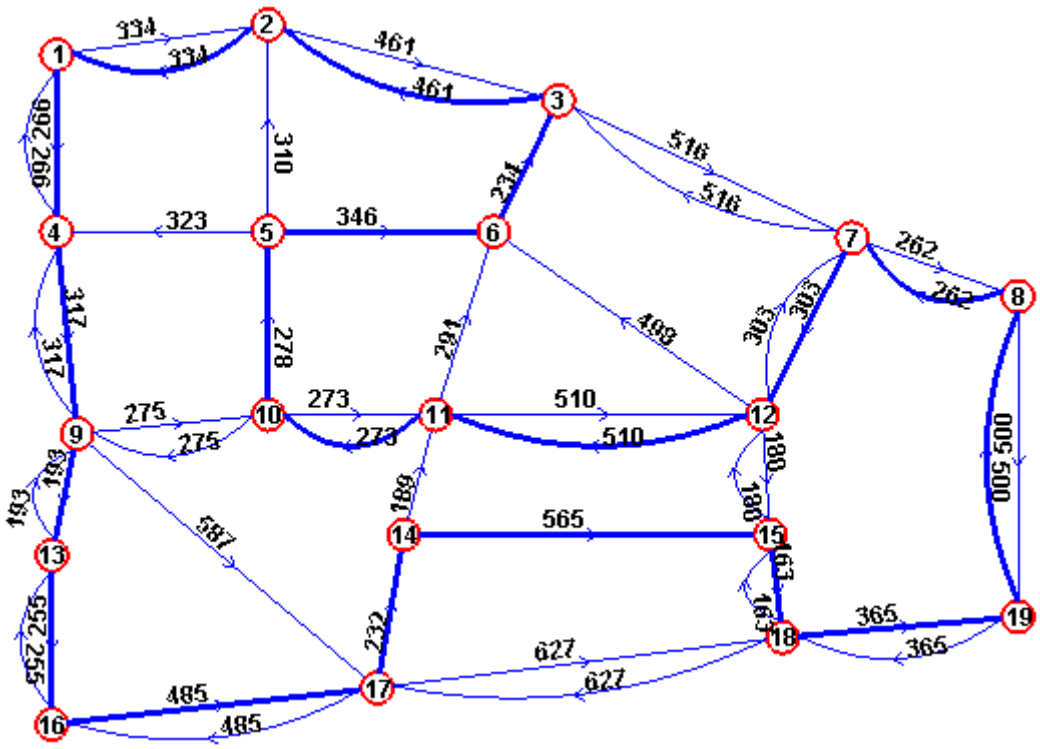
Remarque :
Le problème du voyageur de commerce (PVC) consiste à rechercher un cycle Hamiltonien de poids minimum.
Le PVC est difficile dans le cas général car on ne connait pas d'algorithme s'exécutant en un temps polynomial qui donne la solution optimale (problème NP-complet).
Actuellement, on peut le résoudre pour des graphes possédant jusqu'à 100 000 sommets.
Un exemple : en 2006, on a obtenu la solution exacte pour un graphe à 85 900 sommets représentant le tracé d'un circuit intégré VLSI (Calcul parallèle avec 128 machines - Temps total CPU cumulé = 136 années).
⇨ http://www.tsp.gatech.edu/index.html

Mathématique & Informatique

DESSIN D'UN GRAPHE AVEC SCILAB

Soit g un graphe, sa représentation graphique est définie essentiellement par les 2 vecteurs *g.nodes.graphics.x* et *g.nodes.graphics.y* qui contiennent respectivement les abscisses et les ordonnées des différents sommets (nœuds) dans l'ordre de leur numérotation.
En pratique, dans l'exemple du centre ville, on peut importer le plan dans un logiciel de dessin, puis pointer chaque carrefour avec la souris pour lire ses coordonnées (x,y) en pixels.

```
tail = [1 1 2 2 3 3 4 4 5 5 5 6 7 7 7 8 8 9 9 9 9 10 10 10 11 11 11 ...
        12 12 12 12 13 13 14 14 15 15 16 16 17 17 17 18 18 18 19 19] ;
head = [2 4 1 3 2 7 1 9 2 4 6 3 3 8 12 7 19 4 10 13 17 5 9 11 6 10 12 ...
         6 7 11 15 9 16 11 15 12 18 13 17 14 16 18 15 17 19 8 18] ;
g2 = make_graph('Example 4', 1, 19, tail, head) ;
// ------------------------------------------------------------------------------------------------------------------------
l = [334 266 334 461 461 516 266 317 310 323 346 234 516 262 303 262 500 317 275 193 587 278 275 273 ...
     291 273 510 498 303 510 180 193 255 189 565 180 163 255 485 232 485 627 163 627 365 500 365] ;
g2 = add_edge_data(g2, 'length', l) ;
g2.nodes.graphics.x = + [53 187 371 53 187 330 557 662 66 187 293 500 50 273 505 50 256 513 662] ;
g2.nodes.graphics.y = - [46 27 75 158 158 158 162 199 286 274 274 274 363 350 350 470 447 415 402] ;
N = ones(1,node_number(g2)) ; A = ones(1,arc_number(g2)) ;
g2.nodes.graphics.colors = [5;0]*N ; g2.nodes.graphics.border = 2*N ;
g2.nodes.graphics.diam = 20*N ; g2.nodes.graphics.font = [2;0;0]*N ;
g2.edges.graphics.foreground = 2*A ; g2.edges.graphics.width = 1*A ;
g2.edges.graphics.font = [2;0;0]*A ;
g2.nodes.graphics.display = 'number' ; g2.edges.graphics.display = 'length' ;
show_graph(g2) ;
// ------------------------------------------------------------------------------------------------------------------------
c = salesman(g2) ;
for i = 1:length(c) , xpause(1e6) ; hilite_edges(c(i)) ; end
```



Note : Si les champs "display" d'un graphe ne sont pas renseignés, alors seuls les sommets et les arcs du graphe sont dessinés. Pour afficher en plus les numéros des sommets et les longueurs des arcs, il faut ouvrir le menu "View" de la fenêtre graphique Metanet, puis sélectionner "Options" et enfin cliquer sur les boutons "number" , "length" et "OK".

EXEMPLE 5

Un problème de flot.
Un réseau de distribution d'eau est composé d'une station émettrice, de stations intermédiaires, et d'une station réceptrice. Les différentes stations sont reliées entre elles par des canalisations. On souhaite envoyer le maximum de fluide de la station émettrice vers la station réceptrice en sachant que chaque canalisation ne peut en acheminer qu'une certaine quantité, appelée capacité de la canalisation. Pour simplifier, on considère qu'il n'y a pas de fuite dans le réseau, c'est-à-dire qu'au niveau d'une station intermédiaire, il entre autant de fluide qu'il en sort. Rien ne se perd, le flot est dit conservatif.
Ce problème peut être modélisé de 2 manières :

1) par un GRAPHE :

- chaque sommet i est une station.
- chaque arc (i,j) est une canalisation pouvant acheminer du fluide de i vers j.
  Son poids correspond à sa capacité $c_{ij} \geq 0$ donnée.
- Le flot circulant d'une station i à une station j est une variable inconnue f_{ij} obéïssant à la contrainte : $0 \leq f_{ij} \leq c_{ij}$

Convenons que la station émettrice (Source) est le sommet 1 et que la station réceptrice (Puits) est le sommet n.
On cherche à maximiser la quantité :

$$\sum_j f_{1j} = \sum_i f_{in}$$

La fonction Scilab *max_flow* calcule les f_{ij} pour des c_{ij} données.

2) par un PROGRAMME LINÉAIRE :

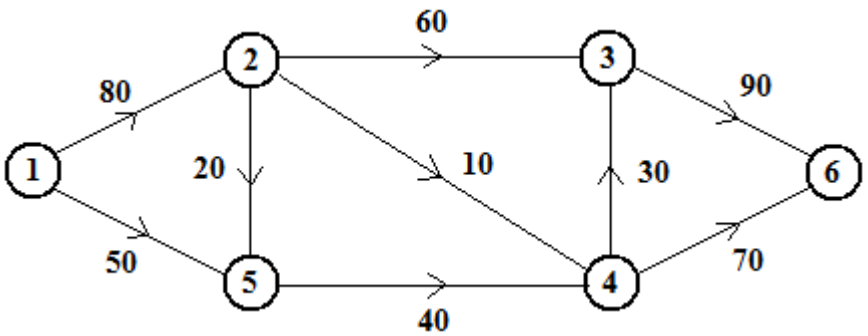
Avec les notations adoptées, on écrit le système constitué de la fonction à maximiser et des contraintes :

$$\begin{cases} \sum_j f_{1j} = \sum_i f_{in} = F(Max) \\ \sum_j f_{ji} - \sum_j f_{ij} = 0 \qquad \forall\, i \neq 1,\ i \neq n \\ 0 \leq f_{ij} \leq c_{ij} \end{cases}$$

La fonction Scilab *linpro* calcule les f_{ij} pour des c_{ij} données.

Application numérique.

Réseau



c12 = 80 ; c15 = 50 ; c23 = 60 ; c24 = 10 ; c25 = 20 ; c36 = 90 ; c43 = 30 ; c46 = 70 ; c54 = 40.

Mathématique & Informatique

Programme

On note F la valeur du flot maximum et f le vecteur des valeurs des flots sur les arcs ($f = [f_{ij}]$).
Pour utiliser la fonction *linpro*, on minimise - F et on inverse le résultat.

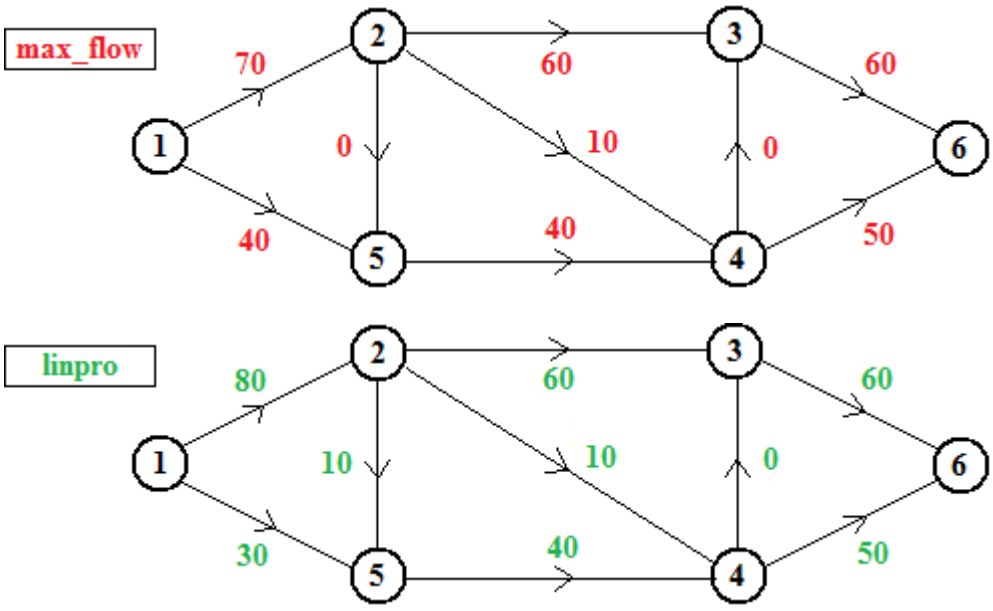
```
// ------------------- DATA
c12 = 80 ; c15 = 50 ; c23 = 60 ; c24 = 10 ; c25 = 20 ; c36 = 90 ; c43 = 30 ; c46 = 70 ; c54 = 40 ;

// ------------------- GRAPH
// Tail
t = [1 1 2 2 2 3 4 4 5] ;
// Head
h = [2 5 3 4 5 6 3 6 4] ;
// Graph
g = make_graph('Example 5', 1, 6, t, h) ;
g = add_edge_data(g, 'min_cap', zeros(1,9)) ;
g = add_edge_data(g, 'max_cap', [c12 c15 c23 c24 c25 c36 c43 c46 c54]) ;
// Calling max_flow
source = 1 ; sink = 6 ;
[F, f, flag] = max_flow(source, sink, g)

// ------------------- LINEAR PROGRAM
// Equality constraints
C1 = [1 0 -1 -1 -1 0 0 0 0 ; 0 0 1 0 0 -1 1 0 0 ; 0 0 0 1 0 0 -1 -1 1 ; 0 1 0 0 1 0 0 0 -1 ; 1 1 0 0 0 -1 0 -1 0] ;
b1 = zeros(5,1) ;
// Inequality constraints
C2 = [] ;
b2 = [] ;
// Bound constraints
ci = zeros(9,1) ;
cs = [c12 ; c15 ; c23 ; c24 ; c25 ; c36 ; c43 ; c46 ; c54] ;
// Calling linpro
C = [C1 ; C2] ; b = [b1 ; b2] ; me = size(C1,'r') ; f0 = 'v' ;
p = - [1 ; 1 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0 ; 0] ;
[f, lagr, F] = linpro(p, C, b, ci, cs, me, f0)
```

Solution

Les 2 fonctions donnent la même valeur du flot maximum ($F = 110$), mais une répartition différente des flots sur les arcs (valeurs en rouge pour max_flow, valeurs en vert pour linpro).



⇨ La programmation linéaire

Scilab fournit la fonction *linpro*, basée sur une factorisation de **Cholesky**, qui permet de minimiser une fonction linéaire $f(x) = c\,x$ de $\mathbb{R}^n \rightarrow \mathbb{R}$, sous un ensemble de contraintes de la variable x.

La forme générale de ce type de problème est :

$$\begin{cases} c\,x = f(\text{Min}) & \text{fonction à minimiser} \\ A_1\,x = b_1 & \text{contraintes d'égalité} \\ A_2\,x \leq b_2 & \text{contraintes d'inégalité} \\ m \leq x \leq M & \text{contraintes de bornes} \end{cases}$$

Le vecteur x de dimension (n,1) est l'inconnue et les matrices c, A_1, b_1, A_2, b_2, m, M, de dimensions respectives (1,n), (ne,n), (ne,1), (ni,n), (ni,1), (n,1), (n,1) sont les données.

Définissons les paramètres suivants :

$p_1 = c'$ (transposée du vecteur c = vecteur colonne)
$p_2 = [A_1 ; A_2]$
$p_3 = [b_1 ; b_2]$
$p_4 = m$
$p_5 = M$
$p_6 = \text{size}(A_1,'r')$ (nombre de lignes de A_1 = nombre d'égalités)
$p_7 = 'v'$ (valeur initiale de x calculée par l'algorithme)

L'appel à *linpro* s'écrit :

[x, lagr, f] = linpro(p_1 , p_2 , p_3 , p_4 , p_5 , p_6 , p_7)

Le résultat est le vecteur optimal **x** pour lequel la fonction prend la valeur minimum f_{min} = **f**
La variable **lagr** est le vecteur de **Lagrange** associé aux contraintes (propre à l'algorithme).

Important :

- Les contraintes inutiles sont remplacées par des matrices vides, c'est-à-dire []
- Lorsque le problème consiste à maximiser une fonction f, on se ramène au cas ci-dessus en minimisant - f et en inversant le résultat : $p_1 = -c' \Rightarrow f_{max} = -f$
- Dans le cas où la fonction f à optimiser n'est pas linéaire mais quadratique linéaire, d'expression :

$$f(x) = \frac{1}{2}\,{}^t x\,Q\,x + {}^t p\,x = \sum_i \frac{1}{2} q_{ii} x_i^2 + \sum_{i<j} q_{ij} x_i x_j + \sum_i p_i x_i$$

Q = $[q_{ij}]$ est une matrice carrée (n,n) symétrique réelle
p = $[p_i]$ est un vecteur colonne (n,1) réel

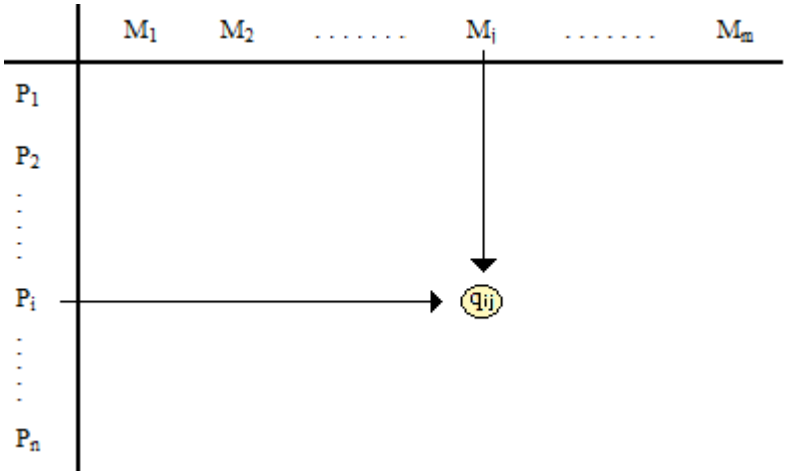
on utilise la fonction Scilab *quapro* :
[x, lagr, f] = quapro(p_1 , p_2 , p_3 , p_4 , p_5 , p_6 , p_7 , p_8)
avec les paramètres :
$p_1 = Q$; $p_2 = p$; $p_3 = [A_1 ; A_2]$; $p_4 = [b_1 ; b_2]$; $p_5 = m$; $p_6 = M$; $p_7 = \text{size}(A_1,'r')$; $p_8 = 'v'$

En posant Z = zeros(n,n), on a :
linpro(p_1 , p_2 , p_3 , p_4 , p_5 , p_6 , p_7) ≡ quapro(Z, p_1 , p_2 , p_3 , p_4 , p_5 , p_6 , p_7)

NB.
Suivant les auteurs, la transposée de A est notée : A', ${}^tA$ ou ${}^TA$ (t à gauche pour les Français), A^t ou A^T (t à droite pour les Anglais).

EXEMPLE 1 : UN PROBLÈME DE PRODUCTION

Une usine fabrique n produits P_1 , P_2 , ... , P_n à partir de m matières premières M_1 , M_2 , ... , M_m .
Il faut une quantité q_{ij} de matière première M_j pour fabriquer le produit P_i .
Le stock de matière première M_j est s_j . Le gain réalisé par la vente du produit P_i est g_i .
Quelle production doit être envisagée par la direction pour obtenir le profit maximum ?



Soit x_i le nombre de produits P_i à fabriquer. On veut que le profit global soit maximum, tout en respectant les contraintes sur les quantités de matières premières nécessaires à la fabrication et sur les stocks disponibles.
Le problème revient à résoudre le programme linéaire suivant :

$$\begin{cases} \sum_i g_i \, x_i \;=\; f(Max) \\ \sum_i q_{ij} \, x_i \;\le\; s_j \qquad \forall \; j = 1, \ldots, m \\ x_i \ge 0 \end{cases}$$

Application numérique : 4 produits à partir de 5 matières premières.

Quantités de matières premières nécessaires à la fabrication des produits :

	M_1	M_2	M_3	M_4	M_5
P_1	2	1	1	3	2
P_2	2	2	3	0	1
P_3	1	3	2	1	2
P_4	1	3	0	3	1

Stocks :

M_1	M_2	M_3	M_4	M_5
130	152	136	129	143

Gains unitaires :

P_1	P_2	P_3	P_4
5	6	4	3

Programme :

```
// ------ Matrices
c   = [5 6 4 3] ;
A1 = [] ;
b1 = [] ;
A2 = [2 2 1 1 ; 1 2 3 3 ; 1 3 2 0 ; 3 0 1 3 ; 2 1 2 1] ;
b2 = [130 ; 152 ; 136 ; 129 ; 143] ;
m   = [0 ; 0 ; 0 ; 0] ;
M   = [] ;

// ------ Résolution
p1 = - c' ; p2 = [A1;A2] ; p3 = [b1;b2] ; p4 = m ; p5 = M ; p6 = size(A1,'r') ; p7 = 'v' ;
[x, lagr, f] = linpro(p1, p2, p3, p4, p5, p6, p7)
```

Résultat :

$x_1 = 30.571429$; $x_2 = 21.285714$; $x_3 = 20.785714$; $x_4 = 5.5$
f = - 380.21429

La fonction *linpro* donne le vecteur réel x pour lequel la fonction prend sa valeur optimale.
Le nombre de produits à fabriquer devant être un entier, on prend la valeur entière par défaut des x_i , donc on envisage de fabriquer 30 produits P_1 , 21 produits P_2 , 20 produits P_3 , et 5 produits P_4 pour obtenir un gain g "proche" du maximum de la fonction : $g = 5\times30 + 6\times21 + 4\times20 + 3\times5 = 371$
Existe t'il une meilleure solution entière ?
Il faut la rechercher parmi les valeurs entières du gain comprises entre 372 et 380. Considérons le nouveau programme linéaire où l'on rajoute la contrainte $\Sigma\, g_i\, x_i = Max$, avec $Max \in \{372, 373, \ldots, 380\}$.
Le programme Scilab correspondant s'écrit :

```
for Max = 372:380
  c   = [5 6 4 3] ;
  A1 = c ;
  b1 = Max ;
  A2 = [2 2 1 1 ; 1 2 3 3 ; 1 3 2 0 ; 3 0 1 3 ; 2 1 2 1] ;
  b2 = [130 ; 152 ; 136 ; 129 ; 143] ;
  m   = [0 ; 0 ; 0 ; 0] ;
  M   = [] ;
  p1 = - c' ; p2 = [A1;A2] ; p3 = [b1;b2] ; p4 = m ; p5 = M ; p6 = size(A1,'r') ; p7 = 'v' ;
  [x, lagr, f] = linpro(p1, p2, p3, p4, p5, p6, p7)
end
```

Pour Max = 380, on obtient la solution entière : $x_1 = 30$; $x_2 = 22$; $x_3 = 20$; $x_4 = 6$ qui est la solution optimale.

Finalement, on décide de fabriquer 30 produits P_1 , 22 produits P_2 , 20 produits P_3 , et 6 produits P_4 pour obtenir le gain maximum $g = 5\times30 + 6\times22 + 4\times20 + 3\times6 = 380$.

Remarque :

Il est préférable de remplacer la boucle croissante "for Max = 372:380 ... end" du dernier programme par la boucle décroissante "for Max = 380:-1:372 ... end" afin d'arrêter les calculs dès qu'une solution entière est trouvée.

```
function [x, I, S] = PLNE(Max)
  c  = [5 6 4 3]
  if Max == [] then, A1 = [], else, A1 = c, end
  b1 = Max
  A2 = [2 2 1 1 ; 1 2 3 3 ; 1 3 2 0 ; 3 0 1 3 ; 2 1 2 1]
  b2 = [130 ; 152 ; 136 ; 129 ; 143]
  m  = [0 ; 0 ; 0 ; 0]
  M  = []
  p1 = - c' ; p2 = [A1;A2] ; p3 = [b1;b2] ; p4 = m ; p5 = M ; p6 = size(A1,'r') ; p7 = 'v'
  [x, lagr, f] = linpro(p1, p2, p3, p4, p5, p6, p7)
  I = c*floor(x)
  S = floor(-f)
endfunction

[x, Inf, Sup] = PLNE([]) ;
for Max = Sup:-1:Inf
  [x, I, S] = PLNE(Max) ;
  if max(abs(x-round(x))) <= 1d-10 then, Max, Sol = round(x), break, end
end
```

Le programme ci-dessus ne permet pas de résoudre systématiquement un programme linéaire en nombres entiers.
La fonction *linpro* peut très bien retourner un vecteur x réel au lieu d'un vecteur x entier pour une valeur f_{max} donnée. Par exemple pour $f_{max} = 371$:
$x_1 = 32.875$; $x_2 = 14.375$; $x_3 = 30.$; $x_4 = 0.125$ au lieu de $x_1 = 30$; $x_2 = 21$; $x_3 = 20$; $x_4 = 5$ car
$371 = 5\times263/8 + 6\times115/8 + 4\times30 + 3\times1/8 = 5\times30 + 6\times21 + 4\times20 + 3\times5$

Conclusion

Cette manière de procéder en 2 étapes a permis de trouver la solution optimale :

I) on résoud le programme linéaire pour $x \in \mathbb{R}$ (i.e. la contrainte $x \in \mathbb{N}$ est relâchée).

II) on résoud une série de programmes linéaires en rajoutant à chaque fois une contrainte correspondant à une valeur optimale possible de la fonction objectif ($f_{max} \in$ [Inf , Sup]).

En règle générale, la programmation linéaire en nombres entiers (PLNE), autrement dit la programmation linéaire où les inconnues doivent être des nombres entiers, est un problème difficile.
La méthode des coupes de **Gomory**, qui converge toujours, et la procédure de Séparation et Evaluation Progressive (Branch and Bound) sont utilisées pour obtenir la solution optimale ou, au moins, une solution proche lorsque le système est de grande dimension.
La Recherche Opérationnelle a fait, et fait toujours, l'objet de nombreux travaux. Des bibliothèques de programmes concernant la programmation linéaire sont disponibles. Elles sont proposées par des sociétés spécialisées ou des centres de recherche, certaines sont gratuites : ⇨ http://www.gnu.org/software/glpk/glpk.html

EXEMPLE 2 : UN PROBLÈME DE TRANSPORT

Une centrale d'achat possède p entrepôts E_1 , E_2 , ... , E_p et doit régulièrement réapprovisionner en marchandise q magasins M_1 , M_2 , ... , M_q . Les demandes de marchandise des magasins sont d_1 , d_2 , ... , d_q et les stocks disponibles dans les entrepôts sont s_1 , s_2 , ... , s_p . Le transport d'une quantité t_{ij} de marchandise de l'entrepôt E_i vers le magasin M_j a un coût unitaire c_{ij} (donc un coût global $c_{ij} \times t_{ij}$).
Quel est le "plan de transport" de coût minimal ?

On écrit que le coût total d'approvisionnement des magasins est une fonction à minimiser, sous les contraintes des demandes et des stocks. Soit :

$$\begin{cases} \displaystyle\sum_i \sum_j c_{ij}\, t_{ij} = f(Min) \\ \displaystyle\sum_i t_{ij} = d_j \qquad \forall\, j = 1, \ldots, q \\ \displaystyle\sum_j t_{ij} \le s_i \qquad \forall\, i = 1, \ldots, p \\ t_{ij} \ge 0 \end{cases}$$

Le problème admet une solution si les stocks sont supérieurs aux demandes :

$$\sum_i s_i \ge \sum_j d_j$$

On montre que si les données s_i et d_j sont entières, alors les inconnues t_{ij} sont également entières :

$$s_i \text{ et } d_j \in \mathbb{N} \quad \Rightarrow \quad t_{ij} \in \mathbb{N}$$

Application numérique : 3 entrepôts approvisionnent 7 magasins.

Stocks dans les entrepôts :

E_1	E_2	E_3
60	81	77

Demandes des magasins :

M_1	M_2	M_3	M_4	M_5	M_6	M_7
19	36	32	24	19	41	28

Coûts unitaires de transport :

	M_1	M_2	M_3	M_4	M_5	M_6	M_7
E_1	27	29	54	21	50	50	58
E_2	23	61	28	22	52	21	22
E_3	48	38	30	17	18	57	53

Programme :

```
// ------ Matrices
c  = [27 29 54 21 50 50 58 ...
      23 61 28 22 52 21 22 ...
      48 38 30 17 18 57 53] ;
A1 = [1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0  ; ...
      0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0  ; ...
      0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0  ; ...
      0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0  ; ...
      0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0  ; ...
      0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0  ; ...
      0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1] ;
b1 = [19 ; 36 ; 32 ; 24 ; 19 ; 41 ; 28] ;
A2 = [1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0  ; ...
      0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0  ; ...
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1] ;
b2 = [60 ; 81 ; 77] ;
m  = zeros(21,1) ;
M  = [] ;

// ------ Résolution
p1 = c'; p2 = [A1;A2]; p3 = [b1;b2]; p4 = m; p5 = M; p6 = size(A1,'r'); p7 = 'v';
[x, lagr, f] = linpro(p1, p2, p3, p4, p5, p6, p7) ;
Min = round(f) , Sol = round(x)
```

Quantités livrées :

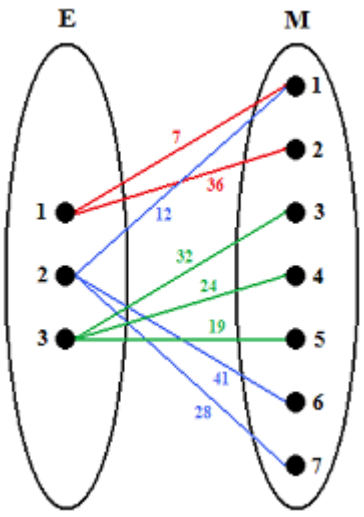
	M_1	M_2	M_3	M_4	M_5	M_6	M_7
E_1	7	36	0	0	0	0	0
E_2	12	0	0	0	0	41	28
E_3	0	0	32	24	19	0	0

Stocks restants :

$E_1 \to 17$, $E_2 \to 0$, $E_3 \to 2$.

Coût minimum du transport :

Min = 4696



Remarque :

Avec 3 entrepôts et 7 magasins, la matrice des contraintes a pour dimension $(3 + 7 , 3 \times 7) = (10 , 21)$ et possède 210 éléments.
Les problèmes de transport ont souvent des dimensions importantes, mais compte tenu de leur structure particulière (matrice creuse totalement unimodulaire), des algorithmes spéciaux ont étés développés pour les résoudre.
Ci-contre, le dessin du graphe biparti correspondant à l'exemple traité.

EXEMPLE 3 : UN PROBLÈME D'AFFECTATION

Un ensemble de n tâches doit être réalisé par n personnes.
Chaque personne i peut effectuer la tâche j avec une productivité évaluée à p_{ij}.
Quelle est l'affectation des tâches qui donne la meilleure productivité globale ?

Soit a_{ij} la variable prenant la valeur 1, ou 0, suivant que la personne i est, ou n'est pas, affectée à la tâche j.
Il faut résoudre le programme linéaire suivant :

$$\begin{cases} \sum_i \sum_j p_{ij} a_{ij} = f(Max) \\ \sum_i a_{ij} = 1 \qquad \forall\, j = 1, \ldots, n \\ \sum_j a_{ij} = 1 \qquad \forall\, i = 1, \ldots, n \\ a_{ij} \in \{0,1\} \end{cases}$$

Application numérique : Productivité de 4 personnes (P_1, P_2, P_3, P_4) pour 4 tâches (T_1, T_2, T_3, T_4).

	T_1	T_2	T_3	T_4
P_1	13	16	7	12
P_2	6	10	2	5
P_3	4	7	0	4
P_4	12	14	6	10

Programme :

```
// ------ Matrices
c  = [13 16 7 12 6 10 2 5 4 7 0 4 12 14 6 10] ;
A1 = [1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0  ; ...
      0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0  ; ...
      0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0  ; ...
      0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1  ; ...
      1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0  ; ...
      0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0  ; ...
      0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0  ; ...
      0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1] ;
b1 = ones(8,1) ;
A2 = [] ;
b2 = [] ;
m  = zeros(16,1) ;
M  = ones(16,1) ;

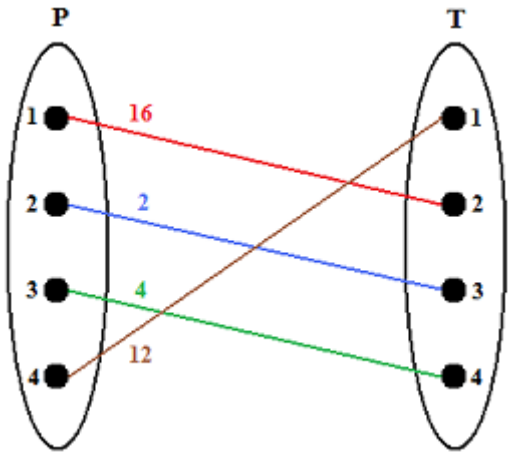
// ------ Résolution
p1 = - c'; p2 = [A1;A2]; p3 = [b1;b2]; p4 = m; p5 = M; p6 = size(A1,'r'); p7 = 'v';
[x, lagr, f] = linpro(p1, p2, p3, p4, p5, p6, p7) ;
Max = - round(f) , Sol = round(x)
```

Résultat :

	T_1	T_2	T_3	T_4
P_1	0	1	0	0
P_2	0	0	1	0
P_3	0	0	0	1
P_4	1	0	0	0

Productivité maximale : Max = 34

Graphe biparti correspondant à l'exemple ci-dessus :



Remarque :

La matrice des contraintes d'un tableau (P,T) de dimension (n,n) possède $2n^3$ éléments dont $2n^2$ non nuls égaux à 1.
La matrice creuse A1 est obtenue avec le programme :

```
A1 = zeros(2*n , n^2) ;
for i = 1:n
  for j = 1:n
    A1(i , i + (j - 1)*n) = 1 ;
    A1(n + i , (i - 1)*n + j) = 1 ;
  end
end
```

Pour n = 10, par exemple, la matrice A1 contient 2000 éléments dont 1800 sont nuls.

En considérant le tableau de départ, le problème d'affectation consiste à prendre une seule valeur par ligne et par colonne, de sorte que la somme obtenue soit maximale (ou minimale selon l'objectif à atteindre).
Si n est grand, le problème se complique à cause de l'accumulation des erreurs d'arrondi dans les calculs.

Mathématique & Informatique

Soit le problème d'affectation à minimiser, défini par le tableau suivant :

```
1 1 5 2 3 4 7 2 9 1
9 5 3 0 4 8 2 0 7 3
4 1 1 2 7 5 2 4 1 2
8 4 5 2 2 4 3 3 6 8
2 5 4 6 1 7 6 2 9 9
6 3 4 0 6 8 4 0 2 4
7 1 3 2 4 7 9 2 3 1
1 6 1 5 0 3 0 2 4 1
9 5 6 3 2 0 0 4 7 4
1 9 8 0 0 1 2 3 7 6
```

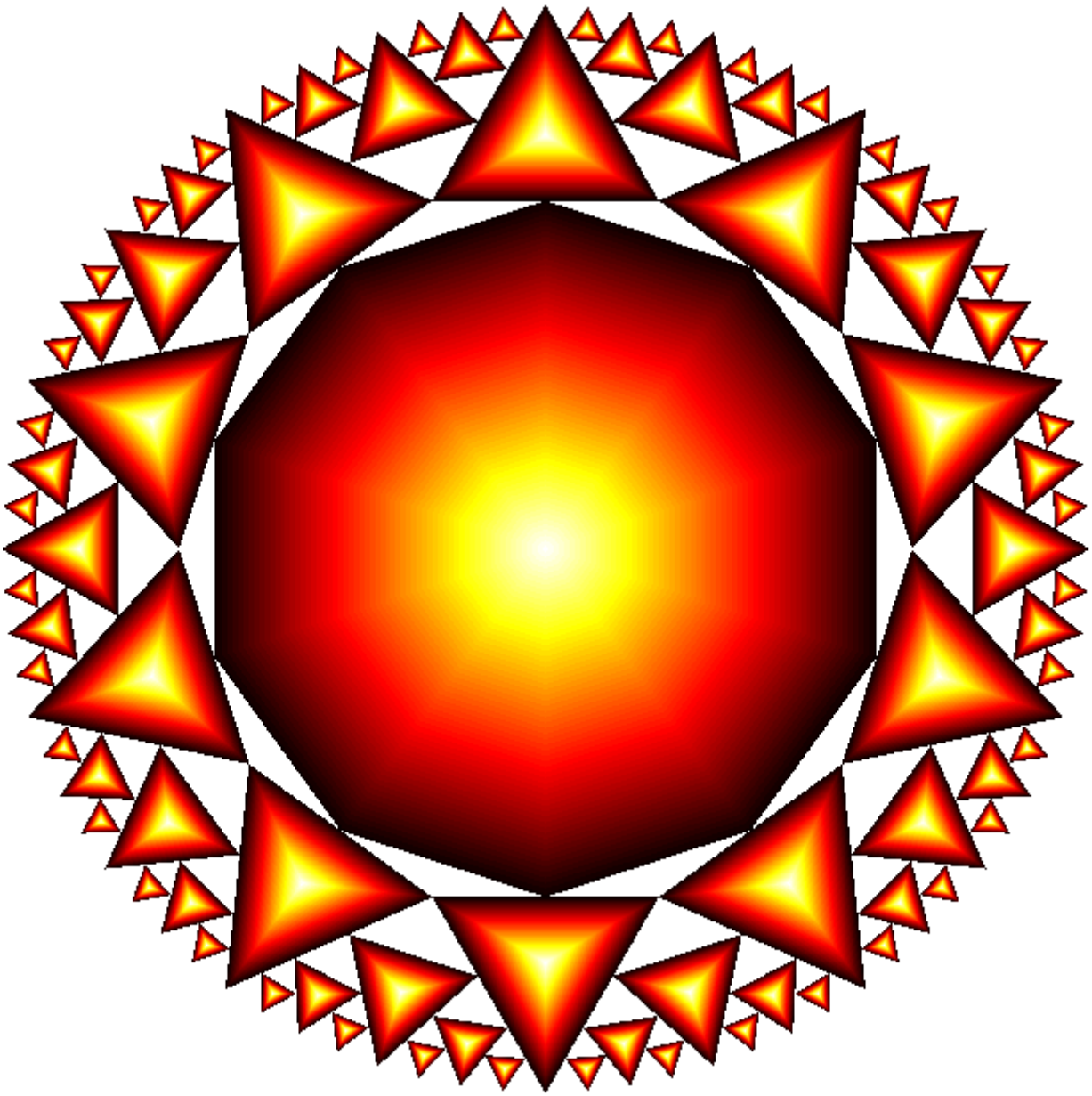
Programme :

```
// ------ Matrices
c = [1 1 5 2 3 4 7 2 9 1 ...
     9 5 3 0 4 8 2 0 7 3 ...
     4 1 1 2 7 5 2 4 1 2 ...
     8 4 5 2 2 4 3 3 6 8 ...
     2 5 4 6 1 7 6 2 9 9 ...
     6 3 4 0 6 8 4 0 2 4 ...
     7 1 3 2 4 7 9 2 3 1 ...
     1 6 1 5 0 3 0 2 4 1 ...
     9 5 6 3 2 0 0 4 7 4 ...
     1 9 8 0 0 1 2 3 7 6] ;
n = 10 ;
A1 = zeros(2*n, n^2) ;
for i = 1:n
  for j = 1:n
    A1(i, i + (j-1)*n) = 1 ; A1(n + i, (i-1)*n + j) = 1 ;
  end
end
b1 = ones(2*n,1) ; A2 = [] ; b2 = [] ; m = zeros(n^2,1) ; M = ones(n^2,1) ;
// ------ Résolution
p1 = c' ; p2 = [A1;A2] ; p3 = [b1;b2] ; p4 = m ; p5 = M ; p6 = size(A1,'r') ; p7 = 'v' ;
[x, lagr, f] = linpro(p1, p2, p3, p4, p5, p6, p7) ;
A = zeros(n,n) ;
for i = 1:n
  for j = 1:n
    A(i,j) = round(x((i-1)*n + j)) ;
  end
end
Min = round(f) ;
A, Min
```

Solution :

$$A = \begin{pmatrix} 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. \\ 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. \\ 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0 \end{pmatrix}, \quad Min = 9$$

Mathématique & Informatique



Mathématique & Informatique

SUDOKU 数独

Le Sudoku est un jeu dont le nom d'origine japonaise signifie "Chiffre Unique".
Il consiste à remplir une grille de 81 cases, divisée en 9 carrés (ou régions) de 3 × 3 cases.
Dans une grille, chaque ligne, chaque colonne et chaque région doit contenir tous les chiffres de 1 à 9.
Le nombre de grilles possibles a été calculé par Bertram **Felgenhauer** et Frazer **Jarvis** en 2005 :

N = 6 670 903 752 021 072 936 960

Plus de 6 10^{21} , c'est-à-dire 6 mille milliards de milliards !
Un exemple de grille solution :

3	8	7	4	2	1	9	5	6
4	1	5	9	6	7	2	3	8
6	2	9	5	8	3	1	4	7
8	7	4	3	5	2	6	1	9
5	9	6	7	1	4	8	2	3
2	3	1	8	9	6	4	7	5
1	5	2	6	7	8	3	9	4
9	4	8	1	3	5	7	6	2
7	6	3	2	4	9	5	8	1

Un problème de Sudoku consiste à compléter une grille de départ de $3^2 \times 3^2$ cases partiellement vide pour obtenir une grille solution où toutes les cases sont remplies. On se propose d'écrire un logiciel Scilab qui résoud ce problème.
D'une manière générale, placer des nombres sur une grille de $n^2 \times n^2$ cases est un problème NP-complet.

INDICATIONS POUR LA COMPRÉHENSION DU PROGRAMME.

La grille du Sudoku est numérotée de gauche à droite et de haut en bas. Soit i une case ($1 \leq i \leq 81$) :

Vecteur D
D(i) est la valeur donnée à la case i de la grille de départ. D(i) ≠ 0 si la valeur est connue, 0 sinon.

Vecteur S
S(i) est la valeur donnée à la case i de la grille solution. S(i) ∈ {1,2,3,4,5,6,7,8,9}

Vecteur L et matrice Lig
L(i) est le numéro de ligne de la case i et Lig(i,j) est le j^e numéro de case de la ligne i.

Vecteur C et matrice Col
C(i) est le numéro de colonne de la case i et Col(i,j) est le j^e numéro de case de la colonne i.

Vecteur R et matrice Reg
R(i) est le numéro de région de la case i et Reg(i,j) est le j^e numéro de case de la région i.

Matrice T
T(i,j) ≠ 0 est la j^e valeur possible de la case i. T(i,10) est le nombre de valeurs possibles de la case i.

La fonction "simplification"

Dans le tableau T on supprime les valeurs correspondant à des grilles impossibles.

```
i = 1
répéter
|       si  T(i,10) = 1  alors  Eliminer les doublons
|       i = i + 1
tantque  i ≤ 81


trouvé = faux
répéter
|       Rechercher si une case i a une valeur déterminée
|       si  trouvé  alors  Eliminer les doublons
tantque  trouvé
```

La fonction "exhaustive"

A partir d'un tableau T on recherche une grille solution S telle que : ligne = colonne = région = {1,2,3,4,5,6,7,8,9}

```
début :   pour  i = 1  à  81  faire  I(i) = 1
test :    Vérifier si la grille S(i) = T(i,I(i)) est solution
          n = 0
suite :   n = n + 1
          I(n) = I(n) + 1
          si  I(n) ≤ T(n,10)  alors  aller à  test
          I(n) = 1
          si  n < 81  alors  aller à  suite
fin :     Toutes les grilles possibles ont été vérifiées
```

Théoriquement, le tableau de départ $T = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 9\}^{81}$ permettrait d'obtenir les N grilles possibles du SUDOKU. Irréalisable en pratique puisqu'il faudrait vérifier 9^{81} grilles.

La fonction "hypothèse"

Si le temps de calcul nécessaire à une résolution par élimination est trop long, et si le tableau T ne peut être simplifié davantage sans faire des hypothèses, alors on procède selon la méthode heuristique suivante :

I) Trier les cases indéterminées dans l'ordre croissant du nombre de chiffres admissibles
⇒ Tableau TH dans le programme

II) Sélectionner un certain nombre d'entre elles et rechercher la solution de manière exhaustive parmi toutes les combinaisons de chiffres possibles pour ces cases
⇒ Liste LH dans le programme

La fonction "aléatoire"

Lorsque le nombre de grilles à examiner est très grand, on effectue des hypothèses multiples en utilisant la fonction Scilab *grand*(1, 1, 'uin', n_1 , n_2) qui génère un entier aléatoire n compris entre n_1 et n_2 ($n_1 \le n \le n_2$).

Les entrées/sorties

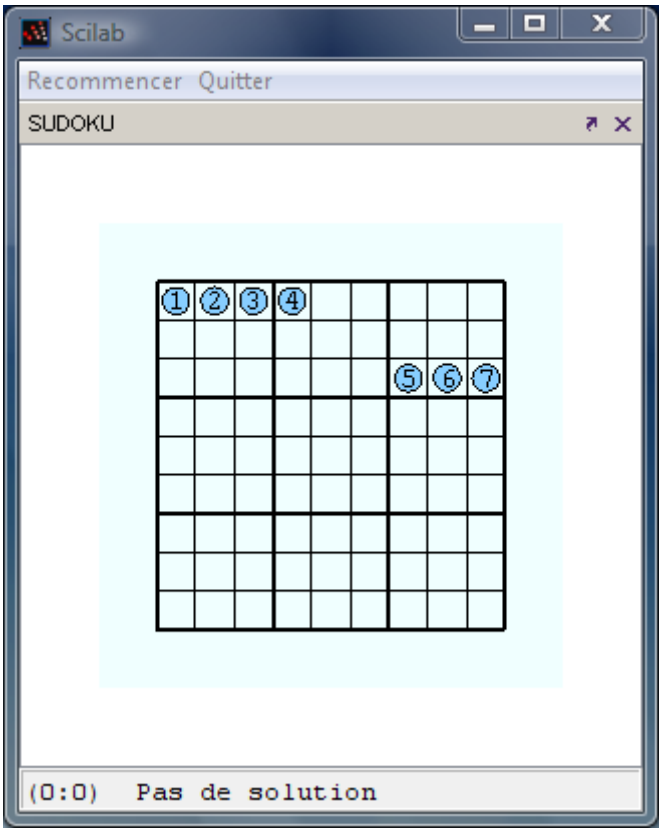
Pour lire la grille de départ au clavier on utilise la fonction *x_dialog* ou la fonction *x_mdialog* de l'Interface graphique et pour afficher la grille solution à l'écran on fait appel à la Bibliothèque graphique (*Graphics*).

Remarques

- Typiquement, pour une grille de départ d'environ 25 chiffres dévoilés, le temps de recherche est de l'ordre de la seconde avec un micro-ordinateur et un programme interprété.
  Pour certaines grilles particulièrement difficiles, le temps de calcul peut atteindre une minute.
  En compilant le programme au lieu de l'interpréter, on obtient un temps d'exécution bien inférieur.
  A ce sujet, un générateur de code C à partir d'un script est téléchargeable en complément de la version de base de Scilab. Pour ce faire, cliquer sur l'icône du gestionnaire de modules ATOMS dans la barre d'outils, ensuite sur *Tous les modules*, puis sur *Scilab 2 C* dans la liste déroulante, et terminer en cliquant sur le bouton *Installer*.

- Lorsqu'il existe de nombreuses grilles solutions, des exécutions successives donnent généralement des résultats différents. En particulier, pour une grille initiale vide, le solveur affiche 2 grilles solutions quelconques parmi les N possibles.

- Il arrive qu'une grille de départ n'admette aucune solution. Par exemple :



```
//
//       Loïc Vrillon - Janvier 2009
//
//
// ---------- Solveur de SUDOKU ----------
//

// -------- Fonction vérification

function [D, valid1, valid2] = verification(rep)
  M = [] ; D = zeros(1,81) ; valid1 = %f ; valid2 = %f ;
  err = execstr('M = evstr(rep)', 'errcatch') ;
  if (err <> 0) | (size(M,1) <> 9) | (size(M,2) <> 9) then, D = [] ; return, end
  for i = 1:9
    for j = 1:9
      digit = prod(M(i,j)*ones(1,10) - [0:9]) == 0 ;
      if ~digit then, D = [] ; return, end
      D((i-1)*9 + j) = M(i,j) ;
    end
  end
  valid1 = %t ;
  for v = 1:9
    for i = 1:9
      lig = 0 ; col = 0 ; reg = 0 ;
      for j = 1:9
        if D(Lig(i,j)) == v then, lig = lig + 1 ; end
        if D(Col(i,j)) == v then, col = col + 1 ; end
        if D(Reg(i,j)) == v then, reg = reg + 1 ; end
        if (lig > 1) | (col > 1) | (reg > 1) then, D = [] ; return, end
      end
    end
  end
  valid2 = %t ;
endfunction

// -------- Fonction initialisation

function T = initialisation(D)
  T = zeros(81,10) ;
  for i = 1:81
    if D(i) == 0 then
      for j = 1:9, T(i,j) = j ; end
      T(i,10) = 9 ;
    else
      T(i,1) = D(i) ; T(i,10) = 1 ;
    end
  end
endfunction
```

```
// -------- Fonction simplification

function T = simplification(T)

  function T = maj(T, i, v)
    lig = L(i) ; col = C(i) ; reg = R(i) ;
    for j = 1:9
      if Lig(lig,j) <> i then, T(Lig(lig,j),v) = 0 ; end
      if Col(col,j) <> i then, T(Col(col,j),v) = 0 ; end
      if Reg(reg,j) <> i then, T(Reg(reg,j),v) = 0 ; end
    end
  endfunction

  if T == [] then, return, end
  if prod(T(:,10)) == 1 then, return, end
  for i = 1:81
    memory = T(i,1:9) ; T(i,1:9) = 0 ;
    for j = 1:T(i,10), T(i,memory(j)) = 1 ; end
  end
  for i = 1:81
    if T(i,10) == 1 then
      v = find(T(i,1:9) == 1) ;
      T = maj(T, i, v) ;
    end
  end

  found = %t ;
  while found
    found = %f ;
    for v = 1:9
      for i = 1:9
        vl = 0 ; lig = 0 ;
        for j = 1:9
          k = Lig(i,j) ;
          if T(k,v) == 1 then
            vl = vl + 1 ;
            if T(k,10) <> 1 then
              nl = k ; lig = lig + 1 ;
              if lig > 1 then, break, end
            end
          end
        end
        if vl == 0 then, T = [] ; return, end
        if lig == 1 then
          T(nl,1:9) = 0 ; T(nl,v) = 1 ; T(nl,10) = 1 ;
          T = maj(T, nl, v) ; found = %t ;
        end
```

```
        vc = 0 ; col = 0 ;
        for j = 1:9
          k = Col(i,j) ;
          if T(k,v) == 1 then
            vc = vc + 1 ;
            if T(k,10) <> 1 then
              nc = k ; col = col + 1 ;
              if col > 1 then, break, end
            end
          end
        end
        if vc == 0 then, T = [] ; return, end
        if col == 1 then
          T(nc,1:9) = 0 ; T(nc,v) = 1 ; T(nc,10) = 1 ;
          T = maj(T, nc, v) ; found = %t ;
        end
        vr = 0 ; reg = 0 ;
        for j = 1:9
          k = Reg(i,j) ;
          if T(k,v) == 1 then
            vr = vr + 1 ;
            if T(k,10) <> 1 then
              nr = k ; reg = reg + 1 ;
              if reg > 1 then, break, end
            end
          end
        end
        if vr == 0 then, T = [] ; return, end
        if reg == 1 then
          T(nr,1:9) = 0 ; T(nr,v) = 1 ; T(nr,10) = 1 ;
          T = maj(T, nr, v) ; found = %t ;
        end
      end
    end
  end

  for i = 1:81
    k = 0 ;
    for j = 1:9
      if T(i,j) == 1 then
        k = k + 1 ; T(i,k) = j ;
      end
    end
    if k < 9 then, T(i,k+1:9) = 0 ; end
    T(i,10) = k ;
  end
  if prod(T(:,10)) == 0 then, T = [] ; end

endfunction
```

```
// -------- Fonction exhaustive

function LS = exhaustive(LS, T)
  S = zeros(1,81) ; I = ones(1,81) ;
  loop = %t ; test = %t ;
  while loop
    if test then
      for i = 1:81, S(i) = T(i,I(i)) ; end
      solution = %t ;
      for v = 1:9
        for i = 1:9
          lig = 0 ; col = 0 ; reg = 0 ;
          for j = 1:9
            if S(Lig(i,j)) == v then, lig = lig + 1 ; end
            if S(Col(i,j)) == v then, col = col + 1 ; end
            if S(Reg(i,j)) == v then, reg = reg + 1 ; end
            if (lig > 1) | (col > 1) | (reg > 1) then, break, end
          end
          if (lig <> 1) | (col <> 1) | (reg <> 1) then
            solution = %f ; break
          end
        end
        if ~solution then, break, end
      end
      if solution then
        new = %t ;
        for k = 1:length(LS)
          if isequal(LS(k),S) then, new = %f ; break, end
        end
        if new then, LS($+1) = S ; end
        if length(LS) > 1 then, return, end
      end
      n = 0 ;
    else
      I(n) = 1 ;
    end
    n = n + 1 ;
    I(n) = I(n) + 1 ;
    test = I(n) <= T(n,10) ;
    loop = test | (n < 81) ;
  end
endfunction
```

```
// -------- Fonction hypothèse

function LH = hypothese(LH, T)

  n = length(find(T(:,10) > 1)) ;
  TH = zeros(n,11) ;
  k = 0 ;
  for i = 1:81
    if T(i,10) > 1 then
      k = k + 1 ;
      for j = 1:10, TH(k,j) = T(i,j) ; end
      TH(k,11) = i ;
    end
  end

  j = 1 ;
  while j < n
    i = j ;
    while i >= 1
      if TH(i,10) > TH(i+1,10) then
        memory = TH(i+1,:) ;
        for k = 1:11, TH(i+1,k) = TH(i,k) ; end
        for k = 1:11, TH(i,k) = memory(k) ; end
        i = i - 1 ;
      else
        i = 0 ;
      end
    end
    j = j + 1 ;
  end

  nv = 0 ;
  for k = 1:n
    if prod(TH(1:k,10)) > MAXTRY2 then, break, end
    nv = nv + 1 ;
  end
  if nv == 0 then, return, end
  LH(1) = TH(1:nv,11) ;
  I = ones(1,nv) ;
  loop = %t ; test = %t ;
  while loop
    if test then
      LH($+1) = I ;
      n = 0 ;
    else
      I(n) = 1 ;
    end
    n = n + 1 ;
    I(n) = I(n) + 1 ;
    test = I(n) <= TH(n,10) ;
    loop = test | (n < nv)
  end

endfunction
```

```
// -------- Fonction choix

function TC = choix(T, i, j)
  TC = T ;
  TC(i,1) = T(i,j) ; TC(i,2:9) = 0 ; TC(i,10) = 1 ;
endfunction

// -------- Fonction suppression

function T = suppression(T, i, j)
  for k = j:T(i,10)-1, T(i,k) = T(i,k+1) ; end
  T(i,T(i,10)) = 0 ; T(i,10) = T(i,10) - 1 ;
endfunction

// -------- Fonction postulat

function T = postulat(T, h)
  for k = 1:nv, T = choix(T, LH(1)(k), LH(h)(k)) ; end
  T = simplification(T) ;
endfunction

// -------- Fonction aléatoire

function T = aleatoire(T)
  random = %t ;
  while random
    I = find(T(:,10) > 1) ; J = find(T(:,10) == min(T(I,10))) ;
    irand = J(grand(1, 1, 'uin', 1, length(J))) ;
    jrand = grand(1, 1, 'uin', 1, T(irand,10)) ;
    T = choix(T, irand, jrand) ; T = simplification(T) ;
    if T == [] then, break, else, N = prod(T(:,10)) ; end
    random = N > MAXITER3 ;
  end
endfunction
```

```
// -------- Fonction grille

function grille(S)
  xoffset = 0.5 ; yoffset = 0.5 ; d = 0.15 ;
  a = gca() ; a.isoview = 'on' ; a.axes_visible = 'off' ; a.data_bounds = [0,0;9+2*xoffset,9+2*yoffset] ;
  xrects([0;9+2*yoffset;9+2*xoffset;9+2*yoffset],color('azure1')) ;
  for i = 0:9
    xsegs([xoffset,xoffset+i;xoffset+9,xoffset+i], [yoffset+i,yoffset;yoffset+i,yoffset+9]) ;
    if modulo(i,3) == 0 then, e = gce() ; e.thickness = 2 ; end
  end
  for i = 1:81
    if D(i) <> 0 then
      xfarc(C(i)-1+xoffset+d, 10-L(i)+yoffset-d, 1-2*d, 1-2*d, 0, 64*360) ;
      e = gce() ; e.background = 12 ;
      xarc(C(i)-1+xoffset+d, 10-L(i)+yoffset-d, 1-2*d, 1-2*d, 0, 64*360) ;
      e = gce() ; e.foreground = 1 ;
    end
    if S <> [] then, c = string(S(i)) ; elseif D(i) <> 0 then, c = string(D(i)) ; else, c = '' ; end
    xstringb(C(i)-1+xoffset, 9-L(i)+yoffset, c, 1, 1) ;
    e = gce() ; e.font_style = 0 ; e.font_size = 3 ;
  end
endfunction

// -------- Fonction écrire

function ecrire(G)
  for i = 1:9
    printf('\n' + ' |' + strcat(string(G((i-1)*9+1:i*9)),'|') + '|')
  end
  printf('\n')
endfunction
```

```
// -------- Programme Principal

funcprot(0) ; lines(0) ;

// -------- Tableaux

L = zeros(1,81) ; C = zeros(1,81) ; R = zeros(1,81) ;
Lig = zeros(9,9) ; Col = zeros(9,9) ; Reg = zeros(9,9) ;
for i = 1:81
   L(i) = floor(i/9) ; if modulo(i,9) <> 0 then, L(i) = L(i) + 1 ; end
   C(i) = i - (L(i) - 1)*9 ;
   Lig(L(i),C(i)) = i ;
   Col(C(i),L(i)) = i ;
   I = floor(L(i)/3) ; if modulo(L(i),3) <> 0 then, I = I + 1 ; end
   J = floor(C(i)/3) ; if modulo(C(i),3) <> 0 then, J = J + 1 ; end
   K = i - (I - 1)*27 - (J - 1)*3 ; K = K - floor(K/9)*6 ;
   R(i) = (I - 1)*3 + J ;
   Reg(R(i),K) = i ;
end

// -------- Grille de départ

rep = strsubst(string(zeros(9,1)), '0', strcat(string(zeros(1,9)),' ')) ;
text = '' ; valid = %f ;
while ~valid
   rep = x_dialog(['SUDOKU :   Grille de départ   9x9', text], rep) ;
   if rep == [] then, abort, end
   [D, valid1, valid2] = verification(rep) ;
   if ~valid1 then
      text = 'ENTRER 81 CHIFFRES (0 à 9).' ;
   elseif ~valid2 then
      text = 'GRILLE DE DEPART INEXACTE.' ;
   end
   valid = valid1 & valid2 ;
end

// -------- Recherche

cpu = 0 ; timer() ; winId = progressionbar('Recherche d''une solution ...') ;
MAXITER1 = 1E3 ; MAXITER2 = 1E6 ; MAXITER3 = 1E9 ;
MAX = 1E99 ; MAXTRY1 = 1E2 ; MAXTRY2 = 1E2 ;
LOOP = %t ; loop = %t ; computation = %t ; hyp = %f ;
LS = list() ; LH = list() ;
T = initialisation(D) ; T = simplification(T) ;
```

```
while LOOP
  while loop
    stop = %f ; repeat = %f ; flag = %f ; all = %t ; Min = MAX ; i = 1 ;
    if T == [] then, break, else, N = prod(T(:,10)) ; end
    if N <= MAXITER1 then, LS = exhaustive(LS, T) ; break, end
    while i <= 81
      if T(i,10) == 1 then, i = i + 1 ; continue, else, j = 1 ; end
      while j <= T(i,10)
        TC = choix(T, i, j) ; TC = simplification(TC) ;
        if TC == [] then
          T = suppression(T, i, j) ;
          if T(i,10) == 1 then, T = simplification(T) ; repeat = %t ; break, end
        else
          N = prod(TC(:,10)) ;
          if N <= MAXITER1 then
            LS = exhaustive(LS, TC) ;
            if length(LS) > 1 then, stop = %t ; break, end
            T = suppression(T, i, j) ;
            if T(i,10) == 1 then, T = simplification(T) ; repeat = %t ; break, end
          else
            if N < Min then, imin = i ; jmin = j ; Min = N ; flag = %t ; end
            j = j + 1 ;
          end
        end
      end
      if stop | repeat then, break, end
      i = i + 1 ;
    end
    if repeat then, continue, end
    if stop | ~flag then, break, end
    TC = choix(T, imin, jmin) ; TC = simplification(TC) ;
    if Min <= MAXITER2 then
      LS = exhaustive(LS, TC) ;
      if length(LS) > 1 then, break, end
      T = suppression(T, imin, jmin) ; T = simplification(T) ;
    else
      if ~hyp then
        T = simplification(T) ; TM = T ;
        LH = hypothese(LH, T) ; nh = length(LH) ;
        if nh <> 0 then, nv = length(LH(1)) ; else, nv = 0 ; end
        h = 1 ; nbtry = 0 ; computation = nh <> 0 ; hyp = %t ;
      end
      all = %f ; break
    end
  end
  cpu = cpu + timer() ; second = floor(cpu) ;
  progressionbar(winId, 'Temps CPU :   ' + string(second) + ' s') ;
  if (length(LS) > 1) | ~hyp then, break, end
  if nbtry == MAXTRY1 then, all = %t ; end
  T = TM ; nbtry = nbtry + 1 ;
  if nbtry <= MAXTRY1 then
    T = aleatoire(T) ;
  else
    computation = computation & all ;
    h = h + 1 ; if h <= nh then, T = postulat(T, h) ; else, break, end
  end
end
```

```
// -------- Grille solution

cpu = cpu + timer() ; second = floor(cpu) ; tenth = floor((cpu - second)*10) ;
time = '(' + string(second) + ':' + string(tenth) + ')' ;
close(winId) ;

f = scf(1) ;
drawlater() ;
f.figure_position = [0,0] ; f.figure_name = 'SUDOKU' ;
toolbar(1, 'off') ;
delmenu(1, 'Fichier') ; delmenu(1, 'Outils') ; delmenu(1, 'Édition') ; delmenu(1, '?') ;
addmenu(1, 'Recommencer') ; Recommencer_1 = 'xdel() ; exec("SUDOKU.sce",-1)' ;
addmenu(1, 'Quitter') ; Quitter_1 = 'xdel()' ;

select length(LS)
  case 0
    f.figure_size = [500,500] ;
    grille([]) ;
    if computation then
      f.info_message = time + '   Pas de solution' ;
      printf('\n' + ' Grille de départ :' + '\n')
      ecrire(D)
      printf('\n' + ' Pas de solution' + '\n')
    else
      f.info_message = time + '   Non trouvé' ;
      printf('\n' + ' Grille de départ :' + '\n')
      ecrire(D)
      printf('\n' + ' Non trouvé' + '\n')
    end
  case 1
    f.figure_size = [500,500] ;
    grille(LS(1)) ;
    if computation then
      f.info_message = time + '   Solution unique' ;
      printf('\n' + ' Grille de départ :' + '\n')
      ecrire(D)
      printf('\n' + ' Solution unique :' + '\n')
      ecrire(LS(1))
    else
      f.info_message = time + '   Solution' ;
      printf('\n' + ' Grille de départ :' + '\n')
      ecrire(D)
      printf('\n' + ' Solution :' + '\n')
      ecrire(LS(1))
    end
  else
    f.figure_size = [1000,500] ;
    subplot(1, 2, 1) ; grille(LS(1)) ;
    subplot(1, 2, 2) ; grille(LS(2)) ;
    f.info_message = time + '   Plusieurs solutions' ;
    printf('\n' + ' Grille de départ :' + '\n')
    ecrire(D)
    printf('\n' + ' Plusieurs solutions :' + '\n')
    ecrire(LS(1)) , ecrire(LS(2))
end
drawnow() ;
```

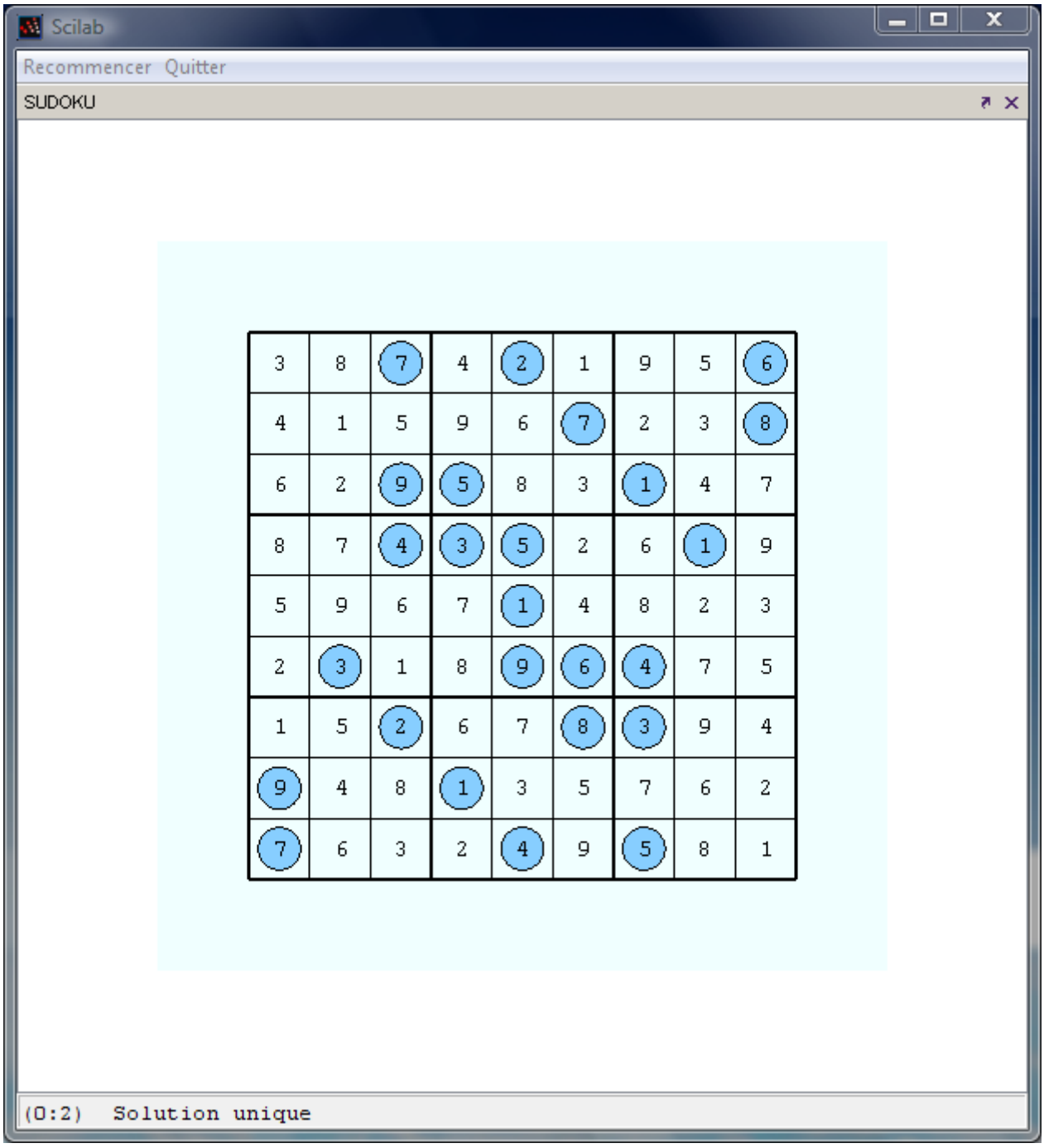
Version avec la fonction *x_mdialog* de saisie d'une matrice.

```
// -------- Fonction vérification

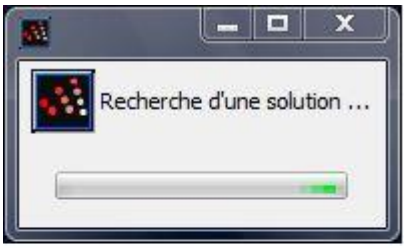
function [D, valid1, valid2] = verification(rep)
  D = zeros(1,81) ; valid1 = %f ; valid2 = %f ;
  for i = 1:9
    for j = 1:9
      if rep(i,j) == '0' then, D = [] ; return, end
      if rep(i,j) == '' then rep(i,j) = '0' ; end
      if length(rep(i,j)) == 1 then, digit = isdigit(rep(i,j)) ; else, digit = %f ; end
      if digit then, D((i-1)*9 + j) = evstr(rep(i,j)) ; else, D = [] ; return, end
    end
  end
  valid1 = %t ;
  for v = 1:9
    for i = 1:9
      lig = 0 ; col = 0 ; reg = 0 ;
      for j = 1:9
        if D(Lig(i,j)) == v then, lig = lig + 1 ; end
        if D(Col(i,j)) == v then, col = col + 1 ; end
        if D(Reg(i,j)) == v then, reg = reg + 1 ; end
        if (lig > 1) | (col > 1) | (reg > 1) then, D = [] ; return, end
      end
    end
  end
  valid2 = %t ;
endfunction

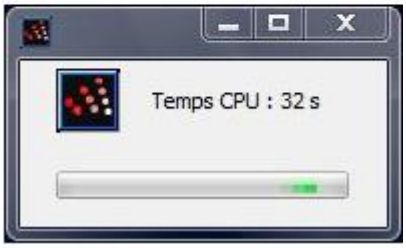

// -------- Grille de départ

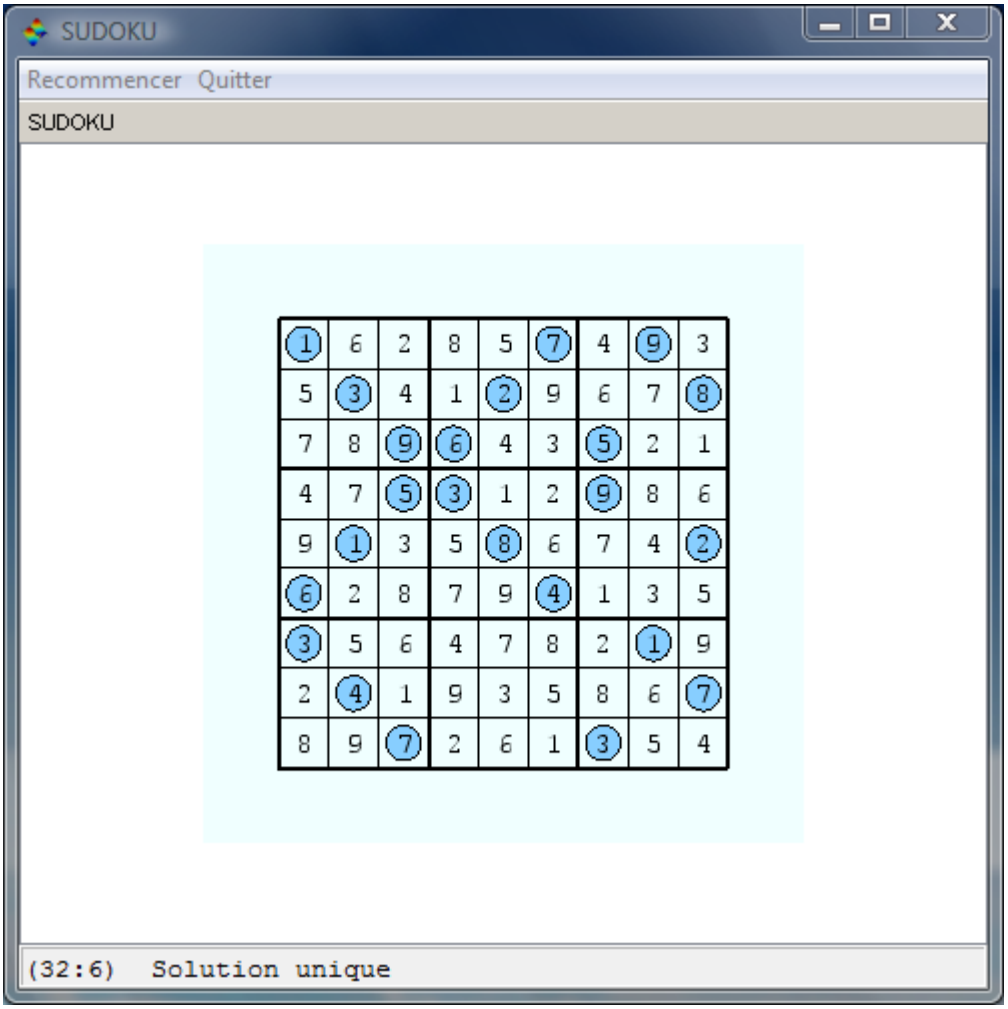
e = emptystr(1,9) ; E = emptystr(9,9) ;
text = '' ; rep = E ; valid = %f ;
while ~valid
  rep = x_mdialog(['SUDOKU :   Grille de départ   9x9', text], e, e, rep) ;
  if rep == [] then, abort, end
  rep = stripblanks(rep) ;
  [D, valid1, valid2] = verification(rep) ;
  if ~valid1 then
    text = 'ENTRER DES CHIFFRES COMPRIS ENTRE 1 ET 9.' ;
  elseif ~valid2 then
    text = 'GRILLE DE DEPART INEXACTE.' ;
  end
  valid = valid1 & valid2 ;
end
```


Scilab
Recommencer Quitter
SUDOKU
3 8 7 4 2 1 9 5 6
4 1 5 9 6 7 2 3 8
6 2 9 5 8 3 1 4 7
8 7 4 3 5 2 6 1 9
5 9 6 7 1 4 8 2 3
2 3 1 8 9 6 4 7 5
1 5 2 6 7 8 3 9 4
9 4 8 1 3 5 7 6 2
7 6 3 2 4 9 5 8 1
(0:2) Solution unique


Requête de valeur d'entrée...
SUDOKU : Grille de départ 9x9
1 0 0 0 0 7 0 9 0
0 3 0 0 2 0 0 0 8
0 0 9 6 0 0 5 0 0
0 0 5 3 0 0 9 0 0
0 1 0 0 8 0 0 0 2
6 0 0 0 0 4 0 0 0
3 0 0 0 0 0 0 1 0
0 4 0 0 0 0 0 0 7
0 0 7 0 0 0 3 0 0
OK
Cancel

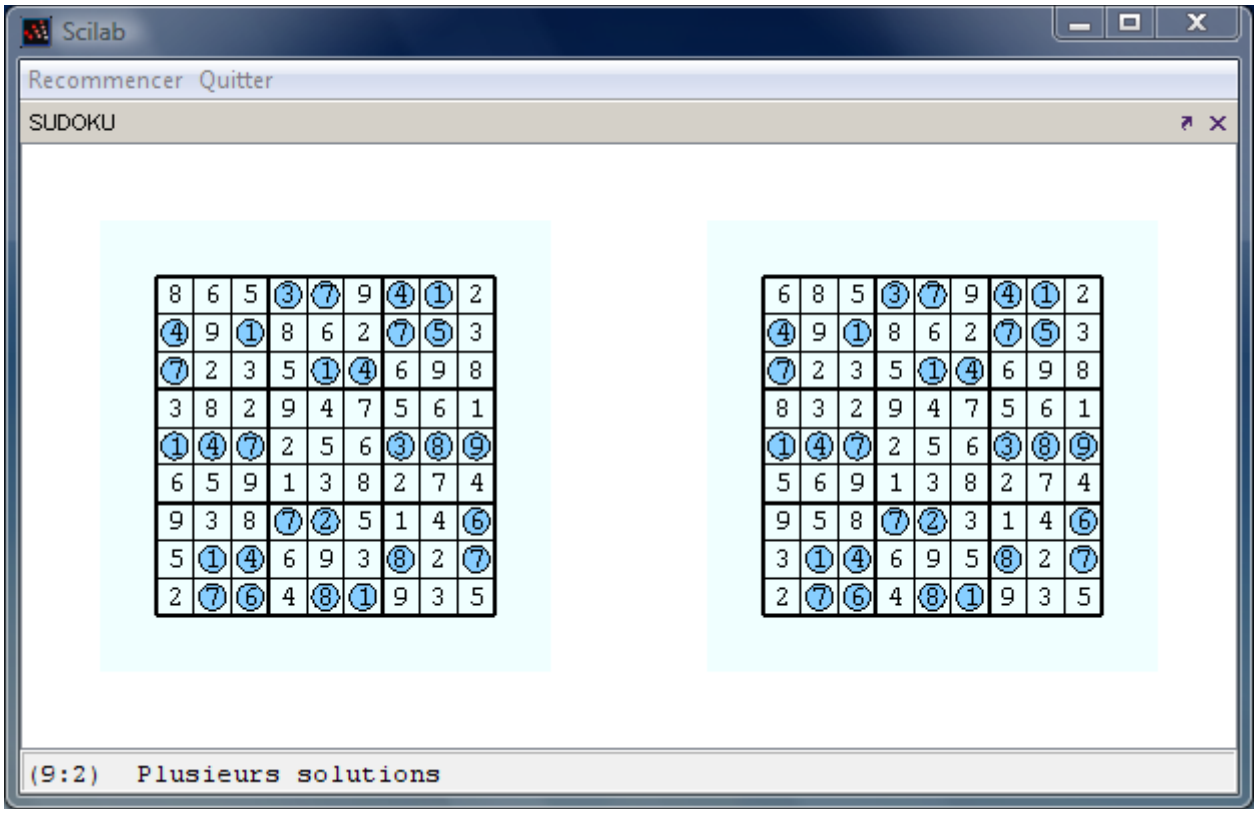

Recherche d'une solution ...

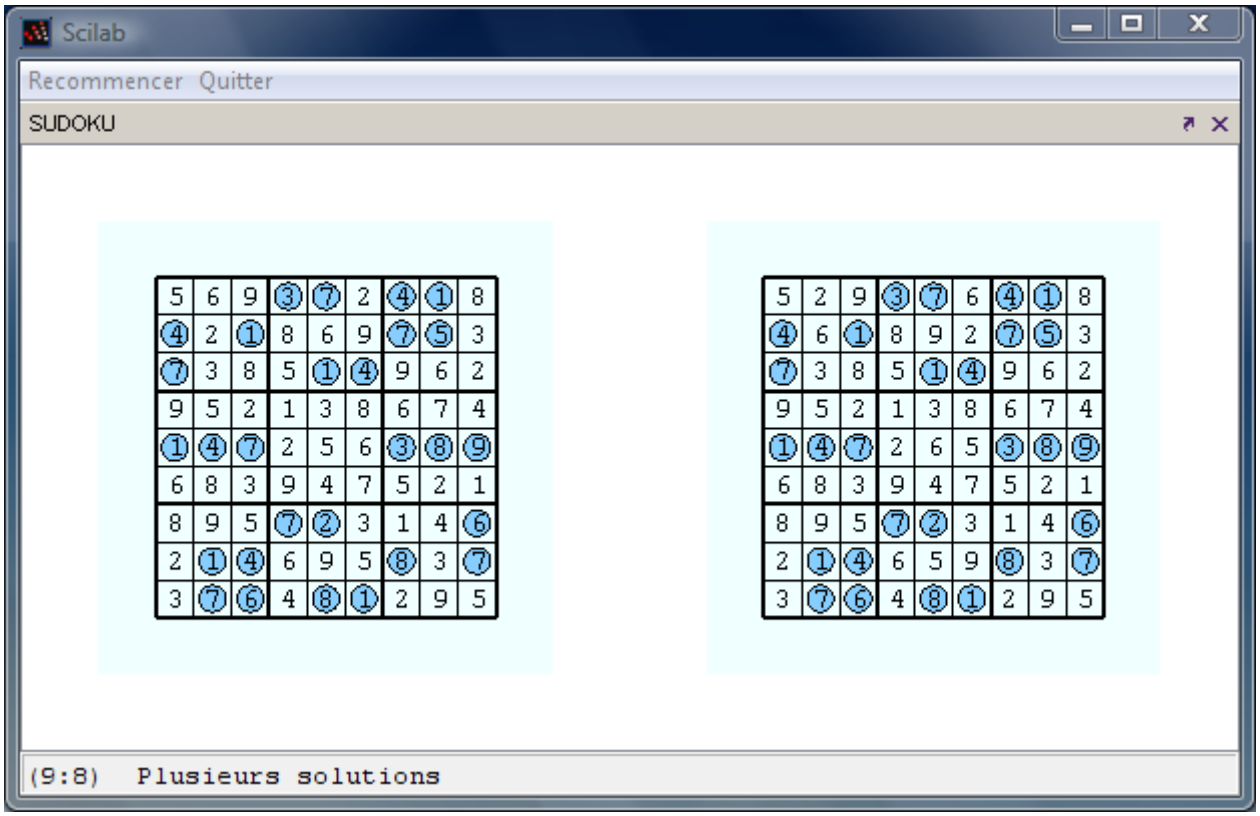

Temps CPU : 32 s


SUDOKU
Recommencer Quitter
SUDOKU
(32:6) Solution unique

Mathématique & Informatique

Sudoku à solutions multiples.





• • •

Mathématique & Informatique

Sudoku très difficile à solution unique.

1	6	2	8	5	7	4	9	3
5	3	4	1	2	9	6	7	8
7	8	9	6	4	3	5	2	1
4	7	5	3	1	2	9	8	6
9	1	3	5	8	6	7	4	2
6	2	8	7	9	4	1	3	5
3	5	6	4	7	8	2	1	9
2	4	1	9	3	5	8	6	7
8	9	7	2	6	1	3	5	4

7	5	1	8	4	6	2	3	9
8	9	2	3	7	1	4	6	5
6	4	3	2	5	9	8	7	1
2	3	8	1	9	7	5	4	6
9	7	4	5	6	2	3	1	8
1	6	5	4	3	8	9	2	7
3	1	9	6	8	4	7	5	2
5	2	7	9	1	3	6	8	4
4	8	6	7	2	5	1	9	3

1	7	4	3	8	5	9	6	2
2	9	3	4	6	7	1	5	8
5	8	6	1	9	2	7	3	4
4	5	1	9	2	3	8	7	6
9	2	8	6	7	4	3	1	5
3	6	7	8	5	1	2	4	9
7	1	9	5	4	8	6	2	3
6	3	5	2	1	9	4	8	7
8	4	2	7	3	6	5	9	1

7	6	1	3	5	4	2	8	9
2	9	8	1	6	7	3	4	5
4	5	3	9	2	8	1	6	7
8	1	2	6	4	9	7	5	3
9	7	6	5	1	3	4	2	8
5	3	4	8	7	2	6	9	1
3	2	7	4	8	5	9	1	6
1	8	9	2	3	6	5	7	4
6	4	5	7	9	1	8	3	2

9	2	5	4	1	3	7	8	6
6	8	4	5	7	9	1	3	2
3	1	7	6	8	2	5	9	4
1	4	3	2	9	6	8	7	5
8	6	9	3	5	7	2	4	1
7	5	2	8	4	1	3	6	9
5	7	6	1	3	4	9	2	8
2	3	8	9	6	5	4	1	7
4	9	1	7	2	8	6	5	3

1	2	3	4	5	7	6	8	9
4	5	6	1	8	9	3	2	7
7	8	9	2	6	3	5	1	4
2	4	5	6	3	1	9	7	8
3	1	7	9	4	8	2	6	5
9	6	8	7	2	5	1	4	3
5	3	1	8	7	6	4	9	2
6	7	2	3	9	4	8	5	1
8	9	4	5	1	2	7	3	6

Mathématique & Informatique

8	5	1	9	6	2	4	7	3
7	3	9	1	4	8	2	5	6
6	2	4	7	5	3	8	9	1
3	8	6	2	9	5	1	4	7
4	1	2	8	7	6	5	3	9
9	7	5	3	1	4	6	8	2
5	9	7	4	2	1	3	6	8
1	4	8	6	3	9	7	2	5
2	6	3	5	8	7	9	1	4

9	3	7	8	6	4	1	5	2
4	8	6	5	2	1	7	3	9
2	1	5	7	9	3	4	6	8
7	9	1	3	5	6	8	2	4
5	6	4	1	8	2	9	7	3
3	2	8	9	4	7	5	1	6
6	4	9	2	7	5	3	8	1
1	7	2	4	3	8	6	9	5
8	5	3	6	1	9	2	4	7

1	2	3	4	5	7	6	8	9
4	5	6	1	8	9	3	2	7
7	8	9	2	6	3	5	1	4
2	4	5	6	3	1	9	7	8
3	1	7	9	4	8	2	6	5
9	6	8	7	2	5	1	4	3
5	3	1	8	7	6	4	9	2
6	7	2	3	9	4	8	5	1
8	9	4	5	1	2	7	3	6

4	8	5	7	2	9	3	6	1
7	1	9	5	3	6	8	2	4
2	3	6	4	8	1	5	7	9
8	2	4	6	1	3	9	5	7
1	6	7	2	9	5	4	3	8
5	9	3	8	7	4	6	1	2
3	4	1	9	5	7	2	8	6
6	5	2	1	4	8	7	9	3
9	7	8	3	6	2	1	4	5

7	2	3	9	1	5	8	6	4
9	4	8	3	7	6	5	2	1
6	5	1	4	8	2	3	7	9
8	6	5	1	3	4	7	9	2
4	7	2	8	5	9	1	3	6
3	1	9	2	6	7	4	8	5
2	3	7	5	9	1	6	4	8
1	9	6	7	4	8	2	5	3
5	8	4	6	2	3	9	1	7

6	4	8	5	7	1	3	2	9
3	2	7	9	4	8	5	6	1
1	9	5	6	2	3	8	4	7
8	1	2	4	9	5	7	3	6
7	5	9	2	3	6	1	8	4
4	6	3	8	1	7	9	5	2
9	7	4	3	8	2	6	1	5
2	8	6	1	5	9	4	7	3
5	3	1	7	6	4	2	9	8

Mathématique & Informatique

COMPLÉMENTS.

⇨ Les variables locales/globales

Soit une fonction f :

function [résultats] = f (données)
|
| **v**
|
endfunction

et soit une variable v, utilisée dans le corps de la fonction f, qui ne fait pas partie des résultats ni des données.
Si v apparaît dans une instruction d'affectation (v = ...) alors elle est locale, c'est-à-dire qu'elle est différente de la variable v de même nom définie dans le programme principal.
Par contre, si v n'apparaît pas dans une instruction d'affectation alors elle est globale, c'est-à-dire qu'elle représente la variable v de même nom définie dans le programme principal.
Ce procédé permet à la fonction f d'avoir accès en lecture à toutes les variables du programme principal, sans devoir lui transmettre en paramètre une trop longue liste de données.

⇨ Les ruptures de séquence

- Dans les boucles :
  L'instruction *continue* revient au départ de la boucle (while, for, ...).
  L'instruction *break* sort de la boucle en cours (effectuer n break pour sortir de n boucles imbriquées).

- Dans les fonctions :
  L'instruction *return* sort de la fonction et revient au programme appelant.
  L'instruction *pause* suspend l'exécution dans l'attente de la saisie au clavier d'une instruction *resume*.

- Dans le programme principal :
  L'instruction *abort* termine le programme et ramène à la Console.
  L'instruction *exit* ferme la session Scilab.

Il est préférable de limiter l'utilisation de ces instructions pour garder l'aspect d'une programmation structurée.
L'instruction *pause* permet de déboguer un programme en accédant aux variables en cours d'exécution.

⇨ Les fonctions vectorielles

V est un vecteur de composantes V(i) :

- *length*(V) renvoie la dimension de V, c'est-à-dire Max i. (*length*([]) = 0)
- *sum*(V) renvoie la somme des éléments de V, c'est-à-dire ΣV(i). (*sum*([]) = 0)
- *prod*(V) renvoie le produit des éléments de V, c'est-à-dire ΠV(i). (*prod*([]) = 1)
- *find*(V *op* x) renvoie le vecteur des indices i pour lesquels V(i) *op* x est vrai. (*find*(%f) = [])

Exemple : V = [1 3 4 2 5 1]
length(V) = 6
sum(V) = 16
prod(V) = 120
find(V > 2) = [2 3 5]

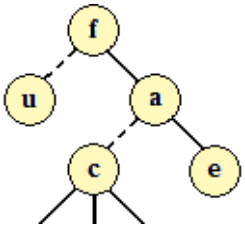
Mathématique & Informatique

⇨ Le mode graphique

La fonction *xset*(paramètres) permet une gestion simplifiée du contexte graphique, mais pour des opérations plus complexes on utilise d'autres fonctions.
La fonction *scf*() crée une figure (i.e. une fenêtre Windows) qui contiendra des axes et des entités graphiques.
Le tout est organisé selon une structure arborescente où les nœuds et les feuilles représentent les différents objets graphiques avec leurs caractéristiques.
Les fonctions *gcf*(), *gca*(), *gce*() fournissent les handles, c'est-à-dire les pointeurs qui permettent d'accéder à l'ensemble des éléments de la fenêtre graphique courante afin de pouvoir modifier leurs propriétés. Ainsi :

f = *gcf*() renvoie le handle **f** de la figure courante.
a = *gca*() renvoie le handle **a** des axes courants.
e = *gce*() renvoie le handle **e** de l'entité courante.

Cliquer sur *?* → *Aide de Scilab* → *Graphiques* → *graphics_entities* pour lister les propriétés modifiables.

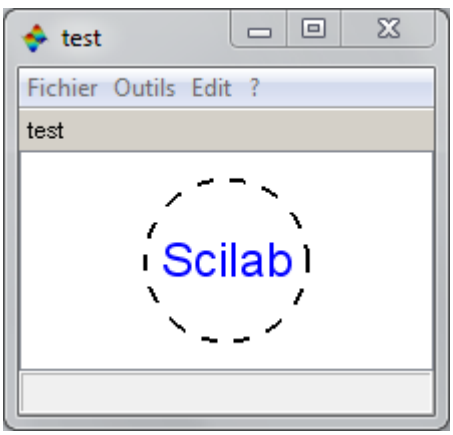
Objets	Propriétés	Instructions	Handles
figure	*figure_properties*	-	
axes	*axes_properties*	-	
composé	*compound_properties*	-	
courbe	*polyline_properties*	*plot2d*	
surface	*surface_properties*	*plot3d*	
rectangle	*rectangle_properties*	*xrect*	
arc d'ellipse	*arc_properties*	*xarc*	
segments de droites	*segs_properties*	*xsegs*	
texte	*text_properties*	*xstring*	

La figure est la racine de l'arbre. Aux niveaux inférieurs on trouve les axes, puis les composés et les objets élémentaires.
Un composé (compound) est un objet regroupant plusieurs objets graphiques, c'est un nœud de l'arbre.
Un objet élémentaire (courbe, surface, rectangle, arc, segments, texte, ...) est indivisible, c'est une feuille de l'arbre.
Une branche de l'arbre symbolise un lien parent-enfant entre 2 entités graphiques.
Soit **h** le handle d'un objet graphique. On descend dans la hiérarchie avec la syntaxe **h.children** et on remonte avec la syntaxe **h.parent**. Par exemple, pour n courbes dessinées dans le même repère Oxyz, les différents handles sont :
f = *gcf*() ⇒ figure, **a** = **f**.children ⇒ axes, **c** = **a**.children ⇒ composé, **ci** = **c**.children(i) ⇒ courbe i (i ∈ [1:n]).
Les objets GUI (*uimenu* et *uicontrol*) sont des enfants de la figure, au même titre que les axes : **u** = **f**.children(k).

NB.
Une figure possède au moins un système d'axes. Lorsqu'une figure **f** est divisée en n sous-figures avec l'instruction *subplot*(p,q,i), 1 ≤ i ≤ n, elle possède n systèmes d'axes accessibles via le vecteur d'handles **f.children** dont les composantes **f.children**(j), 1 ≤ j ≤ n, référencient les sous-figures dans l'ordre inverse de leur création (f.children(1) est le handle des axes de la dernière sous-figure affichée, f.children(n) est celui de la première sous-figure affichée).

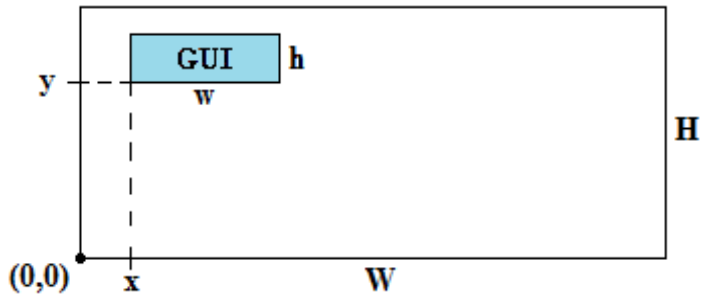
Exemple :
Créons une petite figure nommée "test", sans axes visibles, à l'intérieur de laquelle on écrit le mot "Scilab" en bleu, entouré d'un cercle en pointillés.

```
scf() ;
f = gcf() ; f.figure_size = [190,210] ; f.figure_name = "test" ;
a = gca() ; a.isoview = "on" ; a.axes_visible = "off" ;
xstringb(0, 0, "Scilab", 1, 1)
e = gce() ; e.font_size = 5 ; e.font_foreground = 2 ;
xarc(0, 1, 1, 1, 0, 64*360)
e = gce() ; e.thickness = 2 ; e.line_style = 3 ;
```



⇨ Les objets GUI

Dimensions de la fenêtre = [W,H] ; Coordonnées et dimensions de l'objet = [x,y,w,h]



L'interface graphique permet d'utiliser des objets GUI (Graphic User Interface).
Ces objets sont attachés à une figure, donc à une fenêtre Windows, et peuvent être insérés :

- dans la barre de menus → instruction *uimenu*
- dans la zone graphique → instruction *uicontrol*

Un objet GUI possède des **propriétés** (type, nom, taille, position, ...) qui le caractérisent et des **procédures** (instructions Scilab, programme C, ...) qui seront exécutées lorsqu'il sera selectionné par un clic de souris.

Exemple :

Le fichier *Supplement_GUI.sce*, placé sous le répertoire courant, contient un programme qui dessine la surface 3D donnée comme exemple par défaut pour la fonction *plot3d* de Scilab. En réponse au prompt de la Console, saisir la commande : *exec('Supplement_GUI.sce',-1)*. Une fenêtre graphique (numérotée "0") s'ouvre. Elle possède un bouton "Cartes de couleurs" pour changer la carte de couleurs du graphique, un menu "Vue" pour définir les angles de rotation en degrés (colatitude, longitude) du point de vue de la surface et un menu "Fichier" pour enregistrer la figure ou la copier dans le presse-papier.

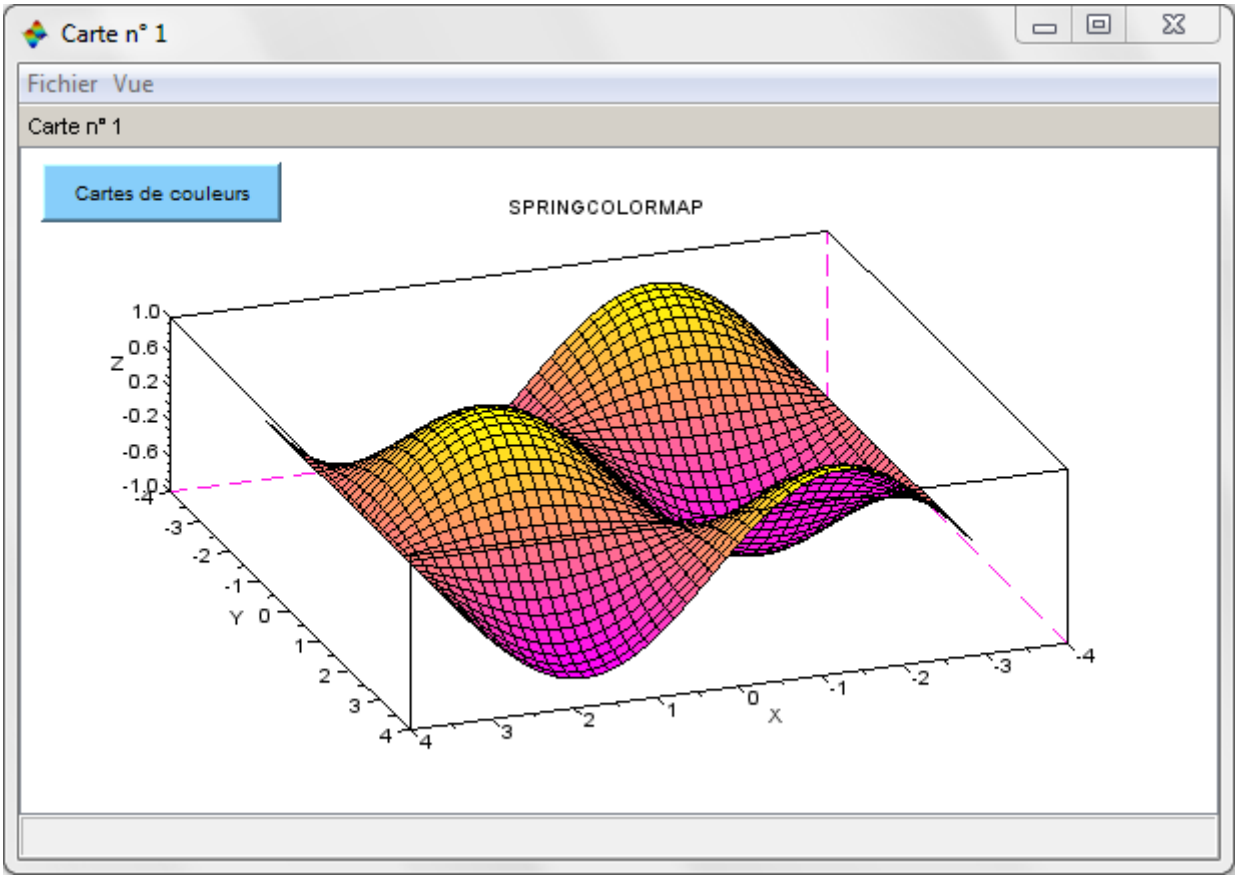
```
//--- Fenêtre graphique
f = scf() ; a = gca() ;
toolbar(0, 'off') ; delmenu(0, 'Outils') ; delmenu(0, 'Edit') ; delmenu(0, '?') ;
screen = get(0, 'screensize_px') ; f.figure_size = floor(screen(3:4)/sqrt(2)) ; f.figure_position = floor((screen(3:4) - f.figure_size)/2) ;

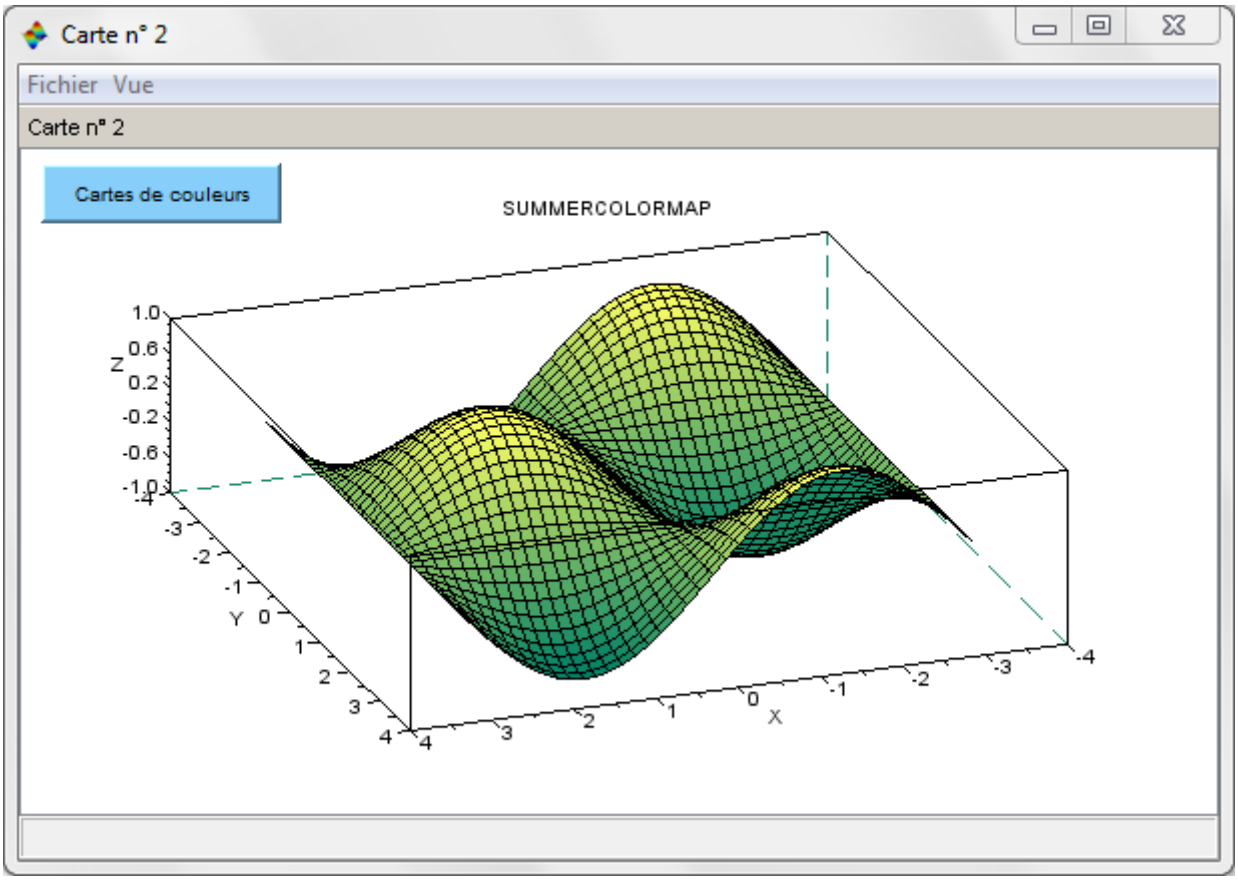
//--- Cartes de couleurs
COLORMAP = list('SPRINGCOLORMAP','SUMMERCOLORMAP','AUTUMNCOLORMAP','WINTERCOLORMAP',...
                'WHITECOLORMAP','GRAYCOLORMAP','BONECOLORMAP','COPPERCOLORMAP',...
                'PINKCOLORMAP','OCEANCOLORMAP','RAINBOWCOLORMAP','JETCOLORMAP',...
                'HSVCOLORMAP','COOLCOLORMAP','HOTCOLORMAP') ;
colormap = list(springcolormap(32),summercolormap(32),autumncolormap(32),wintercolormap(32),...
                whitecolormap(32),graycolormap(32),bonecolormap(32),coppercolormap(32),...
                pinkcolormap(32),oceancolormap(32),rainbowcolormap(32),jetcolormap(32),...
                hsvcolormap(32),coolcolormap(32),hotcolormap(32)) ;

//--- Surface 3D
plot3d()
n = 1 ; f.figure_name = 'Carte n° ' + string(n) ; xtitle(COLORMAP(n)) ; f.color_map = colormap(n) ;

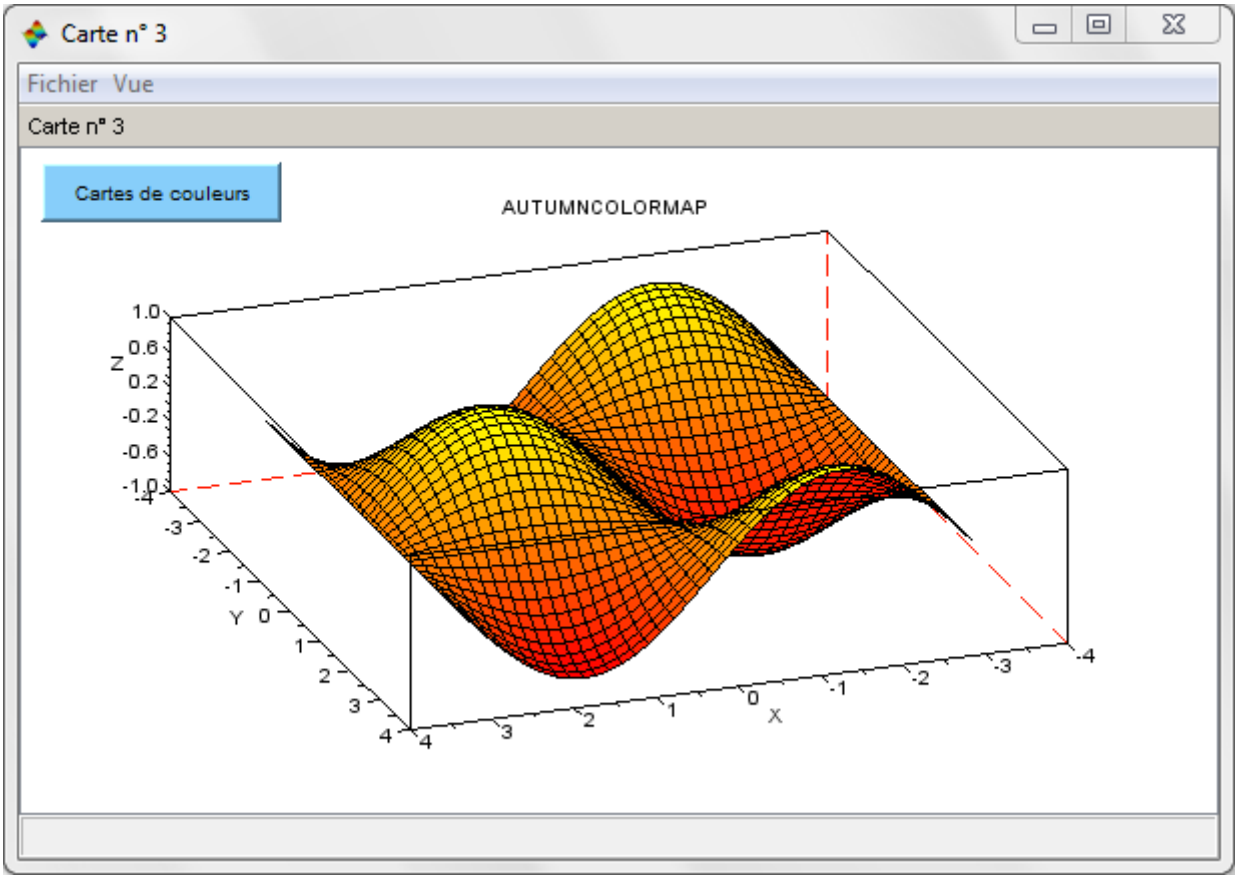
//--- Objet uimenu
menu = uimenu(f, 'label', 'Vue', 'callback', 'view = a.rotation_angles ; ...
              in = x_mdialog(''Angles de rotation'',[''alpha :'',''theta :''],string(view)) ; ...
              if in <> [] & isnum(in) then, a.rotation_angles = evstr(in), end') ;

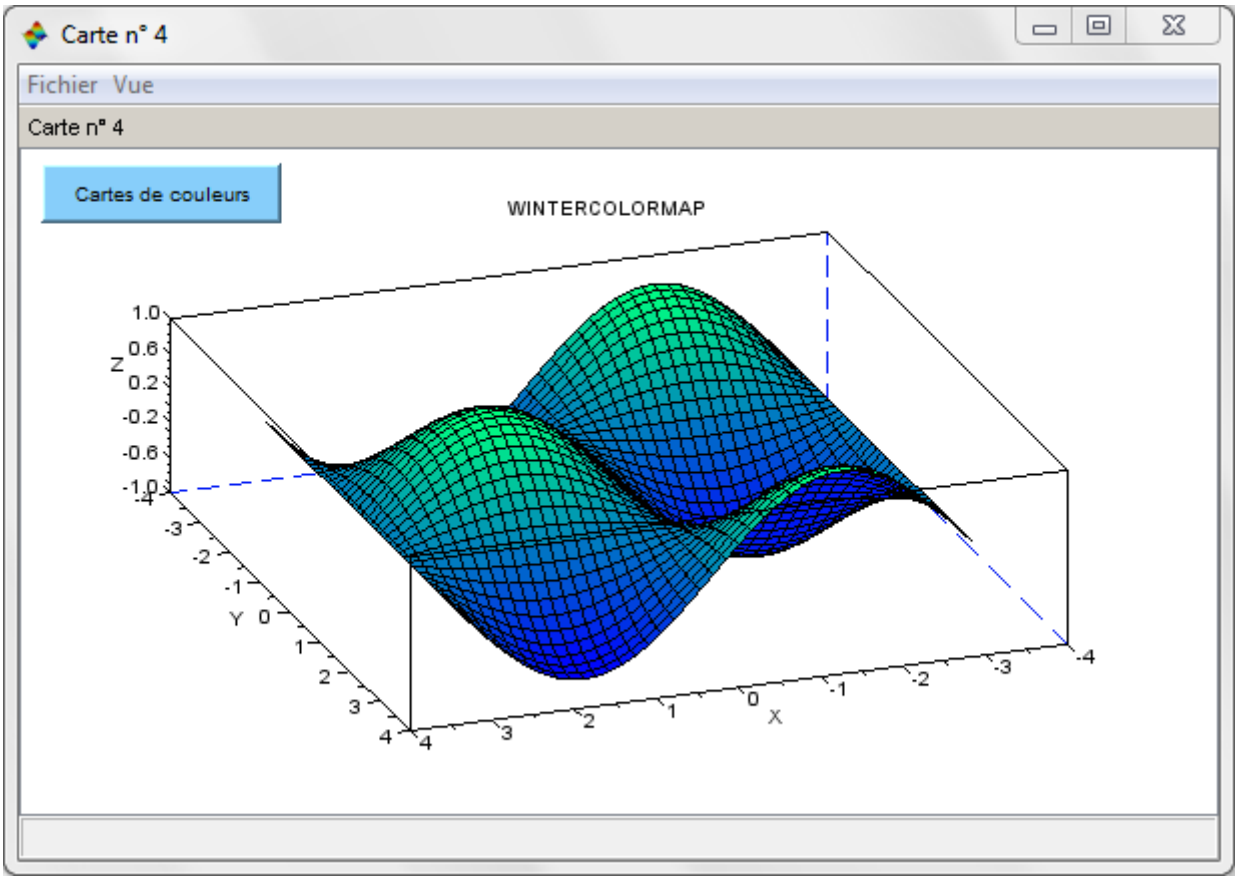
//--- Objet uicontrol
button = uicontrol(f, 'style', 'pushbutton', 'string', 'Cartes de couleurs', 'position', [10,f.figure_size(2)-140,120,30], ...
                   'backgroundcolor', [135,206,250]/255, 'callback', ...
                   'n = max(1,modulo(n+1,16)) ; f.figure_name = ''Carte n° '' + string(n) ; ...
                   xtitle(COLORMAP(n)) ; f.color_map = colormap(n)') ;
```


Carte n° 1
Fichier Vue
Carte n° 1
Cartes de couleurs
SPRINGCOLORMAP
Z
1.0
0.6
0.2
-0.2
-0.6
-1.0
Y
X


Carte n° 2
Fichier Vue
Carte n° 2
Cartes de couleurs
SUMMERCOLORMAP
Z
1.0
0.6
0.2
-0.2
-0.6
-1.0
Y
X

Mathématique & Informatique





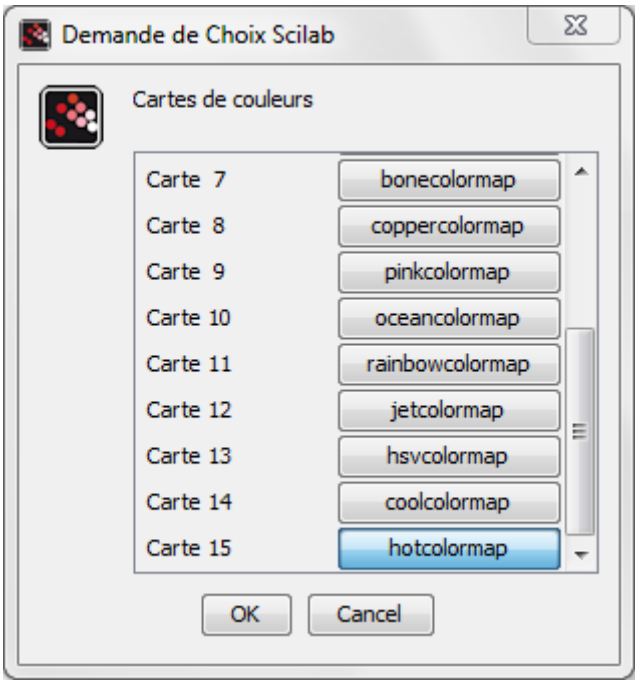
➪ Les cartes de couleurs

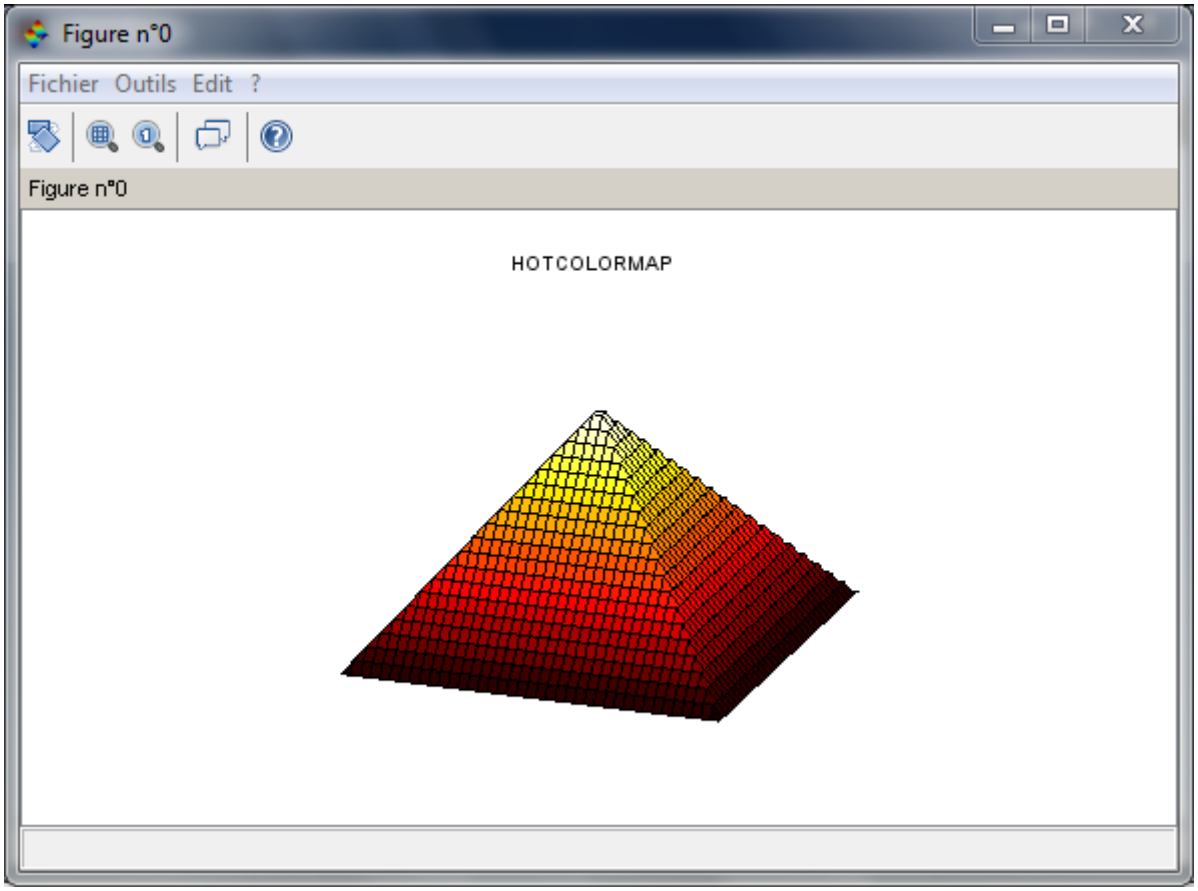
La version 5 de Scilab offre 15 cartes de couleurs prédéfinies. Un programme d'essai :

```
function z = pyramide(x, y)
  z = h*(1 - max(abs([x/a, y/b])))
endfunction
a = 230/2 ; b = 230/2 ; h = 137 ;
x = linspace(-a, +a, 40) ; y = linspace(-b, +b, 40) ;
fplot3d1(x, y, pyramide, theta = 20, alpha = 70, flag = [7, 6, 0]) ;

c01 = list('  Carte  1',0,'springcolormap') ;
c02 = list('  Carte  2',0,'summercolormap') ;
c03 = list('  Carte  3',0,'autumncolormap') ;
c04 = list('  Carte  4',0,'wintercolormap') ;
c05 = list('  Carte  5',0,'whitecolormap') ;
c06 = list('  Carte  6',0,'graycolormap') ;
c07 = list('  Carte  7',0,'bonecolormap') ;
c08 = list('  Carte  8',0,'coppercolormap') ;
c09 = list('  Carte  9',0,'pinkcolormap') ;
c10 = list('  Carte 10',0,'oceancolormap') ;
c11 = list('  Carte 11',0,'rainbowcolormap') ;
c12 = list('  Carte 12',0,'jetcolormap') ;
c13 = list('  Carte 13',0,'hsvcolormap') ;
c14 = list('  Carte 14',0,'coolcolormap') ;
c15 = list('  Carte 15',0,'hotcolormap') ;

f = gcf() ;
f.figure_position = [0,0] ; f.figure_size = [500,500] ;
while %t
  rep = x_choices('Cartes de couleurs', list(c01,c02,c03,c04,c05,c06,c07,c08,c09,c10,c11,c12,c13,c14,c15)) ;
  if rep == [] then, break, end
  n = find(rep == 1) ;
  if length(n) == 1 then
    select n
      case 1,  xtitle('SPRINGCOLORMAP') ; f.color_map = springcolormap(32) ;
      case 2,  xtitle('SUMMERCOLORMAP') ; f.color_map = summercolormap(32) ;
      case 3,  xtitle('AUTUMNCOLORMAP') ; f.color_map = autumncolormap(32) ;
      case 4,  xtitle('WINTERCOLORMAP') ; f.color_map = wintercolormap(32) ;
      case 5,  xtitle('WHITECOLORMAP') ; f.color_map = whitecolormap(32) ;
      case 6,  xtitle('GRAYCOLORMAP') ; f.color_map = graycolormap(32) ;
      case 7,  xtitle('BONECOLORMAP') ; f.color_map = bonecolormap(32) ;
      case 8,  xtitle('COPPERCOLORMAP') ; f.color_map = coppercolormap(32) ;
      case 9,  xtitle('PINKCOLORMAP') ; f.color_map = pinkcolormap(32) ;
      case 10, xtitle('OCEANCOLORMAP') ; f.color_map = oceancolormap(32) ;
      case 11, xtitle('RAINBOWCOLORMAP') ; f.color_map = rainbowcolormap(32) ;
      case 12, xtitle('JETCOLORMAP') ; f.color_map = jetcolormap(32) ;
      case 13, xtitle('HSVCOLORMAP') ; f.color_map = hsvcolormap(32) ;
      case 14, xtitle('COOLCOLORMAP') ; f.color_map = coolcolormap(32) ;
      case 15, xtitle('HOTCOLORMAP') ; f.color_map = hotcolormap(32) ;
    end
  end
end
```


Demande de Choix Scilab
Cartes de couleurs
Carte 7
bonecolormap
Carte 8
coppercolormap
Carte 9
pinkcolormap
Carte 10
oceancolormap
Carte 11
rainbowcolormap
Carte 12
jetcolormap
Carte 13
hsvcolormap
Carte 14
coolcolormap
Carte 15
hotcolormap
OK
Cancel


Figure n°0
Fichier Outils Edit ?
Figure n°0
HOTCOLORMAP

⇨ Les fichiers d'Entrées/Sorties

Généralités.

Il existe essentiellement 2 modèles de fichiers :

- Le fichier séquentiel (SEQUENTIAL) : on accède aux enregistrements les uns à la suite des autres. Le temps d'accès peut être long si l'on doit parcourir une partie du fichier pour retrouver un enregistrement particulier.

- Le fichier indexé (INDEXED) : on accède à un enregistrement grâce à une clé (unique) qui le caractérise. Le temps d'accès est très court, mais le système doit gérer l'ensemble des clés dans un fichier spécifique.

Primitives usuelles de manipulation des fichiers :
Ouverture (OPEN) — Fermeture (CLOSE) — Fin de fichier (EOF).
Lecture (READ) — Ecriture (WRITE) — Suppression (DELETE).

Remarque :
En plus de la clé primaire, les enregistrements peuvent être référencés par des clés secondaires, éventuellement identiques pour des enregistrements différents. Elles permettent des accès multicritères (voir base de données).
Par exemple, pour un fichier d'individus, la clé primaire peut être le numéro d'INSEE, et la clé secondaire peut être le patronyme (il est possible que 2 personnes aient le même nom).

Avec Scilab nous n'avons pas, en principe, à organiser de grandes quantités de données sur disque (fichiers de plusieurs milliers, voire millions d'enregistrements). On se contente donc d'utiliser des fichiers séquentiels avec des fonctions similaires à celles du langage C : le préfixe "m" (management) remplace le préfixe "f" (file).
Les fichiers sont de type texte (chaînes de caractères regroupés en lignes) ou binaire (suite d'octets mis bout à bout).
Ils sont accessibles en lecture (r), écriture (w), ajout (a), ou mise à jour (r+, w+, a+).

Les principales fonctions Scilab :

- Ouverture / Fermeture : *mopen* / *mclose*
- Lecture : *mget*, *mgetl*, *mgetstr*, *mfscanf*
- Ecriture : *mput*, *mputl*, *mputstr*, *mfprintf*
- Positionnement / Fin de fichier : *mseek*, *mtell* / *meof*
- Suppression : *mdelete*

Citons également les fonctions *read* et *write*, issues du FORTRAN, qui permettent respectivement de lire et d'écrire une matrice (p,q) dans un fichier.
⇨ Pour plus de détails, consulter le dossier *Entrées/Sorties* en cliquant sur le menu *? → Aide de Scilab*

Exemple :
Le programme suivant crée un fichier "données" dans le répertoire courant, écrit la valeur des 10 premiers entiers, ferme le fichier, puis l'ouvre en lecture pour calculer et afficher la somme "s" des valeurs enregistrées, et termine en supprimant le fichier.

```
mopen('données', 'wb') ;
for k = 1:10
   mput(k, 'i') ;
end
mclose() ;
mopen('données', 'rb') ;
s = 0 ;
while meof() == 0
   x = mget(1,'i') ;
   if x <> [] then, s = s + x ; end
end
disp(s) ;
mclose() ;
mdelete('données') ;
```



données

⇨ Les fichiers sources

Un programme entier est enregistré dans un fichier d'extension •sce :



programme.sce

Le texte du programme est saisi avec SciNotes et le fichier •sce est créé avec le menu : *Fichier* → *Enregistrer*

Une fonction seule est enregistrée dans un fichier d'extension •sci :



fonction.sci

Le texte de la fonction est saisi avec SciNotes et le fichier •sci est créé avec le menu : *Fichier* → *Enregistrer*

⇨ Le répertoire courant

Lorsqu'on lance Scilab en double-cliquant sur un raccourci Windows placé sur le Bureau, le répertoire courant est celui spécifié dans le champ *Démarrer dans* qui apparaît dans l'onglet *Raccourci* après avoir effectué un clic droit sur l'icône du programme et sélectionné la commande *Propriétés*.
Pour exécuter un fichier "programme.sce" à partir de la fenêtre Console, on utilise au choix la souris avec le menu *Fichier* → *Exécuter* ... ou le clavier avec l'instruction : - -> *exec("programme.sce", -1)*
Si le fichier "programme.sce" n'est pas sous le répertoire courant utilisé par Scilab, il faut écrire son chemin d'accès complet sous la forme : - -> *exec("C:\Users\...\programme.sce", -1)*
Le répertoire courant est modifiable avec le menu *Fichier* → *Changer le répertoire courant* ...

⇨ L'éditeur SciNotes

Scilab dispose de son propre éditeur pour écrire les scripts des programmes. On le lance en cliquant sur la première icône de la barre d'outils : , ou en cliquant sur le menu : *Applications* → *SciNotes* , ou encore en saisissant au clavier la commande *editor*. Particularités de SciNotes :

1. coloration du texte
2. indentation automatique
3. exécution du programme

La coloration du texte n'a pas uniquement un but esthétique, elle permet en outre de véhiculer une information pour déceler des erreurs lors de la saisie. Par exemple, une variable s'affichera en bleu si son nom est celui d'une fonction connue ; on évite ainsi toute confusion, sans avoir à connaître la liste des fonctions Scilab.
L'indentation automatique est utile pour dégager la structure générale d'un programme et accroître sa lisibilité.
En décalant les instructions, on distingue plus facilement les imbrications des boucles et on repère les erreurs de syntaxe.
L'exécution du programme peut s'effectuer sans sortir de l'éditeur, dès la fin de la saisie, en cliquant sur le menu *Exécuter*.
Le programme étant interprété, il est rapide à déboguer par des éditions et des exécutions successives, sans passer par la phase de compilation.
Attention à l'encodage des fichiers sources. Par défaut, lors de l'installation de Scilab, l'éditeur SciNotes utilise le jeu de caractères UNICODE UTF-8. On peut changer de format, par exemple choisir Windows 1252, en cliquant sur *Edition* → *Préférences*, puis en déroulant la liste en regard de *Encodage de fichier par défaut* , et finalement en sélectionnant *Windows 1252 (ANSI)*.

Mathématique & Informatique

Les expressions régulières avec l'éditeur SciNotes de Scilab

⇨ **http://prevert.upmf-grenoble.fr/Prog/Java/CoursJava/expressionsRegulieres.html**

Dans un éditeur, la fonction "Expression régulière" (ou "Expression rationnelle" ou encore "Pattern") est un outil très utile pour effectuer des recherches et des remplacements plus ou moins complexes dans tout ou partie du texte.
La symbolique adoptée par SciNotes est celle du langage Javascript :

^	le début de ligne
$	la fin de ligne
c	le caractère c sauf caractère spécial
\c	le caractère c y compris les caractères spéciaux (,), [,], {, }, \, ^, $, ., *
.	un caractère quelconque
[...]	un caractère appartenant à l'ensemble entre crochets
[^...]	un caractère n'appartenant pas à l'ensemble entre crochets
{n}	la répétition n fois du caractère précédent
*	la répétition un nombre quelconque de fois du caractère précédent
(...)	la capture d'une sous expression d'occurrence i référencée par la suite \i

Après avoir chargé le texte à modifier dans le buffer de SciNotes, cliquer sur *Rechercher* dans la barre de menus, puis sur *Rechercher/Remplacer*. Dans la petite fenêtre qui apparaît, cocher la case *Expressions rationnelles*.
Saisir l'expression régulière dans la zone *Rechercher* et le texte de substitution dans la zone *Remplacer*.

Exemples

La notation ∪ désignera un espace (caractère obtenu en appuyant sur la barre d'espace).

	Rechercher	Remplacer
Ajouter 3 espaces en début de ligne	^	∪∪∪
Supprimer 3 espaces en début de ligne	^∪∪∪	
Supprimer les commentaires	//.*	
Remplacer par un commentaire toutes les lignes débutant par la lettre a et se terminant par 2 chiffres	^a.*[0-9]{2}$	//commentaire

⇨ L'overloading

L'overloading — la surcharge en français — s'applique à des fonctions, à des opérateurs, ou à des affichages.

Overloading des fonctions.

Le même nom de fonction peut servir à effectuer des traitements différents.
Soit une fonction **f** déjà définie dans Scilab. La fonction **f** surchargée s'écrira :

function [résultats] = %*type*_**f** (données)
|
|
endfunction

où *type* est un caractère de code qui représente un nouveau type d'argument :
type = s pour un réel ou un complexe, *type* = c pour une chaîne, *type* = b pour un booléen, *type* = l pour une liste, ...

L'overloading des fonctions améliore la lisibilité des programmes en adoptant le même nom pour des fonctions qui présentent des similitudes.

Exemple :

La fonction *maxi* habituelle renvoie le plus grand élément d'une matrice de réels.
Définissons la fonction *maxi* surchargée qui renvoie la plus grande longueur d'une matrice de chaînes de caractères :

```
function l = %c_maxi(S)
  l = maxi(length(S))
endfunction

x = maxi([0 , 2 ; -30 , 1.5])
y = maxi(['a' , 'abc' ; '' , '+z'])
```

Suivant le type d'argument utilisé, l'interpréteur Scilab fera la différence entre la fonction "maxi" habituelle et la fonction "maxi" surchargée. On obtient : x = 2 (habituel) et y = 3 (surcharge).

Overloading des opérateurs.

Il est possible de surcharger les opérateurs +, -, *, /, ^, ==, <>, ~, ...

Pour un opérateur binaire :

function résultat = %*type_opcode_type* (donnée1, donnée2)
|
|
endfunction

Pour un opérateur unaire :

function résultat = %*type_opcode* (donnée)
|
|
endfunction

Exemple :

Définissons l'**a**ddition et la **m**ultiplication **b**ooléennes :

```
function z = %b_a_b(x,y)
   z = x | y
endfunction
```

```
function z = %b_m_b(x,y)
   z = x & y
endfunction
```

```
x = %t ; y = %f ;
z1 = x | (x&~y)
z2 = x + x*~y
```

Sans overloading, on obtient : z1 = T et z2 = 2
Avec overloading, on obtient : z1 = z2 = T

Donc, en surchargeant l'opérateur +, l'addition dans Scilab s'applique à 3 types d'opérandes :
1) les nombres : 1 + 2 == 3
2) les chaînes : 'a' + 'b' == 'ab'
3) les booléens : %t + %f == %t

Overloading des affichages.

L'instruction *disp* (display) peut être surchargée pour choisir le format d'affichage d'une liste typée.

function %NOM_*p*(tl)
|
|
endfunction

tl = *tlist*(['NOM', 'nom$_1$', ... , 'nom$_n$'], v$_1$, ... , v$_n$)
disp(tl)

Exemple :

```
function %Tarifs_p(tl)
   for i = 1:size(tl.article,'r')
      disp(tl.article(i) + ' = ' + string(tl.prix(i)) + ' euros')
   end
endfunction
```

```
tl = tlist(['Tarifs','article','prix'], ['article1' ; 'article2' ; 'article3'], [10 ; 20 ; 30]) ;
disp(tl)
```

On obtient :

article1 = 10 euros

article2 = 20 euros

article3 = 30 euros

Sans overloading, c'est-à-dire sans définir la fonction %Tarifs_p, l'instruction disp(tl) affiche la liste tl dans son intégralité ; les valeurs de tl(1), tl(2) et tl(3) apparaissent à l'écran dans la fenêtre Console.

⇨ Les commandes annexes

- **clear**

En tapant *clear* après la flèche de l'invite de commande (- ->), toutes les variables et les fonctions définies lors des exécutions précédentes sont effacées. Scilab se retrouve dans son environnement initial.

- **lines**

La fenêtre Console est automatiquement paginée en *nl* lignes et *nc* colonnes, *nl* et *nc* sont 2 entiers qui dépendent de la dimension de la fenêtre. En entrant la commande *lines(0)*, la pagination des lignes est supprimée, le message `[Continue display? n (no) to stop, any other key to continue]` n'apparaît plus et les données pourront défiler sans interruption.

- **stacksize**

La taille de la pile Scilab est initialisée à 10^7 mots (double-précision). Si cette valeur ne convient pas, on peut l'augmenter ou la diminuer à *n* mots en utilisant la commande *stacksize(n)*.

- **funcprot**

La commande *funcprot(0)* permet d'éviter le message `Attention: redéfinition de la fonction:` qui s'affiche lorsqu'on charge dans Scilab le script d'une fonction déjà présente.

- **warning**

La commande *warning('off')* est utilisée pour désactiver les messages d'avertissement (erreurs non fatales).
La commande *warning('on')* les rétablis.

⇨ Les itérations

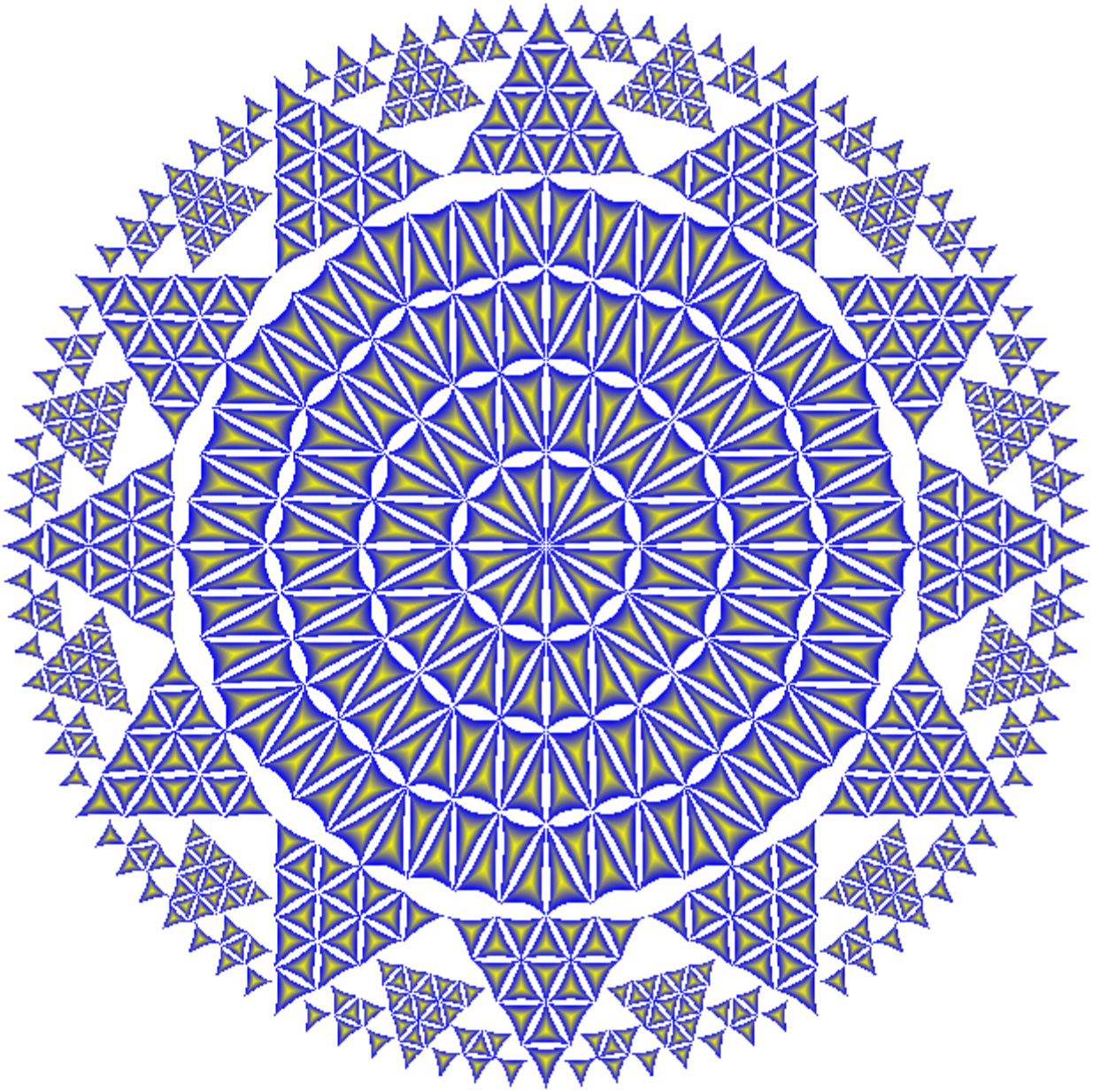
Les 2 boucles classiques, celle où l'on entre avec un test et celle où l'on entre sans test, s'écrivent :

tantque **condition** faire	répéter
|	|
|	|
|	|
|	|
|	|
fin	jusqu'à **condition**

Elles se programment, avec Scilab, de la manière suivante :

while **condition**	*while %t*
|	|
|	|
|	|
|	|
|	*if* **condition** *then, break, end*
end	*end*

Mathématique & Informatique



ALGORITHMES DE TRI

Les méthodes de tri sont nombreuses. Parmi les plus connues, citons : le tri à bulles (Bubble sort), le tri par insertion (Insertion sort), le tri de **Shell** (Shell sort), le tri fusion (Merge sort), le tri par tas (Heap sort), le tri rapide (Quick sort). Les trois derniers tris sont définis de manière récursive ; l'algorithme de tri rapide, appelé aussi tri par segmentation, est dû à **Tony Hoare** (1960). Détaillons cet algorithme. Soit un tableau *A* de *n* éléments, pour trier *A* dans l'ordre croissant on effectue l'appel *tri*(*1*,*n*) à la fonction suivante :

```
fonction tri(inf,sup)
|   si inf < sup alors
|   |   k = segmentation(inf,sup)
|   |   tri(inf,k-1)
|   |   tri(k+1,sup)
|   fin
fin

fonction k = segmentation(inf,sup)
|   j = inf + 1
|   k = sup
|   pivot = A(inf)
|   tantque j ≤ k faire
|   |   si A(j) ≤ pivot alors
|   |   |   j = j + 1
|   |   sinon
|   |   |   tantque A(k) > pivot faire
|   |   |   |   k = k - 1
|   |   |   fin
|   |   |   si j < k alors
|   |   |   |   échange(A(j),A(k))
|   |   |   |   j = j + 1
|   |   |   |   k = k - 1
|   |   |   fin
|   |   fin
|   fin
|   échange(A(inf),A(k))
fin
```

Dans la fonction segmentation précédente, il est possible de modifier le test de comparaison des éléments de *A* afin de trier, selon les critères de son choix, un tableau ou une liste de données qui ne sont pas des nombres réels. Pour cela, on définit une fonction de comparaison qui est une application de $A \times A$ dans {-1, 0, 1} permettant d'établir une relation d'ordre entre deux éléments quelconques e_1 et e_2 de *A*, autrement dit de distinguer les cas $e_1 < e_2$, $e_1 = e_2$, $e_1 > e_2$.

On souhaite classer un ensemble de *n* nombres complexes $z_1 , z_2 , \ldots , z_n \in \mathbb{C}$, de la manière suivante :

1) dans l'ordre croissant de leurs parties réelles (prioritairement)
2) dans l'ordre croissant de leurs parties imaginaires (secondairement)

Ainsi, pour 2 nombres complexes quelconques z_1 et z_2 :

$$z_1 = a_1 + b_1 i \text{ , } z_2 = a_2 + b_2 i : \quad \begin{cases} a_1 < a_2 \Rightarrow z_1 < z_2 \\ a_1 = a_2 \Rightarrow \begin{cases} b_1 < b_2 \Rightarrow z_1 < z_2 \\ b_1 = b_2 \Rightarrow z_1 = z_2 \\ b_1 > b_2 \Rightarrow z_1 > z_2 \end{cases} \\ a_1 > a_2 \Rightarrow z_1 > z_2 \end{cases}$$

Lorsque 2 nombres complexes ont la même partie réelle, on compare leurs parties imaginaires.

L'ensemble $\mathbb{C}$ est totalement ordonné : la partie réelle des nombres complexes l'emportant sur la partie imaginaire et les valeurs positives sur les valeurs négatives.

Appelons *csort* la fonction qui trie un tableau A de nombres complexes.
Supposons que A = [*-1 – i, 1 - 3i, 0, 3 + 4i, 3, 1 - 3i, -1, -i, 1 + 5i, -1 + i, -2, 1, -5i, 1, 1 + i, 1 – i, i*]
Pour ordonner les éléments de A, on écrit :

```
A = [-1 - %i, 1 - 3*%i, 0, 3 + 4*%i, 3, 1 - 3*%i, -1, -%i, 1 + 5*%i, -1 + %i, -2, 1, -5*%i, 1, 1 + %i, 1 - %i, %i] ;
B = csort(A) ;
disp(A,'---- Tableau non trié')
disp(B,'---- Tableau trié')
```

On obtient :

```
---- Tableau non trié
- 1. - i    1. - 3.i    0    3. + 4.i    3.    1. - 3.i    - 1.    - i    1. + 5.i    - 1. + i    - 2.    1.    - 5.i    1.    1. + i    1. - i    i

---- Tableau trié
- 2.    - 1. - i    - 1.    - 1. + i    - 5.i    - i    0    i    1. - 3.i    1. - 3.i    1. - i    1.    1.    1. + i    1. + 5.i    3.    3. + 4.i
```

Le programme de la fonction *csort* est donné page suivante.

Application

Utilisons *csort* pour trier les racines d'un polynôme.
Par exemple, l'instruction *csort(roots(poly([-1,zeros(1,9),1],'x','c')))* fournit les racines dixièmes de l'unité :

```
- 1.
- 0.8090170 - 0.5877853i
- 0.8090170 + 0.5877853i
- 0.3090170 - 0.9510565i
- 0.3090170 + 0.9510565i
  0.3090170 - 0.9510565i
  0.3090170 + 0.9510565i
  0.8090170 - 0.5877853i
  0.8090170 + 0.5877853i
  1.
```

Tri par segmentation

```
function B = csort(A)
    function tri(inf,sup)
        if inf < sup then
            k = segmentation(inf,sup)
            tri(inf,k-1)
            tri(k+1,sup)
        end
    endfunction
    function k = segmentation(inf,sup)
        global B
        j = inf + 1
        k = sup
        pivot = B(inf)
        while j <= k
            if compare(B(j),pivot) < 1 then
                j = j + 1
            else
                while compare(B(k),pivot) == 1
                    k = k - 1
                end
                if j < k then
                    temp = B(j)
                    B(j) = B(k)
                    B(k) = temp
                    j = j + 1
                    k = k - 1
                end
            end
        end
        temp = B(inf)
        B(inf) = B(k)
        B(k) = temp
    endfunction
    global B
    B = A
    n = length(B)
    tri(1,n)
endfunction
```

Fonction de comparaison

```
function c = compare(z1,z2)
    if real(z1) < real(z2) then
        c = -1
    elseif real(z1) > real(z2) then
        c = 1
    else
        if imag(z1) < imag(z2) then
            c = -1
        elseif imag(z1) > imag(z2) then
            c = 1
        else
            c = 0
        end
    end
endfunction
```

Notes

- La fonction de comparaison retourne 0 en cas d'égalité, -1 en cas d'infériorité et 1 en cas de supériorité.

$$\begin{cases} z_1 < z_2 \;\Rightarrow\; compare(z_1, z_2) = -1 \\ z_1 = z_2 \;\Rightarrow\; compare(z_1, z_2) = 0 \\ z_1 > z_2 \;\Rightarrow\; compare(z_1, z_2) = 1 \end{cases}$$

  En échangeant les valeurs -1 et 1 dans la fonction de comparaison, les données seront triées dans l'ordre inverse. Après avoir trié le tableau B, l'ordre peut être inversé en effectuant $\mathrm{E}(n/2)$ échanges $B(i) \leftrightarrow B(n - i + 1)$.

- Le tri par segmentation est d'autant plus performant que les tailles des deux sous-tableaux sont voisines. Il s'agit là d'un principe général de la récursivité où un problème est décomposé en sous-problèmes de tailles égales pour aboutir à un algorithme efficace. On améliore donc le temps moyen d'exécution du tri en attribuant au pivot de la procédure de segmentation, une estimation de la valeur médiane (au sens statistique du terme) du sous-tableau $B(inf{:}sup)$, au lieu de prendre systématiquement la première valeur, c'est-à-dire $B(inf)$. En pratique, cette estimation de la médiane sera obtenue à partir d'un échantillon d'au moins trois valeurs prises dans $B(inf{:}sup)$: la stratégie 3-médiane consiste à prendre comme pivot l'élément central issu du tri des trois valeurs $B(inf)$, $B([(inf + sup)/2])$ et $B(sup)$.

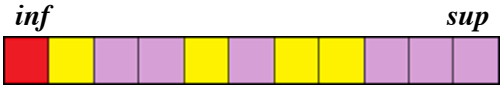
Mathématique & Informatique

Choix du pivot

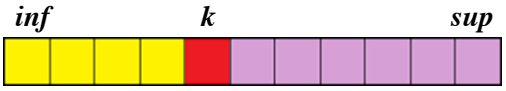
Le tri par segmentation consiste à trier partiellement des sous-tableaux $B(inf{:}sup)$ inclus dans le tableau de départ $B(1{:}n)$. Dans chaque sous-tableau $B(inf{:}sup)$, on choisit un élément $B(i)$ appelé pivot et on classe les autres éléments en 2 groupes :

1) Le groupe des éléments inférieurs ou égaux au pivot $B(i)$
2) Le groupe des éléments supérieurs ou égaux au pivot $B(i)$

Prenons un tableau $B(inf{:}sup)$ de 11 éléments où $B(inf)$ est le pivot. Supposons que 4 éléments soient inférieurs à $B(inf)$ et que 6 éléments soient supérieurs à $B(inf)$. Attribuons la couleur rouge au pivot, jaune aux éléments du groupe 1, mauve aux éléments du groupe 2. A l'entrée de la fonction *segmentation*, le tableau $B(inf{:}sup)$ est, par exemple, de la forme :



A la sortie de la fonction *segmentation*, les éléments du tableau $B(inf{:}sup)$ sont ordonnés de la façon suivante :



Après avoir effectué le classement des éléments de $B(inf{:}sup)$, le pivot occupe sa place définitive $B(k)$ dans le tableau $B(1{:}n)$. L'idéal serait que le pivot se retrouve à chaque fois au milieu du tableau $B(inf{:}sup)$, c'est-à-dire que $k = [(inf + sup)/2]$, mais comme on ne connait pas a priori la médiane de ce tableau, on prend pour pivot la médiane de l'ensemble restreint aux trois valeurs $B(inf)$, $B([(inf + sup)/2])$, $B(sup)$. Une autre solution consiste à choisir le pivot aléatoirement parmi les $sup - inf + 1$ valeurs $B(inf)$, … , $B(i)$, … , $B(sup)$. Pour intégrer l'une de ces deux méthodes dans le programme *csort* précédent, sans modifier sa structure, il suffit de remplacer l'instruction d'affectation *pivot = B(inf)* par les instructions adéquates de recherche du pivot :

Pivot 3-médiane

```
mid = floor((inf + sup)/2)
if compare(B(inf),B(sup)) == 1 then
    temp = B(sup)
    B(sup) = B(inf)
    B(inf) = temp
end
if compare(B(mid),B(sup)) == 1 then
    pivot = B(sup)
    B(sup) = B(inf)
    B(inf) = pivot
elseif compare(B(mid),B(inf)) == 1 then
    pivot = B(mid)
    B(mid) = B(inf)
    B(inf) = pivot
else
    pivot = B(inf)
end
```

Pivot aléatoire

```
i = grand(1,1,'uin',inf,sup)
pivot = B(i)
B(i) = B(inf)
B(inf) = pivot
```

Noter que les opérations d'affectation ($x = y$) sont peu coûteuses en temps de calcul par rapport aux appels à des fonctions externes (fonction *compare* pour des comparaisons ou fonction *grand* pour des nombres aléatoires).
Pour une étude des différents algorithmes de tri et leurs temps d'exécution, lire le document à l'adresse internet :
⇨ **http://infoscience.epfl.ch/record/101014/files/ramatouFinal.pdf**

Ordre de grandeur du temps d'exécution

Le programme ci-dessous utilise la fonction *csort* pour trier 10 000 nombres complexes générés de manière aléatoire :

```
n = 10000 ;
A = grand(1,n,'uin',-n,n) + grand(1,n,'uin',-n,n)*%i ;
timer() ;
B = csort(A) ;
printf('\n %.2f secondes',timer())
```

Le temps de calcul est d'environ 4 secondes.

Vers une procédure optimisée

Rappelons que tout programme récursif peut être traduit en un programme itératif équivalent. La fonction *csort* devient :

```
function B = csort(A)
    B = A
    inf = 1
    sup = length(B)
    stack = list(0)
    while length(stack) <> 0
        stack($) = null()
        while inf < sup
            j = inf + 1
            k = sup
            mid = floor((inf + sup)/2)
            if compare(B(inf),B(sup)) == 1 then
                temp = B(sup)
                B(sup) = B(inf)
                B(inf) = temp
            end
            if compare(B(mid),B(sup)) == 1 then
                pivot = B(sup)
                B(sup) = B(inf)
                B(inf) = pivot
            elseif compare(B(mid),B(inf)) == 1 then
                pivot = B(mid)
                B(mid) = B(inf)
                B(inf) = pivot
            else
                pivot = B(inf)
            end
            while j <= k
                if compare(B(j),pivot) < 1 then
                    j = j + 1
                else
                    while compare(B(k),pivot) == 1
                        k = k - 1
                    end
                    if j < k then
                        temp = B(j)
                        B(j) = B(k)
                        B(k) = temp
                        j = j + 1
                        k = k - 1
                    end
                end
            end
            temp = B(inf)
            B(inf) = B(k)
            B(k) = temp
            stack($+1) = sup
            sup = k - 1
        end
        if length(stack) <> 0 then
            inf = sup + 2
            sup = stack($)
        end
    end
endfunction
```

Cette version non récursive de la fonction *csort* permet, d'une part, de réduire la taille de la pile (on sauvegarde uniquement le paramètre *sup*) et, d'autre part, de diminuer le temps d'exécution (on gagne environ 20 % en temps de calcul).
De plus, sous cette forme, il est possible d'écrire la fonction *csort* en langage non récursif (langage assembleur notamment).

Evaluation du tri par segmentation

Soit un tableau A composé de n éléments.
Notons $C(n)$ le nombre d'appels à la fonction de comparaison des éléments de A lors du tri.
La segmentation nécessite de comparer le pivot aux $n-1$ autres éléments du tableau afin de déterminer sa position $A(k)$.
Par conséquent, d'après l'expression récursive du tri par segmentation, il existe un entier $k \in [1,n]$ tel que :

$$\begin{cases} C(n) = n-1+C(k-1)+C(n-k) \\ C(0) = 0 \end{cases}$$

A partir de cette relation de récurrence :

- le cas le plus favorable se produit lorsque $n = 2^m - 1$ et $k = 2^{m-1}$, on a :
  $C(n) = n-1+2C\left(\frac{n+1}{2}-1\right) \Rightarrow C(n) = (n+1)\log_2(n+1) - 2n \Rightarrow C(n) = O(n\log n)$
- le cas le plus défavorable se produit lorsque $k = 1$, on a :
  $C(n) = n-1+C(n-1) \Rightarrow C(n) = \frac{n(n-1)}{2} \Rightarrow C(n) = O(n^2)$

NB Dans $C(n) = O(f(n))$ le grand O est la notation de **Landau** signifiant que le rapport $\frac{C(n)}{f(n)}$ est borné quand $n \to +\infty$

Le nombre minimal de comparaisons du tri par segmentation est de l'ordre de $n \log n$, le nombre maximal est de l'ordre de n^2.

Considérons, pour simplifier, que le tableau A est constitué d'éléments différents et que les $n!$ permutations possibles de tous ses éléments sont équiprobables. Notons $C(n)$ le nombre moyen d'appels à la fonction de comparaison des éléments de A lors du tri. La probabilité pour que la fonction de segmentation retourne une valeur k (quelconque) est égale à $1/n$. On établit que :

$$\begin{cases} C(n) = \frac{1}{n}\sum_{k=1}^{n}\left[n-1+C(k-1)+C(n-k)\right] \\ C(0) = 0 \end{cases}$$

D'où : $nC(n) = n(n-1) + 2\sum_{k=0}^{n-1} C(k)$, et aussi : $(n-1)C(n-1) = (n-1)(n-2) + 2\sum_{k=0}^{n-2} C(k)$

Soustrayons membre à membre les deux identités précédentes : $nC(n) - (n+1)C(n-1) = 2(n-1)$

Divisons par $n(n+1)$: $\frac{C(n)}{n+1} - \frac{C(n-1)}{n} = \frac{2(n-1)}{n(n+1)}$

Posons $c_n = \frac{C(n)}{n+1}$, l'équation récurrente s'écrit : $\begin{cases} c_n = c_{n-1} + \frac{2(n-1)}{n(n+1)} \\ c_0 = 0 \end{cases}$

La solution est : $c_n = \sum_{i=1}^{n}\frac{2(i-1)}{i(i+1)} = \sum_{i=1}^{n}\left(\frac{2}{i} - \frac{4}{i} + \frac{4}{i+1}\right) = 2\sum_{i=1}^{n}\frac{1}{i} - 4 + \frac{4}{n+1}$

On en déduit : $C(n) = 2(n+1)\sum_{i=1}^{n}\frac{1}{i} - 4n$

On sait que : $\sum_{i=1}^{n}\frac{1}{i} - \log n \xrightarrow[n\,\infty]{} \gamma = 0.577\cdots$ (constante d'**Euler**)

Par suite : $C(n) \underset{n\,\infty}{\sim} 2n\log n - (4-2\gamma)n + 2\log n + 2\gamma$

Donc : $C(n) = O(n\log n)$ quand $n \to +\infty$

Le nombre moyen de comparaisons du tri par segmentation est de l'ordre de $n \log n$.

Mathématique & Informatique

Le tri par segmentation est un tri de complexité $O(nlogn)$ où n est la taille du tableau à trier, il est adapté aux grandes valeurs de n. Pour de petites valeurs de n ou pour des tableaux dont les éléments sont déjà presque ordonnés, on peut utiliser un tri par insertion de complexité $O(n^2)$.

Tri par insertion (version 1)

```
function B = csort(A)
    B = A
    n = length(B)
    for i = 1:n-1
        j = i
        temp = B(i+1)
        while (j >= 1) & compare(B(j),temp) == 1
            B(j+1) = B(j)
            j = j - 1
        end
        B(j+1) = temp
    end
endfunction
```

Tri par insertion (version 2)

```
function B = csort(A)
    B = A
    n = length(B)
    i = 1
    while i < n
        j = i
        temp = B(i+1)
        loop = %t
        while loop
            loop = %f
            if compare(B(j),temp) == 1 then
                B(j+1) = B(j)
                j = j - 1
                loop = (j >= 1)
            end
        end
        B(j+1) = temp
        i = i + 1
    end
endfunction
```

Dans la version 1 du programme de tri, l'écriture *while compare(B(j),temp) == 1 & (j >= 1)* provoquera une erreur d'exécution car le premier membre du test est toujours évalué en premier et le terme *B(j)* est indéfini lorsque *j* est nul. Par contre, l'instruction *while (j >= 1) & compare(B(j),temp) == 1* donne le résultat souhaité car l'interpréteur Scilab ne calcule pas le deuxième membre du test lorsque le premier est égal à *%f*, le résultat final étant nécessairement *%f*. Attention toutefois lors de la traduction de ce programme dans d'autres langages …
Dans la version 2, plus rigoureuse, l'introduction d'un booléen supplémentaire (*loop*) permet d'éviter ce problème.

A noter que plusieurs algorithmes de tri peuvent être réunis en un seul. Par exemple, dans le tri par segmentation, on peut ajouter un tri par insertion qui se révèle plus efficace pour trier les sous-tableaux *B*(*inf:sup*) de moins de 10 éléments.

Pour plus d'informations sur les tris et les méthodes de recherche associées, consulter *The Art of Computer Programming* (Volume 3 : Sorting and Searching) de **Donald Ervin Knuth** : ⇨ **http://www-cs-faculty.stanford.edu/~knuth/taocp.html**

Mathématique & Informatique

Médiane statistique

Par définition, la médiane statistique est la valeur qui sépare un ensemble de données en 2 groupes de même effectif : le groupe des valeurs inférieures ou égales à la médiane et le groupe des valeurs supérieures ou égales à la médiane.
Ainsi, pour un tableau trié T de n valeurs :

$$médiane = \begin{cases} T\left(\frac{n+1}{2}\right) & \text{si } n \text{ est impair} \\ \frac{T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}+1\right)}{2} & \text{si } n \text{ est pair} \end{cases}$$

La fonction de calcul de la médiane d'un tableau de nombres complexes ($z \in \mathbb{C}$) est :

```
function med = mediane(A)
    T = csort(A)
    n = length(T)
    if modulo(n,2) <> 0 then
        med = T((n+1)/2)
    elseif n <> 0 then
        med = (T(n/2) + T(n/2+1))/2
    else
        med = []
    end
endfunction
```

Par exemple :

```
A = [-1 - %i, 1 - 3*%i, 0, 3 + 4*%i, 3, 1 - 3*%i, -1, -%i, 1 + 5*%i, -1 + %i, -2, 1, -5*%i, 1, 1 + %i, 1 - %i, %i] ;
med = mediane(A) ;
disp(med,'médiane = ')
```

donne : *médiane* = 1. - 3.i

Remarque 1

La fonction Scilab *mean* fournit la moyenne d'un tableau de nombres complexes.
Dans l'exemple ci-dessus, on a : *moy = mean(A)* = 0.4705882 - 0.1176471i
Lorsqu'un tableau est constitué uniquement de nombres réels, la médiane peut être calculée avec la fonction Scilab *median* : *med = median(A)*. Le résultat est le même que celui donné par la fonction *mediane*.

Remarque 2

Le calcul de la médiane d'un tableau de n valeurs avec un tri par segmentation est de complexité $O(nlogn)$.
Le calcul de la moyenne d'un tableau de n valeurs est de complexité $O(n)$.
L'algorithme BFPRT (**Blum**, **Floyd**, **Pratt**, **Rivest**, **Tarjan**) permet de trouver en un temps linéaire le k-ième plus petit élément d'une liste de n éléments non triés. En particulier, pour $k = \frac{n+1}{2}$ (n impair) ou $k = \frac{n}{2}, \frac{n}{2}+1$ (n pair), on obtient un algorithme de complexité $O(n)$ pour calculer la médiane.

Signature d'une permutation

La fonction de calcul de la signature d'une permutation avec l'algorithme Quicksort est :

```
function sgn = signature(sigma)
    function tri(inf,sup)
        if inf < sup then
            k = segmentation(inf,sup)
            tri(inf,k-1)
            tri(k+1,sup)
        end
    endfunction
    function k = segmentation(inf,sup)
        global S m
        i = grand(1,1,'uin',inf,sup)
        j = inf + 1
        k = sup
        pivot = S(i)
        if i <> inf then
            S(i) = S(inf)
            S(inf) = pivot
            m = m + 1
        end
        while j <= k
            if S(j) <= pivot then
                j = j + 1
            else
                while S(k) > pivot
                    k = k - 1
                end
                if j < k then
                    temp = S(j)
                    S(j) = S(k)
                    S(k) = temp
                    m = m + 1
                    j = j + 1
                    k = k - 1
                end
            end
        end
        if k <> inf then
            temp = S(inf)
            S(inf) = S(k)
            S(k) = temp
            m = m + 1
        end
    endfunction
    global S m
    m = 0
    S = sigma
    n = length(S)
    tri(1,n)
    check = isequal(S,[1:n])
    if check then
        sgn = (-1)^m
    else
        sgn = []
    end
endfunction
```

Par exemple :

```
sigma = [24,2,38,34,3,33,36,16,23,9,40,4,39,19,26,18,12,29,21,6,10,15,25,5,28,8,30,31,32,14,1,35,7,37,22 ,17,27,11,20,13] ;
sgn = signature(sigma) ;
disp(sgn,'signature =')
```

donne : *signature* = - 1

Tri fusion de 2 tableaux

Lorsqu'on souhaite fusionner 2 tableaux triés A_1 et A_2 en un unique tableau trié A, on utilise un algorithme d'interclassement qui permet d'obtenir A en un temps linéaire.
Soient A_1 et A_2 deux tableaux de nombres complexes déjà triés, la fonction *cmerge* fournit le tableau trié $A = A_1 \cup A_2$.
Si A_1 et A_2 possèdent respectivement n_1 et n_2 nombres complexes, alors $A = A_1 \cup A_2$ possède $n = n_1 + n_2$ nombres complexes et la fonction *cmerge* effectue au plus $n - 1$ appels à la fonction *compare* pour former progressivement le tableau A.

```
function A = cmerge(A1,A2)
    n1 = length(A1)
    n2 = length(A2)
    n = n1 + n2
    A = zeros(1,n)
    i1 = 1
    i2 = 1
    i = 0
    while (i1 <= n1) | (i2 < = n2)
        i = i + 1
        if (i1 <= n1) & (i2 < = n2) then
            if compare(A1(i1),A2(i2)) == 1 then
                A(i) = A2(i2)
                i2 = i2 + 1
            else
                A(i) = A1(i1)
                i1 = i1 + 1
            end
        elseif i1 > n1 then
            A(i) = A2(i2)
            i2 = i2 + 1
        else
            A(i) = A1(i1)
            i1 = i1 + 1
        end
    end
endfunction
```

Par exemple :

```
A1 = [- 2, - 1 - %i, - 1, - 1 + %i, - 5*%i, - %i, 0, %i, 1 - 3*%i, 1 - 3*%i, 1 - %i, 1, 1, 1 + %i, 1 + 5*%i, 3, 3 + 4*%i] ;
A2 = [-1 + 2*%i, 1 - %i, 1, 1, 2, 3 + 5*%i] ;
A = cmerge(A1,A2) ;
disp(A1,'---- Tableau trié A1')
disp(A2,'---- Tableau trié A2')
disp(A,'---- Tableau trié A1 + A2')
```

donne le résultat suivant :

```
---- Tableau trié A1

- 2.    - 1. - i    - 1.    - 1. + i    - 5.i    - i    0    i    1. - 3.i    1. - 3.i    1. - i    1.    1.    1. + i    1. + 5.i    3.    3. + 4.i

---- Tableau trié A2

- 1. + 2.i    1. - i    1.    1.    2.    3. + 5.i

---- Tableau trié A1 + A2

- 2.    - 1. - i    - 1.    - 1. + i    - 1. + 2.i    - 5.i    - i    0    i    1. - 3.i    1. - 3.i    1. - i    1. - i    1.    1.    1.    1.    1. + i
1. + 5.i    2.    3.    3. + 4.i    3. + 5.i
```

Mathématique & Informatique

Le principe du tri fusion est simple : pour trier un tableau A de n éléments, on trie récursivement ses 2 moitiés et on fusionne le résultat. L'algorithme est de complexité $O(nlogn)$ dans tous les cas (*). Il s'agit donc d'un tri rapide. Par contre, ce type de tri ne s'effectue pas en place (en général) et sa mise en œuvre requiert un espace mémoire supplémentaire de même taille que celui occupé par le tableau à trier. Le plus souvent, cette contrainte ne constitue pas un obstacle car les systèmes informatiques actuels disposent d'une mémoire centrale rapide et de grande capacité. Le tri fusion effectue un minimum de comparaisons, il est d'autant plus approprié que la fonction de comparaison des éléments du tableau nécessite un temps de calcul important. Le tri fusion est également intéressant pour les systèmes temps réel puisqu'il permet de garantir un temps d'exécution limite non quadratique.

```
function B = mergesort(A)
    function B = sort(A)
        n = length(A)
        if n > 1 then
            m = floor(n/2)
            B1 = sort(A(1:m))
            B2 = sort(A(m+1:n))
            B = merge(B1,B2)
        else
            B = A
        end
    endfunction
    function B = merge(B1,B2)
        n1 = length(B1)
        n2 = length(B2)
        n = n1 + n2
        B = zeros(1,n)
        i1 = 1
        i2 = 1
        i = 0
        while (i1 <= n1) | (i2 < = n2)
            i = i + 1
            if (i1 <= n1) & (i2 < = n2) then
                if compare(B1(i1),B2(i2)) == 1 then
                    B(i) = B2(i2)
                    i2 = i2 + 1
                else
                    B(i) = B1(i1)
                    i1 = i1 + 1
                end
            elseif i1 > n1 then
                B(i) = B2(i2)
                i2 = i2 + 1
            else
                B(i) = B1(i1)
                i1 = i1 + 1
            end
        end
    endfunction
    B = sort(A)
endfunction
```

```
function B = quicksort(A)
    function sort(inf,sup)
        if inf < sup then
            k = split(inf,sup)
            sort (inf,k-1)
            sort (k+1,sup)
        end
    endfunction
    function k = split(inf,sup)
        global B
        i = grand(1,1,'uin',inf,sup)
        j = inf + 1
        k = sup
        pivot = B(i)
        B(i) = B(inf)
        B(inf) = pivot
        while j <= k
            if compare(B(j),pivot) < 1 then
                j = j + 1
            else
                while compare(B(k),pivot) == 1
                    k = k - 1
                end
                if j < k then
                    temp = B(j)
                    B(j) = B(k)
                    B(k) = temp
                    j = j + 1
                    k = k - 1
                end
            end
        end
        temp = B(inf)
        B(inf) = B(k)
        B(k) = temp
    endfunction
    global B
    B = A
    n = length(B)
    sort(1,n)
endfunction
```

Dans les 2 fonctions *mergesort* et *quicksort* ci-dessus, A et B désignent des tableaux d'éléments du type $[e_1, e_2, \ldots, e_n]$, par exemple les e_k sont des nombres complexes. En entrée le tableau A est constitué d'éléments dans un ordre quelconque, en sortie le tableau B est constitué des éléments de A triés dans l'ordre croissant. La fonction $compare(e_i,e_j)$ retourne -1, 0, 1 suivant que $e_i < e_j$, $e_i = e_j$, $e_i > e_j$ conformément à la relation d'ordre qui a été définie.

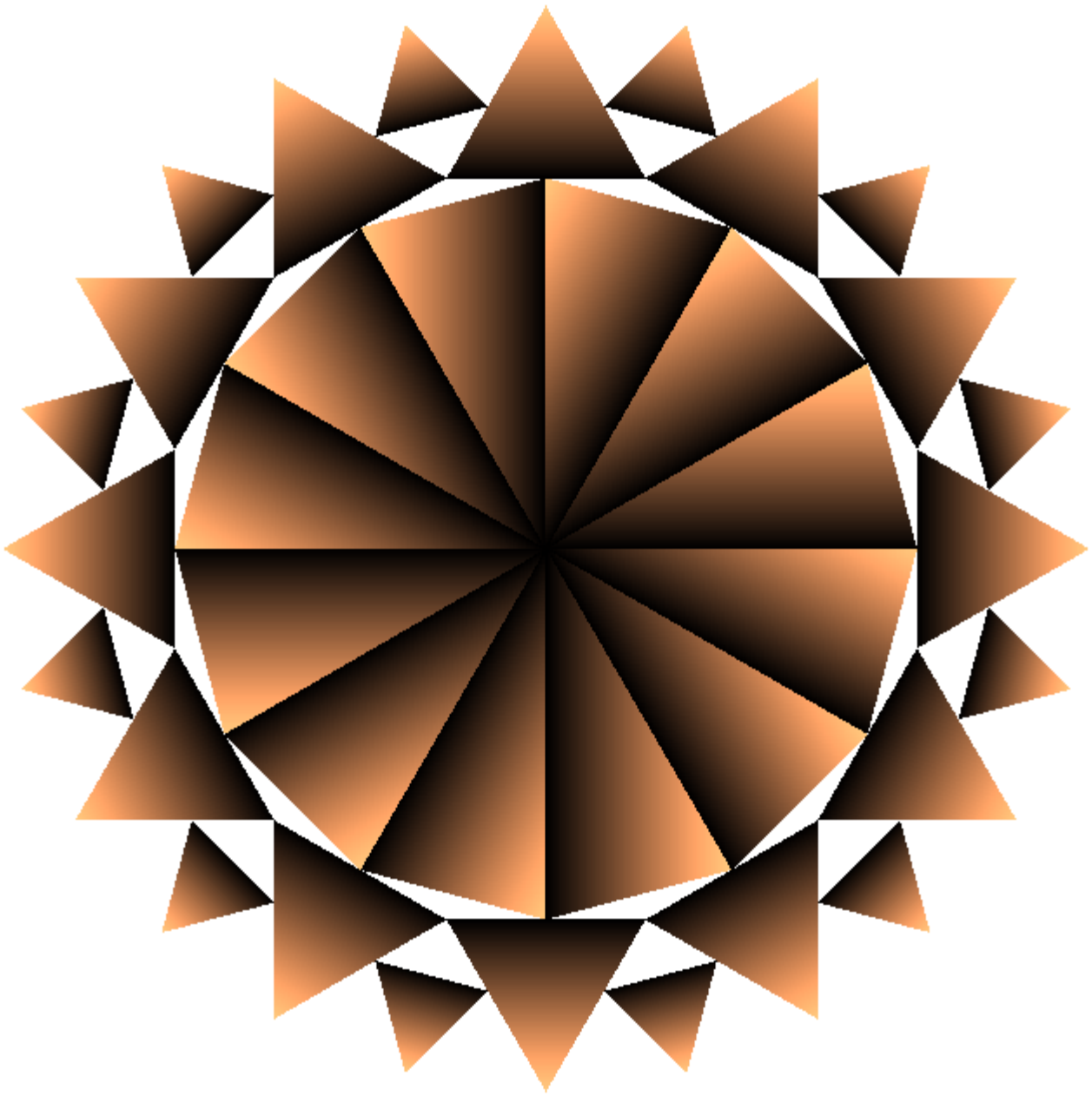
(*) Evaluation de la complexité d'une procédure récursive

Soit l'équation de récurrence : $\begin{cases} C(n) = aC(n/b) + f(n) \\ C(1) = c \end{cases}$ où a, b, c sont des constantes et $f(n)$ est une fonction de l'ordre de n.

On démontre que la solution $C(n)$ est de l'ordre de n si $a < b$, de l'ordre $n \log n$ si $a = b$, de l'ordre de $n^{\log_b a}$ si $a > b$.

Si $C(n)$ désigne le nombre d'appels à la fonction *compare* dans le programme *mergesort*, on a : $a = b = 2$, $f(n) \le n - 1$, $c = 0$. Il s'ensuit que le temps d'exécution du tri fusion d'un tableau de n éléments est proportionnel à $n \log n$.

Mathématique & Informatique



LA TOUR D'HANOÏ

Ce problème a été inventé par **Édouard Lucas** (1842 – 1891).

La légende

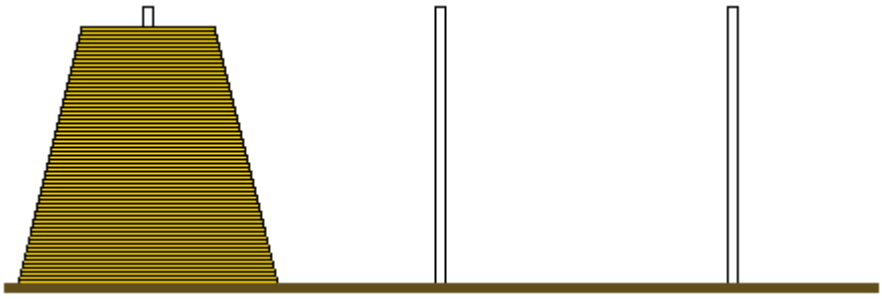
Dans le grand temple de Bénarès, au-dessous du dôme qui marque le centre du monde, on aperçoit trois aiguilles de diamant, plantées dans une dalle d'airain, hautes d'une coudée et grosses comme le corps d'une abeille. Sur l'une de ces aiguilles, le dieu PARABAVASTÛ enfila, au commencement des siècles, soixante-quatre disques d'or pur, le plus large reposant sur le bronze et les autres, de plus en plus étroits, superposés jusqu'au sommet. C'est la Tour Sacrée de VISHNOU. Nuit et jour, les bonzes se succèdent sur les marches de l'autel, occupés à transporter la Tour Sacrée, de la première aiguille sur la troisième, étage par étage, sans jamais intervertir, sans jamais s'écarter des règles immuables imposées par BRAHMA. Quand tout sera fini, la Tour et les Brahmes tomberont et ce sera la fin des mondes. (**Édouard Lucas**, Jeux Scientifiques ; p 3 ; Imp. Girard & Fils, 1889 ⇨ **http://www.math.harvard.edu/~elkies/hanoi.pdf**)

La solution

Les disques doivent être déplacés un par un : on prend un disque sur le sommet d'une pile et on le dépose sur un autre disque de diamètre supérieur ou sur un piquet vide. Désignons les 3 piquets par les lettres a, b, c. Pour transférer n disques du piquet a vers le piquet c, on commence par transférer n - 1 disques du piquet a vers le piquet b (le piquet c servant de piquet auxilliaire), puis on déplace le disque restant sur le piquet a vers le piquet c et on termine en transférant les n - 1 disques du piquet b vers le piquet c (le piquet a servant de piquet auxilliaire). Calculons le nombre D(n) des déplacements nécessaires :

$$\begin{aligned} D(n) &= 2\,D(n-1) + 1 \\ 2\,D(n-1) &= 2^2\,D(n-2) + 2 \\ &\cdots\cdots\cdots\cdots \\ 2^k\,D(n-k) &= 2^{k+1}\,D(n-k-1) + 2^k \\ &\cdots\cdots\cdots\cdots \\ 2^{n-2}\,D(2) &= 2^{n-1}\,D(1) + 2^{n-2} \\ 2^{n-1}\,D(1) &= 2^{n-1} \end{aligned}$$

$$D(n) = \sum_{k=0}^{n-1} 2^k = 2^n - 1$$



Admettons qu'un déplacement s'effectue en 1 seconde, il faut $2^{64} - 1$ secondes ≈ 585 milliards d'années pour transporter la tour !

L'algorithme

Le principe récurrent de résolution conduit à une fonction *hanoï* récursive.
La méthode d'élimination de la récursivité permet d'obtenir une fonction *hanoï* itérative.

```
fonction hanoï(a,b,c,n)
|   si n > 0 alors
|   |   hanoï(a,c,b,n - 1)
|   |   déplacer(a,c)
|   |   hanoï(b,a,c,n - 1)
|   fin
fin
```

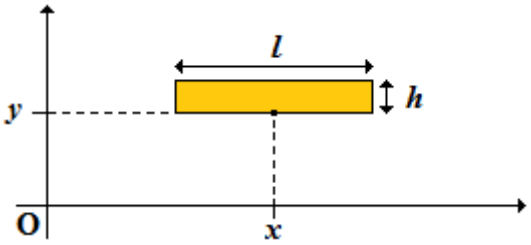
```
fonction hanoï(a,b,c,n)
|   k = 2
|   tantque k > 1 faire
|   |   k = k - 1
|   |   tantque k < 2^n faire
|   |   |   échanger(b,c)
|   |   |   k = k × 2
|   |   fin
|   |   tantque k × (k mod 2) > 1 faire
|   |   |   échanger(a,b)
|   |   |   k = k div 2
|   |   fin
|   |   si k > 1 alors
|   |   |   échanger(b,c)
|   |   |   déplacer(a,c)
|   |   |   échanger(a,b)
|   |   |   k = k + 2
|   |   fin
|   fin
fin
```

Mathématique & Informatique

Indications pour la compréhension du programme graphique

- Les caractéristiques d'un disque

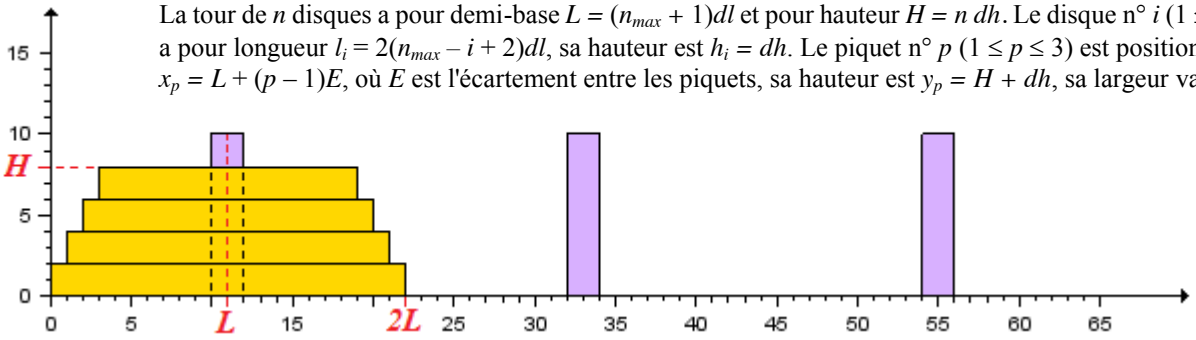
Un disque est un rectangle coloré défini par 6 valeurs réelles : x, y, l, h, c_1, c_2



x = abscisse du centre de la base
y = ordonnée du centre de la base
l = longueur du rectangle
h = hauteur du rectangle
c_1 = couleur de remplissage (jaune)
c_2 = couleur de la bordure (noir)

Les rectangles sont dessinés avec la fonction *xrect*.

- Les piquets et les disques



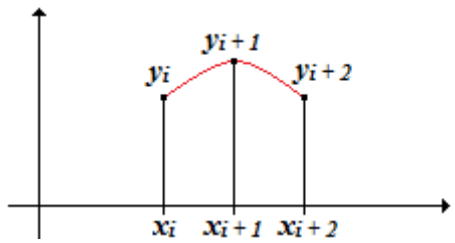
La tour de n disques a pour demi-base $L = (n_{max} + 1)dl$ et pour hauteur $H = n\,dh$. Le disque n° i $(1 \le i \le n)$ a pour longueur $l_i = 2(n_{max} - i + 2)dl$, sa hauteur est $h_i = dh$. Le piquet n° p $(1 \le p \le 3)$ est positionné en $x_p = L + (p - 1)E$, où E est l'écartement entre les piquets, sa hauteur est $y_p = H + dh$, sa largeur vaut $2dl$.

Notons D le handle associé à une entité graphique représentant un disque.
Les n disques sont organisés en listes de handles :

$$disque = liste(disque_1, disque_2, disque_3)$$
$$disque_p = liste(D_1, D_2, \ldots, D_m)$$

Ainsi, $disque_p = disque(p)$ est la liste ordonnée des handles des disques enfilés sur le piquet p à l'instant t.
Pour déplacer un disque, on le saisit symboliquement par sa "poignée", c'est-à-dire par son handle D, et on fait varier sa position dans le plan d'une quantité $[dx,dy]$ en utilisant l'instruction Scilab *move* : $move(D,[dx,dy])$.

- La trajectoire parabolique du déplacement d'un disque entre 2 piquets



Par 3 points distincts (x_i,y_i), (x_{i+1},y_{i+1}), (x_{i+2},y_{i+2}) il passe une unique parabole.
La formule d'interpolation de **Lagrange** donne l'équation $y = f(x)$ de la courbe :

$$y = y_i \frac{(x - x_{i+1})(x - x_{i+2})}{(x_i - x_{i+1})(x_i - x_{i+2})} + y_{i+1} \frac{(x - x_i)(x - x_{i+2})}{(x_{i+1} - x_i)(x_{i+1} - x_{i+2})} + y_{i+2} \frac{(x - x_i)(x - x_{i+1})}{(x_{i+2} - x_i)(x_{i+2} - x_{i+1})}$$

L'expression se simplifie si $y_{i+2} = y_i$ et $x_{i+1} = (x_i + x_{i+2})/2$: $\quad y = \dfrac{y_i (2x - x_i - x_{i+2})^2 - 4 y_{i+1} (x - x_i)(x - x_{i+2})}{(x_i - x_{i+2})^2}$

Le programme Scilab

```
// -----------------------------------------------------
//                    La Tour d'Hanoï
// -----------------------------------------------------
//
//          Loïc Vrillon - Décembre 2013
//

funcprot(0) ;

// Choix du nombre de disques

nmax = 10 ;
texte = 'Nombre de disques de la Tour d''Hanoï :' ;
n = x_choices(texte,list(list('',1,string([1:nmax])))) ;
if n == [] then, abort, end

// Paramètres

dl = 1 ; L = (nmax + 1)*dl ;
dh = 2 ; H = n*dh ; E = 2*L ;
xmin = - 2*dl ; xmax = 2*(L + E) + 2*dl ;
ymin = - dh ; ymax = H + 5*dh ;
dx = dl/4 ; dy = dh/8 ;
xtempo = 1e3 ; ytempo = 1e3 ;

// Fenêtre graphique et axes de coordonnées

f = scf() ;
f.figure_name = 'La Tour d''Hanoï' ;
X = getsystemmetrics('SM_CXSCREEN') ;
Y = getsystemmetrics('SM_CYSCREEN') ;
f.figure_size = [X/2,Y/2] ;
f.figure_position = [X/4,Y/4] ;
Noir = color(0,0,0) ; Blanc = color(255,255,255) ;
Bronze = color(97,78,26) ; Or = color(255,215,0) ;
toolbar(0, 'off') ;
delmenu(0, 'Fichier') ; delmenu(0, 'Édition') ;
delmenu(0, 'Outils') ; delmenu(0, '?') ;
a = gca() ;
a.data_bounds = [xmin,ymin;xmax,ymax] ;
a.isoview = 'on' ;
a.axes_visible = 'off' ;
a.x_label.visible = 'off' ;
a.y_label.visible = 'off' ;


// Echange de 2 valeurs

function [x,y] = echanger(x,y)
    z = x
    x = y
    y = z
endfunction
```

```
// Dessin de la dalle

function dessin_dalle()
    xrect(xmin,0,xmax - xmin,- ymin)
    e = gce()
    e.fill_mode = 'on'
    e.background = Bronze
    e.foreground = Noir
endfunction

// Dessin d'un piquet

function dessin_piquet(p)
    xrect(L + (p - 1)*E - dl,H + dh,2*dl,H + dh)
    e = gce()
    e.fill_mode = 'on'
    e.background = Blanc
    e.foreground = Noir
endfunction

// Dessin d'un disque

function D = dessin_disque(x,y,l,h)
    xrect(x - l/2,y + h,l,h)
    e = gce()
    e.fill_mode = 'on'
    e.background = Or
    e.foreground = Noir
    D = e
endfunction
```

```
// Déplacement d'un disque

function deplacer(p1,p2)
    function y = f(x)
        y = ((H + dh)*(2*x - x1 - x2)^2 - 4*(ymax - dh)*(x - x1)*(x - x2))/(x1 - x2)^2
    endfunction
    function dplx(x1,y1,x2,y2)
        x = x1
        y = y1
        ndx = abs(x2 - x1)/dx
        if x2 < x1 then
            dx = - dx
        end
        for i = 1:ndx
            x = x + dx
            dy = f(x) - y
            y = y + dy
            move(D,[dx,dy])
            xpause(xtempo)
        end
    endfunction
    function dply(x1,y1,x2,y2)
        ndy = abs(y2 - y1)/dy
        if y2 < y1 then
            dy = - dy
        end
        for i = 1:ndy
            move(D,[0,dy])
            xpause(ytempo)
        end
    endfunction
    global disque
    d1 = disque(p1)
    d2 = disque(p2)
    D = d1($)
    d1($) = null()
    disque(p1) = d1
    x1 = L + (p1 - 1)*E
    y1 = length(d1)*dh
    x2 = L + (p2 - 1)*E
    y2 = length(d2)*dh
    dply(x1,y1,x1,H + dh)
    dplx(x1,H + dh,x2,H + dh)
    dply(x2,H + dh,x2,y2)
    d2($+1) = D
    disque(p2) = d2
endfunction
```

```
// Déplacement de l'ensemble des disques

function hanoi(a,b,c,n)                 // version récursive
    if n > 0 then
        hanoi(a,c,b,n - 1)
        deplacer(a,c)
        hanoi(b,a,c,n - 1)
    end
endfunction

function hanoi(a,b,c,n)                 // version itérative
    N = 2^n
    k = 2
    while k > 1
        k = k - 1
        while k < N
            [b,c] = echanger(b,c)
            k = k*2
        end
        while k*modulo(k,2) > 1
            [a,b] = echanger(a,b)
            k = floor(k/2)
        end
        if k > 1 then
            [b,c] = echanger(b,c)
            deplacer(a,c)
            [a,b] = echanger(a,b)
            k = k + 2
        end
    end
endfunction

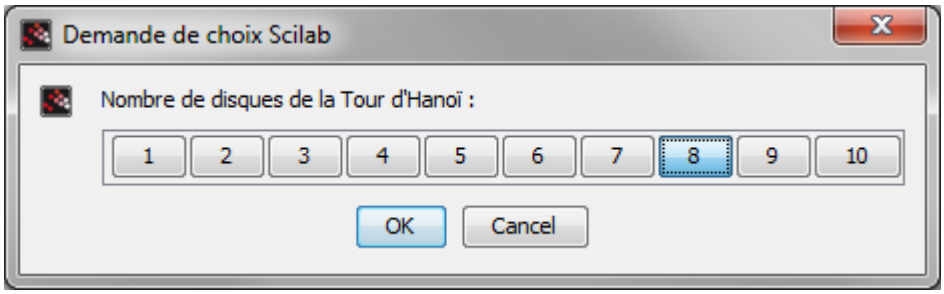
// Initialisation

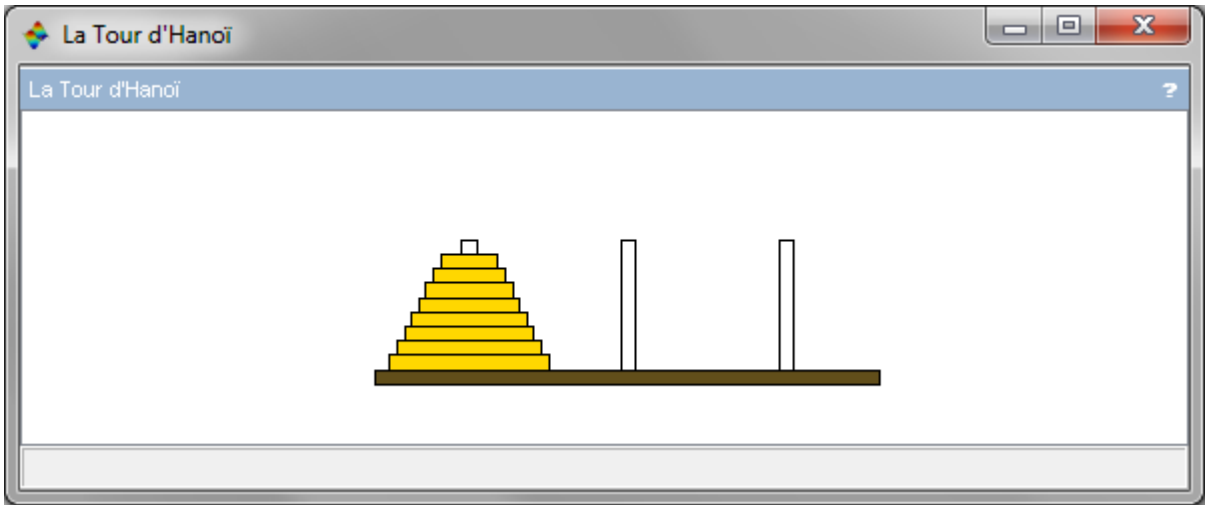
global disque
drawlater()
dessin_dalle() ;
for p = 1:3
    dessin_piquet(p) ;
end
disque1 = list() ; disque2 = list() ; disque3 = list() ;
x = L ; y = 0 ; l = 2*L ; h = dh ;
for i = 1:n
    D = dessin_disque(x,y,l,h) ;
    disque1($+1) = D ;
    y = y + dh ;
    l = l - 2*dl ;
end
disque = list(disque1,disque2,disque3) ;
drawnow()
xpause(2e6)

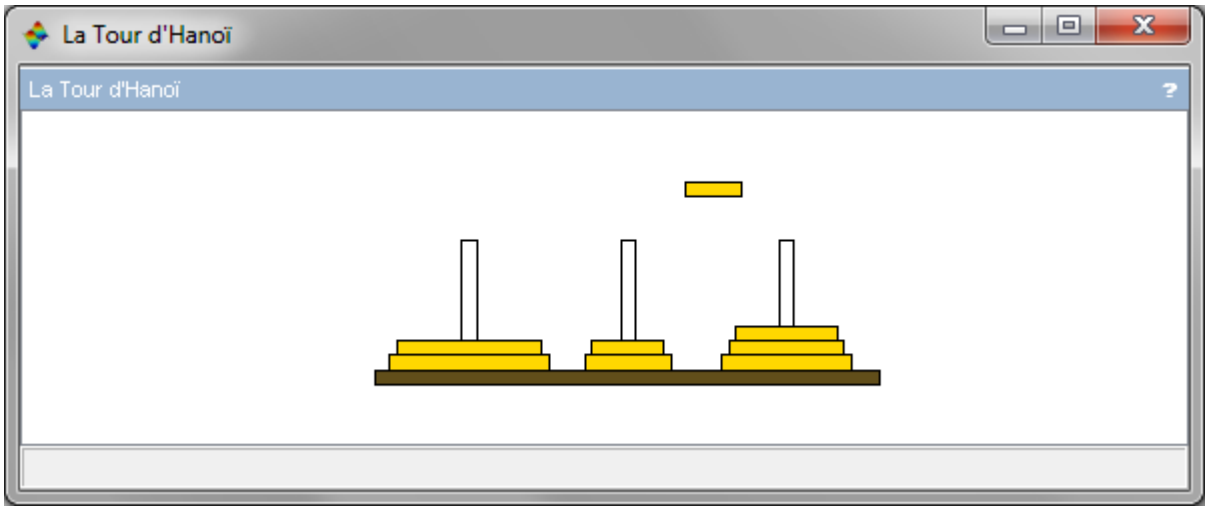
// Programme principal

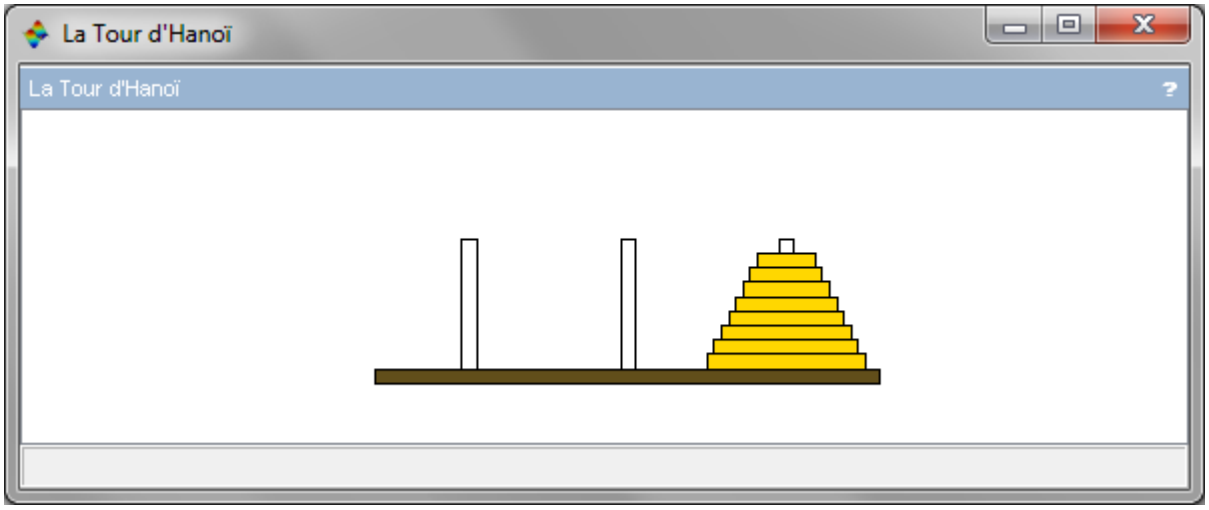
hanoi(1,2,3,n)
```

Mathématique & Informatique









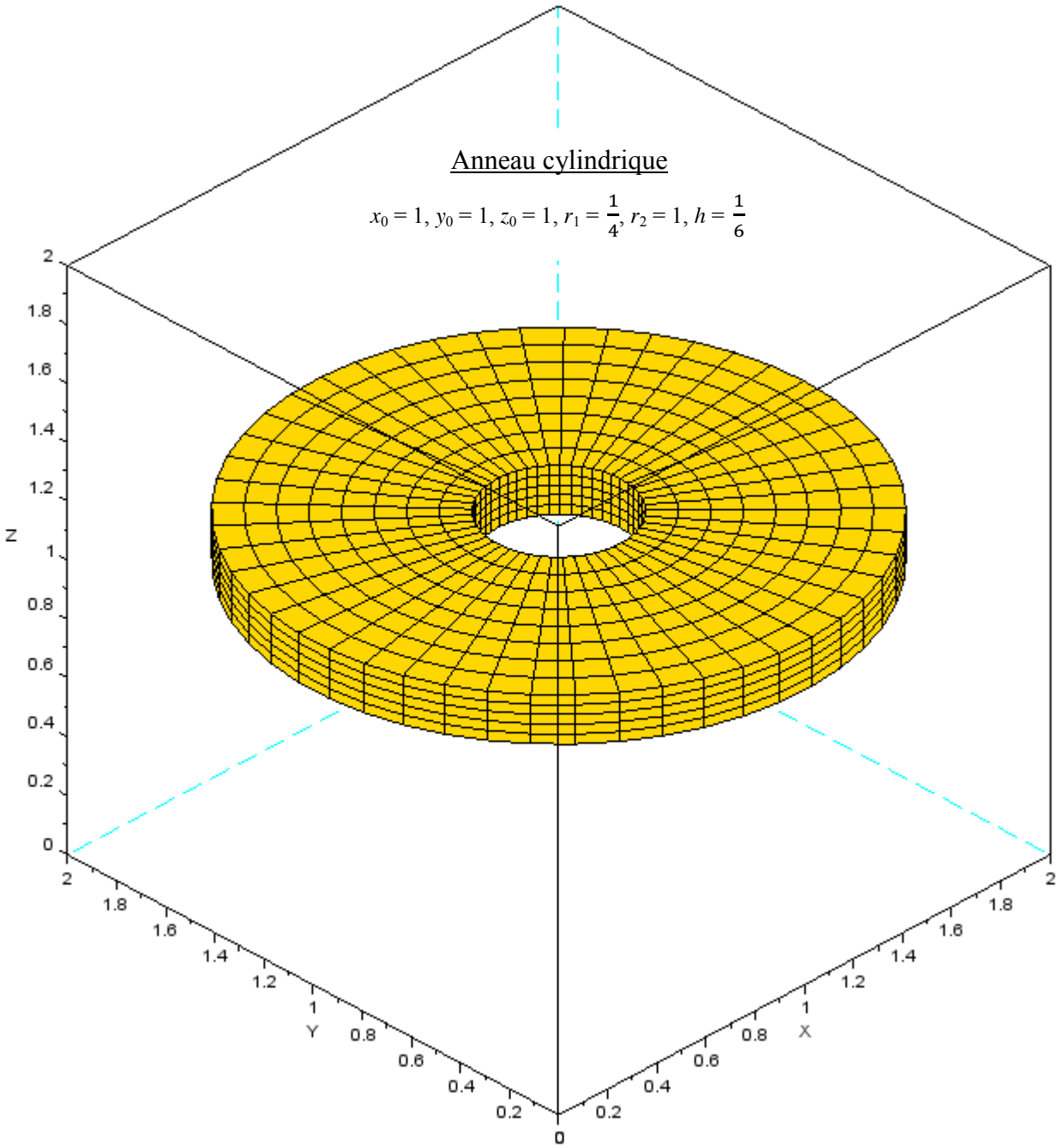
Mathématique & Informatique

La Tour d'Hanoï graphisme *3d*

La structure générale du programme est identique à celle du graphisme *2d* précédent.
Au lieu d'utiliser des objets graphiques définis en 2 dimensions (x,y), on utilise des objets graphiques définis en 3 dimensions (x,y,z) et dessinés à l'aide de la fonction *plot3d*. Pour un disque dont le handle est D, un déplacement dans l'espace de la quantité $[dx,dy,dz]$ s'obtient avec l'instruction *move*(D,$[dx,dy,dz]$).

Géométrie des objets :

- La dalle est un parallélépipède dont la face supérieure est située à la cote $z = 0$.
  L'objet est la réunion de 6 surfaces rectangulaires.
- Un piquet est un cylindre dont le centre de la base est situé en x_p sur l'axe des abscisses.
  L'objet est la réunion de 3 surfaces : une surface latérale cylindrique de hauteur h_0 et de rayon r_0, plus deux surfaces circulaires (une inférieure et une supérieure) de rayon r_0.
- Un disque est un anneau cylindrique dont le centre de la base est situé au point (x_0, y_0, z_0).
  L'objet est la réunion de 4 surfaces : deux surfaces latérales cylindriques, une interne et une externe, de hauteur h et de rayons respectifs r_1 et r_2, plus deux surfaces en forme de couronne circulaire (une inférieure et une supérieure) de rayon interne r_1 et de rayon externe r_2. On prend $r_1 = r_0$.



```
// -----------------------------------------------------
//                   La Tour d'Hanoï
// -----------------------------------------------------
//
//          Loïc Vrillon - Décembre 2013
//

// Choix du nombre de disques

nmax = 10 ;
texte = 'Nombre de disques de la Tour d''Hanoï :' ;
n = x_choices(texte,list(list('',1,string([1:nmax])))) ;
if n == [] then, abort, end

// Paramètres

dr = 1 ; R = (nmax + 1)*dr ;
dh = 2 ; H = n*dh ; E = 2*R ;
xmin = - 2*dr ; xmax = 2*(R + E) + 2*dr ;
ymin = - R - 2*dr ; ymax = R + 2*dr ;
zmin = - dh ; zmax = H + 5*dh ;
dx = dr/4 ; dz = dh/8 ;
xtempo = 1e3 ; ztempo = 1e3 ;
angle1 = 35 ; angle2 = - 105 ;

// Fenêtre graphique et axes de coordonnées

f = scf() ;
f.figure_name = 'La Tour d''Hanoï' ;
X = getsystemmetrics('SM_CXSCREEN') ;
Y = getsystemmetrics('SM_CYSCREEN') ;
f.figure_size = [X,Y] ;
f.figure_position = [0,0] ;
Noir = color(0,0,0) ; Blanc = color(255,255,255) ;
Bronze = color(97,78,26) ; Or = color(255,215,0) ;
a = gca() ;
a.data_bounds = [xmin,ymin,zmin;xmax,ymax,zmax] ;
a.isoview = 'on' ;
a.box = 'off' ;
a.axes_visible = 'off' ;
a.x_label.visible = 'off' ;
a.y_label.visible = 'off' ;
a.z_label.visible = 'off' ;

// Echange de 2 valeurs

function [x,y] = echanger(x,y)
    z = x
    x = y
    y = z
endfunction
```

```
// Dessin de la dalle

function dessin_dalle()
    function [X,Y,Z] = surface1(v,w)
        X = x*ones(1,j*k)
        Y = v
        Z = w
    endfunction
    function [X,Y,Z] = surface2(u,w)
        X = u
        Y = y*ones(1,i*k)
        Z = w
    endfunction
    function [X,Y,Z] = surface3(u,v)
        X = u
        Y = v
        Z = z*ones(1,i*j)
    endfunction
    x0 = xmin ; y0 = ymin ; z0 = zmin
    a = xmax - xmin ; b = ymax - ymin ; c = - zmin
    i = 2 ; j = 2 ; k = 5
    x = x0
    v = linspace(y0,y0 + b,j) ; w = linspace(z0,z0 + c,k)
    [xf,yf,zf] = eval3dp(surface1,v,w)
    plot3d(xf,yf,zf,flag = [Bronze,6,0],alpha = angle1,theta = angle2)
    e1 = gce() ; e1.hiddencolor = Bronze
    x = x0 + a
    v = linspace(y0,y0 + b,j) ; w = linspace(z0 + c,z0,k)
    [xf,yf,zf] = eval3dp(surface1,v,w)
    plot3d(xf,yf,zf,flag = [Bronze,6,0],alpha = angle1,theta = angle2)
    e2 = gce() ; e2.hiddencolor = Bronze
    y = y0
    u = linspace(x0,x0 + a,i) ; w = linspace(z0 + c,z0,k)
    [xf,yf,zf] = eval3dp(surface2,u,w)
    plot3d(xf,yf,zf,flag = [Bronze,6,0],alpha = angle1,theta = angle2)
    e3 = gce() ; e3.hiddencolor = Bronze
    y = y0 + b
    u = linspace(x0,x0 + a,i) ; w = linspace(z0,z0 + c,k)
    [xf,yf,zf] = eval3dp(surface2,u,w)
    plot3d(xf,yf,zf,flag = [Bronze,6,0],alpha = angle1,theta = angle2)
    e4 = gce() ; e4.hiddencolor = Bronze
    z = z0
    u = linspace(x0,x0 + a,i) ; v = linspace(y0,y0 + b,j)
    [xf,yf,zf] = eval3dp(surface3,u,v)
    plot3d(xf,yf,zf,flag = [Bronze,6,0],alpha = angle1,theta = angle2)
    e5 = gce() ; e5.hiddencolor = Bronze
    z = z0 + c
    u = linspace(x0,x0 + a,i) ; v = linspace(y0 + b,y0,j)
    [xf,yf,zf] = eval3dp(surface3,u,v)
    plot3d(xf,yf,zf,flag = [Bronze,6,0],alpha = angle1,theta = angle2)
    e6 = gce() ; e6.hiddencolor = Bronze
endfunction
```

```
// Dessin d'un piquet

function dessin_piquet(p)
    function [x,y,z] = surface1(u,v)
        x = x0 + r0*cos(u)
        y = y0 + r0*sin(u)
        z = z0 + v
    endfunction
    function [x,y,z] = surface2(u,w)
        x = x0 + w.*cos(u)
        y = y0 + w.*sin(u)
        z = z0*ones(1,length(u))
    endfunction
    x0 = R + (p - 1)*E ; y0 = 0 ; z0 = 0
    r0 = dr ; h0 = H + dh
    nu = 50 ; nv = 2 ; nw = 2
    u = linspace(0,2*%pi,nu) ; v = linspace(h0,0,nv)
    [xf,yf,zf] = eval3dp(surface1,u,v)
    plot3d(xf,yf,zf,flag = [-Noir,6,0],alpha = angle1,theta = angle2)
    e1 = gce() ; e1.hiddencolor = Noir
    u = linspace(0,2*%pi,nu) ; w = linspace(r0,0,nw)
    [xf,yf,zf] = eval3dp(surface2,u,w)
    plot3d(xf,yf,zf,flag = [-Blanc,6,0],alpha = angle1,theta = angle2)
    e2 = gce() ; e2.hiddencolor = Noir
    z0 = z0 + h0
    u = linspace(0,2*%pi,nu) ; w = linspace(0,r0,nw)
    [xf,yf,zf] = eval3dp(surface2,u,w)
    plot3d(xf,yf,zf,flag = [-Blanc,6,0],alpha = angle1,theta = angle2)
    e3 = gce() ; e3.hiddencolor = Noir
endfunction
```

```
// Dessin d'un disque

function D = dessin_disque(x0,y0,z0,r1,r2,h)
    function [x,y,z] = surface1(u,v)
        x = x0 + r*cos(u)
        y = y0 + r*sin(u)
        z = z0 + v
    endfunction
    function [x,y,z] = surface2(u,w)
        x = x0 + w.*cos(u)
        y = y0 + w.*sin(u)
        z = z0*ones(1,length(u))
    endfunction
    nu = 50 ; nv = 6 ; nw = r2/r1
    r = r1
    u = linspace(0,2*%pi,nu) ; v = linspace(0,h,nv)
    [xf,yf,zf] = eval3dp(surface1,u,v)
    plot3d(xf,yf,zf,flag = [-Or,6,0],alpha = angle1,theta = angle2)
    e1 = gce() ; e1.hiddencolor = Or
    r = r2
    u = linspace(0,2*%pi,nu) ; v = linspace(h,0,nv)
    [xf,yf,zf] = eval3dp(surface1,u,v)
    plot3d(xf,yf,zf,flag = [Or,6,0],alpha = angle1,theta = angle2)
    e2 = gce() ; e2.hiddencolor = Or
    u = linspace(0,2*%pi,nu) ; w = linspace(r2,r1,nw)
    [xf,yf,zf] = eval3dp(surface2,u,w)
    plot3d(xf,yf,zf,flag = [Or,6,0],alpha = angle1,theta = angle2)
    e3 = gce() ; e3.hiddencolor = Or
    z0 = z0 + h
    u = linspace(0,2*%pi,nu) ; w = linspace(r1,r2,nw)
    [xf,yf,zf] = eval3dp(surface2,u,w)
    plot3d(xf,yf,zf,flag = [Or,6,0],alpha = angle1,theta = angle2)
    e4 = gce() ; e4.hiddencolor = Or
    D = glue([e1,e2,e3,e4])
endfunction
```

```
// Déplacement d'un disque

function deplacer(p1,p2)
    function z = f(x)
        z = ((H + dh)*(2*x - x1 - x2)^2 - 4*(zmax - dh)*(x - x1)*(x - x2))/(x1 - x2)^2
    endfunction
    function dplx(x1,z1,x2,z2)
        x = x1
        z = z1
        ndx = abs(x2 - x1)/dx
        if x2 < x1 then
            dx = - dx
        end
        for i = 1:ndx
            x = x + dx
            dz = f(x) - z
            z = z + dz
            move(D,[dx,0,dz])
            xpause(xtempo)
        end
    endfunction
    function dplz(x1,z1,x2,z2)
        ndz = abs(z2 - z1)/dz
        if z2 < z1 then
            dz = - dz
        end
        for i = 1:ndz
            move(D,[0,0,dz])
            xpause(ztempo)
        end
    endfunction
    global disque
    d1 = disque(p1)
    d2 = disque(p2)
    D = d1($)
    d1($) = null()
    disque(p1) = d1
    x1 = R + (p1 - 1)*E
    z1 = length(d1)*dh
    x2 = R + (p2 - 1)*E
    z2 = length(d2)*dh
    dplz(x1,z1,x1,H + dh)
    dplx(x1,H + dh,x2,H + dh)
    dplz(x2,H + dh,x2,z2)
    d2($+1) = D
    disque(p2) = d2
endfunction
```

```
// Déplacement de l'ensemble des disques

function hanoi(a,b,c,n)
    N = 2^n
    k = 2
    while k > 1
        k = k - 1
        while k < N
            [b,c] = echanger(b,c)
            k = k*2
        end
        while k*modulo(k,2) > 1
            [a,b] = echanger(a,b)
            k = floor(k/2)
        end
        if k > 1 then
            [b,c] = echanger(b,c)
            deplacer(a,c)
            [a,b] = echanger(a,b)
            k = k + 2
        end
    end
endfunction

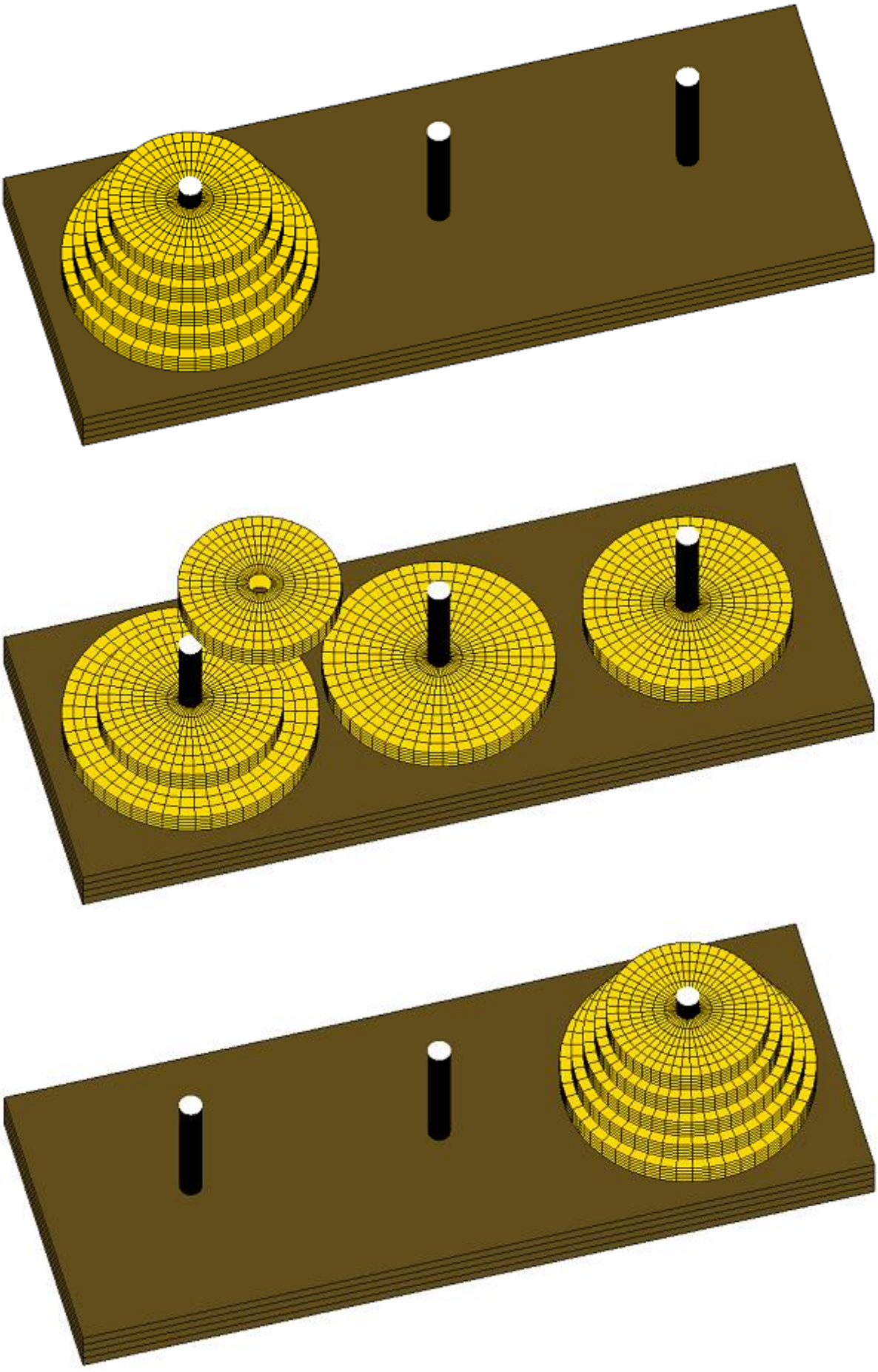
// Initialisation

global disque
drawlater()
dessin_dalle() ;
for p = 1:3
    dessin_piquet(p) ;
end
disque1 = list() ; disque2 = list() ; disque3 = list() ;
x0 = R ; y0 = 0 ; z0 = 0 ; r1 = dr ; r2 = R ; h = dh ;
for i = 1:n
    D = dessin_disque(x0,y0,z0,r1,r2,h) ;
    disque1($+1) = D ;
    z0 = z0 + dh ;
    r2 = r2 - dr ;
end
disque = list(disque1,disque2,disque3) ;
drawnow()
xpause(2e6)

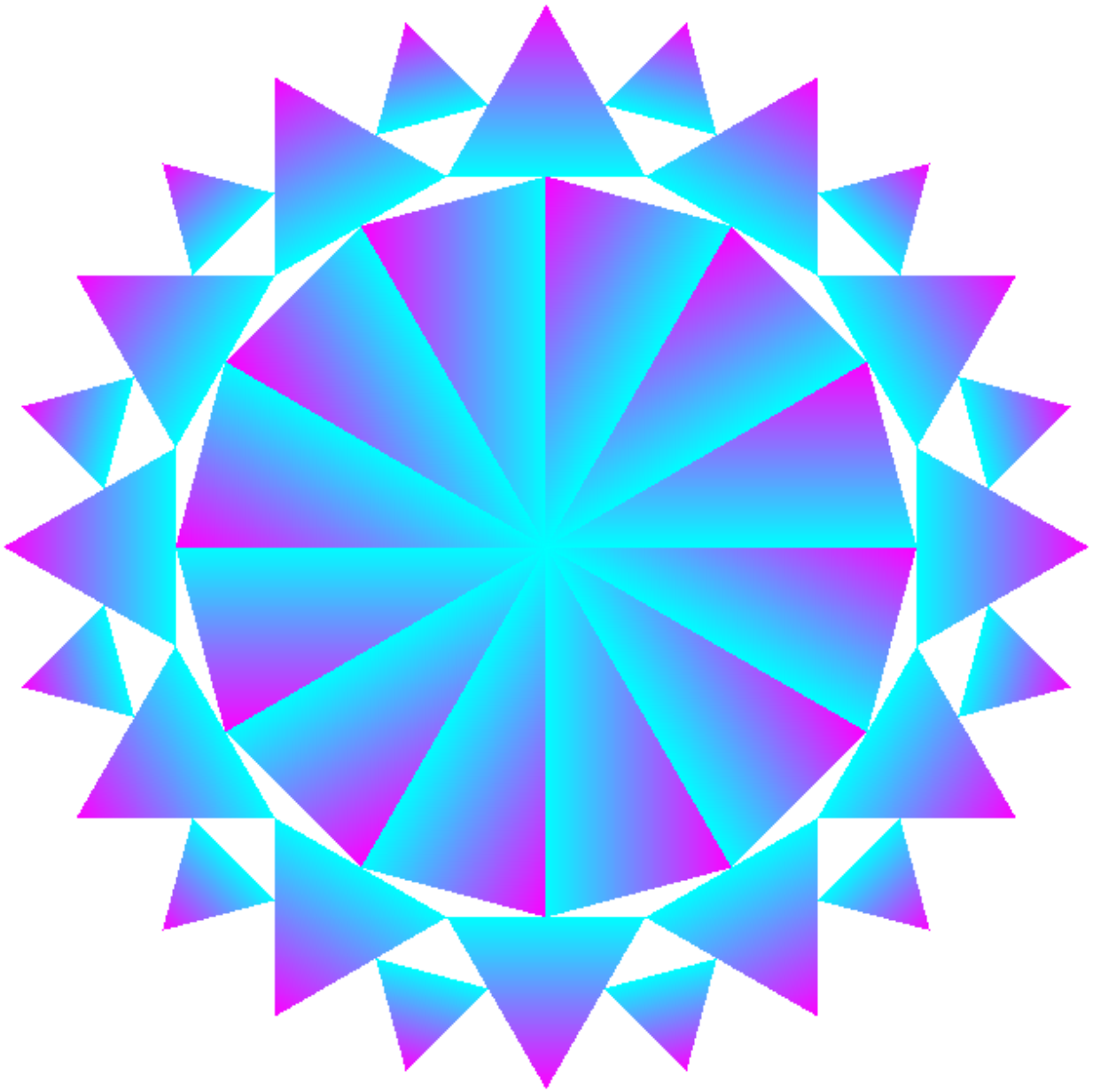
// Programme principal

hanoi(1,2,3,n)
```

Mathématique & Informatique



Mathématique & Informatique



Mathématique & Informatique

STATISTIQUES

Cliquer sur *? → Aide de Scilab → Statistiques* pour afficher la liste des fonctions statistiques fournies par Scilab.

Exemple 1 : Histogramme

Les fonctions *histplot* et *hist3d* de la Bibliothèque graphique sont utilisées pour représenter des histogrammes en 2D et en 3D respectivement (entrer "help histplot "ou "help hist3d" à la Console pour obtenir plus d'information sur l'utilisation de ces fonctions).

Application numérique

Les pesées (précision 10 g) de 161 nouveaux-nés ont conduit à des résultats s'échelonnant entre 2.250 kg et 4.530 kg.

Classe	Limites de la classe	Centre de la classe	Effectif de la classe
1	2.2 - 2.5	2.350	5
2	2.5 - 2.8	2.650	11
3	2.8 - 3.1	2.950	24
4	3.1 - 3.4	3.250	40
5	3.4 - 3.7	3.550	42
6	3.7 - 4.0	3.850	20
7	4.0 - 4.3	4.150	13
8	4.3 - 4.6	4.450	6

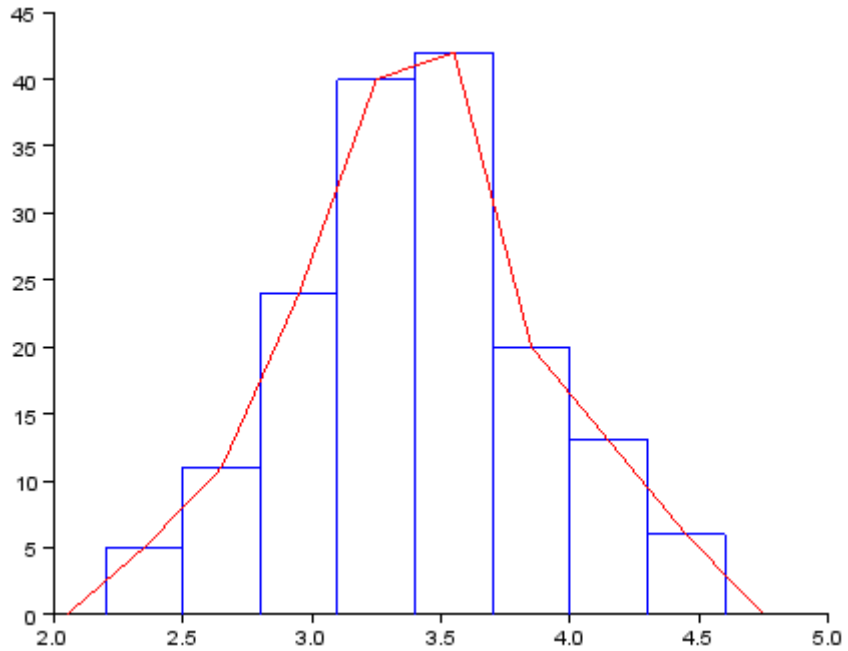
Le programme :

```
x = [2.2 2.5 2.8 3.1 3.4 3.7 4.0 4.3 4.6] ;
y = [2.050 2.350 2.650 2.950 3.250 3.550 3.850 4.150 4.450 4.750] ;
z = [0 5 11 24 40 42 20 13 6 0] ;

data = [...
2.33 2.38 2.29 2.28 2.25 2.52 2.79 2.56 2.56 2.75   ...
2.72 2.71 2.53 2.79 2.68 2.52 2.90 2.83 3.02 3.02   ...
2.82 3.01 2.98 2.89 2.94 2.81 3.05 2.94 3.02 2.99   ...
3.08 3.00 2.92 3.01 3.09 2.87 3.04 2.97 2.98 2.87   ...
3.11 3.38 3.12 3.17 3.11 3.28 3.35 3.15 3.18 3.32   ...
3.15 3.23 3.26 3.26 3.27 3.20 3.28 3.37 3.38 3.11   ...
3.25 3.18 3.38 3.34 3.30 3.30 3.24 3.39 3.39 3.15   ...
3.23 3.23 3.37 3.14 3.33 3.39 3.19 3.14 3.31 3.22   ...
3.59 3.46 3.64 3.67 3.61 3.65 3.43 3.53 3.59 3.52   ...
3.52 3.44 3.45 3.41 3.57 3.63 3.57 3.53 3.62 3.56   ...
3.62 3.57 3.69 3.46 3.44 3.42 3.53 3.48 3.55 3.41   ...
3.45 3.68 3.69 3.41 3.62 3.48 3.66 3.68 3.45 3.52   ...
3.41 3.56 3.71 3.92 3.79 3.76 3.90 3.84 3.87 3.88   ...
3.96 3.94 3.76 3.86 3.74 3.81 3.87 3.74 3.82 3.71   ...
3.96 3.80 4.11 4.11 4.01 4.06 4.14 4.07 4.29 4.28   ...
4.18 4.02 4.02 4.08 4.08 4.46 4.34 4.53 4.33 4.41   ...
4.53] ;

histplot(x, data, style=2, normalization=%f)
plot2d(y, z, style=5)
```

donne l'histogramme suivant :



Le polygone des effectifs est la ligne brisée qui joint les milieux des côtés supérieurs des rectangles (ligne en rouge).
La moyenne de l'échantillon se calcule avec la fonction *mean* : m = mean(data) = 3.400 kg
L'écart-type de l'échantillon se calcule avec la fonction *msd* : e = msd(data) = 0.480 kg

A noter que pour un échantillon, la moyenne d'un caractère X manque de signification si elle est exprimée seule, c'est pourquoi on lui adjoint en complément la valeur de l'écart-type.

Remarque importante

La forme de l'histogramme évoque la loi de distribution pour la population dont on extrait l'échantillon.
Le plus souvent, on reconnaît une courbe en cloche, caractéristique d'une loi "normale" ou **loi de Laplace - Gauss** de densité de probabilité :

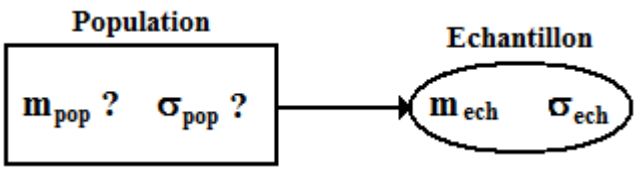
$$p(x) = \frac{1}{\sqrt{2\pi}}\, e^{-\frac{x^2}{2}}$$

Echantillonnage

Les paramètres (moyenne, écart-type) d'un caractère X dans une population sont inconnus : m_{pop} = ? et σ_{pop} = ?
On tire un échantillon x = [x_1 , x_2 , ... , x_n] d'effectif n et on calcule :

1) sa moyenne : $m_{ech} = m = \bar{x} = \Sigma x_i/n$ = mean(x)
2) son écart-type : $\sigma_{ech} = e = [\Sigma(x_i - \bar{x})^2/n]^{1/2}$ = msd(x)

Attention à ne pas confondre l'écart-type σ_{ech} de l'échantillon avec l'estimation σ de l'écart-type σ_{pop} de la population qui se calcule avec la fonction *st_deviation* : $\sigma = s = [\Sigma(x_i - \bar{x})^2/(n - 1)]^{1/2}$ = st_deviation(x).



$$m_{pop} \approx m_{ech}$$

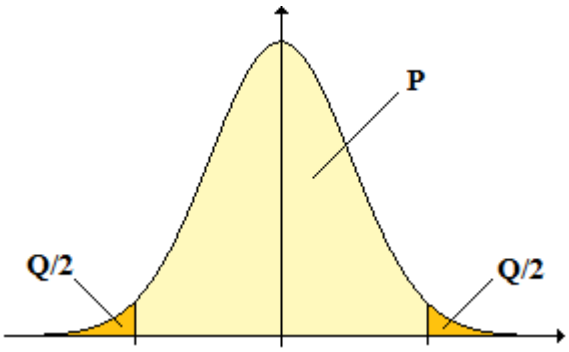
$$\sigma_{pop} \approx \sqrt{\frac{n}{n-1}}\ \sigma_{ech}$$

Exemple 2 : Intervalle de confiance

La moyenne μ d'un caractère X dans une population est estimée à partir de la moyenne m observée sur un échantillon x de n valeurs : $x = [x_1 , x_2 , ... , x_n]$.

On établit que : $$m - k\frac{\sigma}{\sqrt{n}} \leq \mu \leq m + k\frac{\sigma}{\sqrt{n}}$$ avec une probabilité P

Précisons la signification de m, σ, P, k.



La moyenne m

$$m = \frac{\sum_{i=1}^{n} x_i}{n}$$ est la moyenne de l'échantillon.

La fonction Scilab *mean* donne : m = ***mean***(x)

L'écart-type σ

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - m)^2}{n - 1}}$$ est l'estimation de l'écart-type (déviation standard) de la population.

La fonction Scilab *st_deviation* donne : sigma = ***st_deviation***(x)

La probabilité P

P est appelée coefficient (ou seuil) de sécurité et Q = 1 - P est appelée coefficient (ou seuil) de risque.
Conventionnellement, on prend P = 95 % ou 99 %, donc Q = 5 % ou 1 %.

Le paramètre k

k dépend à la fois de n et de P.

Si n ≥ 30 (grand échantillon) alors k ≡ u. La valeur de u est donnée par la fonction Scilab *cdfnor* pour une **loi de Gauss** (loi normale) centrée réduite et un coefficient de sécurité P : u = ***cdfnor***("X", 0, 1, (1+P)/2, (1-P)/2)

Si n < 30 (petit échantillon) alors k ≡ t. La valeur de t est donnée par la fonction Scilab *cdft* pour une **loi de Student** à (n - 1) degrés de liberté et un coefficient de sécurité P : t = ***cdft***("T", n-1, (1+P)/2, (1-P)/2)

Un des intérêts d'utiliser Scilab est de ne pas avoir à rechercher des valeurs dans des tables de lois de distribution.

Application numérique

On mesure au 1/10e de mm le diamètre de 20 coquillages d'une même espèce. A partir de cet échantillon x, quel est l'intervalle de confiance du diamètre moyen de ce type de coquillage, avec un coefficient de sécurité de 95 % ?

```
x = [141 131 131 128 137 133 133 113 110 111 112 121 144 110 117 116 107 110 130 115] ;
n = length(x) ; m = mean(x) ; sigma = st_deviation(x) ; P = 95/100 ;
if n >= 30 then, k = cdfnor("X", 0, 1, (1+P)/2, (1-P)/2) ; else, k = cdft("T", n-1, (1+P)/2, (1-P)/2) ; end
a = m - k*sigma/sqrt(n) , b = m + k*sigma/sqrt(n)
```

Résultat : *a* = 116.98428 ; *b* = 128.01572. Par conséquent, le diamètre moyen se situe dans l'intervalle [117 , 128] avec une probabilité de 95 % ou, ce qui est équivalent, avec un risque d'erreur de 5 %.

Exemple 3 : Test du χ^2

On classe des individus d'une population selon 2 critères.
Le test du χ^2 étudie l'existence d'une liaison entre ces 2 critères de classement.
Il se présente sous la forme d'un tableau d'effectifs et permet de conclure :

- soit à une homogénéité de la répartition des valeurs observées : les 2 critères sont indépendants et les différences sont dues au hasard.
- soit au contraire à une non homogénéité : les 2 critères sont dépendants et les différences sont significatives.

Supposons que le premier critère soit divisé en **l** classes et que le second soit divisé en **c** classes.
On obtient un tableau à l lignes et c colonnes :

$$T = \begin{pmatrix} n_{11} \cdots n_{1j} \cdots n_{1c} \\ \vdots \quad \vdots \quad \vdots \\ n_{i1} \cdots n_{ij} \cdots n_{ic} \\ \vdots \quad \vdots \quad \vdots \\ n_{l1} \cdots n_{lj} \cdots n_{lc} \end{pmatrix}$$

A l'intersection de la ligne i et de la colonne j se trouve l'effectif observé n_{ij} qui correspond au nombre d'individus appartenant à la classe i du 1er critère et à la classe j du 2e critère.
Pour chaque case (i,j) du tableau T, on estime un effectif calculé (théorique) : $x_{ij} = L_i * C_j / N$ avec :

$$L_i = \sum_j n_{ij} \; ; \; C_j = \sum_i n_{ij} \; ; \; N = \sum_{i,j} n_{ij}$$

On pose :

$$\chi^2 = \sum_{i,j} \frac{(n_{ij} - x_{ij})^2}{x_{ij}}$$

Le nombre de degrés de liberté du test est par définition :

$$d = (l-1)(c-1)$$

Si l'hypothèse d'homogénéité est vraie, alors il y a une probabilité Q que la valeur du χ^2 atteigne ou dépasse une valeur seuil χ_0^2. La fonction Scilab *cdfchi* permet :
- de calculer χ_0^2 à partir de Q. On écrit : KHI2_0 = **cdfchi**("X", d, 1-Q, Q)
- de calculer Q à partir de χ_0^2. On écrit : [P,Q] = **cdfchi**("PQ", KHI2_0, d)

Conclusion du test :

- $\chi^2 < \chi_0^2$: on retient l'hypothèse d'homogénéité, au seuil de signification Q
- $\chi^2 \geq \chi_0^2$: on rejette l'hypothèse d'homogénéité, au seuil de signification Q

Remarques
Le χ^2 nécessite des $n_{ij} \geq 5$ (sinon effectuer un regroupement).
La probabilité Q correspond au risque de rejeter l'hypothèse d'homogénéité alors qu'en réalité elle est vraie.
En général, on choisit Q = 5 % ou Q = 1 % pour retenir l'hypothèse de non homogénéité d'un tableau, c'est-à-dire pour considérer que les 2 critères sont dépendants lorsque la valeur du χ^2 est significativement trop grande.
Le χ^2 est un test non paramétrique : il ne fait aucune hypothèse sur les lois de distribution des variables (lois qui sont généralement inconnues). D'où son intérêt.

Application numérique

A l'aide d'un échantillon pris au hasard, on recense le nombre d'hommes mariés et célibataires dans 8 grandes villes :

	A	B	C	D	E	F	G	H
Mariés	133	164	155	106	153	123	146	150
Célibataires	46	57	40	37	55	43	39	51

Ces données indiquent elles une différence à se marier dans ces villes ?

Programmons un test du χ^2 :

```
T = [133,164,155,106,153,123,146,150 ; ...
     46,57,40,37,55,43,39,51] ;
l = size(T,1) ; c = size(T,2) ; N = sum(T) ;
KHI2 = 0 ;
for i = 1:l
  for j = 1:c
    n = T(i,j) ;
    x = sum(T(i,:))*sum(T(:,j))/N ;
    KHI2 = KHI2 + (n-x)^2/x ;
  end
end
KHI2_0 = KHI2 , d = (l-1)*(c-1) ;
[P,Q] = cdfchi("PQ", KHI2_0, d)
```

On obtient :

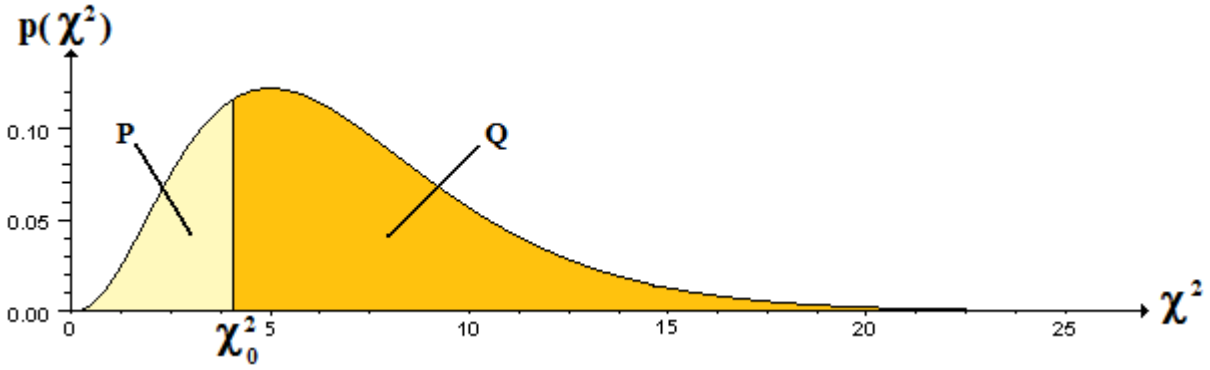
KHI2_0 = 4.0023428
Q = 0.7795075

Conclusion :

Pour rejeter l'hypothèse d'homogénéité on doit prendre un seuil $\chi_0^2 = 4.0023428$. Le risque de se tromper est alors de $Q \approx 78$ %. Donc, au vu de l'échantillon fourni, on retient l'hypothèse d'homogénéité (d'indépendance) : il n'y a pas de lien entre le fait d'être marié - ou célibataire - et le fait d'habiter dans l'une des villes A à H.

Loi du χ^2 de Pearson

La loi du χ^2 à d degrés de liberté est la somme des carrés de d lois normales centrées réduites N(0,1) : $\chi^2 = \Sigma N^2$
Densité de probabilité : $p(x) = e^{-x/2} x^{d/2 - 1} / 2^{d/2} \Gamma(d/2)$. Espérance mathématique : $E(\chi^2) = d$. Variance : $V(\chi^2) = 2d$.
Représentation de la courbe du χ^2 dans l'exemple étudié ci-dessus :



Exemple 4 : Régression & Corrélation

Un couple de variables aléatoires (X,Y) prend n valeurs $(x_i , y_i)_{1 \le i \le n}$ que l'on représente par n points dans le plan.
La fonction Scilab *regress* calcule les coefficients a et b de la droite y = ax + b qui passe "au mieux" par l'ensemble des points (x_i , y_i) au sens des moindres carrés, c'est-à-dire qui minimise la quantité $\Sigma(y_i - ax_i - b)^2$.
De la même manière que l'on a effectué une régression de Y par rapport à X pour obtenir la droite y = ax + b, on effectue une régression de X par rapport à Y pour obtenir la droite x = a'y + b'. Le coefficient de corrélation r mesure l'angle de ces 2 droites. Par définition : $r^2 = aa'$ (produit des pentes des 2 droites). La fonction Scilab *correl* calcule r :

$$r = \frac{Cov(X,Y)}{\sqrt{Var(X)\,Var(Y)}} = \frac{\sum (X-\overline{X})(Y-\overline{Y})}{\sqrt{\sum (X-\overline{X})^2 \sum (Y-\overline{Y})^2}}$$

La valeur de $r \in [-1,+1]$ renseigne sur le lien existant entre les 2 variables X et Y :

- $r < 0 \quad \Rightarrow$ X et Y varient en sens inverse
- $r > 0 \quad \Rightarrow$ X et Y varient dans le même sens
- $r \approx 0 \quad \Rightarrow$ X et Y sont indépendantes (le nuage de points est élargi)
- $r \approx \pm 1 \quad \Rightarrow$ X et Y sont fortement liées (le nuage de points est aplati)

Attention : si 2 variables sont fortement corrélées cela ne signifie pas qu'il y a une relation de cause à effet entre elles, comme on pourrait le croire. Tout au plus, on peut affirmer qu'elles varient linéairement ensemble, mais on ignore les causes de ces variations.

Intervalle de confiance du coefficient de corrélation
Dans une population, le coefficient de corrélation entre 2 caractères X et Y est ρ (inconnu). Un échantillon pris au hasard, d'effectif n, fournit un coefficient de corrélation r que l'on peut calculer. Soit R la variable aléatoire des coefficients de corrélation sur l'ensemble des échantillons. La transformation de **Fisher** consiste à définir une variable aléatoire Z = Arg th(R) qui suit approximativement une loi de **Gauss** de moyenne Arg th(ρ) et de variance 1/(n - 3).
On en déduit un intervalle de confiance [Inf , Sup] qui contient ρ avec la probabilité P.
Posons u = cdfnor("X", 0, 1, (1+P)/2, (1-P)/2), on trouve :
Inf = (r - tanh(u/sqrt(n-3)))/(1 - r*tanh(u/sqrt(n-3))) , Sup = (r + tanh(u/sqrt(n-3)))/(1 + r*tanh(u/sqrt(n-3)))

Application numérique

Le relevé du poids (variable X) et de la taille (variable Y) de 100 jeunes personnes a conduit au tableau suivant :

		Y en cm			
		150 - 155	155 - 160	160 - 165	165 - 170
X en kg	40 - 45	**20**	**2**	**0**	**0**
	45 - 50	**9**	**18**	**5**	**1**
	50 - 55	**1**	**4**	**12**	**7**
	55 - 60	**0**	**1**	**6**	**14**

Les effectifs sont regroupés en 16 classes. Par exemple, il y a 20 personnes dont le poids se situe entre 40 et 45 kilogrammes avec une taille comprise entre 150 et 155 centimètres.
Ecrivons un programme Scilab qui représente le nuage de points (X,Y), trace les droites de régression de Y en X (couleur rouge) et de X en Y (couleur bleue), puis calcule le coefficient de corrélation entre X et Y.

Mathématique & Informatique

// Données

```
X = [...
42.0 40.0 42.6 45.0 40.4 44.9 44.1 44.4 43.6 40.1 ...
41.6 40.8 40.0 43.0 43.8 42.7 43.9 41.9 41.6 43.4 ...
41.7 44.9 49.4 49.0 49.7 47.7 48.7 45.4 46.1 47.2 ...
46.3 46.0 48.0 46.9 47.7 49.6 45.4 49.4 48.8 47.9 ...
49.5 45.7 49.6 47.1 49.8 47.7 49.6 48.5 45.8 45.3 ...
45.9 49.4 46.1 45.7 47.3 53.7 51.7 52.8 53.5 50.7 ...
54.0 51.2 52.7 50.9 50.3 50.1 51.3 50.3 50.2 52.4 ...
52.6 54.3 51.0 53.0 54.2 52.7 51.0 53.1 53.9 55.9 ...
57.9 57.4 60.0 55.1 58.7 57.4 58.3 58.4 57.1 57.0 ...
56.0 59.5 55.5 59.6 59.8 58.7 59.3 56.2 55.1 59.5 ...
] ;
```

```
Y = [...
155 151 154 153 150 150 155 150 155 151 ...
154 154 155 150 154 150 150 155 154 150 ...
160 156 155 153 153 153 151 153 152 151 ...
155 156 156 157 159 157 160 156 159 156 ...
160 157 158 156 159 156 160 157 156 161 ...
161 164 161 164 169 153 158 157 159 159 ...
164 162 162 164 165 163 161 165 164 164 ...
164 164 170 170 167 170 168 166 170 158 ...
165 162 161 162 164 161 167 166 169 170 ...
169 168 168 170 168 167 167 166 167 166 ...
] ;
```

// Nuage de points

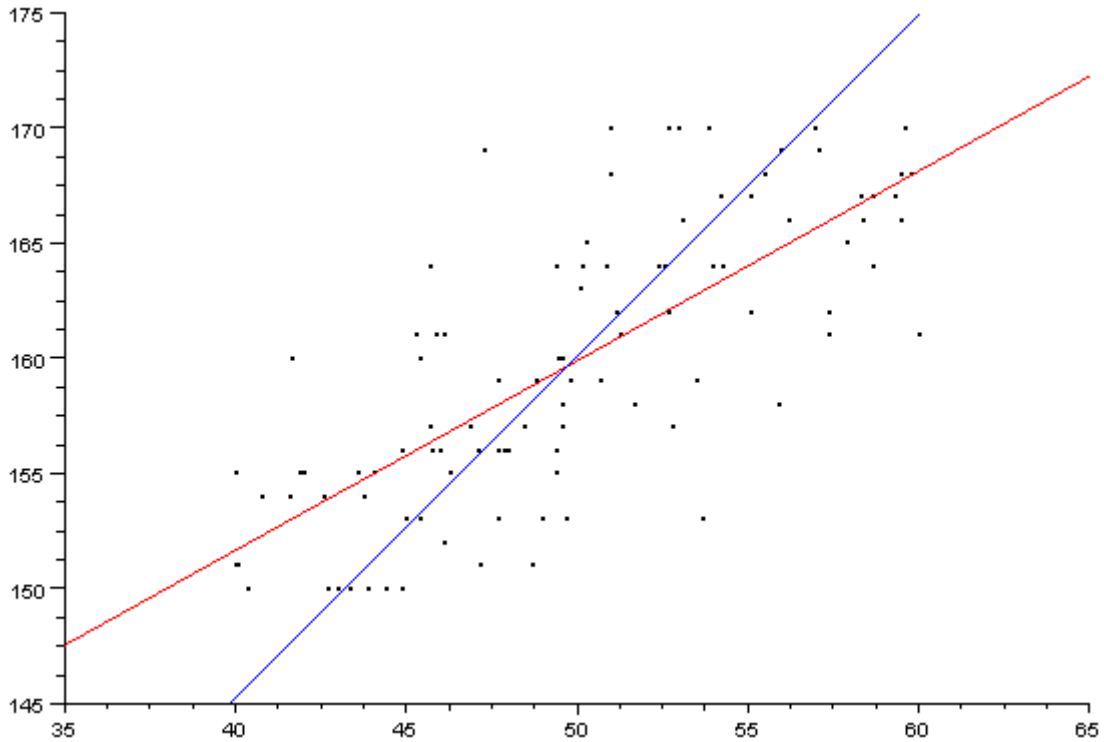
```
plot2d(X, Y, style=0, frameflag=1, rect=[35,145,65,175])
```

// Régressions

```
coeffs_1 = regress(X, Y) ;
x1 = linspace(35, 65, 300) ; y1 = coeffs_1(2)*x1 + coeffs_1(1) ; plot2d(x1, y1, style=5)
coeffs_2 = regress(Y, X) ;
y2 = linspace(145, 175, 300) ; x2 = coeffs_2(2)*y2 + coeffs_2(1) ; plot2d(x2, y2, style=2)
```

// Corrélation

```
fre = eye(length(X),length(Y)) ; r = correl(X, Y, fre)
```



Les 2 droites de régression se rencontrent au point moyen (m_X , m_Y) = (mean(X) , mean(Y)) = (49.73 , 159.67).
Le coefficient de corrélation est r = 0.7451979. Par conséquent, en première approximation, on admet que le poids et la taille sont deux caractères qui varient linéairement dans le même sens.

Note : r = sum((X - mean(X)).*(Y - mean(Y)))/sqrt(sum((X - mean(X))^2)*sum((Y - mean(Y))^2))

Exemple 5 : Analyse en Composantes Principales (ACP)

On considère p variables aléatoires X_i ($1 \le i \le p$), observées simultanément sur un échantillon de n individus.

	X_1	X_2	. . .	X_p
1^{er} individu	$x_1^{(1)}$	$x_1^{(2)}$	. . .	$x_1^{(p)}$
2^{e} individu	$x_2^{(1)}$	$x_2^{(2)}$	. . .	$x_2^{(p)}$
. . .	. . .	. . .		. . .
n^{e} individu	$x_n^{(1)}$	$x_n^{(2)}$	. . .	$x_n^{(p)}$

L'ensemble des données peut être représenté par un nuage de n points de $\mathbb{R}^p$.
On souhaite effectuer une projection dans le plan $\mathbb{R}^2$ en choisissant de nouveaux axes, appelés axes factoriels ou composantes principales, qui réalisent le meilleur étirement du nuage de points.
On montre que ce problème d'analyse de données, dit ACP, revient à diagonaliser la matrice C des corrélations :

$$C = [r(X_i, X_j)] = \left[\frac{Cov(X_i, X_j)}{\sigma(X_i)\ \sigma(X_j)} \right]$$

La matrice C, de dimension p×p, est symétrique puisque $r(X_i,X_j) = r(X_j,X_i)$, donc ses p valeurs propres sont réelles.
L'axe principal d'étirement du nuage est le vecteur propre associé à la plus grande valeur propre de C.
La représentation plane des observations fait généralement apparaître des groupes d'individus qui possèdent des similitudes et sont d'autant plus "proches" qu'ils sont corrélés. Il reste à faire une bonne interprétation de ces regroupements ...

Remarque :
Les calculs de l'ACP se font avec des variables centrées réduites : $Y_i = (X_i - m(X_i))/\sigma(X_i) \Rightarrow r(X_i,X_j) = r(Y_i,Y_j)$

Programmation Scilab :

Soit x la matrice des données :

$$x = \begin{pmatrix} x_1^{(1)} & \cdots & x_1^{(p)} \\ \vdots & \ddots & \vdots \\ x_n^{(1)} & \cdots & x_n^{(p)} \end{pmatrix}$$

Pour effectuer l'ACP on écrit simplement :

[lambda, facpr, comprinc] = pca(x)

et pour obtenir une représentation plane :

show_pca(lambda, facpr, N)

Mathématique & Informatique

CALCULS FINANCIERS

Limitons nous à quelques problèmes courants.
Dans les formules, on notera τ la valeur réelle d'un taux d'intérêt.
L'arrondi d'une somme x € portera sur la deuxième décimale (centimes d'euro) : $x \approx \text{round}(x*100)/100$.

Problème 1 : INTERETS COMPOSES

On place une somme A à un taux d'intérêt annuel τ.
En laissant les intérêts s'ajouter au capital, quel est le capital disponible C au bout de n années ?

Formule :

$$C = (1+\tau)^n A$$

Exemple : A = 10 000 € , τ = 3 % , n = 4 ans

```
A = 10000 ; taux = 3/100 ; n = 4 ;
C = round((1+taux)^n*A*100)/100
```

donne : C = 11 255.09 €

Problème 2 : EPARGNE

En fin d'année, on décide de souscrire à un placement au taux d'intérêt annuel fixe τ pendant n années. Le capital d'apport est A et chaque année on effectue m versements réguliers d'un montant *a*. Valeur de l'épargne E ?

Formules :

Appelons t_p le **taux proportionnel** à τ et t_e le **taux équivalent** à τ, pour la périodicité m.

$$t_p = \frac{\tau}{m} \quad et \quad t_e = (1+\tau)^{1/m} - 1$$

Exemple : A = 10 000 € , *a* = 300 € , τ = 4 % , m = 4 , n = 3 ans

```
A = 10000 ; a = 300 ; taux = 4/100 ; m = 4 ; n = 3 ;
E = A ; tp = taux/m ;
for i = 1:n
  I = 0 ;
  for j = 1:m
    E = E + a ; I = I + E*tp ;
  end
  E = E + I ;
end
E = round(E*100)/100
```

donne : E = 15 088.21 €

```
A = 10000 ; a = 300 ; taux = 4/100 ; m = 4 ; n = 3 ;
E = A ; te = (1+taux)^(1/m) -1 ;
for i = 1:n
  for j = 1:m
    E = E + a ; I = E*te ; E = E + I ;
  end
end
E = round(E*100)/100
```

donne : E = 15 087.75 €

Remarque :
On constate que les taux t_p et t_e conduisent à des valeurs à peu près identiques de l'épargne E.
En fait, lorsque τ est petit, on a : $(1+\tau)^{1/m} = 1 + \tau/m - [(m-1)/(2m^2)]\tau^2 + o(\tau^2)$. D'où : $(1+\tau)^{1/m} - 1 \sim \tau/m$

Mathématique & Informatique

Problème 3 : REMBOURSEMENT D'UN EMPRUNT

Une somme A est empruntée à un taux d'intérêt annuel fixe τ.
On fait m remboursements par an pendant n années.
Le nombre total de remboursements est $N = m \times n$.
Quel est le montant R des remboursements :

1) dans le cas où R est constant.

2) dans le cas où R est dégressif (on rembourse alors une part constante de capital égale à A/N).

Pour chaque remboursement, quelle est la part intérêt et quelle est la part capital ?

Formules :

Désignons par t le taux équivalent à τ pour la périodicité m, R_c le montant des remboursements constants, et R_i le montant du i^e remboursement dégressif.

$$R_c = \left[\frac{t(1+t)^N}{(1+t)^N - 1}\right] A$$

$$R_i = \left[\frac{1+(N+1-i)t}{N}\right] A$$

Exemple : A = 100 000 € , τ = 4.5 % , m = 12 , n = 15 ans

Echéancier des remboursements constants.

```
A = 100000 ; taux = 4.5/100 ; m = 12 ; n = 15 ;
printf('\n\n ************************************')
printf('\n Capital (euros) :   %6.2f', A)
printf('\n Taux (%%) :   %2.2f', taux*100)
printf('\n Périodicité :   %i', m)
printf('\n Années : %i', n)
printf('\n ************************************\n')
N = m*n ; t = (1+taux)^(1/m) - 1 ;
A0 = A ; S = 0 ;
R = t*(1+t)^N/((1+t)^N - 1)*A ;
for i = 1:N
   I = A*t ; C = R - I ; A = A - C ;
   M = round(100*R)/100 ; PC = round(100*C)/100 ;
   PI = M - PC ; S = S + M ;
   printf('\n\n Remboursement numéro %i', i)
   printf('\n ======================')
   printf('\n Montant : %6.2f', M)
   printf('\n Part intérêt : %6.2f', PI)
   printf('\n Part capital : %6.2f', PC)
end
printf('\n\n ************************************')
printf('\n SOMME TOTALE VERSEE :   %6.2f', S)
printf('\n INTERETS VERSES :   %6.2f', S - A0)
printf('\n ************************************\n')
```

donne :

R = 760.39 € (constant)
SOMME TOTALE VERSEE : 136 870.20 €
INTERETS VERSES : 36 870.20 €

Echéancier des remboursements dégressifs.

```
A = 100000 ; taux = 4.5/100 ; m = 12 ; n = 15 ;
printf('\n\n ************************************')
printf('\n Capital (euros) :  %6.2f', A)
printf('\n Taux (%%) :  %2.2f', taux*100)
printf('\n Périodicité :  %i', m)
printf('\n Années : %i', n)
printf('\n ************************************\n')
N = m*n ; t = (1+taux)^(1/m) - 1 ;
A0 = A ; S = 0 ;
C = A/N ;
for i = 1:N
  I = A*t ; R = C + I ; A = A - C ;
  M = round(100*R)/100 ; PC = round(100*C)/100 ;
  PI = M - PC ; S = S + M ;
  printf('\n\n Remboursement numéro %i', i)
  printf('\n ========================')
  printf('\n Montant : %6.2f', M)
  printf('\n Part intérêt : %6.2f', PI)
  printf('\n Part capital : %6.2f', PC)
end
printf('\n\n ************************************')
printf('\n SOMME TOTALE VERSEE :  %6.2f', S)
printf('\n INTERETS VERSES :  %6.2f', S - A0)
printf('\n ************************************\n')
```

donne :

R = 923.04 € à 557.60 € (dégressif)
SOMME TOTALE VERSEE : 133 257.02 €
INTERETS VERSES : 33 257.02 €

Problème 4 : CALCUL D'UN TAUX D'INTERET

Le remboursement d'une somme A s'effectue par le versement d'un montant constant R, à raison de m paiements par an pendant n années.
Quel est le taux d'intérêt annuel τ du prêt ?

Ce problème est l'inverse du précédent, il consiste à trouver τ connaissant les valeurs de A, R, m, n.
On fait appel à la fonction Scilab *fsolve*.

Exemple : A = 100 000 € , R = 760.39 € , m = 12 , n = 15 années

```
A = 100000 ; R = 760.39 ; m = 12 ; n = 15 ;
deff('y = f(t)', 'y = ((1+t)^(m*n) - 1)*R - t*(1+t)^(m*n)*A')
t = fsolve(1, f) ; taux = round(((1+t)^m - 1)*1D4)/1D4
```

donne : taux = 0.045 soit τ = 4.5 %

Problème 5 : TEG

Dans un remboursement d'emprunt, divers frais annexes (frais d'assurance, frais de dossier, ...) s'ajoutent à la somme initiale empruntée.
Le **taux effectif global** (TEG) intègre tous ces frais et correspond aux remboursements réels. Il est supérieur au **taux nominal annuel** (TNA) qui ne prend pas en compte les frais du crédit. Le TEG d'un prêt doit obligatoirement être précisé afin de pouvoir mieux comparer les différentes propositions des établissements bancaires. Il doit rester inférieur au taux de l'usure, taux maximum régi par la loi pour chaque type de prêt et régulièrement mis à jour tous les trimestres :
⇨ http://www.banque-france.fr/fr/statistiques/taux/usure.htm
En ce qui concerne les calculs, le TEG est le taux qu'il faudrait appliquer au capital emprunté diminué des frais de dossier, de garantie bancaire et autres frais, pour obtenir les mêmes remboursements assurance comprise.

Exemple : A = 100 000 € , τ = 4.5 % , m = 12 , n = 15 ans

On se place dans le cas des versements mensuels constants : R = 760.39 €.

1) Prise en compte des frais d'assurance

Ils représentent un pourcentage du capital emprunté.
Par exemple : τ_a = 0.36 % ⇒ coût supplémentaire de 100 000 × 0.0036 / 12 = 30 € par mois.
Le montant des mensualités augmente : R = 760.39 + 30 = 790.39 €.
Le TEG vaut (en utilisant le mode de calcul du problème 4) : 5.11 %

2) Prise en compte des frais de dossier

Ils représentent également un pourcentage du capital emprunté.
Par exemple : τ_d = 1 % ⇒ coût supplémentaire de 100 000 × 0.01 = 1 000 € au total.
Pour calculer le nouveau TEG, on retire les frais de dossier du capital emprunté, qui devient donc 99 000 €, puis on calcule le taux d'intérêt qui conduit aux mêmes remboursements que précédemment, c'est-à-dire 790.39 €.
Le TEG vaut (en utilisant le mode de calcul du problème 4) : 5.27 %

Conclusion : le TNA du prêt est de 4.50 % et le TEG est de 5.27 %.

Remarque :

Le calcul du TEG est plus complexe lorsqu'il faut tenir compte des paiements différés.
⇨ http://www.cbanque.com/credit/teg2.php fournit une feuille de calcul pour Microsoft Excel.
L'équation générale, dite des intérêts composés, est la suivante :

$$\sum_{k=1}^{k=m} \frac{A_k}{(1+i)^{t_k}} = \sum_{p=1}^{p=n} \frac{A_p}{(1+i)^{t_p}}$$

Avec :
i : le taux effectif global annuel,
k : le numéro d'ordre d'un déblocage de fond,
m : le numéro d'ordre du dernier déblocage,
A_k : le montant du déblocage numéro k,
t_k : l'intervalle de temps entre le premier déblocage et le déblocage numéro k,
p : le numéro d'ordre d'une échéance de remboursement,
n : le numéro d'ordre de la dernière échéance,
A_p : le montant de l'échéance numéro p,
t_p : l'intervalle de temps entre le premier déblocage et l'échéance numéro p.
Les intervalles de temps doivent être exprimés en années et fractions d'années.

SIMULATION D'UN PLAN ÉPARGNE LOGEMENT

Ce type de simulation se trouve facilement sur Internet. Par exemple, sur le site de la Caisse d'Epargne :
http://www.caisse-epargne.fr/particuliers/simulations-en-ligne.aspx
Un PEL (Plan Epargne Logement) est un livret d'épargne adapté à un projet immobilier, il est rémunéré à un taux annuel fixe pour une durée d'au moins 4 ans et ouvre droit à un prêt.
On effectue un versement initial, puis des versements mensuels pendant toute la durée du plan. Les sommes placées produisent des intérêts en fonction d'un **taux bimensuel proportionnel**, comme pour le Livret A.
Au début de chaque année, les intérêts sont ajoutés à l'épargne en cours pour produire de nouveaux intérêts.
Les prélèvements sociaux sont payés sur les intérêts acquis chaque année.

Prenons l'exemple suivant :

Date d'ouverture :	14/10/2012	Durée du plan :	5 ans
Versement initial :	1 500 €	Versement mensuel :	500 €
Taux d'intérêt :	2.5 %	Prélèvements sociaux :	15.5 %

Le programme Scilab de simulation s'écrit :

```
Date = [14,10,2012] ;
duree = 5 ;
apport = 1500 ;
mensualite = 500 ;
taux_1 = 2.5/100 ;
taux_2 = 15.5/100 ;

n = duree*24 ;
epargne = apport ;
t = taux_1/24 ;
q = 2*Date(2) - 1 + floor(Date(1)/16) ;
flag = modulo(q,2) ;
while n > 0
  I = 0 ;
  c = 0 ;
  for k = q+1:min(n,24)
    if modulo(k,2) <> flag then
      epargne = epargne + mensualite ;
    end
    I = I + epargne*t ;
    c = c + 1 ;
  end
  epargne = epargne + I ;
  n = n - c ;
  q = 0 ;
end

capital = apport + 12*mensualite*duree ;
interets = epargne - capital ;
prelevements = interets*taux_2 ;
capital = round(capital*100)/100
interets = round(interets*100)/100
prelevements = round(prelevements*100)/100
```

Après exécution, les résultats s'affichent à la Console :

```
capital      =    31 500.00 €
intérêts     =     2 163.73 €
prélèvements =       335.38 €
```

Progression arithmético-géométrique

Une suite récurrente (u_n) définie par la relation :

$$\begin{cases} u_n = q\,u_{n-1} + r \\ u_0 \quad donné \end{cases}$$

est appelée *progression arithmético-géométrique*. Les constantes q et r sont les *raisons* de la progression.
On démontre que l'expression du terme général est :

$$u_n = q^n u_0 + \left(\frac{q^n - 1}{q - 1}\right) r$$

et que la somme des n premiers termes est :

$$S = \sum_{i=0}^{n-1} u_i = \left(\frac{q^n - 1}{q - 1}\right) u_0 + \left[\frac{q^n - nq + n - 1}{(q-1)^2}\right] r$$

Application : Epargne

Reprenons le problème 2.
On peut exprimer algébriquement l'épargne E dans le cas des intérêts composés et du taux équivalent.
Soit N = m*n le nombre total de versements d'un montant *a* et C_i le capital acquis après le i^e versement (i = 1, ... , N).
On a : $C_i = (C_{i-1}+a) + (C_{i-1}+a)*t_e = (1+t_e)*C_{i-1} + (1+t_e)*a$, avec $C_0 = A$.
La progression arithmético-géométrique (C_i) de raisons $\{(1+t_e) , (1+t_e)a\}$ a pour terme de rang N : $C_N = E$.
On obtient immédiatement :

$$E = (1+t_e)^N A + \left(1 + \frac{1}{t_e}\right)\left[(1+t_e)^N - 1\right] a$$

Application : Emprunt

Reprenons le problème 3.
Soit C_i le capital restant à rembourser après le i^e versement (i = 1, ... , N) d'un montant constant R_c.
On a : $C_i = C_{i-1} - (R_c - C_{i-1}*t) = (1+t)*C_{i-1} - R_c$, avec $C_0 = A$ et $C_N = 0$.
Dans l'égalité $C_N = (1+t)*C_{N-1} - R_c = 0$, reportons l'expression du terme de rang (N-1) de la progression arithmético-géométrique (C_i) de raisons $\{(1+t) , -R_c\}$. On déduit, après simplifications :

$$R_c = \left[\frac{t\,(1+t)^N}{(1+t)^N - 1}\right] A$$

Note

Pour un remboursement dégressif, le calcul est plus simple. On a :
$C_i = A - i*A/N$ et $R_i = A/N + C_{i-1}*t$ (i = 1, ... , N et $C_0 = A$)
D'où :

$$R_i = \left[\frac{1 + (N+1-i)t}{N}\right] A$$

Mathématique & Informatique

LA CONJECTURE DE SYRACUSE

Soit (s_n) la suite de nombres entiers définie par la récurrence :

$$\begin{cases} s_n = \dfrac{s_{n-1}}{2} & \text{si } s_{n-1} \text{ est pair} \\ s_n = 3s_{n-1} + 1 & \text{si } s_{n-1} \text{ est impair} \\ s_0 \in \mathbb{N}^* \end{cases}$$

Cette mystérieuse suite - introduite en 1937 par Lothar **Collatz** de l'Université de Syracuse (USA) - converge vers 1.
La démonstration de cette propriété est un problème ouvert.
La suite obtenue à partir de s_0 est appelée *vol* s_0. Le plus grand entier de la suite est l'*altitude* maximale du vol, et le nombre d'étapes nécessaires pour atterrir à 1 est la *durée* du vol.

```
while %t
  s = input('s ? ') ;
  if s == [] | s == 0 then, break, else, printf('\n %i',s), end
  v = s ; d = 0 ; a = s ;
  while s > 1
    if modulo(s,2) == 0 then, s = s/2 ; else, s = 3*s+1 ; end
    d = d + 1 ; if s > a then, a = s ; end
    if modulo(d,10) == 0 then, printf(' -\n %i',s), else, printf(' - %i',s), end
  end
  printf('\n==> vol %i :   duree du vol = %i ; altitude maximale = %i\n',v,d,a)
end
```

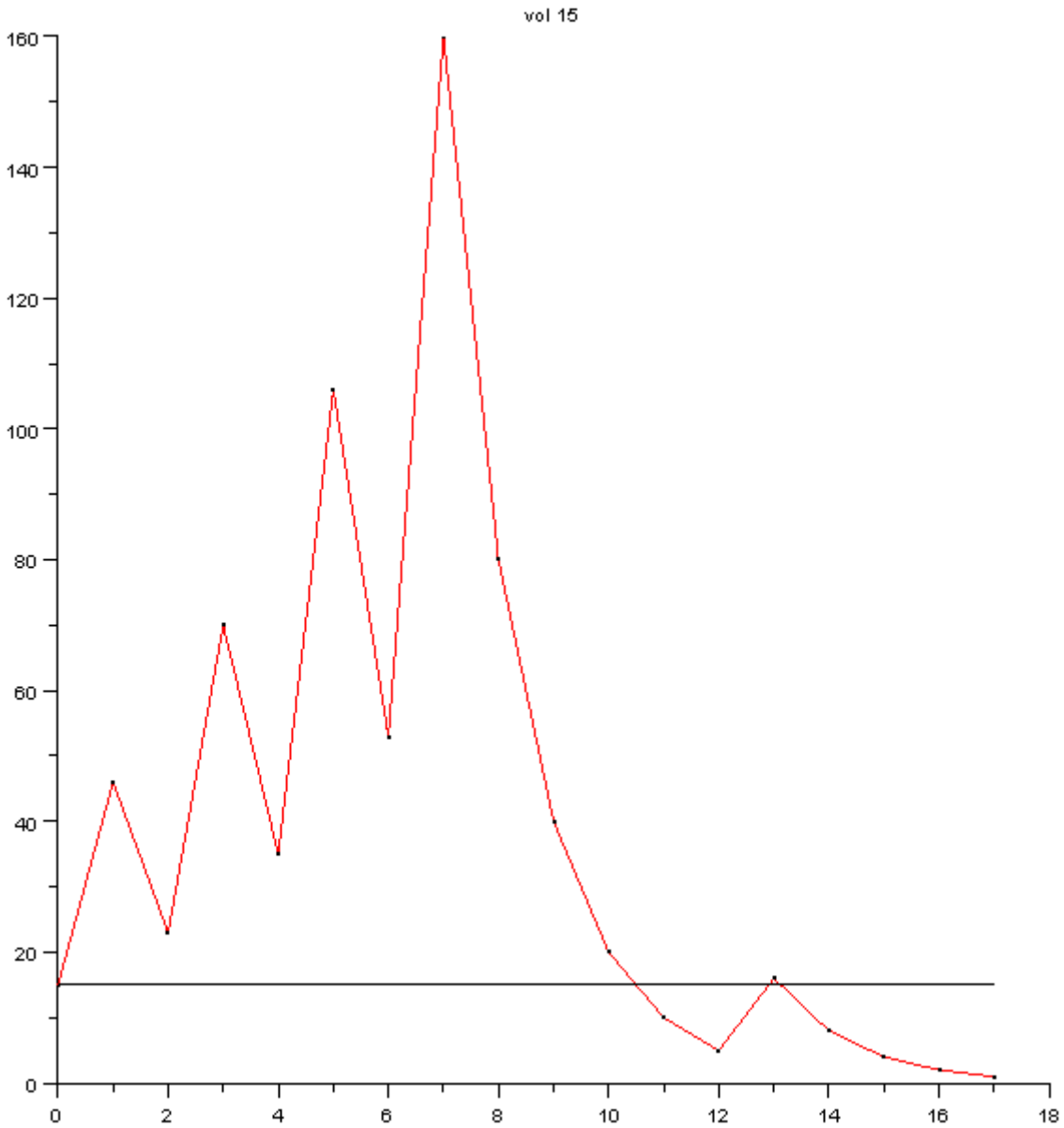
Description des vols numéros 1 à 10	Durée	Altitude maximale
1	0	1
2 - 1	1	2
3 - 10 - 5 - 16 - 8 - 4 - 2 - 1	7	16
4 - 2 - 1	2	4
5 - 16 - 8 - 4 - 2 - 1	5	16
6 - 3 - 10 - 5 - 16 - 8 - 4 - 2 - 1	8	16
7 - 22 - 11 - 34 - 17 - 52 - 26 - 13 - 40 - 20 - 10 - 5 - 16 - 8 - 4 - 2 - 1	16	52
8 - 4 - 2 - 1	3	8
9 - 28 - 14 - 7 - 22 - 11 - 34 - 17 - 52 - 26 - 13 - 40 - 20 - 10 - 5 - 16 - 8 - 4 - 2 - 1	19	52
10 - 5 - 16 - 8 - 4 - 2 - 1	6	16

Autres valeurs :
==> vol 77670 : durée du vol = 24 ; altitude maximale = 174760
==> vol 77671 : durée du vol = 231 ; altitude maximale = 1570824736
==> vol 77672 : durée du vol = 125 ; altitude maximale = 77672
==> vol 77673 : durée du vol = 107 ; altitude maximale = 589840

En partant de $s_0 = 2^n$, la suite de Syracuse est : $2^n \to 2^{n-1} \to 2^{n-2} \to \ldots \to 2 \to 1$, le vol 2^n a une durée égale à n, aussi grande que souhaitée. Donc, une suite de Syracuse peut avoir une infinité de termes avant d'atteindre 1.
Pour étudier la convergence des vols, on arrête les calculs dès que l'on passe au dessous du point de départ ($s_n < s_0$).
En pratique, la conjecture a été vérifiée pour toutes les valeurs $s_0 < 3.5\ 10^{17}$ (**Tomás Oliveira e Silva**).
⇨ Lire le document sur le Web : **http://www2.lifl.fr/~delahaye/SIME/Cours/Syracuse.PDF**

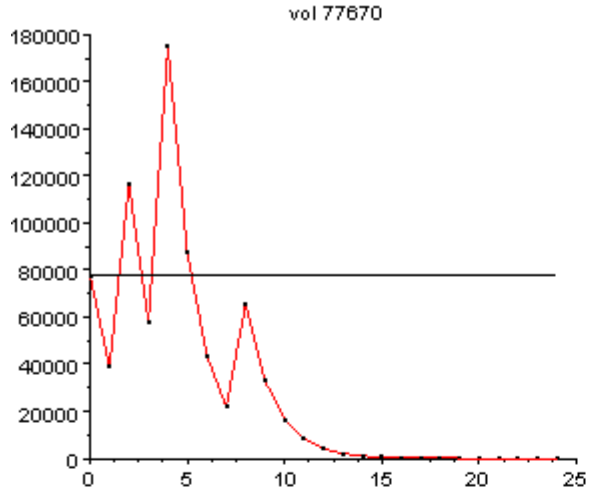
Représentation graphique d'un vol

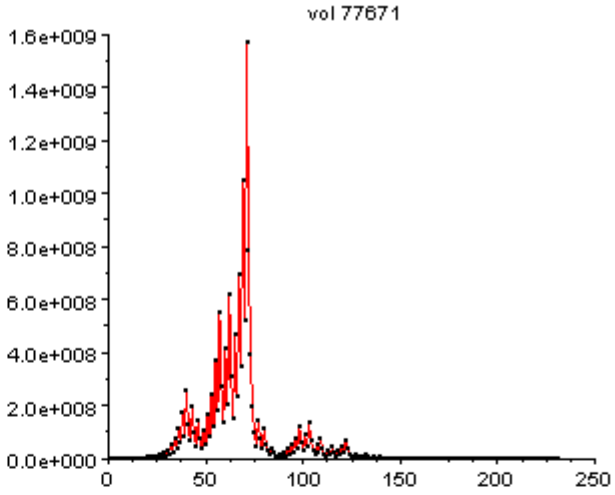
```
while %t
  s = input('s ? ') ;
  if s == [] | s == 0 then, xdel(), break, end
  i = 0 ; x = 0 ; y = s ;
  while s > 1
    if modulo(s,2) == 0 then, s = s/2 ; else, s = 3*s+1 ; end
    i = i + 1 ; x = [x,i] ; y = [y,s] ;
  end
  clf() ; xtitle('vol ' + string(y(1)))
  plot2d(x, y, style=5)
  plot2d(x, y, style=0)
  plot2d([x(1),x($)], [y(1),y(1)], style=1)
end
```

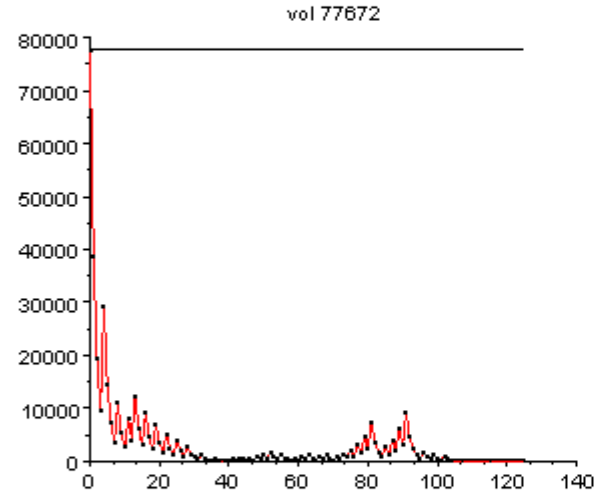


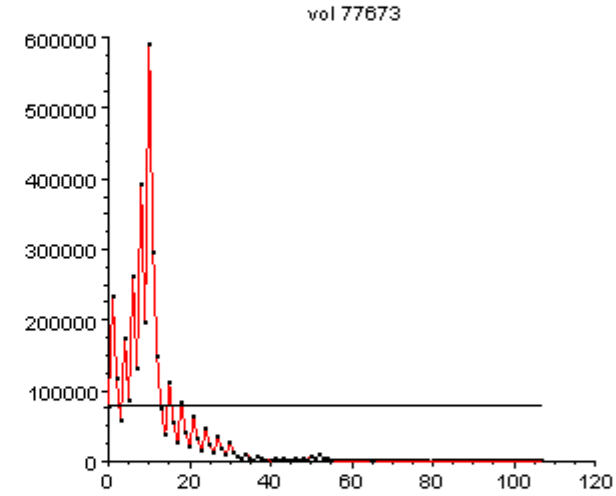
15 - 46 - 23 - 70 - 35 - 106 - 53 - 160 - 80 - 40 - 20 - 10 - 5 - 16 - 8 - 4 - 2 - 1
==> vol 15 : durée du vol = 17 ; altitude maximale = 160

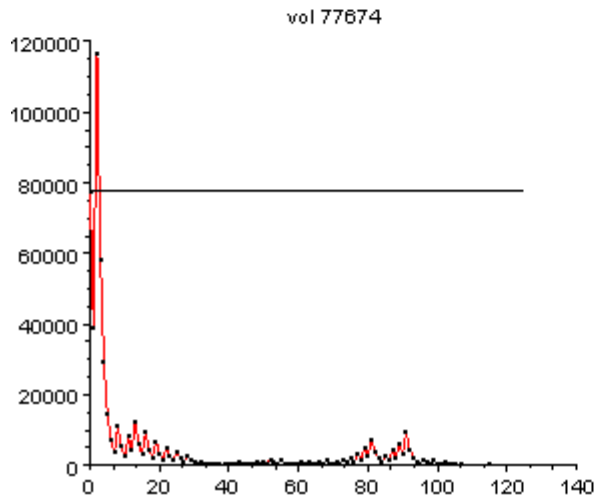
Mathématique & Informatique

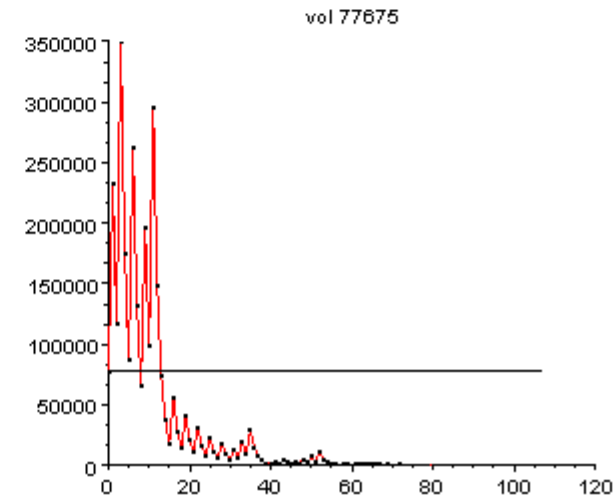




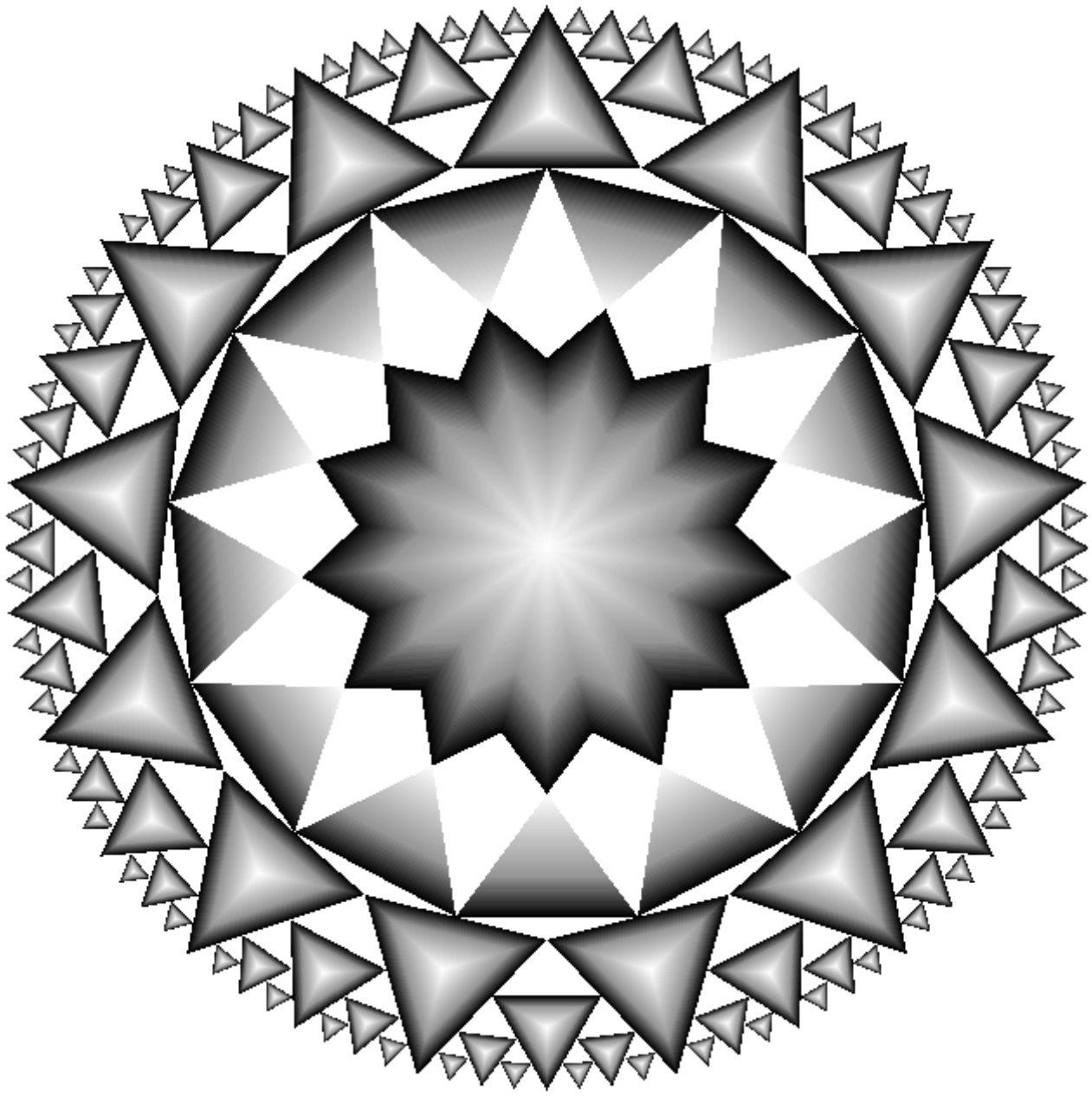








Mathématique & Informatique



Mathématique & Informatique

LES NOMBRES PREMIERS

Un nombre premier n'admet aucun diviseur autre que lui-même (on exclut le cas des nombres 0 et 1).
Il y a une infinité de nombres premiers (**Euclide**).
Le nombre de nombres premiers inférieurs à N est asymptotiquement égal à N / LogN (**Hadamard**).
Le premier nombre premier est 2, le millième 7 919, le dix-millième 104 729, le cent-millième 1 299 709, le millionième 15 485 863, ... A partir de 5, ils sont tous de la forme 6k ± 1.
Le programme suivant calcule et affiche les nombres premiers inférieurs (ou égal) à un nombre N.
Il permet d'obtenir, par une méthode de crible (**Eratosthène**), jusqu'à 100 000 nombres premiers.

```
MaxInt = 1299709 ;

//----- Lecture du nombre N

err = 1 ;
while err <> 0
  mess = 'Calcul des nombres premiers inférieurs à N = ?' ;
  rep = x_dialog(mess, '') ; if rep == [] then, abort, end
  err = execstr('N = evstr(rep)', 'errcatch') ; if err == 0 then, err = N - floor(N) ; end
end
if (N < 5)|(N > MaxInt) then
  if (N == 0)|(N == 1) then
    mess = 'Il n''y a pas de nombre premier inférieur à ' + string(N) ;
  elseif (N == 2) then
    mess = ['Il y a 1 nombre premier inférieur à ' + string(N) + ' :' ; '' ; '2'] ;
  elseif (N == 3)|(N == 4) then
    mess = ['Il y a 2 nombres premiers inférieurs à ' + string(N) + ' :' ; '' ; '2 3'] ;
  else
    mess = 'Le nombre N doit être compris entre 0 et ' + string(MaxInt) ;
  end
  messagebox(mess,"Résultats","scilab","OK","modeless") ; abort
end

//----- Sous programme de test d'un nombre premier

function Test(X)
  global Lp Np
  Compose = %f ; Premier = %f ; J = 1 ;
  while ~(Compose|Premier)
    P = Lp(J) ;
    Compose = modulo(X,P) == 0 ;
    Premier = P*P > X ;
    J = J + 1 ;
  end
  if Premier & (X <= N) then
    Np = Np + 1 ; Lp($+1) = X ;
    if modulo(Np,1000) == 0 then
      dt = getdate() ; h = dt(7) ; m = dt(8) ; s = dt(9) ;
      printf('\n Np = %i ----- Horloge = %i:%i:%i',Np,h,m,s)
    end
  end
endfunction
```

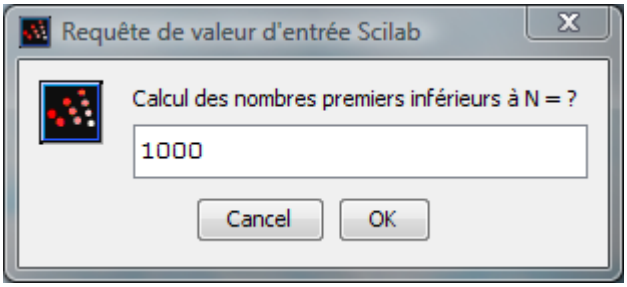
```
//----- Programme principal

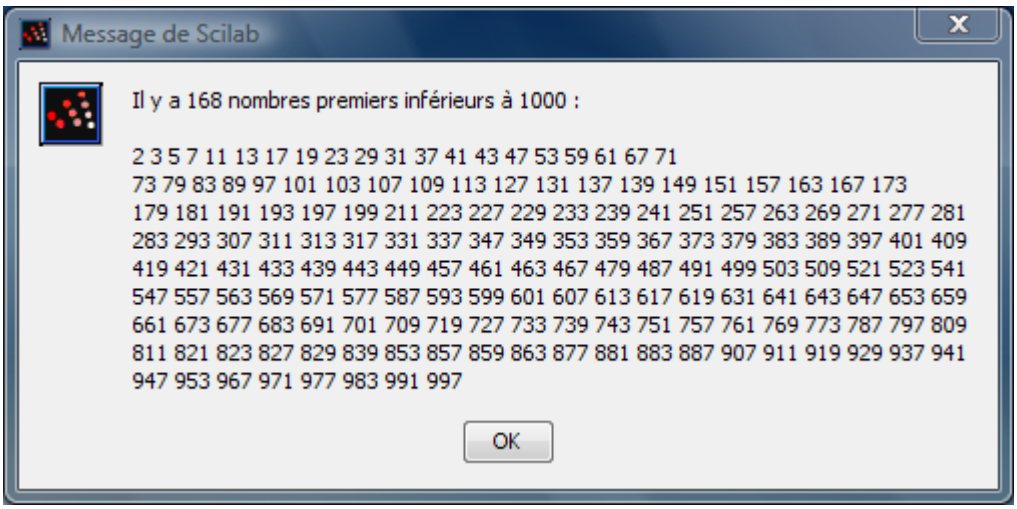
global Lp Np
Lp = list(2,3) ; Np = 2 ; I = 6 ;
while I <= N+1
  Test(I-1) ;
  Test(I+1) ;
  I = I + 6 ;
end

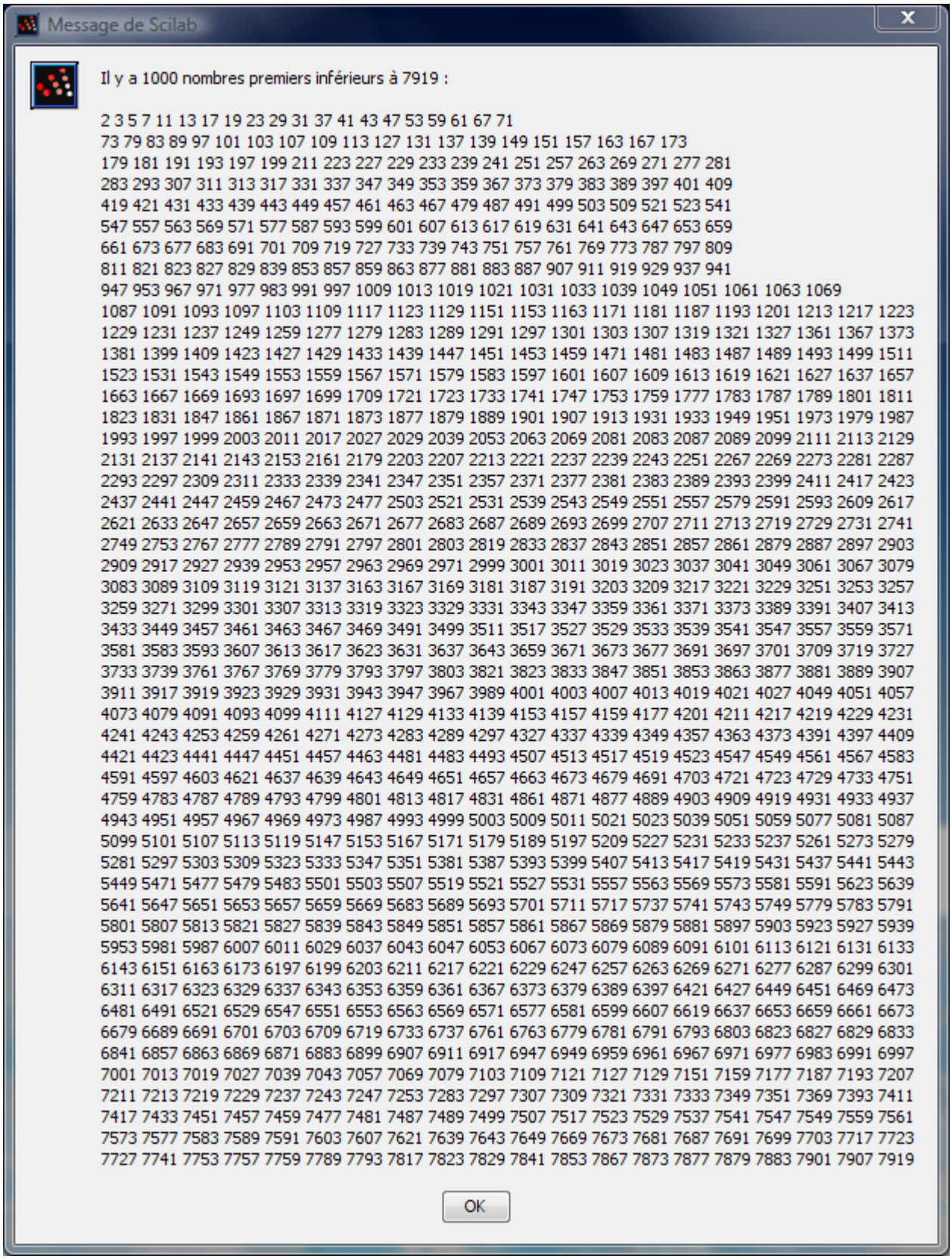
//----- Ecriture de la liste des nombres premiers

MaxCol = 20 ;
NbLig = ceil(Np/MaxCol) ;
Table = emptystr(NbLig+2,1) ;
Table(1) = 'Il y a ' + string(Np) + ' nombres premiers inférieurs à ' + string(N) + ' :' ;
Table(2) = '' ;
for i = 3:NbLig+2
  for j = 1:MaxCol
    k = (i-3)*MaxCol + j ;
    if k <= Np then, p = string(Lp(k)) ; else, p = '' ; end
    Table(i) = Table(i) + ' ' + p ;
  end
end
messagebox(Table,"Résultats","scilab","OK","modeless") ;
```

Par exemple, pour N = 1000, on obtient :



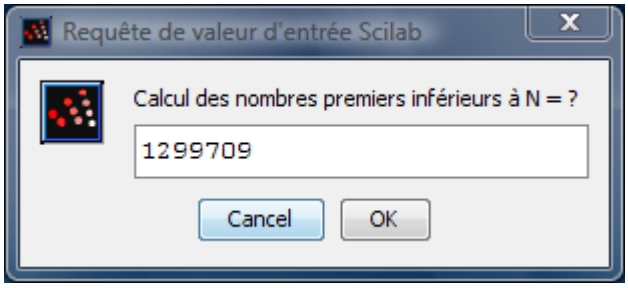



Message de Scilab
Il y a 1000 nombres premiers inférieurs à 7919 :
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71
73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281
283 293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409
419 421 431 433 439 443 449 457 461 463 467 479 487 491 499 503 509 521 523 541
547 557 563 569 571 577 587 593 599 601 607 613 617 619 631 641 643 647 653 659
661 673 677 683 691 701 709 719 727 733 739 743 751 757 761 769 773 787 797 809
811 821 823 827 829 839 853 857 859 863 877 881 883 887 907 911 919 929 937 941
947 953 967 971 977 983 991 997 1009 1013 1019 1021 1031 1033 1039 1049 1051 1061 1063 1069
1087 1091 1093 1097 1103 1109 1117 1123 1129 1151 1153 1163 1171 1181 1187 1193 1201 1213 1217 1223
1229 1231 1237 1249 1259 1277 1279 1283 1289 1291 1297 1301 1303 1307 1319 1321 1327 1361 1367 1373
1381 1399 1409 1423 1427 1429 1433 1439 1447 1451 1453 1459 1471 1481 1483 1487 1489 1493 1499 1511
1523 1531 1543 1549 1553 1559 1567 1571 1579 1583 1597 1601 1607 1609 1613 1619 1621 1627 1637 1657
1663 1667 1669 1693 1697 1699 1709 1721 1723 1733 1741 1747 1753 1759 1777 1783 1787 1789 1801 1811
1823 1831 1847 1861 1867 1871 1873 1877 1879 1889 1901 1907 1913 1931 1933 1949 1951 1973 1979 1987
1993 1997 1999 2003 2011 2017 2027 2029 2039 2053 2063 2069 2081 2083 2087 2089 2099 2111 2113 2129
2131 2137 2141 2143 2153 2161 2179 2203 2207 2213 2221 2237 2239 2243 2251 2267 2269 2273 2281 2287
2293 2297 2309 2311 2333 2339 2341 2347 2351 2357 2371 2377 2381 2383 2389 2393 2399 2411 2417 2423
2437 2441 2447 2459 2467 2473 2477 2503 2521 2531 2539 2543 2549 2551 2557 2579 2591 2593 2609 2617
2621 2633 2647 2657 2659 2663 2671 2677 2683 2687 2689 2693 2699 2707 2711 2713 2719 2729 2731 2741
2749 2753 2767 2777 2789 2791 2797 2801 2803 2819 2833 2837 2843 2851 2857 2861 2879 2887 2897 2903
2909 2917 2927 2939 2953 2957 2963 2969 2971 2999 3001 3011 3019 3023 3037 3041 3049 3061 3067 3079
3083 3089 3109 3119 3121 3137 3163 3167 3169 3181 3187 3191 3203 3209 3217 3221 3229 3251 3253 3257
3259 3271 3299 3301 3307 3313 3319 3323 3329 3331 3343 3347 3359 3361 3371 3373 3389 3391 3407 3413
3433 3449 3457 3461 3463 3467 3469 3491 3499 3511 3517 3527 3529 3533 3539 3541 3547 3557 3559 3571
3581 3583 3593 3607 3613 3617 3623 3631 3637 3643 3659 3671 3673 3677 3691 3697 3701 3709 3719 3727
3733 3739 3761 3767 3769 3779 3793 3797 3803 3821 3823 3833 3847 3851 3853 3863 3877 3881 3889 3907
3911 3917 3919 3923 3929 3931 3943 3947 3967 3989 4001 4003 4007 4013 4019 4021 4027 4049 4051 4057
4073 4079 4091 4093 4099 4111 4127 4129 4133 4139 4153 4157 4159 4177 4201 4211 4217 4219 4229 4231
4241 4243 4253 4259 4261 4271 4273 4283 4289 4297 4327 4337 4339 4349 4357 4363 4373 4391 4397 4409
4421 4423 4441 4447 4451 4457 4463 4481 4483 4493 4507 4513 4517 4519 4523 4547 4549 4561 4567 4583
4591 4597 4603 4621 4637 4639 4643 4649 4651 4657 4663 4673 4679 4691 4703 4721 4723 4729 4733 4751
4759 4783 4787 4789 4793 4799 4801 4813 4817 4831 4861 4871 4877 4889 4903 4909 4919 4931 4933 4937
4943 4951 4957 4967 4969 4973 4987 4993 4999 5003 5009 5011 5021 5023 5039 5051 5059 5077 5081 5087
5099 5101 5107 5113 5119 5147 5153 5167 5171 5179 5189 5197 5209 5227 5231 5233 5237 5261 5273 5279
5281 5297 5303 5309 5323 5333 5347 5351 5381 5387 5393 5399 5407 5413 5417 5419 5431 5437 5441 5443
5449 5471 5477 5479 5483 5501 5503 5507 5519 5521 5527 5531 5557 5563 5569 5573 5581 5591 5623 5639
5641 5647 5651 5653 5657 5659 5669 5683 5689 5693 5701 5711 5717 5737 5741 5743 5749 5779 5783 5791
5801 5807 5813 5821 5827 5839 5843 5849 5851 5857 5861 5867 5869 5879 5881 5897 5903 5923 5927 5939
5953 5981 5987 6007 6011 6029 6037 6043 6047 6053 6067 6073 6079 6089 6091 6101 6113 6121 6131 6133
6143 6151 6163 6173 6197 6199 6203 6211 6217 6221 6229 6247 6257 6263 6269 6271 6277 6287 6299 6301
6311 6317 6323 6329 6337 6343 6353 6359 6361 6367 6373 6379 6389 6397 6421 6427 6449 6451 6469 6473
6481 6491 6521 6529 6547 6551 6553 6563 6569 6571 6577 6581 6599 6607 6619 6637 6653 6659 6661 6673
6679 6689 6691 6701 6703 6709 6719 6733 6737 6761 6763 6779 6781 6791 6793 6803 6823 6827 6829 6833
6841 6857 6863 6869 6871 6883 6899 6907 6911 6917 6947 6949 6959 6961 6967 6971 6977 6983 6991 6997
7001 7013 7019 7027 7039 7043 7057 7069 7079 7103 7109 7121 7127 7129 7151 7159 7177 7187 7193 7207
7211 7213 7219 7229 7237 7243 7247 7253 7283 7297 7307 7309 7321 7331 7333 7349 7351 7369 7393 7411
7417 7433 7451 7457 7459 7477 7481 7487 7489 7499 7507 7517 7523 7529 7537 7541 7547 7549 7559 7561
7573 7577 7583 7589 7591 7603 7607 7621 7639 7643 7649 7669 7673 7681 7687 7691 7699 7703 7717 7723
7727 7741 7753 7757 7759 7789 7793 7817 7823 7829 7841 7853 7867 7873 7877 7879 7883 7901 7907 7919
OK

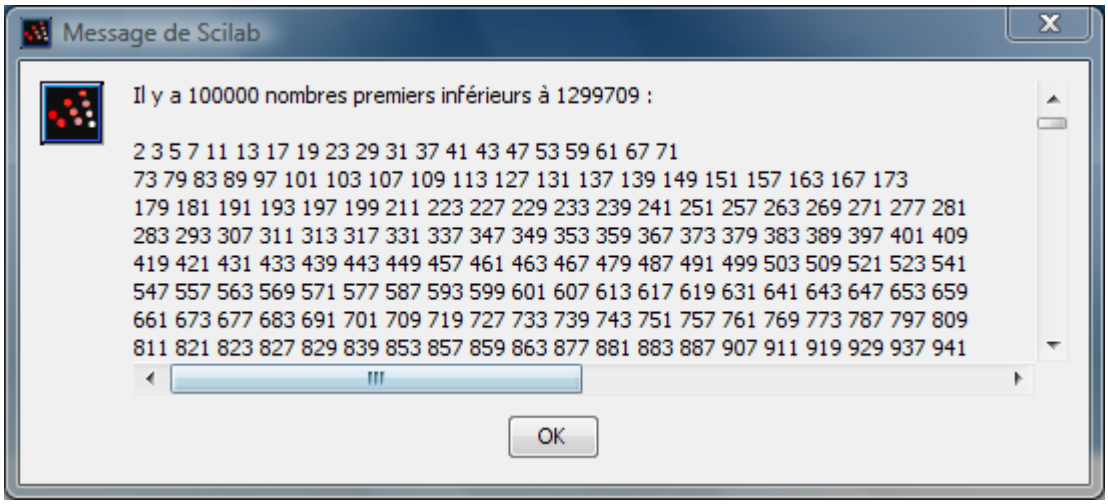
Remarque

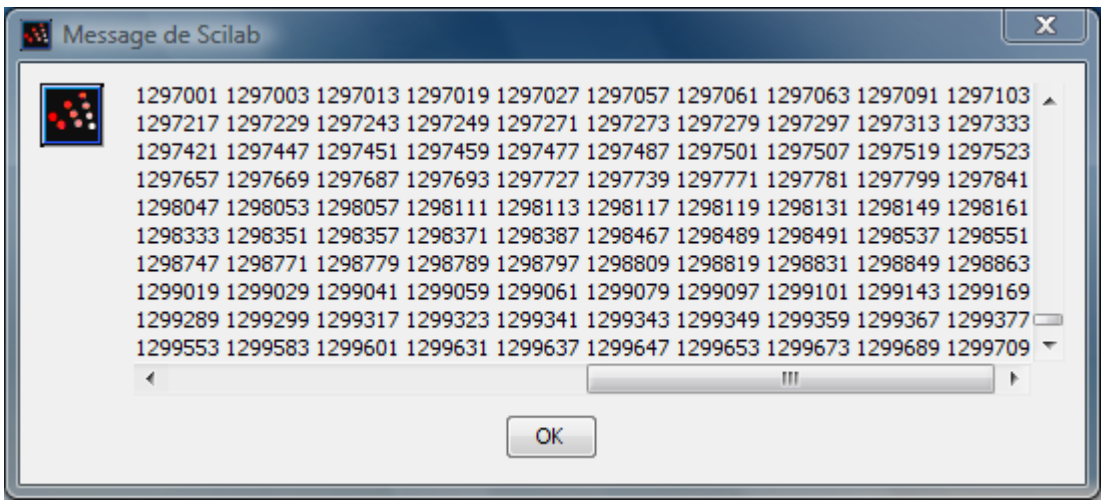
Il est facile de prouver que tous les nombres premiers sont de la forme 6k ± 1.
En effet, un nombre premier ne peut pas s'écrire 6k, ni 6k + 2, ni 6k + 3, ni 6k + 4, puisqu'on pourrait le diviser par 2, 2, 3, 2 respectivement. Il est donc de la forme 6k + 1 ou 6k + 5, c'est-à-dire 6k ± 1.
La réciproque est fausse, un nombre peut être de la forme 6k ± 1 sans être premier, par exemple 25.
De manière analogue on établit que tous les nombres premiers sont de la forme 2k + 1, 3k ± 1 ou 4k ± 1
Ajoutons que **Dirichlet** a démontré qu'il existe une infinité de nombres premiers de la forme ak + b
où a et b sont deux nombres entiers premiers entre eux (difficile).

Table des 100 000 nombres premiers



Après environ 30 mn de calcul sur un PC :





Théorème fondamental de l'arithmétique

Tout nombre entier n se décompose de manière unique en un produit de nombres premiers :

$$n = {p_1}^{\alpha_1} {p_2}^{\alpha_2} \cdots {p_k}^{\alpha_k}$$

Définissons les fonctions Scilab *premiers*, *facteurs* et *diviseurs* suivantes :

- La fonction *premiers(N)* retourne les N premiers nombres premiers sous la forme d'un tableau P d'entiers p_i
- La fonction *facteurs(n)* retourne les facteurs premiers d'un entier n sous la forme d'un tableau F de couples d'entiers (p_i , α_i)
- La fonction *diviseurs(n)* retourne les diviseurs d'un entier n sous la forme d'un tableau D d'entiers d_i

```
//------ Tableau ordonné des N premiers nombres premiers

function P = premiers(N)
  function Test(x)
    global P i
    if i >= N then, return, end
    Compose = %f ; Premier = %f ; k = 1
    while ~(Compose|Premier)
      p = P(k)
      Compose = modulo(x,p) == 0
      Premier = p*p > x
      k = k + 1
    end
    if Premier then
      i = i + 1
      P(i) = x
    end
  endfunction
  global P i
  if N == 0 then, P = [], return, end
  if N == 1 then, P = [2], return, end
  P = zeros(1,N) ; P(1) = 2 ; P(2) = 3
  i = 2 ; j = 6
  while i < N
    Test(j-1)
    Test(j+1)
    j = j + 6
  end
endfunction
```

Par exemple, l'instruction *premiers(100)* affiche :

```
 2   3   5   7   11  13  17  19  23  29  31  37  41  43  47  53  59  61  67  71  73  79  83  89  97
101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193
197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307
311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409 419 421
431 433 439 443 449 457 461 463 467 479 487 491 499 503 509 521 523 541
```

```
//------ Tableau des facteurs premiers d'un nombre n

function F = facteurs(n)
  F = []
  global P
  if n < 2 | n > P($)^2 then, return, end
  i = 1
  while n >= P(i)^2
    p = P(i)
    e = 0
    while modulo(n,p) == 0
      n = n/p
      e = e + 1
    end
    if e > 0 then
      F = [F ; [p,e]]
    end
    i = i + 1
  end
  if n <> 1 then
    F = [F ; [n,1]]
  end
endfunction
```

Programme :

```
while %t
  n = input('n ?') ;
  if n == [] then, break, end
  F = facteurs(n) ;
  printf('\n Décomposition du nombre %.0f :',n)
  for i = 1:size(F,1)
    printf('\n %.0f puissance %i',F(i,1),F(i,2))
  end
end
```

Exemple :

Décomposition du nombre 9876543210 :
2 puissance 1
3 puissance 2
5 puissance 1
17 puissance 2
379721 puissance 1

Remarques

- Soit M la valeur du plus grand nombre premier du tableau P : M = P($) où $ est la taille du tableau P
  Tout nombre entier n vérifiant $n \leq M^2$ admet un diviseur premier appartenant à P ou bien est premier.
  On a l'équivalence : p premier $\Leftrightarrow$ F = [p,1] (test de primalité des entiers inférieurs à M^2)
- Deux nombres sont premiers entre eux si et seulement si leurs facteurs premiers sont tous différents.
- La décomposition en facteurs premiers permet de calculer le **pgcd** et le **ppcm** de plusieurs nombres.
  Le pgcd est le produit des facteurs premiers communs, affectés du plus petit exposant.
  Le ppcm est le produit de tous les facteurs premiers, affectés du plus grand exposant.

```
//------ Tableau des diviseurs d'un nombre n

function D = diviseurs(n)
    D = []
    F = facteurs(n)
    if F == [] then, D = 1, return, end
    a = size(F,1)
    b = max(F(:,2)) + 2
    I = ones(1,a)
    T = zeros(a,b)
    for i = 1:a
        for j = 2:F(i,2)+1
            T(i,j) = j - 1
        end
        T(i,b) = F(i,2) + 1
    end
    loop = %t
    test = %t
    while loop
        if test then
            d = 1
            for i = 1:a
                d = d*F(i,1)^T(i,I(i))
            end
            D = [D,d]
            p = 0
        else
            I(p) = 1
        end
        p = p + 1
        I(p) = I(p) + 1
        test = I(p) <= T(p,b)
        loop = test | p < a
    end
    D = gsort(D,'c','i')
endfunction
```

Programme :

```
while %t
    n = input('n?') ;
    if n == [] then, break, end
    D = diviseurs(n) ;
    printf('\n Le nombre %.0f possède %i diviseurs :\n',n,length(D))
    for i = 1:length(D)
        printf('   %.0f',D(i))
    end
end
```

Exemple :

Le nombre 9876543210 possède 72 diviseurs :
1 2 3 5 6 9 10 15 17 18 30 34 45 51 85 90 102 153 170 255 289 306 510 578 765 867 1445 1530 1734 2601 2890 4335 5202 8670 13005 26010 379721 759442 1139163 1898605 2278326 3417489 3797210 5695815 6455257 6834978 11391630 12910514 17087445 19365771 32276285 34174890 38731542 58097313 64552570 96828855 109739369 116194626 193657710 219478738 290486565 329218107 548696845 580973130 658436214 987654321 1097393690 1646090535 1975308642 3292181070 4938271605 9876543210

Fichier de nombres premiers

Il n'est pas nécessaire d'exécuter la fonction *premiers* pour chaque nombre entier *n*.
On procède de la manière suivante :

1) Créer, une fois pour toutes, un fichier "PREMIERS" contenant N nombres premiers

2) Charger en mémoire le fichier "PREMIERS" dans un tableau P

3) Décomposer chaque entier *n* en utilisant la fonction *facteurs*

1) Création du fichier PREMIERS

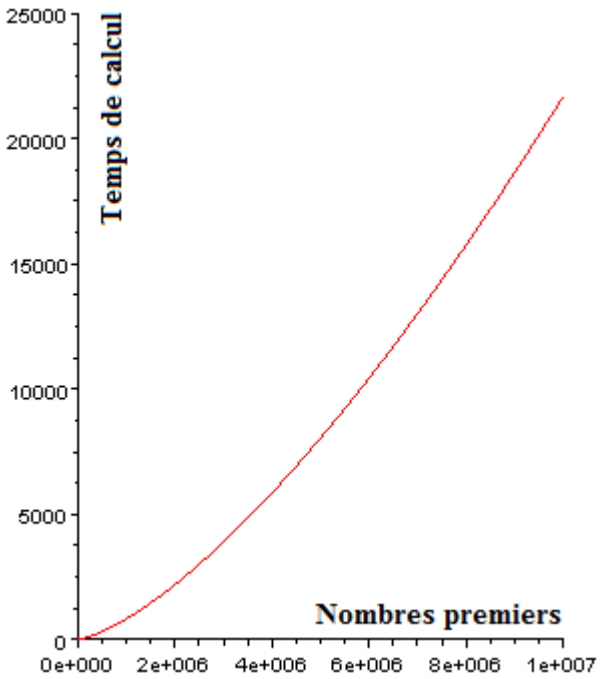
```
N = 1e7 ;
P = premiers(N) ;
fd = mopen("PREMIERS",'wb') ;
mput(P) ;
mclose(fd) ;
```

2) Chargement du tableau P

```
global P
stacksize(2e7) ;
fd = mopen("PREMIERS",'rb') ;
P = mget(1e7) ;
mclose(fd) ;
```

3) Décomposition d'un entier *n*

```
while %t
  n = input('n ?') ;
  if n == [] then, break, end
  F = facteurs(n) ;
  printf('\n Décomposition du nombre %.0f :',n)
  for i = 1:size(F,1)
    printf('\n %.0f puissance %i',F(i,1),F(i,2))
  end
end
```



Le temps de calcul pour obtenir un fichier de N nombres premiers en utilisant la fonction *premiers* est exponentiel (voir l'allure de la courbe ci-dessus). Pour $N = 10^7$ il faut plusieurs heures avec un micro – ordinateur.
Le fichier **PREMIERS** contient 10 000 000 de nombres premiers, allant de 2 à 179 424 673 ; sa taille est de 40 Mo.
Il permet de calculer rapidement les facteurs premiers de tout nombre entier $n \leq (179424673)^2 \approx 3.2\ 10^{16}$

Remarque

L'instruction P = *mget*(n_e) charge un nombre n_e d'éléments du fichier ouvert en lecture dans la variable tableau P.
Si le fichier contient moins de n_e éléments, alors tout le fichier est chargé dans P, sans qu'il soit nécessaire de connaître le nombre total d'éléments du fichier en question. Pour charger d'un coup l'intégralité d'un fichier dans un tableau P, on donnera donc à n_e une borne supérieure du nombre de ses éléments, sans dépasser la taille disponible de la pile Scilab (exécuter l'instruction *stacksize('max')* si besoin). Une boucle qui chargerait les éléments les uns à la suite des autres, en testant la fin de fichier avec la fonction *meof*(), serait plus longue à exécuter avec un grand fichier. De même en utilisant la fonction *mfscanf*.

EXEMPLES

n = 27 137 413
Décomposition du nombre 27137413 :
277 puissance 1
313 puissance 2

n = 609 576 704
Décomposition du nombre 609576704 :
2 puissance 8
11 puissance 3
1789 puissance 1

n = 1 000 123 456 789
Décomposition du nombre 1000123456789 :
37 puissance 1
109147 puissance 1
247651 puissance 1

n = 2 793 563 544 114 613
Décomposition du nombre 2793563544114613 :
41161739 puissance 1
67867967 puissance 1

n = 3 143 016 821 730 911
Décomposition du nombre 3143016821730911 :
3143016821730911 puissance 1

Quel est le pgcd et le ppcm des 3 nombres : $n_1 = 564\ 362\ 253$; $n_2 = 662\ 718\ 875$; $n_3 = 759\ 402\ 000$?

Décomposition du nombre 564362253 :
3 puissance 2
7 puissance 6
13 puissance 1
41 puissance 1

Décomposition du nombre 662718875 :
5 puissance 3
7 puissance 3
13 puissance 1
29 puissance 1
41 puissance 1

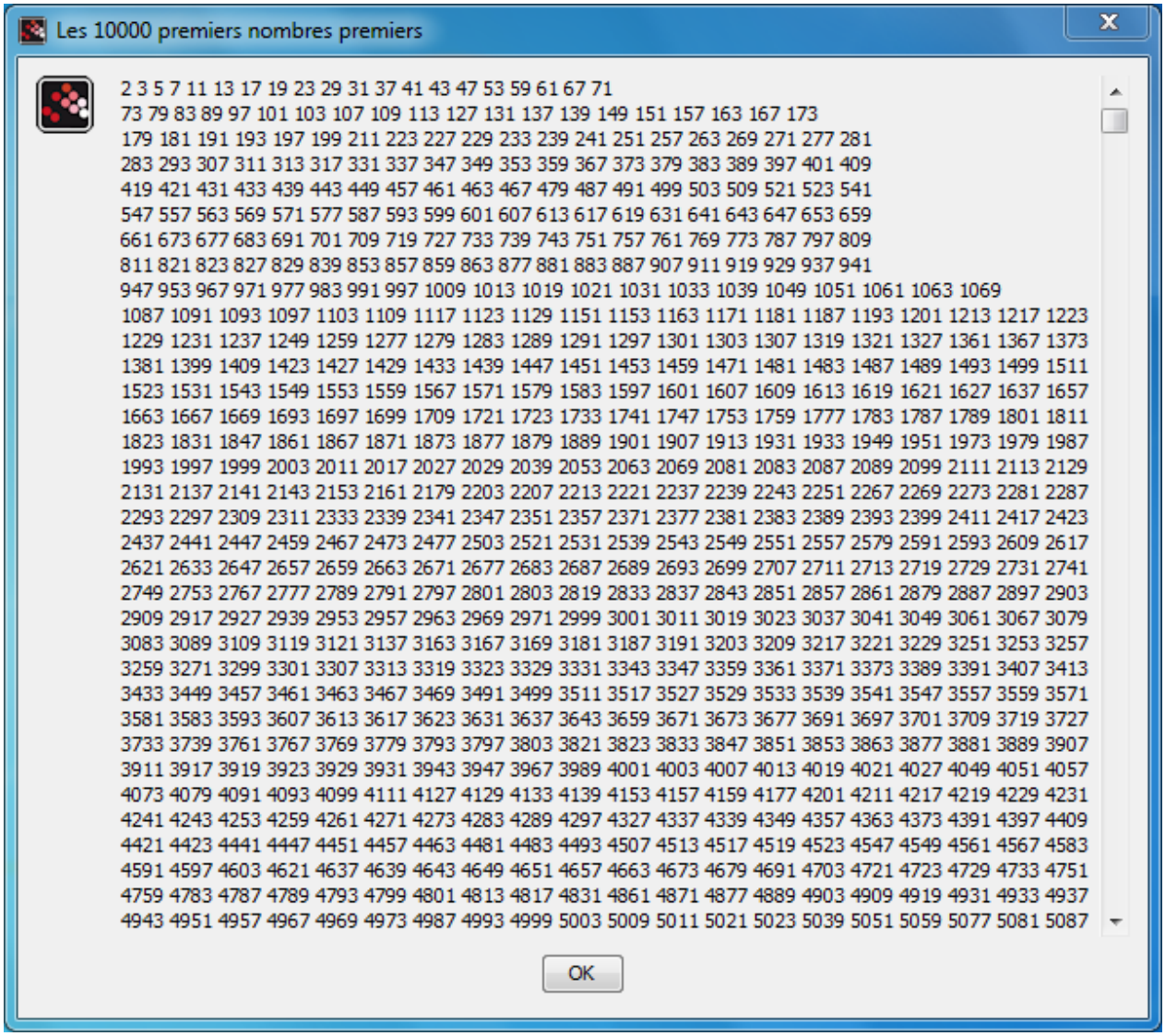
Décomposition du nombre 759402000 :
2 puissance 4
3 puissance 3
5 puissance 3
7 puissance 3
41 puissance 1

D'où :

$\text{pgcd}(n_1, n_2, n_3) = 7^3 . 41 = 14\ 063$
$\text{ppcm}(n_1, n_2, n_3) = 2^4 . 3^3 . 5^3 . 7^6 . 13 . 29 . 41 = 98\ 199\ 032\ 022\ 000$

Le programme suivant ouvre une fenêtre contenant les *N* premiers nombres premiers à raison de *l* valeurs par ligne.

```
N = 10000 ;
l = 20 ;
//------ Lecture à partir du fichier des nombres premiers
stacksize(2e7) ;
fd = mopen("PREMIERS",'rb') ;
P = mget(N) ;
mclose(fd) ;
//----- Ecriture de la table des nombres premiers
NbCol = l ;
NbLig = ceil(N/l) ;
Table = emptystr(NbLig,1) ;
for i = 1:NbLig
  for j = 1:NbCol
    k = (i-1)*NbCol + j ;
    if k <= N then, p = string(P(k)) ; else, p = '' ; end
    Table(i) = Table(i) + ' ' + p ;
  end
end
messagebox(Table,"Les " + string(N) + " premiers nombres premiers") ;
```



Application 1 : indicatrice d'Euler

L'indicatrice d'**Euler** est la fonction φ qui à un entier n associe le nombre d'entiers strictement positifs inférieurs à n et premiers avec n : $\varphi(n) = \text{Card}(\{m,\ 0 < m \le n \mid \text{pgcd}(m,n) = 1\})$

On a : $\varphi(1) = 1$. On peut définir la fonction φ sur l'intégralité de l'ensemble $\mathbb{N}$ en convenant que $\varphi(0) = 0$.
Par exemple, si l'on prend $n = 10$, il y a 4 entiers inférieurs à 10 qui sont premiers avec 10 : 1, 3, 7, 9. Donc : $\varphi(10) = 4$.
Pour un nombre premier p, tous les entiers inférieurs à p sont premiers avec p. Donc : $\varphi(p) = p - 1$.
Plus généralement, en décomposant n en facteurs premiers :

$$n = {p_1}^{\alpha_1}\, {p_2}^{\alpha_2} \cdots {p_k}^{\alpha_k} \qquad \Rightarrow \qquad \varphi(n) = \prod_{i=1}^{k} (p_i - 1)\, {p_i}^{\alpha_i - 1}$$

Propriété 1 : soit un entier a premier avec n

$$a^{\varphi(n)} \equiv 1 \quad (n)$$

En particulier si n est un nombre premier p, on retrouve le petit théorème de **Fermat** : $a^{p-1} \equiv 1 \quad (p)$

Propriété 2 : considérons tous les diviseurs d de n (1 et n compris)

$$\sum_{d|n} \varphi(d) = n$$ (relation de **Gauss**)

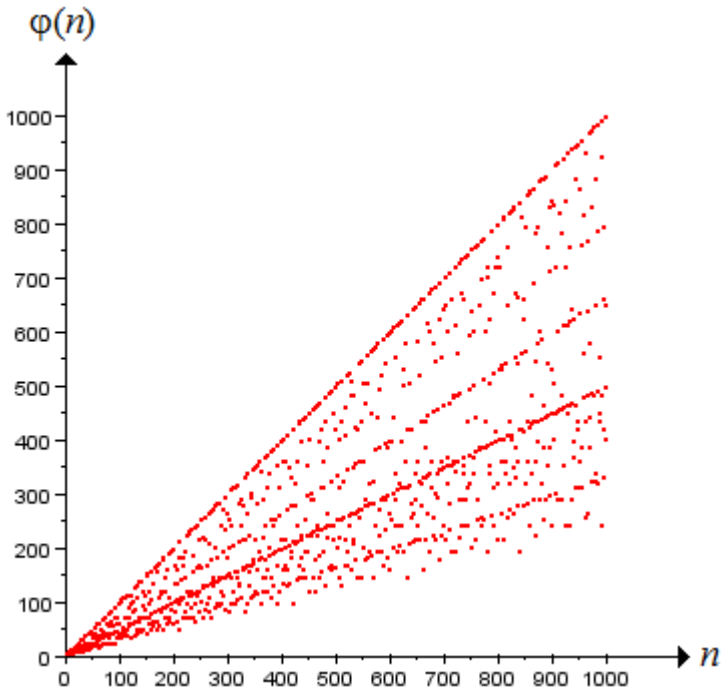
La fonction Scilab *facteurs* fournit les nombres premiers p_i $(1 \le i \le k)$ et leurs exposants α_i tels que : $n = {p_1}^{\alpha_1}\, {p_2}^{\alpha_2} \cdots {p_k}^{\alpha_k}$

Le résultat se présente sous la forme d'un tableau F à k lignes et 2 colonnes : $F(i,1) = p_i$, $F(i,2) = \alpha_i$

Ecrivons une fonction Scilab *indicatrice* qui cacule $\varphi(n)$:

```
function phi = indicatrice(n)
    if n == 0 | n == 1 then, phi = n, return, end
    F = facteurs(n)
    phi = 1
    for i = 1:size(F,1)
        phi = phi*(F(i,1) - 1)*F(i,1)^(F(i,2) - 1)
    end
endfunction

global P
stacksize(2e7) ;
fd = mopen("PREMIERS",'rb') ;
P = mget(1e7) ;
mclose(fd) ;
while %t
    n = input('n?') ;
    if n == [] then, break, end
    phi = indicatrice(n) ;
    printf('\n phi(%.0f) = %.0f',n,phi)
end
```



Exemples :

$n = 15485863 \Rightarrow \varphi(n) = 15485862$
$n = 1234567890 \Rightarrow \varphi(n) = 329040288$
$n = 137438691328 \Rightarrow \varphi(n) = 68719214592$
$n = 3080978096305787 \Rightarrow \varphi(n) = 3069609173440920$

Application 2 : fonction de Möbius

La fonction de **Möbius** est la fonction μ qui à un entier *n* associe la valeur 0 ou ±1 suivant que *n* est, ou n'est pas, divisible par un carré :

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k} \quad \Rightarrow \quad \mu(n) = \begin{cases} (-1)^k & \text{si } \alpha_1 = \alpha_2 = \cdots = \alpha_k = 1 \\ 0 & \text{sinon} \end{cases}$$

NB $\mu(n) = 0 \Leftrightarrow \exists\, \alpha_i > 1 \Leftrightarrow p_i^2 \mid n$

On convient que : $\mu(1) = 1$ et $\mu(0) = 0$
Pour un nombre premier p : $\alpha_1 = k = 1$. Donc : $\mu(p) = -1$

Exemples

$\mu(12) = \mu(2^2.3^1) = 0$; $\mu(13) = \mu(13^1) = -1$; $\mu(14) = \mu(2^1.7^1) = 1$

Propriété 1 : soit d un diviseur de n (1 et n compris)

$$\sum_{d \mid n} \mu(d) = 0$$

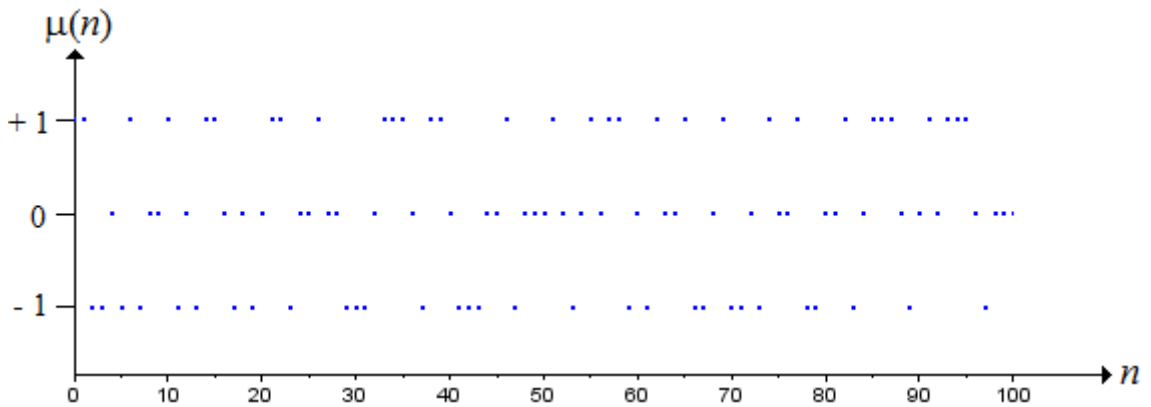
Propriété 2 : soit $\varphi(n)$ l'indicatrice d'**Euler** de n

$$\varphi(n) = \sum_{d \mid n} \mu(d) \frac{n}{d}$$

Ecrivons une fonction Scilab *Mobius* qui calcule $\mu(n)$:

```
function mu = Mobius(n)
    if n == 0 | n == 1 then, mu = n, return, end
    F = facteurs(n)
    mu = 0
    if find(F(:,2) > 1) == [] then
        mu = (-1)^size(F,1)
    end
endfunction
```

Graphe de la fonction de **Möbius** pour les 100 premiers entiers :



Application 3 : recherche des nombres parfaits

Soit : $n = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k}$

Le nombre *N*, la somme S et le produit P des diviseurs de *n* (1 et *n* compris) sont donnés par les formules :

$$N = \prod_{i=1}^{k} (\alpha_i + 1) \qquad S = \prod_{i=1}^{k} \frac{p_i^{\alpha_i + 1} - 1}{p_i - 1} \qquad P = n^{\frac{\prod_{i=1}^{k} (\alpha_i + 1)}{2}}$$

Un nombre parfait est un nombre égal à la somme de ses diviseurs, autres que lui-même.
Avec les formules ci-dessus, *n* est parfait si : $n = S - n$
Remarquons qu'un nombre premier *p* n'est jamais parfait puisque : $p \neq S - p = 1$
Les nombres parfaits sont rares.
En 2009, seulement 47 nombres parfaits étaient connus : **http://www.recreomath.qc.ca/dict_parfait_nombre.htm**
Le premier nombre parfait est 6 car 1 + 2 + 3 = 6, le deuxième est 28 car 1 + 2 + 4 + 7 + 14 = 28
Le 12e est $2^{126}(2^{12} - 1)$, nombre à 77 chiffres :
14 474 011 154 664 524 427 946 373 126 085 988 481 573 677 491 474 835 889 066 354 349 131 199 152 128
Le 47e est $2^{43\,112\,608}(2^{43\,112\,609} - 1)$, nombre à $E[\log_{10}(n)] + 1 = 25\,956\,377$ chiffres !
Euler a démontré la propriété suivante : tous les nombres parfaits pairs sont de la forme $2^{p-1}(2^p - 1)$ où $2^p - 1$, *p* premier, est un nombre premier de **Mersenne**.
On ignore s'il existe des nombres parfaits impairs (très probablement non). Le problème est ouvert ...
La formule d'**Euler** conduit au programme suivant :

```
global P
fd = mopen("PREMIERS",'rb') ;
P = mget(1e3) ;
mclose(fd) ;

Max = P($)^2 ;
i = 1 ;
while (2^P(i)-1) < Max
  F = facteurs(2^P(i)-1) ;
  premier = (size(F,1) == 1)&(F(1,2) == 1) ;
  if premier then
    Mersenne = 2^P(i) - 1 ;
    printf('\n Mersenne = %.0f',Mersenne)
    Parfait = 2^(P(i)-1)*(2^P(i)-1) ;
    printf(' ===> Parfait = %.0f',Parfait)
  end
  i = i + 1 ;
end
```

On obtient les 7 premiers nombres parfaits :

Mersenne = 3	===>	Parfait = 6
Mersenne = 7	===>	Parfait = 28
Mersenne = 31	===>	Parfait = 496
Mersenne = 127	===>	Parfait = 8 128
Mersenne = 8 191	===>	Parfait = 33 550 336
Mersenne = 131 071	===>	Parfait = 8 589 869 056
Mersenne = 524 287	===>	Parfait = 137 438 691 328

Remarque
Pour vérifier si un nombre *n* est parfait on peut aussi effectuer un test qui est la traduction de la définition :
D = diviseurs(n) ; Parfait = D($) == sum(D) - D($)

Application 4 : résolution d'équations diophantiennes

Les équations diophantiennes sont des équations où les coefficients et les solutions sont des nombres entiers.
Pour trouver les solutions d'une équation du type $x^2 - y^2 = n$, on utilise l'équivalence :

$$x^2 - y^2 = n \quad \Leftrightarrow \quad (x - y)(x + y) = n$$

En décomposant n en facteurs premiers, on obtient :

$$n = ab \qquad (a \text{ et } b \text{ de même parité}, a \le b)$$

Puis, on résoud le système linéaire :

$$\begin{cases} x - y = a \\ x + y = b \end{cases} \quad ==> \quad x = (a+b)/2 \ , \ y = (b-a)/2$$

Plus précisément, la décomposition du nombre n en facteurs premiers s'écrit :

$$n = 2^{\alpha 1} \, p_2^{\alpha 2} \, p_3^{\alpha 3} \ldots p_k^{\alpha k} \qquad \text{avec} \quad \alpha_1 \ge 0 \text{ et } \alpha_i \ge 1 \ (i = 2, \ldots , k)$$

Si $\alpha_1 = 1$, a et b sont de parité différente, par suite $(a+b)$ et $(b-a)$ ne sont pas divisibles par 2, et donc l'équation n'a pas de solution. Pour $\alpha_1 \neq 1$, on prend :

$$a = 2^{i1} \, p_2^{i2} \, p_3^{i3} \ldots p_k^{ik}$$

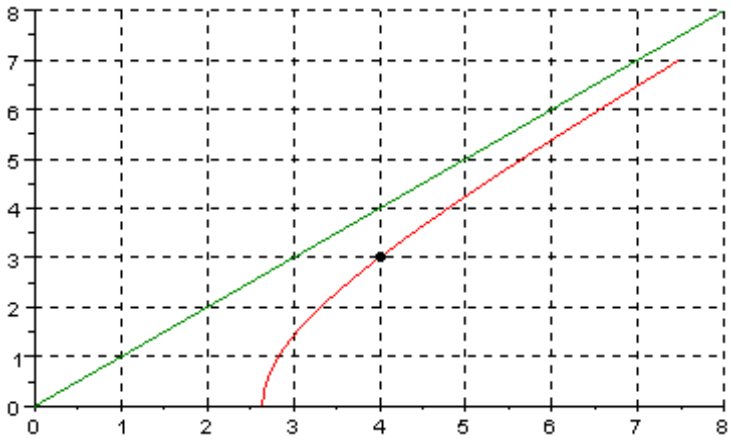
$$b = 2^{\alpha 1 - i1} \, p_2^{\alpha 2 - i2} \, p_3^{\alpha 3 - i3} \ldots p_k^{\alpha k - ik}$$

avec $1 \le i_1 \le \alpha_1 - 1$ si $\alpha_1 > 1$, ou $i_1 = 0$ si $\alpha_1 = 0$, et $0 \le i_2 \le \alpha_2 , \ldots , 0 \le i_k \le \alpha_k$

Tout couple $(x,y) = ((a+b)/2 , (b-a)/2)$ avec $a \le b$ est solution de l'équation diophantienne $x^2 - y^2 = n$

Remarque

Dans le cas général, l'équation en nombres entiers $x^2 - y^2 = n$ admet un nombre fini de solutions. Si n est impair, il existe au moins la solution triviale : $x = (n+1)/2 , y = (n-1)/2$. C'est d'ailleurs la seule solution lorsque n est premier supérieur à 2.
La résolution revient à trouver les points de coordonnées entières appartenant à l'hyperbole équilatère $x^2/n - y^2/n = 1$



Résolution géométrique de l'équation $x^2 - y^2 = 7$

L'hyperbole (en rouge) d'équation x /7 – y /7 = 1 rencontre un nœud de la grille au point (4,3).
On a : $4^2 - 3^2 = 7$

Programme Scilab

La fonction *Diophante* prend en paramètre d'entrée un nombre entier *n* et retourne le tableau S des solutions entières (x_i , y_i) de l'équation $x^2 - y^2 = n$

```
function S = Diophante(n)
  S = []
  if n == 0 then, S = [0,0], return, end
  if n == 1 then, S = [1,0], return, end
  F = facteurs(n)
  if F == [] then, return, end
  if F(1,:) == [2,1] then, return, end
  k = size(F,1)
  I = ones(k,1)
  loop = %t
  test = %t
  while loop
    if test then
      if F(1,1) == 2 then
        a = F(1,1)^I(1)
        b = F(1,1)^(F(1,2)-I(1))
      else
        a = F(1,1)^(I(1)-1)
        b = F(1,1)^(F(1,2)-I(1)+1)
      end
      for i = 2:k
        a = a*F(i,1)^(I(i)-1)
        b = b*F(i,1)^(F(i,2)-I(i)+1)
      end
      x = (a+b)/2
      y = (b-a)/2
      if y >= 0 then, S = [S ; [x,y]], end
      j = 0
    else
      I(j) = 1
    end
    j = j + 1
    I(j) = I(j) + 1
    if F(j,1) == 2 then
      test = I(j) <= (F(j,2)-1)
    else
      test = I(j) <= (F(j,2)+1)
    end
    loop = test|(j < k)
  end
  S = gsort(S,'r','i')
endfunction
```

Note

La fonction Scilab *gsort*(p_1 , p_2 , p_3) permet de trier un tableau T :

- Le paramètre p_1 est le nom du tableau (p_1 = T)
- Le paramètre p_2 indique si le tri s'effectue colonne par colonne (p_2 = 'r') ou ligne par ligne (p_2 = 'c')
- Le paramètre p_3 précise si le résultat est présenté dans l'ordre décroissant (p_3 = 'd') ou croissant (p_3 = 'i')

Dans le cas de la fonction *Diophante*, le tri par colonne de la matrice $S = [x_1,y_1 ; x_2,y_2 ; \dots ; x_k,y_k]$ est convenable car :

$x_i^2 - y_i^2 = n$, $x_j^2 - y_j^2 = n \quad \Rightarrow \quad (x_i - x_j)(x_i + x_j) = (y_i - y_j)(y_i + y_j) \quad \Rightarrow$ si $x_i < x_j$ alors $y_i < y_j$

```
//------ Chargement de la table des nombres premiers

global P
stacksize(2e7) ;
fd = mopen("PREMIERS",'rb') ;
P = mget(1e7) ;
mclose(fd) ;

//------ Programme principal

while %t
  n = input('n ?') ;
  if n == [] then, break, end
  S = Diophante(n) ;
  printf('\n Résolution de l''équation diophantienne x^2 - y^2 = %.0f',n)
  printf('\n --- %u solution(s) ---',size(S,1))
  for i = 1:size(S,1)
    printf('\n x%i = %.0f , y%i = %.0f',i,S(i,1),i,S(i,2))
  end
end
```

Exemples

Résolution de l'équation diophantienne x - y = 39057375178
--- 0 solution(s) ---

Résolution de l'équation diophantienne x - y = 34192452823
--- 1 solution(s) ---
x1 = 17096226412 , y1 = 17096226411

Résolution de l'équation diophantienne x - y = 2793563544114613
--- 2 solution(s) ---
x1 = 54514853 , y1 = 13353114
x2 = 1396781772057307 , y2 = 1396781772057306

Résolution de l'équation diophantienne x - y = 2534134864
--- 3 solution(s) ---
x1 = 158383433 , y1 = 158383425
x2 = 316766860 , y2 = 316766856
x3 = 633533717 , y3 = 633533715

Résolution de l'équation diophantienne x - y = 986710371763
--- 4 solution(s) ---
x1 = 1028882 , y1 = 268119
x2 = 487024382 , y2 = 487023369
x3 = 656931382 , y3 = 656930631
x4 = 493355185882 , y4 = 493355185881

Résolution de l'équation diophantienne x - y = 12382570729
--- 5 solution(s) ---
x1 = 111277 , y1 = 0
x2 = 149365 , y2 = 99636
x3 = 12407635 , y3 = 12407136
x4 = 27763723 , y4 = 27763500
x5 = 6191285365 , y5 = 6191285364

Mathématique & Informatique

L'équation diophantienne x - y = 759402000 admet 192 solutions entières :

x1 = 27563 , y1 = 563
x2 = 27580 , y2 = 1120
x3 = 27615 , y3 = 1785
x4 = 27723 , y4 = 3027
x5 = 27735 , y5 = 3135
x6 = 27748 , y6 = 3248
x7 = 27804 , y7 = 3696
x8 = 28220 , y8 = 6080
x9 = 28245 , y9 = 6195
x10 = 28581 , y10 = 7581
x11 = 28707 , y11 = 8043
x12 = 28740 , y12 = 8160
x13 = 28772 , y13 = 8272
x14 = 28945 , y14 = 8855
x15 = 29540 , y15 = 10640
x16 = 29761 , y16 = 11239
x17 = 29805 , y17 = 11355
x18 = 30345 , y18 = 12705
x19 = 30660 , y19 = 13440
x20 = 30715 , y20 = 13565
x21 = 31661 , y21 = 15589
x22 = 31983 , y22 = 16233
x23 = 32249 , y23 = 16751
x24 = 33105 , y24 = 18345
x25 = 33180 , y25 = 18480
x26 = 33635 , y26 = 19285
x27 = 34535 , y27 = 20815
x28 = 34876 , y28 = 21376
x29 = 35315 , y29 = 22085
x30 = 36435 , y30 = 23835
x31 = 36924 , y31 = 24576
x32 = 37020 , y32 = 24720
x33 = 37121 , y33 = 24871
x34 = 37527 , y34 = 25473
x35 = 38815 , y35 = 27335
x36 = 39835 , y36 = 28765
x37 = 41167 , y37 = 30583
x38 = 41412 , y38 = 30912
x39 = 41916 , y39 = 31584
x40 = 42045 , y40 = 31755
x41 = 42169 , y41 = 31919
x42 = 43645 , y42 = 33845
x43 = 44905 , y43 = 35455
x44 = 46689 , y44 = 37689
x45 = 47303 , y45 = 38447
x46 = 47460 , y46 = 38640
x47 = 48405 , y47 = 39795
x48 = 50241 , y48 = 42009
x49 = 50405 , y49 = 42205
x50 = 51268 , y50 = 43232
x51 = 54005 , y51 = 46445
x52 = 55140 , y52 = 47760
x53 = 55335 , y53 = 47985
x54 = 57743 , y54 = 50743
x55 = 58569 , y55 = 51681
x56 = 58780 , y56 = 51920
x57 = 59627 , y57 = 52877
x58 = 63420 , y58 = 57120
x59 = 64587 , y59 = 58413
x60 = 64815 , y60 = 58665
x61 = 67515 , y61 = 61635
x62 = 69020 , y62 = 63280
x63 = 73015 , y63 = 67615
x64 = 74396 , y64 = 69104
x65 = 74949 , y65 = 69699
x66 = 76083 , y66 = 70917
x67 = 79635 , y67 = 74715
x68 = 79940 , y68 = 75040
x69 = 86628 , y69 = 82128
x70 = 87964 , y70 = 83536
x71 = 88305 , y71 = 83895
x72 = 92505 , y72 = 88305
x73 = 94308 , y73 = 90192
x74 = 94660 , y74 = 90560
x75 = 96509 , y75 = 92491
x76 = 102340 , y76 = 98560
x77 = 104745 , y77 = 101055
x78 = 109389 , y78 = 105861
x79 = 110236 , y79 = 106736
x80 = 111972 , y80 = 108528
x81 = 112415 , y81 = 108985
x82 = 122115 , y82 = 118965
x83 = 128067 , y83 = 125067
x84 = 130101 , y84 = 127149
x85 = 130620 , y85 = 127680
x86 = 133735 , y86 = 130865
x87 = 139747 , y87 = 137003
x88 = 141980 , y88 = 139280
x89 = 144823 , y89 = 142177
x90 = 151935 , y90 = 149415
x91 = 155580 , y91 = 153120
x92 = 156205 , y92 = 153755
x93 = 166523 , y93 = 164227
x94 = 169881 , y94 = 167631
x95 = 172607 , y95 = 170393
x96 = 181860 , y96 = 179760
x97 = 185529 , y97 = 183471
x98 = 186245 , y98 = 184195
x99 = 194705 , y99 = 192745
x100 = 201845 , y100 = 199955
x101 = 211845 , y101 = 210045
x102 = 216132 , y102 = 214368
x103 = 217847 , y103 = 216097
x104 = 221361 , y104 = 219639
x105 = 232345 , y105 = 230705
x106 = 251881 , y106 = 250369
x107 = 253884 , y107 = 252384
x108 = 257988 , y108 = 256512
x109 = 259035 , y109 = 257565
x110 = 271915 , y110 = 270515
x111 = 277436 , y111 = 276064
x112 = 281935 , y112 = 280585
x113 = 301980 , y113 = 300720
x114 = 309315 , y114 = 308085
x115 = 323463 , y115 = 322287
x116 = 331324 , y116 = 330176
x117 = 352115 , y117 = 351035
x118 = 362145 , y118 = 361095
x119 = 380201 , y119 = 379201
x120 = 386367 , y120 = 385383
x121 = 387940 , y121 = 386960
x122 = 422340 , y122 = 421440
x123 = 430941 , y123 = 430059
x124 = 452445 , y124 = 451605
x125 = 463460 , y125 = 462640
x126 = 502628 , y126 = 501872
x127 = 506643 , y127 = 505893
x128 = 514869 , y128 = 514131
x129 = 542780 , y129 = 542080
x130 = 553843 , y130 = 553157
x131 = 603015 , y131 = 602385
x132 = 633135 , y132 = 632535
x133 = 646044 , y133 = 645456
x134 = 661787 , y134 = 661213
x135 = 703420 , y135 = 702880
x136 = 753627 , y136 = 753123
x137 = 759652 , y137 = 759152
x138 = 771996 , y138 = 771504
x139 = 775145 , y139 = 774655
x140 = 844005 , y140 = 843555
x141 = 904260 , y141 = 903840
x142 = 926305 , y142 = 925895
x143 = 968821 , y143 = 968429
x144 = 1004689 , y144 = 1004311
x145 = 1054905 , y145 = 1054545
x146 = 1085035 , y146 = 1084685
x147 = 1157789 , y147 = 1157461
x148 = 1265820 , y148 = 1265520
x149 = 1291647 , y149 = 1291353
x150 = 1356215 , y150 = 1355935
x151 = 1406435 , y151 = 1406165
x152 = 1506876 , y152 = 1506624
x153 = 1518929 , y153 = 1518679
x154 = 1543623 , y154 = 1543377
x155 = 1757983 , y155 = 1757767
x156 = 1808205 , y156 = 1807995
x157 = 1898605 , y157 = 1898405
x158 = 1937348 , y158 = 1937152
x159 = 2109540 , y159 = 2109360
x160 = 2260209 , y160 = 2260041
x161 = 2315332 , y161 = 2315168
x162 = 2531415 , y162 = 2531265
x163 = 2712220 , y163 = 2712080
x164 = 3013563 , y164 = 3013437
x165 = 3164235 , y165 = 3164115
x166 = 3515804 , y166 = 3515696
x167 = 3797060 , y167 = 3796960
x168 = 3874549 , y168 = 3874451
x169 = 4218945 , y169 = 4218855
x170 = 4520292 , y170 = 4520208
x171 = 4630541 , y171 = 4630459
x172 = 5273661 , y172 = 5273589
x173 = 5424335 , y173 = 5424265
x174 = 6328380 , y174 = 6328320
x175 = 6780403 , y175 = 6780347
x176 = 7031527 , y176 = 7031473
x177 = 7594045 , y177 = 7593995
x178 = 9040521 , y178 = 9040479
x179 = 9492545 , y179 = 9492505
x180 = 10547268 , y180 = 10547232
x181 = 12656715 , y181 = 12656685
x182 = 13560764 , y182 = 13560736
x183 = 15820887 , y183 = 15820863
x184 = 18985060 , y184 = 18985040
x185 = 21094509 , y185 = 21094491
x186 = 27121507 , y186 = 27121493
x187 = 31641756 , y187 = 31641744
x188 = 37970105 , y188 = 37970095
x189 = 47462629 , y189 = 47462621
x190 = 63283503 , y190 = 63283497
x191 = 94925252 , y191 = 94925248
x192 = 189850501 , y192 = 189850499

Mathématique & Informatique

L'ÉQUATION DE FERMAT

C'est une équation diophantienne du type : $x^2 - n y^2 = 1$ (n est un entier naturel non carré parfait).
Elle admet une infinité de solutions, qui peuvent toutes s'obtenir à partir de la plus petite d'entre elles (x_1 , y_1), appelée solution fondamentale. La récurrence suivante permet de calculer les solutions (x_i , y_i), $i > 1$:

$$\begin{cases} x_i = x_1 x_{i-1} + n y_1 y_{i-1} \\ y_i = y_1 x_{i-1} + x_1 y_{i-1} \end{cases}$$

Pour trouver la solution fondamentale, on fait appel au développement en fraction continue de $\sqrt{n}$

Développement en fraction continue d'un réel x

E(x) désigne la partie entière de x, c'est-à-dire le plus grand entier inférieur ou égal à x : $E(x) \le x < E(x) + 1$

$$x = a_0 + \frac{1}{x_1} \quad , a_0 = E(x)$$

$$x_1 = a_1 + \frac{1}{x_2} \quad , a_1 = E(x_1)$$

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{x_2}}$$

$$x_2 = a_2 + \frac{1}{x_3} \quad , a_2 = E(x_2)$$

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{x_3}}}$$

etc, ...

```
x = input('x = ?') ;
Max = 16 ; Eps = 1e-6 ;
x0 = x ; a = floor(x0) ; fc = list(a) ;
while ((x - a) > Eps) & (length(fc) < Max)
  x = 1/(x - a) ; a = floor(x) ; fc($+1) = a ;
end
printf('\n %.10f = [%.0f',x0,fc(1))
for i = 2:length(fc), printf(', %.0f',fc(i)), end
printf(']')
```

On obtient une suite de nombres entiers (a_k) et une suite de nombres réels (x_k).
Le développement en fraction continue de x est noté $[a_0 , a_1 , a_2 , ...]$, il est unique.
Si x est rationnel, on trouvera un $a_k = x_k$ et on arrêtera les calculs ; la fraction continue est limitée : $x = [a_i]_0^k$
Si x est irrationnel, la décomposition se poursuit indéfiniment ; la fraction continue est illimitée : $x = [a_i]_0^\infty$

Considérons la suite des nombres rationnels (p_k/q_k) :

$$\frac{p_0}{q_0} = a_0$$

$$\frac{p_1}{q_1} = a_0 + \frac{1}{a_1} = \frac{a_1 a_0 + 1}{a_1}$$

$$\frac{p_2}{q_2} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2}} = \frac{a_2 (a_1 a_0 + 1) + a_0}{a_2 a_1 + 1}$$

...

$$\frac{p_k}{q_k} = \frac{a_k p_{k-1} + p_{k-2}}{a_k q_{k-1} + q_{k-2}}$$

```
k = input('k = ?') ;
if k < length(fc) then
  pkm2 = 1 ; qkm2 = 0 ; pkm1 = fc(1) ; qkm1 = 1 ;
  if k == 0 then, pk = pkm1 ; qk = qkm1 ; end
  for i = 1:k
    pk = fc(i+1)*pkm1 + pkm2 ;
    qk = fc(i+1)*qkm1 + qkm2 ;
    pkm2 = pkm1 ; qkm2 = qkm1 ;
    pkm1 = pk ; qkm1 = qk ;
  end
  printf('\n p%i/q%i = %.0f/%.0f',k,k,pk,qk)
end
```

Les fractions ordinaires p_k/q_k sont irréductibles. Elles approximent x avec une précision croissante.
La fraction continue $[a_0 , a_1 , ... , a_k] = p_k/q_k$ est appelée réduite d'ordre k de $[a_0 , a_1 , ... , a_k , ...] = x$

Mathématique & Informatique

Nombre d'or : $\varphi = \frac{1+\sqrt{5}}{2} = 1+\frac{1}{1+\frac{1}{1+\cdots}} = [1,1,1,\cdots]$

Théorème de **Lagrange**

Soit r la racine carrée d'un nombre entier n non carré parfait : $r = \sqrt{n}$
Le développement en fraction continue de r est périodique ; il existe un entier T (la période) tel que :

$$\sqrt{n} = [a_0 , a_1 , \ldots , a_T , a_1 , \ldots , a_T , a_1 , \ldots , a_T , \ldots\ldots] = [a_0 , \overline{a_1 , \ldots , a_T}]$$

Conjecture : en moyenne, la période T est de l'ordre de $\sqrt{n}\ \mathrm{Log}\ \mathrm{Log}\ \sqrt{n}$

Résolution de l'équation de **Fermat**

On montre que : $a_T = 2\ a_0$, et que la solution fondamentale (x_1 , y_1) de l'équation de **Fermat** est (p_{T-1} , q_{T-1}) si T est paire ou (p_{2T-1} , q_{2T-1}) si T est impaire. Les autres solutions (x_i , y_i) se déduisent de (x_1 , y_1) en appliquant la formule du binôme de **Newton** à l'identité : $(x_1 + y_1 \sqrt{n})^i = x_i + y_i \sqrt{n}$

Programme Scilab

La fonction *Fermat* prend en paramètre d'entrée le nombre n et retourne la solution (x,y) de l'équation $x^2 - n\ y^2 = 1$

```
function [x,y] = Fermat(n)
  m = floor(sqrt(n))
  if m^2 == n then
    x = 1 ; y = 0
    return
  end
  fc = list()
  a = 0 ; b = 1 ; q = 0
  while q <> 2*m
    r = modulo(m+a,b)
    q = (m+a-r)/b
    a = m-r
    b = (n-a^2)/b
    fc($+1) = q
  end
  T = length(fc) - 1
  for k = 1:T-1
    fc($+1) = fc(k+1)
  end
  if modulo(T,2) == 0 then
    K = T-1
  else
    K = 2*T-1
  end
  x = 1 ; y = 0
  for k = K+1:-1:1
    z = x
    x = fc(k)*x + y
    y = z
  end
endfunction
```

```
n = 0 ; N = 0 ; Max = 100 ;
while N < Max
  n = n + 1 ;
  [x,y] = Fermat(n) ;
  if (x == 1) & (y == 0) then, continue, end
  N = N + 1 ;
  printf('\n n = %u , x = %.0f , y = %.0f',n,x,y)
end
```

Solutions fondamentales des 100 premières équations de Fermat $x^2 - ny^2 = 1$

n = 2 , x = 3 , y = 2
n = 3 , x = 2 , y = 1
n = 5 , x = 9 , y = 4
n = 6 , x = 5 , y = 2
n = 7 , x = 8 , y = 3
n = 8 , x = 3 , y = 1
n = 10 , x = 19 , y = 6
n = 11 , x = 10 , y = 3
n = 12 , x = 7 , y = 2
n = 13 , x = 649 , y = 180
n = 14 , x = 15 , y = 4
n = 15 , x = 4 , y = 1
n = 17 , x = 33 , y = 8
n = 18 , x = 17 , y = 4
n = 19 , x = 170 , y = 39
n = 20 , x = 9 , y = 2
n = 21 , x = 55 , y = 12
n = 22 , x = 197 , y = 42
n = 23 , x = 24 , y = 5
n = 24 , x = 5 , y = 1
n = 26 , x = 51 , y = 10
n = 27 , x = 26 , y = 5
n = 28 , x = 127 , y = 24
n = 29 , x = 9801 , y = 1820
n = 30 , x = 11 , y = 2
n = 31 , x = 1520 , y = 273
n = 32 , x = 17 , y = 3
n = 33 , x = 23 , y = 4
n = 34 , x = 35 , y = 6
n = 35 , x = 6 , y = 1
n = 37 , x = 73 , y = 12
n = 38 , x = 37 , y = 6
n = 39 , x = 25 , y = 4
n = 40 , x = 19 , y = 3
n = 41 , x = 2049 , y = 320
n = 42 , x = 13 , y = 2
n = 43 , x = 3482 , y = 531
n = 44 , x = 199 , y = 30
n = 45 , x = 161 , y = 24
n = 46 , x = 24335 , y = 3588
n = 47 , x = 48 , y = 7
n = 48 , x = 7 , y = 1
n = 50 , x = 99 , y = 14
n = 51 , x = 50 , y = 7
n = 52 , x = 649 , y = 90
n = 53 , x = 66249 , y = 9100
n = 54 , x = 485 , y = 66
n = 55 , x = 89 , y = 12
n = 56 , x = 15 , y = 2
n = 57 , x = 151 , y = 20
n = 58 , x = 19603 , y = 2574
n = 59 , x = 530 , y = 69
n = 60 , x = 31 , y = 4
n = 61 , x = 1766319049 , y = 226153980
n = 62 , x = 63 , y = 8
n = 63 , x = 8 , y = 1
n = 65 , x = 129 , y = 16
n = 66 , x = 65 , y = 8
n = 67 , x = 48842 , y = 5967
n = 68 , x = 33 , y = 4
n = 69 , x = 7775 , y = 936
n = 70 , x = 251 , y = 30
n = 71 , x = 3480 , y = 413
n = 72 , x = 17 , y = 2
n = 73 , x = 2281249 , y = 267000
n = 74 , x = 3699 , y = 430
n = 75 , x = 26 , y = 3
n = 76 , x = 57799 , y = 6630
n = 77 , x = 351 , y = 40
n = 78 , x = 53 , y = 6
n = 79 , x = 80 , y = 9
n = 80 , x = 9 , y = 1
n = 82 , x = 163 , y = 18
n = 83 , x = 82 , y = 9
n = 84 , x = 55 , y = 6
n = 85 , x = 285769 , y = 30996
n = 86 , x = 10405 , y = 1122
n = 87 , x = 28 , y = 3
n = 88 , x = 197 , y = 21
n = 89 , x = 500001 , y = 53000
n = 90 , x = 19 , y = 2
n = 91 , x = 1574 , y = 165
n = 92 , x = 1151 , y = 120
n = 93 , x = 12151 , y = 1260
n = 94 , x = 2143295 , y = 221064
n = 95 , x = 39 , y = 4
n = 96 , x = 49 , y = 5
n = 97 , x = 62809633 , y = 6377352
n = 98 , x = 99 , y = 10
n = 99 , x = 10 , y = 1
n = 101 , x = 201 , y = 20
n = 102 , x = 101 , y = 10
n = 103 , x = 227528 , y = 22419
n = 104 , x = 51 , y = 5
n = 105 , x = 41 , y = 4
n = 106 , x = 32080051 , y = 3115890
n = 107 , x = 962 , y = 93
n = 108 , x = 1351 , y = 130
n = 109 , x = 158070671986249 , y = 15140424455100
n = 110 , x = 21 , y = 2

A propos de l'équation $x^2 - n y^2 = -1$

Cette équation diophantienne est similaire à l'équation de **Fermat**, mais elle n'admet pas toujours de solution.

Soit T la période du développement en fraction continue de $\sqrt{n}$. Lorsque T est paire, l'équation $x - n y = -1$ n'a pas de solution et lorsque T est impaire, le couple (p_{T-1} , q_{T-1}) est solution.

On vérifie que si (a , b) est solution de $x - n y = -1$ alors $(a + n b , 2 a b)$ est solution de $x - n y = +1$.
Ce résultat peut être utile pour résoudre l'équation de **Fermat** avec de grandes valeurs de n.

Exemple :

$x - 797 y = -1 \quad \Rightarrow \quad x = 24\,715\,982$
$y = 875\,485$

$x - 797 y = +1 \quad \Rightarrow \quad x = (24\,715\,982) + 797\,(875\,485) = 1\,221\,759\,532\,448\,649$
$y = 2\,(24\,715\,982)(875\,485) = 43\,276\,943\,002\,540$

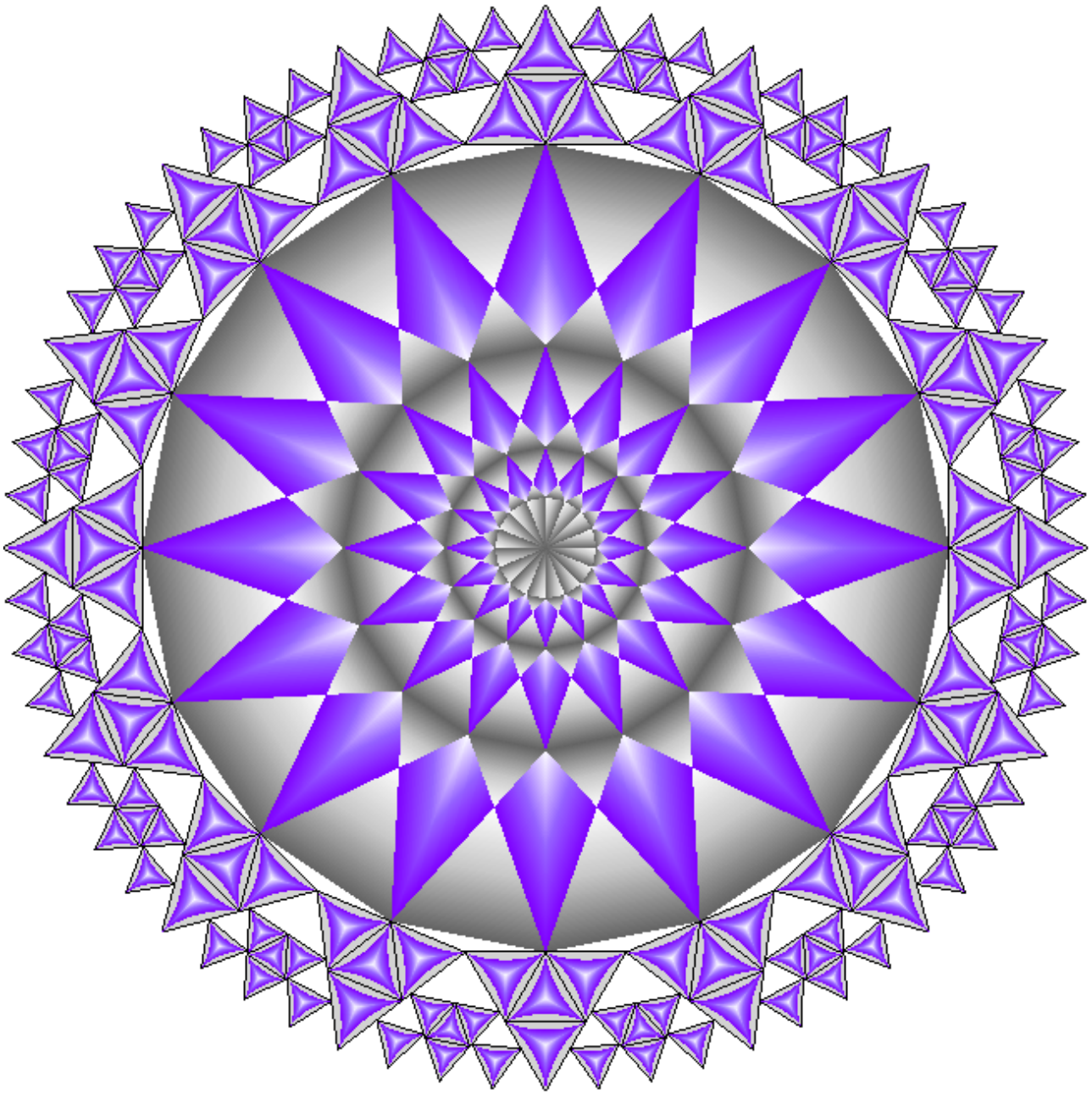
Programme :

Appelons *Fermat_negative* la fonction correspondant à la résolution de l'équation $x - n y = -1$

```
function [x,y] = Fermat_negative(n)
  m = floor(sqrt(n))
  if m^2 == n then
    if n == 1 then
      x = 0 ; y = 1
    else
      x = [] ; y = []
    end
    return
  end
  fc = list()
  a = 0 ; b = 1 ; q = 0
  while q <> 2*m
    r = modulo(m+a,b)
    q = (m+a-r)/b
    a = m-r
    b = (n-a^2)/b
    fc($+1) = q
  end
  T = length(fc) - 1
  if modulo(T,2) == 0 then
    x = [] ; y = []
    return
  end
  x = 1 ; y = 0
  for k = T:-1:1
    z = x
    x = fc(k)*x + y
    y = z
  end
endfunction
```

```
n = 797 ;
[x,y] = Fermat_negative (n) ;
if (x <> []) & (y <> []) then
  printf('\n x = %.0f , y = %.0f',x,y)
else
  printf('Pas de solution')
end
```

Mathématique & Informatique



Mathématique & Informatique

ARITHMÉTIQUE DES GRANDS NOMBRES

Un entier $x = x_1\ x_2\ ...\ x_n$ de n chiffres décimaux x_i peut être représenté par un vecteur $[x_1\ ,\ x_2\ ,\ ...\ ,\ x_n]$ de dimension n.
Ecrivons des fonctions Scilab qui permettent d'effectuer les opérations élémentaires *d'addition*, *de soustraction*, *de multiplication*, *de division*, avec des entiers naturels possédant un nombre quelconque de chiffres.

//--- ADDITION : c = a + b

```
function c = addition(a,b)
  p = length(a)
  q = length(b)
  n = max(p,q) + 1
  a = [zeros(1,n-p),a]
  b = [zeros(1,n-q),b]
  c = zeros(1,n)
  r = 0
  for j = n:-1:1
    c(j) = a(j) + b(j) + r
    if c(j) < 10 then
      r = 0
    else
      r = 1
      c(j) = c(j) - 10
    end
  end
  J = find(c <> 0)
  j = min([J,length(c)])
  c = c(j:$)
endfunction
```

//--- SOUSTRACTION : c = a - b

```
function c = soustraction(a,b)
  p = length(a)
  q = length(b)
  n = max(p,q)
  a = [zeros(1,n-p),a]
  b = [zeros(1,n-q),b]
  c = zeros(1,n)
  r = 0
  for j = n:-1:1
    c(j) = a(j) - b(j) - r
    if c(j) >= 0 then
      r = 0
    else
      r = 1
      c(j) = c(j) + 10
    end
  end
  J = find(c <> 0)
  j = min([J,length(c)])
  c = c(j:$)
endfunction
```

//--- COMPARAISON

```
function c = comparaison(a,b)
  p = length(a)
  q = length(b)
  n = max(p,q)
  a = [zeros(1,n-p),a]
  b = [zeros(1,n-q),b]
  c = a - b
  j = min(find(c <> 0))
  if j == [] then
    c = 0
  elseif c(j) > 0 then
    c = +1
  else
    c = -1
  end
endfunction
```

//--- CONVERSION_1

```
function b = entier(a)
  b = []
  while %t
    b = [modulo(a,10),b]
    a = floor(a/10)
    if a == 0 then, break, end
  end
endfunction
```

//--- CONVERSION_2

```
function b = nombre_naturel(a)
  b = 0
  for j = 1:length(a)
    b = 10*b + a(j)
  end
endfunction
```

Mathématique & Informatique

```
//---  MULTIPLICATION :  c = a * b

function c = multiplication(a,b)
  p = length(a)
  q = length(b)
  n = p + q
  a = [0,a]
  c = 0
  for i = 1:q
    x = zeros(1,n)
    r = 0
    for j = 0:p
      x(n-i-j+1) = a(p-j+1)*b(q-i+1) + r
      if x(n-i-j+1) < 10 then
        r = 0
      else
        r = floor(x(n-i-j+1)/10)
        x(n-i-j+1) = modulo(x(n-i-j+1),10)
      end
    end
    c = addition(c,x)
  end
endfunction
```

```
//---  PUISSANCE :  c = a ^ b

function c = puissance(a,b)
  c = 1
  while b <> 0
    if modulo(b,2) <> 0 then
      c = multiplication(c,a)
    end
    b = floor(b/2)
    if b <> 0 then
      a = multiplication(a,a)
    end
  end
endfunction
```

```
//---  ECRITURE

function ecrire(a)
  for j = 1:length(a)
    printf('%u',a(j))
  end
endfunction
```

```
//---  DIVISION :  a = b * c + d

function [c,d] = division(a,b)
  p = length(a)
  q = length(b)
  n = p - q + 1
  c = 0
  d = a
  if (q > p)|(b == 0) then
    return
  end
  x = [b,zeros(1,p-q)]
  for i = 1:n
    y = 0
    for k = 1:9
      z = multiplication(x,k)
      s = comparaison(d,z)
      if s < 0 then
        k = k - 1
        break
      else
        y = z
      end
    end
    d = soustraction(d,y)
    c = [c,k]
    if i <> n then
      x = x(1:$-1)
    end
  end
  J = find(c <> 0)
  j = min([J,length(c)])
  c = c(j:$)
endfunction
```

```
//---  PGCD :  c = pgcd(a,b)

function c = pgcd(a,b)
  c = a
  r = b
  while ~isequal(r,0)
    d = c
    c = r
    [q,r] = division(d,c)
  end
endfunction
```

« Diviser chacune des difficultés que j'examinerais, en autant de parcelles qu'il se pourrait, et qu'il serait requis pour les mieux résoudre. »
« Conduire par ordre mes pensées, en commençant par les objets les plus simples et les plus aisés à connaître, pour monter peu à peu comme par degrés jusques à la connaissance des plus composés, et supposant même de l'ordre entre ceux qui ne se précèdent point naturellement les uns les autres. »
René Descartes : Discours de la méthode (1637)

Mathématique & Informatique

EXEMPLE 1

Effectuer le produit : 1234567890123456789 × 9876543210987654321

Les instructions :

```
N = multiplication([1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9],[9,8,7,6,5,4,3,2,1,0,9,8,7,6,5,4,3,2,1]) ;
ecrire(N)
```

donnent :

12193263113702179522374638011112635269

EXEMPLE 2

Quel est le pgcd des 2 nombres :

A = 13807164647207120874924653602728663842646698433 43
B = 88769597772021188347832701081595801386544431 3

On écrit :

```
A = [1,3,8,0,7,1,6,4,6,4,7,2,0,7,1,2,0,8,7,4,9,2,4,6,5,3,6,0,2,7,2,8,6,6,3,8,4,2,6,4,6,6,9,8,4,3,3,4,3] ;
B = [8,8,7,6,9,5,9,7,7,7,2,0,2,1,1,8,8,3,4,7,8,3,2,7,0,1,0,8,1,5,9,5,8,0,1,3,8,6,5,4,4,4,3,1,3] ;
P = pgcd(A,B) ;
ecrire(P)
```

Le plus grand commun diviseur de A et B est :

239033083310602345005679

EXEMPLE 3

Calculer 100 !

Introduisons une nouvelle fonction :

```
function f = factorielle(n)
  f = 1
  for i = 1:n
    f = multiplication(f,entier(i))
  end
endfunction
```

Effectuons un appel avec $n = 100$:

```
f = factorielle(100) ;
ecrire(f)
```

Le résultat est un nombre entier à 158 chiffres :

93326215443944152681699238856266700490715968264381621
46859296389521759999322991560894146397615651828625369
79208272237582511852109168640000000000000000000000000

EXEMPLE 4

Calculer les coefficien ts binomiaux $C_n^k = \binom{n}{k}$ pour de grandes valeurs de n et de k :

$$C_n^k = \frac{n\,!}{k\,!\,(n-k)\,!} = \frac{n(n-1)\cdots(n-k+1)}{1\times 2\times\cdots\times k}$$

La fonction :

```
function C = binomiaux(n,k)
  A = 1 ; B = 1
  for i = 1:k
    A = multiplication(A,entier(n-i+1))
    B = multiplication(B,entier(i))
  end
  [C,r] = division(A,B)
endfunction
```

permet, par exemple, de calculer C_{1000}^{500} :

```
C = binomiaux(1000,500) ; ecrire(C)
```

On obtient un nombre entier à 300 chiffres :

270288240945436569515614693625975275496152008446548287007392875106625428705522193898612483924502370165362606085021546104802209750050679917549894219699518475423665484263751733356162464079737887344364574161119497604571044985756287880514600994219426752366915856603136862602484428109296905863799821216320

Remarque

Le nombre de combinaisons de k éléments parmi n se note C_n^k ou $\binom{n}{k}$. Attention toutefois à ne pas confondre la deuxième notation avec les coordonnées d'un vecteur colonne de dimension 2.

Par exemple, $C_5^2\binom{6}{3}$ est identique à $\binom{5}{2}\binom{6}{3}$ mais la seconde notation présente a priori une ambigüité : il peut s'agir du nombre $10 \times 20 = 200$, du vecteur $10 \times \binom{6}{3} = \binom{60}{30}$ ou du produit scalaire $\binom{5}{2} \times \binom{6}{3} = 36$.

Seul le contexte permet de connaître la signification du symbole $\binom{n}{k}$.

On choisit la notation C_n^k pour les combinaisons, A_n^k pour les arrangements et P_n pour les permutations.

EXEMPLE 5

Valeur exacte du 20e nombre premier de **Mersenne** : $M_{20} = 2^{4423} - 1$
Exprimons M_{20} avec les fonctions prédéfinies :

```
M20 = soustraction(puissance(entier(2),4423),entier(1)) ;
ecrire(M20)
```

M_{20} est un nombre entier à 1332 chiffres :

28554254222827961390156356610216400832616423864470
28891992474566022844003906006538759545715055398432
39754513915896150297878399377056071435169747221107
98879119820098847753133921428277201605900990458668
62549890848157354224804090223442975883525260043838
90632616124076317387416881148592486188361873904175
78314569601691957439076559828018859903557844859107
76836771755204340742877265780062667596159707595213
27828555662781678385691581844436444812511562428136
74249045936321281018027609608811140100337757036354
57251209240736469215767971461993876192965603026802
61790118132925012323046444438622308877924609373773
01248168167242449367447448853777015578300688085264
81615130671448147902883666640622572746652757871273
74649231096375001170901890786263324619578795731425
69380507305611967758033808433338198750090296883193
59130952698213111413223933564901784887289822881562
82600813831296143663845945431144043753821542871277
74560644785856415921332844358020642271469491309176
27164470416896780700967735904298089096167504529272
58000843500344831628297089902728649981994387647234
57427626372969484830475091717418618113068851879274
86226122933413689280566343844666463265724761672756
60839105650528975713899320211121495795311427946254
55330538706782106760176875097786610046001460213840
84480212250536890547937420030957220967329547507217
18115531871310231057902608580607

EXEMPLE 6

Donner l'altitude maximale du vol numéro 1 980 976 057 694 848 447 de la suite de Syracuse.

On recherche $a = Max\ s_n$ avec $\begin{cases} s_n = \dfrac{s_{n-1}}{2} \text{ si } s_{n-1} \text{ est pair, } 3s_{n-1} + 1 \text{ sinon} \\ s_0 = 1980976057694848447 \end{cases}$

Pour trouver la valeur exacte de a, on exécute le programme :

```
exec("Scilab-Arithmetique.sce",-1)
s0 = [1,9,8,0,9,7,6,0,5,7,6,9,4,8,4,8,4,4,7] ;
s = s0 ; a = s0 ;
while comparaison(s,1) > 0
    [q,r] = division(s,2) ;
    if nul(r) then
        s = q ;
    else
        s = addition(multiplication(s,3),1) ;
    end
    if comparaison(s,a) > 0 then, a = s ; end
end
ecrire(a)
```

L'altitude du vol 1 980 976 057 694 848 447 est 64 024 667 322 193 133 530 165 877 294 264 738 020
Il s'agit d'une altitude record, c'est-à-dire qu'aucun vol de numéro inférieur n'atteint cette altitude.

EXEMPLE 7

Soit le n-uplet $I_n = (1, 2, \dots, n)$ des n premiers indices.

Une permutation σ de I_n est le n-uplet $\sigma(I_n) = (\sigma(1), \sigma(2), \dots, \sigma(n))$. On note aussi : $\sigma = \begin{pmatrix} 1 & 2 & \cdots & n \\ \sigma(1) & \sigma(2) & \cdots & \sigma(n) \end{pmatrix}$

On dit que σ présente une coïncidence en i si $\sigma(i) = i$.

Une transposition est une permutation σ qui présente $n - 2$ coïncidences : $\sigma(i) = j$, $\sigma(j) = i$, $\sigma(k) = k$, $k \neq i, j$

Une permutation σ est complète lorsqu'elle ne présente aucune coïncidence, c'est-à-dire si $\sigma(i) \neq i$, $\forall\ i$.

Par exemple : $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix}$ est une permutation qui présente une coïncidence en $i = 2$; $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix}$ est une transposition qui échange les deux premiers indices $i = 1$ et $j = 2$; $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}$ est une permutation complète car tous les indices i ont changé de place.

Désignons par $\xi_{n,k}$ le nombre des permutations σ qui présentent exactement k coïncidences. On montre que :

$$\xi_{n,k} = \frac{n!}{k!} \sum_{p=0}^{n-k} \frac{(-1)^p}{p!} \qquad (k = 0, 1, 2, \dots, n)$$

En particulier, la permutation identité est la seule permutation à présenter n coïncidences ($\xi_{n,n} = 1$), il n'y a pas de permutation présentant $n - 1$ coïncidences ($\xi_{n,n-1} = 0$), il existe $n(n-1)/2$ transpositions différentes ($\xi_{n,n-2}$), le nombre des permutations complètes ($\xi_{n,0}$) est donné par :

$$\xi_{n,0} = n! \sum_{p=0}^{n} \frac{(-1)^p}{p!} \underset{n\infty}{\sim} \frac{n!}{e}$$

Ce problème a été étudié par **Pierre Rémond de Montmort**.

⇨ **http://www.link.cs.cmu.edu/15859-s11/notes/Dickerson_the_problem_of_coincidences.pdf**

Application numérique : combien de permutations de (1, 2, 3, … , 52) présentent 10 coïncidences ?

On peut utiliser directement la formule ou remarquer que $\xi_{n,k}$ se déduit de $\xi_{n-k,0}$ obtenu par récurrence :

$$\xi_{n,k} = C_n^k \xi_{n-k,0} \quad \text{avec} \quad \begin{cases} \xi_{0,0} = 1 \\ \xi_{n,0} = n\,\xi_{n-1,0} + (-1)^n \end{cases}$$

D'où le programme :

```
n = 52 ; k = 10 ;
xi = 1 ;
for i = 1:n-k
    if modulo(i,2) == 0 then
        xi = addition(multiplication(xi,entier(i)),1) ;
    else
        xi = soustraction(multiplication(xi,entier(i)),1) ;
    end
end
xi = multiplication(binomiaux(n,k),xi) ;
printf('\n xi(%i,%i) = ',n,k), ecrire(xi)
```

On obtient :

$$\xi_{52,10} = 817694124994354560688917669853657171166173703756390403694742 0$$

EXEMPLE 8

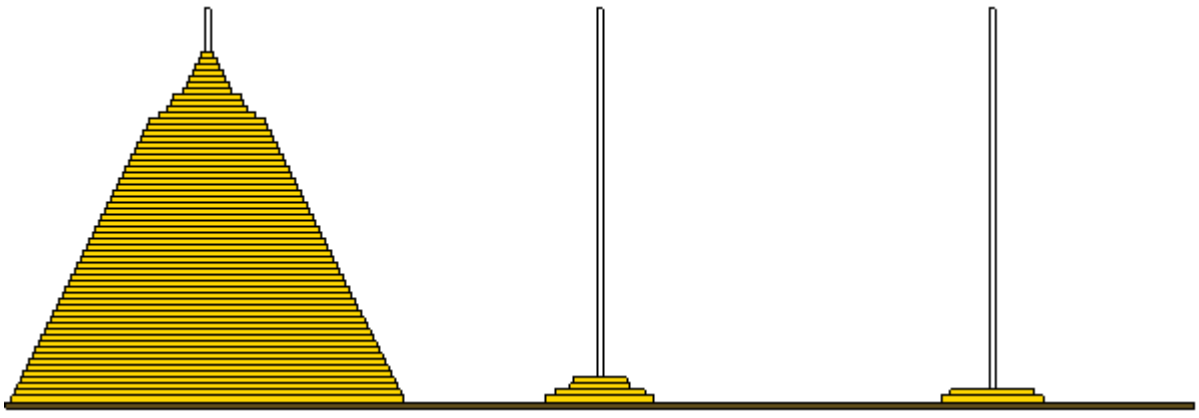
Considérons le problème de la Tour d'Hanoï de 64 disques (**Lucas**).
Initialement, la Tour est positionnée sur le piquet 1. En supposant qu'un déplacement de disque peut être effectué en une seconde, quelle est la configuration de l'ensemble des disques sur les 3 piquets au bout d'une journée ?

Solution (voir l'étude précédente du problème)

On doit réaliser $d = 60 \times 60 \times 24 = 86400$ déplacements de disques à partir de l'état initial de la Tour de $n = 64$ disques. Avec la fonction *hanoi* récursive, l'arbre des appels a une profondeur égale à 64 et possède $2^{64} = 18446744073709551616$ feuilles. Avec la fonction *hanoi* itérative, la pile des appels est remplacée par une variable *k* qui prend des valeurs entières comprises entre 1 et 2^{65}. Ecrivons une fonction *HANOI* itérative permettant d'utiliser des grands entiers :

```
function HANOI(a,b,c,n,d)
    N = puissance(2,n)
    d = entier(d)
    k = 2
    x = 0
    while comparaison(k,1) == 1
        k = soustraction(k,1)
        while comparaison(k,N) == -1
            [b,c] = echanger(b,c)
            k = multiplication(k,2)
        end
        while comparaison(multiplication(k,parite(k)),1) == 1
            [a,b] = echanger(a,b)
            [k,r] = division(k,2)
        end
        if comparaison(k,1) == 1 then
            [b,c] = echanger(b,c)
            deplacer(a,c)
            x = addition(x,1)
            if comparaison(x,d) == 0 then, return, end
            [a,b] = echanger(a,b)
            k = addition(k,2)
        end
    end
endfunction
```

Résultat de l'appel *HANOI*(1,2,3,64,86400) :



Mathématique & Informatique

Exponentiation modulaire

Calculer le reste de la division euclidienne de :

$$10587930195352580^{4}{}^{228789254670245678631229897978374572587058912054 03}$$

par : 7975125892537893690154823100678721045689

Le problème général consiste à calculer le reste r de la division euclidienne de a^n par p : $a^n \equiv r \ (p)$
Reprenons le procédé de calcul de a^n en ne gardant dans les résultats intermédiaires que les restes modulo p.
On obtient la fonction suivante :

```
function r = reste(a,n,p)
    r = 1
    while ~nul(n)
        [n,m] = division(n,2)
        if ~nul(m) then
            [q,r] = division(multiplication(a,r),p)
        end
        if ~nul(n) then
            [q,a] = division(multiplication(a,a),p)
        end
    end
endfunction
```

Pour les valeurs ci-dessus, on écrit :

```
r = reste([1,0,5,8,7,9,3,0,1,9,5,3,5,2,5,8,0,4], ...
          [2,2,8,7,8,9,2,5,4,6,7,0,2,4,5,6,7,8,6,3,1,2,2,9,8,9,7,9,7,8,3,7,4,5,7,2,5,8,7,0,5,8,9,1,2,0,5,4,0,3], ...
          [7,9,7,5,1,2,5,8,9,2,5,3,7,8,9,3,6,9,0,1,5,4,8,2,3,1,0,0,6,7,8,7,2,1,0,4,5,6,8,9]) ;
ecrire(r)
```

et on trouve : 472883147782850262056639251105399797915 9

Critère d'Euler

La fonction *reste* permet de mettre en œuvre le critère d'**Euler** pour un grand nombre premier p :

$$\forall a : \quad a^{\frac{p-1}{2}} \equiv \left(\frac{a}{p}\right) \quad \text{modulo } p$$

Le symbole de **Legendre** $\left(\frac{a}{p}\right)$ vaut 0 si a est divisible par p, +1 si a est résidu quadratique modulo p
(c'est à dire si a est le reste de la division d'un carré par p : $\exists\, x, x^2 \equiv a \ (p)$), ou -1 si a n'est ni l'un ni l'autre.

Application 1 : Test de primalité de **Solovay - Strassen**

Pour rechercher si un nombre impair n est premier, on prend au hasard plusieurs nombres a ($0 < a < n$) et on compare le reste de la division de $a^{\frac{n-1}{2}}$ par n à 1 et à n - 1. En cas d'égalité, n est probablement premier, sinon n est composé.

Application 2 : Résidu quadratique

On peut facilement calculer $\left(\frac{a}{p}\right)$ en utilisant la congruence avec $a^{\frac{p-1}{2}}$ modulo p

EXEMPLE 1 : Grand nombre premier

Recherchons un grand nombre premier en prenant au hasard un nombre impair n et en lui appliquant le test probabiliste de primalité basé sur le critère d'**Euler**.

Rappelons que la fonction Zêta de **Riemann**, $\zeta(z) = \sum_{n=1}^{+\infty} \frac{1}{n^z}$, lie l'ensemble des nombres entiers ($n = 1, 2, 3, 4, 5, 6, ..$)

à l'ensemble des nombres premiers ($p = 2, 3, 5, 7, 11, 13, ..$) par la relation :

$$\zeta(z) = \frac{1}{1^z} + \frac{1}{2^z} + \frac{1}{3^z} + \frac{1}{4^z} + \ \ldots\ + \frac{1}{n^z} + \ \ldots\ = \frac{1}{\left(1 - \frac{1}{2^z}\right)\left(1 - \frac{1}{3^z}\right)\left(1 - \frac{1}{5^z}\right)\left(1 - \frac{1}{7^z}\right) \ \ldots \left(1 - \frac{1}{p^z}\right) \ \ldots}$$

L'écriture est légitime pour tout complexe z dont la partie réelle est supérieure à 1 : $z = x + iy,\ x > 1$ (demi-plan de **Cauchy**)

A partir de cette identité, **Hadamard** prouva que le nombre de nombres premiers inférieurs à n est asymptotiquement $\frac{n}{Log(n)}$

Ainsi, pour un grand entier n de k chiffres choisis de manière aléatoire, il y a 10^k nombres entiers inférieurs à n parmi lesquels on trouve approximativement $\frac{10^k}{Log(10^k)}$ nombres premiers ; on a donc 1 chance sur $Log(10^k)$ que n soit premier.

Par exemple, si n possède une vingtaine de chiffres pris au hasard, on a environ 1 chance sur 50 d'obtenir un nombre premier par cette méthode.

Pour générer le nombre n, utilisons la fonction Scilab *grand* qui fournit des chiffres aléatoires :

```
exec("Scilab-Arithmetique.sce",-1)
for i = 1:50
    x = [grand(1,1,'uin',1,9),grand(1,20,'uin',0,9)] ;
    m = multiplication(x,2) ;
    n = addition(m,1) ;
    printf('\n n = '), ecrire(n)
    for a = [2,3,5,7]
        r = reste(a,x,n) ;
        premier = comparaison(r,1) == 0 | comparaison(r,m) == 0 ;
        if ~premier then, printf(' NOK'), break, end
    end
    if premier then, printf(' OK'), break, end
end
```

Résultat d'une exécution :

Le nombre 582935821004027256343 est très vraisemblablement premier.

NB

Plus les nombres premiers ont de chiffres, plus ils sont rares.
Il y a une infinité de nombres premiers, mais la probabilité d'en trouver un à l'infini est nulle…

EXEMPLE 2 : Test de primalité de **Lucas - Lehmer**

Un nombre premier de **Mersenne** est un nombre premier de la forme $M_p = 2^p - 1$.

- Si $2^n - 1$ est premier, alors n est nécessairement premier. En effet, dans le cas contraire, $n = ab$ d'où $2^{ab} - 1$ est divisible par $2^a - 1 \neq 1$ et par $2^b - 1 \neq 1$, donc $2^n - 1$ n'est pas premier (contradiction).
- Si n est premier, alors $2^n - 1$ n'est pas nécessairement premier. Par exemple : $2^{11} - 1 = 2047 = 23 \times 89$.

Pour un nombre premier impair p donné ($p > 2$), le test de primalité de **Lucas – Lehmer** permet de vérifier si l'entier $M_p = 2^p - 1$ est effectivement premier. Soit la suite récurrente :

$$\begin{cases} S_n = (S_{n-1})^2 - 2 \quad , n > 2 \\ S_2 = 4 \end{cases}$$

On a l'équivalence : S_p est divisible par $M_p \Leftrightarrow M_p$ est premier

Par exemple, pour le nombre premier $p = 13$, on calcule :

S_{13} = 2231399586789790076960379634 22 … 485914888669136966553469714434 (1172 chiffres)

M_{13} = 8191

On vérifie que S_{13} est divisible par M_{13} et on en déduit que M_{13} est premier.

Il n'est guère envisageable de calculer S_n pour de grandes valeurs de n, on fait donc intervenir le reste r_n de la division euclidienne de S_n par M_p :

$$\begin{cases} r_n \equiv (r_{n-1})^2 - 2 \quad , n > 2 \\ r_2 \equiv 4 \end{cases}$$

Si r_p est nul, alors $2^p - 1$ est premier. D'où le programme pour rechercher les nombres premiers de **Mersenne** :

```
while %t
    p = input('Nombre premier impair = ? ') ;
    if p == [] then, break, end
    Mp = soustraction(puissance(2,p),1) ;
    r = 4 ;
    for n = 3:p
        r = reste(r,2,Mp) ;
        r = soustraction(r,2) ;
    end
    if nul(r) then
        printf('\n ===> M%i = ',p), ecrire(Mp), printf(' est premier')
    else
        printf('\n ===> M%i = ',p), ecrire(Mp), printf(' n''est pas premier')
    end
end
```

Par exemple, pour $p = 127$, on vérifie que M_{127} = 170141183460469231731687303715884105727 est un nombre premier. A ce propos, lire *Recherches sur plusieurs ouvrages de Léonard de Pise et sur diverses questions d'arithmétique supérieure* par **Édouard Lucas** (page 26 et suivantes) via le portail numérique Gallica de la Bibliothèque nationale de France :
⇨ **http://gallica.bnf.fr/ark:/12148/bpt6k99634z.r=Edouard+Lucas.langFR**

Le projet *Great Internet Mersenne Prime Search* (GIMPS) a pour but de rechercher les plus grands nombres premiers de **Mersenne**. Au 25 janvier 2013, $M_{57\,885\,161}$ est le plus grand nombre premier de **Mersenne** connu (17 425 170 chiffres).
⇨ **http://mersenne.org/**

EXEMPLE 3 : Résidu quadratique

Existe t-il un couple (X,Y) d'entiers premiers entre eux tels que :

$X^2 + 2710201119071959\ Y^2$ soit divisible par $58293582100402725634 3$

Il s'agit de rechercher le caractère quadratique de -2710201119071959 modulo $58293582100402725634 3$

Appliquons le critère d'Euler pour exprimer $\left(\frac{-2710201119071959}{58293582100402725634 3}\right)$ comme la puissance de 2 nombres :

$$\left(\frac{-2710201119071959}{58293582100402725634 3}\right) \equiv (-2710201119071959)^{\frac{58293582100402725634 3 - 1}{2}}$$

$$= - 2710201119071959^{29146791050201362817 1}$$

Utilisons la fonction *reste* pour calculer $2710201119071959^{29146791050201362817 1}$ modulo $58293582100402725634 3$

```
r = reste([2,7,1,0,2,0,1,1,1,9,0,7,1,9,5,9], ...
          [2,9,1,4,6,7,9,1,0,5,0,2,0,1,3,6,2,8,1,7,1], ...
          [5,8,2,9,3,5,8,2,1,0,0,4,0,2,7,2,5,6,3,4,3]) ;
ecrire(r)
```

On obtient :

$r = 58293582100402725634 2$

D'où :

$$r \equiv -1 \quad (58293582100402725634 3) \qquad \Rightarrow \qquad \left(\frac{-2710201119071959}{58293582100402725634 3}\right) = 1$$

Conclusion :

Le nombre -2710201119071959 est résidu quadratique modulo $58293582100402725634 3$

La forme $X^2 + 2710201119071959\ Y^2$ est divisible par $58293582100402725634 3$

Donc, on est sûr qu'il existe un couple (X,Y) d'entiers tels que $X^2 + 2710201119071959\ Y^2$ soit divisible par $58293582100402725634 3$

Symbole de Jacobi

Le symbole de **Jacobi** $\left(\frac{m}{n}\right)$ généralise la notation de **Legendre** $\left(\frac{a}{p}\right)$ au cas où le module p n'est pas premier.

On a la décomposition :

$$\left(\frac{m}{n}\right) = \left(\frac{m_1\, m_2\, \ldots\, m_k}{n_1\, n_2\, \ldots\, n_l}\right) = \prod_{\substack{i=1,\cdots,k \\ j=1,\cdots,l}} \left(\frac{m_i}{n_j}\right)$$

Pour calculer $\left(\frac{m}{n}\right)$, m entier quelconque et n entier impair différent de 1, sans factoriser m et n, on utilise les formules suivantes :

$m = -1$: $\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}}$

$m = 2$: $\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}}$

m *impair* : $\left(\frac{m}{n}\right) = (-1)^{\frac{(m-1)(n-1)}{4}} \left(\frac{n}{m}\right)$ (Loi de réciprocité quadratique)

Soit $s = \mathrm{sgn}\left(\frac{m}{n}\right)$ le signe de $\frac{m}{n}$, la fonction Scilab notée *Jacobi(s,m,n)* calcule le rapport symbolique $\left(s\left|\frac{m}{n}\right|\right) = \left(\frac{m}{n}\right)$:

```
function J = Jacobi(s,m,n)
    J = 1
    if s < 0 then
        [q,r] = division(soustraction(n,1),entier(4))
        if ~nul(r) then, J = - J, end
    end
    if comparaison(m,n) >= 0 then
        [q,r] = division(m,n)
        m = r
    end
    while comparaison(m,2) >= 0
        [q,r] = division(m,2)
        if nul(r) then
            m = q
            [q,r] = division(soustraction(multiplication(n,n),1),entier(16))
            if ~nul(r) then, J = - J, end
        else
            [q,r] = division(multiplication(soustraction(m,1),soustraction(n,1)),entier(8))
            if ~nul(r) then, J = - J, end
            [q,r] = division(n,m)
            n = m
            m = r
        end
    end
    if nul(m) then, J = 0, end
endfunction
```

En sortie, la variable J contient la valeur du symbole de **Jacobi** de m modulo n : $J = \left(\frac{m}{n}\right) \in \{0,\ -1,\ +1\}$

- Si J = 0, alors m et n ne sont pas premiers entre eux
- Si J = -1, alors m n'est pas résidu quadratique modulo n
- Si J = +1, alors m est résidu quadratique modulo n si n est premier, mais pas nécessairement si n est composé

Par exemple, calculons :

$$\left(\frac{8976356428\ 0125478930\ 8900205087\ 013204}{7658995890\ 2145630132\ 058903}\right)$$

Pour cela, appelons la fonction *Jacobi* :

```
J = Jacobi(1,[8,9,7,6,3,5,6,4,2,8,0,1,2,5,4,7,8,9,3,0,8,9,0,0,2,0,5,0,8,7,0,1,3,2,0,4],[7,6,5,8,9,9,5,8,9,0,2,1,4,5,6,3,0,1,3,2,0,5,8,9,0,3])
printf('\n J = %i',J)
```

Résultat :

J = - 1

Donc, 8976356428012547893089002050870132 04 n'est pas résidu quadratique modulo 765899589021456301320589 03
Ainsi, on peut affirmer que dans la suite de nombres :

$$8976356428012547893089002050870132 04 \ + \ 765899589021456301320589 03\ k$$

il n'y a aucun carré.

La fonction *Jacobi* permet a fortiori de calculer le symbole de **Legendre**.
Dans l'exemple 2 précédent, on recherche :

$$\left(-\frac{2710201119071959}{582935821004027256343}\right)$$

On écrit :

```
J = Jacobi(-1,[2,7,1,0,2,0,1,1,1,9,0,7,1,9,5,9],[5,8,2,9,3,5,8,2,1,0,0,4,0,2,7,2,5,6,3,4,3]) ;
printf('\n J = %i',J)
```

Et on trouve :

J = 1

Puisque 582935821004027256343 est premier, le nombre - 2710201119071959 est résidu quadratique modulo 582935821004027256343 et on peut affirmer que dans la suite de nombres :

$$-\ 2710201119071959 \ + \ 582935821004027256343\ k$$

il y a au moins un carré.

Application au test de primalité

Le calcul du symbole de **Jacobi** repose sur la loi de réciprocité quadratique (découverte par **Legendre** et démontrée par **Gauss**).

Il intervient lors de la recherche d'un grand nombre premier p avec le critère d'**Euler** : $a^{\frac{p-1}{2}} \equiv \left(\frac{a}{p}\right)$ modulo p

La valeur de $a^{\frac{p-1}{2}}$ modulo p est obtenue avec la fonction *reste* et la valeur de $\left(\frac{a}{p}\right)$ est obtenue avec la fonction *Jacobi*.

Si les 2 valeurs sont égales pour plusieurs nombres a pris au hasard, alors il est très probable que p soit premier.

Algorithme d'Euclide étendu — Théorème de Bachet-Bézout

Soit 2 entiers a et b. Il existe un couple d'entiers relatifs (u,v) tel que : a u + b v = pgcd(a,b)
La fonction *Euclide* calcule à la fois u, v, et p = pgcd(a,b).

```
function [u,v,p] = Euclide(a,b)
  r0 = a ; r1 = b
  u0 = 1 ; u1 = 0
  v0 = 0 ; v1 = 1
  while r1 <> 0
    q = floor(r0/r1)
    r2 = r0 - q*r1
    u2 = u0 - q*u1
    v2 = v0 - q*v1
    r0 = r1 ; r1 = r2
    u0 = u1 ; u1 = u2
    v0 = v1 ; v1 = v2
  end
  u = u0 ; v = v0 ; p = r0
endfunction
```

On en déduit la résolution des équations diophantiennes du type **a x + b y = c**
La fonction *Bachet* fournit une solution (x,y).

```
function [x,y] = Bachet(a,b,c)
  [u,v,p] = Euclide(a,b)
  if modulo(c,p) == 0 then
    x = u*c/p
    y = v*c/p
  else
    x = []
    y = []
  end
endfunction
```

Lorsque l'équation **a x + b y = c** admet une solution (x_0 , y_0), elle en admet une infinité d'autres, données par :
(x_0 + k*b/p , y_0 - k*a/p) , où p est le pgcd de a et b, et k est un entier relatif quelconque.

Exemple

Résoudre en nombres entiers : 73 374 673 x + 37 291 409 y = 10 309 504

On écrit sous Scilab : [x,y] = Bachet(73374673,37291409,10309504)

Résultat : x = 27 648 , y = - 54 400

Remarque

La résolution géométrique de l'équation diophantienne **a x + b y = c** revient à rechercher les points de coordonnées entières appartenant à la droite d'équation **a x + b y = c**.

EXEMPLE

La moitié des nombres premiers sont de la forme 4k + 1. Soit p un nombre premier de ce type.
Recherchons un grand nombre entier x tel que :

$$x + 1 \equiv 0 \quad (p)$$

Application : p = 2017

Solution

D'après le théorème des 2 carrés de **Fermat**, il existe un unique couple (a,b) tel que : $a + b = p$
Si p n'est pas trop grand, il est facile de trouver a et b en testant tous les entiers dans $[1,E(\sqrt{p})]$
Multiplions la relation obtenue par un nombre q premier avec p : $(qa) + (qb) = q\,p$
L'identité de **Bachet-Bézout** pour qb et p s'écrit : $\exists\ u, v\ /\ u(qb) + vp = 1$
D'où : $(qa) + [(1 - vp)/u] = q\,p \Rightarrow (uqa) + 1 = (q\,u + 2v - v\,p)p = kp$
Posons x = |uqa|, on obtient : $x + 1 \equiv 0 \quad (p)$

```
p = 2017 ; q = 104723 ;
n = ceil(sqrt(p)) ;
for a = 1:n
    for b = 1:n
        trouve = a^2 + b^2 == p ;
        if trouve then, disp(a), disp(b), break, end
    end
    if trouve then, break, end
end
[u,v] = Bachet(q*b,p,1) ;
x = abs(u*q*a) ;
printf('\n x = %.0f',x)
```

Ainsi, pour p = 2017 et q = 104723, on trouve x = 465598458 ($9 + 44 = 2017 \Rightarrow 465598458 + 1 \equiv 0\ (2017)$)

Remarquons que l'on peut calculer un très grand entier x par une autre méthode.
Pour tout nombre a non multiple de p, le petit théorème de **Fermat** donne la congruence : $a^{p-1} - 1 \equiv 0 \quad (p)$
Posons p = 4k + 1, on a : $a^{4k} - 1 \equiv 0 \quad (p)$
D'où : $(a^{2k} - 1)(a^{2k} + 1) \equiv 0 \quad (p)$
Supposons que $a^{2k} - 1$ ne soit pas divisible par p, alors : $a^{2k} + 1 \equiv 0 \quad (p)$
Prenons $x = a^k = a^{(p-1)/4}$, il vient : $x + 1 \equiv 0 \quad (p)$
Ecrivons le programme de calcul de x dans le cas p = 2017 :

```
exec('Scilab-Arithmetique.sce',-1)
p = 2017 ;
a = 1 ; r = 0 ;
while nul(r)
    a = a + 1 ; disp(a)
    x = puissance(entier(a),(p-1)/4) ;
    n = soustraction(multiplication(x,x),1) ;
    [q,r] = division(n,entier(p)) ;
end
printf('\n x = '), ecrire(x)
```

On obtient la valeur de x suivante :

19093352271872529262824871207585110623683767128577041649335086933008072665779502860636607719425095176605469704334940419720736583923776774810844581963755736810192027628336080326387722430122074273130103179174563621799460396187106578227281118266888250220281763070119296551263247509341482569333769249094229892728239762755837460872498922981321811676025390625

Algorithme d'Euclide étendu — Grands nombres

Pour tenir compte du signe +/- des entiers relatifs, on introduit le vecteur $[s, x_1, x_2, ..., x_n]$ où $s \in \{-1, 0, +1\}$ définit le signe de l'entier $x_1\ x_2\ ...\ x_n$. Ainsi, un entier relatif x non nul est représenté par $[\pm 1, |x|]$.
Par convention, 0 n'est ni positif ni négatif ; on le note [0,0].
Exemple : 12345 = [1, 1, 2, 3, 4, 5] ; -12345 = [-1, 1, 2, 3, 4, 5]
Ecrivons de nouvelles fonctions Scilab qui permettent de calculer dans $\mathbb{Z}$.

```
//---  CONVERSION_1

function b = ENTIER(a)
  if a > 0 then, sb = +1, end
  if a < 0 then, sb = -1, end
  if a == 0 then, sb = 0, end
  vb = entier(abs(a))
  b = [sb,vb]
endfunction

//---  CONVERSION_2

function b = nombre_relatif(a)
  sa = a(1) ; va = a(2:$)
  b = entier_naturel(va)
  if sa < 0 then, b = - b, end
endfunction

//---  COMPARAISON

function c = COMPARAISON(a,b)
  sa = a(1) ; va = a(2:$)
  sb = b(1) ; vb = b(2:$)
  if sa > sb then, c = +1, end
  if sa < sb then, c = -1, end
  if sa == sb then
    s = comparaison(va,vb)
    c = sa*s
  end
endfunction

//---  TEST :  a = 0

function t = NUL(a)
  if length(a) == 0 then, t = %f, return, end
  t = COMPARAISON(a,[0,0]) == 0
endfunction

//---  SIGNE

function b = SIGNE(a)
  b = a(1)
endfunction

//---  VALEUR

function b = VALEUR(a)
  b = a(2:$)
endfunction

//---  MOINS :  b = - a

function b = MOINS(a)
  sa = a(1) ; va = a(2:$)
  b = [-sa,va]
endfunction

//---  ABSOLU :  b = |a|

function b = ABSOLU(a)
  sa = a(1) ; va = a(2:$)
  b = [abs(sa),va]
endfunction

//---  PARITE

function b = parite(a)
  b = modulo(a($),2)
endfunction
```

Mathématique & Informatique

//--- ADDITION : c = a + b

```
function c = ADDITION(a,b)
  sa = a(1) ; va = a(2:$)
  sb = b(1) ; vb = b(2:$)
  if sa*sb > 0 then
    sc = sa
    vc = addition(va,vb)
  end
  if sa*sb < 0 then
    s = comparaison(va,vb)
    if s > 0 then
      sc = sa
      vc = soustraction(va,vb)
    elseif s < 0 then
      sc = - sa
      vc = soustraction(vb,va)
    else
      sc = 0
      vc = 0
    end
  end
  if sa*sb == 0 then
    if sa == 0 then
      sc = sb
      vc = vb
    end
    if sb == 0 then
      sc = sa
      vc = va
    end
  end
  c = [sc,vc]
endfunction
```

//--- SOUSTRACTION : c = a - b

```
function c = SOUSTRACTION(a,b)
  sa = a(1) ; va = a(2:$)
  sb = b(1) ; vb = b(2:$)
  if sa*sb < 0 then
    sc = sa
    vc = addition(va,vb)
  end
  if sa*sb > 0 then
    s = comparaison(va,vb)
    if s > 0 then
      sc = sa
      vc = soustraction(va,vb)
    elseif s < 0 then
      sc = - sa
      vc = soustraction(vb,va)
    else
      sc = 0
      vc = 0
    end
  end
  if sa*sb == 0 then
    if sa == 0 then
      sc = - sb
      vc = vb
    end
    if sb == 0 then
      sc = sa
      vc = va
    end
  end
  c = [sc,vc]
endfunction
```

//--- MULTIPLICATION : c = a * b

```
function c = MULTIPLICATION(a,b)
  sa = a(1) ; va = a(2:$)
  sb = b(1) ; vb = b(2:$)
  vc = multiplication(va,vb)
  sc = sa*sb
  c = [sc,vc]
endfunction
```

//--- PUISSANCE : c = a ^ b

```
function c = PUISSANCE(a,b)
  sa = a(1) ; va = a(2:$)
  vc = puissance(va,b)
  if modulo(b,2) == 0 then
    sc = +1
  else
    sc = sa
  end
  c = [sc,vc]
endfunction
```

```
//---  DIVISION :  a = b * c + d

function [c,d] = DIVISION(a,b)
  sa = a(1) ; va = a(2:$)
  sb = b(1) ; vb = b(2:$)
  [vc,vd] = division(va,vb)
  if sa*sb > 0 then
    sc = +1
    sd = sa
  end
  if sa*sb < 0 then
    sc = -1
    sd = sa
  end
  if sa*sb == 0 then
    sc = 0
    sd = sa
  end
  if vc == 0 then, sc = 0, end
  if vd == 0 then, sd = 0, end
  c = [sc,vc]
  d = [sd,vd]
endfunction
```

```
//---  PARTIE  ENTIERE :  a = b * c + d

function [c,d] = PARTIE_ENTIERE(a,b)
  [c,d] = DIVISION(a,b)
  if SIGNE(a)*SIGNE(b) < 0 & ~NUL(d) then
    c = SOUSTRACTION(c,ENTIER(1))
    d = ADDITION(d,b)
  end
endfunction
```

Remarque

La fonction *DIVISION* effectue la division euclidienne de a par b dans $\mathbb{Z}$, soit a = bc + d, en se ramenant à la division de |a| par |b| dans $\mathbb{N}$, soit |a| = |b||c| + |d|. On attribue ensuite les signes convenables au quotient c et au reste d.
Plusieurs choix sont possibles pour exprimer la division de 2 entiers relatifs dans $\mathbb{Z}$.
Par exemple, effectuons la division euclidienne de 17 par -6 et de -17 par 6 :

1) En prenant le même quotient et le même reste que dans $\mathbb{N}$ au signe près : $17 = -6 \times -2 + 5$; $-17 = 6 \times -2 - 5$
2) En prenant un quotient qui est la partie entière de la fraction : $17 = -6 \times -3 - 1$; $-17 = 6 \times -3 + 1$
3) En prenant un reste toujours positif : $17 = -6 \times -2 + 5$; $-17 = 6 \times -3 + 1$
4) En prenant un reste toujours négatif : $17 = -6 \times -3 - 1$; $-17 = 6 \times -2 - 5$

Dans tous les cas le reste est inférieur au diviseur en module.

```
//---  PGCD :  c = pgcd(a,b)

function c = PGCD(a,b)
  c = a ; r = b
  while ~NUL(r)
    d = c ; c = r
    [q,r] = DIVISION(d,c)
  end
  c = ABSOLU(c)
endfunction

//---  EUCLIDE :  a u + b v = pgcd(a,b)

function [u,v,p] = EUCLIDE(a,b)
  r0 = a ; r1 = b
  u0 = [1,1] ; u1 = [0,0]
  v0 = [0,0] ; v1 = [1,1]
  while ~NUL(r1)
    [q,r] = DIVISION(r0,r1)
    t = MULTIPLICATION(q,r1)
    r2 = SOUSTRACTION(r0,t)
    t = MULTIPLICATION(q,u1)
    u2 = SOUSTRACTION(u0,t)
    t = MULTIPLICATION(q,v1)
    v2 = SOUSTRACTION(v0,t)
    r0 = r1 ; r1 = r2
    u0 = u1 ; u1 = u2
    v0 = v1 ; v1 = v2
  end
  u = u0 ; v = v0 ; p = r0
endfunction

//---  BACHET :  a x + b y = c

function [x,y] = BACHET(a,b,c)
  [u,v,p] = EUCLIDE(a,b)
  [q,r] = DIVISION(c,p)
  if NUL(r) then
    x = MULTIPLICATION(u,q)
    y = MULTIPLICATION(v,q)
  else
    x = []
    y = []
  end
endfunction
```

EXEMPLE

Résoudre dans $\mathbb{Z}$: A X + B Y = C

avec :

A = 13807164647207120874924653602728663842646698433 43

B = 88769597772021188347832701081595801386544431 3

C = 36133816043239514770706033836896832173 91

Le programme :

```
A = [1,1,3,8,0,7,1,6,4,6,4,7,2,0,7,1,2,0,8,7,4,9,2,4,6,5,3,6,0,2,7,2,8,6,6,3,8,4,2,6,4,6,6,9,8,4,3,3,4,3] ;
B = [1,8,8,7,6,9,5,9,7,7,7,2,0,2,1,1,8,8,3,4,7,8,3,2,7,0,1,0,8,1,5,9,5,8,0,1,3,8,6,5,4,4,4,3,1,3] ;
C = [1,3,6,1,3,3,8,1,6,0,4,3,2,3,9,5,1,4,7,7,0,7,0,6,0,3,3,8,3,6,8,9,6,8,3,2,1,7,3,9,1] ;
[X,Y] = BACHET(A,B,C) ;
printf('\n X = ') ; ECRIRE(X)
printf('\n Y = ') ; ECRIRE(Y)
```

donne la solution :

X = - 10023186656793665427743431452764337266

Y = 15589998370327030327433900924947520588133

Théorème des restes chinois

On souhaite résoudre un système de k congruences d'inconnue x :

$$\begin{cases} x \equiv r_1 \ (n_1) \\ x \equiv r_2 \ (n_2) \\ \vdots \\ \vdots \\ x \equiv r_k \ (n_k) \end{cases}$$

Supposons que les nombres n_i soient 2 à 2 premiers entre eux, et notons n leur produit : $n = n_1 \times n_2 \times ... \times n_k$
Pour chaque i de 1 à k, il existe un couple d'entiers (u_i , v_i) vérifiant l'équation de **Bachet-Bézout** : $u_i n_i + v_i (n/n_i) = 1$
On déduit la solution modulo n :

$$x = \sum_{i=1}^{k} r_i \ v_i \ \frac{n}{n_i}$$

EXEMPLE

Résoudre : $x \equiv 1 \ (7)$, $x \equiv 10 \ (7919)$, $x \equiv 100 \ (104729)$, $x \equiv 1000 \ (1299709)$, $x \equiv 10000 \ (1234567890)$

Programme :

```
R = list([1,1],[1,1,0],[1,1,0,0],[1,1,0,0,0],[1,1,0,0,0,0]) ;
N = list([1,7],[1,7,9,1,9],[1,1,0,4,7,2,9],[1,1,2,9,9,7,0,9],[1,1,2,3,4,5,6,7,8,9,0]) ;
for i = 1:length(N)
  for j = i+1:length(N)
    if ~isequal(PGCD(N(i),N(j)),ENTIER(1)) then, printf('\n Non Premier'), abort, end
  end
end
n = ENTIER(1) ;
for i = 1:length(N)
  n = MULTIPLICATION(n,N(i)) ;
end
s = ENTIER(0) ;
for i = 1:length(N)
  [q,r] = DIVISION(n,N(i)) ;
  [u,v] = BACHET(N(i),q,ENTIER(1)) ;
  y = MULTIPLICATION(q,v) ;
  z = MULTIPLICATION(y,R(i)) ;
  s = ADDITION(s,z) ;
end
[q,x] = DIVISION(s,n) ;
if SIGNE(x) < 0 then, x = ADDITION(x,n) ; end
printf('\n x = '), ECRIRE(x)
```

Solution : $x = 764772068738762259295636 0$

Le pgcd et le ppcm de plusieurs nombres entiers

- Le pgcd de 2 entiers a et b se calcule avec l'algorithme d'**Euclide** :

```
c = a ; r = b
tantque  r ≠ 0
|     d = c ; c = r
|     r = d modulo c
fin
pgcd = |c|
```

La fonction Scilab *PGCD* est la traduction de cet algorithme fondamental.

```
function c = PGCD(a,b)
  c = a ; r = b
  while ~NUL(r)
    d = c ; c = r
    [q,r] = DIVISION(d,c)
  end
  c = ABSOLU(c)
endfunction
```

- Le ppcm de 2 entiers a et b se déduit de la formule : $pgcd \times ppcm = a \times b$

La fonction Scilab *PPCM* effectue l'opération : $ppcm = ab / pgcd$

```
function c = PPCM(a,b)
  x = MULTIPLICATION(a,b)
  y = PGCD(a,b)
  [c,r] = DIVISION(x,y)
  c = ABSOLU(c)
endfunction
```

- Pour n entiers $a_1, a_2, a_3, \ldots, a_n$ on utilise les relations suivantes :

$$pgcd(a_1, a_2, a_3, \ldots, a_n) = pgcd(pgcd(a_1, a_2), a_3, \ldots, a_n)$$
$$ppcm(a_1, a_2, a_3, \ldots, a_n) = ppcm(ppcm(a_1, a_2), a_3, \ldots, a_n)$$

La fonction Scilab *PGCDPPCM* calcule le pgcd et le ppcm d'une liste L d'entiers relatifs.

```
function [pgcdl,ppcml] = PGCDPPCM(L)
  pgcdl = ENTIER(0)
  ppcml = ENTIER(1)
  for i = 1:length(L)
    pgcdl = PGCD(pgcdl,L(i))
    ppcml = PPCM(ppcml,L(i))
  end
endfunction
```

Remarque
Une autre méthode pour obtenir le pgcd et le ppcm d'une liste de nombres entiers consiste à les décomposer en facteurs premiers, mais s'ils sont de grande dimension, le temps de calcul est rédhibitoire ...

EXEMPLE

Calculer le pgcd et le ppcm des 7 nombres suivants :

a_1 = 37909408394210922470104077
a_2 = 1262119108246645625376456
a_3 = - 84223080952069776885286384
a_4 = - 12790385222974513081054057
a_5 = 25862698186873718388338379607
a_6 = 20901606819821195486381051
a_7 = - 61053442301724856877917596

Créons la liste des entiers vectoriels a_i :

```
L = list([1,3,7,9,0,9,4,0,8,3,9,4,2,1,0,9,2,2,4,7,0,1,0,4,0,7,7],...
         [1,1,2,6,2,1,1,9,1,0,8,2,4,6,6,4,5,6,2,5,3,7,6,4,5,6],...
         [-1,8,4,2,2,3,0,8,0,9,5,2,0,6,9,7,7,6,8,8,5,2,8,6,3,8,4],...
         [-1,1,2,7,9,0,3,8,5,2,2,2,9,7,4,5,1,3,0,8,1,0,5,4,0,5,7],...
         [1,2,5,8,6,2,6,9,8,1,8,6,8,7,3,7,1,8,3,8,3,3,8,3,7,9,6,0,7],...
         [1,2,0,9,0,1,6,0,6,8,1,9,8,2,1,1,9,5,4,8,6,3,8,1,0,5,1],...
         [-1,6,1,0,5,3,4,4,2,3,0,1,7,2,4,8,5,6,8,7,7,9,1,7,5,9,6]) ;
```

Appelons la fonction *PGCDPPCM* :

```
[pgcdl,ppcml] = PGCDPPCM(L) ;
```

Affichons le résultat à l'écran :

```
printf('\n pgcd ='), ECRIRE(pgcdl)
printf('\n ppcm ='), ECRIRE(ppcml)
```

On obtient :

pgcd = 392488048668479537
ppcm = 47106466591931248335593848568430731841172182937980285104631244675147 2

Mathématique & Informatique

Grands nombres rationnels

Un nombre rationnel f = a/b est représenté par une liste de 2 éléments : f = *list*(a,b), où a = f(1) et b = f(2) sont les vecteurs lignes représentant respectivement le numérateur et le dénominateur de la fraction.
Exemple : f = -3/10 ⇔ f = *list*([-1,3],[1,1,0])
Définissons les fonctions d'addition, de soustraction, de multiplication, de division, ainsi que la fonction de réduction des fractions, qui permettent de calculer dans ℚ.

```
//---  CONVERSION_1
function c = FRACTION(a,b)
  c = list(ENTIER(a),ENTIER(b))
endfunction

//---  CONVERSION_2
function b = nombre_rationnel(a)
  b = nombre_relatif(a(1))/nombre_relatif(a(2))
endfunction

//---  SIGNE
function s = F_SIGNE(f)
  s = SIGNE(f(1))*SIGNE(f(2))
endfunction

//---  MOINS :  g = - f
function g = F_MOINS(f)
  a = MOINS(f(1)) ; b = f(2)
  g = list(a,b)
endfunction

//---  ABSOLU :  g = |f|
function g = F_ABSOLU(f)
  a = ABSOLU(f(1)) ; b = ABSOLU(f(2))
  g = list(a,b)
endfunction

//---  INVERSION :  g = 1 / f
function g = F_INVERSION(f)
  a = f(1) ; b = f(2)
  g = list(b,a)
endfunction

//---  COMPARAISON
function s = F_COMPARAISON(f1,f2)
  f = F_SOUSTRACTION(f1,f2)
  sa = f(1)(1) ; sb = f(2)(1)
  s = sa*sb
endfunction

//---  REDUCTION
function g = F_REDUCTION(f)
  a = f(1) ; b = f(2)
  p = PGCD(a,b)
  [c,r] = DIVISION(a,p)
  [d,r] = DIVISION(b,p)
  g = list(c,d)
endfunction

//---  ADDITION :  f = f1 + f2
function f = F_ADDITION(f1,f2)
  a1 = f1(1) ; b1 = f1(2)
  a2 = f2(1) ; b2 = f2(2)
  t1 = MULTIPLICATION(a1,b2)
  t2 = MULTIPLICATION(a2,b1)
  a = ADDITION(t1,t2)
  b = MULTIPLICATION(b1,b2)
  f = F_REDUCTION(list(a,b))
endfunction

//---  SOUSTRACTION :  f = f1 - f2
function f = F_SOUSTRACTION(f1,f2)
  f3 = F_MOINS(f2)
  f = F_ADDITION(f1,f3)
endfunction

//---  MULTIPLICATION :  f = f1 * f2
function f = F_MULTIPLICATION(f1,f2)
  a1 = f1(1) ; b1 = f1(2)
  a2 = f2(1) ; b2 = f2(2)
  a = MULTIPLICATION(a1,a2)
  b = MULTIPLICATION(b1,b2)
  f = F_REDUCTION(list(a,b))
endfunction

//---  DIVISION :  f = f1 / f2
function f = F_DIVISION(f1,f2)
  f3 = F_INVERSION(f2)
  f = F_MULTIPLICATION(f1,f3)
endfunction

//---  PUISSANCE :  g = f ^ n
function g = F_PUISSANCE(f,n)
  if n < 0 then
    f = F_INVERSION(f)
    n = -n
  end
  f = F_REDUCTION(f)
  a = PUISSANCE(f(1),n)
  b = PUISSANCE(f(2),n)
  g = list(a,b)
endfunction

//---  TEST :  f = 0
function t = F_NUL(f)
  if length(f) == 0 then, t = %f, return, end
  t = F_COMPARAISON(f,FRACTION(0,1)) == 0
endfunction
```

La fonction *DECIMALE* est utilisée pour obtenir la représentation décimale (sans arrondi) d'une fraction.
Un réel $x \in \mathbb{R}$ est noté x = list(u, v, w) , où u est le signe de x, v sa partie entière, et w sa partie décimale.

```
function x = DECIMALE(f,d)
  a = [f(1)(2:$),zeros(1,d)]
  b = [f(2)(2:$)]
  [q,r] = division(a,b)
  k = length(q)
  if k > d then
    u = f(1)(1)*f(2)(1)
    v = q(1:k-d)
    w = q(k-d+1:k)
  else
    u = f(1)(1)*f(2)(1)
    v = 0
    w = [zeros(1,d-k),q(1:k)]
  end
  x = list(u,v,w)
endfunction
```

La fonction *ECRIRE* affiche la valeur X d'un entier relatif, d'un rationnel, d'un réel, ou d'un complexe.

```
function ECRIRE(X)
  select typeof(X)
  case "constant"
    s = X(1) ; v = X(2:$)
    if s < 0 then, printf('- '), end
    ecrire(v)
  case "list"
    if length(X) == 2 then
      s1 = X(1)(1) ; v1 = X(1)(2:$)
      s2 = X(2)(1) ; v2 = X(2)(2:$)
      if s1*s2 < 0 then, printf('- '), end
      ecrire(v1)
      if ~isequal(v1,0) & ~isequal(v2,1) then
        printf('/') , ecrire(v2)
      end
    end
    if length(X) == 3 then
      if X(1) < 0 then, printf('- '), end
      ecrire(X(2)), printf('.'), ecrire(X(3))
    end
    if length(X) == 4 then
      x = REELLE(X) ; y = IMAGINAIRE(X)
      if F_NUL(y) then, ECRIRE(x), return, end
      if ~F_NUL(x) then, ECRIRE(x), printf(' '), end
      if F_SIGNE(y) < 0 then, printf('- i'), end
      if F_SIGNE(y) > 0 then, printf('+ i'), end
      if F_COMPARAISON(F_ABSOLU(y),FRACTION(1,1)) <> 0 then
        ECRIRE(F_ABSOLU(y))
      end
    end
  end
endfunction
```

Par exemple, pour obtenir la représentation décimale de la fraction f = - 617283945/152 avec d = 57 décimales :

```
ECRIRE(DECIMALE(FRACTION(-617283945,152),57))
```

La valeur affichée est : - 4061078.585**526315789473684210**526315789473684210526315789473684210
Dans ce cas particulier, on constate la répétition d'une séquence de 18 chiffres à partir de la 4^{e} décimale.
Plus généralement : f nombre rationnel $\Leftrightarrow$ la suite des décimales est périodique à partir d'un certain rang

Mathématique & Informatique

EXEMPLE 1 : signature d'une permutation

On appelle permutation toute bijection σ de l'ensemble {1, 2, … , *n*} sur lui-même.
L'ensemble S_n (idem $\mathfrak{S}_n$) des permutations, muni de la loi de composition des applications, est un groupe fini d'ordre *n*!.

Soit une permutation $\sigma = \begin{pmatrix} 1 & 2 & \cdots & n \\ \sigma(1) & \sigma(2) & \cdots & \sigma(n) \end{pmatrix}$ représentée de manière abrégée par le vecteur ligne $[\sigma(1), \sigma(2), \ldots, \sigma(n)]$.

La permutation σ peut se décomposer en un nombre *t* de transpositions ($0 \le t \le n-1$ puisqu'il faut au plus $n-1$ échanges $\sigma(i) \leftrightarrow \sigma(j)$ pour revenir à la permutation identité). Selon que ce nombre *t* est pair ou impair, on dit que la signature de σ, notée sgn σ, est paire ou impaire et on lui attribue la valeur +1 ou −1 respectivement.

$$\operatorname{sgn}\sigma = \begin{cases} +1 & \text{si } \sigma \text{ est paire} \\ -1 & \text{si } \sigma \text{ est impaire} \end{cases}$$

On dispose de 3 méthodes pour calculer sgn σ :

- $\operatorname{sgn}\sigma = (-1)^t$ où *t* est le nombre de transpositions τ telles que $\sigma = \tau_1 \circ \tau_2 \circ \ldots \circ \tau_t$
- $\operatorname{sgn}\sigma = (-1)^c$ où *c* est le nombre de couples (i,j) tels que $i < j$ et $\sigma(i) > \sigma(j)$
- $\operatorname{sgn}\sigma = \prod_{i<j} \frac{\sigma(i) - \sigma(j)}{i - j}$

Quelle est la signature de la permutation $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\ 15 & 2 & 6 & 1 & 3 & 14 & 7 & 16 & 10 & 9 & 4 & 17 & 19 & 5 & 18 & 12 & 8 & 11 & 20 & 13 \end{pmatrix}$?

Solution

Les 3 méthodes de calcul de la signature conduisent à 3 fonctions différentes qui donnent le même résultat.

1) Méthode des transpositions

```
function sgn = signature(sigma)
    n = length(sigma)
    t = 0
    for i = 1:n-1
        j = find(sigma == i)
        if j <> i then
            k = sigma(i) ; sigma(i) = sigma(j) ; sigma(j) = k
            t = t + 1
        end
    end
    sgn = (-1)^t
endfunction
```

2) Méthode des couples

```
function sgn = signature(sigma)
    n = length(sigma)
    c = 0
    for j = 2:n
        v = find(sigma(1:j-1) > sigma(j))
        c = c + length(v)
    end
    sgn = (-1)^c
endfunction
```

3) Méthode des fractions (calcul dans $\mathbb{Q}$)

```
function sgn = signature(sigma)
    n = length(sigma)
    f = FRACTION(1,1)
    for j = 2:n
        for i = 1:j-1
            r = FRACTION(sigma(i) - sigma(j),i - j)
            f = F_MULTIPLICATION(f,r)
        end
    end
    sgn = nombre_rationnel(f)
endfunction
```

Pour répondre à la question posée, on exécute les instructions suivantes :

```
sigma = [15,2,6,1,3,14,7,16,10,9,4,17,19,5,18,12,8,11,20,13] ;
sgn = signature(sigma) ;
printf('\n sgn = %i',sgn)
```

On obtient : $sgn = -1$

Donc, la permutation $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\ 15 & 2 & 6 & 1 & 3 & 14 & 7 & 16 & 10 & 9 & 4 & 17 & 19 & 5 & 18 & 12 & 8 & 11 & 20 & 13 \end{pmatrix}$ est impaire.

Remarque 1

Si l'on souhaite effectuer un contrôle de validité du paramètre *sigma* de la fonction *signature*, il suffit d'introduire un booléen *check* qui vaut *%t* (vrai) lorsque le vecteur *sigma* représente effectivement une permutation et *%f* (faux) sinon. Pour cela, on trie les coordonnées du vecteur *sigma* suivant les valeurs croissantes (fonction *gsort*), puis on les compare à la suite des entiers de 1 à *n* où *n* est la dimension du vecteur *sigma* (fonction *isequal* et *length*). Le résultat est affecté à la variable *check*.

```
function sgn = signature(sigma)
    check = isequal(gsort(sigma,'g','i'),[1:length(sigma)])
    if ~check then, sgn = [], return, end
    n = length(sigma)
    f = FRACTION(1,1)
    for j = 2:n
        for i = 1:j-1
            r = FRACTION(sigma(i) - sigma(j),i - j)
            f = F_MULTIPLICATION(f,r)
        end
    end
    sgn = nombre_rationnel(f)
endfunction
```

Remarque 2

Pour calculer sgn σ, la méthode des fractions est la plus simple (sgn σ = produit de $n(n-1)/2$ nombres rationnels) mais elle n'est pas la plus rapide. Une méthode optimisée consiste, pour de grandes valeurs de *n*, à appliquer au vecteur *sigma* un algorithme de tri efficace (par exemple le tri par segmentation de **Hoare**) en comptant le nombre *m* d'échanges de 2 valeurs $\sigma(i)$ et $\sigma(j)$, puis à vérifier si *m* est pair (sgn $\sigma = +1$) ou si *m* est impair (sgn $\sigma = -1$) .

EXEMPLE 2 : tri de nombres rationnels

Ecrire une fonction *F_TRI* qui permet de classer dans l'ordre croissant un ensemble de n fractions F_1 , F_2 , ... , $F_n \in \mathbb{Q}$.
On utilisera l'algorithme de tri par segmentation de **Hoare** (Quicksort).

```
function B = F_TRI(A)
    function tri(inf,sup)
        if inf < sup then
            k = segmentation(inf,sup)
            tri(inf,k-1)
            tri(k+1,sup)
        end
    endfunction
    function k = segmentation(inf,sup)
        global B
        i = grand(1,1,'uin',inf,sup)
        j = inf + 1
        k = sup
        pivot = B(i)
        B(i) = B(inf)
        B(inf) = pivot
        while j <= k
            if F_COMPARAISON(B(j),pivot) < 1 then
                j = j + 1
            else
                while F_COMPARAISON(B(k),pivot) == 1
                    k = k - 1
                end
                if j < k then
                    temp = B(j)
                    B(j) = B(k)
                    B(k) = temp
                    j = j + 1
                    k = k - 1
                end
            end
        end
        if k <> inf then
            temp = B(inf)
            B(inf) = B(k)
            B(k) = temp
        end
    endfunction
    global B
    B = A
    n = length(B)
    tri(1,n)
endfunction
```

Exemple d'exécution :

```
A = list(FRACTION(4,3),FRACTION(123456789,123456788),FRACTION(-1,2),FRACTION(1,1),FRACTION(-5,9)) ;
B = F_TRI(A) ;
printf('\n ---- Tableau non trié'), AFFICHE(A)
printf('\n ---- Tableau trié'), AFFICHE(B)
```

Résultat :

```
---- Tableau non trié
4/3    123456789/123456788    - 1/2    1    - 5/9
---- Tableau trié
- 5/9    - 1/2    1    123456789/123456788    4/3
```

EXEMPLE 3 : développement archimédien d'une fraction

Soit $\frac{a}{b}$ $(0 < a < b)$ une fraction irréductible comprise entre 0 et 1.

Montrer qu'il existe un nombre fini d'entiers naturels n_i croissants tels que :

$$\frac{a}{b} = \sum_{i=1}^{k} \frac{1}{n_i} = \frac{1}{n_1} + \frac{1}{n_2} + \cdots + \frac{1}{n_k}$$

Effectuer le développement archimédien de $\frac{14112011}{9561076584}$

Solution

La division euclidienne de b par a s'écrit : $b = aq + r$ avec $0 \le r < a$

Si $r = 0$, on a immédiatement : $\frac{a}{b} = \frac{1}{q}$

Lorsque $r \neq 0$, on peut obtenir un reste $(r - a)$ négatif en incrémentant q de 1 :

$$b = a(q+1) + (r - a) \Rightarrow \frac{a}{b} = \frac{1}{q+1} + \frac{a - r}{b(q+1)}$$

Les inégalités $0 < r < a$, $a < b$ et $q \ge 1$ entraînent $0 < a - r < a < b < b(q+1)$

Notons $\frac{a_1}{b_1}$ la fraction irréductible équivalente à $\frac{a - r}{b(q+1)}$ et posons $n_1 = q + 1$, il vient :

$$\frac{a}{b} = \frac{1}{n_1} + \frac{a_1}{b_1} \quad \text{avec } 0 < a_1 \le a - r < a \text{ et } a_1 < b_1$$

On peut appliquer à $\frac{a_1}{b_1}$ la même décomposition qu'à $\frac{a}{b}$:

$$\frac{a_1}{b_1} = \frac{1}{n_2} + \frac{a_2}{b_2} \quad \text{avec } 0 < a_2 < a_1 \text{ et } a_2 < b_2$$

On déduit :

$$\frac{a}{b} = \frac{1}{n_1} + \frac{1}{n_2} + \frac{a_2}{b_2}$$

En continuant ainsi, on obtient une suite strictement décroissante de termes a_i minorée par 0

Donc, il existe nécessairement un $a_k = 0$ vérifiant :

$$\frac{a_{k-1}}{b_{k-1}} = \frac{1}{n_k} + \frac{a_k}{b_k} = \frac{1}{n_k}$$

Finalement, on aboutit à la formule recherchée :

$$\frac{a}{b} = \frac{1}{n_1} + \frac{1}{n_2} + \cdots + \frac{1}{n_k}$$

Il reste à écrire une fonction Scilab *archimedien* qui fournit la liste L des $\frac{1}{n_i}$ à partir de la fraction $f = \frac{a}{b}$

Remarque

Le développement archimédien d'une fraction $\frac{a}{b}$ conduit généralement à de très grands entiers n_i

```
function L = archimedien(f)
    L = list()
    while ~F_NUL(f)
        f = F_REDUCTION(f)
        a = NUMERATEUR(f)
        b = DENOMINATEUR(f)
        [q,r] = DIVISION(b,a)
        if ~NUL(r) then
            q = ADDITION(q,ENTIER(1))
            a = SOUSTRACTION(a,r)
            b = MULTIPLICATION(b,q)
            f = list(a,b)
        else
            f = FRACTION(0,1)
        end
        L($+1) = list(ENTIER(1),q)
    end
endfunction
```

Pour développer la fraction $\frac{14112011}{9561076584}$, on exécute le programme suivant :

```
a = [1,1,4,1,1,2,0,1,1] ;
b = [1,9,5,6,1,0,7,6,5,8,4] ;
f = list(a,b) ;
L = archimedien(f) ;
printf('\n Développement archimédien de '), ECRIRE(f)
for i = 1:length(L), printf('\n '), ECRIRE(L(i)), end
//---  Vérification
F = FRACTION(0,1) ;
for i = 1:length(L), F = F_ADDITION(F,L(i)), end
if F_COMPARAISON(F,f) == 0 then, printf('\n OK'), end
```

On obtient :

$$\frac{14112011}{9561076584} = \cfrac{1}{678 + \cfrac{1}{944012 + \cfrac{1}{3987832649 + \cfrac{1}{4780487626775335050344421 + \cfrac{1}{80211984709467863391075306547667690487014273185 + \cfrac{1}{24993469781781054224332856163019398115906382081520058049774678090246031194374690482209618023400}}}}}}$$

EXEMPLE 4 : calcul de probabilités

Par définition, la probabilité P d'un événement est :

$$P = \frac{nombre\ de\ cas\ favorables}{nombre\ de\ cas\ possibles}$$

Le calcul de P est un problème de dénombrement, sa résolution fait appel à l'analyse combinatoire.

Application

On choisit au hasard un ensemble de n personnes.
Quelle est la probabilité P_n pour qu'au moins deux personnes soient nées le même jour de l'année ? Calculer P_{23}.
Quel est le nombre minimal de personnes qu'il faut réunir pour être sûr à plus de 99 % qu'au moins deux d'entre elles aient le même anniversaire ? Même question avec une certitude supérieure à 99.9 %.
Quelle est la probabilité $P(n,k)$ pour que k personnes exactement aient le même anniversaire dans un ensemble de n personnes ? Si $k = 2$, quel est le nombre de personnes qu'il faut réunir pour que cette probabilité soit maximale ? Calculer cette valeur P_{max}.
Ce problème, appelé paradoxe des anniversaires, est dû à **Richard von Mises**.

Pour simplifier, considérons que toutes les années ont 365 jours (on fait abstraction des années bissextiles).
Remarquons d'abord que dans un ensemble de plus de 365 personnes, il y en a forcément au moins 2 qui sont nées le même jour de l'année (appliquer le principe des tiroirs de **Dirichlet**). D'où : $P_n = 1$ si $n > 365$.
Dans un ensemble de n personnes ($1 \le n \le 365$), la probabilité p_n pour qu'il n'y ait pas deux personnes ayant le même anniversaire est égale au produit des probabilités p_i pour que le jour de naissance de la personne numérotée i ($1 \le i \le n$) soit différent de celui des personnes numérotées 1, 2, … , $i - 1$:

$$p_n = \prod_{i=1}^{n} p_i = \prod_{i=1}^{n} \frac{365-(i-1)}{365} = \frac{365\times 364\times\cdots\times(365-n+1)}{365^n} = \frac{365!}{(365-n)!\ 365^n}$$

On en déduit la probabilité P_n pour que deux personnes au moins aient le même jour anniversaire (événement contraire du précédent) :

$$P_n = 1 - p_n = 1 - \frac{365!}{(365-n)!\ 365^n}$$

Pour $n = 23$, le programme de calcul de P_{23} s'écrit :

```
n = 23 ;
N = entier(1) ;
D = entier(1) ;
for i = 1:n
    N = multiplication(N,entier(365-i+1)) ;
    D = multiplication(D,entier(365)) ;
end
F = list([1,N],[1,D]) ;
P = F_SOUSTRACTION(FRACTION(1,1),F) ;
printf('\n P%i = ',n), ECRIRE(P), printf(' = '), ECRIRE(DECIMALE(P,10))
```

L'exécution donne :

$$P_{23} = \frac{38093904702297390785243708291056390518886454060947061}{75091883268515350125426207425223147563269805908203125} \approx 0.5072972343$$

Donc, dans un ensemble de 23 personnes, la probabilité pour qu'au moins 2 personnes aient le même anniversaire est égale à 50.73 %.

Cherchons la plus petite valeur de n telle que $P_n > \frac{a}{b}$ (a et b entiers vérifiant $b > a > 0$).

Posons $p_n = \frac{x}{y}$, on veut que $1 - \frac{x}{y} > \frac{a}{b}$, soit $(b-a)y > bx$, c'est-à-dire $(b-a) \times 365^n > b \times 365 \times 364 \times \ldots \times (365 - n + 1)$

Notons $X = b \times 365 \times 364 \times \ldots \times (365 - n + 1)$ et $Y = (b-a) \times 365^n$, la condition $Y > X$ conduit au programme suivant :

```
while %t
  a = input('a = ? ') ;
  if a == [] then, break, end
  b = input('b = ? ') ;
  if b == [] then, break, end
  X = entier(b) ;
  Y = entier(b-a) ;
  for n = 1:365
      X = multiplication(X,entier(365-n+1)) ;
      Y = multiplication(Y,entier(365)) ;
      if comparaison(Y,X) > 0 then, break, end
  end
  printf('\n n = %i',n)
end
```

On trouve :

$a = 50, b = 100 \quad \Rightarrow \quad n = 23$
$a = 99, b = 100 \quad \Rightarrow \quad n = 57$
$a = 999, b = 1000 \quad \Rightarrow \quad n = 70$

Ainsi, sur 57 personnes choisies au hasard, on peut affirmer avec un risque d'erreur de 1 % qu'il y en a au moins deux qui ont le même anniversaire. Avec 70 personnes, ce risque tombe à 1 ‰.

Soit $P(n,k)$ la probabilité pour que k personnes exactement aient le même anniversaire dans un ensemble de n personnes. Pour un groupe de k personnes, il y a 365 façons de choisir un jour anniversaire commun. Par suite, afin de satisfaire la condition d'unicité, il reste 364 jours anniversaires différents convenables pour la $(k+1)^e$ personne, 363 pour la $(k+2)^e$ personne, etc … jusqu'à la n^e personne pour laquelle il ne reste plus que $365 - (n-k)$ jours anniversaires différents convenables. D'une part, le nombre de groupes de k personnes parmi n est C_n^k. D'autre part, le nombre de valeurs possibles que peut prendre le n-uplet $(J_1, J_2, \ldots, J_n)$ des jours anniversaires de n personnes est 365^n. En conséquence :

$$P(n,k) = C_n^k \times \frac{365 \times 364 \times \cdots \times [365 - (n-k)]}{365^n} = \frac{n!\ 365!}{k!\ (n-k)!\ (364+k-n)!\ 365^n} \qquad (1 \le k \le n \le 365)$$

On en déduit la fonction de calcul de $P(n,k)$:

```
function P = anniversaire(n,k)
    N = binomiaux(n,k)
    for i = 1:n-k
        N = multiplication(N,entier(365-i))
    end
    D = puissance(entier(365),n-1)
    P = list([1,N],[1,D])
    P = F_REDUCTION(P)
endfunction
```

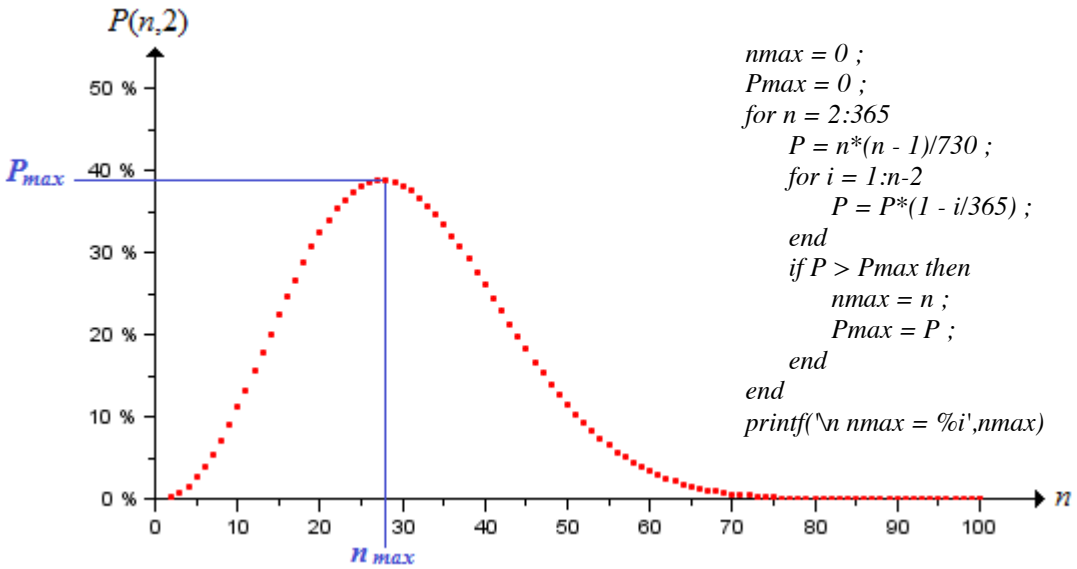
En particulier, pour $n = 23$ et $k = 2$:

```
P = anniversaire(23,2) ;
printf('\n P = '), ECRIRE(P), printf(' = '), ECRIRE(DECIMALE(P,10))
```

Ce qui donne : $P = \dfrac{272900541610878825453824264750559461581020058815037 44}{750918832685153501254262074252231475632698059082031 25} \approx 0.3634221566$

Donc, dans un ensemble de 23 personnes, la probabilité pour que 2 personnes, et 2 seulement, aient le même anniversaire est égale à 36.34 %.

Traçons la courbe $P(n,2) = f(n)$ et recherchons, avec la méthode numérique, l'entier n_{max} tel que $P(n_{max},2) = \underset{2 \le n \le 365}{\text{Max}} P(n,2)$




```
nmax = 0 ;
Pmax = 0 ;
for n = 2:365
    P = n*(n - 1)/730 ;
    for i = 1:n-2
        P = P*(1 - i/365) ;
    end
    if P > Pmax then
        nmax = n ;
        Pmax = P ;
    end
end
printf('\n nmax = %i',nmax)
```

Résultat : $n_{max} = 28$
Noter que la partie ascendante de la courbe correspond à un nombre n trop petit pour trouver (en probabilité) 2 personnes ayant le même anniversaire, et que la partie descendante de la courbe correspond à un nombre n trop grand pour trouver (en probabilité) seulement 2 personnes ayant le même anniversaire.

On peut également obtenir n_{max} avec la méthode algébrique suivante :

$$\frac{P(n+1,2)}{P(n,2)} = \frac{(n+1)!\,(n-2)!\,(366-n)!\,365^n}{n!\,(n-1)!\,(365-n)!\,365^{n+1}} = \frac{-n^2 + 365n + 366}{365n - 365}$$

$$\frac{P(n+1,2)}{P(n,2)} - 1 = \frac{-n^2 + 731}{365\,(n-1)} \quad \Rightarrow \quad n_{max} = \left\lceil \sqrt{731} \right\rceil = 28$$

Exécutons les instructions ci-dessous :

```
Pmax = anniversaire(28,2) ;
printf('\n Pmax = '), ECRIRE(Pmax), printf(' = '), ECRIRE(DECIMALE(Pmax,10))
```

On obtient la valeur exacte de la probabilité $P_{max} = P(28,2)$:

$$P_{max} = \frac{375974934421091048052701976471976438061563054621757500476075540 48}{972942812930194772684187860193473618121636280143246650695800781 25} \approx 0.3864306611$$

Si $n > 365$, il y a au moins 2 personnes qui ont le même anniversaire et la probabilité que se soient les seules décroît avec le nombre n. D'où : $P(n,2) < P(365,2) < P(28,2)$ si $n > 365$.

Donc, la probabilité pour que 2 personnes, et 2 seulement, aient le même anniversaire est maximale dans un ensemble de 28 personnes. Elle est égale à 38.64 %.

Mathématique & Informatique

Loi binomiale

Une urne contient a boules blanches et b boules noires.
La probabilité de tirer une boule blanche est $p = \frac{a}{a + b}$, la probabilité de tirer une boule noire est $q = \frac{b}{a + b}$.
On effectue n tirages d'une boule avec remise. La probabilité P d'obtenir k boules blanches est :

$$P = C_n^k \, p^k \, q^{n - k}$$

Fonction Scilab pour calculer la valeur exacte de P :

```
function P = binomiale(p,q,n,k)
    P = FRACTION(0,1)
    if n < k then, return, end
    F1 = list([1,binomiaux(n,k)],[1,1])
    F2 = F_PUISSANCE(p,k)
    F3 = F_PUISSANCE(q,n-k)
    P = F_MULTIPLICATION(F1,F_MULTIPLICATION(F2,F3))
endfunction
```

Application

Un quiz est composé de 50 QCM (Questions à Choix Multiples).
A chaque question, il est proposé quatre réponses dont une seule est correcte.
On suppose qu'une personne répond au hasard à toutes les questions du quiz.
Quelle est la probabilité P pour qu'elle ait obtenu au moins la moitié de bonnes réponses ?

Si l'on se réfère au modèle de l'urne, une bonne réponse est une boule blanche et une mauvaise réponse est une boule noire.
En choisissant les réponses au hasard, le quiz obéit à une loi binomiale de paramètres : $p = \frac{1}{4}$; $q = \frac{3}{4}$; $n = 50$

La probabilité $P(k)$ de donner k bonnes réponses sur 50 est : $P(k) = C_{50}^k \left(\frac{1}{4}\right)^k \left(\frac{3}{4}\right)^{50 - k}$

La probabilité P de donner au moins 25 bonnes réponses sur 50 est : $P = \sum_{k=25}^{50} P(k)$

Le programme de calcul de P s'écrit :

```
p = FRACTION(1,4) ; q = FRACTION(3,4) ; n = 50 ;
P = FRACTION(0,1) ;
for k = 25:50
    P = F_ADDITION(P,binomiale(p,q,n,k)) ;
end
printf('\n P = '), ECRIRE(P), printf(' = '), ECRIRE(DECIMALE(P,10))
```

On trouve :

$$P = \frac{388260626299381781417125 93}{316912650057057350374175801344} \approx 0.0001225134$$

Donc, il y a 0.012 % de chances d'obtenir au moins la moitié de bonnes réponses en répondant au hasard aux questions du quiz.

Mathématique & Informatique

Note

Utilisons les fonctions statistiques de Scilab qui calculent les valeurs de la fonction de répartition (cumulative distribution function, cdf en abrégé) pour différentes lois : binomiale (bin), Poisson (poi), normale (nor), …

- Loi binomiale

  La probabilté P_B est donnée par la formule :

$$P_B = \sum_{k=25}^{50} C_{50}^{k} \left(\frac{1}{4}\right)^k \left(\frac{3}{4}\right)^{50-k} = 1 - \sum_{k=0}^{24} C_{50}^{k} \left(\frac{1}{4}\right)^k \left(\frac{3}{4}\right)^{50-k}$$

  Le programme de calcul de P_B s'écrit :

```
[P,Q] = cdfbin("PQ",24,50,1/4,3/4) ;
printf('\n P_B = %.10f',Q)
```

  On retrouve la valeur décimale $P = P_B = 0.0001225135 \approx 0.012$ %.

- Approximation de la loi binomiale par une loi de **Poisson** : $\mathscr{B}(n,p) \sim \mathscr{P}(np)$

  La probabilté P_P est donnée par la formule :

$$P_p = \sum_{k=25}^{+\infty} e^{-m} \frac{m^k}{k!} = 1 - \sum_{k=0}^{24} e^{-m} \frac{m^k}{k!} \qquad \text{avec } m = np = \frac{25}{2}$$

  Le programme de calcul de P_P s'écrit :

```
[P,Q] = cdfpoi("PQ",24,25/2) ;
printf('\n P_P = %.10f',Q)
```

  On trouve $P_P = 0.0011924488 \approx 0.12$ % alors que la valeur de P est 0.012 %.
  L'erreur est due au fait que la probabilité p n'est pas suffisamment petite pour utiliser la loi de **Poisson**.

- Approximation de la loi binomiale par une loi normale : $\mathscr{B}(n,p) \sim \mathscr{N}(np,npq)$

  La probabilté P_N est donnée par la formule :

$$P_N = \int_{25}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2} dx = 1 - \int_{-\infty}^{25} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2} dx \qquad \text{avec } \begin{cases} m = np = \dfrac{25}{2} \\ \sigma = \sqrt{npq} = \sqrt{\dfrac{75}{8}} \end{cases}$$

  Le programme de calcul de P_N s'écrit :

```
[P,Q] = cdfnor("PQ",25,25/2,sqrt(75/8)) ;
printf('\n P_N = %.10f',Q)
```

  On trouve $P_N = 0.0000222785 \approx 0.0022$ % alors que la valeur de P est 0.012 %.
  L'erreur est due au fait que le nombre n n'est pas suffisamment grand pour utiliser la loi normale.

Loi binomiale négative

Une urne contient a boules blanches et b boules noires.

La probabilité de tirer une boule blanche est $p = \frac{a}{a + b}$, la probabilité de tirer une boule noire est $q = \frac{b}{a + b}$.

On tire une boule avec remise jusqu'à ce que l'on ait obtenu n boules blanches.
La probabilité P pour que k tirages soient nécessaires est :

$$P = C_{k-1}^{n-1} p^n q^{k-n}$$

Fonction Scilab pour calculer la valeur exacte de P :

```
function P = binomiale_negative(p,q,n,k)
    P = FRACTION(0,1)
    if n < 1 | n > k then, return, end
    F1 = list([1,binomiaux(k-1,n-1)],[1,1])
    F2 = F_PUISSANCE(p,n)
    F3 = F_PUISSANCE(q,k-n)
    P = F_MULTIPLICATION(F1,F_MULTIPLICATION(F2,F3))
endfunction
```

Application

Un tireur atteint une certaine cible avec une probabilité égale à $\frac{3}{5}$.

Il participe à une épreuve sportive où on est autorisé à interrompre les tirs dès que l'on a effectué 30 coups au but.
Quelle est la probabilité P pour qu'il réussisse l'épreuve en 50 tirs au plus ?

On est dans le cas d'une loi binomiale négative de paramètres : $p = \frac{3}{5}$; $q = \frac{2}{5}$; $n = 30$

La probabilité $P(k)$ de s'arrêter au k^e tir ($k \geq 30$) est : $P(k) = C_{k-1}^{29} \left(\frac{3}{5}\right)^{30} \left(\frac{2}{5}\right)^{k-30}$

La probabilité P de s'arrêter au plus tard au 50e tir est : $P = \sum_{k=30}^{50} P(k)$

Le programme de calcul de P s'écrit :

```
p = FRACTION(3,5) ; q = FRACTION(2,5) ; n = 30 ;
P = FRACTION(0,1) ;
for k = 30:50
    P = F_ADDITION(P,binomiale_negative(p,q,n,k)) ;
end
printf('\n P = '), ECRIRE(P), printf(' = '), ECRIRE(DECIMALE(P,10))
```

On trouve :

$$P = \frac{4982991193359134354075018266354824 9}{8881784197001252323389053344726 5625} \approx 0.5610349320$$

Donc, le tireur a 56.1 % de chances de réussir l'épreuve en 50 tirs au plus.
On retrouve ce résultat avec la fonction Scilab *cdfnbn* :

```
[P,Q] = cdfnbn("PQ",20,30,3/5,2/5) ;
printf('\n P_BN = %.10f',P)
```

Loi hypergéométrique

Une urne contient a boules blanches et b boules noires.
On tire simultanément n boules. La probabilité P d'avoir k boules blanches est :

$$P = \frac{C_a^k C_b^{n-k}}{C_{a+b}^n}$$

Fonction Scilab pour calculer la valeur exacte de P :

```
function P = hypergeometrique(a,b,n,k)
    P = FRACTION(0,1)
    if n > a+b then, return, end
    if k >= max(0,n-b) & k <= min(a,n) then
        N1 = binomiaux(a,k)
        N2 = binomiaux(b,n-k)
        N3 = binomiaux(a+b,n)
        P = list([1,multiplication(N1,N2)],[1,N3])
        P = F_REDUCTION(P)
    end
endfunction
```

Application 1

Dans un grand bassin, on élève 1000 poissons dont 20 % présentent le caractère C.
On pêche au hasard 50 poissons, quelle est la probabilité P pour que 10 d'entre eux présentent le caractère C ?

On reconnait une loi hypergéométrique (un poisson présentant le caractère C est une boule blanche, un poisson ne présentant pas le caractère C est une boule noire) avec les paramètres :

a = 200 (nombre de poissons du bassin présentant le caractère C)
b = 800 (nombre de poissons du bassin ne présentant pas le caractère C)
n = 50 (nombre de poissons pêchés)
k = 10 (nombre de poissons pêchés présentant le caractère C)

D'où :

$$P = \frac{C_{200}^{10} C_{800}^{40}}{C_{1000}^{50}}$$

Calculons P :

```
P = hypergeometrique(200,800,50,10) ;
printf('\n P = '), ECRIRE(P), printf(' = '), ECRIRE(DECIMALE(P,10))
```

Résultat :

$$P = \frac{25864426169694235643236699468913178563516803671051661741482560}{180305279160749810257197483555779080176165759882359369559005883} \approx 0.1434479694$$

Donc, il y a environ 14 % de chances pour que 10 des poissons pêchés présentent le caractère C.

Application 2

Un jeu de Tarot comporte 78 cartes : 22 atouts (numérotés de 1 à 21, plus l'Excuse), 16 cartes habillées (Roi, Dame, Cavalier, Valet dans les 4 couleurs : cœur, carreau, pique, trèfle) et 40 cartes basses (Dix, Neuf, Huit, Sept, Six, Cinq, Quatre, Trois, Deux, As dans les 4 couleurs : cœur, carreau, pique, trèfle).
Après battage, chacun des 3 joueurs reçoit 24 cartes et il subsiste sur la table un talon de 6 cartes.
Quelles sont les probabilités pour qu'un joueur désigné possède une simple poignée (13 ou 14 atouts), une double poignée (15, 16 ou 17 atouts), une triple poignée (18 à 22 atouts) ?

⇨ Règlement officiel du jeu de Tarot : **http://fftarot.fr/assets/documents/R-RO201306.pdf**

La probabilité $P(k)$ pour qu'un joueur désigné possède k atouts ($0 \le k \le 22$) est donnée par la loi hypergéométrique :

$$P(k) = \frac{C_{22}^{k} C_{56}^{24-k}}{C_{78}^{24}} \qquad \sum_{k=0}^{22} P(k) = 1$$

Notons P_1 la probabilté d'avoir une simple poignée, P_2 la probabilté d'avoir une double poignée et P_3 la probabilté d'avoir une triple poignée. On a :

$P_1 = P(13) + P(14)$
$P_2 = P(15) + P(16) + P(17)$
$P_3 = P(18) + P(19) + P(20) + P(21) + P(22)$

Pour calculer P_1, P_2, P_3 exécutons le programme suivant :

```
P1 = FRACTION(0,1) ;
for k = 13:14, P1 = F_ADDITION(P1,hypergeometrique(22,56,24,k)) ; end
printf('\n P1 = '), ECRIRE(P1), printf(' = '), ECRIRE(DECIMALE(P1,20))
P2 = FRACTION(0,1) ;
for k = 15:17, P2 = F_ADDITION(P2,hypergeometrique(22,56,24,k)) ; end
printf('\n P2 = '), ECRIRE(P2), printf(' = '), ECRIRE(DECIMALE(P2,20))
P3 = FRACTION(0,1) ;
for k = 18:22, P3 = F_ADDITION(P3,hypergeometrique(22,56,24,k)) ; end
printf('\n P3 = '), ECRIRE(P3), printf(' = '), ECRIRE(DECIMALE(P3,20))
P = F_ADDITION(P1,F_ADDITION(P2,P3)) ;
printf('\n P = '), ECRIRE(P), printf(' = '), ECRIRE(DECIMALE(P,20))
```

On obtient :

$$P_1 = \frac{44930336388}{41571842572157} \approx 0.0010807877064869954 0$$

$$P_2 = \frac{2953111293}{166287370288628} \approx 0.00001775908349428000$$

$$P_3 = \frac{256033}{83143685144314} \approx 0.00000003079404040 79$$

$$P = P_1 + P_2 + P_3 = \frac{6299136859}{5734047251332} \approx 0.00109854986938531620$$

Au jeu de Tarot à 3 joueurs, après la distribution des cartes, il y a environ 1 chance sur 1000 de posséder une poignée.
Calcul similaire pour le Tarot à 4 joueurs : on trouve environ 5 chances sur 1000 de posséder une poignée au départ.

Mathématique & Informatique

Calcul du nombre φ

Le nombre d'or φ est donné par : $\varphi = \dfrac{1+\sqrt{5}}{2}$

La suite récurrente : $\begin{cases} u_n = 1 + \dfrac{1}{u_{n-1}} \\ u_0 = 1 \end{cases}$ $u_0 = 1,\ u_1 = 2,\ u_2 = \dfrac{3}{2},\ u_3 = \dfrac{5}{3},\ u_4 = \dfrac{8}{5},\ u_5 = \dfrac{13}{8},\ u_6 = \dfrac{21}{13},\ \cdots$

converge (éventuellement) vers la limite l telle que : $l = 1 + \dfrac{1}{l} \Rightarrow l^2 - l - 1 = 0 \Rightarrow l = \dfrac{1+\sqrt{5}}{2} = \varphi$ puisque $u_n > 0$

D'une part : $u_n - \varphi = 1 + \dfrac{1}{u_{n-1}} - \left(1 + \dfrac{1}{\varphi}\right) = \dfrac{\varphi - u_{n-1}}{u_{n-1}\,\varphi} = -\dfrac{1}{u_{n-1}\,\varphi}(u_{n-1} - \varphi)$

D'autre part : $\forall n > 1 :\ u_n = 1 + \dfrac{1}{1 + \dfrac{1}{1+a}}$ avec $a \geq 0 \Rightarrow \forall n \geq 1 :\ u_n \geq \dfrac{3}{2} \Rightarrow \forall n > 1 :\ u_{n-1}\,\varphi \geq \dfrac{3}{2}\,\varphi$

Posons $k = \dfrac{3}{2}\,\varphi$ et déduisons de ce qui précède une majoration de l'erreur d'approximation :

$$\forall n \geq 1 :\quad |u_n - \varphi| \leq \frac{1}{k}|u_{n-1} - \varphi| \leq \frac{1}{k^2}|u_{n-2} - \varphi| \leq \cdots \leq \frac{1}{k^{n-1}}|u_1 - \varphi| = \frac{2-\varphi}{k^{n-1}} = \varepsilon$$

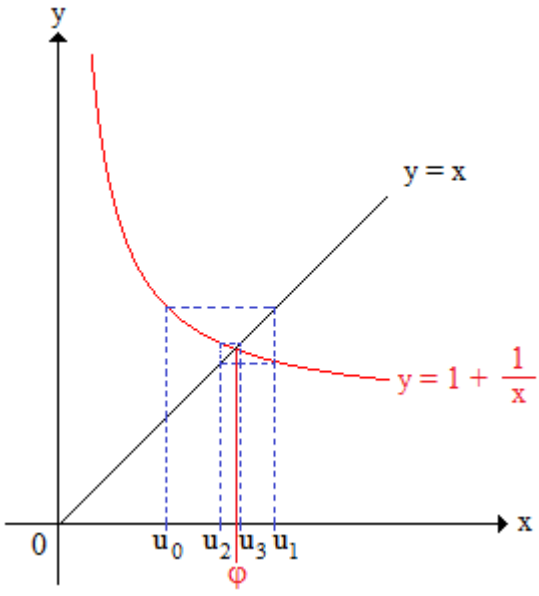
Comme $\varepsilon \to 0$ quand $n \to +\infty$, la suite u_n tend bien vers φ

Pour obtenir d décimales exactes de φ, on prend : $\varepsilon < 10^{-d} \Rightarrow n > \dfrac{Log(2-\varphi) + d\,Log(10)}{Log(\frac{3}{2}\varphi)} + 1$

D'où le programme :

```
exec("Scilab-Arithmetique.sce",-1)

d = 100 ;
phi = (1+sqrt(5))/2 ;
n = ceil((log(2-phi) + d*log(10))/log(3/2*phi)) + 1 ;
u = FRACTION(1,1) ;
for i = 1:n
    u = F_ADDITION(FRACTION(1,1), F_INVERSION(u)) ;
end
phi = DECIMALE(u,d) ;
printf('\n phi = '), ECRIRE(phi)
```



Les 100 premières décimales du nombre d'or sont :

φ = 1.6180339887498948482045868343656381177203091798057628621354486227052604628189024497072072041893911374

On retrouve fréquemment le nombre d'or dans les œuvres de **Léonard de Vinci**.
Le nombre φ est également utilisé en architecture. De nombreux édifices de la Renaissance, comme les châteaux de la Loire, sont bâtis selon des règles de proportion se référant au nombre d'or. **Le Corbusier** (1887 – 1965) a imaginé un système de mesure basé sur le nombre d'or : le Modulor.

Calcul du nombre e

La notion d'exponentielle "*e*" est issue des travaux de **Neper** et d'**Euler** (lettre *e* initiale de **Euler** ?).
La fonction associée e^x est la seule fonction (à une constante multiplicative près) qui est égale à sa dérivée.
On la rencontre dans de nombreux domaines des Mathématiques et de la Physique.
Les 2 constantes fondamentales des mathématiques, *e* et π, sont liées par la relation $e^{i\pi} + 1 = 0$.

Le développement en série entière de la fonction exponentielle est :

$$e^x = \sum_{n=0}^{+\infty} \frac{x^n}{n!}$$

Son rayon de convergence est infini et le reste de **Lagrange** d'ordre *n* s'écrit : $R_n = e^{\theta x} x^{n+1} / (n+1)!$, $0 < \theta < 1$
Pour $x = 1$, on obtient :

$$e = \sum_{n=0}^{+\infty} \frac{1}{n!}$$

avec la majoration du reste : $R_n < 3/(n+1)!$
L'obtention de *d* décimales exactes nécessite une précision $\varepsilon < 10^{-d}$
On arrêtera le calcul de la somme dès que $1/n! \leq \varepsilon/3$ pour que $R_n < 3/(n+1)! \leq 3/n! \leq \varepsilon$
En utilisant les logarithmes, la condition sur *n* devient : $Log\ n! \geq Log3 + dLog10$

Le programme :

```
exec("Scilab-Arithmetique.sce",-1)

d = 200 ;
n = 1 ; Log = 0 ;
while Log < log(3) + d*log(10)
  n = n + 1 ;
  Log = Log + log(n) ;
end
e = FRACTION(0,1) ;
for i = 0:n
  u = list([1,1],[1,factorielle(i)]) ;
  e = F_ADDITION(e,u) ;
end
e = DECIMALE(e,d) ;
printf('\n e = '), ECRIRE(e)
```

permet de calculer 200 décimales de *e* :

```
e = 2.71828182845904523536028747135266249775724709369995
    95749669676277240766303535475945713821785251664274
    27466391932003059921817413596629043572900334295260
    59563073813232862794349076323382988075319525101901
```

Avec le programme précédent, chaque addition de 2 nombres rationnels conduit systématiquement à la recherche d'un PGCD afin d'obtenir une fraction irréductible, ce qui augmente le temps de calcul global. On peut remédier à cela en utilisant l'identité suivante :

$$\sum_{i=0}^{n} \frac{1}{i!} = \frac{1 + \sum_{i=0}^{n-1} \prod_{j=0}^{i} (n - i)}{n!}$$

D'où le programme amélioré pour calculer plus rapidement 1000 décimales de *e* :

```
d = 1000 ;
n = 1 ; Log = 0 ;
while Log < log(3) + d*log(10)
  n = n + 1 ;
  Log = Log + log(n) ;
end

Sum = 1 ;
Prod = 1 ;
for i = 0:n-1
  Prod = multiplication(Prod,entier(n-i)) ;
  Sum = addition(Sum,Prod) ;
end
e = list([1,Sum],[1,factorielle(n)]) ;

e = DECIMALE(e,d) ;
printf('\n e = '), ECRIRE(e)
```

On trouve :

```
e = 2.71828182845904523536028747135266249775724709369995
    95749669676277240766303535475945713821785251664274
    27466391932003059921817413596629043572900334295260
    59563073813232862794349076323382988075319525101901
    15738341879307021540891499348841675092447614606680
    82264800168477411853742345442437107539077744992069
    55170276183860626133138458300075204493382656029760
    67371132007093287091274437470472306969772093101416
    92836819025515108657463772111252389784425056953696
    77078544996996794686445490598793163688923009879312
    77361782154249992295763514822082698951936680331825
    28869398496465105820939239829488793320362509443117
    30123819706841614039701983767932068328237646480429
    53118023287825098194558153017567173613320698112509
    96181881593041690351598888519345807273866738589422
    87922849989208680582574927961048419844436346324496
    84875602336248270419786232090021609902353043699418
    49146314093431738143640546253152096183690888707016
    76839642437814059271456354906130310720851038375051
    01157477041718986106873969655212671546889570350354
```

Mathématique & Informatique

Calcul du nombre π

π est la première lettre du mot περιφέρεια qui signifie circonférence en grec.
Les premiers calculs du nombre π, retrouvés sur le papyrus Rhind, datent de 2000 ans avant Jésus Christ.
Il existe de nombreuses formules pour exprimer π. Parmi celles-ci :

- Des produits infinis :

$$\pi = 2\prod_{n=1}^{\infty}\frac{2}{\sqrt{2_{(1)}+\sqrt{2_{(2)}+\cdots+\sqrt{2_{(n)}}}}}\ \textbf{(Viète)} = 2\prod_{n=1}^{\infty}\frac{4n^2}{4n^2-1}\ \textbf{(Wallis)}$$

- Des sommes infinies :

$$\pi = 4\sum_{n=0}^{\infty}\frac{(-1)^n}{2n+1}\ \textbf{(Leibniz)} = 2\sum_{n=0}^{\infty}\frac{2^n\,(n!)^2}{(2n+1)!}\ \textbf{(Euler)} = \frac{9801}{2\sqrt{2}}\left[\sum_{n=0}^{\infty}\frac{(4n)!(1103+26390n)}{(n!)^4\,(396)^{4n}}\right]^{-1}\ \textbf{(Ramanujan)}$$

$$= \sum_{n=0}^{\infty}\frac{1}{16^n}\left(\frac{4}{8n+1}-\frac{2}{8n+4}-\frac{1}{8n+5}-\frac{1}{8n+6}\right)\ \textbf{(Plouffe)}$$

La dernière formule, la plus récente (1995), est surtout utilisée lorsqu'on souhaite connaître une décimale particulière de π.
Le nombre de décimales connues du nombre π est amélioré régulièrement (5000 milliards de décimales et 90 jours de calcul avec un ordinateur personnel en 2010). Consulter le site "L'univers de π" à l'adresse : **http://www.pi314.net/fr/index.php**

Procédé mnémotechnique pour retenir les premières décimales
π = 3.1415926535 = "Que j'aime à faire apprendre un nombre utile aux sages"
1/π = 0.3183098 = "Les 3 jours de 1830 ont renversé 89"

Utilisons la formule de **Machin** (1706) améliorée par **Hwang Chien-Lih** (2003) :

$$\frac{\pi}{4} = 183\text{Arctan}\frac{1}{239} + 32\text{Arctan}\frac{1}{1023} - 68\text{Arctan}\frac{1}{5832} + 12\text{Arctan}\frac{1}{113021}$$
$$- 100\text{Arctan}\frac{1}{6826318} - 12\text{Arctan}\frac{1}{33366019650} + 12\text{Arctan}\frac{1}{43599522992503626068}$$

Le but du calcul est de trouver un grand nombre rationnel qui approxime π avec une précision suffisante.
La fonction Arc tangente se développe en série entière de rayon de convergence 1 :

$$\text{Arctan}(x) = \sum_{n=0}^{+\infty}(-1)^n\frac{x^{2n+1}}{2n+1}\quad,\ |x|<1$$

Le nombre π est la somme de 7 séries de la forme $k\text{Arctan}(x)$. Pour obtenir d décimales exactes de π, on calcule chacune de ces séries avec une précision $\varepsilon < 10^{-d}/7$. Comme le développement de Arctan(x) est celui d'une série alternée dont le terme général u_n décroît en valeur absolue vers 0, le critère de **Leibniz** permet de majorer le reste d'ordre n :

$$|R_n| = |u_{n+1} + u_{n+2} + \ldots| \le |u_{n+1}| \le |u_n|$$

Dans l'approximation de π, on arrête la procédure de sommation dès que $|ku_n| \le \varepsilon$, donc pour le plus petit entier n tel que :

$$\left|k\,(-1)^n\frac{x^{2n+1}}{2n+1}\right| < \frac{10^{-d}}{7}$$

En prenant le logarithme de cette expression, on est ramené à résoudre une équation en n :

$$(2n+1)Log|x| - Log(2n+1) + Log|k| + dLog10 + Log7 = 0$$

Finalement, on attribue à n la partie entière supérieure de la valeur de la racine donnée par la fonction Scilab *fsolve*.
Nous écrirons 2 fonctions Scilab, une première qui calcule Arctan(x), et une seconde qui fournit le nombre n permettant d'atteindre la d^e décimale de π.

Le programme pour calculer *d* décimales de π est le suivant :

```
exec("Scilab-Arithmetique.sce",-1)
function Arctanx = Arctan(x,n)
  u = x ; Arctanx = u
  x2 = F_MULTIPLICATION(x,x)
  for i = 3:2:2*n+1
    u = F_MULTIPLICATION(u,F_MOINS(x2))
    v = F_DIVISION(u,FRACTION(i,1))
    Arctanx = F_ADDITION(Arctanx,v)
  end
endfunction
function n = precision(k,x,d)
  deff('y = f(n)','y = (2*n+1)*log(abs(x)) - log(2*n+1) + log(abs(k)) +   d*log(10) + log(7)')
  n = ceil(fsolve(0,f))
endfunction

d = 200 ;
n1 = precision(4*183,1/239,d) ;
n2 = precision(4*32,1/1023,d) ;
n3 = precision(-4*68,1/5832,d) ;
n4 = precision(4*12,1/113021,d) ;
n5 = precision(-4*100,1/6826318,d) ;
n6 = precision(-4*12,1/33366019650,d) ;
n7 = precision(4*12,1/43599522992503626068,d) ;
u1 = Arctan(list([1,1],[1,2,3,9]),n1) ;
u2 = Arctan(list([1,1],[1,1,0,2,3]),n2) ;
u3 = Arctan(list([1,1],[1,5,8,3,2]),n3) ;
u4 = Arctan(list([1,1],[1,1,1,3,0,2,1]),n4) ;
u5 = Arctan(list([1,1],[1,6,8,2,6,3,1,8]),n5) ;
u6 = Arctan(list([1,1],[1,3,3,3,6,6,0,1,9,6,5,0]),n6) ;
u7 = Arctan(list([1,1],[1,4,3,5,9,9,5,2,2,9,9,2,5,0,3,6,2,6,0,6,8]),n7) ;
v1 = F_MULTIPLICATION(u1,list([1,1,8,3],[1,1])) ;
v2 = F_MULTIPLICATION(u2,list([1,3,2],[1,1])) ;
v3 = F_MULTIPLICATION(u3,list([-1,6,8],[1,1])) ;
v4 = F_MULTIPLICATION(u4,list([1,1,2],[1,1])) ;
v5 = F_MULTIPLICATION(u5,list([-1,1,0,0],[1,1])) ;
v6 = F_MULTIPLICATION(u6,list([-1,1,2],[1,1])) ;
v7 = F_MULTIPLICATION(u7,list([1,1,2],[1,1])) ;
pi4 = FRACTION(0,1) ;
pi4 = F_ADDITION(pi4,v1) ;
pi4 = F_ADDITION(pi4,v2) ;
pi4 = F_ADDITION(pi4,v3) ;
pi4 = F_ADDITION(pi4,v4) ;
pi4 = F_ADDITION(pi4,v5) ;
pi4 = F_ADDITION(pi4,v6) ;
pi4 = F_ADDITION(pi4,v7) ;
pi = F_MULTIPLICATION(pi4,FRACTION(4,1)) ;
pi = DECIMALE(pi,d) ;
printf('\n pi = '), ECRIRE(pi)
```

On obtient le nombre π avec 200 décimales exactes :

π = 3.14159265358979323846264338327950288419716939937510
58209749445923078164062862089986280348253421170679
82148086513282306647093844609550582231725359408128
48111745028410270193852110555964462294895493038196

Les nombres de Bernoulli

Ces nombres rationnels B_n interviennent dans diverses formules mathématiques, notamment dans le développement en série entière de tan(x), dans la formule de sommation d'**Euler-MacLaurin**, et dans la fonction zêta de **Riemann**.
Les nombres de **Bernoulli** s'obtiennent en développant en série entière la fonction génératrice $x/(e^x - 1) = \Sigma\ (B_n/n!)\ x^n$.
A partir de $B_0 = 1$, on applique la relation de récurrence suivante :

$$B_n = -\frac{\sum_{k=0}^{n-1} C_{n+1}^{k} B_k}{n+1} \qquad (n = 1, 2, 3, 4, 5, \ldots)$$

De proche en proche :

$B_0 = 1$, $B_1 = -1/2$, $B_2 = 1/6$, $B_3 = 0$, $B_4 = -1/30$, $B_5 = 0$, $B_6 = 1/42$, $B_7 = 0$, $B_8 = -1/30$, $B_9 = 0$, $B_{10} = 5/66$, ...

On montre que les B_{2p+1} sont nuls pour $p > 0$.

Ecrivons un programme qui calcule et affiche les nombres de **Bernoulli** non nuls de B_0 à B_{100}

```
N = 100 ;
B0 = FRACTION(1,1) ;
B1 = FRACTION(-1,2) ;
if N >= 1 then, printf('\n B0 = '), ECRIRE(B0), end
if N >= 2 then, printf('\n B1 = '), ECRIRE(B1), end
if N <= 2 then, abort, end
B = list(B0,B1) ;
for n = 2:2:N
  S = FRACTION(0,1) ;
  for k = [0,1,2:2:n-2]
    CNK = binomiaux(n+1,k) ;
    T1 = list([1,CNK],[1,1]) ;
    if k > 1 then, T2 = B(k/2+2) ; else, T2 = B(k+1) ; end
    T3 = F_MULTIPLICATION(T1,T2) ;
    S = F_ADDITION(S,T3) ;
  end
  T = FRACTION(-1,n+1) ;
  B($+1) = F_MULTIPLICATION(S,T) ;
  printf('\n B%u = ',n), ECRIRE(B($))
end
```

Formule asymptotique

En utilisant l'approximation de **Stirling** pour les grandes factorielles, on établit que :

$$\left| B_{2p} \right| \sim 4\sqrt{\pi p}\left(\frac{p}{\pi e}\right)^{2p}$$

Limite : $B_{2p} \to \infty$ quand $p \to \infty$

Mathématique & Informatique

Nombres de Bernoulli non nuls de B0 à B100

B0 = 1
B1 = - 1 / 2
B2 = 1 / 6
B4 = - 1 / 30
B6 = 1 / 42
B8 = - 1 / 30
B10 = 5 / 66
B12 = - 691 / 2730
B14 = 7 / 6
B16 = - 3617 / 510
B18 = 43867 / 798
B20 = - 174611 / 330
B22 = 854513 / 138
B24 = - 236364091 / 2730
B26 = 8553103 / 6
B28 = - 23749461029 / 870
B30 = 8615841276005 / 14322
B32 = - 7709321041217 / 510
B34 = 2577687858367 / 6
B36 = - 26315271553053477373 / 1919190
B38 = 2929993913841559 / 6
B40 = - 261082718496449122051 / 13530
B42 = 1520097643918070802691 / 1806
B44 = - 27833269579301024235023 / 690
B46 = 596451111593912163277961 / 282
B48 = - 560940336899781768624912754 7 / 46410
B50 = 495057205241079648212477525 / 66
B52 = - 80116571813548995734792499 1853 / 1590
B54 = 291499636348848624214181238126 91 / 798
B56 = - 247939292931322675368541573966 3229 / 870
B58 = 844836133488800418620467759940 36021 / 354
B60 = - 12152331404837555720403049940798202460414 91 / 56786730
B62 = 123005854340868585419530398574033861 51 / 6
B64 = - 10678383014786652988638544497914264794201 7 / 510
B66 = 14726000221263356540516194285519323422418991 01 / 64722
B68 = - 787731308587187281419091492084746062443470 01 / 30
B70 = 150538134733336700380307656737785720851143816 0235 / 4686
B72 = - 58279549616699441104382772446410673652824883018442604 29 / 140100870
B74 = 34152417289221168014330073731472635186688307783087 / 6
B76 = - 24655088825935372707687196040585199904365267828865801 / 30
B78 = 41484636557540082829517903554954207349219937537240048 3487 / 3318
B80 = - 46037842994794576469355749690190468497942578727512889196 56867 / 230010
B82 = 16770141491851458368231545097862699002077360275702534148816 13 / 498
B84 = - 20245761959352903602311311601117310099899173911980908772810839324 77 / 3404310
B86 = 66071461941767865357384784742626149627783068665338893176199698 3 / 6
B88 = - 13114264886740175079955114240193118433457502755720286442969198905740 47 / 61410
B90 = 11790572790210827998841233512492150837752549496696471162315452157279225 35 / 272118
B92 = - 12955859482075375279894278285385767496593414837194351430233163268299462 47 / 1410
B94 = 12208138065797444696073016794132012039585084152026966214362151052846494 47 / 6
B96 = - 21160044959726651309759772810982423367304395438906023415063873342005066834998 7259 / 4501770
B98 = 67908260672905495624051117546403605607342195728504487509073961249992947058239 / 6
B100 = - 945980378191221252952274330694937218727028415330669361333856962043113954151972477 11 / 33330

Les nombres d'Euler

Les nombres d'**Euler** E_n sont des entiers naturels définis par la série :

$$\frac{1}{\cos x} = \sum_{n=0}^{+\infty} \frac{E_n}{n!} x^n$$

La fonction 1/cos x est paire, donc les E_{2p+1} sont nuls pour $p \geq 0$.
Partant de $E_0 = 1$, les E_{2p} ($p \geq 1$) s'obtiennent par récurrence :

$$E_{2p} = \sum_{k=1}^{p} (-1)^{k+1} C_{2p}^{2k} E_{2(p-k)}$$

Premières valeurs :

$E_0 = 1$, $E_1 = 0$, $E_2 = 1$, $E_3 = 0$, $E_4 = 5$, $E_5 = 0$, $E_6 = 61$, $E_7 = 0$, $E_8 = 1385$, $E_9 = 0$, $E_{10} = 50521$, ...

Le programme suivant calcule et affiche les nombres d'**Euler** non nuls de E_0 à E_{50}

```
N = 50 ;
E0 = 1 ;
if N >= 1 then, printf('\n E0 = '), ecrire(E0), end
if N <= 1 then, abort, end
E = list(E0) ;
for n = 2:2:N
  S = 0 ;
  for k = 1:n/2
    T = multiplication(E(n/2-k+1),binomiaux(n,2*k)) ;
    if modulo(k,2) <> 0 then
      S = addition(S,T) ;
    else
      S = soustraction(S,T) ;
    end
  end
  E($+1) = S ;
  printf('\n E%u = ',n), ecrire(E($))
end
```

Formule asymptotique

$$E_{2p} \sim 8\sqrt{\frac{p}{\pi}}\left(\frac{4p}{\pi e}\right)^{2p}$$

Limite : $E_{2p} \to \infty$ quand $p \to \infty$

Mathématique & Informatique

Nombres d'Euler non nuls de E0 à E50

E0 = 1
E2 = 1
E4 = 5
E6 = 61
E8 = 1385
E10 = 50521
E12 = 2702765
E14 = 199360981
E16 = 19391512145
E18 = 2404879675441
E20 = 370371188237525
E22 = 69348874393137901
E24 = 15514534163557086905
E26 = 4087072509293123892361
E28 = 1252259641403629865468285
E30 = 441543893249023104553682821
E32 = 177519391579539289436664789665
E34 = 80723299235887898062168247453281
E36 = 41222060339517702122347079671259045
E38 = 23489580527043108252017828576198947741
E40 = 14851150718114980017877156781405826684425
E42 = 10364622733519612119397957304745185976310201
E44 = 7947579422597592703608040510088070619519273805
E46 = 6667537516685544977435028474773748197524107684661
E48 = 6096278645568542158691685742876843153976539044435185
E50 = 6053285248188621896314383785111649088103498225146815121

Sécante et sécante hyperbolique

$$\sec(x) = \frac{1}{\cos x} = \sum_{p=0}^{+\infty} \frac{E_{2p}}{(2p)!} x^{2p} \quad , \ |x| < \frac{\pi}{2}$$

$$\operatorname{sech}(x) = \frac{1}{\operatorname{ch} x} = \sum_{p=0}^{+\infty} (-1)^p \frac{E_{2p}}{(2p)!} x^{2p} \quad , \ |x| < \frac{\pi}{2}$$

Calcul de la constante d'Euler γ

La constante d'**Euler** apparaît dans la définition de **Weierstrass** de la fonction gamma, dans l'expression des fonctions de **Bessel**, ou encore dans la transformée de **Laplace** de la fonction logarithme.
γ est elle irrationnelle comme *e* ou π ? Cette question n'est pas résolue.

La constante d'**Euler** est définie par :

$$\gamma = \lim_{n \to +\infty} \left(\sum_{i=1}^{n} \frac{1}{i} - Log\, n \right)$$

A partir de la double inégalité $\frac{1}{n+1} < \int_n^{n+1} \frac{dx}{x} < \frac{1}{n}$ on établit que la suite $u_n = \sum_{i=1}^{n} \frac{1}{i} - Log\, n$ est décroissante et minorée par 0, donc convergente. Il existe plusieurs algorithmes pour calculer $\gamma = \lim_{n \to +\infty} u_n$; à ce sujet consulter le document internet :

⇨ **http://www-fourier.ujf-grenoble.fr/~demailly/manuscripts/gamma_gazmath.pdf**

Jusqu'à quelques centaines de décimales, on peut utiliser la formule d'**Euler-MacLaurin**. On montre que :

$$\gamma = 1 + \frac{1}{2} + \cdots + \frac{1}{n} - Log\, n - \frac{1}{2n} + \frac{B_2}{2\, n^2} + \cdots + \frac{B_{2k}}{2k\, n^{2k}} + Rk$$

avec la majoration : $|Rk| \le \frac{4}{n} \sqrt{\frac{k}{\pi}} \left(\frac{k}{n \pi e} \right)^k$

La somme 1 + 1/2 + ... + 1/*n* - 1/2*n* s'obtient sans difficulté.
Les nombres de **Bernoulli** B_{2p} , $1 \le p \le k$, sont issus du programme précédent.
Il reste à approximer Log *n*.
Pour calculer Log *x*, *x* > *0*, on écrit *x* sous la forme : $x = 2^q y$ avec *q* entier et $\frac{1}{2} \le y \le 1$
Introduisons la fonction Argth(*x*) = $\frac{1}{2}$ Log [(1+*x*) / (1-*x*)]
Il vient : Log *x* = 2*q* Argth($\frac{1}{3}$) - 2 Argth[(1-*y*) / (1+*y*)]
Dans le cas présent : $x = n = 2^q$ et *y* = 1, d'où : Log *n* = 2*q* Argth($\frac{1}{3}$)
Le développement en série entière de Argth(*x*) est :

$$\text{Argth}(x) = \sum_{m=0}^{+\infty} \frac{x^{2m+1}}{2m+1} \quad , \ |x| < 1$$

Son reste R_m d'ordre *m* vérifie :

$$|Rm| \le \frac{|x|^{2m+3}}{2m+3} \frac{1}{1-x^2}$$

Remarque
Le logarithme népérien de *x* s'écrit habituellement Log *x* (avec un L majuscule) ou ln *x* ou encore $\log_e x$.
Le logarithme décimal de *x* s'écrit log *x* (avec un l minuscule) ou lg *x* ou encore $\log_{10} x$.
Attention : Scilab utilise la notation log(*x*) pour désigner le logarithme népérien de *x* (log(%e) = 1).

Mathématique & Informatique

Précision des calculs

Tout d'abord, proposons l'approximation $\gamma \approx \sum_{i=1}^{i=n} \frac{1}{i} - Log\ n$, conduisant au programme suivant pour n = 1 million :

```
n = 1e6 ;
S = 0 ;
for i = 1:n
  S = S + 1/i ;
end
GAMMA = S - log(n)
```

Avec l'incertitude standard ε = 2.220446 × 10^{-16} dans les calculs (Norme IEEE-754), le résultat final est :

GAMMA = 0.5772162

L'erreur apparaît à partir de la 6e décimale. Le manque de précision est dû à la lente convergence de la suite utilisée.

Appliquons maintenant la méthode précédente pour obtenir d = 100 décimales exactes de γ.
Avec q = 9, on a $n = 2^q$ = 512 et Log $n = 2q$ Argth($\frac{1}{3}$) = 18 Argth($\frac{1}{3}$)
Pour atteindre la précision souhaitée, $\varepsilon = |R_k| + 2q|R_m| < 10^{-d}$, on prend :

- $|R_k| < 10^{-d} / 2 = 10^{-100} / 2$
- $|R_m| < 10^{-d} / 4q = 10^{-100} / 36$

Ce qui conduit aux inéquations :

$$\frac{4}{512}\sqrt{\frac{k}{\pi}}\left(\frac{k}{512\pi e}\right)^k < \frac{10^{-100}}{2} \qquad \text{et} \qquad \frac{9}{8}\frac{(1/3)^{2m+3}}{2m+3} < \frac{10^{-100}}{36}$$

En passant aux logarithmes et en utilisant la fonction Scilab *fsolve*, on trouve les valeurs minimales de k et m :

- k = 52
- m = 103

Algorithme

La constante γ est approximée par l'addition de 3 sommes : $\gamma \approx S1 + S2 + S3$

$$S1 = \sum_{i=1}^{n-1} \frac{1}{i} + \frac{1}{2n} \qquad (S1 \text{ est la valeur exacte de } 1 + \frac{1}{2} + \cdots + \frac{1}{n} - \frac{1}{2n})$$

$$S2 = -2q \sum_{i=0}^{m} \frac{(1/3)^{2i+1}}{2i+1} \qquad (S2 \text{ est la valeur approchée de } -Log\ n)$$

$$S3 = \sum_{i=1}^{k} \frac{B_{2i}}{2i\ n^{2i}} \qquad (S3 \text{ est la valeur exacte de } \frac{B_2}{2\ n^2} + \cdots + \frac{B_{2k}}{2k\ n^{2k}})$$

Mathématique & Informatique

Programme

```
exec("Scilab-Arithmetique.sce",-1)

// k et m

d = 100 , q = 9 , n = 2^q
deff('y = f(k)','y = log(k)/2 + k*log(k) - k*log(n*%pi*%e) + log(4/n) - log(%pi)/2 + d*log(10) + log(2)') ;
k = ceil(fsolve(1,f))
deff('y = f(m)','y = - (2*m+3)*log(3) - log(2*m+3) + log(9/8) + d*log(10) + log(4*q)') ;
m = ceil(fsolve(1,f))

// S1

S1 = FRACTION(0,1) ;
for i = 1:n-1
  printf('\n S1 > %i', i)
  S1 = F_ADDITION(S1,FRACTION(1,i)) ;
end
S1 = F_ADDITION(S1,FRACTION(1,2*n)) ;
printf('\n'), ECRIRE(S1), printf('\n')

// S2

function Argthx = Argth(x,n)
  u = x ; Argthx = u
  x2 = F_MULTIPLICATION(x,x)
  for i = 3:2:2*n+1
    printf('\n S2 > %i',i)
    u = F_MULTIPLICATION(u,x2)
    v = F_DIVISION(u,FRACTION(i,1))
    Argthx = F_ADDITION(Argthx,v)
  end
endfunction

Argthx = Argth(FRACTION(1,3),m) ;
S2 = F_MULTIPLICATION(Argthx,FRACTION(-2*q,1)) ;
printf('\n'), ECRIRE(S2), printf('\n')

// S3

function C = binomiaux(n,k)
  A = 1 ; B = 1
  for i = 1:k
    A = multiplication(A,entier(n-i+1))
    B = multiplication(B,entier(i))
  end
  [C,r] = division(A,B)
endfunction
```

```
K = ENTIER(1) ;
n2 = ENTIER(n^2) ;
S3 = FRACTION(0,1) ;
B0 = FRACTION(1,1) ;
B1 = FRACTION(-1,2) ;
B = list(B0,B1) ;
for i = 2:2:2*k
  printf('\n S3 > %i',i)
  S = FRACTION(0,1) ;
  for j = [0,1,2:2:i-2]
    CNK = binomiaux(i+1,j) ;
    T1 = list([1,CNK],[1,1]) ;
    if j > 1 then, T2 = B(j/2+2) ; else, T2 = B(j+1) ; end
    T3 = F_MULTIPLICATION(T1,T2) ;
    S = F_ADDITION(S,T3) ;
  end
  T = FRACTION(-1,i+1) ;
  B($+1) = F_MULTIPLICATION(S,T) ;
  K = MULTIPLICATION(K,n2) ;
  T = MULTIPLICATION(K,ENTIER(i)) ;
  T = list(T,[1,1]) ;
  T = F_DIVISION(B($),T) ;
  S3 = F_ADDITION(S3,T) ;
end
printf('\n'), ECRIRE(S3), printf('\n')

// GAMMA

GAMMA = FRACTION(0,1) ;
GAMMA = F_ADDITION(GAMMA,S1) ;
GAMMA = F_ADDITION(GAMMA,S2) ;
GAMMA = F_ADDITION(GAMMA,S3) ;
GAMMA = DECIMALE(GAMMA,d) ;
printf('\n GAMMA = '), ECRIRE(GAMMA)
```

Résultat

Les 100 premières décimales de γ :

γ = 0.57721566490153286060651209008240243104215933593992
359880576723488486772677766467093694706329174674 95

Mathématique & Informatique

Calcul des constantes e, π, γ

(Mercredi 30 Avril 2009)

Interpréteur Scilab — Windows Vista — Intel Pentium 2.8 Ghz

---

e = 2.71828182845904523536028747135266249775724709369995
957496696762772407663035354759457138217852516642 74
274663919320030599218174135966290435729003342952 60
595630738132328627943490763233829880753195251019 01

----- Horloge = 10:5:8
----- Horloge = 10:28:34

π = 3.14159265358979323846264338327950288419716939937510
582097494459230781640628620899862803482534211706 79
821480865132823066470938446095505822317253594081 28
481117450284102701938521105559644622948954930381 96

----- Horloge = 10:37:55
----- Horloge = 12:3:54

γ = 0.57721566490153286060651209008240243104215933593992
359880576723488486772677766467093694706329174674 95

----- Horloge = 12:7:40
----- Horloge = 14:24:2

Mathématique & Informatique

Racine carrée entière

Pour rechercher si un grand nombre entier n est un carré parfait, on utilise la fonction *intsqrt* définie ci-dessous.
Elle renvoie la partie entière (inférieure) m de $\sqrt{n}$, c'est-à-dire le plus grand entier m tel que : $n = m^2 + r$.
Si r = 0, alors n est un carré parfait et $m = \sqrt{n}$. L'algorithme procède par dichotomie sur l'intervalle $[0,n] \subset \mathbb{N}$.

```
function m = intsqrt(n)
  inf = 0 ; sup = n
  c = 0
  while c <= 0
    x = addition(inf,sup)
    [x,r] = division(x,2)
    x2 = multiplication(x,x)
    c = comparaison(x2,n)
    select c
      case  0, m = x ; return
      case -1, inf = addition(x,1)
      case +1, sup = soustraction(x,1)
    end
    c = comparaison(inf,sup)
  end
  m = sup
endfunction
```

EXEMPLE

Vérifier que le nombre n suivant :

152415787745770472325864959531473860922679469587501905 21

est un carré parfait.

On écrit le programme :

```
n = [1,5,2,4,1,5,7,8,7,7,4,5,7,7,0,4,7,2,3,2,5,8,6,4,9,5,9,5,3,1,4,7,3,8,6,0,9,2,2,2,6,7,9,4,6,9,5,8,7,5,0,1,9,0,5,2,1] ;
m = intsqrt(n) ;
m2 = multiplication(m,m) ;
if comparaison(m2,n) == 0 then, ecrire(m), end
```

La racine carrée de n s'affiche à la Console :

12345678909876543210123456789

Racine cubique entière

Pour rechercher la racine cubique entière d'un grand nombre entier n, on définit la fonction *intcbrt* suivante :

```
function m = intcbrt(n)
  inf = 0 ; sup = n
  c = 0
  while c <= 0
    x = addition(inf,sup)
    [x,r] = division(x,2)
    x3 = puissance(x,3)
    c = comparaison(x3,n)
    select c
      case  0, m = x ; return
      case -1, inf = addition(x,1)
      case +1, sup = soustraction(x,1)
    end
    c = comparaison(inf,sup)
  end
  m = sup
endfunction
```

On suppose n ≥ 0, sinon : $\sqrt[3]{n} = -\sqrt[3]{-n}$
Si n n'est pas un cube parfait, alors *incbrt(n)* fournit la partie entière de la racine cubique de n.

EXEMPLE

Pour calculer : $\sqrt[3]{533323083404365176437867855481345297887424226300447182 63}$

On exécute l'instruction :

ecrire(intcbrt([5,3,3,3,2,3,0,8,3,4,0,4,3,6,5,1,7,6,4,3,7,8,6,7,8,5,5,4,8,1,3,4,5,2,9,7,8,8,7,4,2,4,2,2,6,3,0,0,4,4,7,1,8,2,6 ,3]))

On obtient :

3764120001488502567

A noter que l'instruction Scilab :

printf('%.0f',533323083404365176437867855481345297887424226300447182 63^(1/3))

donne pour résultat : 3764120001488493600

soit une erreur de 8967 unités.

Mathématique & Informatique

Racine carrée

La méthode de **Héron d'Alexandrie** est basée sur la suite récurrente :

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{n}{x_k} \right)$$

En prenant $x_0 > 0$, cette suite converge vers $\sqrt{n}$. La convergence est d'autant plus rapide que x_0 est proche de $\sqrt{n}$ et le nombre de chiffres exacts double approximativement à chaque itération. Si n et x_0 sont rationnels, alors x_k l'est aussi, donc le processus fournit un grand nombre rationnel A/B tel que : $| \sqrt{n} - A/B | < \varepsilon$

```
function R = RACINE_CARREE(n,d)
  A = intsqrt(n)
  B = 1
  Maxiter = ceil(log(d+1)/log(2) + 1)
  for i = 1:Maxiter
    t1 = multiplication(A,A)
    t2 = multiplication(B,B)
    t3 = multiplication(A,B)
    t4 = multiplication(t2,n)
    t5 = addition(t1,t4)
    t6 = multiplication(t3,2)
    p = pgcd(t5,t6)
    [A,r] = division(t5,p)
    [B,r] = division(t6,p)
  end
  R = list([1,A],[1,B])
endfunction
```

Remarque
En augmentant la valeur de d et par conséquent celle de la constante Maxiter, on accroît la précision du calcul de $\sqrt{n}$
Le processus de **Héron** est une application de la méthode des tangentes de **Newton** à la fonction $f(x) = x - n$
D'une manière générale, la suite $x_{k+1} = x_k - f(x_k)/f'(x_k)$ converge vers la racine de l'équation $f(x) = 0$

EXEMPLE

Calculer le nombre d'or $\varphi = \frac{1 + \sqrt{5}}{2}$ avec 100 décimales :

```
RC5 = RACINE_CARREE(5,100) ;
N = F_ADDITION(RC5,FRACTION(1,1)) ;
PHI = F_DIVISION(N,FRACTION(2,1)) ;
ECRIRE(DECIMALE(PHI,100))
```

Résultat :

φ = 1.6180339887498948482045868343656381177203091798057628621354486227052604628189024497072072041893911374

Equation de Fermat & Grands nombres

Pour résoudre $x^2 - n y^2 = 1$, il est généralement nécessaire développer $\sqrt{n}$ en fraction continue (*).
En utilisant les fonctions Scilab standards, la solution ne peut être obtenue que pour de petites valeurs de x, y, n.
Ecrivons une fonction *FERMAT* avec les "entiers vectoriels" définis précédemment.

```
function [x,y] = FERMAT(n)
  m = intsqrt(n)
  m2= multiplication(m,m)
  if comparaison(m2,n) == 0 then
    x = 1 ; y = 0
    return
  end
  fc = list()
  a = 0 ; b = 1 ; q = 0
  p = multiplication(m,2)
  while ~isequal(q,p)
    t = addition(m,a)
    [q,r] = division(t,b)
    t1 = soustraction(m,r)
    t2 = multiplication(t1,t1)
    t3 = soustraction(n,t2)
    [t4,r] = division(t3,b)
    a = t1
    b = t4
    fc($+1) = q
  end
  T = length(fc) - 1
  for k = 1:T-1
    fc($+1) = fc(k+1)
  end
  if modulo(T,2) == 0 then
    K = T-1
  else
    K = 2*T-1
  end
  x = 1 ; y = 0
  for k = K+1:-1:1
    z = x
    x = multiplication(fc(k),x)
    x = addition(x,y)
    y = z
  end
endfunction
```

(*) Pour certaines valeurs de n, par exemple si n est un carré ±1 ou un carré ±2, la solution de $x^2 - n y^2 = 1$ est simple :

$n = m^2 - 1 \Rightarrow x = m, y = 1$
$n = m^2 + 1 \Rightarrow x = 2m^2 + 1, y = 2m$
$n = m^2 - 2 \Rightarrow x = m^2 - 1, y = m$
$n = m^2 + 2 \Rightarrow x = m^2 + 1, y = m$

Attention
La résolution de l'équation de **Fermat** conduit à de très grands nombres entiers et le temps de calcul peut être long.

Mathématique & Informatique

EXEMPLE 1

Résoudre l'équation de **Fermat** : $x^2 - 7759\ y^2 = 1$

On écrit :

```
exec("Scilab-Arithmetique.sce",-1)
n = [7,7,5,9] ;
[x1,y1] = FERMAT(n) ;
printf('\n x1 = '), ecrire(x1)
printf('\n y1 = '), ecrire(y1)
```

La solution fondamentale de l'équation est :

x1 = 401969627429723362885915137834215741335301239867281965946941563858548560

y1 = 456341917969557457152007954028214327683826238049662123076689761481343 1

Les 2 solutions suivantes s'obtiennent par :

```
x = x1 ; y = y1 ;
for i = 2:3
  a = multiplication(x,x1) ;
  b = multiplication(x,y1) ;
  c = multiplication(y,x1) ;
  d = multiplication(y,y1) ;
  e = multiplication(d,n) ;
  x = addition(a,e) ;
  y = addition(b,c) ;
  printf('\n x%i = ',i), ecrire(x)
  printf('\n y%i = ',i), ecrire(y)
end
```

Soit :

x2 = 323159162751981217939072143255452973005259696175702468218534219291211061
493004790107327672607859780062591464405719919384628800745408379756147199

y2 = 366871181493576784044295553757634263428818934934477467071984360701005202
978380512486736936060832185313087066525830100928932989479631770741872 0

x3 = 259800336503830451702305346991857976143206839667943522833657198053826912
931653338956637206625257498824380634606162136606267505419645071558878437
959497401168582548350410675274355987298936447716811861114469167840418320

y3 = 294942144279350961270987531173321982912255864834280477074427737975542366
688537403426403177092284207210450198772247147282238756000363218241801061
058751459771626819441158610844490058960460566037295771576550913127296 9

EXEMPLE 2

Un nombre carré triangulaire t, est un nombre qui est à la fois un carré parfait ($t = m^2$) et un nombre triangulaire ($t = n (n + 1)/2$, somme des n premiers entiers). Il est donc de la forme : $t = m = n (n + 1)/2$.
En posant $k = 2n + 1$, on obtient l'équation de **Fermat** :

$$k^2 - 8 m^2 = 1$$

Le programme suivant calcule les 30 premiers nombres carrés triangulaires :

```
[k1,m1] = FERMAT(8) ;
k = 1 ; m = 0 ;
for i =1:30
  a = multiplication(k,k1) ;
  b = multiplication(k,m1) ;
  c = multiplication(m,k1) ;
  d = multiplication(m,m1) ;
  e = multiplication(d,8) ;
  k = addition(a,e) ;
  m = addition(b,c) ;
  t = multiplication(m,m) ;
  printf('\n t%i = ',i), ecrire(t)
end
```



Résultat :

```
t1 = 1
t2 = 36
t3 = 1225
t4 = 41616
t5 = 1413721
t6 = 48024900
t7 = 1631432881
t8 = 55420693056
t9 = 1882672131025
t10 = 63955431761796
t11 = 2172602007770041
t12 = 73804512832419600
t13 = 2507180834294496361
t14 = 85170343853180456676
t15 = 2893284510173841030625
t16 = 98286503002057414584576
t17 = 3338847817559778254844961
t18 = 113422539294030403250144100
t19 = 3853027488179473932250054441
t20 = 130889512058808083293251706896
t21 = 4446390382511295358038307980025
t22 = 151046383493325234090009219613956
t23 = 5131130648390546663702275158894481
t24 = 174307395661785261331787346182798400
t25 = 5921320321852308338617067495056251121
t26 = 201150583547316698251648507485729739716
t27 = 683319852028691543221743218701975489922 5
t28 = 2321275991062078079971410458511859368339 36
t29 = 7885505171090778556470578126753302097454601
t30 = 267875048217980263112002515263761085376622500
```

Mathématique & Informatique

Les "Bœufs du Soleil"

Ce célèbre problème, attribué à **Archimède**, revient à résoudre l'équation de **Fermat** :

$$X^2 - N\,Y^2 = 1 \qquad \text{avec} \quad N = 410\,286\,423\,278\,424$$

Le nombre N n'est pas premier, il se décompose en :

2 puissance 3
3 puissance 1
7 puissance 1
11 puissance 1
29 puissance 1
353 puissance 1
4657 puissance 2

Par suite, on peut ramener l'équation de départ à :

$$x^2 - n\,y^2 = 1 \qquad \text{avec} \quad n = 4\,729\,494$$

La solution de cette dernière équation s'obtient plus facilement (la fraction continue de $\sqrt{n}$ a une période de 92) :

$$x_1 = 109931986732829734979866232821433543901088049$$
$$y_1 = 50549485234315033074477819735540408986340$$

On montre que la solution initiale (X,Y) se déduit de (x_1,y_1) par les relations :

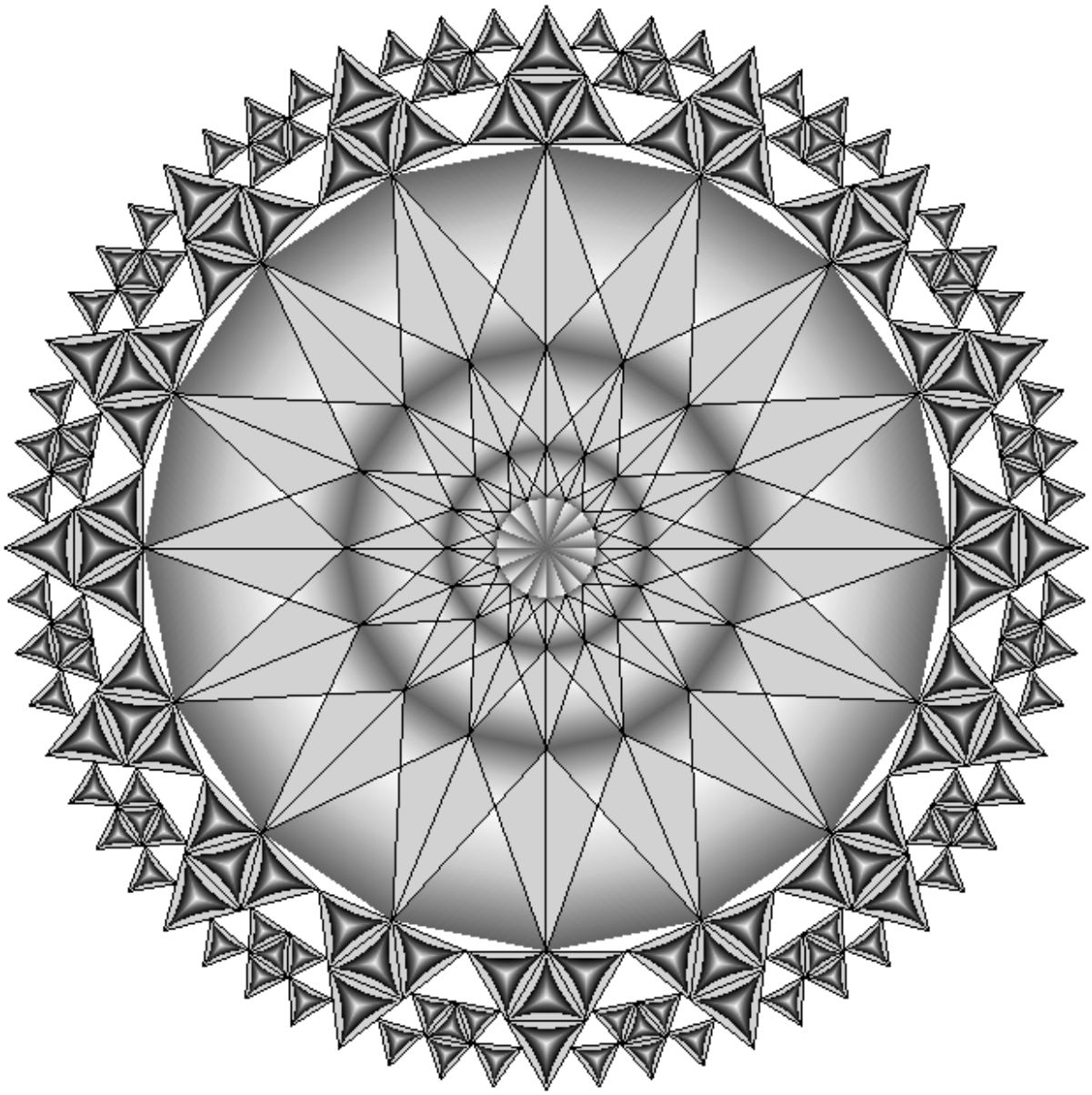
$$X = x_{2329}\ ,\ Y = y_{2329} / 9314$$

$$\text{avec} \quad x_{2329} + y_{2329}\sqrt{n} = (x_1 + y_1\sqrt{n})^{2329}$$

En appliquant le schéma récurrent qui permet de calculer la solution d'ordre i = 2329 d'une équation de **Fermat** à partir de la solution fondamentale (x_1,y_1), on obtient le résultat recherché.
Les nombres entiers utilisés pour résoudre le problème d'**Archimède** ont plus de 200 000 chiffres !

Voir la solution détaillée par Lazare Georges **Vidiani**, à l'adresse internet :
http://www.dma.ens.fr/culturemath/maths/pdf/nombres/BetailC.pdf

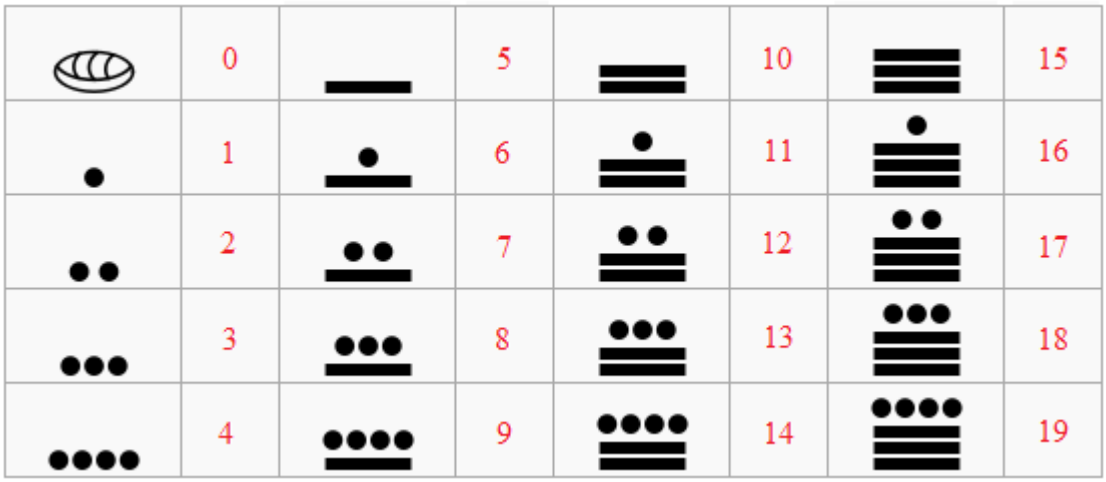
Mathématique & Informatique



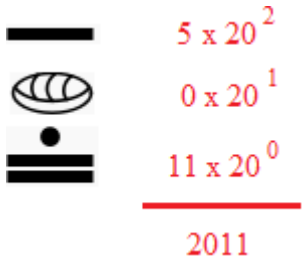
Mathématique & Informatique

Un peu d'histoire du monde

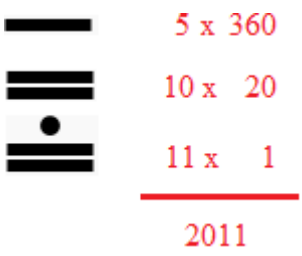
L'idée d'un système de numération est très ancienne.
La civilisation précolombienne maya, dont les origines remontent au III^{e} millénaire avant notre ère, utilisait une écriture composée de glyphes (dessins souvent gravés dans la pierre) comparables aux hiéroglyphes égyptiens. Le code maya comporte environ 800 glyphes. De nombreux chercheurs se sont intéressés à ce graphisme complexe et c'est récemment que les textes mayas ont pu être presque entièrement déchiffrés, notamment en étudiant les rares codex non détruits après la conquête espagnole du Yucatán au XVI^{e} siècle.
Les Mayas faisaient beaucoup de calculs relatifs à l'astronomie et avaient recours à un système de numération de base 20 appelé système vicésimal positionnel. L'unité était représentée par un point et cinq unités par une barre. Le zéro était connu et associé à un glyphe particulier. En combinant ces symboles, on obtient tous les nombres entiers de 0 à 19 :



Pour constituer un grand nombre, le scribe maya superpose plusieurs chiffres compris entre 0 et 19.
Par exemple, en base 20, la représentation du nombre décimal 2011 = $5\times20^2 + 0\times20^1 + 11\times20^0$ est la suivante :



En réalité, la numération maya n'est pas exactement un système vicésimal car on a remarqué, dans le calcul des dates, que le troisième chiffre n'est pas multiplié par $20^2 = 400$ mais par 360. Hormis cette irrégularité au rang 3, on retrouve le facteur 20 aux rangs supérieurs, ce qui conduit à la suite de multiplicateurs 1, 20, 360, 7200, 144000, … au lieu de 1, 20, 400, 8000, 160000, … Dans ces conditions : 2011 = 5×360 + 10×20 + 11×1.



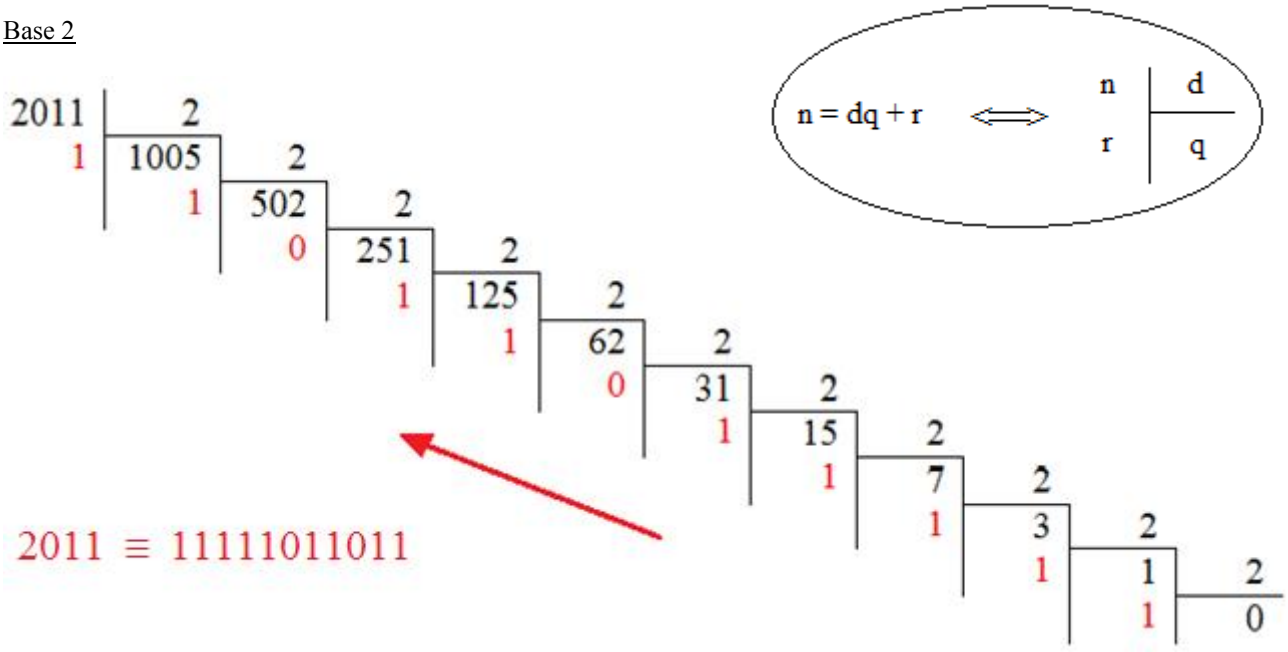
Pour plus de précisions sur le calendrier maya, consulter la page Web à l'adresse **http://www.louisg.net/C_maya.htm**
Voir aussi un extrait du documentaire "Breaking the Maya Code" **http://www.youtube.com/watch?v=qhWItvjk9Yg**

Numération factorielle

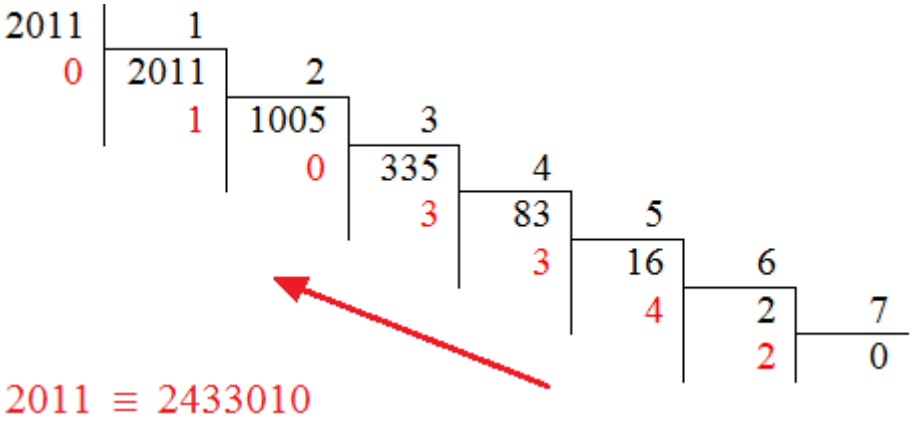
Introduction

Dans un système de numération classique, de base N, les nombres sont représentés par des coefficients entiers qui multiplient des puissances de N. Le système le plus couramment utilisé est le système décimal, de base 10. C'est également un des plus anciens : apparu en Chine et en Inde, puis adopté par les mathématiciens arabes, il est introduit en Occident aux alentours de l'an mille. En informatique, le système fondamental est le système binaire, de base 2.
Ce dernier est particulièrement adapté au fonctionnement des calculateurs électroniques : sommairement le chiffre "1" et le chiffre "0" correspondent à 2 niveaux logiques, c'est-à-dire à 2 tensions électriques différentes.
Définissons un autre système de numération, de base factorielle, dans lequel les nombres sont représentés par des coefficients entiers qui multiplient des factorielles ($n! = 1 \times 2 \times 3 \times 4 \times \ldots \times n$).
Afin de montrer la différence entre l'algorithme de décomposition en base N et l'algorithme de décomposition en base factorielle, effectuons les calculs "à la main" sur des petits nombres.
Prenons l'exemple de l'entier 2011 en base 10 que l'on souhaite écrire en base 2 et en base factorielle.
Disposons les divisions successives de la manière suivante :

Base 2



Base factorielle



Le nombre 2011 en base 10 s'écrit 11111011011 en base 2 et 2433010 en base factorielle.
En base N, le diviseur d est constant : d = N (par exemple, en base 2, d = 2)
En base factorielle, le diviseur d croît de 1 à chaque division : d = 1, 2, 3, 4, 5, …
Les restes r successifs, jusqu'à ce que le quotient q soit nul, donnent l'expression du nombre de départ dans la nouvelle base : le premier reste correspond au chiffre de poids faible (à droite) et le dernier reste au chiffre de poids fort (à gauche).

Mathématique & Informatique

Base factorielle

Soit N un entier supérieur ou égal à 2.

Rappelons que tout nombre rationnel $\frac{a}{b}$ (≥ 0) se met, d'une seule façon, sous la forme :

$$\frac{a}{b} = x_q N^q + \ldots + x_2 N^2 + x_1 N^1 + x_0 N^0 + \frac{x_{-1}}{N^1} + \frac{x_{-2}}{N^2} + \ldots + \frac{x_{-p}}{N^p}$$

où les coefficients $x_{\pm i}$ sont des entiers naturels compris entre 0 et $N-1$.

Lorsque le développement est fini, la partie fractionnaire de $\frac{a}{b}$ est un rationnel N-adique.

Lorsqu'il est infini ($p \to +\infty$), la série est convergente et la suite des x_{-i} est périodique à partir d'un certain rang.

Notons $\left[x_q, \ldots, x_2, x_1, x_0 ; x_{-1}, x_{-2}, \ldots, x_p\right]$ la représentation de $\frac{a}{b}$ en base N.

Dans le vecteur des $x_{\pm i}$ la partie entière de $\frac{a}{b}$ est séparée de la partie fractionnaire par un point-virgule.

Généralement : $N = 2$ (représentation binaire), $N = 10$ (représentation décimale) ou $N = 16$ (représentation hexadécimale)

Par analogie, définissons un système de numération factoriel dans lequel le rationnel $\frac{a}{b}$ s'exprime, de manière unique, par :

$$\frac{a}{b} = n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \ldots + \frac{n_{-p}}{p!}$$

où les coefficients n_i sont des entiers naturels compris entre 0 et i et les n_{-i} entre 0 et $i-1$.

On notera $\left[n_q, \ldots, n_2, n_1, n_0 ; n_{-1}, n_{-2}, \ldots, n_p\right]$ la représentation de $\frac{a}{b}$ en base factorielle.

- Propriété :

Le développement d'une fraction en base factorielle est toujours fini, ce qui n'est pas le cas en base N.

La partie fractionnaire de la représentation factorielle de $\frac{a}{b}$ contient au plus b éléments.

Par exemple, $\frac{4}{3} = \left[1,0;0,0,2\right]$ en base factorielle alors que $\frac{4}{3} = \left[1;3,3,3,3,3,\ldots\right]$ en base 10.

- Corollaire :

Si b est un nombre premier, le développement factoriel de la fraction irréductible $\frac{a}{b}$ se termine par $\frac{b - a \bmod b}{b!}$

Il s'ensuit que b n'est pas premier si la partie fractionnaire de $\frac{a}{b}$ a moins de b éléments ($p < b$).

- Application :

Utilisons la base factorielle pour représenter des grands entiers.
En base factorielle, l'entier $n!$ s'écrit avec un 1 suivi de n zéros : $n! = 10 \ldots 0$; donc $n!$ a $n+1$ chiffres.

En base 10, la formule de **Stirling** permet d'établir que l'entier $n!$ possède environ $E(n[\ln(n) - 1]/\ln(10))$ chiffres.

Symbolisons les valeurs 0, 1, 2, … , 9, 10, 11, … , 33, 34, 35 par les caractères 0, 1, 2, … , 9, A, B, … , X, Y, Z

Soit le nombre n = ZYXWVUTSRQPONMLKJIHGFEDCBA9876543210 en base factorielle.

On a : $n = 35.35! + 34.34! + 33.33! + 32.32! + \ldots + 3.3! + 2.2! + 1.1! + 0.0! = 36! - 1$

L'instruction *soustraction(factorielle(36),1)* donne n = 371993326789901217467999448150835199999999 en base 10

En numération décimale n a 42 chiffres (symbolisés par les caractères 0 à 9)

En numération factorielle n a 36 "chiffres" (symbolisés par les caractères 0 à Z)

Mathématique & Informatique

Preuve de l'existence et de l'unicité du développement factoriel fini d'un nombre rationnel

$$\frac{a}{b} = n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \ldots + \frac{n_{-p}}{p!} \qquad (0 \le n_i \le i\,,\, 0 \le n_{-i} \le i\text{ -}1)$$

Soit $\frac{a}{b}$ une fraction irréductible positive.
Effectuons la division euclidienne de a par b : $a = b\,n + m$
Les entiers n ($n \ge 0$) et m ($0 \le m < b$) sont respectivement le quotient et le reste de la division entière de a par b.
Faisons apparaître le rapport $\frac{a}{b}$: $\frac{a}{b} = n + \frac{m}{b}$
La quantité n est la partie entière de $\frac{a}{b}$ et la quantité $\frac{m}{b}$ est la partie fractionnaire de $\frac{a}{b}$

- Partie entière n

Notons x_1 le quotient de la division euclidienne de n par 2 et n_1 son reste : $n = 2\,x_1 + n_1$ avec $0 \le n_1 \le 1$
On a : $2\,x_1 \le n \Rightarrow x_1 < n$
Notons x_2 le quotient de la division euclidienne de x_1 par 3 et n_2 son reste : $x_1 = 3\,x_2 + n_2$ avec $0 \le n_2 \le 2$
On a : $3\,x_2 \le x_1 \Rightarrow x_2 < x_1 < n$
On déduit : $n = 2\,x_1 + n_1 = 2\,(3\,x_2 + n_2) + n_1 = 2\,3\,x_2 + 2\,n_2 + n_1$
Ce qui s'écrit : $n = x_2\, 3! + n_2\, 2! + n_1\, 1! + n_0\, 0!$ avec $n_0 = 0$
En poursuivant ainsi, on obtient une suite strictement décroissante de termes x_i : $x_i < x_{i-1} < \ldots < x_2 < x_1 < n$
Donc, on finira nécessairement par trouver un $x_q = 0$ et un $n_q \le q$ tels que :

$$n = x_q\,(q+1)! + n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! = n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0!$$

L'unicité de la décomposition est due à l'unicité des quotients et des restes lors des divisions entières successives.

- Partie fractionnaire $\frac{m}{b}$

Admettons que le développement soit possible sous la forme :

$$\frac{m}{b} = \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \frac{n_{-3}}{3!} + \frac{n_{-4}}{4!} + \ldots + \frac{n_{-p}}{p!}$$

La condition $m < b$, c'est-à-dire $\frac{m}{b} < 1$, implique $n_{-1} = 0$. D'où :

$$\frac{2m}{b} - n_{-2} = \frac{n_{-3}}{3} + \frac{n_{-4}}{3.4} + \ldots + \frac{n_{-p}}{3.4 \ldots (p\text{ -}1)\,p}$$

Notons s_{-2} le second membre de cette égalité. Comme $n_{-i} \le i$ -1 (par hypothèse), la fraction s_{-2} est inférieure à :

$$\frac{2}{3} + \frac{3}{3.4} + \ldots + \frac{p\text{ -}1}{3.4 \ldots (p\text{ -}1)\,p}$$

Il est facile de montrer par récurrence sur p que cette somme vaut $1 - \frac{2!}{p!}$
On a : $\frac{2m}{b} - n_{-2} = s_{-2} \Rightarrow 2\,m = b\,n_{-2} + b\,s_{-2}$
Posons $y_{-2} = b\,s_{-2} \in \mathbb{N}$, il vient : $2\,m = b\,n_{-2} + y_{-2}$ avec $y_{-2} \le b\left(1 - \frac{2!}{p!}\right) < b$
On voit que n_{-2} et y_{-2} sont nécessairement le quotient et le reste de la division entière de $2\,m$ par b
On vérifie que : $n_{-2} \le \frac{2\,m}{b} < 2 \Rightarrow 0 \le n_{-2} \le 1$
Connaissant n_{-2} et y_{-2}, on déduit :

$$\frac{2m}{b} - n_{-2} = \frac{y_{-2}}{b} = \frac{n_{-3}}{3} + \frac{n_{-4}}{3.4} + \ldots + \frac{n_{-p}}{3.4 \ldots (p\text{ -}1)\,p}$$

Le même raisonnement permet d'établir que n_{-3} est le quotient de $3\,y_{-2}$ par b
La suite des divisions est finie et se termine au plus tard pour $p = b$
Dans chaque division le quotient et le reste sont déterminés de façon unique, donc la décomposition obtenue est unique.

Mathématique & Informatique

La profondeur d'un développement factoriel

$$\frac{a}{b} = n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \ldots + \frac{n_{-p}}{p!} \qquad (0 \le n_i \le i\ ,\ 0 \le n_{-i} \le i-1)$$

NB Si $\frac{a}{b}$ est supérieur à 1 alors $n_q \neq 0$, si $\frac{a}{b}$ n'est pas un entier alors $n_{-p} \neq 0$

La valeur de p sera appelée profondeur du développement factoriel de la fraction irréductible $\frac{a}{b}$.
En réduisant au même dénominateur le second membre de l'égalité, il existe un entier A tel que :

$$\frac{a}{b} = \frac{A}{p!} \quad \Leftrightarrow \quad a\,p! = Ab$$

L'entier b se décompose en facteurs premiers p_i :

$$b = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k} \quad \text{avec} \quad p_1 < p_2 < \ldots < p_k$$

Reportons dans la relation précédente :

$$a\,p! = A\,p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k}$$

L'entier a n'est divisible par aucun des p_i puisque $\frac{a}{b}$ est irréductible. Il s'ensuit que p_i divise $p!$ pour tout i.
Avec $p_k = \underset{i}{Max}\ p_i$, on a :

$$p_k! = 1 \times 2 \times \ldots \times p_i \times \ldots \times (p_k - 1) \times p_k = (p_k - 1)!\, p_k$$

La plus petite factorielle divisible par p_k est $p_k!$. En effet, le nombre premier p_k n'apparaît pas dans la décomposition en facteurs premiers d'un nombre qui lui est inférieur, donc p_k ne figure jamais dans la décomposition en facteurs premiers d'un produit de nombres qui lui sont inférieurs, ce qui signifie qu'aucun produit de nombres entiers inférieurs au nombre premier p_k n'est un multiple de p_k. Par conséquent, pour que p_k divise $p!$, il faut que : $p \ge p_k$
La profondeur p du développement factoriel de la fraction $\frac{a}{b}$ est au moins égale au plus grand diviseur premier p_k de b.

Cas particulier b premier

Si b est un nombre premier, la profondeur p du développement factoriel de $\frac{a}{b}$ est supérieure ou égale à b mais ne peut pas dépasser b d'après l'algorithme de calcul des coefficients n_{-i}. Donc $p = b$.
Ainsi, pour tout nombre premier b, la fraction $\frac{a}{b}$ admet un développement factoriel de profondeur b :

$$\frac{a}{b} = n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \ldots + \frac{n_{-b}}{b!}$$

Montrons que l'on peut préciser la valeur du coefficient n_{-b} dans ce cas.

Effectuons la division euclidienne de a par b : $\frac{a}{b} = \frac{nb+m}{b} = n + \frac{m}{b} \quad \Rightarrow \quad n = E\left(\frac{a}{b}\right)$ et $m = a \bmod b$

Le rationnel $\frac{a}{b}$ a pour partie entière n et pour partie fractionnaire $\frac{m}{b}$

Par suite : $\frac{m}{b} = \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \ldots + \frac{n_{-(b-1)}}{(b-1)!} + \frac{n_{-b}}{b!}$

Additionnons les termes $\frac{n_{-i}}{i!}$ de $\frac{n_{-1}}{1!}$ à $\frac{n_{-(b-1)}}{(b-1)!}$: $\sum_{i=1}^{b-1} \frac{n_{-i}}{i!} = \frac{\lambda}{(b-1)!}$

L'identité devient :

$$\frac{m}{b} = \frac{\lambda}{(b-1)!} + \frac{n_{-b}}{b!}$$

Multiplions par $b!$:

$$m(b-1)! = \lambda b + n_{-b} \qquad \text{avec } 1 \le m \le b-1 \text{ et } 1 \le n_{-b} \le b-1$$

Sous cette forme, il apparaît que n_{-b} est le reste de la division euclidienne de $m(b-1)!$ par b
Exprimons le théorème de **Wilson** :

$$p \text{ premier} \quad \Leftrightarrow \quad (p-1)! + 1 \equiv 0 \quad (p)$$

On déduit qu'il existe un entier μ tel que :

$$\begin{aligned} (b-1)! + 1 &= \mu b \\ m(b-1)! &= m\mu b - m \\ m(b-1)! &= (m\mu - 1)b + (b-m) \end{aligned}$$

Comme $1 \le b-m \le b-1$ et que le reste de la division euclidienne de $m(b-1)!$ par b est unique, on a nécessairement :

$$n_{-b} = b - m = b - a \bmod b$$

En conclusion, si b est un nombre premier, le développement factoriel de $\frac{a}{b}$ a une profondeur égale à b et s'écrit :

$$\frac{a}{b} = n_q q! + n_{(q-1)}(q-1)! + \cdots + n_2 2! + n_1 1! + n_0 0! + \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \ldots + \frac{n_{-(b-1)}}{(b-1)!} + \frac{b - a \bmod b}{b!}$$

Par exemple, le rationnel $\frac{96}{13}$ admet un développement factoriel de profondeur 13 puisque 13 est un nombre premier.
Sa partie fractionnaire se décompose en 13 termes $\frac{n_{-i}}{i!}$ avec pour dernier terme $\frac{13 - 96 \bmod 13}{13!} = \frac{13-5}{13!} = \frac{8}{13!}$
En appliquant l'algorithme de calcul des $n_{\pm i}$, on trouve :

$$\frac{96}{13} = 1.3! + 0.2! + 1.1! + 0.0! + \frac{0}{1!} + \frac{0}{2!} + \frac{2}{3!} + \frac{1}{4!} + \frac{1}{5!} + \frac{0}{6!} + \frac{6}{7!} + \frac{3}{8!} + \frac{6}{9!} + \frac{2}{10!} + \frac{3}{11!} + \frac{4}{12!} + \frac{8}{13!}$$

Autre exemple, 7919 est un nombre premier, donc le développement factoriel de $\frac{1}{7919}$ a une profondeur égale à 7919 et se termine par $\frac{7919 - 1 \bmod 7919}{7919!} = \frac{7918}{7919!}$

$$\frac{1}{7919} = \frac{0}{1!} + \frac{0}{2!} + \frac{0}{3!} + \frac{0}{4!} + \frac{0}{5!} + \frac{0}{6!} + \frac{0}{7!} + \frac{5}{8!} + \frac{0}{9!} + \frac{8}{10!} + \frac{2}{11!} + \ldots + \frac{7585}{7915!} + \frac{1319}{7916!} + \frac{3958}{7917!} + \frac{0}{7918!} + \frac{7918}{7919!}$$

A noter le caractère infinitésimal du dernier terme $\frac{7918}{7919!} \approx 1{,}438 \times 10^{-27433}$

Cas particulier $a = 1$ et b premier

Recherchons l'expression générale du développement factoriel de la fraction $1/p$ lorsque p est un nombre premier.
Tout d'abord, pour un nombre premier p et un entier non nul k inférieur ou égal à p, établissons la congruence :

$$(k-1)!\,(p-k)! + (-1)^{k-1} \equiv 0 \quad (p)$$

La démonstration se fait par récurrence à partir du théorème de **Wilson** : $(p-1)! + 1 \equiv 0 \quad (p)$

- Pour $k = 1$

  $$0!(p-1)! + (-1)^0 = (p-1)! + 1 \equiv 0 \quad (p)$$

- Supposons la formule vraie pour un entier k

  $$(k-1)!\,(p-k)! + (-1)^{k-1} \equiv 0 \quad (p)$$
  $$(k-1)!\,(p-k-1)!\,(p-k) + (-1)^{k-1} \equiv 0 \quad (p)$$
  $$(k-1)!\,(p-k-1)!\,p - (k-1)!\,(p-k-1)!\,k + (-1)^{k-1} \equiv 0 \quad (p)$$
  $$(k-1)!\,k\,(p-k-1)! - (-1)^{k-1} \equiv 0 \quad (p)$$
  $$k!\,(p-k-1)! + (-1)^k \equiv 0 \quad (p)$$
  $$((k+1)-1)!\,(p-(k+1))! + (-1)^{(k+1)-1} \equiv 0 \quad (p)$$

  La formule est vraie pour $k + 1$

Ainsi :

$$(p-1)! + 1 \equiv 0 \quad (p)$$
$$(p-2)! - 1 \equiv 0 \quad (p)$$
$$2(p-3)! + 1 \equiv 0 \quad (p)$$
$$6(p-4)! - 1 \equiv 0 \quad (p)$$

Si p est un nombre premier, la fraction $\frac{1}{p}$ admet un développement factoriel de profondeur p :

$$\frac{1}{p} = \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \frac{n_{-3}}{3!} + \ldots + \frac{n_{-(p-3)}}{(p-3)!} + \frac{n_{-(p-2)}}{(p-2)!} + \frac{n_{-(p-1)}}{(p-1)!} + \frac{n_{-p}}{p!} \qquad (0 \le n_{-i} \le i-1)$$

En utilisant la formule de **Wilson** modifiée, on obtient successivement la valeur des coefficients n_{-p}, $n_{-(p-1)}$, $n_{-(p-2)}$

$$n_{-p} = p - 1$$
$$n_{-(p-1)} = 0$$
$$n_{-(p-2)} = \frac{p-3}{2}$$

Ajoutons que n_{-1} est nul et que n_{-2} est égal à la partie entière de la fraction $2/p$

$$n_{-1} = 0$$
$$n_{-2} = \mathrm{E}\left(\frac{2}{p}\right)$$

Finalement, le développement factoriel de $\frac{1}{p}$ (p premier) s'écrit :

$$\frac{1}{p} = \frac{0}{1!} + \frac{\mathrm{E}\left(\frac{2}{p}\right)}{2!} + \ldots + \frac{\frac{p-3}{2}}{(p-2)!} + \frac{0}{(p-1)!} + \frac{p-1}{p!}$$

Mathématique & Informatique

Formule

Tout nombre rationnel $\frac{a}{b}$ (≥ 0) se met, de manière unique, sous la forme :

$$\frac{a}{b} = n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \ldots + \frac{n_{-p}}{p!}$$

où les coefficients n_i sont des entiers naturels compris entre 0 et i et les n_{-i} entre 0 et $i - 1$

Introduisons la notation : $(-n)! = \frac{1}{n!}$ pour désigner la factorielle d'un entier négatif $-n$

Par exemple : $(-3)! = \frac{1}{3!} = \frac{1}{6}$, à ne pas confondre avec : $-3! = -(3)! = -6$

Le développement factoriel de $\frac{a}{b}$ est défini par :

$$\frac{a}{b} = \sum_{k=-p}^{k=+q} n_k\,(k)!$$

Algorithme et programmation

On a : $\frac{a}{b} = n + \frac{m}{b}$

L'entier n est la partie entière de $\frac{a}{b}$

La fraction $\frac{m}{b}$ est la partie fractionnaire de $\frac{a}{b}$

- Partie entière

$$n = \sum_{i=0}^{q} n_i\, i! \qquad\qquad n_i \in \{0, 1, 2, \ldots, i\}$$

L'algorithme de calcul des n_i est le suivant :

```
i = 1
tantque  n ≠ 0
|    i = i + 1
|    n = i x + n_{i-1}
|    n = x
fin
```

NB : on a toujours $n_o = 0$ et $n_q \neq 0$

L'identité $n = i\,x + n_{i-1}$ signifie que x et n_{i-1} sont le quotient et le reste de la division euclidienne de n par i.

On déduit la fonction Scilab *factoriel1* qui fournit le vecteur *V1* des n_i :

```
function V1 = factoriel1(n)
    V1 = 0
    i = 1
    while ~nul(n)
        i = i + 1
        [x,ni] = division(n,entier(i))
        V1 = [nombre_naturel(ni),V1]
        n = x
    end
endfunction
```

Remarque : Le vecteur *V1* n'est jamais vide puisque *a/b* possède toujours une partie entière (éventuellement nulle)

- Partie fractionnaire

$$\frac{m}{b} = \sum_{i=1}^{p} \frac{n_i}{i!} \qquad n_i \in \{0, 1, 2, \ldots, i-1\}$$

L'algorithme de calcul des n_i est le suivant :

```
i = 1
tantque  m ≠ 0
|    i = i + 1
|    m = b n_{i+1} + y
|    m = i y
fin
```

NB : on a toujours $n_1 = 0$ et $n_p \neq 0$

L'identité $m = b\, n_{i+1} + y$ signifie que n_{i+1} et y sont le quotient et le reste de la division euclidienne de m par b.

On déduit la fonction Scilab *factoriel2* qui fournit le vecteur *V2* des n_i :

```
function V2 = factoriel2(m,b)
    V2 = []
    i = 1
    while ~nul(m)
        i = i + 1
        [ni,y] = division(m,b)
        V2 = [V2,nombre_naturel(ni)]
        m = multiplication(y,entier(i))
    end
endfunction
```

Remarque : Le vecteur *V2* est égal à l'ensemble vide si *a/b* n'a pas de partie fractionnaire, c'est-à-dire si *a/b* est un entier.

Regroupons les fonctions Scilab *factoriel1* et *factoriel2* pour obtenir une unique fonction *base_factorielle* qui fournit le développement factoriel d'un nombre rationnel positif quelconque $\frac{a}{b}$, sous la forme d'une liste *f* constituée de 2 vecteurs *V1* et *V2* représentant respectivement la partie entière et la partie fractionnaire de $\frac{a}{b}$:

```
function f = base_factorielle(a,b)
    function V1 = factoriel1(n)
        V1 = 0
        i = 1
        while ~nul(n)
            i = i + 1
            [x,ni] = division(n,entier(i))
            V1 = [nombre_naturel(ni),V1]
            n = x
        end
    endfunction
    function V2 = factoriel2(m,b)
        V2 = []
        i = 1
        while ~nul(m)
            i = i + 1
            [ni,y] = division(m,b)
            V2 = [V2,nombre_naturel(ni)]
            m = multiplication(y,entier(i))
        end
    endfunction
    [n,m] = division(a,b)
    V1 = factoriel1(n)
    V2 = factoriel2(m,b)
    f = list(V1,V2)
endfunction
```

Donc :

$$\frac{a}{b} = \sum_{k=-p}^{k=+q} n_k\,(k)! \quad \Leftrightarrow \quad f = \text{list}(V_1, V_2) = \text{list}\left(\left[n_q, \ldots, n_2, n_1, n_0\right], \left[n_{-1}, n_{-2}, \ldots, n_{-p}\right]\right)$$

L'algorithme de décomposition d'un nombre rationnel dans la base factorielle calcule les n_k $(-p \le k \le q)$ dans l'ordre croissant de l'indice *k* lorsque $k \ge 0$, c'est-à-dire $n_0, n_1, n_2, \ldots, n_q$, puis dans l'ordre décroissant de l'indice *k* lorsque $k < 0$, c'est-à-dire $n_{-1}, n_{-2}, \ldots, n_{-p}$

Profondeur

La profondeur du développement factoriel *f* de la fraction $\frac{a}{b}$ est l'entier non nul *p* défini de la manière suivante :

- $p = 1$ lorsque $\frac{a}{b}$ est un entier $\left(\frac{a}{b} = n_q.q! + \cdots + n_1.1! + 0.0! + \frac{0}{1!}\right)$
- $p > 1$ lorsque $\frac{a}{b}$ n'est pas un entier $\left(\frac{a}{b} = n_q.q! + \cdots + n_1.1! + 0.0! + \frac{0}{1!} + \cdots + \frac{n_{-p}}{p!} \;\; avec \;\; n_{-p} \neq 0\right)$

La fonction *profondeur* retourne la valeur de *p* :

```
function p = profondeur(f)
    p = max(1,length(f(2)))
endfunction
```

Mathématique & Informatique

Graphique de la fonction factorielle

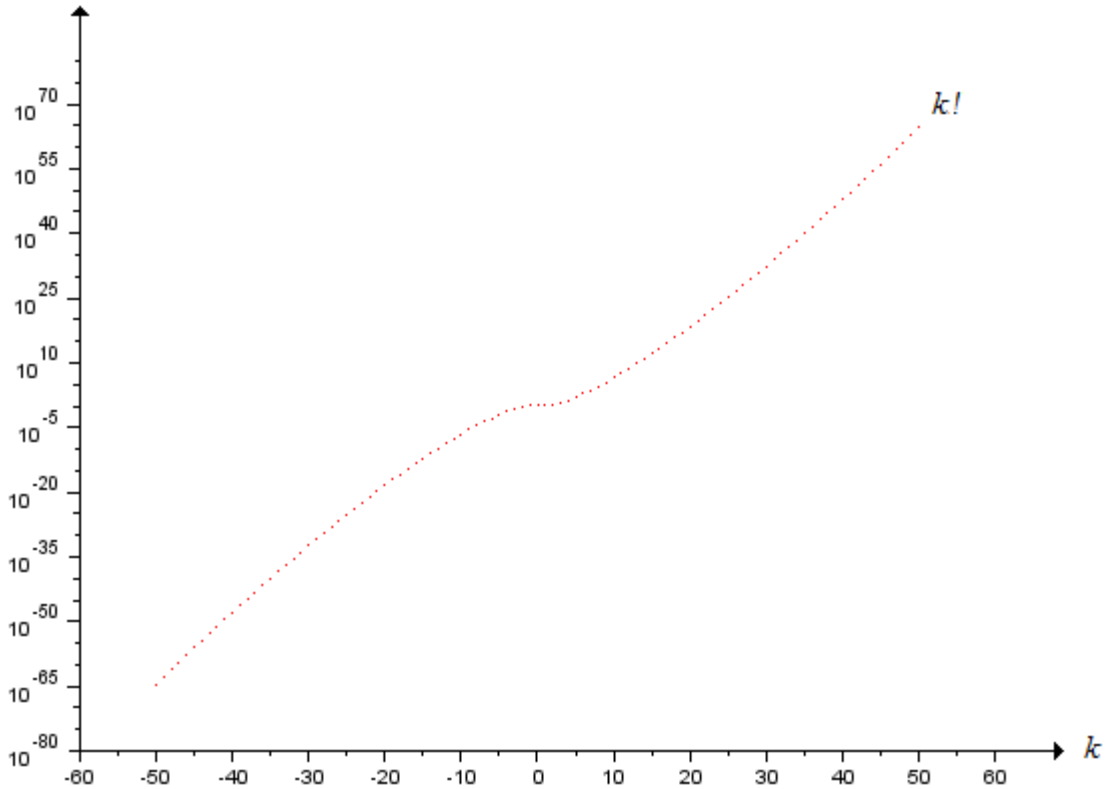
Par définition :

$$\forall\, k \in \mathbb{Z} \;:\quad \begin{cases} k! = 1 \times 2 \times \cdots \times k & \text{si } k > 0 \\ k! = \dfrac{1}{1 \times 2 \times \cdots \times (-k)} & \text{si } k < 0 \\ k! = 1 & \text{si } k = 0 \end{cases}$$

Un repère orthonormé O*xy* habituel est mal adapté pour représenter la courbe de la fonction *f(k) = k!* en raison des très grandes valeurs et des très petites valeurs prises par la factorielle. Pour améliorer la lisibilité du graphique, on adopte une échelle logarithmique sur l'axe des ordonnées et une échelle linéaire sur l'axe des abscisses, ce qui revient à reporter *Log k!* sur l'axe des *y* et *k* sur l'axe des *x*.
La documentation Scilab concernant les propriétés d'une courbe s'affiche en saisissant *help polyline_properties* en réponse au prompt de la Console et celle concernant les propriétés des axes en saisissant *help axes_properties*.

```
function f = factorielle(k)
    f = factorial(abs(k))
    if k < 0 then, f = 1/f, end
endfunction

// Tracé de la courbe
drawlater()
for k = -50:+50
    plot2d(k,factorielle(k))
    e = gce() ;
    e.children.mark_style = 0 ;
    e.children.mark_foreground = 5 ;
end
a = gca() ;
a.log_flags = "nl" ;
drawnow()
```



Ecriture dans la base factorielle

Programmons 2 fonctions qui affichent la représentation factorielle de *a*/*b* à partir de la liste *f* précédemment définie :

```
function ecrire1(f)
    V1 = f(1)
    V2 = f(2)
    l1 = length(V1)
    l2 = length(V2)
    lg = 10
    k = 0
    for i = 1:l1
        k = k + 1
        if modulo(k-1,lg) == 0 then, printf('\n'), end
        if k == 1 then, printf('      '), else, printf('  +   '), end
        printf('%i*%i!',V1(i),l1-i)
    end
    for j = 1:l2
        k = k + 1
        if modulo(k-1,lg) == 0 then, printf('\n'), end
        if k == 1 then, printf('      '), else, printf('  +   '), end
        printf('%i/%i!',V2(j),j)
    end
endfunction


function ecrire2(f)
    V1 = f(1)
    V2 = f(2)
    l1 = length(V1)
    l2 = length(V2)
    printf('[')
    for i = 1:l1
        if i > 1 then, printf(','), end
        printf('%i',V1(i))
    end
    if l2 <> 0 then, printf(';'), end
    for j = 1:l2
        if j > 1 then, printf(','), end
        printf('%i',V2(j))
    end
    printf(']')
endfunction
```

Programme de test

Pour des nombres entiers *a* et *b* de moins de 16 chiffres :

```
while %t
    a = input('a = ?') ;
    if a == [] then, break, end
    b = input('b = ?') ;
    if b == [] then, break, end
    a = entier (a) ;
    b = entier(b) ;
    f = base_factorielle(a,b) ;
    printf('\n Développement factoriel de '), ecrire(a), printf('/'), ecrire(b)
    ecrire1(f), printf('\n   =   '), ecrire2(f)
end
```

Par exemple :

$0 = 0*0! = [0]$

$1 = 1*1! + 0*0! = [1,0]$

$2 = 1*2! + 0*1! + 0*0! = [1,0,0]$

$3 = 1*2! + 1*1! + 0*0! = [1,1,0]$

$2011 = 2*6! + 4*5! + 3*4! + 3*3! + 0*2! + 1*1! + 0*0! = [2,4,3,3,0,1,0]$

$15120 = 3*7! + 0*6! + 0*5! + 0*4! + 0*3! + 0*2! + 0*1! + 0*0! = [3,0,0,0,0,0,0,0]$

$123456789 = 3*11! + 1*10! + 0*9! + 1*8! + 7*7! + 2*6! + 4*5! + 2*4! + 3*3! + 1*2! + 1*1! + 0*0! = [3,1,0,1,7,2,4,2,3,1,1,0]$

$\frac{1}{2} = 0*0! + 0/1! + 1/2! = [0;0,1]$

$\frac{1}{3} = 0*0! + 0/1! + 0/2! + 2/3! = [0;0,0,2]$

$\frac{1}{4} = 0*0! + 0/1! + 0/2! + 1/3! + 2/4! = [0;0,0,1,2]$

$\frac{1}{720} = 0*0! + 0/1! + 0/2! + 0/3! + 0/4! + 0/5! + 1/6! = [0;0,0,0,0,0,1]$

$\frac{7}{11} = 0*0! + 0/1! + 1/2! + 0/3! + 3/4! + 1/5! + 2/6! + 1/7! + 2/8! + 1/9! + 6/10! + 4/11! = [0;0,1,0,3,1,2,1,2,1,6,4]$

$\frac{13}{5} = 1*2! + 0*1! + 0*0! + 0/1! + 1/2! + 0/3! + 2/4! + 2/5! = [1,0,0;0,1,0,2,2]$

$\frac{121}{120} = 1*1! + 0*0! + 0/1! + 0/2! + 0/3! + 0/4! + 1/5! = [1,0;0,0,0,0,1]$

$\frac{112011}{24} = 6*6! + 2*5! + 4*4! + 1*3! + 2*2! + 1*1! + 0*0! + 0/1! + 0/2! + 0/3! + 3/4! = [6,2,4,1,2,1,0;0,0,0,3]$

Exemple

Effectuons le développement factoriel du nombre rationnel suivant :

$$\frac{5221204737007144853572582912189632359737 7483}{114632590922446012416}$$

On écrit :

```
exec("Scilab-Arithmetique.sce",-1)
a = [5,2,2,1,2,0,4,7,3,7,0,0,7,1,4,4,8,5,3,5,7,2,5,8,2,9,1,2,1,8,9,6,3,2,3,5,9,7,3,7,7,4,8,3] ;
b = [1,1,4,6,3,2,5,9,0,9,2,2,4,4,6,0,1,2,4,1,6] ;
f = base_factorielle(a,b) ;
printf('\n Développement factoriel de '), ecrire(a), printf('/'), ecrire(b)
ecrire1(f), printf('\n   =   '), ecrire2(f)
```

Résultat :

$$\frac{5221204737007144853572582912189632359737 7483}{114632590922446012416}$$

| |

17*23! + 14*22! + 4*21! + 19*20! + 17*19! + 3*18! + 15*17! + 7*16! + 1*15! + 7*14!
+ 9*13! + 0*12! + 10*11! + 6*10! + 9*9! + 1*8! + 0*7! + 2*6! + 1*5! + 2*4!
+ 0*3! + 2*2! + 1*1! + 0*0! + 0/1! + 0/2! + 2/3! + 2/4! + 0/5! + 0/6!
+ 0/7! + 7/8! + 5/9! + 5/10! + 5/11! + 9/12! + 3/13! + 6/14! + 6/15! + 6/16!
+ 0/17! + 17/18! + 1/19! + 7/20! + 18/21! + 19/22! + 6/23! + 6/24! + 17/25! + 18/26!
+ 16/27! + 16/28! + 24/29! + 28/30! + 29/31! + 6/32! + 31/33! + 7/34! + 22/35! + 5/36!
+ 5/37! + 10/38! + 33/39! + 17/40! + 5/41! + 36/42!

| |

[17,14,4,1 9,17,3,15, 7,1,7,9,0, 10,6,9,1,0 ,2,1,2,0,2 ,1,0;
0,0,2,2,0, 0,0,7,5,5, 5,9,3,6,6, 6,0,17,1,7 ,18,19,6,6 ,17,18,16, 16,24,28,2 9,6,31,7,2 2,5,5,10,3 3,17,5,36]

Mathématique & Informatique

Cas des nombres $n!$, $\frac{1}{n!}$, $n!-1$, $1-\frac{1}{n!}$

Les nombres $n!$, $\frac{1}{n!}$, $n!-1$, $1-\frac{1}{n!}$ ont des développements factoriels caractéristiques.

- $n!$

$$n! = 1.n! + \sum_{i=0}^{n-1} 0.i! \quad \Leftrightarrow \quad n! = [1,0,0,\ldots,0,0,0]$$

- $\frac{1}{n!}$

$$\frac{1}{n!} = \sum_{i=1}^{n-1} \frac{0}{i!} + \frac{1}{n!} \quad \Leftrightarrow \quad \frac{1}{n!} = [0;0,0,0,\ldots,0,0,1]$$

- $n!-1$

On vérifie par récurrence sur n que :

$$n!-1 = \sum_{i=0}^{n-1} i.i! \quad \Leftrightarrow \quad n!-1 = [n-1, n-2, n-3, \ldots, 2, 1, 0]$$

- $1-\frac{1}{n!}$

On vérifie par récurrence sur n que :

$$1-\frac{1}{n!} = \sum_{i=1}^{n} \frac{i-1}{i!} \quad \Leftrightarrow \quad 1-\frac{1}{n!} = [0;0,1,2,\ldots,n-3,n-2,n-1]$$

Par exemple :

$$10! = 3628800 = 1.10! + 0.9! + 0.8! + 0.7! + 0.6! + 0.5! + 0.4! + 0.3! + 0.2! + 0.1! + 0.0! = [1,0,0,0,0,0,0,0,0,0,0]$$

$$\frac{1}{10!} = \frac{1}{3628800} = \frac{0}{1!} + \frac{0}{2!} + \frac{0}{3!} + \frac{0}{4!} + \frac{0}{5!} + \frac{0}{6!} + \frac{0}{7!} + \frac{0}{8!} + \frac{0}{9!} + \frac{1}{10!} = [0;0,0,0,0,0,0,0,0,0,1]$$

$$10! - 1 = 3628799 = 9.9! + 8.8! + 7.7! + 6.6! + 5.5! + 4.4! + 3.3! + 2.2! + 1.1! + 0.0! = [9,8,7,6,5,4,3,2,1,0]$$

$$1 - \frac{1}{10!} = \frac{3628799}{3628800} = \frac{0}{1!} + \frac{1}{2!} + \frac{2}{3!} + \frac{3}{4!} + \frac{4}{5!} + \frac{5}{6!} + \frac{6}{7!} + \frac{7}{8!} + \frac{8}{9!} + \frac{9}{10!} = [0;0,1,2,3,4,5,6,7,8,9]$$

Cas des coefficients binomiaux

Pour $m \le n$, on a :

$$C_n^m = \frac{n!}{m!(n-m)!} = n!(-m)!(m-n)!$$

Les coefficients binomiaux sont tous des nombres entiers.
Leur développement dans la base factorielle est fini et ne comporte pas de partie fractionnaire :

$$C_n^m = \sum_{k=0}^{k=+q} n_k\,(k)!$$

Par exemple, pour décomposer C_{1000}^{500}, on écrit :

```
a = binomiaux(1000,500) ;
b = 1 ;
f = base_factorielle(a,b) ;
printf('\n C(1000,500) = '), ecrire1(f)
```

On obtient :

C_{1000}^{500} = 30*166! + 3*165! + 44*164! + 154*163! + 1*162! + 2*161! + 67*160! + 5*159!
+ 124*158! + 83*157! + 2*156! + 61*155! + 111*154! + 120*153! + 27*152! + 1*151!
+ 70*150! + 93*149! + 123*148! + 20*147! + 91*146! + 107*145! + 64*144! + 37*143!
+ 42*142! + 54*141! + 134*140! + 53*139! + 23*138! + 137*137! + 109*136! + 27*135!
+ 98*134! + 30*133! + 8*132! + 69*131! + 111*130! + 65*129! + 30*128! + 40*127!
+ 45*126! + 87*125! + 54*124! + 117*123! + 59*122! + 95*121! + 0*120! + 57*119!
+ 8*118! + 37*117! + 69*116! + 15*115! + 11*114! + 71*113! + 47*112! + 81*111!
+ 34*110! + 1*109! + 37*108! + 61*107! + 100*106! + 65*105! + 12*104! + 13*103!
+ 23*102! + 80*101! + 64*100! + 14*99! + 3*98! + 58*97! + 39*96! + 47*95!
+ 50*94! + 69*93! + 20*92! + 55*91! + 26*90! + 27*89! + 86*88! + 54*87!
+ 77*86! + 35*85! + 2*84! + 52*83! + 75*82! + 59*81! + 52*80! + 47*79!
+ 58*78! + 53*77! + 72*76! + 49*75! + 30*74! + 29*73! + 56*72! + 63*71!
+ 69*70! + 45*69! + 41*68! + 36*67! + 61*66! + 12*65! + 7*64! + 9*63!
+ 5*62! + 30*61! + 18*60! + 36*59! + 40*58! + 49*57! + 37*56! + 9*55!
+ 44*54! + 30*53! + 45*52! + 50*51! + 47*50! + 42*49! + 44*48! + 6*47!
+ 37*46! + 1*45! + 5*44! + 33*43! + 16*42! + 34*41! + 0*40! + 18*39!
+ 0*38! + 7*37! + 36*36! + 12*35! + 18*34! + 19*33! + 1*32! + 11*31!
+ 11*30! + 15*29! + 11*28! + 0*27! + 5*26! + 22*25! + 6*24! + 10*23!
+ 0*22! + 8*21! + 8*20! + 10*19! + 17*18! + 7*17! + 4*16! + 3*15!
+ 5*14! + 9*13! + 2*12! + 5*11! + 5*10! + 7*9! + 1*8! + 7*7!
+ 3*6! + 0*5! + 0*4! + 0*3! + 0*2! + 0*1! + 0*0!

Cas des nombres de **Bernoulli**

Les nombres de **Bernoulli** sont tous rationnels, par conséquent ils admettent tous un développement factoriel fini.

$$B_0 = 1 = 1.1! + 0.0!$$

$$B_1 = -\frac{1}{2} = -\left(0.0! + \frac{0}{1!} + \frac{1}{2!}\right)$$

$$B_2 = \frac{1}{6} = 0.0! + \frac{0}{1!} + \frac{0}{2!} + \frac{1}{3!}$$

$$B_3 = 0 = 0.0!$$

$$B_4 = -\frac{1}{30} = -\left(0.0! + \frac{0}{1!} + \frac{0}{2!} + \frac{0}{3!} + \frac{0}{4!} + \frac{4}{5!}\right)$$

$$B_5 = 0 = 0.0!$$

Plus généralement, pour $n > 0$:

$$B_{2n} = (-1)^{n+1} \sum_{k=-p}^{k=+q} n_k\,(k)!$$

$$B_{2n+1} = 0.0!$$

Prenons $B_{100} = -\dfrac{94598037891221252932743306949372187272084153306603613338596204311394151972471 1}{33330}$

Développons $|B_{100}|$ dans la base factorielle :

```
a = [9,4,5,9,8,0,3,7,8,1,9,1,2,2,1,2,5,2,9,5,2,2,7,4,3,3,0,6,9,4,9,3,7,2,1,8, 7,2,7,0,2,8,...
     4,1,5,3,3,0,6,6,9,3,6,1,3,3,3,8,5,6,9,6,2,0,4,3,1,1,3,9,5,4,1,5,1,9,7,2,4,7,7,1,1] ;
b = [3,3,3,3,0] ;
f = base_factorielle(a,b) ;
printf('\n |B100| = '), ecrire1(f)
```

On obtient :

```
|B100| = 1*58! + 12*57! + 1*56! + 49*55! + 32*54! + 19*53! + 14*52! + 16*51! + 23*50! + 3*49!
       + 36*48! + 11*47! + 2*46! + 0*45! + 7*44! + 25*43! + 41*42! + 27*41! + 39*40! + 19*39!
       + 3*38! + 33*37! + 34*36! + 16*35! + 13*34! + 10*33! + 9*32! + 24*31! + 2*30! + 1*29!
       + 9*28! + 24*27! + 18*26! + 20*25! + 17*24! + 4*23! + 0*22! + 12*21! + 10*20! + 17*19!
       + 2*18! + 7*17! + 12*16! + 9*15! + 5*14! + 4*13! + 4*12! + 9*11! + 5*10! + 1*9!
       + 5*8! + 0*7! + 2*6! + 2*5! + 2*4! + 3*3! + 1*2! + 0*1! + 0*0! + 0/1!
       + 0/2! + 0/3! + 3/4! + 1/5! + 0/6! + 4/7! + 0/8! + 5/9! + 9/10! + 6/11!
       + 10/12! + 1/13! + 4/14! + 0/15! + 4/16! + 12/17! + 14/18! + 4/19! + 17/20! + 17/21!
       + 5/22! + 15/23! + 6/24! + 4/25! + 11/26! + 22/27! + 20/28! + 6/29! + 26/30! + 22/31!
       + 22/32! + 26/33! + 26/34! + 32/35! + 20/36! + 24/37! + 34/38! + 23/39! + 37/40! + 25/41!
       + 24/42! + 5/43! + 4/44! + 35/45! + 29/46! + 28/47! + 18/48! + 26/49! + 9/50! + 45/51!
       + 49/52! + 22/53! + 30/54! + 26/55! + 7/56! + 43/57! + 26/58! + 24/59! + 32/60! + 4/61!
       + 51/62! + 35/63! + 35/64! + 31/65! + 35/66! + 19/67! + 16/68! + 10/69! + 65/70! + 10/71!
       + 39/72! + 15/73! + 13/74! + 14/75! + 8/76! + 21/77! + 27/78! + 2/79! + 27/80! + 58/81!
       + 44/82! + 54/83! + 19/84! + 81/85! + 54/86! + 43/87! + 6/88! + 8/89! + 73/90! + 6/91!
       + 28/92! + 22/93! + 9/94! + 29/95! + 15/96! + 20/97! + 16/98! + 49/99! + 0/100! + 100/101!
```

Remarque

La profondeur du développement factoriel de la partie fractionnaire d'un nombre de **Bernoulli** a/b (irréductible) est égale à la valeur du plus grand nombre premier figurant dans la décomposition en facteurs premiers du dénominateur b.

Dans le cas de B_{100} , on a $b = 33330 = 2 \times 3 \times 5 \times 11 \times 101$, d'où une profondeur égale à 101.

Conversion décimale

La fonction *base_decimale* convertit un nombre rationnel écrit en base factorielle, en un nombre rationnel écrit en base 10 sous la forme d'une liste de 2 entiers vectoriels signés représentant le numérateur et le dénominateur de la fraction.

```
function F = base_decimale(f)
    V1 = f(1)
    V2 = f(2)
    l1 = length(V1)
    l2 = length(V2)
    F = list([0,0],[1,1])
    N = 1
    for i = 1:max(l1,l2)
        N = multiplication(N,entier(i))
        if i < l1 then
            F1 = list([1,multiplication(entier(V1(l1-i)),N)],[1,1])
            F = F_ADDITION(F,F1)
        end
        if i <= l2 then
            F2 = list([1,entier(V2(i))],[1,N])
            F = F_ADDITION(F,F2)
        end
    end
endfunction
```

Le programme suivant part d'un nombre aléatoire exprimé en base factorielle et calcule sa valeur en base 10 :

```
q = 40 ; p = 35 ;
V1 = [] ; for i = 0:q, V1 = [grand(1,1,'uin',max(0,i+1-q),i),V1] ; end
V2 = [] ; for i = 1:p, V2 = [V2,grand(1,1,'uin',max(0,i+1-p),i-1)] ; end
f = list(V1,V2) ;
F = base_decimale(f) ;
printf('\n f = '), ecrire2(f)
printf('\n F = '), ECRIRE(F)
```

Résultat d'une exécution :

[4,23,25,33,26,9,9,30,15,2,28,6,15,0,16,13,2,7,3,8,15,12,3,17,3,4,4,12,11,3,6,7,4,1,1,5,10,0,1,0;
0,1,2,0,35,6,2,3,8,9,6,1,13,21,12,12,7,9,9,17,15,17,4,13,12,6,21,16,19,0,28,15,20]

$$\frac{77423218239162440954364983727131299002359212391691061396062715243331613824847956459573}{2066629592772289859333026754640000000}$$

Mathématique & Informatique

Soit un grand entier $n = [n_q , n_{q-1} , \ldots , n_2 , n_1 , n_0]$ en base factorielle.
Si $0 \le n_i \le 9$ ($\forall$ i), on peut écrire simplement $n = n_q\, n_{q-1} \ldots n_2\, n_1\, n_0$ puisqu'il n'y a pas de risque de confusion entre les nombres n_i
Voici un programme qui calcule l'équivalent *N* en base 10 d'un entier aléatoire *n* en base factorielle :

```
q = 100 ;
n = [] ; for i = 0:q, n = [grand(1,1,'uin',max(0,i+1-q),min(i,9)),n] ; end
N = base_decimale(list(n,[])) ;
printf('\n n = '), ecrire(n)
printf('\n N = '), ECRIRE(N)
```

Par exemple, les 2 nombres ci-dessous sont identiques : le premier est écrit en base factorielle, selon les factorielles croissantes (0!, 1!, 2!, 3!, …), et le second en base 10, selon les puissances de 10 croissantes (10^0, 10^1, 10^2, 10^3, …).

525422025655645152717459525421287041277157713219639956297386082129225917912618714376859854893730012 10

| | |

468545122755900995903512814388477630006586047999411435752348106181872259660334767209863962031956198 96603425801805215474733667319965766071804908601233224426913 1

Dans ce cas particulier, *n* a 101 chiffres et *N* en a 159.

Autre exemple, on retrouve la valeur en base 10 de 100! en écrivant :

```
n = [1,zeros(1,100)] ;
N = base_decimale(list(n,[])) ;
printf('\n n = '), ecrire(n)
printf('\n N = '), ECRIRE(N)
```

10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

| | |

933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182 86253697920827223758251185210916864000000000000000000000000

Remarque

Le symbole ≡ désigne habituellement une relation d'équivalence.
Ici, $x \equiv y$ signifie que *x* et *y* représentent le même nombre dans 2 systèmes de numération différents.

Code factoriel

Par convention, les nombres 0, 1, 2, … , 9 seront représentés par les chiffres 0, 1, 2, … , 9 ; les nombres 10, 11, 12, … , 35 par les lettres majuscules A, B, C, … , Z ; les nombres 36, 37, 38, … , 61 par les lettres minuscules a, b, c, … , z
En adoptant ce code pour les nombres entiers de 0 à 61, écrivons une fonction *bf(n)* qui retourne la représentation factorielle d'un entier *n* sous la forme d'une chaîne de caractères *s* :

```
function s = bf(n)
    s = '0'
    i = 1
    while ~nul(n)
        if i > 61 then, s = '', return, end
        i = i + 1
        [x,ni] = division(n,entier(i))
        ni = nombre_naturel(ni)
        if ni >= 0 & ni <= 9 then, si = ascii(ni + ascii('0')), end
        if ni >= 10 & ni <= 35 then, si = ascii(ni + ascii('A') - 10), end
        if ni >= 36 & ni <= 61 then, si = ascii(ni + ascii('a') - 36), end
        s = si + s
        n = x
    end
endfunction
```

Le plus grand entier représentable est Max = 61.61! + 60.60! + 59.59! + … + 3.3! + 2.2! + 1.1! + 0.0! = 62! - 1
Soit, en base 10 :
Max = 31469973260387937525653122354950764088012280797258232192163168247821107199999999999999
En base factorielle :
Max = zyxwvutsrqponmlkjihgfedcbaZYXWVUTSRQPONMLKJIHGFEDCBA9876543210

Note Si l'entier *n* est supérieur à Max, la fonction *bf(n)* retourne la chaîne vide ('' sous Scilab).

Par exemple, pour obtenir la représentation factorielle du nombre $n = 2^{200}$, on écrit :

```
n = puissance(2,200) ;
s = bf(n) ;
ecrire(n), printf(' = '), printf(s)
```

Résultat :

160693804425899027554196209234116260252220299378279283530137 6 ≡
6A1Jb6CKR8HB1V5ONPHPHN7D9FDJ0F823DCB676200202200

A propos des chaînes de caractères

Cliquer sur la petite icône dans la barre d'outils de la console Scilab pour accéder à la documentation concernant la manipulation des chaînes de caractères (fonctions *ascii*, *strsplit*, …)
La concaténation de 2 chaînes de caractères s'effectue en utilisant le signe +. Ainsi, la concaténation des chaînes s_1 et s_2 est la chaîne *s* obtenue avec l'affectation $s = s_1 + s_2$. Remarquons que cette addition n'est pas commutative : *s* = '*A*' + '*B*' est équivalent à *s* = '*AB*' alors que *s* = '*B*' + '*A*' est équivalent à *s* = '*BA*' ; par suite : $s = si + s \neq s = s + si$

Inversement, la fonction *bd(s)* retourne la valeur en base 10 d'un entier *n* représenté en base factorielle par la chaîne de caractères *s* :

```
function n = bd(s)
    n = 0
    m = 1
    S = strsplit(s)
    l = size(S,1)
    for i = 0:l-1
        si = S(l - i)
        ni = ascii(si)
        if ni >= ascii('0') & ni <= ascii('9') then, ni = ni - ascii('0'), end
        if ni >= ascii('A') & ni <= ascii('Z') then, ni = ni - ascii('A') + 10, end
        if ni >= ascii('a') & ni <= ascii('z') then, ni = ni - ascii('a') + 36, end
        check = (ni < 0 | ni > i) | (ni == 0 & i == l-1 & l > 1)
        if check then, n = [], return, end
        if i > 0 then, m = multiplication(m,entier(i)), end
        n = addition(n,multiplication(m,entier(ni)))
    end
endfunction
```

Note Si la chaîne *s* n'est pas valide, la fonction *bd(s)* retourne l'ensemble vide ([] sous Scilab).

On retrouve le nombre 2^{200} en exécutant les instructions :

```
s = '6A1Jb6CKR8HB1V5ONPHPHN7D9FDJ0F823DCB676200202200' ;
n = bd(s) ;
printf(s), printf(' = '), ecrire(n)
```

Ce qui donne :

6A1Jb6CKR8HB1V5ONPHPHN7D9FDJ0F823DCB676200202200 ≡
1606938044258990275541962092341162602522202993782792835301376

A propos du codage

Il est évidemment possible de représenter les nombres entiers de 0 à 61 autrement que par la suite ordonnée de caractères alphanumériques 0, … , 9, A, … , Z, a, … , z. Par exemple, toute permutation de ces 62 symboles constitue une clé de codage, parmi 62! possibles, avec laquelle on peut écrire en base factorielle un nombre entier quelconque compris entre 0 et 62! - 1. D'une manière générale, les entiers de 0 à N sont représentables par N + 1 symboles permettant de coder en base factorielle tous les nombres entiers de 0 à (N + 1)! – 1. En pratique, si l'on se limite aux 256 caractères ASCII étendu sur 8 bits, on peut coder en base factorielle les 256! premiers entiers naturels (256! ≈ 8,57817775342842654119e+506). Avec le standard Unicode sur 16 bits, on irait jusqu'à 65536! (65536! ≈ 5,162948e+287193).

Développement en série factorielle

La série à termes positifs $\sum_{i=1}^{\infty} \frac{a_i}{i!}$, $0 \le a_i \le i-1$, est convergente d'après la règle de **Cauchy** :

$$\sqrt[i]{\frac{a_i}{i!}} \underset{i\to\infty}{\sim} \frac{\sqrt[i]{a_i}}{i\, e^{-1}\, \sqrt[2i]{2\pi i}} < \frac{e}{1-\frac{1}{2i}} \underset{i\to\infty}{\longrightarrow} 0 < 1$$

On appellera série factorielle, une série convergente de la forme $\sum_{k=-\infty}^{k=+q} n_k\,(k)!$ où les n_k sont des entiers naturels vérifiant :

$0 \le n_k \le |k|$ si $k \ge 0$ et $0 \le n_k < |k|$ si $k < 0$

Par exemple, le développement en série factorielle du nombre irrationnel e est :

$$e = 1.2! + 0.1! + 0.0! + 0.(-1)! + \sum_{k=-\infty}^{k=-2} (k)!$$

Un nombre irrationnel n'est jamais représenté par un développement factoriel fini puisqu'on pourrait le réduire à une fraction $\frac{a}{b}$.
Cependant, il est possible d'écrire un nombre rationnel sous la forme d'un développement factoriel infini. Prenons par exemple :

$$\sum_{i=1}^{\infty} \frac{i-1}{i!} = \lim_{n\to\infty}\left(\sum_{i=1}^{n} \frac{i-1}{i!}\right) = \lim_{n\to\infty}\left(1-\frac{1}{n!}\right) = 1 \quad \Rightarrow \quad 1 = 1.1! + 0.0! = \frac{0}{1!} + \frac{1}{2!} + \frac{2}{3!} + \cdots + \frac{n-1}{n!} + \cdots$$

En fait, comme on le verra plus loin, certains développements factoriels infinis ne le sont qu'en apparence car ils se ramènent à des développements factoriels finis. Un développement factoriel nécessairement infini, c'est-à-dire qui ne peut pas se ramener à un développement factoriel fini, caractérise un nombre irrationnel.

On obtient un développement en série factorielle d'un nombre irrationnel x en effectuant le développement factoriel d'un nombre rationnel $\frac{a}{b}$ qui approche x d'aussi près que l'on veut.

Par exemple, pour obtenir un développement en série factorielle du nombre π, exécutons le programme suivant :

```
a = [3,1,4,1,5,9,2,6,5,3,5,8,9,7,9,3,2,3,8,4,6,2,6,4,3,3,8,3,2,7,9,5,0,2,8,8,4,1,9,7,1,6,9,3,9,9,3,7,5,1,0, ...
      5,8,2,0,9,7,4,9,4,4,5,9,2,3,0,7,8,1,6,4,0,6,2,8,6,2,0,8,9,9,8,6,2,8,0,3,4,8,2,5,3,4,2,1,1,7,0,6,7,9, ...
      8,2,1,4,8,0,8,6,5,1,3,2,8,2,3,0,6,6,4,7,0,9,3,8,4,4,6,0,9,5,5,0,5,8,2,2,3,1,7,2,5,3,5,9,4,0,8,1,2,8, ...
      4,8,1,1,1,7,4,5,0,2,8,4,1,0,2,7,0,1,9,3,8,5,2,1,1,0,5,5,5,9,6,4,4,6,2,2,9,4,8,9,5,4,9,3,0,3,8,1,9,6] ;
b = [1,zeros(1,length(a)-1)] ;
pi = base_factorielle(a,b) ;
printf('\n Développement factoriel de pi')
ecrire1(pi)
```

Résultat :

$$\pi = 1.2! + 1.1! + 0.0! + \frac{0}{1!} + \frac{0}{2!} + \frac{0}{3!} + \frac{3}{4!} + \frac{1}{5!} + \frac{5}{6!} + \frac{6}{7!} + \frac{5}{8!} + \frac{0}{9!} + \frac{1}{10!} + \frac{4}{11!} + \frac{7}{12!} + \frac{8}{13!}$$
$$+ \frac{0}{14!} + \frac{6}{15!} + \frac{7}{16!} + \frac{10}{17!} + \frac{7}{18!} + \frac{10}{19!} + \frac{4}{20!} + \frac{10}{21!} + \frac{6}{22!} + \frac{16}{23!} + \frac{1}{24!} + \frac{11}{25!} + \frac{20}{26!} + \frac{3}{27!}$$
$$+ \frac{18}{28!} + \frac{12}{29!} + \frac{9}{30!} + \frac{13}{31!} + \frac{18}{32!} + \frac{21}{33!} + \frac{14}{34!} + \frac{34}{35!} + \frac{27}{36!} + \frac{11}{37!} + \frac{27}{38!} + \frac{33}{39!} + \frac{36}{40!} + \frac{18}{41!}$$
$$+ \frac{5}{42!} + \frac{18}{43!} + \frac{5}{44!} + \frac{23}{45!} + \frac{39}{46!} + \frac{1}{47!} + \frac{10}{48!} + \frac{42}{49!} + \frac{28}{50!} + \frac{17}{51!} + \frac{20}{52!} + \frac{51}{53!} + \frac{8}{54!} + \frac{42}{55!}$$
$$+ \frac{47}{56!} + \frac{0}{57!} + \frac{27}{58!} + \frac{23}{59!} + \frac{16}{60!} + \frac{52}{61!} + \frac{32}{62!} + \frac{52}{63!} + \frac{53}{64!} + \frac{24}{65!} + \frac{43}{66!} + \frac{61}{67!} + \frac{64}{68!} + \frac{18}{69!}$$
$$+ \frac{17}{70!} + \frac{11}{71!} + \frac{0}{72!} + \frac{53}{73!} + \frac{14}{74!} + \frac{62}{75!} + \frac{0}{76!} + \frac{38}{77!} + \frac{26}{78!} + \frac{27}{79!} + \frac{23}{80!} + \frac{18}{81!} + \frac{13}{82!} + \frac{66}{83!}$$
$$+ \frac{47}{84!} + \frac{83}{85!} + \frac{78}{86!} + \frac{23}{87!} + \frac{37}{88!} + \frac{73}{89!} + \frac{37}{90!} + \frac{41}{91!} + \frac{35}{92!} + \frac{68}{93!} + \frac{63}{94!} + \frac{15}{95!} + \frac{23}{96!} + \frac{36}{97!}$$
$$+ \frac{90}{98!} + \frac{26}{99!} + \frac{47}{100!} + R_{100}$$

La partie principale de ce développement est une valeur approchée du nombre π à $\frac{1}{100\,!}$ près, c'est-à-dire :

$$R_{100} \leq (-100)\,!$$

On a pris 200 décimales de π pour effectuer les calculs afin d'être certain d'obtenir la bonne valeur des coefficients n_k jusqu'à n_{-100}, puisque : $10^{-200} < (-100)\,! \approx 10^{-158}$
Dans le même ordre d'idées, le programme :

```
a = [5,7,7,2,1,5,6,6,4,9,0,1,5,3,2,8,6,0,6,0,6,5,1,2,0,9,0,0,8,2,4,0,2,4,3,1,0,4,2,1,5,9,3,3,5,9,3,9,9,2, ...
     3,5,9,8,8,0,5,7,6,7,2,3,4,8,8,4,8,6,7,7,2,6,7,7,7,6,6,4,6,7,0,9,3,6,9,4,7,0,6,3,2,9,1,7,4,6,7,4,9,5] ;
b = [1,zeros(1,length(a))] ;
Gamma = base_factorielle(a,b) ;
printf('\n Développement factoriel de la constante d''Euler')
ecrire1(Gamma)
```

donne l'expression de la constante d'**Euler** dans la base factorielle avec une erreur inférieure à $\frac{1}{50\,!}$:

$$\gamma = 0.0! + \frac{0}{1!} + \frac{1}{2!} + \frac{0}{3!} + \frac{1}{4!} + \frac{4}{5!} + \frac{1}{6!} + \frac{4}{7!} + \frac{1}{8!} + \frac{3}{9!} + \frac{0}{10!} + \frac{2}{11!} + \frac{3}{12!} + \frac{0}{13!} + \frac{5}{14!}$$
$$+ \frac{14}{15!} + \frac{12}{16!} + \frac{16}{17!} + \frac{14}{18!} + \frac{7}{19!} + \frac{13}{20!} + \frac{18}{21!} + \frac{17}{22!} + \frac{19}{23!} + \frac{11}{24!} + \frac{22}{25!} + \frac{13}{26!} + \frac{13}{27!}$$
$$+ \frac{26}{28!} + \frac{12}{29!} + \frac{16}{30!} + \frac{2}{31!} + \frac{26}{32!} + \frac{1}{33!} + \frac{2}{34!} + \frac{28}{35!} + \frac{18}{36!} + \frac{3}{37!} + \frac{27}{38!} + \frac{31}{39!} + \frac{27}{40!}$$
$$+ \frac{9}{41!} + \frac{7}{42!} + \frac{37}{43!} + \frac{28}{44!} + \frac{13}{45!} + \frac{26}{46!} + \frac{2}{47!} + \frac{34}{48!} + \frac{29}{49!} + \frac{47}{50!} + R_{50}$$

Si l'on peut démontrer que ce développement est nécessairement infini, alors γ est irrationnelle …

Le reste d'ordre n

D'une manière générale, on peut majorer le reste R_n d'ordre n d'une série factoriell e quelconque : $R_n \leq (-n)!$

Tout d'abord, montrons par récurrence une formule déjà utilisée lors de la preuve de l'existence du développement factoriel fini d'un nombre rationnel. Soient deux entiers a et b, $a < b$, on a :

$$\frac{a}{a+1} + \frac{a+1}{(a+1)\,.\,(a+2)} + \cdots + \frac{b-1}{(a+1)\,.\,(a+2)\cdots(b-1)\,b} = 1 - \frac{a!}{b!}$$

- Pour $b = a + 1$:

$$\frac{a}{a+1} = 1 - \frac{1}{a+1} = 1 - \frac{a!}{(a+1)!}$$

- Supposons l'égalité vérifiée pour $b > a$, alors pour $b + 1$:

$$\frac{a}{a+1} + \frac{a+1}{(a+1)\,.\,(a+2)} + \cdots + \frac{b-1}{(a+1)\,.\,(a+2)\cdots(b-1)\,b} + \frac{b}{(a+1)\,.\,(a+2)\cdots b(b+1)}$$
$$= 1 - \frac{a!}{b!} + \frac{b}{(a+1)\,.\,(a+2)\cdots b(b+1)} = 1 - \frac{a!}{b!} + \frac{a!\,b}{(b+1)!} = 1 - \left[\frac{a!\,(b+1) - a!\,b}{(b+1)!}\right]$$
$$= 1 - \frac{a!}{(b+1)!}$$

D'où :

$$R_n = \sum_{i=n+1}^{\infty} \frac{a_i}{i!} \leq \sum_{i=n+1}^{\infty} \frac{i-1}{i!} = \frac{1}{n!} \lim_{i \to \infty} \left(\sum_{j=n+1}^{i} \frac{j-1}{\prod_{k=1}^{j-n} (n+k)} \right) = \frac{1}{n!} \lim_{i \to \infty} \left(1 - \frac{n!}{i!}\right) = \frac{1}{n!} = (-n)!$$

Comme $(-n)! \xrightarrow[n \infty]{} 0$, on déduit que $R_n \xrightarrow[n \infty]{} 0$, ce qui est en accord avec le fait que la série soit convergente.

Note

On peut établir ce résultat plus simplement à partir de la formule : $\sum_{i=1}^{n} \frac{i-1}{i!} = 1 - \frac{1}{n!}$ (cas $a = 1$ et $b = n$)

$$R_n = \sum_{i=n+1}^{\infty} \frac{a_i}{i!} \leq \sum_{i=n+1}^{\infty} \frac{i-1}{i!} = \sum_{i=1}^{\infty} \frac{i-1}{i!} - \sum_{i=1}^{n} \frac{i-1}{i!} = \lim_{n \to \infty} \left(\sum_{i=1}^{n} \frac{i-1}{i!} \right) - \sum_{i=1}^{n} \frac{i-1}{i!}$$
$$= \lim_{n \to \infty} \left(1 - \frac{1}{n!}\right) - \left(1 - \frac{1}{n!}\right) = 1 - \left(1 - \frac{1}{n!}\right) = \frac{1}{n!} = (-n)!$$

Mathématique & Informatique

Sur le caractère fini/infini du développement factoriel d'un nombre rationnel

On sait que tout nombre rationnel x positif se met sous une forme factorielle finie :

$$x = n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \ldots + \frac{n_{-p}}{p!}$$

où les coefficients n_i sont des entiers naturels compris entre 0 et i et les n_{-i} entre 0 et $i-1$.

En introduisant la série $\displaystyle\sum_{i=p+1}^{\infty} \frac{i-1}{i!} = \frac{1}{p!}$, le rationnel x peut aussi se mettre sous une forme factorielle infinie équivalente.

Distinguons 2 cas :

1) x est un entier non nul : $x = m + 1$

$$x = n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + 1 = n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + \sum_{i=1}^{\infty} \frac{i-1}{i!}$$

$$= n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + \frac{0}{1!} + \frac{1}{2!} + \frac{2}{3!} + \frac{3}{4!} + \frac{4}{5!} + \ldots$$

2) x n'est pas un entier : $x = \dfrac{a}{b}$

$$x = n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \ldots + \frac{n_{-p}}{p!} \qquad (n_{-p} \neq 0)$$

$$= n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \ldots + \frac{n_{-p}-1}{p!} + \frac{1}{p!}$$

$$= n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \ldots + \frac{n_{-p}-1}{p!} + \sum_{i=p+1}^{\infty} \frac{i-1}{i!}$$

$$= n_q\, q! + \ldots + n_2\, 2! + n_1\, 1! + n_0\, 0! + \frac{n_{-1}}{1!} + \frac{n_{-2}}{2!} + \ldots + \frac{n_{-p}-1}{p!} + \frac{p}{(p+1)!} + \frac{p+1}{(p+2)!} + \frac{p+2}{(p+3)!} + \ldots$$

Par exemple :

$$2012 = 2.6! + 4.5! + 3.4! + 3.3! + 1.2! + 0.1! + 0.0!$$

$$= 2.6! + 4.5! + 3.4! + 3.3! + 0.2! + 1.1! + 0.0! + \frac{0}{1!} + \frac{1}{2!} + \frac{2}{3!} + \frac{3}{4!} + \frac{4}{5!} + \ldots$$

$$\frac{2012}{21} = 3.4! + 3.3! + 2.2! + 1.1! + 0.0! + \frac{0}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{3}{4!} + \frac{2}{5!} + \frac{0}{6!} + \frac{6}{7!}$$

$$= 3.4! + 3.3! + 2.2! + 1.1! + 0.0! + \frac{0}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{3}{4!} + \frac{2}{5!} + \frac{0}{6!} + \frac{5}{7!} + \frac{7}{8!} + \frac{8}{9!} + \frac{9}{10!} + \ldots$$

Remarque Seul le nombre 0 n'admet pas de développement factoriel infini : $0 = 0.0!$

Notons l'analogie entre l'infinité de chiffres $p + k$ (en progression arithmétique de raison 1) terminant la partie fractionnaire d'un nombre en base factorielle et l'infinité de chiffres $N - 1$ (tous identiques) terminant la partie fractionnaire d'un nombre en base N.

Par exemple, dans le système décimal, soit le nombre rationnel : $x = \frac{5503003}{25000000} = 0{,}22012012$

Avec nos conventions, x est représentable en base 10 par :

x = [0;2,2,0,1,2,0,1,2]

= [0;2,2,0,1,2,0,1,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0…]

= [0;2,2,0,1,2,0,1,1,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9…]

En base factorielle, x est représentable par :

x = [0;0,0,1,1,1,2,3,3,2,1,9,9,8,10,5,1,5,16,0,17,20,20,11,23,23,2,2,4,13,27,18,19,6,20,14]

= [0;0,0,1,1,1,2,3,3,2,1,9,9,8,10,5,1,5,16,0,17,20,20,11,23,23,2,2,4,13,27,18,19,6,20,14,0,0,0,0,0,0,0,0,0,0,0,…]

= [0;0,0,1,1,1,2,3,3,2,1,9,9,8,10,5,1,5,16,0,17,20,20,11,23,23,2,2,4,13,27,18,19,6,20,13,35,36,37,38,39,40,41,…]

On a évidemment :

$$\sum_{i=p+1}^{\infty} \frac{0}{N^i} = 0 \qquad \text{et} \qquad \sum_{i=p+1}^{\infty} \frac{0}{i!} = 0$$

La série géométrique de terme général $\frac{N-1}{N^i}$ et la série factoriell e de terme général $\frac{i-1}{i!}$ donnent les formules suivantes :

$$\sum_{i=p+1}^{\infty} \frac{N-1}{N^i} = \frac{1}{N^p} \qquad \text{et} \qquad \sum_{i=p+1}^{\infty} \frac{i-1}{i!} = \frac{1}{p!}$$

On déduit qu'un rationnel x qui s'écrit avec un nombre fini de chiffres, peut également s'écrire, de manière équivalente, avec un nombre infini de chiffres :

$x = [\, n_q\, ,\, \ldots\, ,\, n_0\, ;\, n_{-1}\, ,\, \ldots\, ,\, n_{-p}\,]$

$= [\, n_q\, ,\, \ldots\, ,\, n_0\, ;\, n_{-1}\, ,\, \ldots\, ,\, n_{-p}\, ,\, 0\, ,\, 0\, ,\, 0\, ,\, 0\, ,\, \ldots\,]$

$= [\, n_q\, ,\, \ldots\, ,\, n_0\, ;\, n_{-1}\, ,\, \ldots\, ,\, n_{-p} - 1\, ,\, a_{p+1}\, ,\, a_{p+2}\, ,\, a_{p+3}\, ,\, a_{p+4}\, ,\, \ldots\,]$

En base N, les a_i $(i \geq p+1)$ sont tous égaux à $N - 1$: $a_{p+1} = N - 1,\ a_{p+2} = N - 1,\ a_{p+3} = N - 1,\ a_{p+4} = N - 1,\ \ldots$

En base factorielle, les a_i $(i \geq p+1)$ sont égaux à $i - 1$: $a_{p+1} = p,\ a_{p+2} = p+1,\ a_{p+3} = p+2,\ a_{p+4} = p+3, \ldots$

On dira que la notation $[\, n_q\, ,\, \ldots\, ,\, n_0\, ;\, n_{-1}\, ,\, \ldots\, ,\, n_{-p}\,]$ est l'écriture "propre" du rationnel x et que les notations $[\, n_q\, ,\, \ldots\, ,\, n_0\, ;\, n_{-1}\, ,\, \ldots\, ,\, n_{-p}\, ,\, 0\, ,\, 0\, ,\, 0\, ,\, 0\, ,\, \ldots\,]$ et $[\, n_q\, ,\, \ldots\, ,\, n_0\, ;\, n_{-1}\, ,\, \ldots\, ,\, n_{-p} - 1\, ,\, a_{p+1}\, ,\, a_{p+2}\, ,\, a_{p+3}\, ,\, a_{p+4}\, ,\, \ldots\,]$ sont des écritures "impropres" du rationnel x.

Mathématique & Informatique

Exemple d'une somme infinie convergeant vers un nombre rationnel

Prenons la série (u_n) dont le terme général est le produit de 2 entiers impairs successifs : $u_n = \dfrac{1}{(2n-1)(2n+1)}$

Décomposons u_n en éléments simples : $u_n = \dfrac{1}{(2n-1)(2n+1)} = \dfrac{1/2}{2n-1} - \dfrac{1/2}{2n+1}$

Additionnons les n premiers termes $u_1, \ldots, u_n$ pour obtenir la somme partielle d'ordre n : $U_n = \sum_{k=1}^{n} u_k = \dfrac{1}{2} - \dfrac{1/2}{2n+1}$

Par définition, la limite de U_n est la somme de la série (u_n) : $\lim_{n \to \infty} U_n = \dfrac{1}{2}$

Donc :

$$\sum_{n=1}^{\infty} \frac{1}{(2n-1)(2n+1)} = \frac{1}{1.3} + \frac{1}{3.5} + \frac{1}{5.7} + \frac{1}{7.9} + \frac{1}{9.11} + \frac{1}{11.13} + \cdots = \frac{1}{2} \qquad \text{(rationnel)}$$

Chaque terme $\dfrac{1}{(2n-1)(2n+1)}$ se décompose de manière unique dans la base factorielle :

$$\frac{1}{(2n-1)(2n+1)} = \frac{a_1}{1!} + \frac{a_2}{2!} + \frac{a_3}{3!} + \cdots + \frac{a_p}{p!} \qquad \text{avec } p \le \text{Max}(2n-1, 2n+1) \text{ et } a_p \neq 0$$

Lorsque $2n \pm 1$ est un nombre premier, on atteint la profondeur $p = 2n \pm 1$

Le développement factoriel de la série (u_n), obtenu en décomposant son terme général, s'écrit :

$$\sum_{n=1}^{\infty} u_n = \sum_{n=1}^{\infty} \sum_{i=1}^{p_n} \frac{a_{n,i}}{i!} \qquad (0 \le a_{n,i} \le i-1)$$

Après une succession de décompositions et de regroupements des premiers termes de la série (u_n), on trouve :

$$\sum_{n=1}^{\infty} u_n = \frac{0}{1!} + \frac{0}{2!} + \frac{2}{3!} + \frac{3}{4!} + \frac{4}{5!} + \frac{5}{6!} + \frac{6}{7!} + \cdots = \frac{0}{1!} + \frac{0}{2!} + \sum_{i=3}^{\infty} \frac{i-1}{i!}$$

Comme $\displaystyle\sum_{i=3}^{\infty} \frac{i-1}{i!} = \frac{1}{2!}$, on a finalement :

$$\sum_{n=1}^{\infty} u_n = \frac{0}{1!} + \frac{1}{2!} = \frac{1}{2}$$

Cet exemple montre qu'il est difficile de prévoir, d'après son expression analytique, si une somme infinie converge vers un nombre rationnel ou irrationnel :

$$\sum_{n=0}^{\infty} \frac{1}{(2n+1)(2n+3)} = \frac{1}{2} \quad \text{mais} \quad \sum_{n=0}^{\infty} \frac{1}{(4n+1)(4n+3)} = \frac{\pi}{8} \quad \textbf{(Leibniz)}$$

Exemple d'un produit infini convergeant vers un nombre rationnel

Considérons le produit infini :

$$P = \prod_{n=2}^{\infty} \frac{n^2}{n^2 - 1}$$

Le produit partiel d'ordre n est :

$$P_n = \prod_{k=2}^{n} \frac{k^2}{k^2 - 1}$$

D'où :

$$\ln P_n = \sum_{k=2}^{n} \ln \frac{k^2}{k^2 - 1} = \sum_{k=2}^{n} \left(\ln \frac{k}{k-1} + \ln \frac{k}{k+1} \right) = \left(\ln \frac{2}{1} + \ln \frac{2}{3} \right) + \left(\ln \frac{3}{2} + \ln \frac{3}{4} \right) + \cdots + \left(\ln \frac{n}{n-1} + \ln \frac{n}{n+1} \right)$$

$$= \ln \frac{2}{1} + \ln \frac{n}{n+1} = \ln \frac{2n}{n+1}$$

On en déduit : $\ln P_n \xrightarrow[n \infty]{} \ln 2 \quad \Rightarrow \quad P_n \xrightarrow[n \infty]{} e^{\ln 2} = 2$

Donc :

$$\prod_{n=2}^{\infty} \frac{n^2}{n^2 - 1} = \frac{2.2.3.3.44.5.5.6.67.7\cdots}{1.3.2.4.35.4.6.5.7\cdots} = 2 \qquad \text{(rationnel)}$$

Cet exemple montre qu'il est difficile de prévoir, d'après son expression analytique, si un produit infini converge vers un nombre rationnel ou irrationnel :

$$\prod_{n=2}^{\infty} \frac{n^2}{n^2 - 1} = 2 \quad \text{mais} \quad \prod_{n=1}^{\infty} \frac{4n^2}{4n^2 - 1} = \frac{\pi}{2} \quad \textbf{(Wallis)}$$

Remarque

$P_1 = \prod_{i=1}^{\infty} \frac{{p_i}^2}{{p_i}^2 - 1}$ où p_i est le i^{e} nombre premier ($p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, p_5 = 11, p_6 = 13, \ldots$)

$P_2 = \prod_{i=1}^{\infty} \frac{{c_i}^2}{{c_i}^2 - 1}$ où c_i est le i^{e} nombre composé ($c_1 = 4, c_2 = 6, c_3 = 8, c_4 = 9, c_5 = 10, c_6 = 12, \ldots$)

On a : $P = P_1 P_2$ avec $P_1 = \zeta(2) = \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$, d'où : $P_2 = \frac{P}{P_1} = \frac{12}{\pi^2}$.

Donc : $\prod_{i=1}^{\infty} \frac{{p_i}^2}{{p_i}^2 - 1} = \frac{\pi^2}{6}$ $\qquad$ $\prod_{i=1}^{\infty} \frac{{c_i}^2}{{c_i}^2 - 1} = \frac{12}{\pi^2}$

Mathématique & Informatique

Critère factoriel d'irrationalité d'un nombre réel

Un développement factoriel infini $\sum_{i=1}^{\infty} \frac{a_i}{i!}$, $0 \le a_i \le i-1$ et a_i non tous nuls à partir d'un certain rang, représente la partie fractionnaire d'un nombre rationnel si et seulement si son reste d'ordre n peut être ramené à un nombre rationnel.

Posons :

$$\sum_{i=1}^{\infty} \frac{a_i}{i!} = \sum_{i=1}^{n} \frac{a_i}{i!} + \sum_{i=n+1}^{\infty} \frac{a_i}{i!} = \sum_{i=1}^{n} \frac{a_i}{i!} + R_n$$

D'après un calcul précédent, le reste R_n d'ordre n est rationnel si : $R_n = \sum_{i=n+1}^{\infty} \frac{a_i}{i!} = \sum_{i=n+1}^{\infty} \frac{i-1}{i!} = \frac{1}{n!}$

Donc : $\sum_{i=1}^{\infty} \frac{a_i}{i!}$ est un nombre rationnel $\Leftrightarrow$ $a_i = i-1$ à partir d'un certain rang

Un nombre est rationnel si et seulement si son développement factoriel propre est fini.

Propriété

$$\sum_{i=1}^{\infty} \frac{a_i}{i!} \text{ est irrationnel} \quad \Leftrightarrow \quad \forall\, k, \exists\, i \ge k : \; a_i \ne i-1$$

Un nombre est irrationnel si et seulement si son développement factoriel propre est infini

Exemple 1

$x_2 = \frac{0}{1!} + \frac{0}{2!} + \frac{2}{3!} + \frac{2}{4!} + \frac{4}{5!} + \frac{4}{6!} + \cdots + \frac{2k-2}{(2k-1)!} + \frac{2k-2}{(2k)!} + \cdots$ est irrationnel car $a_i \ne i-1$ au rang $i = 2k$ $(\forall\, k)$

$$x_2 = \sum_{i=1}^{\infty} \frac{i-1}{i!} - \left(\sum_{k=0}^{\infty} \frac{1}{(2k)!} - 1 \right) = 2 - ch1$$

Exemple 2

$x_1 = \frac{0}{1!} + \frac{1}{2!} + \frac{0}{3!} + \frac{1}{4!} + \frac{0}{5!} + \frac{4}{6!} + \frac{1}{7!} + \frac{7}{8!} + \frac{8}{9!} + \cdots + \frac{i-1}{i!} + \cdots$ est rationnel car $a_i = i-1$ à partir du rang 8

$$x_1 = \left(\frac{0}{1!} + \frac{1}{2!} + \frac{0}{3!} + \frac{1}{4!} + \frac{0}{5!} + \frac{4}{6!} + \frac{1}{7!} \right) + \sum_{i=8}^{\infty} \frac{i-1}{i!} = \frac{0}{1!} + \frac{1}{2!} + \frac{0}{3!} + \frac{1}{4!} + \frac{0}{5!} + \frac{4}{6!} + \frac{1}{7!} + \frac{1}{7!} = \frac{23}{42}$$

Application au nombre *e*

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

De cette formule, on déduit immédiatement le développement factoriel de *e* :

$$e = 1.2! + 0.1! + 0.0! + \frac{0}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \cdots + \frac{1}{i!} + \cdots = [1,0,0;0,1,1,1,1,\cdots,1,\cdots]$$

Il apparaît que la partie fractionnaire est infinie et que la série $\sum_{i=1}^{\infty} \frac{a_i}{i!}$ a pour coefficients $a_i = 1 \neq i - 1$ dès que $i > 2$.

Donc *e* est irrationnel.

Remarque générale

Le terme a_i doit impérativement être compris entre 0 et $i - 1$, sinon la série $\sum_{i=1}^{\infty} \frac{a_i}{i!}$ n'est pas le développement factoriel de la partie fractionnaire d'un nombre réel et la propriété d'irrationalité ne se s'applique pas. Considérons, par exemple, le nombre :

$$x = \frac{0}{1!} + \frac{1}{2!} + \frac{0}{3!} + \frac{3}{4!} + \frac{34}{5!} + \frac{53}{6!} + \frac{76}{7!} + \frac{103}{8!} + \frac{134}{9!} + \cdots + \frac{2n^2 - 3n - 1}{n!} + \cdots$$

La série de terme général $u_n = \frac{2n^2 - 3n - 1}{n!}$ est convergente d'après la règle de d'**Alembert**, puisque :

$$\frac{u_{n+1}}{u_n} = \frac{2n^2 + n - 2}{2n^3 - n^2 - 4n - 1} \xrightarrow[n\infty]{} 0 \ (<1)$$

Donc *x* est une valeur réelle finie.

On peut mettre *x* sous la forme $\sum_{i=1}^{\infty} \frac{a_i}{i!}$, avec :

$$\begin{cases} a_1 = 0,\ a_2 = 1,\ a_3 = 0,\ a_4 = 3 \\ a_i = 2i^2 - 3i - 1 \text{ si } i \geq 5 \end{cases}$$

A partir de l'indice $i = 5$, on a $a_i \neq i - 1$ mais avec $a_i \notin [0, i - 1]$ ce qui ne permet pas de conclure sur l'irrationalité de *x*.
En fait, il s'avère que *x* est un nombre rationnel :

$$\sum_{n=2}^{\infty} \frac{2n^2 - 3n - 1}{n!} = \sum_{n=2}^{\infty} \frac{2n(n-1) - n - 1}{n!} = 2\sum_{n=2}^{\infty} \frac{1}{(n-2)!} - \sum_{n=2}^{\infty} \frac{1}{(n-1)!} - \sum_{n=2}^{\infty} \frac{1}{n!} = 2e - (e-1) - (e-2) = 3$$

$$x = \frac{0}{1!} + \frac{1}{2!} + \frac{0}{3!} + \frac{3}{4!} + \left(\sum_{n=2}^{\infty} \frac{2n^2 - 3n - 1}{n!} - \frac{1}{2!} - \frac{8}{3!} - \frac{19}{4!} \right) = \frac{1}{2} + \frac{1}{8} + 3 - \frac{1}{2} - \frac{4}{3} - \frac{19}{24} = 1$$

Le développement de *x* dans la base factorielle est :

$x = 1.1! + 0.0!$ (développement factoriel propre)

$$x = \frac{0}{1!} + \frac{1}{2!} + \frac{2}{3!} + \frac{3}{4!} + \frac{4}{5!} + \frac{5}{6!} + \cdots + \frac{i-1}{i!} + \cdots$$ (développement factoriel impropre)

Application au cosinus et au sinus

Montrer que $\cos\frac{1}{n}$ et $\sin\frac{1}{n}$ sont des nombres irrationnels, pour tout entier naturel n non nul.

Preuve

Dans un premier temps, établissons que cos 1 et sin 1 sont irrationnels.

- Nombre *cos 1*

La formule de **Taylor-Maclaurin** $f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} x^n$ appliquée à la fonction $f(x) = \cos x$ donne :

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \qquad (\forall\, x \in \mathbb{R})$$

En particulier, pour $x = 1$:

$$\cos 1 = 1 - \frac{1}{2!} + \frac{1}{4!} - \frac{1}{6!} + \frac{1}{8!} - \frac{1}{10!} + \frac{1}{12!} - \frac{1}{14!} + \frac{1}{16!} - \frac{1}{18!} + \frac{1}{20!} - \cdots$$

Introduisons la série factorielle $\sum_{i=1}^{\infty} \frac{i-1}{i!} = 1$ dans le but de faire disparaître les termes négatifs :

$$\cos 1 + 1 = 1 - \frac{1}{2!} + \frac{1}{4!} - \frac{1}{6!} + \frac{1}{8!} - \frac{1}{10!} + \frac{1}{12!} - \frac{1}{14!} + \frac{1}{16!} - \frac{1}{18!} + \frac{1}{20!} - \cdots$$
$$+ \frac{0}{1!} + \frac{1}{2!} + \frac{2}{3!} + \frac{3}{4!} + \frac{4}{5!} + \frac{5}{6!} + \frac{6}{7!} + \frac{7}{8!} + \frac{8}{9!} + \frac{9}{10!} + \frac{10}{11!} + \frac{11}{12!} + \frac{12}{13!} + \frac{13}{14!} + \frac{14}{15!} + \frac{15}{16!} + \frac{16}{17!} + \frac{17}{18!} + \frac{18}{19!} + \frac{19}{20!} + \cdots$$

Regroupons et additionnons les termes selon les factorielles croissantes :

$$\cos 1 = \frac{0}{2!} + \frac{2}{3!} + \frac{4}{4!} + \frac{4}{5!} + \frac{4}{6!} + \frac{6}{7!} + \frac{8}{8!} + \frac{8}{9!} + \frac{8}{10!} + \frac{10}{11!} + \frac{12}{12!} + \frac{12}{13!} + \frac{12}{14!} + \frac{14}{15!} + \frac{16}{16!} + \frac{16}{17!} + \frac{16}{18!} + \frac{18}{19!} + \frac{20}{20!} + \cdots$$

Soit, après simplification par blocs de 3 termes successifs :

$$\cos 1 = \frac{1}{2!} + \frac{4}{5!} + \frac{5}{6!} + \frac{8}{9!} + \frac{9}{10!} + \frac{12}{13!} + \frac{13}{14!} + \frac{16}{17!} + \frac{17}{18!} + \cdots$$

Ou encore, avec tous les entiers $i!$:

$$\cos 1 = \left(\frac{0}{0!} + \frac{0}{1!} + \frac{1}{2!} + \frac{0}{3!}\right) + \left(\frac{0}{4!} + \frac{4}{5!} + \frac{5}{6!} + \frac{0}{7!}\right) + \left(\frac{0}{8!} + \frac{8}{9!} + \frac{9}{10!} + \frac{0}{11!}\right) + \left(\frac{0}{12!} + \frac{12}{13!} + \frac{13}{14!} + \frac{0}{15!}\right) + \left(\frac{0}{16!} + \frac{16}{17!} + \frac{17}{18!} + \frac{0}{19!}\right) + \cdots$$

On en déduit l'expression générale de cos 1 :

$$\cos 1 = \sum_{k=0}^{\infty} \frac{0}{(4k)!} + \sum_{k=0}^{\infty} \frac{4k}{(4k+1)!} + \sum_{k=0}^{\infty} \frac{4k+1}{(4k+2)!} + \sum_{k=0}^{\infty} \frac{0}{(4k+3)!}$$

Ce qui s'écrit :

$$\cos 1 = \sum_{i=1}^{\infty} \frac{a_i}{i!} \quad \text{avec} \quad a_i = \begin{cases} 0 \text{ si } i = 4k \\ 4k \text{ si } i = 4k+1 \\ 4k+1 \text{ si } i = 4k+2 \\ 0 \text{ si } i = 4k+3 \end{cases}$$

On reconnaît un développement factoriel propre infini, donc cos 1 est un nombre irrationnel.

Effectuons le calcul de la somme des premiers termes de la série et comparons avec la valeur de la fonction cos(x) en $x = 1$:

```
cos1 = 0 ;
for k = 0:10
    cos1 = cos1 + (4*k)/factorial(4*k+1) + (4*k+1)/factorial(4*k+2) ;
end
printf('\n cos(1) = %.10f = %.10f',cos(1),cos1)
```

On trouve :

cos(1) = 0.5403023059 = 0.5403023059

- Nombre *sin 1*

La démonstration est identique à la précédente, en partant cette fois de l'expression :

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \qquad (\forall\, x \in \mathbb{R})$$

On aboutit au développement factoriel de sin 1 :

$$\sin 1 = \sum_{i=1}^{\infty} \frac{a_i}{i!} \quad \text{avec} \quad a_i = \begin{cases} 0 \text{ si } i = 4k \\ 0 \text{ si } i = 4k+1 \\ 4k+1 \text{ si } i = 4k+2 \\ 4k+2 \text{ si } i = 4k+3 \end{cases}$$

Sous cette forme, il apparaît que sin 1 est un nombre irrationnel.

A titre de vérification numérique succincte, le programme Scilab :

```
sin1 = 0 ;
for k = 0:10
    sin1 = sin1 + (4*k+1)/factorial(4*k+2) + (4*k+2)/factorial(4*k+3) ;
end
printf('\n sin(1) = %.10f = %.10f',sin(1),sin1)
```

donne le résultat suivant :

sin(1) = 0.8414709848 = 0.8414709848

Remarque

La fonction sin(x) admet le développement en fraction continue généralisée suivant :

$$\sin(x) = \cfrac{x}{1 + \cfrac{x^2}{(2.3 - x^2) + \cfrac{2.3\,x^2}{(4.5 - x^2) + \cfrac{4.5\,x^2}{(6.7 - x^2) + \cfrac{6.7\,x^2}{(8.9 - x^2) + \cdots}}}}}$$

D'où, pour $x = 1$:

$$\sin 1 = \cfrac{1}{1 + \cfrac{1}{(2.3 - 1) + \cfrac{2.3}{(4.5 - 1) + \cfrac{4.5}{(6.7 - 1) + \cfrac{6.7}{(8.9 - 1) + \cdots}}}}}$$

$$\sin 1 = \left[0; \frac{1}{1}, \frac{1}{5}, \frac{6}{19}, \frac{20}{41}, \frac{42}{71}, \cdots, \frac{k(k+1)}{(k+2)(k+3) - 1}, \cdots\right]$$

Le critère d'irrationalité des fractions continues généralisées s'applique car $(k + 2)(k + 3) - 1 \geq k(k + 1) > 0$ à partir du rang $k = 2$.
Donc sin 1 est irrationnel.
A noter que pour un entier $n > 1$ on ne peut pas conclure sur l'irrationalité de sin(n) puisque $k(k + 1)n^2 > (k + 2)(k + 3) - n^2$ dès que k est suffisamment grand :

$$\lim_{k \to +\infty} \frac{k^2 + 5k + 6}{k^2 + k + 1} = 1^+ \quad \Rightarrow \quad \exists \kappa, \forall k > \kappa : \left| \frac{k^2 + 5k + 6}{k^2 + k + 1} - 1 \right| < \varepsilon = n^2 - 1 \quad \Rightarrow \quad k(k+1)n^2 > (k+2)(k+3) - n^2$$

Les formules de l'arc multiple

Appliquons la formule du binôme de **Newton** à la relation $(\cos x + i \sin x)^n = \cos nx + i \sin nx$ (**Moivre**), puis séparons les indices pairs et les indices impairs, et finalement identifions les parties réelles et imaginaires :

$$\cos nx + i \sin nx = (\cos x + i \sin x)^n = (i \sin x + \cos x)^n = \sum_{k=0}^{n} C_n^k i^k \sin^k x \cos^{n-k} x$$

$$= \sum_{p=0}^{E\left(\frac{n}{2}\right)} C_n^{2p} i^{2p} \sin^{2p} x \cos^{n-2p} x + \sum_{p=0}^{E\left(\frac{n-1}{2}\right)} C_n^{2p+1} i^{2p+1} \sin^{2p+1} x \cos^{n-(2p+1)} x$$

Comme $i^{2p} = (-1)^p$ et $i^{2p+1} = (-1)^p i$, on obtient :

$$\cos nx = \sum_{p=0}^{E\left(\frac{n}{2}\right)} (-1)^p C_n^{2p} \cos^{n-2p} x \sin^{2p} x$$

$$\sin nx = \sum_{p=0}^{E\left(\frac{n-1}{2}\right)} (-1)^p C_n^{2p+1} \cos^{n-2p-1} x \sin^{2p+1} x$$

- Nombre $cos\frac{1}{n}$ $(n > 1)$

Supposons $\cos\frac{1}{n}$ rationnel, alors $\cos^m\frac{1}{n}$ est rationnel ($\forall$ *m* entier).

On écrit :

$$\cos(n\times\frac{1}{n}) = \sum_{p=0}^{E\left(\frac{n}{2}\right)}(-1)^p\complement_n^{2p}\cos^{n-2p}\left(\frac{1}{n}\right)\sin^{2p}\left(\frac{1}{n}\right)$$

$$\cos 1 = \sum_{p=0}^{E\left(\frac{n}{2}\right)}(-1)^p\complement_n^{2p}\cos^{n-2p}\left(\frac{1}{n}\right)\left[1-\cos^2\left(\frac{1}{n}\right)\right]^p$$

Le second membre est un nombre rationnel. Par suite, cos 1 est rationnel, ce qui est faux. Donc : $\cos\frac{1}{n}$ est irrationnel.

- Nombre $sin\frac{1}{n}$ $(n > 1)$

Supposons $\sin\frac{1}{n}$ rationnel, alors $\sin^m\frac{1}{n}$ est rationnel ($\forall$ *m* entier).

Cas 1 : *n* est impair

On écrit :

$$\sin(n\times\frac{1}{n}) = \sum_{p=0}^{E\left(\frac{n-1}{2}\right)}(-1)^p\complement_n^{2p+1}\cos^{n-2p-1}\left(\frac{1}{n}\right)\sin^{2p+1}\left(\frac{1}{n}\right)$$

$$\sin 1 = \sum_{p=0}^{\frac{n-1}{2}}(-1)^p\complement_n^{2p+1}\left[1-\sin^2\left(\frac{1}{n}\right)\right]^{\frac{n-1}{2}-p}\sin^{2p+1}\left(\frac{1}{n}\right)$$

Le second membre est un nombre rationnel. Par suite, sin 1 est rationnel, ce qui est faux. Donc : $\sin\frac{1}{n}$ est irrationnel.

Cas 2 : *n* est pair

On écrit :

$$\cos(n\times\frac{1}{n}) = \sum_{p=0}^{E\left(\frac{n}{2}\right)}(-1)^p\complement_n^{2p}\cos^{n-2p}\left(\frac{1}{n}\right)\sin^{2p}\left(\frac{1}{n}\right)$$

$$\cos 1 = \sum_{p=0}^{\frac{n}{2}}(-1)^p\complement_n^{2p}\left[1-\sin^2\left(\frac{1}{n}\right)\right]^{\frac{n}{2}-p}\sin^{2p}\left(\frac{1}{n}\right)$$

Le second membre est un nombre rationnel. Par suite, cos 1 est rationnel, ce qui est faux. Donc : $\sin\frac{1}{n}$ est irrationnel.

C.Q.F.D.

Extension

Lambert a montré que le développement en fraction continue généralisée de tan *x* est :

$$\tan x = \left[0; \frac{x}{1}, -\frac{x^2}{3}, -\frac{x^2}{5}, -\frac{x^2}{7}, \cdots, -\frac{x^2}{2n+1}, \cdots\right]$$

On en déduit le théorème suivant :

$$r \in \mathbb{Q} \Rightarrow \tan r \notin \mathbb{Q}$$

En particulier $\tan\frac{1}{n}$ est irrationnel pour tout entier *n* non nul.

De même $\operatorname{cotan}\frac{1}{n} = \frac{1}{\tan\frac{1}{n}}$ est irrationnel puisque $\frac{1}{x}$ est irrationnel lorsque *x* est irrationnel.

Finalement :

Les nombres $\sin\frac{1}{n}, \cos\frac{1}{n}, \tan\frac{1}{n}$ et $\operatorname{cotan}\frac{1}{n}$ sont tous irrationnels, quel que soit l'entier naturel *n* non nul.

Note

Le théorème de **Lambert** donne immédiatement l'irrationalité de π.

Par contraposition : $\tan r \in \mathbb{Q} \Rightarrow r \notin \mathbb{Q}$

En particulier : $\tan\frac{\pi}{4} = 1 \in \mathbb{Q} \Rightarrow \frac{\pi}{4} \notin \mathbb{Q} \Rightarrow \pi$ irrationnel.

Application à la série de **Riemann**

$$\zeta(\alpha) = \sum_{n=1}^{+\infty} \frac{1}{n^{\alpha}}$$

La série $\sum_1^{+\infty} \frac{1}{n^{\alpha}}$ est de même nature que l'intégrale $\int_1^{+\infty} \frac{dx}{x^{\alpha}}$, c'est-à-dire convergente pour $\alpha > 1$.
On cherche à montrer que $\zeta(m)$ est irrationnel pour tout entier m strictement supérieur à 1.

Ebauche d'une preuve

Chaque terme $\frac{1}{n^m}$ de la série ayant pour somme $\zeta(m)$ admet un développement factoriel fini, de sorte que :

$$\zeta(m) = \sum_{n=1}^{\infty} \frac{1}{n^m} = 1.1! + 0.0! + \sum_{n=2}^{\infty} \left(\frac{a_{n,1}}{1!} + \frac{a_{n,2}}{2!} + \ldots + \frac{a_{n,p_n}}{p_n!} \right) = 1.1! + 0.0! + \sum_{i=1}^{\infty} \frac{a_i}{i!}$$

Il s'agit d'établir que les coefficients a_i ne sont pas tous nuls ou tous égaux à $i - 1$ à partir d'un certain rang.

- Lorsque n est un nombre premier p supérieur à m ($n = p \geq m$), le développement factoriel de la fraction $\frac{1}{p^m}$ a une profondeur égale à mp :

  $$\frac{1}{p^m} = \frac{0}{1!} + \frac{0}{2!} + \ldots + \frac{r}{(mp)!} \qquad , \; 0 < r < (mp - 1)$$

  où r est le reste de la division entière de $\frac{(mp)!}{p^m}$ par mp

  L'ensemble des nombres premiers est infini, par conséquent les termes $\frac{1}{p^m}$, p premier, de la série $\left(\sum \frac{1}{n^m}\right)$ donneront lieu à des développements factoriels de profondeur aussi grande que l'on veut :

  $$\forall k > 0 \, , \exists \, p : \qquad mp > k \;\Rightarrow\; \frac{1}{(mp)!} < \frac{1}{k!}$$

- Entre 2 nombres premiers successifs p et p' , aucun des développements factoriels des termes $\frac{1}{n^m}$ vérifiant $p < n < p'$ n'atteint la profondeur mp (a fortiori mp') puisque tous les facteurs premiers de n sont inférieurs à p.

- D'après la loi de raréfaction des nombres premiers, l'intervalle séparant p de p' peut être rendu arbitrairement grand, par suite le coefficient a_{mp} du terme $\frac{a_{mp}}{(mp)!}$ dans le développement factoriel de $\zeta(m)$ tend vers r quand le nombre premier p tend vers l'infini.Comme r ≠ 0 et $r \neq mp - 1$, $\zeta(m)$ satisfait le critère factoriel d'irrationalité d'un nombre réel, et donc $\zeta(m)$ est irrationnel.

Cas particulier : Pour $m = 2$, $\zeta(2) = \sum_{n=1}^{+\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$ est irrationnel, donc π est irrationnel.

Mathématique & Informatique

Conclusion

Tout nombre réel x positif s'écrit sous la forme d'une série factorielle :

$$x = \sum_{k=-\infty}^{k=+q} n_k\,(k)!$$

La notation $(k)!$ désigne la factorielle d'un entier relatif $k \in \mathbb{Z}$: $\quad (k)! = \begin{cases} |k|! & \text{si } k \geq 0 \\ \dfrac{1}{|k|!} & \text{si } k < 0 \end{cases}$

L'ensemble des $(k)!$ lorsque k décrit $\mathbb{Z}$ constitue la base factorielle, notée $\mathbb{Z}!$

$$\mathbb{Z}! = \{ \ldots, (-n)!, \ldots, (-3)!, (-2)!, (-1)!, 0!, 1!, 2!, 3!, \ldots, n!, \ldots \}$$

Les entiers naturels n_k sont les composantes du nombre réel x dans $\mathbb{Z}!$. On a les encadrements :

$$0 \leq n_k \leq |k| \quad \text{si } k \geq 0 \quad \text{et} \quad 0 \leq n_k < |k| \quad \text{si } k < 0$$

Un nombre x est rationnel si et seulement si, à partir d'un certain rang $k < 0$, les coefficients n_k sont tous nuls ou vérifient tous la relation $n_k = |k| - 1$. La série factorielle se ramène alors à une somme factorielle et il existe un entier p tel que :

$$x = \frac{a}{b} = \sum_{k=-p}^{k=+q} n_k\,(k)!$$

Dans le cas contraire, x est un nombre irrationnel.
On dispose donc d'un critère simple pour vérifier si un nombre réel est rationnel ou s'il est irrationnel (plus simple que la recherche de la périodicité de son développement en base N).
Pour un nombre réel y strictement négatif, on pose $y = -x$ et on effectue le développement factoriel de $x > 0$.
Un nombre rationnel admet un développement factoriel de profondeur p finie alors qu'un nombre irrationnel a toujours un développement factoriel de profondeur infinie. En ce sens, la notion d'irrationalité des nombres découle de celle d'infini.

Puisqu'à chaque somme $\sum_{k=-p}^{k=+q} n_k\,(k)!$ on peut associer une infinité de séries $\sum_{k=-\infty}^{k=+q} n_k\,(k)! = \sum_{k=-p}^{k=+q} n_k\,(k)! + \sum_{k=-\infty}^{k=-(p+1)} n_k\,(k)!$,

où $\sum_{k=-\infty}^{k=-(p+1)} n_k\,(k)!$ représente un nombre irrationnel, on peut dire qu'il y a plus de nombres irrationnels que de nombres rationnels.

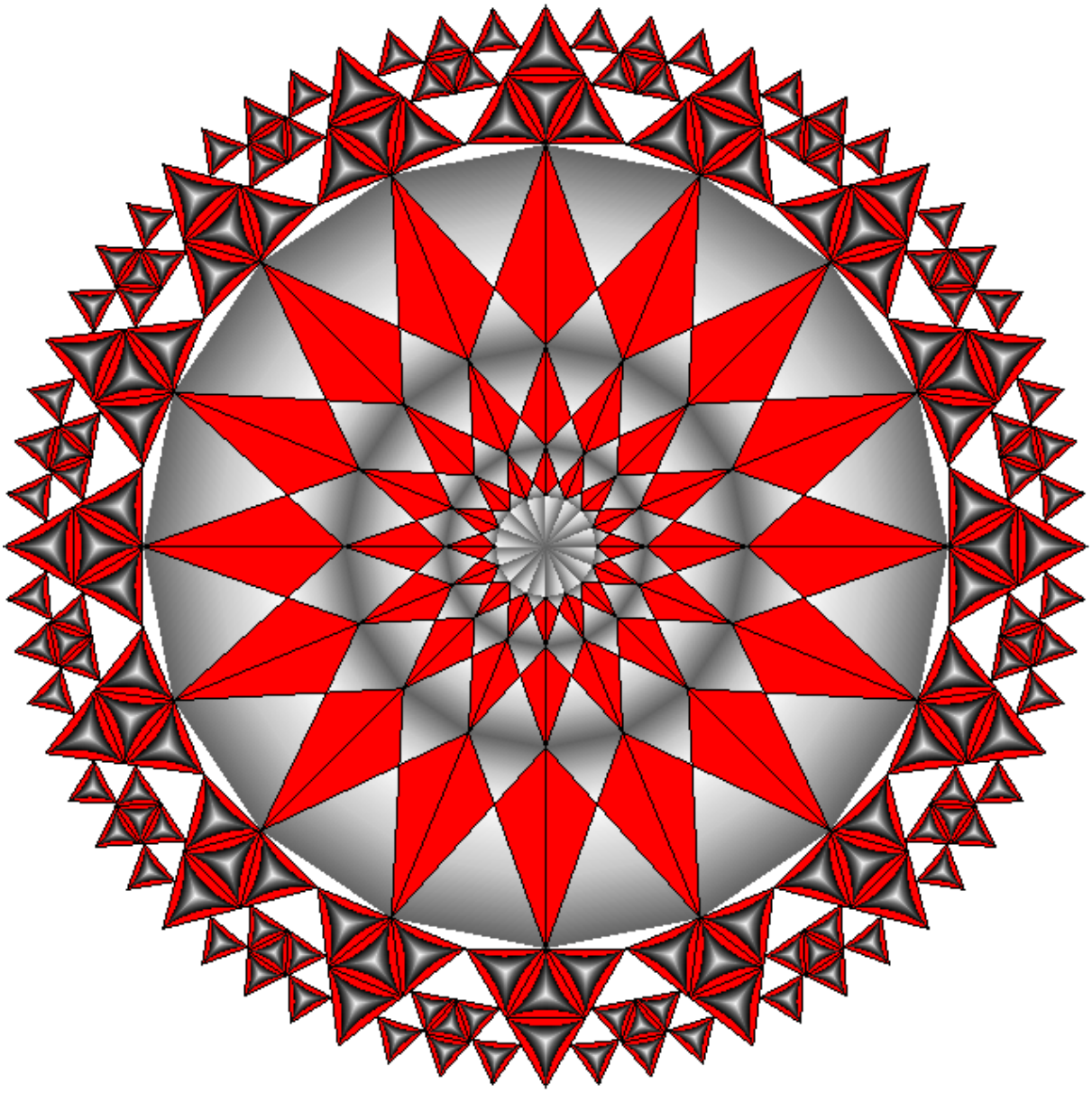
L'ensemble des nombres réels est désigné par la lettre $\mathbb{R}$, l'ensemble des nombres rationnels par la lettre $\mathbb{Q}$ ($\mathbb{Q}$ pour quotient).
Désignons l'ensemble des nombres irrationnels par la lettre $\mathbb{I}$ ($\mathbb{I} = \mathbb{R} - \mathbb{Q}$).

Les cardinaux de ces 3 ensembles vérifient la relation : $\quad \text{Card}(\mathbb{Q}) < \text{Card}(\mathbb{I}) < \text{Card}(\mathbb{R})$

L'infini rationnel est noté $\sum_{k=-p}^{k=+\infty} n_k\,(k)!$ et l'infini irrationnel (plus grand) $\sum_{k=-\infty}^{k=+\infty} n_k\,(k)!$

Le système de numération de base factorielle ne dépend pas d'un nombre particulier, contrairement au système de numération de base N qui est lié au nombre N. En base N, les nombres sont représentables en utilisant un jeu de N caractères, alors qu'en base factorielle il faut une infinité de symboles différents pour pouvoir écrire tous les nombres.
En fait, le système de numération de base factorielle englobe tous les systèmes de numération de base N.

Mathématique & Informatique



Equations algébriques en nombres entiers

On se propose de résoudre en nombre entiers une équation polynomiale diophantienne du type :

$$a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0 = 0 \quad , a_i \in \mathbb{Z}$$

En fait, le but du calcul est de trouver la valeur exacte des racines lorsque les coefficients sont de grands entiers.
Etudions les cas n = 1, n = 2, et n = 3. Tout d'abord, rappelons quelques résultats.

Estimation de **Cauchy**

Soit une équation algébrique de degré n à coefficients réels :

$$a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0 = 0$$

Le module de toute racine x vérifie :

$$(1 + B/|a_0|)^{-1} < |x| < (1 + A/|a_n|)$$

où :

$A = Max(|a_0|, |a_1|, ... , |a_{n-1}|)$
$B = Max(|a_1|, |a_2|, ... , |a_n|)$

NB.

- Si x = a + ib est un nombre complexe, alors $|x| = (a^2 + b^2)^{1/2}$
- L'encadrement de |x| est peu précis en général, mais il est facile à calculer

Propriété

Une racine entière divise nécessairement le terme constant d'un polynôme à coefficients entiers.

$P(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0 \ , a_i \in \mathbb{Z}$
$P(x_0) = 0 \ , x_0 \in \mathbb{Z} \Rightarrow x_0 \mid a_0$

Equation du premier degré : $ax + b = 0$

On a immédiatement : $x = - b/a$
Soit r le reste de la division entière de -b par a
L'équation admet une solution entière si r = 0

```
function x1 = DEGRE1(a,b)
  [x1,r1] = DIVISION(MOINS(b),a)
  if isequal(r1,[0,0]) then
    x1 = list(%t,x1)
  else
    x1 = list(%f,x1)
  end
endfunction
```

Equation du deuxième degré : $ax^2 + bx + c = 0$

Le signe du discriminant $\Delta = b^2 - 4ac$ indique le nombre de racines :

$\Delta < 0$: 0 racine réelle
$\Delta = 0$: 1 racine double : $x = - b/2a$
$\Delta > 0$: 2 racines distinctes : $x_1 = (- b - \sqrt{\Delta})/2a$, $x_2 = (- b + \sqrt{\Delta})/2a$

La valeur de $\sqrt{\Delta}$ s'obtient avec la fonction Scilab *intsqrt* vue précédemment.

```
function [x1,x2] = DEGRE2(a,b,c)
  function y = f(x)
    y = MULTIPLICATION(a,x)
    y = ADDITION(y,b)
    y = MULTIPLICATION(y,x)
    y = ADDITION(y,c)
  endfunction
  t1 = MULTIPLICATION(b,b)
  t2 = MULTIPLICATION(a,c)
  t3 = MULTIPLICATION(t2,[1,4])
  delta = SOUSTRACTION(t1,t3)
  select delta(1)
    case -1
      x1 = list(%f,[])
      x2 = list(%f,[])
    case 0
      A = MULTIPLICATION(a,[1,2])
      B = b
      x1 = DEGRE1(A,B)
      x2 = x1
    case +1
      t1 = [1,intsqrt(VALEUR(delta))]
      t2 = SOUSTRACTION(MOINS(b),t1)
      t3 = ADDITION(MOINS(b),t1)
      t4 = MULTIPLICATION(a,[1,2])
      [x1,r1] = DIVISION(t2,t4)
      [x2,r2] = DIVISION(t3,t4)
      if isequal(f(x1),[0,0]) then
        x1 = list(%t,x1)
      else
        x1 = list(%f,x1)
      end
      if isequal(f(x2),[0,0]) then
        x2 = list(%t,x2)
      else
        x2 = list(%f,x2)
      end
      if COMPARAISON(x1(2),x2(2)) > 0 then
        x0 = x2 ; x2 = x1 ; x1 = x0
      end
  end
endfunction
```

Equation du troisième degré : $ax^3 + bx^2 + cx + d = 0$

La fonction $f(x) = ax^3 + bx^2 + cx + d$ varie continûment de $-\infty$ à $+\infty$ lorsque x décrit $\mathbb{R}$. Par conséquent, l'équation $ax^3 + bx^2 + cx + d = 0$ admet au moins une racine réelle dans l'intervalle $[-M , +M]$ avec $M = 1 + Max(|b|,|c|,|d|)/|a|$

Recherchons la première racine entière x_1 en étudiant la dérivée $f'(x) = 3ax^2 + 2bx + c$

- Si $f'(x)$ ne s'annule pas ou s'annule une fois, alors x_1 est recherchée dans $[-M , +M]$
- Si $f'(x)$ s'annule 2 fois en X_1 et X_2 ($X_1 < X_2$), alors x_1 est recherchée dans $[-M , X_1]$ ou $[X_1 , X_2]$ ou $[X_2 , +M]$

Pour calculer x_1, utilisons la méthode de dichotomie qui ne nécessite pas que la racine soit séparée dans l'intervalle de recherche. Sans être le plus rapide, le processus de dichotomie demeure un algorithme simple et efficace.

Après avoir obtenu la première racine x_1, on divise le polynôme $ax^3 + bx^2 + cx + d$ par $(x - x_1)$ pour trouver les autres solutions possibles. On a :

$$ax^3 + bx^2 + cx + d = 0$$
$$ax_1^3 + bx_1^2 + cx_1 + d = 0$$

Soustrayons membre à membre et factorisons :

$$a(x - x_1)(x^2 + x_1x + x_1^2) + b(x - x_1)(x + x_1) + c(x - x_1) = 0$$
$$a(x^2 + x_1x + x_1^2) + b(x + x_1) + c = 0$$
$$ax^2 + (ax_1 + b)x + (ax_1^2 + bx_1 + c) = 0$$

On aboutit à une équation du second degré :

$$Ax^2 + Bx + C = 0$$

avec :

$$A = a$$
$$B = ax_1 + b$$
$$C = ax_1^2 + bx_1 + c$$

Remarque

Le procédé se généralise. Pour résoudre une équation de degré n, on recherche une première racine x_1, puis on divise le polynôme par $(x - x_1)$ pour se ramener au degré n-1.
Identité remarquable : $x^k - x_1^k = (x - x_1)(x^{k-1} + x_1x^{k-2} + x_1^2x^{k-3} + ... + x_1^{k-2}x + x_1^{k-1})$

```
function [x1,x2,x3] = DEGRE3(a,b,c,d)
  function y = f(x)
    y = MULTIPLICATION(a,x)
    y = ADDITION(y,b)
    y = MULTIPLICATION(y,x)
    y = ADDITION(y,c)
    y = MULTIPLICATION(y,x)
    y = ADDITION(y,d)
  endfunction
  function x = dichotomie(inf,sup)
    x = []
    mil = sup
    loop = %t
    while loop
      y1 = f(inf)
      y2 = f(mil)
      select SIGNE(y1)*SIGNE(y2)
        case 0
          if SIGNE(y1) == 0 then
            x = inf
          else
            x = mil
          end
          return
        case +1
          inf = ADDITION(mil,[1,1])
        case -1
          sup = SOUSTRACTION(mil,[1,1])
      end
      t = ADDITION(inf,sup)
      [mil,r] = DIVISION(t,[1,2])
      loop = COMPARAISON(inf,sup) <= 0
    end
  endfunction
  absa = ABSOLU(a)
  absb = ABSOLU(b)
  absc = ABSOLU(c)
  absd = ABSOLU(d)
  Max = absb
  if COMPARAISON(absc,Max) > 0 then
    Max = absc
  end
  if COMPARAISON(absd,Max) > 0 then
    Max = absd
  end
  [t,r] = DIVISION(Max,absa)
  M = ADDITION(t,[1,1])
```

```
  A = MULTIPLICATION(a,[1,3])
  B = MULTIPLICATION(b,[1,2])
  C = c
  [X1,X2] = DEGRE2(A,B,C)
  if isequal(X1,X2) then
    inf = MOINS(M) ; sup = M
    x1 = dichotomie(inf,sup)
  else
    inf = MOINS(M) ; sup = X1(2)
    x1 = dichotomie(inf,sup)
    if x1 == [] then
      inf = X1(2) ; sup = X2(2)
      x1 = dichotomie(inf,sup)
    end
    if x1 == [] then
      inf = X2(2) ; sup = M
      x1 = dichotomie(inf,sup)
    end
  end
  if x1 <> [] then
    x1 = list(%t,x1)
  else
    x1 = list(%f,[])
    x2 = list(%f,[])
    x3 = list(%f,[])
    return
  end
  A = a
  t = MULTIPLICATION(A,x1(2))
  B = ADDITION(t,b)
  t = MULTIPLICATION(B,x1(2))
  C = ADDITION(t,c)
  [x2,x3] = DEGRE2(A,B,C)
  if x1(1) & x2(1) & x3(1) then
    if COMPARAISON(x1(2),x2(2)) > 0 then
      x0 = x2 ; x2 = x1 ; x1 = x0
    end
    if COMPARAISON(x2(2),x3(2)) > 0 then
      x0 = x3 ; x3 = x2 ; x2 = x0
    end
    if COMPARAISON(x1(2),x2(2)) > 0 then
      x0 = x2 ; x2 = x1 ; x1 = x0
    end
  end
endfunction
```

EXEMPLE 1

Résoudre en nombres entiers l'équation du premier degré :

$$ax + b = 0$$

avec :

a = - 64567801249804197664025779991 27
b = 14888321313216022035890137256385265002362123180777

On écrit le programme Scilab :

```
a = [-1,6,4,5,6,7,8,0,1,2,4,9,8,0,4,1,9,7,6,6,4,0,2,5,7,7,9,9,9,1,2,7] ;
b = [1,1,4,8,8,8,3,2,1,3,1,3,2,1,6,0,2,2,0,3,5,8,9,0,1,3,7,2,5,6,3,8,5,2,6,5,0,0,2,3,6,2,1,2,3,1,8,0,7,7,7] ;

x1 = DEGRE1(a,b) ;
AFFICHE(list(x1))
```

La solution s'affiche à la Console :

```
x1 = 2305843009213693951
```

Remarque

La fonction *AFFICHE(X)* permet de visualiser à l'écran une liste $X = list(x_1, x_2, \ldots, x_n)$ de n solutions au format :

- $x_i = v_i$ où v_i est un vecteur
- $x_i = list(v_i, w_i)$ où v_i et w_i sont deux vecteurs
- $x_i = list(b_i, v_i)$ où b_i est un booléen et v_i est un vecteur

```
function AFFICHE(X)
  i = 0
  for k = 1:length(X)
    if typeof(X(k)) == "constant" then
      i = i+1
      printf('\n '), ECRIRE(X(k))
    elseif typeof(X(k)) == "list" then
      if typeof(X(k)(1)) == "constant" then
        i = i+1
        printf('\n '), ECRIRE(X(k))
      elseif typeof(X(k)(1)) == "boolean" then
        if X(k)(1) then
          i = i+1
          printf('\n '), ECRIRE(X(k)(2))
        end
      end
    end
  end
  if i == 0 then
    printf('\n VIDE')
  end
endfunction
```

EXEMPLE 2

Résoudre en nombres entiers l'équation du deuxième degré :

$$ax^2 + bx + c = 0$$

avec :

a = 25062009
b = 1818997711093403040202607753368345516986086378834076992757 3
c = 1123962191201570048603531263542706699821223928795641742777180064

On écrit le programme Scilab :

```
a = [1,2,5,0,6,2,0,0,9] ;
b = [1,1,8,1,8,9,9,7,7,1,1,0,9,3,4,0,3,0,4,0,2,0,2,6,0,...
     7,7,5,3,3,6,8,3,4,5,5,1,6,9,8,6,0,8,6,3,7,8,8,3,4,...
     0,7,6,9,9,2,7,5,7,3] ;
c = [1,1,1,2,3,9,6,2,1,9,1,2,0,1,5,7,0,0,4,8,6,0,3,5,3,...
     1,2,6,3,5,4,2,7,0,6,6,9,9,8,2,1,2,2,3,9,2,8,7,9,5,...
     6,4,1,7,4,2,7,7,7,1,8,0,0,6,4] ;

[x1,x2] = DEGRE2(a,b,c) ;
AFFICHE(list(x1,x2))
```

L'unique solution s'affiche à la Console :

```
x1 = - 725798842021564608089721679282912043079262472006621
```

Remarque

- Dans $\mathbb{R}$, l'équation de l'exemple 2 a un discriminant $\Delta = b^2 - 4ac > 0$, donc elle possède 2 racines réelles.
  Dans $\mathbb{Z}$, une seule racine est entière.

- Lorsque $\Delta = b^2 - 4ac < 0$, l'équation du second degré $ax^2 + bx + c = 0$ possède 2 racines complexes conjugées :

$$x = \frac{-b \pm i\sqrt{|b^2 - 4ac|}}{2a}$$

EXEMPLE 3

Résoudre en nombres entiers l'équation du troisième degré :

$$ax^3 + bx^2 + cx + d = 0$$

avec :

a = 3924587312411024698024957361
b = - 22292143957546572530306471911847353811171754403801165416541853557265414
c = - 94210154314772447147305842364160798953752307483338373541992506368293 22
d = - 88602358268895826264494930457551844829476969046348785791635561415506 35

On écrit le programme Scilab :

```
a = [1,3,9,2,4,5,8,7,3,1,2,4,1,1,0,2,4,6,9,8,0,2,4,9,5,7,3,6,1] ;
b = [-1,2,2,2,9,2,1,4,3,9,5,7,5,4,6,5,7,2,5,3,0,3,0,6,4,7,1,9,...
     1,1,8,4,7,3,5,3,8,1,1,1,7,1,7,5,4,4,0,3,8,0,1,1,6,5,4,1,...
     6,5,4,1,8,5,3,5,5,7,2,6,5,4,1,4] ;
c = [-1,9,4,2,1,0,1,5,4,3,1,4,7,7,2,4,4,7,1,4,7,3,0,5,8,4,2,3,...
     6,4,1,6,0,7,9,8,9,5,3,7,5,2,3,0,7,4,8,3,3,3,8,3,7,3,5,4,...
     1,9,9,2,5,0,6,3,6,8,2,9,3,2,2] ;
d = [-1,8,8,6,0,2,3,5,8,2,6,8,8,9,5,8,2,6,2,6,4,4,9,4,9,3,0,4,...
     5,7,5,5,1,8,4,4,8,2,9,4,7,6,9,6,9,0,4,6,3,4,8,7,8,5,7,9,...
     1,6,3,5,5,6,1,4,1,5,5,0,6,3,5] ;

[x1,x2,x3] = DEGRE3(a,b,c,d) ;
AFFICHE(list(x1,x2,x3))
```

L'unique solution s'affiche à la Console :

```
x1 = 5680124350157890215015457890256688801475383
```

Remarque

Dans $\mathbb{R}$, l'équation de l'exemple 3 se ramène à une équation du type $X^3 + pX + q = 0$, en posant $x = X - b/3a$. Le discriminant $\Delta = 4p^3 + 27q^2 > 0$ indique qu'il y a une seule racine réelle X_1 qui peut s'exprimer en utilisant la formule de **Cardan**.

$$X_1 = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}$$

Dans ce cas, on a en plus 2 racines complexes conjuguées X_2 et X_3. Appelons u et v les 2 radicaux cubiques :

$$X_2 = -\frac{1}{2}(u+v) + \frac{1}{2}i\sqrt{3}(u-v)$$
$$X_3 = -\frac{1}{2}(u+v) - \frac{1}{2}i\sqrt{3}(u-v)$$

EXEMPLE 4

Résoudre en nombres entiers l'équation du troisième degré :

$$ax^3 + bx^2 + cx + d = 0$$

avec :

a = 1
b = - 24602497019894315673415298743296186483 96
c = 345688007201682950814010852090964633489554620214228914914244
d = - 12143096514394389068068267260589746039339918860248695704861704090639435116337120

On écrit le programme Scilab :

```
a = [1,1] ;
b = [-1,2,4,6,0,2,4,9,7,0,1,9,8,9,4,3,1,5,6,7,3,4,1,5,2,9,8,7,4,3,2,9,6,1,8,6,4,8,3,9,6] ;
c = [1,3,4,5,6,8,8,0,0,7,2,0,1,6,8,2,9,5,0,8,1,4,0,1,0,8,5,2,0,9,0,9,6,4,6,3,3,4,8,9,5,5,...
     4,6,2,0,2,1,4,2,2,8,9,1,4,9,1,4,2,4,4] ;
d = [-1,1,2,1,4,3,0,9,6,5,1,4,3,9,4,3,8,9,0,6,8,0,6,8,2,6,7,2,6,0,5,8,9,7,4,6,0,3,9,3,3,9,...
      9,1,8,8,6,0,2,4,8,6,9,5,7,0,4,8,6,1,7,0,4,0,9,0,6,3,9,4,3,5,1,1,6,3,3,7,1,2,0] ;

[x1,x2,x3] = DEGRE3(a,b,c,d) ;
AFFICHE(list(x1,x2,x3))
```

Les 3 solutions s'affichent à la Console :

```
x1 = 70254658891361571658
x2 = 70254658891361571658
x3 = 24602497019894315672010205565468955050 80
```

Remarque

Dans $\mathbb{R}$, l'équation de l'exemple 4 se ramène à une équation du type $X^3 + pX + q = 0$, en posant $x = X - b/3a$.
Le discriminant $\Delta = 4p^3 + 27q^2 = 0$ indique qu'il y a 3 racines réelles : une double $X_1 = X_2$ et une simple X_3.

$$X_1 = X_2 = -\frac{3q}{2p} \quad , \quad X_3 = \frac{3q}{p}$$

Dans le cas particulier où $\Delta = 0$ avec $p = 0$ (donc $q = 0$), on a une racine triple $X_1 = X_2 = X_3 = 0$, ce qui correspond à une factorisation de l'équation de départ sous la forme d'un cube : $a(x + b/3a)^3 = 0$.

EXEMPLE 5

Résoudre en nombres entiers l'équation du troisième degré :

$$ax^3 + bx^2 + cx + d = 0$$

avec :

a = 1
b = - 9863201133054933449124188336927095090
c = 8458134725036117551808433736764548822210087217530179236712480261 7
d = 3106171615133851036607398188816150989798561126746888780462035079591430696946219681913445 08

On écrit le programme Scilab :

```
a = [1,1] ;
b = [-1,9,8,6,3,2,0,1,1,3,3,0,5,4,9,3,3,4,4,9,1,2,4,1,8,...
     8,3,3,6,9,2,7,0,9,5,0,9,0] ;
c = [1,8,4,5,8,1,3,4,7,2,5,0,3,6,1,1,7,5,5,1,8,0,8,4,3,3,...
     7,3,6,7,6,4,5,4,8,8,2,2,2,1,0,0,8,7,2,1,7,5,3,0,1,7,...
     9,2,3,6,7,1,2,4,8,0,2,6,1,7] ;
d = [1,3,1,0,6,1,7,1,6,1,5,1,3,3,8,5,1,0,3,6,6,0,7,3,9,8,...
     1,8,8,8,1,6,1,5,0,9,8,9,7,9,8,5,6,1,1,2,6,7,4,6,8,8,...
     8,7,8,0,4,6,2,0,3,5,0,7,9,5,9,1,4,3,0,6,9,6,9,4,6,2,...
     1,9,6,8,1,9,1,3,4,4,5,0,8] ;

[x1,x2,x3] = DEGRE3(a,b,c,d) ;
AFFICHE(list(x1,x2,x3))
```

Les 3 solutions s'affichent à la Console :

```
x1 = - 367083600116538902010586 1
x2 = 857911669611549962333028928 4
x3 = 986320112447948758900985410261691166 7
```

Remarque

Dans $\mathbb{R}$, l'équation de l'exemple 5 se ramène à une équation du type $X^3 + pX + q = 0$, en posant $x = X - b/3a$.
Le discriminant $\Delta = 4p^3 + 27q^2 < 0$ indique qu'il y a 3 racines réelles distinctes X_1, X_2, X_3 qui peuvent s'exprimer en utilisant des cosinus.

$$X_1 = \sqrt{-\frac{4p}{3}}\cos\left[\frac{1}{3}\,Arc\cos\left(-q\sqrt{-\frac{27}{4\,p^3}}\right)\right]$$

$$X_2 = \sqrt{-\frac{4p}{3}}\cos\left[\frac{1}{3}\,Arc\cos\left(-q\sqrt{-\frac{27}{4\,p^3}}\right)+\frac{2\pi}{3}\right]$$

$$X_3 = \sqrt{-\frac{4p}{3}}\cos\left[\frac{1}{3}\,Arc\cos\left(-q\sqrt{-\frac{27}{4\,p^3}}\right)+\frac{4\pi}{3}\right]$$

Application à la résolution de systèmes arithmétiques en nombres entiers

Le problème de base consiste à trouver 2 nombres x et y connaissant leur somme S et leur produit P :

$S = x + y$
$P = xy$

On écrit : $P = xy = x(S - x) = Sx - x^2 \quad \Rightarrow \quad x^2 - Sx + P = 0$

x est solution d'une équation du second degré.
y est donné par la relation : $y = S - x$
Dans $\mathbb{R}$:

$$x = \frac{S \pm \sqrt{S^2 - 4P}}{2} \qquad y = \frac{S \mp \sqrt{S^2 - 4P}}{2}$$

Résolvons un ensemble de problèmes élémentaires qui se ramènent au problème ci-dessus.
Ils n'admettent pas toujours des solutions entières, mais si c'est le cas, alors on peut calculer leurs valeurs.

Problème 1

$x^2 + y^2 = a$
$x + y = b$

$(x + y)^2 = (x^2 + y^2) + 2xy$
$x + y = b$

$x + y = S = b$
$xy = P = (b^2 - a)/2$

Problème 2

$x^3 + y^3 = a$
$x + y = b$

$(x + y)^3 = (x^3 + y^3) + 3xy(x + y)$
$x + y = b$

$x + y = S = b$
$xy = P = (b^3 - a)/3b$

Problème 3

$x^2 + y^2 = a$
$xy = b$

$(x + y)^2 = (x^2 + y^2) + 2xy$
$xy = b$

$x + y = S = (a^2 + 2b)^{1/2}$
$xy = P = b$

Problème 4

$x^3 + y^3 = a$
$xy = b$

$x^3 + y^3 = a$
$x^3 y^3 = b^3$

$X + Y = S = a$
$XY = P = b^3$

$x = X^{1/3}$
$y = Y^{1/3}$

Problème 5

$x + y = a$
$x^2y + xy^2 = b$

$x + y = a$
$xy(x + y) = b$

$x + y = S = a$
$xy = P = b/a$

Problème 6

$xy = a$
$x^2y + xy^2 = b$

$xy = a$
$xy(x + y) = b$

$x + y = S = b/a$
$xy = P = a$

Problème 7

$x^2 + y^2 = a$
$x^2y + xy^2 = b$

$(x + y)^2 = (x^2 + y^2) + 2xy$
$xy(x + y) = b$

$X^2 = a + 2Y$
$XY = b$

$X^3 - aX - 2b = 0$
$2Y^3 + aY^2 - b^2 = 0$

$x + y = S = X$
$xy = P = Y$

Mathématique & Informatique

Problème 8

$x^3 + y^3 = a$
$x^2y + xy^2 = b$

$(x + y)^3 = (x^3 + y^3) + 3(x^2y + xy^2)$
$xy(x + y) = b$

$x + y = S = (a + 3b)^{1/3}$
$xy = P = b(a + 3b)^{-1/3}$

Problème 9

$x^2 + y^2 = a$
$x^3 + y^3 = b$

$(x + y)^2 = (x^2 + y^2) + 2xy$
$(x + y)^3 = (x^3 + y^3) + 3xy(x + y)$

$X^2 = a + 2Y$
$X^3 = b + 3XY$

$X^3 - 3aX + 2b = 0$
$2Y^3 - 3aY^2 + (a^3 - b^2) = 0$

$x + y = S = X$
$xy = P = Y$

Problème 10

$x + y = a$
ppcm(x,y) = b

$x + y = a$
pgcd(x,y) = pgcd(a,b) = c

$x + y = S = a$
$xy = P = bc$

EXEMPLE 1

Trouver les 2 nombres entiers x et y tels que :

x + y = 333429409135499518984192192101
ppcm(x,y) = - 928797537770731335811607418196200550961139535191 6

Programme de résolution :

```
a = [1,3,3,3,4,2,9,4,0,9,1,3,5,4,9,9,5,1,8,9,8,4,1,9,2,1,9,2,1,0,1] ;
b = [-1,9,2,8,7,9,7,5,3,7,7,7,0,7,3,1,3,3,5,8,1,1,6,0,7,4,1,8,1,9,6,2,0,0,5,5,0,9,6,1,1,3,9,5,3,5,1,9,1,6] ;
S = a ;
P = MULTIPLICATION(b,PGCD(a,b)) ;
[x1,x2] = DEGRE2(ENTIER(1),MOINS(S),P) ;
if x1(1) then
   x = x1(2) ;
   y = SOUSTRACTION(S,x) ;
   printf('\n x = '), ECRIRE(x)
   printf('\n y = '), ECRIRE(y)
else
   printf('\n Pas de solution')
end
```

Valeurs obtenues :

x = - 1293108322212363928140505516 47
y = 462740241356735911798242743748

EXEMPLE 2

Trouver les 2 nombres entiers x et y tels que :

$x^2 + y^2$ = 4976006798892438823449846787286413783113376896 25
$x^3 + y^3$ = 3510115827293573227252673198483670896964193390275040616644187345128173 17

Programme de résolution :

```
a = [1,4,9,7,6,0,0,6,7,9,8,8,9,2,4,3,8,8,2,3,4,4,9,8,4,6,7,8,7,2,8,6,4,1,3,7,8,3,1,1,3,3,7,6,8,9,6,2,5] ;
b = [1,3,5,1,0,1,1,5,8,2,7,2,9,3,5,7,3,2,2,7,2,5,2,6,7,3,1,9,8,4,8,3,6,7,0,8,9,6,9,6,4,1,9,3,3,9,0,2,7,...
     5,0,4,0,6,1,6,6,4,4,1,8,7,3,4,5,1,2,8,1,7,3,1,7] ;
A = ENTIER(1) ;
B = ENTIER(0) ;
C = MULTIPLICATION(a,ENTIER(-3)) ;
D = MULTIPLICATION(b,ENTIER(2)) ;
[X1,X2,X3] = DEGRE3(A,B,C,D) ;
LX = list(X1,X2,X3) ;
solution = %f ;
for i = 1:length(LX)
  X = LX(i) ;
  if ~X(1) then, continue, end
  S = X(2) ;
  A = ENTIER(2) ;
  B = MULTIPLICATION(S,ENTIER(-2)) ;
  C = SOUSTRACTION(PUISSANCE(S,2),a) ;
  [x1,x2] = DEGRE2(A,B,C) ;
  Lx = list(x1,x2) ;
  for j = 1:length(Lx)
    x = Lx(j) ;
    if ~x(1) then, continue, end
    x = x(2) ;
    y = SOUSTRACTION(S,x) ;
    printf('\n x = '), ECRIRE(x)
    printf('\n y = '), ECRIRE(y)
    solution = %t ;
  end
end
if ~solution then
  printf('\n Pas de solution')
end
```

Valeurs obtenues :

x = 24009891005986557
y = 705408165454046403520124

Mathématique & Informatique

Equation de degré n : $a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 = 0$

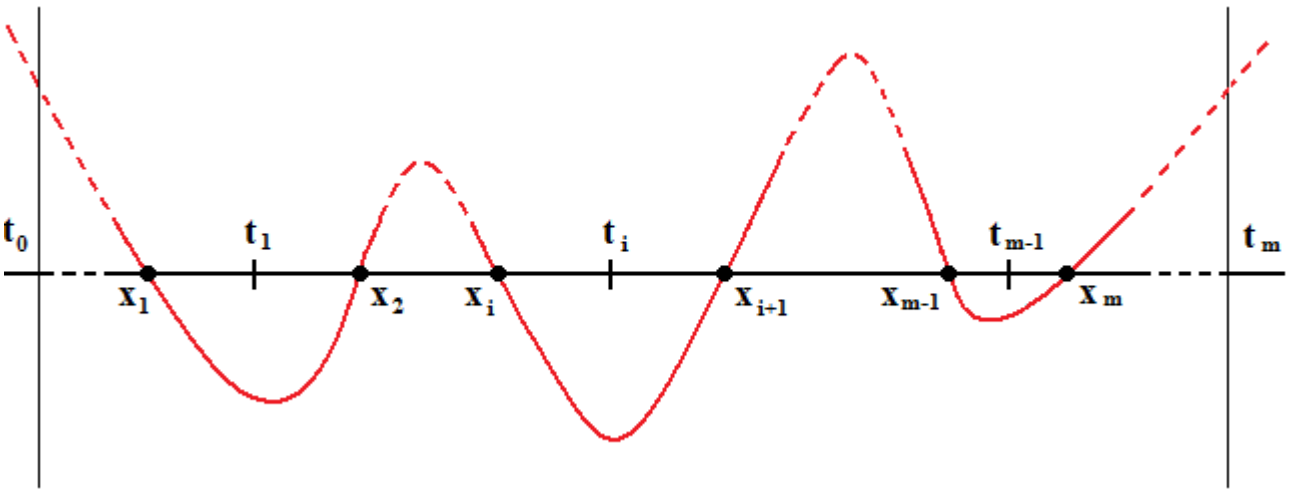
Posons : $P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$, $a_i \in \mathbb{Z}$
Pour résoudre en nombres entiers l'équation P(x) = 0, on détermine une suite d'intervalles [a,b] vérifiant P(a)P(b) < 0, puis on applique la méthode de dichotomie sur chacun de ces intervalles pour calculer la valeur exacte des racines.
Rappelons le théorème fondamental de l'algèbre : l'équation polynomiale P(x) = 0 admet n racines dans $\mathbb{C}$.
On a : $P(x) = a_n (x - x_1) \ldots (x - x_m)(x - x_{m+1}) \ldots (x - x_n)$
où $x_1, \ldots, x_m$ sont les m racines réelles de P et $x_{m+1}, \ldots, x_n$ sont les n-m racines complexes de P.

Hypothèses :

1) La fonction Scilab *roots*, basée sur le calcul des valeurs propres de la matrice compagnon du polynôme P, fournit une approximation suffisamment précise des racines réelles x_i pour pouvoir les séparer.
Limite de la fonction *roots* : $|x_i| \approx 10^{308}$
2) Les racines réelles x_i sont triées dans l'ordre croissant de leurs valeurs : $x_1 \leq x_2 \leq \ldots \leq x_m$

Considérons alors la suite des points t_i (i = 0, 1, ... , m) définie par :

$$t_0 = x_1 - |x_1|$$
$$t_i = \frac{x_i + x_{i+1}}{2} \qquad (0 < i < m)$$
$$t_m = x_m + |x_m|$$



On recherche si la racine x_i dans l'intervalle $[t_{i-1}, t_i]$ est un entier.
Le calcul est effectué par une fonction Scilab *racines* qui prend en paramètre d'entrée la liste des coefficients du polynôme P et renvoie en paramètre de sortie la liste **x** des racines entières du polynôme P.
Les racines multiples, lorsque P(x) ne change pas de signe, ne sont pas recherchées.

Méthode de dichotomie

Le polynôme P est continu sur [a,b] et P(a)P(b) < 0
Le processus itératif suivant donne une approximation à ε près de la racine x de P contenue dans l'intervalle [a,b]

```
répéter
|     x = (a + b)/2
|     si P(x)P(a) > 0 alors a = x sinon b = x
jusqu'à P(x) = 0 ou |b - a| < ε
```

```
function x = racines(P)

  function Y = tri(X)
    Y = []
    for i = 1:length(X)
     if imag(X(i)) == 0 then
       Y = [Y ; real(X(i))]
      end
    end
    Y = unique(Y)
  endfunction

  function I = intervalle(X)
    Y = list()
    for i = 1:length(X)
      if abs(X(i)) < 10^p then
        E = round(X(i))
        Y($+1) = ENTIER(E)
      else
        e = floor(log10(abs(X(i))))
        E = round(X(i)/10^(e-p))
        Y($+1) = [ENTIER(E),zeros(1,e-p)]
      end
    end
    I = list()
    for i = 1:length(Y)
      if isequal(f(Y(i)),[0,0]) then
        I($+1) = list(Y(i),Y(i))
        continue
      end
      if i == 1 then
        a = SOUSTRACTION(Y(1),ABSOLU(Y(1)))
      else
        t = ADDITION(Y(i-1),Y(i))
        [a,reste] = DIVISION(t,[1,2])
      end
      if i == length(Y) then
        b = ADDITION(Y($),ABSOLU(Y($)))
      else
        t = ADDITION(Y(i),Y(i+1))
        [b,reste] = DIVISION(t,[1,2])
      end
      I($+1) = list(a,b)
    end
  endfunction

  function y = f(x)
    y = [0,0]
    for i = 1:length(P)
      y = MULTIPLICATION(y,x)
      y = ADDITION(y,P(i))
    end
  endfunction
```

```
function r = dichotomie(a,b)
  r = []
  m = b
  y1 = f(a)
  loop = %t
  while loop
    y2 = f(m)
    select SIGNE(y1)*SIGNE(y2)
      case 0
        if SIGNE(y1) == 0 then
          r = a
        else
          r = m
        end
        return
      case +1
        a = ADDITION(m,[1,1])
        y1 = f(a)
      case -1
        b = SOUSTRACTION(m,[1,1])
    end
    [m,reste] = DIVISION(ADDITION(a,b),[1,2])
    loop = COMPARAISON(a,b) <= 0
  end
endfunction

p = 10
while length(P) > 1
  if isequal(P(1),[0,0]) then
    P(1) = null()
  else
    break
  end
end
x = poly(0,'x')
polynome = 0
for i = 1:length(P)
  polynome = polynome + nombre_relatif(P(i))*x^(length(P)-i)
end
X = roots(polynome,'e')
X = tri(X)
I = intervalle(X)
x = list()
for i = 1:length(I)
  a = I(i)(1)
  b = I(i)(2)
  r = dichotomie(a,b)
  if r <> [] then, x($+1) = r, end
end
endfunction
```

EXEMPLE 1

Le polynôme de **Wilkinson** d'ordre 15 est un polynôme unitaire qui admet pour racines les nombres entiers de 1 à 15

$$W_{15}(x) = x^{15} - 120x^{14} + 6580x^{13} - 218400x^{12} + 4899622x^{11} - 78558480x^{10} + 928095740x^{9} - 8207628000x^{8} + 54631129553x^{7} - 272803210680x^{6} + 1009672107080x^{5} - 2706813345600x^{4} + 5056995703824x^{3} - 6165817614720x^{2} + 4339163001600x - 1307674368000$$

Calculons les racines de $W_{15}(x)$ avec la fonction Scilab *roots* puis avec la fonction *racines* précédemment définie.

```
// Fonction roots
x = poly(0,'x') ;
W15 = x^15 - 120*x^14 + 6580*x^13- 218400*x^12 + 4899622*x^11 - 78558480*x^10 + 928095740*x^9 ...
      - 8207628000*x^8 + 54631129553*x^7 - 272803210680*x^6 + 1009672107080*x^5 - 2706813345600*x^4 ...
      + 5056995703824*x^3 - 6165817614720*x^2 + 4339163001600*x - 1307674368000 ;
r = roots(W15) ;
for i = 1:length(r), printf('\n r(%i) = ',i), printf('%.10f',r(n-i+1)), end
```

Résultat :

```
r(1) = 1.0000000000
r(2) = 2.0000000000
r(3) = 3.0000000000
r(4) = 3.9999999997
r(5) = 5.0000000028
r(6) = 5.9999999961
r(7) = 6.9999999387
r(8) = 8.0000004015
r(9) = 8.9999987777
r(10) = 10.0000022577
r(11) = 10.9999972833
r(12) = 12.0000021569
r(13) = 12.9999989029
r(14) = 14.0000003260
r(15) = 14.9999999567
```

```
// Fonction racines
exec('Scilab-Equations.sce',-1)
W15 = list(ENTIER(1),ENTIER(-120),ENTIER(6580),ENTIER(-218400),ENTIER(4899622), ...
           ENTIER(-78558480),ENTIER(928095740),ENTIER(-8207628000), ENTIER(54631129553), ...
           ENTIER(-272803210680),ENTIER(1009672107080),ENTIER(-2706813345600), ...
           ENTIER(5056995703824),ENTIER(-6165817614720),ENTIER(4339163001600),ENTIER(-1307674368000)) ;
r = racines(W15) ;
for i = 1:length(r), printf('\n r(%i) = ',i), ECRIRE(r(i)), end
```

Résultat :

```
r(1) = 1
r(2) = 2
r(3) = 3
r(4) = 4
r(5) = 5
r(6) = 6
r(7) = 7
r(8) = 8
r(9) = 9
r(10) = 10
r(11) = 11
r(12) = 12
r(13) = 13
r(14) = 14
r(15) = 15
```

EXEMPLE 2

Quelle est la valeur n qui vérifie l'égalité :

$$\sum_{i=0}^{n} i^5 = \begin{array}{l}24774999353870154146618684434078061286114697307368470882394034802674313283082051376823429100567343904021030506005303306547744080915296159213436189827452771510297876\end{array}$$

La formule de **Bernoulli** permet d'exprimer la somme en fonction de n :

$$\Sigma i^5 = n^6/6 + n^5/2 + 5n^4/12 - n^2/12$$

Notons N la valeur de Σi^5. L'inconnue n est solution de :

$$2n^6 + 6n^5 + 5n^4 - n^2 - 12N = 0$$

Programme :

```
exec("Scilab-Equations.sce",-1)
N = [1,2,4,7,7,4,9,9,9,3,5,3,8,7,0,1,5,4,1,4,...
     6,6,1,8,6,8,4,4,3,4,0,7,8,0,6,1,2,8,6,1,...
     1,4,6,9,7,3,0,7,3,6,8,4,7,0,8,8,2,3,9,4,...
     0,3,4,8,0,2,6,7,4,3,1,3,2,8,3,0,8,2,0,5,...
     1,3,7,6,8,2,3,4,2,9,1,0,0,5,6,7,3,4,3,9,...
     0,4,0,2,1,0,3,0,5,0,6,0,0,5,3,0,3,3,0,6,...
     5,4,7,7,4,4,0,8,0,9,1,5,2,9,6,1,5,9,2,1,...
     3,4,3,6,1,8,9,8,2,7,4,5,2,7,7,1,5,1,0,2,...
     9,7,8,7,6] ;
a = [1,2] ;
b = [1,6] ;
c = [1,5] ;
d = [0,0] ;
e = [-1,1] ;
f = [0,0] ;
g = MULTIPLICATION(N,[-1,1,2]) ;
x = racines(list(a,b,c,d,e,f,g)) ;
AFFICHE(x)
```

Résultat :

x1 = – 230158746545646545491652045 2
x2 = + 230158746545646545491652045 1

La solution est la racine entière positive :

n = 230158746545646545491652045 1

EXEMPLE 3

Rechercher les solutions entières de l'équation de degré 7 suivante :

$$x^7 - 413000860222\,x^6 - 173096078646203647841160 63\,x^5 + 302513259489769193770554351192232 4455\,x^4 + 731366169112371296498216349378161205946219985011 82\,x^3 - 12116725505234255348881244 1\,x^2 + 211759281642838435639388045834562711 36\,x + 511956318378659907548751444564712844182591031659152 = 0$$

On programme :

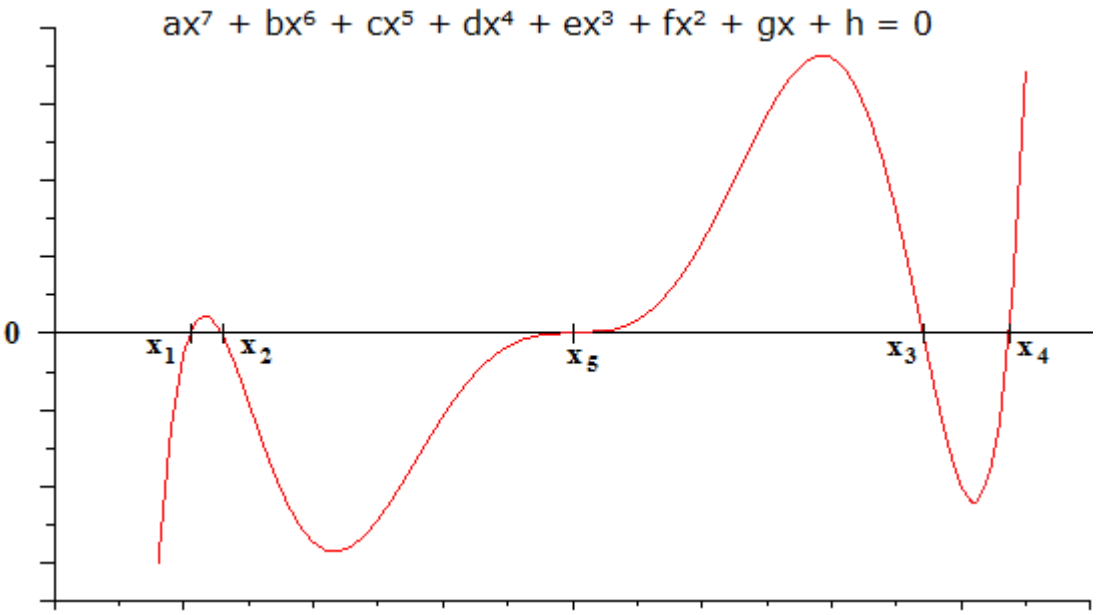
```
exec("Scilab-Equations.sce",-1)
a = [1,1] ;
b = [-1,4,1,3,0,0,0,8,6,0,2,2,2] ;
c = [-1,1,7,3,0,9,6,0,7,8,6,4,6,2,0,3,6,4,7,8,4,1,1,6,0,6,3] ;
d = [1,3,0,2,5,1,3,2,5,9,4,8,9,7,6,9,1,9,3,7,7,0,5,5,4,3,5,1,1,9,2,2,3,2,4,4,5,5] ;
e = [1,7,3,1,3,6,6,1,6,9,1,1,2,3,7,1,2,9,6,4,9,8,2,1,6,3,4,9,3,7,8,1,6,1,2,0,5,9,4,6,2,1,9,9,8,5,0,1,1,8,2] ;
f = [-1,1,2,1,1,6,7,2,5,5,0,5,2,3,4,2,5,5,3,4,8,8,8,1,2,4,4,1] ;
g = [1,2,1,1,7,5,9,2,8,1,6,4,2,8,3,8,4,3,5,6,3,9,3,8,8,0,4,5,8,3,4,5,6,2,7,1,1,3,6] ;
h = [1,5,1,1,9,5,6,3,1,8,3,7,8,6,5,9,9,0,7,5,4,8,7,5,1,4,4,4,5,6,4,7,1,2,8,4,4,1,8,2,5,9,1,0,3,1,6,5,9,1,5,2] ;
x = racines(list(a,b,c,d,e,f,g,h)) ;
AFFICHE(x)
```

On obtient 4 racines entières :

```
x1 = - 2960120685509
x2 = - 2706429567228
x3 =   2706429567228
x4 =   3373121545731
```

Remarque

Le polynôme de départ possède une cinquième racine réelle qui n'est pas entière : $x5 = -\sqrt[3]{7}$



EXEMPLE 4

Montrer que le système d'équations non linéaires d'inconnues a, b, c, d

$$\begin{cases} a + b + c + d = \alpha_1 \\ a^2 + b^2 + c^2 + d^2 = \alpha_2 \\ a^3 + b^3 + c^3 + d^3 = \alpha_3 \\ a^4 + b^4 + c^4 + d^4 = \alpha_4 \end{cases}$$

est équivalent à une équation du quatrième degré dont les racines sont a, b, c, d

$$\beta_4 x^4 + \beta_3 x^3 + \beta_2 x^2 + \beta_1 x + \beta_0 = 0$$

Application 1

Résoudre dans $\mathbb{C}$:

$$\begin{cases} a + b + c + d = 1 \\ a^2 + b^2 + c^2 + d^2 = 2 \\ a^3 + b^3 + c^3 + d^3 = 3 \\ a^4 + b^4 + c^4 + d^4 = 4 \end{cases}$$

Application 2

Résoudre dans $\mathbb{Z}$:

$$\begin{cases} a + b + c + d = 2305843008118644853 \\ a^2 + b^2 + c^2 + d^2 = 5316911983139663487994502847463640107 \\ a^3 + b^3 + c^3 + d^3 = 12259964326927110850916040266818187139766510415099134849 \\ a^4 + b^4 + c^4 + d^4 = 28269553036454149224292902704178252818625866027583444431216003086333580659 \end{cases}$$

La démonstration repose sur l'identité de **Newton** liant les sommes S_p des puissances p de n nombres $x_1, x_2, \ldots, x_n$ aux sommes σ_p des produits p à p de ces nombres.
On montre que :

$$S_p = \sum_{k=1}^{p-1} (-1)^{k-1} \sigma_k S_{p-k} + (-1)^{p-1} p \sigma_p \qquad (1 \le p \le n)$$

Connaissant les sommes de **Newton** $S_p = x_1^p + x_2^p + \ldots x_n^p$, on peut calculer de proche en proche la valeur des σ_p.
On dispose donc des coefficients de l'équation polynomiale de degré n dont les racines sont $x_1, x_2, \ldots, x_n$:

$$x^n - \sigma_1 x^{n-1} + \sigma_2 x^{n-2} + \cdots + (-1)^{n-1} \sigma_{n-1} x + (-1)^n \sigma_n = 0$$

Tout polynôme de degré n admet exactement n racines complexes (théorème fondamental de l'algèbre de **D'Alembert**) [(*)].
Dans $\mathbb{C}$, la solution $x_1, x_2, \ldots, x_n$ existe toujours et est unique.
Dans $\mathbb{Z}$, les nombres $x_1, x_2, \ldots, x_n$ sont des diviseurs de σ_n.

Cas particulier $n = 4$

On a :

$$S_1 = a + b + c + d \qquad \sigma_1 = a + b + c + d$$
$$S_2 = a^2 + b^2 + c^2 + d^2 \qquad \sigma_2 = ab + ac + ad + bc + bd + cd$$
$$S_3 = a^3 + b^3 + c^3 + d^3 \qquad \sigma_3 = abc + abd + acd + bcd$$
$$S_4 = a^4 + b^4 + c^4 + d^4 \qquad \sigma_4 = abcd$$

La formule générale de récurrence entre les S_p et les σ_p permet d'obtenir les relations suivantes :

$$\sigma_1 = S_1$$
$$\sigma_2 = \frac{S_1^2 - S_2}{2}$$
$$\sigma_3 = \frac{S_1^3 - 3S_1S_2 + 2S_3}{6}$$
$$\sigma_4 = \frac{S_1^4 - 6S_1^2S_2 + 8S_1S_3 + 3S_2^2 - 6S_4}{24}$$

D'où l'équation algébrique dont les racines sont a, b, c, d :

$$x^4 - \sigma_1 x^3 + \sigma_2 x^2 - \sigma_3 x + \sigma_4 = 0$$

Pour résoudre cette équation du 4^e degré dans $\mathbb{C}$, on utilise la fonction *roots* de Scilab.
Pour la résoudre dans $\mathbb{Z}$, on utilise la fonction *racines* du module Scilab-Equations.

[(*)] par racine complexe on entend racine réelle ou racine complexe avec partie imaginaire non nulle. Chaque racine est comptée avec son ordre de multiplicité. La démonstration rigoureuse de ce théorème (de 4 manières différentes) est due à **Gauss**.

Mathématique & Informatique

Application 1

$$\begin{cases} a + b + c + d = 1 \\ a^2 + b^2 + c^2 + d^2 = 2 \\ a^3 + b^3 + c^3 + d^3 = 3 \\ a^4 + b^4 + c^4 + d^4 = 4 \end{cases}$$

```
function X = resolution(S)
    S1 = S(1)
    S2 = S(2)
    S3 = S(3)
    S4 = S(4)
    // Calcul de s1, s2, s3, s4
    s1 = S1
    s2 = (S1^2 - S2)/2
    s3 = (S1^3 - 3*S1*S2 + 2*S3)/6
    s4 = (S1^4 - 6*S1^2*S2 + 8*S1*S3 + 3*S2^2 - 6*S4)/24
    // Calcul des racines
    x = poly(0,'x')
    P = x^4 - s1*x^3 + s2*x^2 - s3*x + s4
    X = roots(P)
endfunction

// Données
S = [1,2,3,4] ;
// Recherche des racines
X = resolution(S) ;
// Ecriture du résultat
for i = 1:length(X), printf('\n Nombre n° %i = ',i), disp(X(i)), end
```

Les nombres a, b, c, d sont les zéros du polynôme $P(x) = x^4 - x^3 - \frac{1}{2}x^2 - \frac{1}{6}x + \frac{1}{24}$

a = 1.4201621
b = 0.1567727
c = - 0.2884674 + 0.3223856i
d = - 0.2884674 – 0.3223856i

Application 2

$$\begin{cases} a + b + c + d = 2305843008118644853 \\ a^2 + b^2 + c^2 + d^2 = 5316911983139663487994502847463640107 \\ a^3 + b^3 + c^3 + d^3 = 12259964326927110850916040266818187139766510415099134849 \\ a^4 + b^4 + c^4 + d^4 = 28269553036454149224292902704178252818625866027583444431216003086333580659 \end{cases}$$

```
funcprot(0) ; lines(0)
exec("Scilab-Equations.sce",-1)

function X = resolution(S)
    S1 = S(1)
    S2 = S(2)
    S3 = S(3)
    S4 = S(4)
    // Calcul de s1
    s1 = S1
    // Calcul de s2
    s2 = PUISSANCE(S1,2)
    s2 = SOUSTRACTION(s2,S2)
    [s2,r] = DIVISION(s2,ENTIER(2))
    // Calcul de s3
    s3 = PUISSANCE(S1,3)
    s3 = SOUSTRACTION(s3,MULTIPLICATION(ENTIER(3),MULTIPLICATION(S1,S2)))
    s3 = ADDITION(s3,MULTIPLICATION(ENTIER(2),S3))
    [s3,r] = DIVISION(s3,ENTIER(6))
    // Calcul de s4
    s4 = PUISSANCE(S1,4)
    s4 = SOUSTRACTION(s4,MULTIPLICATION(ENTIER(6),MULTIPLICATION(MULTIPLICATION(S1,S1),S2)))
    s4 = ADDITION(s4,MULTIPLICATION(ENTIER(8),MULTIPLICATION(S1,S3)))
    s4 = ADDITION(s4,MULTIPLICATION(ENTIER(3),MULTIPLICATION(S2,S2)))
    s4 = SOUSTRACTION(s4,MULTIPLICATION(ENTIER(6),S4))
    [s4,r] = DIVISION(s4,ENTIER(24))
    // Calcul des racines
    P = list(ENTIER(1),MOINS(s1),s2,MOINS(s3),s4)
    X = racines(P)
endfunction

// Données
S1 = [1,2,3,0,5,8,4,3,0,0,8,1,1,8,6,4,4,8,5,3] ;
S2 = [1,5,3,1,6,9,1,1,9,8,3,1,3,9,6,6,3,4,8,7,9,9,4,5,0,2,8,4,7,4,6,3,6,4,0,1,0,7] ;
S3 = [1,1,2,2,5,9,9,6,4,3,2,6,9,2,7,1,1,0,8,5,0,9,1,6,0,4,0,2,6,6,8,1,8,1,8,7,1,3,9,7,6,6,5,1,0,4,1,5,0,9,9,1,3,4,8,4,9] ;
S4 = [1,2,8,2,6,9,5,5,3,0,3,6,4,5,4,1,4,9,2,2,4,2,9,2,9,0,2,7,0,4,1,7,8,2,5,2,8,1,8,6,2,5,8,6,6,0,2,7,5,8,3,4,4,4,4,3,1, ...
      2,1,6,0,0,3,0,8,6,3,3,3,5,8,0,6,5,9] ;
S = list(S1,S2,S3,S4) ;
// Recherche des racines
X = resolution(S) ;
// Ecriture du résultat
for i = 1:length(X), printf('\n Nombre n° %i = ',i), ECRIRE(X(i)), end
```

Les nombres a, b, c, d sont les zéros du polynôme :

$$P(x) = x^4 - 2305843008118644853x^3 - 252501130726497529916012424 9x^2 - 2400058627735384704271328662582917 27x + 45159552199405539657685147843152395434718 28$$

On trouve :

a = - 987654321
b = - 123456789
c = 16062012
d = 2305843009213693951

Cas général

Soit le système de n équations à n inconnues $x_1, x_2, \ldots, x_n$:

$$\begin{cases} x_1 + x_2 + \cdots + x_n = S_1 \\ {x_1}^2 + {x_2}^2 + \cdots + {x_n}^2 = S_2 \\ \vdots \qquad\qquad\qquad \vdots \\ {x_1}^n + {x_2}^n + \cdots + {x_n}^n = S_n \end{cases}$$

Les nombres $x_1, x_2, \ldots, x_n$ sont les zéros d'un polynôme $P(x)$ de degré n :

$$P(x) = \sum_{i=0}^{n} a_i x^i = x^n + \sum_{p=1}^{n} (-1)^p \sigma_p x^{n-p} \qquad \text{avec} \qquad \sigma_p = \sum_{1 \le i_k \le n} \prod_{k=1}^{p} x_{i_k}$$

Il n'est pas nécessaire de rechercher l'expression développée des σ_p en fonction des S_p.
Pour obtenir les coefficients de $P(x)$, il suffit d'appliquer le schéma récurrent suivant pour $p = 1, \ldots, n$:

$$a_{n-p} = (-1)^p \sigma_p = \frac{1}{p}\left[\sum_{k=1}^{p-1} (-1)^{k-1} \sigma_k S_{p-k} - S_p\right] = -\frac{1}{p}\left[\sum_{k=1}^{p-1} a_{n-k} S_{p-k} + S_p\right]$$

On déduit la fonction de calcul des x_p ($X = [x_1, x_2, \ldots, x_n]$) connaissant les S_p ($S = [S_1, S_2, \ldots, S_n]$).

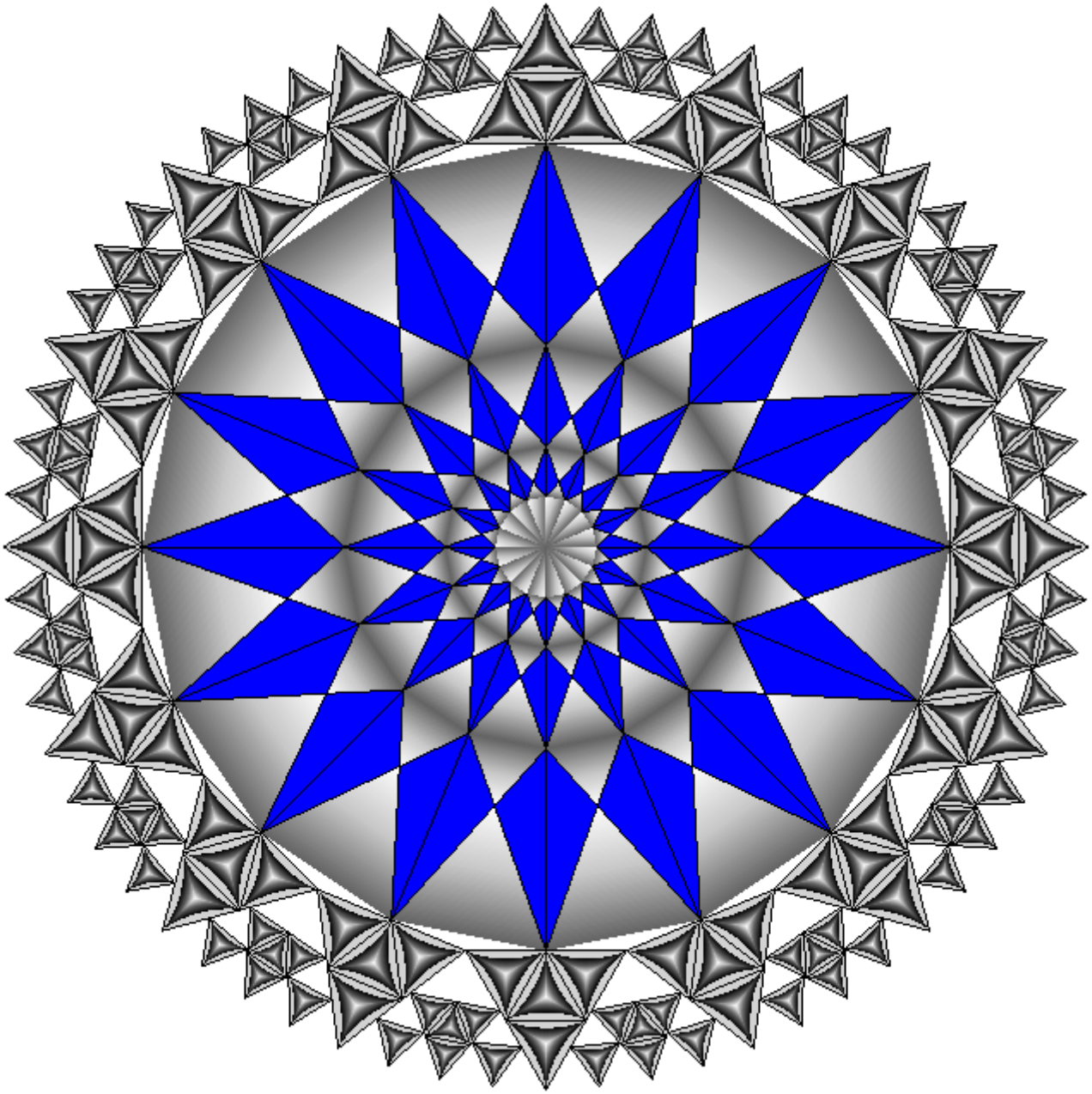
Dans $\mathbb{C}$:

```
function X = resolution(S)
    C = []
    n = length(S)
    for p = 1:n
        if p == 1 then, C = 1, end
        a = 0
        for k = 1:p-1
            a = a + C(p-k)*S(p-k)
        end
        C = [-(a + S(p))/p,C]
    end
    P = poly(C,'x','coeff')
    X = roots(P)
endfunction
```

Dans $\mathbb{Z}$:

```
function X = resolution(S)
    P = list()
    n = length(S)
    for p = 1:n
        if p == 1 then, P($+1) = ENTIER(1), end
        a = ENTIER(0)
        for k = 1:p-1
            a = ADDITION(a,MULTIPLICATION(P(k+1),S(p-k)))
        end
        [P($+1),r] = DIVISION(ADDITION(a,S(p)),ENTIER(-p))
    end
    X = racines(P)
endfunction
```

Mathématique & Informatique



Mathématique & Informatique

SYSTÈMES D'ÉQUATIONS LINÉAIRES

Considérons un système de n équations linéaires à n inconnues x_i :

$$\begin{cases} a_{11}x_1 + \ldots + a_{1n}x_n = b_1 \\ \vdots \qquad\qquad \vdots \qquad \vdots \\ a_{n1}x_1 + \ldots + a_{nn}x_n = b_n \end{cases}$$

On suppose que les coefficients a_{ij} et les seconds membres b_i sont des nombres rationnels.
Soit A la matrice carrée des a_{ij} , b le vecteur colonne des b_i , et x le vecteur colonne des x_i.
Le déterminant du système $Ax = b$ est :

$$\Delta = \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{vmatrix}$$

Si Δ est non nul, la solution $x = [x_1 ; \ldots ; x_n]$ est unique et les x_i sont des nombres rationnels.
En effet, à partir de la formule de **Cramer** $x_i = \Delta_i / \Delta$, où Δ_i est le déterminant obtenu en remplaçant la i^e colonne de Δ par la colonne des seconds membres, on déduit : $\Delta_i \in \mathbb{Q}$, $\Delta \in \mathbb{Q} \quad \Rightarrow \quad x_i \in \mathbb{Q}$
Ecrivons une fonction Scilab qui calcule la valeur exacte de x pour une matrice A et un second membre b donnés.
La méthode utilisée est la méthode de triangularisation de **Gauss**.

Structure des données

Dans un programme classique, les éléments a_{ij} d'une matrice réelle (ou complexe) A sont accessibles en écrivant simplement $A(i,j)$.

Une matrice carrée A d'ordre n, composée de nombres rationnels, est représentée ligne par ligne par une liste de n^2 éléments qui sont eux-mêmes des listes de 2 vecteurs constituant le numérateur et le dénominateur d'une fraction.
En fin de liste, on ajoute le nombre de lignes et le nombre de colonnes de la matrice A.
Si $a_{ij} \equiv$ list(N_{ij} , D_{ij}), alors : $A =$ list(list(N_{11} , D_{11}), ... , list(N_{1n} , D_{1n}), ... , list(N_{n1} , D_{n1}), ... , list(N_{nn} , D_{nn}), n, n)
Donc : $a_{ij} \equiv A((i-1)*n + j)$

Le second membre est un vecteur représenté par une liste de n éléments, complétée par les dimensions de la matrice colonne correspondante.
Ainsi : $b =$ list(list(N_1 , D_1), ... , list(N_n , D_n), n, 1)
D'où : $b_i \equiv b(i)$

La solution x possède la même structure que b : $x_i \equiv x(i)$

La méthode de Gauss

Dans un premier temps, la méthode de **Gauss** consiste à trianguliariser la matrice augmentée (A,b) en $n-1$ étapes.
Lors de l'étape k $(0 < k < n)$ on effectue les substitutions suivantes :

$$a_{ij} = a_{ij} - \frac{a_{ik}}{a_{kk}} a_{kj} \qquad (i, j = k+1, \ldots, n)$$

$$b_i = b_i - \frac{a_{ik}}{a_{kk}} b_k \qquad (i = k+1, \ldots, n)$$

Dans un deuxième temps, on résoud le système triangulaire supérieur obtenu :

$$x_i = \left(b_i - \sum_{j=i+1}^{n} a_{ij} x_j \right) / a_{ii} \qquad (i = n, \ldots, 1)$$

La programmation de l'algorithme de **Gauss** se fait en remplaçant les opérateurs +, –, *, / habituels par des appels aux fonctions vectorielles *F_ADDITION*, *F_SOUSTRACTION*, *F_MULTIPLICATION*, *F_DIVISION*.
La fonction Scilab *GAUSS* est incluse dans le module *Scilab-Equations*.
Pour résoudre le système linéaire $Ax = b$, on écrit : $x = GAUSS(A,b)$

Les entiers vectoriels

L'intérêt d'utiliser des entiers vectoriels est de ne pas perdre de précision en effectuant les calculs.
Avec une méthode numérique où les données arithmétiques sont représentées par un nombre limité de chiffres – voir la norme IEEE 754 : signe + exposant + mantisse = mot de 32 bits (simple précison) ou 64 bits (double précision) – les arrondis successifs conduisent parfois à des erreurs, comme le montre l'exemple simple suivant :

$$\begin{cases} x_1 + (1+\varepsilon) x_2 = 1 \\ (1-\varepsilon) x_1 + x_2 = 1 \end{cases}$$

Si l'on programme : $x = [1,1+\%eps\ ;\ 1-\%eps,1] \setminus [1;1]$, on trouve : $x = [0;1]$, soit : $x_1 = 0$ et $x_2 = 1$
Bien que la solution obtenue vérifie approximativement les équations (à ε infinitésimal près), elle est fausse car très éloignée de la bonne valeur qui est : $x_1 = -1/\varepsilon$ et $x_2 = +1/\varepsilon$
La fonction *GAUSS* développée ci-après fournit la solution exacte d'un système linéaire de n équations à n inconnues lorsque les coefficients sont des nombres rationnels. Elle permet notamment de résoudre des systèmes linéaires dont on sait que la solution est entière (cas des matrices unimodulaires par exemple).
D'autre part, l'algorithme de triangularisation peut être utilisé pour vérifier avec certitude si une matrice rationnelle A est singulière, autrement dit le test $\Delta = 0$ est réalisé sans erreur.
Précisons qu'il est inutile de rechercher à chaque étape du calcul le pivot de module maximum puisqu'il n'y a pas à minimiser l'influence des erreurs d'arrondi. Peu importe également si la matrice A est mal conditionnée, c'est-à-dire si une petite variation du second membre entraîne une grande variation de la solution ($\mathrm{cond}(A) = \|A\|\,\|A^{-1}\|$ est grand).
Par contre, le temps de calcul est plus long. A ce sujet, rappelons que le nombre total d'opérations élémentaires (addition, soustraction, multiplication, division) nécessaires pour résoudre un système linéaire de n équations à n inconnues par la méthode de **Gauss** est de l'ordre de $\frac{2}{3}n^3$ (coût $= \Sigma(n-k)[2(n-k)+3] + \Sigma[2(n-i)+2] = \frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{1}{6}n$).

Remarque

Dans le cas où les éléments a_{ij} de la matrice A sont donnés sous forme décimale, on peut facilement les convertir en nombres rationnels avec un dénominateur qui est une puissance de 10 :

$$a_p \ldots a_2 a_1 a_0 \bullet a_{-1} a_{-2} \ldots a_{-q} = a_p \ldots a_2 a_1 a_0 a_{-1} a_{-2} \ldots a_{-q} / 10^q$$

Lorsque la matrice A est composée de nombres réels, il est toujours possible d'approcher ses éléments d'aussi près que l'on veut par des nombres rationnels puisque $\mathbb{Q}$ est dense dans $\mathbb{R}$:

$$\forall\, a \in \mathbb{R},\ \forall\, q \in \mathbb{N},\ \exists!\, p \in \mathbb{Z} : \quad p/10^q \le a < (p+1)/10^q$$

$p/10^q$ est la valeur approchée de a à $1/10^q$ près par défaut
$(p+1)/10^q$ est la valeur approchée de a à $1/10^q$ près par excès

```
function x = GAUSS(A,b)
  if (b($) <> 1)|(b($-1) <> A($-1))|(A($-1) <> A($)) then
    x = list(0,0)
    return
  end
  n = A($)
  x = list()
  for i = 1:n                                          // Initialisation
    x($+1) = []
  end
  x($+1) = n
  x($+1) = 1
  for k = 1:n                                          // Triangularisation
    p = 0
    for i = k:n                                        // Recherche du pivot
      if ~F_NUL(A((i-1)*n+k)) then
        p = i
        break
      end
    end
    if p == 0 then
      printf('\n Matrice singuliere')
      x = list(0,0)
      return
    end
    if p <> k then
      for j = k:n                                      // Permutation de 2 lignes
        t = A((k-1)*n+j)
        A((k-1)*n+j) = A((p-1)*n+j)
        A((p-1)*n+j) = t
      end
      t = b(k)
      b(k) = b(p)
      b(p) = t
    end
    for i = k+1:n                                      // Combinaisons linéaires
      if ~F_NUL(A((i-1)*n+k)) then
        t = F_DIVISION(A((i-1)*n+k),A((k-1)*n+k))
        for j = k+1:n
          u = F_MULTIPLICATION(t,A((k-1)*n+j))
          A((i-1)*n+j) = F_SOUSTRACTION(A((i-1)*n+j),u)
        end
        v = F_MULTIPLICATION(t,b(k))
        b(i) = F_SOUSTRACTION(b(i),v)
      end
    end
  end
  for i = n:-1:1                                       // Résolution
    t = FRACTION(0,1)
    for j = i+1:n
      u = F_MULTIPLICATION(A((i-1)*n+j),x(j))
      t = F_ADDITION(t,u)
    end
    v = F_SOUSTRACTION(b(i),t)
    x(i) = F_DIVISION(v,A((i-1)*n+i))
  end
endfunction
```

Mathématique & Informatique

EXEMPLE 1

Résoudre dans $\mathbb{Q}$:

$$\begin{cases} \frac{417}{145}x_1 - \frac{11}{21}x_2 + 50x_3 + \frac{14}{531}x_4 = -\frac{153899}{29} \\ 13\,x_1 - \frac{62}{77}x_2 - \frac{913}{3}x_3 + 52\,x_4 = \frac{2009}{317} \\ 21\,x_1 - \frac{89}{33}x_2 + \frac{1}{13}x_3 + 43\,x_4 = -\frac{64446}{5} \\ \frac{151}{71}x_1 + 10\,x_2 + 2\,x_4 = 91270 \end{cases}$$

Programme :

```
exec("Scilab-Equations.sce",-1)
A = list(FRACTION(417,145), FRACTION(-11,21), FRACTION(50,1), FRACTION(14,531), ...
         FRACTION(13,1), FRACTION(-62,77), FRACTION(-913,3), FRACTION(52,1), ...
         FRACTION(21,1), FRACTION(-89,33), FRACTION(1,13), FRACTION(43,1), ...
         FRACTION(151,71), FRACTION(10,1), FRACTION(0,1), FRACTION(2,1), ...
         4,4) ;
b = list(FRACTION(-153899,29), FRACTION(2009,317), FRACTION(-64446,5), FRACTION(91270,1),4,1) ;
//-------------------------------------------------------------------------------------------------------------------------------
x = GAUSS(A,b) ;
for i = 1:4
  printf('\n x%i = ',i), ECRIRE(x(i)), printf(' = '), ECRIRE(DECIMALE(x(i),30))
end
```

Résultat :

$$x_1 = -\frac{3436239084\ 3147509309\ 56}{1739484612\ 614575585} \approx -1975.43517\ 1656865858\ 3255154693\ 21708$$

$$x_2 = \frac{7351328129\ 8449370101\ 57}{7906748239\ 15716175} \approx 9297.53661\ 9970296359\ 8669836534\ 38250$$

$$x_3 = \frac{9065241351\ 9656487552\ 3}{8697423063\ 072877925} \approx 104.229049\ 0668947327\ 8325239100\ 5522$$

$$x_4 = \frac{2170799753\ 8051411926\ 66}{1739484612\ 614575585} \approx 1247.95570\ 9445607951\ 4196791684\ 95914$$

EXEMPLE 2

Soit le système linéaire : $Hx = u$

où H est la matrice de **Hilbert** d'ordre n : $H(i,j) = \dfrac{1}{i+j-1}$ $\quad (i, j = 1,\ldots,n)$

et u est le vecteur colonne unité d'ordre n : $u(i) = 1$ $\quad (i = 1,\ldots,n)$

Quelle est la solution $x = (x_1, x_2, \cdots, x_n)$ de ce système linéaire dans le cas $n = 10$?

$$\begin{pmatrix} 1/1 & 1/2 & 1/3 & 1/4 & 1/5 & 1/6 & 1/7 & 1/8 & 1/9 & 1/10 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 & 1/7 & 1/8 & 1/9 & 1/10 & 1/11 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 & 1/8 & 1/9 & 1/10 & 1/11 & 1/12 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 & 1/9 & 1/10 & 1/11 & 1/12 & 1/13 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 & 1/10 & 1/11 & 1/12 & 1/13 & 1/14 \\ 1/6 & 1/7 & 1/8 & 1/9 & 1/10 & 1/11 & 1/12 & 1/13 & 1/14 & 1/15 \\ 1/7 & 1/8 & 1/9 & 1/10 & 1/11 & 1/12 & 1/13 & 1/14 & 1/15 & 1/16 \\ 1/8 & 1/9 & 1/10 & 1/11 & 1/12 & 1/13 & 1/14 & 1/15 & 1/16 & 1/17 \\ 1/9 & 1/10 & 1/11 & 1/12 & 1/13 & 1/14 & 1/15 & 1/16 & 1/17 & 1/18 \\ 1/10 & 1/11 & 1/12 & 1/13 & 1/14 & 1/15 & 1/16 & 1/17 & 1/18 & 1/19 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

En utilisant l'opérateur matriciel "\", le programme est le suivant :

```
n = 10 ;
H = zeros(n,n) ;
for i = 1:n
    for j = 1:n
        H(i,j) = 1/(i+j-1) ;
    end
end
u = ones(n,1) ;
x = H\u ;
for i = 1:n, printf('\n x%i = %.10f',i,x(i)), end
```

On obtient :

```
x1 = -9.9978041242
x2 = 989.8115012107
x3 = -23756.0050086463
x4 = 240203.8250881990
x5 = -1261088.0113129939
x6 = 3783308.4861997212
x7 = -6725948.1777543351
x8 = 7000535.6127205668
x9 = -3937829.8849767279
x10 = 923694.3398908603
```

En utilisant la fonction "*fsolve*", le programme est le suivant :

```
deff('y = f(x)','y = H*x - u')
deff('J = jac(x)','J = H')
x0 = zeros(n,1) ;
x = fsolve(x0, f, jac) ;
for i = 1:n, printf('\n x%i = %.10f',i,x(i)), end
```

On obtient :

```
x1 = -9.9982512341
x2 = 989.8496023008
x3 = -23756.8091289744
x4 = 240211.0882906369
x5 = -1261122.4913234212
x6 = 3783402.9298191238
x7 = -6726102.6936803423
x8 = 7000684.5972100049
x9 = -3937907.9564813077
x10 = 923711.4827815801
```

En utilisant la fonction "*GAUSS*", le programme est le suivant :

```
n = 10 ;
H = M_ZERO(n,n) ;
for i = 1:n
    for j = 1:n
        H((i-1)*n+j) = FRACTION(1,i+j-1) ;
    end
end
u = M_UN(n,1) ;
x = GAUSS(H,u) ;
for i = 1:n, printf('\n x%i = ',i), ECRIRE(x(i)), end
```

On obtient :

```
x1 = - 10
x2 = 990
x3 = - 23760
x4 = 240240
x5 = - 1261260
x6 = 3783780
x7 = - 6726720
x8 = 7001280
x9 = - 3938220
x10 = 923780
```

Conclusion

La matrice de **Hilbert** est un exemple de matrice mal conditionnée (*H* est un cas particulier de matrice symétrique presque singulière ; pour $n = 10$: det $H = 1/46206893947914691316295628839036278726983680000000000$)
L'opérateur Scilab "\" et la fonction "*fsolve*" fournissent une approximation de la solution x du système linéaire $Hx = u$, la fonction "*GAUSS*" donne la valeur exacte de x.

EXEMPLE 3

Soit la réaction d'oxydoréduction :

$a[Cr(N_2H_4CO)_6]_4[Cr(CN)_6]_3 + bKMnO_4 + cH_2SO_4 \rightleftarrows dK_2Cr_2O_7 + eMnSO_4 + fCO_2 + gKNO_3 + hK_2SO_4 + iH_2O$

Déterminer les coefficients stœchiométriques $a, b, c, d, e, f, g, h, i$ permettant d'équilibrer cette équation chimique.

Lors des réactions chimiques ordinaires, tout dégagement ou absorption de chaleur s'accompagne d'une très faible variation de masse ($\Delta E = \Delta m\, c^2$, loi d'équivalence masse – énergie d'**Einstein**). En conséquence, avec une très bonne approximation, on considère qu'il y a conservation des masses et des éléments (<< Rien ne se perd, rien ne se crée, tout se transforme >>, loi de **Lavoisier**).
D'où les relations :

- Cr (Chrome) : $7a = 2d$
- N (Azote) : $66a = g$
- H (Hydrogène) : $96a + 2c = 2i$
- C (Carbone) : $42a = f$
- O (Oxygène) : $24a + 4b + 4c = 7d + 4e + 2f + 3g + 4h + i$
- K (Potassium) : $b = 2d + g + 2h$
- Mn (Manganèse) : $b = e$
- S (Soufre) : $c = e + h$

Il s'agit d'un système linéaire de 8 équations à 9 inconnues.
En prenant $a = 1$, on est ramené à résoudre un système de **Cramer** d'ordre 8.
Posons $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = (b, c, d, e, f, g, h, i)$, on a :

$$\begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -4 & -4 & 7 & 4 & 2 & 3 & 4 & 1 \\ -1 & 0 & 2 & 0 & 0 & 1 & 2 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} = \begin{pmatrix} 7 \\ 66 \\ 96 \\ 42 \\ 24 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Ecrivons :

```
exec("Scilab-Equations.sce",-1)
A = list(FRACTION(0,1), FRACTION(0,1), FRACTION(2,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), ...
         FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), FRACTION(1,1), FRACTION(0,1), FRACTION(0,1), ...
         FRACTION(0,1), FRACTION(-2,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), FRACTION(2,1), ...
         FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), FRACTION(1,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), ...
         FRACTION(-4,1), FRACTION(-4,1), FRACTION(7,1), FRACTION(4,1), FRACTION(2,1), FRACTION(3,1), FRACTION(4,1), FRACTION(1,1), ...
         FRACTION(-1,1), FRACTION(0,1), FRACTION(2,1), FRACTION(0,1), FRACTION(0,1), FRACTION(1,1), FRACTION(2,1), FRACTION(0,1), ...
         FRACTION(-1,1), FRACTION(0,1), FRACTION(0,1), FRACTION(1,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1), ...
         FRACTION(0,1), FRACTION(-1,1), FRACTION(0,1), FRACTION(1,1), FRACTION(0,1), FRACTION(0,1), FRACTION(1,1), FRACTION(0,1), ...
         8,8) ;
b = list(FRACTION(7,1), FRACTION(66,1), FRACTION(96,1), FRACTION(42,1), FRACTION(24,1), FRACTION(0,1), FRACTION(0,1), FRACTION(0,1),8,1) ;
x = GAUSS(A,b) ;
for i = 1:8
   printf('\n x%i = ',i), ECRIRE(x(i))
end
```

L'exécution de ce programme donne :

x1 = 588/5
x2 = 1399/10
x3 = 7/2
x4 = 588/5
x5 = 42
x6 = 66
x7 = 223/10
x8 = 1879/10

Multiplions tous les coefficients stœchiométriques par *10* pour obtenir des entiers, il vient :

a = 10
b = 1176
c = 1399
d = 35
e = 1176
f = 420
g = 660
h = 223
i = 1879

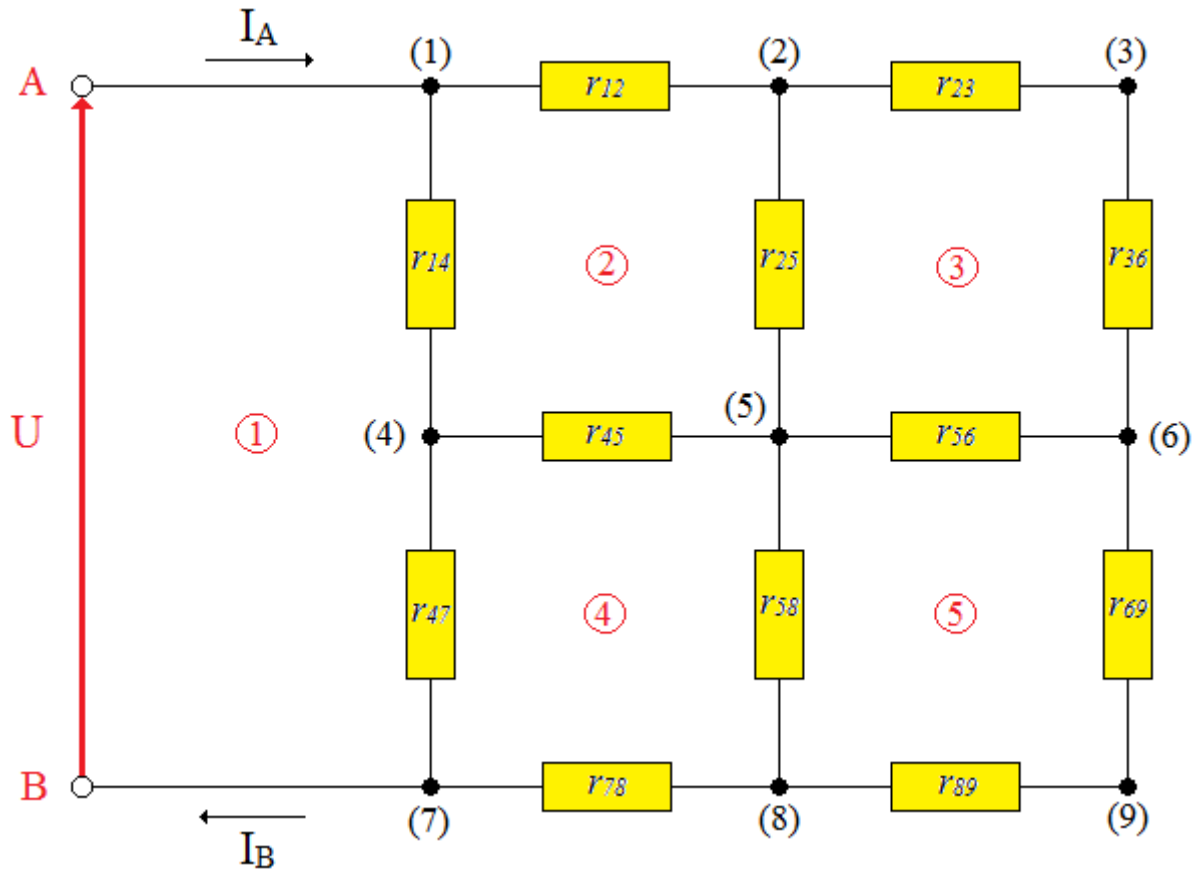
Finalement :

$$10[Cr(N_2H_4CO)_6]_4[Cr(CN)_6]_3 + 1176KMnO_4 + 1399H_2SO_4 \rightleftarrows 35K_2Cr_2O_7 + 1176MnSO_4 + 420CO_2 + 660KNO_3 + 223K_2SO_4 + 1879H_2O$$

Mathématique & Informatique

EXEMPLE 4

Calculer la résistance R vue des bornes A et B du réseau résistif représenté ci-dessous :



On suppose que la résistance r_{ij} entre le nœud i et le nœud j est un multiple rationnel d'une résistance étalon r :

$$r_{ij} = k_{ij}\, r \qquad (k_{ij} \in \mathbb{Q})$$

La résistance R se met sous la forme :

$$R = \frac{p}{q}\, r \qquad (p \in \mathbb{N}, q \in \mathbb{N})$$

Application :

1) $k_{ij} = 1 \qquad \forall i, j$

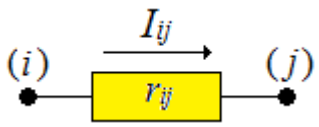
2) $k_{ij} = \left(\frac{j}{i}\right)^2$

Remarque

Pour simplifier, il est possible de regrouper les résistances r_{23} et r_{36} , de même les résistances r_{69} et r_{89} , ce qui conduit à supprimer les pseudo-nœuds (3) et (9) sur le schéma. On les conserve si l'on souhaite homogénéiser les notations.

Mathématique & Informatique

Notons I_{ij} le courant allant du nœud i vers le nœud j ($j > i$) en traversant la résistance r_{ij}.



Exprimons les lois de **Kirchhoff**.
Equations des nœuds (Σ courants = 0) :

(1) $I_A - I_{12} - I_{14} = 0$
(2) $I_{12} - I_{23} - I_{25} = 0$
(3) $I_{23} - I_{36} = 0$
(4) $I_{14} - I_{45} - I_{47} = 0$
(5) $I_{25} + I_{45} - I_{56} - I_{58} = 0$
(6) $I_{36} + I_{56} - I_{69} = 0$
(7) $I_{47} - I_{78} - I_B = 0$
(8) $I_{58} + I_{78} - I_{89} = 0$
(9) $I_{69} + I_{89} = 0$

Equations des mailles (Σ tensions = 0) :

① $r_{14}I_{14} + r_{47}I_{47} - U = 0$
② $r_{12}I_{12} + r_{25}I_{25} - r_{45}I_{45} - r_{14}I_{14} = 0$
③ $r_{23}I_{23} + r_{36}I_{36} - r_{56}I_{56} - r_{25}I_{25} = 0$
④ $r_{45}I_{45} + r_{58}I_{58} - r_{78}I_{78} - r_{47}I_{47} = 0$
⑤ $r_{56}I_{56} + r_{69}I_{69} - r_{89}I_{89} - r_{58}I_{58} = 0$

On obtient un système linéaire de 14 équations à 14 inconnues :

$$\begin{pmatrix}
0 & 0 & 0 & r_{14} & 0 & 0 & 0 & 0 & r_{47} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & r_{12} & -r_{14} & 0 & r_{25} & 0 & -r_{45} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & r_{23} & -r_{25} & r_{36} & 0 & 0 & -r_{56} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{45} & -r_{47} & 0 & r_{58} & 0 & -r_{78} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{56} & -r_{58} & r_{69} & 0 & -r_{89} \\
1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
I_A \\ I_B \\ I_{12} \\ I_{14} \\ I_{23} \\ I_{25} \\ I_{36} \\ I_{45} \\ I_{47} \\ I_{56} \\ I_{58} \\ I_{69} \\ I_{78} \\ I_{89}
\end{pmatrix}
=
\begin{pmatrix}
U \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{pmatrix}$$

Précisons que les solutions I_{ij} sont des grandeurs algébriques (positives ou négatives). Ainsi :

- $I_{ij} > 0$ signifie que le courant circule effectivement du nœud i vers le nœud j.
- $I_{ij} < 0$ signifie que le courant circule en réalité du nœud j vers le nœud i.

Mathématique & Informatique

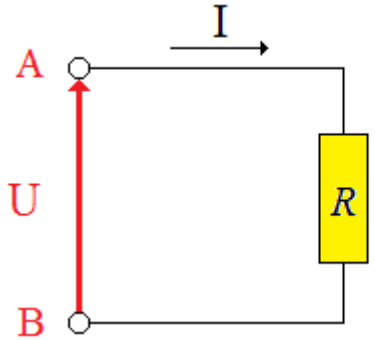
Considérons le système linéaire $Ax = b$ suivant :

$$\begin{pmatrix}
0 & 0 & 0 & k_{14} & 0 & 0 & 0 & 0 & k_{47} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & k_{12} & -k_{14} & 0 & k_{25} & 0 & -k_{45} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & k_{23} & -k_{25} & k_{36} & 0 & 0 & -k_{56} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{45} & -k_{47} & 0 & k_{58} & 0 & -k_{78} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{56} & -k_{58} & k_{69} & 0 & -k_{89} \\
1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1
\end{pmatrix}
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \end{pmatrix}
=
\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

La solution $x = (x_1 , x_2 , x_3 , x_4 , x_5 , x_6 , x_7 , x_8 , x_9 , x_{10} , x_{11} , x_{12} , x_{13} , x_{14})$ permet de déduire :

$$\begin{pmatrix} I_A \\ I_B \\ I_{12} \\ I_{14} \\ I_{23} \\ I_{25} \\ I_{36} \\ I_{45} \\ I_{47} \\ I_{56} \\ I_{58} \\ I_{69} \\ I_{78} \\ I_{89} \end{pmatrix}
= \frac{U}{r}
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \end{pmatrix}$$

Nous recherchons la résistance équivalente R telle que :



$$U = R\,I$$

On a : $x_1 = x_2 = \dfrac{q}{p} \quad (p \in \mathbb{N}, q \in \mathbb{N})$

D'où : $I = I_A = I_B = \dfrac{q}{p} \times \dfrac{U}{r}$

Et : $R = \dfrac{p}{q} r$

Pour $k_{ij} = 1$, le programme de calcul numérique du vecteur *x* s'écrit :

```
exec("Scilab-Equations.sce",-1)
A = list(0,0,0,[1,4],0,0,0,0,[4,7],0,0,0,0,0, ...
        0,0,[1,2],-[1,4],0,[2,5],0,-[4,5],0,0,0,0,0,0, ...
        0,0,0,0,[2,3],-[2,5],[3,6],0,0,-[5,6],0,0,0,0, ...
        0,0,0,0,0,0,0,[4,5],-[4,7],0,[5,8],0,-[7,8],0, ...
        0,0,0,0,0,0,0,0,0,[5,6],-[5,8],[6,9],0,-[8,9], ...
        1,0,-1,-1,0,0,0,0,0,0,0,0,0,0, ...
        0,0,1,0,-1,-1,0,0,0,0,0,0,0,0, ...
        0,0,0,0,1,0,-1,0,0,0,0,0,0,0, ...
        0,0,0,1,0,0,0,-1,-1,0,0,0,0,0, ...
        0,0,0,0,0,1,0,1,0,-1,-1,0,0,0, ...
        0,0,0,0,0,0,1,0,0,1,0,-1,0,0, ...
        0,-1,0,0,0,0,0,0,1,0,0,0,-1,0, ...
        0,0,0,0,0,0,0,0,0,0,1,0,1,-1, ...
        0,0,0,0,0,0,0,0,0,0,0,1,0,1, ...
        14,14) ;
b = list(1,0,0,0,0,0,0,0,0,0,0,0,0,0, ...
        14,1) ;
for n = 1:14^2
    if A(n) == 0 then, A(n) = FRACTION(0,1) ; continue, end
    if A(n) == 1 then, A(n) = FRACTION(1,1) ; continue, end
    if A(n) == -1 then, A(n) = FRACTION(-1,1) ; continue, end
    I = A(n) ;
    i = abs(I(1)) ;
    j = abs(I(2)) ;
    kij = FRACTION(1,1) ;
    s = FRACTION(I(1),abs(I(1))) ;
    A(n) = F_MULTIPLICATION(s,kij) ;
end
for n = 1:14
    b(n) = FRACTION(b(n),1) ;
end
//-------------------------------------------------------------------------
x = GAUSS(A,b) ;
for i = 1:14
  printf('\n x%i = ',i), ECRIRE(x(i))
end
```

Résultat : $x = \left(\dfrac{4}{5}, \dfrac{4}{5}, \dfrac{3}{10}, \dfrac{1}{2}, \dfrac{1}{10}, \dfrac{1}{5}, \dfrac{1}{10}, 0, \dfrac{1}{2}, 0, \dfrac{1}{5}, \dfrac{1}{10}, -\dfrac{3}{10}, -\dfrac{1}{10}\right)$

Par conséquent : $R = \dfrac{5}{4} r$

Noter que l'on peut retrouver cette valeur en remarquant que les nœuds (4), (5), (6) sont au même potentiel par raison de symétrie. Par suite, $I_{45} = I_{56} = 0$. Donc, dans ce cas particulier, on est autorisé à enlever les résistances r_{45} et r_{56} . Il s'ensuit que le calcul de *R* s'effectue simplement en appliquant les formules habituelles d'association série et parallèle des résistances.

Pour $k_{ij} = \left(\frac{j}{i}\right)^2$, le programme de calcul numérique du vecteur x s'écrit :

```
exec("Scilab-Equations.sce",-1)
A = list(0,0,0,[1,4],0,0,0,0,[4,7],0,0,0,0,0, ...
         0,0,[1,2],-[1,4],0,[2,5],0,-[4,5],0,0,0,0,0,0, ...
         0,0,0,0,[2,3],-[2,5],[3,6],0,0,-[5,6],0,0,0,0, ...
         0,0,0,0,0,0,0,[4,5],-[4,7],0,[5,8],0,-[7,8],0, ...
         0,0,0,0,0,0,0,0,0,[5,6],-[5,8],[6,9],0,-[8,9], ...
         1,0,-1,-1,0,0,0,0,0,0,0,0,0,0, ...
         0,0,1,0,-1,-1,0,0,0,0,0,0,0,0, ...
         0,0,0,0,1,0,-1,0,0,0,0,0,0,0, ...
         0,0,0,1,0,0,0,-1,-1,0,0,0,0,0, ...
         0,0,0,0,0,1,0,1,0,-1,-1,0,0,0, ...
         0,0,0,0,0,0,1,0,0,1,0,-1,0,0, ...
         0,-1,0,0,0,0,0,0,1,0,0,0,-1,0, ...
         0,0,0,0,0,0,0,0,0,0,1,0,1,-1, ...
         0,0,0,0,0,0,0,0,0,0,0,1,0,1, ...
         14,14) ;
b = list(1,0,0,0,0,0,0,0,0,0,0,0,0,0, ...
         14,1) ;
for n = 1:14^2
    if A(n) == 0 then, A(n) = FRACTION(0,1) ; continue, end
    if A(n) == 1 then, A(n) = FRACTION(1,1) ; continue, end
    if A(n) == -1 then, A(n) = FRACTION(-1,1) ; continue, end
    I = A(n) ;
    i = abs(I(1)) ;
    j = abs(I(2)) ;
    kij = FRACTION(j^2,i^2) ;
    s = FRACTION(I(1),abs(I(1))) ;
    A(n) = F_MULTIPLICATION(s,kij) ;
end
for n = 1:14
    b(n) = FRACTION(b(n),1) ;
end
//-------------------------------------------------------------------------------
x = GAUSS(A,b) ;
for i = 1:14
  printf('\n x%i = ',i), ECRIRE(x(i))
end
```

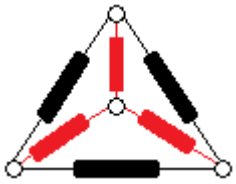
On trouve :

$$x_1 = \frac{30750729600200}{197848960201749}$$

Donc :

$$R = \frac{197848960201749}{30750729600200} r \approx 6.4339598694\, r$$

Une autre méthode pour calculer R consiste à appliquer 3 fois la transformation triangle-étoile de **Kennelly**.



Systèmes tridiagonaux

Soit la matrice carrée A d'ordre n, et le vecteur colonne b d'ordre n :

$$A = \begin{pmatrix} v_1 & w_1 & & & & \\ u_2 & v_2 & w_2 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & u_{n-1} & v_{n-1} & w_{n-1} \\ & & & & u_n & v_n \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$$

Compte tenu de la structure particulière de la matrice A, un algorithme spécifique, dit algorithme de Thomas, donne rapidement la solution du système linéaire Ax = b.

1re phase

- $y_i = w_i / (v_i - u_i\, y_{i-1})$
- $z_i = (b_i - u_i\, z_{i-1}) / (v_i - u_i\, y_{i-1})$

  $i = 1, \ldots, n$

  Par convention : $u_1 = w_n = y_0 = z_0 = 0$

2e phase

- $x_i = z_i - x_{i+1}\, y_i$ $\qquad i = n, \ldots, 1$

  Par convention : $x_{n+1} = 0$

Coût en opérations élémentaires

L'algorithme de Thomas est de l'ordre de 8n (coût = 6n + 2n = 8n)

Application

Les matrices tridiagonales interviennent fréquemment en Physique dans les problèmes de discrétisation d'équations différentielles. Par exemple, le problème de Dirichlet :
Etant données 2 fonctions f et g continues sur un intervalle [a,b], on recherche une fonction u telle que :

$$\begin{cases} -u''(x) + f(x)\,u(x) = g(x) & \forall x \in]a,b[\\ u(a) = \alpha \ ; \ u(b) = \beta & \text{(conditions aux limites)} \end{cases}$$

Si la solution u existe, celle-ci est unique mais, sauf cas particuliers, on ne connait pas son expression analytique. On calcule une valeur approchée u_i de u en des points $x_i = a + ih$ en effectuant un découpage de l'intervalle [a,b] avec un pas h. La formule de Taylor à l'ordre 2 conduit à l'approximation suivante de la dérivée seconde de u :

$$u_i'' \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}$$

L'équation différentielle discrétisée est équivalente à un système linéaire Tu = v où T est une matrice tridiagonale. Le problème se généralise au plan en introduisant le laplacien Δu de u. La matrice T est alors tridiagonale par blocs.

Programme

```
function x = THOMAS(A,b)
  if (b($) <> 1)|(b($-1) <> A($-1))|(A($-1) <> A($)) then
    x = list(0,0) ; return
  end
  n = A($)
  x = list() ; y = list() ; z = list()
  for i = 1:n
   x($+1) = [] ; y($+1) = [] ; z($+1) = []
  end
  x($+1) = n ; y($+1) = n ; z($+1) = n
  x($+1) = 1 ; y($+1) = 1 ; z($+1) = 1
  if n == 0 then, x = list(0,0) ; return, end
  if n == 1 then, x(1) = F_DIVISION(b(1),A(1)) ; return, end
  for i = 1:n
    select i
      case 1
        u = FRACTION(0,1)
        v = A(1)
        w = A(2)
      case n
        u = A(n^2-1)
        v = A(n^2)
        w = FRACTION(0,1)
      else
        u = A((i-1)*n+i-1)
        v = A((i-1)*n+i)
        w = A((i-1)*n+i+1)
    end
    if i == 1 then
      y(1) = F_DIVISION(w,v)
      z(1) = F_DIVISION(b(1),v)
    else
      t1 = F_SOUSTRACTION(v,F_MULTIPLICATION(y(i-1),u))
      t2 = F_SOUSTRACTION(b(i),F_MULTIPLICATION(z(i-1),u))
      y(i) = F_DIVISION(w,t1)
      z(i) = F_DIVISION(t2,t1)
    end
  end
  x(n) = z(n)
  for i = n-1:-1:1
    x(i) = F_SOUSTRACTION(z(i),F_MULTIPLICATION(x(i+1),y(i)))
  end
endfunction
```

EXEMPLE

Soit la matrice carrée tridiagonale A de dimension n, et le vecteur colonne b de dimension n :

$$A = \begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{pmatrix} \qquad b = \begin{pmatrix} 1^1 \\ 2^2 \\ \vdots \\ \vdots \\ \vdots \\ n^n \end{pmatrix}$$

Résoudre en nombres entiers le système linéaire $Ax = b$ pour $n = 20$

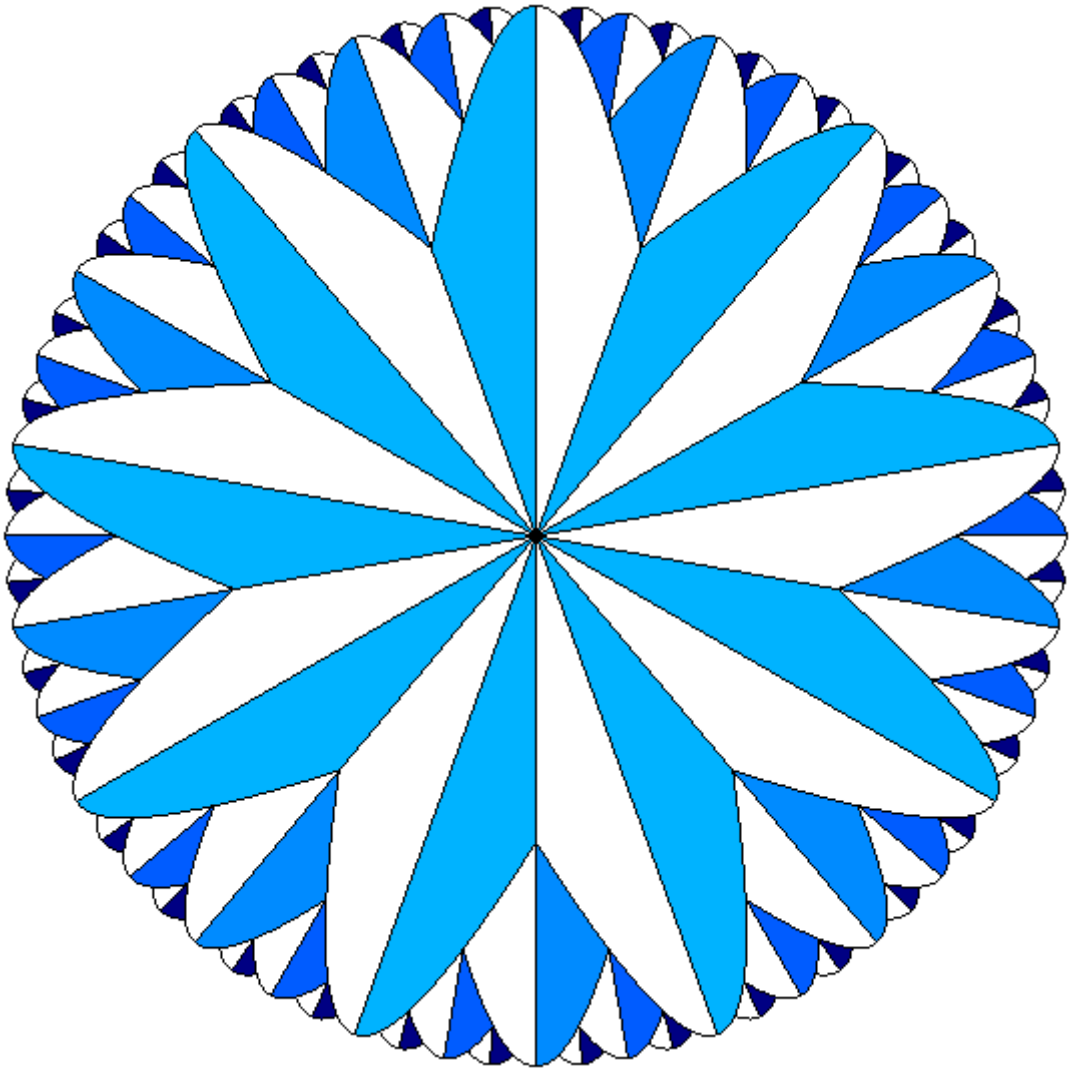
On écrit le programme Scilab :

```
exec("Scilab-Equations.sce",-1)
n = 20 ;
A = M_ZERO(n,n) ; b = M_ZERO(n,1) ;
for i = 1:n
   A((i-1)*n + i) = FRACTION(2,1) ;
   if i == n then, A(n*n) = FRACTION(1,1) ; end
   if i > 1 then, A((i-1)*n + i - 1) = FRACTION(-1,1) ; end
   if i < n then, A((i-1)*n + i + 1) = FRACTION(-1,1) ; end
   b(i) = F_PUISSANCE(FRACTION(i,1),i) ;
end
//---------------------------------------------------------------------
x = THOMAS(A,b) ;
M_ECRIRE(x)
```

La solution est la suivante :

```
x1  =  1068762122000595543032150 24
x2  =  2137524244001191086064300 47
x3  =  3206286366001786629096450 66
x4  =  4275048488002382172128600 58
x5  =  5343810610002977715160747 94
x6  =  6412572732003573258192864 05
x7  =  7481334854004168801224513 60
x8  =  8550096976004764344247927 72
x9  =  9618859098005359887103569 68
x10 = 10687621220005955426085006 75
x11 = 11756383342006550865066443 82
x12 = 12825145464007143450931174 78
x13 = 13893907586007646875791423 18
x14 = 14962669708005121549585749 05
x15 = 16031431829891476155124494 76
x16 = 17100193947398891856854646 72
x17 = 18168955880438866821489282 52
x18 = 19237709541076222922756276 55
x19 = 20306069737632826058647516 34
x20 = 21354645737632826058647516 34
```

Mathématique & Informatique



CALCUL MATRICIEL

Intéressons nous aux matrices dont les éléments sont des nombres rationnels.
Soit A une matrice rectangulaire de dimension (m,n), c'est-à-dire de m lignes et de n colonnes.

$$A = \begin{pmatrix} a11 & \dots & a1n \\ \vdots & aij & \vdots \\ am1 & \dots & amn \end{pmatrix}$$

La matrice A peut être représentée par une liste constituée de mn + 2 éléments :

$$A = \text{list}(a_{11} , \dots , a_{1n} , \dots , a_{ij} , \dots , a_{m1} , \dots , a_{mn} , m , n)$$

où :

- a_{ij} = list(N_{ij},D_{ij}) est un nombre rationnel, N_{ij} est le vecteur équivalent au numérateur de la fraction, D_{ij} est le vecteur équivalent au dénominateur de la fraction.
- m et n sont 2 entiers naturels indiquant respectivement le nombre de lignes et le nombre de colonnes de A.

L'élément a_{ij} de la matrice, situé à l'intersection de la ligne i, $1 \le i \le m$, et de la colonne j, $1 \le j \le n$, correspond à l'élément d'indice k = (i-1)n + j de la liste. Inversement, le k^e élément de la liste, $1 \le k \le mn$, correspond à l'élément de la matrice situé à l'intersection de la ligne i = E((k-1)/n) + 1 et de la colonne j = (k-1) mod n + 1.
L'avant-dernière et la dernière position de la liste A sont indexées $-1 et $ par Scilab : A($-1) = m et A($) = n
Par convention, A = list(0,0) désignera une matrice non définie.

La fonction *M_ECRIRE* affiche à l'écran les éléments a_{ij} de la liste A, une valeur par ligne, sous la forme : (i,j) = a_{ij}

```
function M_ECRIRE(A)
  m = A($-1) ; n = A($)
  printf('\n Matrice %i ligne(s) et %i colonne(s) :',m,n)
  printf('\n ====================================')
  for i = 1:m*n
    lig = floor((i-1)/n) + 1 ; col = modulo(i-1,n) + 1
    printf('\n (%i,%i) = ',lig,col), ECRIRE(A(i))
  end
endfunction
```

Exemple

$$A = \begin{pmatrix} -1 & 2/5 & 1 \\ 0 & 376 & -12/43 \end{pmatrix}$$

```
A = list(list([-1,1],[1,1]),list([1,2],[1,5]),list([1,1],[1,1]),list([0,0],[1,1]),list([1,3,7,6],[1,1]),list([-1,1,2],[1,4,3]),2,3) ;
```

Résultat à l'écran de l'instruction *M_ECRIRE(A)* :

```
 Matrice 2 ligne(s) et 3 colonne(s) :
 ===========================
(1,1) = - 1
(1,2) = 2/5
(1,3) = 1
(2,1) = 0
(2,2) = 376
(2,3) = - 12/43
```

Matrice vide

V = matrice (ou vecteur) dont tous les éléments sont égaux à ∅

```
function V = M_VIDE(m,n)
  V = list()
  for i = 1:m*n
    V($+1) = []
  end
  V($+1) = m ; V($+1) = n
endfunction
```

Matrice zéro

Z = matrice (ou vecteur) dont tous les éléments sont égaux à 0

```
function Z = M_ZERO(m,n)
  Z = list()
  for i = 1:m*n
    Z($+1) = FRACTION(0,1)
  end
  Z($+1) = m ; Z($+1) = n
endfunction
```

Matrice un

U = matrice (ou vecteur) dont tous les éléments sont égaux à 1

```
function U = M_UN(m,n)
  U = list()
  for i = 1:m*n
    U($+1) = FRACTION(1,1)
  end
  U($+1) = m ; U($+1) = n
endfunction
```

Matrice identité

I = matrice carrée dont les éléments diagonaux valent 1 et les tous les autres éléments sont nuls

```
function I = M_IDENTITE(n)
  I = list()
  for i = 1:n
    for j = 1:n
      if i == j then
        I($+1) = FRACTION(1,1)
      else
        I($+1) = FRACTION(0,1)
      end
    end
  end
  I($+1) = n ; I($+1) = n
endfunction
```

Opposée d'une matrice

$B = -A \quad \Rightarrow \quad b_{ij} = -a_{ij}$

```
function B = M_MOINS(A)
  m = A($-1) ; n = A($)
  B = list()
  for i = 1:m*n
    B($+1) = F_MOINS(A(i))
  end
  B($+1) = m ; B($+1) = n
endfunction
```

Valeur absolue d'une matrice

$B = |A| \quad \Rightarrow \quad b_{ij} = |a_{ij}|$

```
function B = M_ABSOLU(A)
  m = A($-1) ; n = A($)
  B = list()
  for i = 1:m*n
    B($+1) = F_ABSOLU(A(i))
  end
  B($+1) = m ; B($+1) = n
endfunction
```

Produit d'une matrice par un scalaire

$B = k\,A \quad \Rightarrow \quad b_{ij} = k\,a_{ij}$

```
function B = M_SCALAIRE(A,k)
  m = A($-1) ; n = A($)
  B = list()
  for i = 1:m*n
    B($+1) = F_MULTIPLICATION(A(i),k)
  end
  B($+1) = m ; B($+1) = n
endfunction
```

Transposée d'une matrice

$B = {}^tA \quad \Rightarrow \quad b_{ij} = a_{ji}$

```
function B = M_TRANSPOSEE(A)
  m = A($-1) ; n = A($)
  B = list()
  for j = 1:n
    for i = 1:m
      B($+1) = A((i-1)*n+j)
    end
  end
  B($+1) = n ; B($+1) = m
endfunction
```

Somme de deux matrices

C = A + B ⇒ $c_{ij} = a_{ij} + b_{ij}$

```
function C = M_ADDITION(A,B)
  ma = A($-1) ; na = A($)
  mb = B($-1) ; nb = B($)
  if (ma == mb)&(na == nb) then
    m = ma ; n = na
  else
    C = list(0,0)
    return
  end
  C = list()
  for i = 1:m*n
    C($+1) = F_ADDITION(A(i),B(i))
  end
  C($+1) = m ; C($+1) = n
endfunction
```

Différence de deux matrices

C = A - B ⇒ $c_{ij} = a_{ij} - b_{ij}$

```
function C = M_SOUSTRACTION(A,B)
  ma = A($-1) ; na = A($)
  mb = B($-1) ; nb = B($)
  if (ma == mb)&(na == nb) then
    m = ma ; n = na
  else
    C = list(0,0)
    return
  end
  C = list()
  for i = 1:m*n
    C($+1) = F_SOUSTRACTION(A(i),B(i))
  end
  C($+1) = m ; C($+1) = n
endfunction
```

Mathématique & Informatique

Produit de deux matrices

$C = A * B \quad \Rightarrow \quad c_{ij} = \sum a_{ik} * b_{kj}$

```
function C = M_MULTIPLICATION(A,B)
  ma = A($-1) ; na = A($)
  mb = B($-1) ; nb = B($)
  if na == mb then
    m = ma ; n = nb ; p = na
  else
    C = list(0,0)
    return
  end
  C = list()
  for i = 1:m
    for j = 1:n
      s = FRACTION(0,1)
      for k = 1:p
        t = F_MULTIPLICATION(A((i-1)*na+k),B((k-1)*nb+j))
        s = F_ADDITION(s,t)
      end
      C($+1) = s
    end
  end
  C($+1) = m ; C($+1) = n
endfunction
```

Rappel

L'expression AB n'a de sens que si le nombre de colonnes de A est égal au nombre de lignes de B.
Le produit matriciel (contrairement à la somme matricielle) n'est pas commutatif.
Le plus souvent : $AB \neq BA \Leftrightarrow$ M_MULTIPLICATION(A,B) $\neq$ M_MULTIPLICATION(B,A)

Propriétés

$(AB)C = A(BC)$
$A(B + C) = AB + AC$
$(A + B)C = AC + BC$
$k(AB) = (kA)B = A(kB)$
${}^t(AB) = {}^tB\ {}^tA$
$(AB)^{-1} = B^{-1}\ A^{-1}$
$tr(AB) = tr(BA)$
$det(AB) = det(A)\ det(B)$

Mathématique & Informatique

Puissance d'une matrice

Si A est une matrice carrée, on peut définir la matrice A^k.
Pour calculer A^k, on applique un algorithme nécessitant environ $\log_2(k)$ multiplications de matrice.
Par exemple, pour calculer A^{100} on effectue 8 multiplications de matrice au lieu de 99 en faisant $A \times A \times ... \times A$

$A \rightarrow A^2 \rightarrow A^4 \rightarrow A^8 \rightarrow A^{16} \rightarrow A^{32} \rightarrow A^{64} \Rightarrow$ 6 multiplications
$A^{100} = A^4 \times A^{32} \times A^{64} \Rightarrow$ 2 multiplications
Total : 6 + 2 = 8 multiplications

Fonction Scilab :

```
function B = M_PUISSANCE(A,k)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then
    B = list(0,0) ; return
  end
  B = M_IDENTITE(n) ; identite = %t
  while k <> 0
    if modulo(k,2) <> 0 then
      if identite then
        B = A ; identite = %f
      else
        B = M_MULTIPLICATION(B,A)
      end
    end
    k = floor(k/2)
    if k <> 0 then
      A = M_MULTIPLICATION(A,A)
    end
  end
endfunction
```

Rappel

$\forall$ k : $I^k = I$ (matrice identité)
$D = diag(d_1 , ... , d_n) \Rightarrow D^k = diag(d_1^k , ... , d_n^k)$
N est nilpotente si : $\exists$ p / $N^p = 0$ (matrice nulle)

Remarque

Dans le cas particulier où 2 matrices A et B commutent, c'est-à-dire lorsque AB = BA, la formule du binôme de **Newton** est applicable (avec la convention $A^0 = B^0 = I$) :

$$(A+B)^p = \sum_{k=0}^{p} \subset_p^k A^k B^{p-k}$$

Simplification d'une matrice

$A = (a_{ij}) = (p_{ij}/q_{ij}) = (r_{ij}/s_{ij}) = (b_{ij}) = B$ avec $pgcd(r_{ij},s_{ij}) = 1$

```
function B = M_SIMPLIFICATION(A)
  m = A($-1) ; n = A($)
  B = list()
  for i = 1:m*n
    B($+1) = F_REDUCTION(A(i))
  end
  B($+1) = m ; B($+1) = n
endfunction
```

Exemple : $\begin{pmatrix} -21/-14 & 285/19 \\ 0/12 & -286/66 \end{pmatrix} = \begin{pmatrix} 3/2 & 15 \\ 0 & -13/3 \end{pmatrix}$

Comparaison de deux matrices

$A = (a_{ij})$ et $B = (b_{ij})$

$a_{ij} = b_{ij} \quad (\forall i,j) \quad \Rightarrow \quad c = 0$

$a_{ij} > b_{ij} \quad (\forall i,j) \quad \Rightarrow \quad c = +1$

$a_{ij} < b_{ij} \quad (\forall i,j) \quad \Rightarrow \quad c = -1$

```
function c = M_COMPARAISON(A,B)
  ma = A($-1) ; na = A($)
  mb = B($-1) ; nb = B($)
  c = []
  if (ma == mb)&(na == nb) then
    m = ma ; n = na
  else
    return
  end
  for i = 1:m*n
    if i == 1 then
      c = F_COMPARAISON(A(1),B(1))
    else
      if F_COMPARAISON(A(i),B(i)) <> c then
        c = []
        break
      end
    end
  end
endfunction
```

Exemple : $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} > \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$

Test matrice nulle

$A = (a_{ij})$ avec $a_{ij} = 0 \quad (\forall i,j)$

```
function t = M_NUL(A)
  if length(A) == 0 then, t = %f, return, end
  t = M_COMPARAISON(A,M_ZERO(A($-1),A($))) == 0
endfunction
```

Norme d'une matrice

$$A = (a_{ij}) \quad \Rightarrow \quad \|A\|_l = \max_i \left(\sum_j |a_{ij}| \right) \ , \ \|A\|_c = \max_j \left(\sum_i |a_{ij}| \right) \ , \ \|A\|_e = \sqrt{\sum_{i,j} a_{ij}^2}$$

```
function N = M_NORME(A,t)
  m = A($-1) ; n = A($)
  if m*n == 0 then, N = [] ; return, end
  if t == 'l' then
    N = FRACTION(0,1)
    for i = 1:m
      s = FRACTION(0,1)
      for j = 1:n
        s = F_ADDITION(F_ABSOLU(A((i-1)*n+j)),s)
      end
      if F_COMPARAISON(s,N) == 1 then, N = s ; end
    end
  end
  if t == 'c' then
    N = FRACTION(0,1)
    for j = 1:n
      s = FRACTION(0,1)
      for i = 1:m
        s = F_ADDITION(F_ABSOLU(A((i-1)*n+j)),s)
      end
      if F_COMPARAISON(s,N) == 1 then, N = s ; end
    end
  end
  if t == 'e' then
    N = FRACTION(0,1)
    for i = 1:m*n
      N = F_ADDITION(F_MULTIPLICATION(A(i),A(i)),N)
    end
  end
endfunction
```

Le paramètre t permet de choisir le type de norme :

- Si t = 'l' (norme ligne), le nombre N est la valeur exacte de la plus grande somme des valeurs absolues des éléments de chaque ligne de A.
- Si t = 'c' (norme colonne), le nombre N est la valeur exacte de la plus grande somme des valeurs absolues des éléments de chaque colonne de A.
- Si t = 'e' (norme euclidienne), le nombre N est la valeur exacte de la somme des carrés de tous les éléments de A.

Remarque

On montre que toute valeur propre λ d'une matrice carrée A vérifie : $|\lambda| \le \text{Min}(\|A\|_l , \|A\|_c , \|A\|_e)$
Ce résultat s'applique à la matrice compagnon C du polynôme unitaire $P(x) = x^n + a_{n-1}x^{n-1} + ... + a_1x + a_0$

$$C = \begin{pmatrix} 0 & \cdots & 0 & -a_0 \\ 1 & \cdots & 0 & -a_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & -a_{n-1} \end{pmatrix}$$

En développant det(xI - C) suivant la dernière colonne, on obtient : $\det(xI - C) = P(x)$
Les valeurs propres de C sont les racines de P : $\det(xI - C) = 0 \Leftrightarrow P(x) = 0$
D'où la majoration des racines r de P : $|r| \le \text{Min}(\|C\|_l , \|C\|_c , \|C\|_e) \le \|C\|_l = \text{Max}(|a_0|, 1+|a_1|, ... , 1+|a_{n-1}|) \le 1 + \text{Max}|a_i|$

Trace d'une matrice

La trace d'une matrice carrée A est la somme de ses éléments diagonaux : $tr(A) = \sum a_{ii}$

```
function t = M_TRACE(A)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then, t = [] ; return, end
  t = FRACTION(0,1)
  for i = 1:n
    t = F_ADDITION(t,A((i-1)*n+i))
  end
endfunction
```

Déterminant d'une matrice

Pour calculer le déterminant d'une matrice carrée A, on triangularise A par r combinaisons linéaires et s permutations, puis on effectue le produit des éléments diagonaux de la matrice T obtenue : $det(A) = (-1)^s det(T) = (-1)^s \prod t_{ii}$

```
function d = M_DETERMINANT(A)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then, d = [] ; return, end
  s = 1
  for k = 1:n
    p = 0
    for i = k:n
      if ~F_NUL(A((i-1)*n+k)) then
        p = i ; break
      end
    end
    if p == 0 then, d = FRACTION(0,1) ; return, end
    if p <> k then
      s = -s
      for j = k:n
        t = A((k-1)*n+j)
        A((k-1)*n+j) = A((p-1)*n+j)
        A((p-1)*n+j) = t
      end
    end
    for i = k+1:n
      if ~F_NUL(A((i-1)*n+k)) then
        t = F_DIVISION(A((i-1)*n+k),A((k-1)*n+k))
        for j = k+1:n
          u = F_MULTIPLICATION(t,A((k-1)*n+j))
          A((i-1)*n+j) = F_SOUSTRACTION(A((i-1)*n+j),u)
        end
      end
    end
  end
  d = FRACTION(s,1)
  for i = 1:n
    d = F_MULTIPLICATION(d,A((i-1)*n+i))
  end
endfunction
```

Comatrice d'une matrice

Soit $A = (a_{ij})_{1 \le i,j \le n}$ une matrice carrée d'ordre n. Notons :

- Δ le déterminant de A
- Δ_{ij} le déterminant (mineur) obtenu à partir de Δ en supprimant la ligne i et la colonne j

L'élément $A_{ij} = (-1)^{i+j}\Delta_{ij}$ est appelé le cofacteur de a_{ij}
La comatrice de A est la matrice des cofacteurs :

$$com(A) = (A_{ij})_{1 \le i,j \le n}$$

Définissons une fonction *M_COMATRICE* qui calcule la comatrice d'une matrice A à coefficients rationnels :

```
function B = M_COMATRICE(A)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then, B = list(0,0) ; return, end
  B = list()
  for i = 1:n
    for j = 1:n
      M = list()
      for l = 1:n
        for c = 1:n
          if (c <> i)&(l <> j) then
            M($+1) = A((c-1)*n + l)
          end
        end
      end
      M($+1) = n-1 ; M($+1) = n-1
      if n > 1 then
        B($+1) = M_DETERMINANT(M)
      else
        B($+1) = FRACTION(1,1)
      end
      if modulo(i+j,2) <> 0 then, B($) = F_MOINS(B($)) ; end
    end
  end
  B($+1) = n ; B($+1) = n
endfunction
```

Remarques

Si A est réduite à un élément, on convient que le cofacteur correspondant vaut 1 afin que la relation $A^{-1} = {}^t com(A)/\Delta$ soit vérifiée pour n = 1.

On utilise parfois le terme d'adjointe pour désigner la comatrice et on la note adj(A). Attention à ne pas confondre avec un concept entièrement différent relatif à des matrices à coefficients complexes où l'adjointe est la matrice transposée des éléments conjugués, symbolisée par A^* : $A^* = {}^t\overline{A}$
Scilab adopte la notation A' (A prime) pour désigner ${}^t\overline{A}$ et la notation A'. (A prime point) pour désigner ${}^tA$
Pour une matrice dont tous les éléments sont des nombres réels, on a : ${}^t\overline{A} = {}^tA$
On appelle matrice complémentaire de A, la matrice $\widetilde{A}$ égale à la transposée de la comatrice : $\widetilde{A} = {}^t com(A)$

Matrice inverse d'une matrice

L'inverse d'une matrice carrée A est notée A^{-1}
Elle est l'unique matrice vérifiant : $A\,A^{-1} = A^{-1}\,A = I$

Méthode 1

Soit det(A) le déterminant non nul de A, on a : $A^{-1} = {}^t\mathrm{com}(A)/\det(A)$

Ecrivons une fonction *M_INVERSE* qui donne A^{-1} en utilisant la formule ci-dessus ; elle nécessite le calcul de n^2 déterminants d'ordre n-1 et 1 déterminant d'ordre n.

```
function B = M_INVERSE(A)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then, B = list(0,0) ; return, end
  d = M_DETERMINANT(A)
  if F_NUL(d) then
    printf('\n Matrice singuliere')
    B = list(0,0)
    return
  end
  B = M_COMATRICE(A)
  B = M_TRANSPOSEE(B)
  B = M_SCALAIRE(B,F_INVERSION(d))
endfunction
```

Le temps de calcul est long si n est grand, c'est pourquoi on préfère employer d'autres méthodes.

Méthode 2

La colonne j de A^{-1} est la solution du système linéaire Ax = b lorsque le vecteur b a toutes ses composantes nulles, sauf la j^e qui vaut 1. Si la matrice A est d'ordre n, on résoud n systèmes linéaires pour trouver A^{-1}.
La fonction *M_INVERSE* fait appel à la fonction *GAUSS* pour résoudre un système d'équations linéaires.

```
function B = M_INVERSE(A)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then, B = list(0,0) ; return, end
  B = M_VIDE(n,n)
  b0 = M_ZERO(n,1)
  for j = 1:n
    b = b0
    b(j) = FRACTION(1,1)
    x = GAUSS(A,b)
    if isequal(x,list(0,0)) then
      B = list(0,0)
      return
    end
    for i = 1:n
      B((i-1)*n + j) = x(i)
    end
  end
  B($+1) = n ; B($+1) = n
endfunction
```

Méthode 3

La matrice A est toujours la même dans les n systèmes linéaires à résoudre. Il est donc inutile de la trianguläriser n fois. L'algorithme amélioré consiste à construire une matrice augmentée (A,I) de dimension (n,2n) où I est la matrice identité d'ordre n, puis à diagonaliser A par des combinaisons linéaires de lignes pour aboutir à (I,A^{-1}).
De cette façon, on résoud les n systèmes linéaires simultanément. C'est la méthode de **Jordan**.

Désignons par M la matrice augmentée, par m_{ij} ses éléments ($1 \le i \le n$, $1 \le j \le 2n$), et par L_i sa ligne i.
Introduisons une écriture abrégée pour les opérations sur les lignes :

- Echange de 2 lignes différentes L_i et L_k : $L_i \longleftrightarrow L_k$
- Multiplication de la ligne L_i par une constante α : $L_i \longrightarrow \alpha L_i$
- Division de la ligne L_i par une constante α : $L_i \longrightarrow L_i / \alpha$
- Ajout à la ligne L_i d'une autre ligne L_k multipliée par une constante α : $L_i \longrightarrow L_i + \alpha L_k$

Algorithme :

```
M = (A,I)
pour  k = 1  à  n
|     p = Min{i, k ≤ i ≤ n :  mik ≠ 0}
|     si  p = ∅  alors
|     |     sortir (Matrice singulière)
|     fin
|     si  p ≠ k  alors
|     |     Lp ⟷ Lk
|     fin
|     si  mkk ≠ 1  alors
|     |     Lk ⟶ Lk /mkk
|     fin
|     pour  i = 1  à  n
|     |     si  i ≠ k  et  mik ≠ 0  alors
|     |     |     Li ⟶ Li - mikLk
|     |     fin
|     fin
fin
M ≡ (I,A-1)
```

On obtient une nouvelle fonction *M_INVERSE* :

```
function B = M_INVERSE(A)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then, B = list(0,0) ; return, end
  c = 2*n
  M = list()
  for i = 1:n
    for j = 1:c
      if j <= n then
        M($+1) = A((i-1)*n+j)
      else
        if j == n+i then
          M($+1) = FRACTION(1,1)
        else
          M($+1) = FRACTION(0,1)
        end
      end
    end
  end
  for k = 1:n
    p = 0
    for i = k:n
      if ~F_NUL(M((i-1)*c+k)) then
        p = i ; break
      end
    end
    if p == 0 then, printf('\n Matrice singuliere') ; B = list(0,0) ; return, end
    if p <> k then
      for j = k:c
        t = M((k-1)*c+j)
        M((k-1)*c+j) = M((p-1)*c+j)
        M((p-1)*c+j) = t
      end
    end
    a = M((k-1)*c+k)
    if F_COMPARAISON(a,FRACTION(1,1)) <> 0 then
      for j = k:c
        M((k-1)*c+j) = F_DIVISION(M((k-1)*c+j),a)
      end
    end
    for i = 1:n
      a = M((i-1)*c+k)
      if (i <> k)&(~F_NUL(a)) then
        for j = k:c
          t = F_MULTIPLICATION(M((k-1)*c+j),a)
          M((i-1)*c+j) = F_SOUSTRACTION(M((i-1)*c+j),t)
        end
      end
    end
  end
  B = list()
  for i = 1:n
    for j = 1:n
      B($+1) = M((i-1)*c+n+j)
    end
  end
  B($+1) = n ; B($+1) = n
endfunction
```

Polynôme caractéristique d'une matrice

La valeur exacte du polynôme caractéristique de la matrice carrée A d'ordre n s'obtient par la méthode de **Le Verrier.**
Posons : $P(x) = \det(xI - A) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$
Soient x_i les racines de $P(x) = 0$. On peut exprimer les coefficients a_i en fonction des sommes de **Newton** $s_k = \Sigma\, x_i^k$
On montre que $s_k = tr(A^k)$ et on en déduit l'algorithme de calcul des a_i :

$a_n = 1$
$B_0 = I$
pour i = 1 à n
| $A_i = B_{i-1}\, A$
| $a_{n-i} = - tr(A_i) / i$
| $B_i = A_i + a_{n-i}\, I$
fin

Coût de la méthode : n produits de matrice d'ordre n.
Intérêt : algorithme simple, sans cas particuliers.
La méthode de **Le Verrier** fournit indirectement tr(A), det(A) et A^{-1}

- Les coefficients a_{n-1} et a_0 du polynôme caractéristique sont, au signe près, la valeur de la trace et du déterminant de la matrice A.
  $a_{n-1} = - tr(A) \quad \Rightarrow \quad tr(A) = - a_{n-1}$
  $a_0 = (-1)^n \det(A) \quad \Rightarrow \quad \det(A) = (-1)^n a_0$

- On montre que la matrice B_n est nulle (application du théorème de **Cayley-Hamilton**)
  On a : $B_n = 0$
  $B_n = A_n + a_0\, I$
  $A_n = B_{n-1}\, A$
  Par suite : $0 = B_{n-1}\, A + a_0\, I$
  D'où : $- (1/a_0)\, B_{n-1}\, A = I$
  Donc : $A^{-1} = - (1/a_0)\, B_{n-1}$

Programme

```
function P = M_POLYNOME(A)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then, P = [] ; return, end
  I = M_IDENTITE(n)
  P = list(FRACTION(1,1))
  BI = I
  for i = 1:n
    AI = M_MULTIPLICATION(BI,A)
    P($+1) = F_MULTIPLICATION(M_TRACE(AI),FRACTION(-1,i))
    BI = M_ADDITION(AI,M_SCALAIRE(I,P($)))
  end
endfunction
```

Remarque

Si le polynôme caractéristique d'une matrice carrée A d'ordre n est défini par $P(x) = \det(A - xI)$, il faut multiplier tous les coefficients a_i précédents par $(-1)^n$.
$\det(xI - A) = x^n - tr(A)x^{n-1} + ... + a_i x^i + ... + (-1)^n \det(A) \quad \Rightarrow \quad \det(A - xI) = (-1)^n x^n + (-1)^{n-1} tr(A) x^{n-1} + ... + (-1)^n a_i x^i + ... + \det(A)$
L'avantage de la première définition est de disposer d'un polynôme caractéristique unitaire.

Valeurs propres entières d'une matrice

Les valeurs propres x_i de la matrice carrée A d'ordre n sont les solutions de l'équation polynomiale :

$$\det(xI - A) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0 = 0$$

Comme $a_i = p_i/q_i$, on peut multiplier tous les coefficients a_i par le ppcm des q_i afin d'obtenir une équation dont les coefficients sont des entiers relatifs :

$$b_n x^n + b_{n-1}x^{n-1} + ... + b_1 x + b_0 = 0 \quad \text{avec} \quad b_i = (p_i/q_i) \times \text{ppcm}(q_n, ... , q_0)$$

La fonction *racines* recherche les racines entières de cette équation.
Si elles existent, alors ce sont les valeurs propres entières de la matrice A.
Le programme est le suivant :

```
P = M_POLYNOME(A) ;
printf('\nPOLYNOME CARACTERISTIQUE'), AFFICHE(P)
n = length(P) ;
ppcm = ENTIER(1) ;
for i = 1:n
  ppcm = PPCM(ppcm,DENOMINATEUR(P(i))) ;
end
ppcm = list(ppcm,ENTIER(1)) ;
Q = list() ;
for i = 1:n
  Q($+1) = NUMERATEUR(F_MULTIPLICATION(P(i),ppcm)) ;
end
printf('\nPOLYNOME TRANSFORME'), AFFICHE(Q)
x = racines(Q) ;
printf('\nRACINES'), AFFICHE(x)
```

Prenons par exemple la matrice carrée A d'ordre 4 :

$$A = \begin{pmatrix} 4 & 7 & -\frac{5}{2} & \frac{1}{9} \\ \frac{21}{11} & -\frac{31}{13} & 10 & \frac{2}{17} \\ \frac{15}{14} & -1 & 0 & -\frac{5}{2} \\ \frac{56}{31} & \frac{19}{4} & \frac{1}{7} & \frac{17}{23} \end{pmatrix}$$

L'équation det(xI - A) = 0 s'écrit :

$$x^4 - \frac{704}{299}x^3 - \frac{4119801067}{436792356}x^2 + \frac{240880424}{764386623}x - \frac{166270831945}{940783536} = 0$$

On la transforme en :

$$12230185968\,x^4 - 28796156928\,x^3 - 115354429876\,x^2 + 3854086784\,x - 2161520815285 = 0$$

Cette dernière équation n'a pas de solution dans $\mathbb{Z}$, donc la matrice A ne possède pas de valeurs propres entières.

Autre exemple

Soit X un vecteur colonne quelconque de dimension n et ${}^tX$ le vecteur ligne transposé.
La matrice carrée symétrique d'ordre n définie par $A = X{}^tX$ a pour polynôme caractéristique $P(x) = x^{n-1}(x - {}^tXX)$, son déterminant est nul ($\det(A) = 0$) puisque $P(0) = 0$, ses valeurs propres sont $x = 0$ (racine d'ordre $n - 1$ de $P(x)$) et $x = {}^tXX$ (racine simple de $P(x)$).

Cas particulier : prenons $X = \left(i^p\right)_{1 \le i \le n}$, les valeurs propres de $A = X{}^tX$ sont $x = 0$ et $x = \| X \|^2 = \sum_{i=1}^{n} i^{2p}$

Pour effectuer le calcul exact dans le cas $n = 10$ et $p = 6$, on écrit :

```
exec("Scilab-Matrices.sce",-1)
n = 10 ; p = 6 ;
X = list() ;
for i = 1:n
    X($+1) = FRACTION(i^p,1) ;
end
X($+1) = n ; X($+1) = 1 ;
A = M_MULTIPLICATION(X,M_TRANSPOSEE(X)) ;
P = M_POLYNOME(A) ;
P_ECRIRE(P)
```

On trouve :

$$P(x) = x^{10} - 1367428536133x^9$$

D'où les valeurs propres entières de A :

$x = 0$ (multiplicité 9) et $x = 1367428536133$

On peut retrouver la valeur propre non nulle en effectuant directement le produit ${}^tXX$:

```
x = M_MULTIPLICATION(M_TRANSPOSEE(X),X) ;
M_ECRIRE(x)
```

Avec la fonction *spec* de Scilab, on obtient uniquement une approximation de ces valeurs propres :

```
X = ([1:10].^6)' ;
A = X*X' ;
spec(A)

- 0.0000040
- 0.0000002
- 1.912D-09
  3.568D-33
  5.996D-29
  2.915D-10
  0.0000004
  0.0000128
  0.0001556
  1.367D+12
```

Mathématique & Informatique

Factorisation LU d'une matrice

Une matrice carrée régulière A est dite factorisable LU s'il existe une matrice carrée triangulaire inférieure L ("Low") et une matrice carrée triangulaire supérieure U ("Up") telles que : A = LU
En choisissant la matrice L avec des 1 sur la diagonale, la décomposition est unique.

$$\begin{pmatrix} a11 & \cdots & a1n \\ \vdots & & \vdots \\ an1 & \cdots & ann \end{pmatrix} = \begin{pmatrix} 1 & & 0 \\ \vdots & \ddots & \\ ln1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} u11 & \cdots & u1n \\ & \ddots & \vdots \\ 0 & & unn \end{pmatrix}$$

L'identification des 2 membres de l'égalité A = LU conduit aux formules suivantes :

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \quad , \quad 1 \le i \le j \le n$$

$$l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \right) / u_{jj} \quad , \quad 1 \le j < i \le n$$

Théorème
Soit $A = (a_{ij})_{1 \le i,j \le n}$. Une condition nécessaire et suffisante pour que A soit factorisable LU est que :

$$\det(A_p) \neq 0 \text{ pour } p = 1, \ldots, n \quad \text{avec} \quad A_p = (a_{ij})_{1 \le i,j \le p}$$

Une matrice carrée régulière A n'est pas toujours factorisable LU. Cependant, on montre que pour toute matrice carrée régulière A, il existe une matrice de permutation inversible P, telle que PA soit factorisable LU.
La triangularisation de A par la méthode de Gauss avec échanges de lignes équivaut à choisir la matrice P.

$$PA = LU \quad \Leftrightarrow \quad A = P^{-1} L U$$

La factorisation LU est utilisée pour résoudre le système linéaire Ax = b
Supposons que l'on ait calculé les matrices P, L, U vérifiant PA = LU
On procède en 2 étapes :

1) $Ly = Pb \quad \Rightarrow \quad y$
2) $Ux = y \quad \Rightarrow \quad x$

La solution des systèmes triangulaires 1) et 2) s'obtient sans difficulté :

$$y_i = \sum_{j=1}^{n} p_{ij} b_j - \sum_{j=1}^{i-1} l_{ij} y_j \quad , \quad i = 1,\ldots,n$$

$$x_i = \left(y_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) / u_{ii} \quad , \quad i = n,\ldots,1$$

La méthode permet de résoudre rapidement un grand nombre de systèmes linéaires Ax = b où seul b change.

La factorisation LU est également à la base d'un algorithme de recherche des valeurs propres d'une matrice carrée régulière A. On construit la suite de matrices : $A_1 = A = L_1U_1$, $A_2 = U_1L_1 = L_2U_2$, $A_3 = U_2L_2 = L_3U_3$, ...

Programmation Scilab

```
function [P,L,U] = M_FACTORISATION(A)
|
|
endfunction
```

La fonction *M_FACTORISATION* fournit les matrices P, L, U telles que : PA = LU

```
function [P,L,U] = M_FACTORISATION(A)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then
    P = list(0,0) ; L = list(0,0) ; U = list(0,0)
    return
  end
  A0 = A
  P = M_IDENTITE(n)
  for k = 1:n
    p = 0
    for i = k:n
      if ~F_NUL(A((i-1)*n+k)) then
        p = i
        break
      end
    end
    if p == 0 then
      printf('\n Matrice singuliere')
      P = list(0,0) ; L = list(0,0) ; U = list(0,0)
      return
    end
    if p <> k then
      for j = k:n
        t = A((k-1)*n+j)
        A((k-1)*n+j) = A((p-1)*n+j)
        A((p-1)*n+j) = t
      end
      for j = 1:n
        t = P((k-1)*n+j)
        P((k-1)*n+j) = P((p-1)*n+j)
        P((p-1)*n+j) = t
      end
    end
    for i = k+1:n
      if ~F_NUL(A((i-1)*n+k)) then
        t = F_DIVISION(A((i-1)*n+k),A((k-1)*n+k))
        for j = k+1:n
          u = F_MULTIPLICATION(t,A((k-1)*n+j))
          A((i-1)*n+j) = F_SOUSTRACTION(A((i-1)*n+j),u)
        end
      end
    end
  end
```

```
  A = M_MULTIPLICATION(P,A0)
  L = M_ZERO(n,n)
  U = M_ZERO(n,n)
  for i = 1:n
    for j = i:n
      s = FRACTION(0,1)
      for k = 1:i-1
        t = F_MULTIPLICATION(L((i-1)*n+k),U((k-1)*n+j))
        s = F_ADDITION(s,t)
      end
      U((i-1)*n+j) = F_SOUSTRACTION(A((i-1)*n+j),s)
      if i == j then
        L((i-1)*n+i) = FRACTION(1,1)
        continue
      end
      s = FRACTION(0,1)
      for k = 1:i-1
        t = F_MULTIPLICATION(L((j-1)*n+k),U((k-1)*n+i))
        s = F_ADDITION(s,t)
      end
      L((j-1)*n+i) = F_DIVISION(F_SOUSTRACTION(A((j-1)*n+i),s),U((i-1)*n+i))
    end
  end
endfunction
```

Exemple

$$A \quad \Rightarrow \quad PA = LU$$

$$\begin{pmatrix} 1 & 2 & 0 \\ 2 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 2 & 0 \\ 2 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & -1/4 & 1 \end{pmatrix}\begin{pmatrix} 1 & 2 & 0 \\ 0 & -4 & 1 \\ 0 & 0 & 9/4 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/2 & 2 & 1 \end{pmatrix}\begin{pmatrix} 2 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & -9/2 \end{pmatrix}$$

EXEMPLE 1

Calculer :

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}^{50}$$

On écrit :

```
exec("Scilab-Matrices.sce",-1)
A = list(FRACTION(1,1),FRACTION(2,1),FRACTION(3,1),...
         FRACTION(4,1),FRACTION(5,1),FRACTION(6,1),...
         FRACTION(7,1),FRACTION(8,1),FRACTION(9,1),...
         3,3) ;
A50 = M_PUISSANCE(A,50) ;
M_ECRIRE(A50)
```

L'instruction *exec("Scilab-Matrices.sce",-1)* exécute le module "Scilab-Matrices" qui contient les fonctions matricielles. Posons :

$$A=\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},\ A^{50}=\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

On trouve :

a = 259022438954504370669068571523570131521605157117222705810640

b = 318264381501980161076067941080039026668494236515267167064856

c = 377506324049455951483067310636507921815383315913311628319072

d = 586583638902203177086618581668302744243309453600739772990606

e = 720743267602314685091893901962434603030717062143476075953481

f = 854902896302426193097169222256566461818124670686212378916356

g = 914144838849901983504168591813035356965013750084256840170572

h = 1123222153702649209107719862844830179392939887771684984842106

i = 1332299468555396434711271133876625001820866025459113129513640

Note : La matrice A est singulière, det(A) = 0, donc A^{50} aussi : det(A^{50}) = 0.

EXEMPLE 2

Le déterminant circulant d'ordre n est défini par :

$$\Delta_n = \begin{vmatrix} 1 & 2 & 3 & \cdots & \cdots & n \\ n & 1 & 2 & \cdots & \cdots & n-1 \\ n-1 & n & 1 & \cdots & \cdots & n-2 \\ \vdots & \vdots & \vdots & & & \vdots \\ \vdots & \vdots & \vdots & & & \vdots \\ 3 & 4 & 5 & \cdots & \cdots & 2 \\ 2 & 3 & 4 & \cdots & \cdots & 1 \end{vmatrix}$$

Calculer Δ_{25}

Le programme suivant construit la matrice A_n correspondant à Δ_n puis appelle la fonction *M_DETERMINANT* :

```
n = 25 ;
A = list() ;
for i = 1:n
  for j = 1:n
    if i == 1 then
      A($+1) = FRACTION(j,1) ;
    else
      A($+1) = F_SOUSTRACTION(A((i-2)*n+j),FRACTION(1,1)) ;
      if F_NUL(A($)) then, A($) = FRACTION(n,1) ; end
    end
  end
end
A($+1) = n ; A($+1) = n ;

d = M_DETERMINANT(A) ;
ECRIRE(d)
```

La valeur de Δ_{25} s'affiche :

$$\Delta_{25} = 461852778244065120816230773925781 25$$

Remarque

On montre que : $\Delta_n = (-1)^{n-1} (n+1) n^{n-1} / 2$
En particulier : $\Delta_{25} = 13 \times 25^{24}$
On vérifie que :

```
d = puissance(entier(25),24) ;
d = multiplication(d,entier(13)) ;
ecrire(d)
```

donne le même résultat.

EXEMPLE 3

Calculer la norme, la trace, le déterminant, l'inverse, le polynôme caractéristique, et la factorisation LU de la matrice :

$$A = \begin{pmatrix} 4 & 7 & -\frac{5}{2} & \frac{1}{9} \\ \frac{21}{11} & -\frac{31}{13} & 10 & \frac{2}{17} \\ \frac{15}{14} & -1 & 0 & -\frac{5}{2} \\ \frac{56}{31} & \frac{19}{4} & \frac{1}{7} & \frac{17}{23} \end{pmatrix}$$

Programme :

```
exec("Scilab-Matrices.sce",-1)

A = list(FRACTION(4,1),FRACTION(7,1),FRACTION(-5,2),FRACTION(1,9),...
         FRACTION(21,11),FRACTION(-31,13),FRACTION(10,1),FRACTION(2,17),...
         FRACTION(15,14),FRACTION(-1,1),FRACTION(0,1),FRACTION(-5,2),...
         FRACTION(56,31),FRACTION(19,4),FRACTION(1,7),FRACTION(17,23),...
         4,4) ;

printf('\n----------------------------------------------------------------------------')
nl = M_NORME(A,'l') ; nc = M_NORME(A,'c') ; ne = M_NORME(A,'e') ;
ECRIRE(nl) , ECRIRE(nc) , ECRIRE(ne)
printf('\n----------------------------------------------------------------------------')
t = M_TRACE(A) ;
ECRIRE(t)
printf('\n----------------------------------------------------------------------------')
d = M_DETERMINANT(A) ;
ECRIRE(d)
printf('\n----------------------------------------------------------------------------')
B = M_INVERSE(A) ;
M_ECRIRE(B)
printf('\n----------------------------------------------------------------------------')
P = M_POLYNOME(A) ;
AFFICHE(P)
printf('\n----------------------------------------------------------------------------')
 [P,L,U] = M_FACTORISATION(A) ;
M_ECRIRE(P) , M_ECRIRE(L) , M_ECRIRE(U)
printf('\n----------------------------------------------------------------------------')
```

Mathématique & Informatique

La norme de A (ligne, colonne, euclidienne) :

$$\|A\|_l = \frac{35034}{2431} \qquad \|A\|_c = \frac{787}{52} \qquad \|A\|_e = \sqrt{\frac{41095193269292353525}{190787562260030736}}$$

La trace et le déterminant de A :

$$\mathrm{tr}(A) = \frac{704}{299} \qquad \det(A) = -\frac{166270831945}{940783536}$$

L'inverse de A :

$$A^{-1} = \begin{pmatrix} \frac{1372558924044}{2161520815285} & \frac{695808113}{4055386145} & \frac{-99070477660}{432304163057} & \frac{-388163905052}{432304163057} \\ \frac{-9855783720}{33254166389} & \frac{-67533180}{811077229} & \frac{5742069256}{33254166389} & \frac{21344011500}{33254166389} \\ \frac{-424733799546}{2161520815285} & \frac{187084898}{4055386145} & \frac{39599036596}{432304163057} & \frac{143533334112}{432304163057} \\ \frac{844489915596}{2161520815285} & \frac{433269837}{4055386145} & \frac{-245239201494}{432304163057} & \frac{-277344819108}{432304163057} \end{pmatrix}$$

Le polynôme caractéristique de A :

$$P(x) = x^4 - \frac{704}{299}x^3 - \frac{4119801067}{436792356}x^2 + \frac{240880424}{764386623}x - \frac{166270831945}{940783536}$$

La factorisation LU de A :

$$\begin{pmatrix} 4 & 7 & -\frac{5}{2} & \frac{1}{9} \\ \frac{21}{11} & -\frac{31}{13} & 10 & \frac{2}{17} \\ \frac{15}{14} & -1 & 0 & -\frac{5}{2} \\ \frac{56}{31} & \frac{19}{4} & \frac{1}{7} & \frac{17}{23} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{21}{44} & 1 & 0 & 0 \\ \frac{15}{56} & \frac{3289}{6550} & 1 & 0 \\ \frac{14}{31} & -\frac{28171}{101525} & -\frac{4977859}{5629538} & 1 \end{pmatrix} \begin{pmatrix} 4 & 7 & -\frac{5}{2} & \frac{1}{9} \\ 0 & -\frac{3275}{572} & \frac{985}{88} & \frac{145}{2244} \\ 0 & 0 & -\frac{90799}{18340} & -\frac{599134}{233835} \\ 0 & 0 & 0 & -\frac{4323041630\,57}{2773448191\,08} \end{pmatrix}$$

EXEMPLE 4

On appelle série de **Liouville** d'une matrice A, la série de matrices $\sum_{k=0}^{\infty} A^k$.

Lorsque le rayon spectral de A est inférieur à 1, la série est convergente et a pour somme $S = (I - A)^{-1}$:

$$I + A + A^2 + \cdots + A^k + \cdots = (I - A)^{-1} \quad , \quad r(A) < 1$$

Calculer la somme exacte S de la série de **Liouville** de la matrice $A = \begin{pmatrix} 1/4 & -1/9 & 1/16 & -1/25 & 1/36 \\ -1/9 & 1/16 & -1/25 & 1/36 & -1/49 \\ 1/16 & -1/25 & 1/36 & -1/49 & 1/64 \\ -1/25 & 1/36 & -1/49 & 1/64 & -1/81 \\ 1/36 & -1/49 & 1/64 & -1/81 & 1/100 \end{pmatrix}$

La condition de convergence est vérifiée puisque :

$$\|A\|_l = \|A\|_c = \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \frac{1}{25} + \frac{1}{36} = \frac{1769}{3600} < 1 \quad \Rightarrow \quad r(A) < 1$$

Programmons :

```
A = list() ;
for i = 1:5
    for j = 1:5
        A($+1) = FRACTION((-1)^(i+j),(i+j)^2) ;
    end
end
A($+1) = 5 ; A($+1) = 5 ;
S = M_INVERSE(M_SOUSTRACTION(M_IDENTITE(5),A)) ;
M_ECRIRE(S)
```

Notons $S(i,j)$ l'élément situé à l'intersection de la ligne i et de la colonne j de la matrice $S = (I - A)^{-1}$, on obtient :

$S(1,1) = 399667379935433176539555750 0/291397591565369756410959734 9$
$S(1,2) = - 494019545070163932396360000/291397591565369756410959734 9$
$S(1,3) = 149046279922739285679600 00/153367153455457766532084071$
$S(1,4) = - 183835140927039774533894400/291397591565369756410959734 9$
$S(1,5) = 129085989148541906540010000/291397591565369756410959734 9$
$S(2,1) = - 494019545070163932396360000/291397591565369756410959734 9$
$S(2,2) = 317908923478033416923835000 0/291397591565369756410959734 9$
$S(2,3) = - 875237649772926132725760 0/153367153455457766532084071$
$S(2,4) = 114278389661495458859040000/291397591565369756410959734 9$
$S(2,5) = - 834458013826101152510400 00/291397591565369756410959734 9$
$S(3,1) = 149046279922739285679600 00/153367153455457766532084071$
$S(3,2) = - 875237649772926132725760 0/153367153455457766532084071$
$S(3,3) = 159206184716176171929600000/153367153455457766532084071$
$S(3,4) = - 419298512193302160384000 0/153367153455457766532084071$
$S(3,5) = 316363561349908807476000 0/153367153455457766532084071$
$S(4,1) = - 183835140927039774533894400/291397591565369756410959734 9$
$S(4,2) = 114278389661495458859040000/291397591565369756410959734 9$
$S(4,3) = - 419298512193302160384000 0/153367153455457766532084071$
$S(4,4) = 297315107747970469239852000 0/291397591565369756410959734 9$
$S(4,5) = - 458475808506806463609600 00/291397591565369756410959734 9$
$S(5,1) = 129085989148541906540010000/291397591565369756410959734 9$
$S(5,2) = - 834458013826101152510400 00/291397591565369756410959734 9$
$S(5,3) = 316363561349908807476000 0/153367153455457766532084071$
$S(5,4) = - 458475808506806463609600 00/291397591565369756410959734 9$
$S(5,5) = 295027256257812486962033760 0/291397591565369756410959734 9$

Mathématique & Informatique

Suite de Fibonacci généralisée

Il s'agit de la suite récurrente définie par :

$$u_n = au_{n-1} + bu_{n-2}$$
$$u_0 = \alpha$$
$$u_1 = \beta$$

Soient λ_1 et λ_2 les 2 racines de l'équation du second degré : $\lambda^2 - a\lambda - b = 0$. Trois cas peuvent se présenter :

1) L'équation possède 2 racines réelles distinctes $\lambda_1 \neq \lambda_2$
$u_n = C_1\lambda_1^n + C_2\lambda_2^n$ avec $C_1 + C_2 = \alpha$ et $\lambda_1 C_1 + \lambda_2 C_2 = \beta$
2) L'équation possède 1 racine réelle double $\lambda_1 = \lambda_2 = \lambda_0$
$u_n = [C_1 + C_2(n+1)]\lambda_0^n$ avec $C_1 + C_2 = \alpha$ et $\lambda_0 C_1 + 2\lambda_0 C_2 = \beta$
3) L'équation possède 2 racines complexes conjuguées $\lambda_1 = re^{i\theta}$ et $\lambda_2 = re^{-i\theta}$
$u_n = [C_1\cos(n\theta) + C_2\sin(n\theta)]r^n$ avec $C_1 = \alpha$ et $(r\cos\theta)C_1 + (r\sin\theta)C_2 = \beta$

On a les relations : $a = \lambda_1 + \lambda_2$ et $b = -\lambda_1\lambda_2$
Les coefficients a, b, α, β ne sont pas nécessairement des entiers.
Pour certaines valeurs du quadruplet (a,b,α,β), les nombres u_n portent des noms particuliers.

a	b	α	β	
1	1	0	1	Nombres de **Fibonacci** : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
1	1	2	1	Nombres de **Lucas** : 2, 1, 3, 4, 7, 11, 18, 29, 47, 76, ...
2	1	0	1	Nombres de **Pell** : 0, 1, 2, 5, 12, 29, 70, 169, 408, 985, ...
1	2	0	1	Nombres de **Jacobsthal** : 0, 1 ,1 ,3, 5, 11, 21, 43, 85, 171, ...

Propriété remarquable

Considérons la suite de **Fibonacci** généralisée de premier terme 0, c'est-à-dire la suite récurrente :

$$u_n = au_{n-1} + bu_{n-2}$$
$$u_0 = 0$$
$$u_1 = \beta$$

On établit l'identité :

$$\forall\ p \geq q\ :\quad u_{p+q}\,u_{p-q} = u_p^2 - (-b)^{p-q}\,u_q^2$$

Expression matricielle

A partir de $u_n = au_{n-1} + bu_{n-2}$, on écrit :

$$\begin{pmatrix} u_{n+1} \\ u_n \end{pmatrix} = \begin{pmatrix} a & b \\ 1 & 0 \end{pmatrix}\begin{pmatrix} u_n \\ u_{n-1} \end{pmatrix} = A\begin{pmatrix} u_n \\ u_{n-1} \end{pmatrix} = A^n\begin{pmatrix} u_1 \\ u_0 \end{pmatrix}$$

On peut donc calculer le terme u_n en élevant la matrice A à la puissance n :

$$A = \begin{pmatrix} a & b \\ 1 & 0 \end{pmatrix} \Rightarrow A^n = \begin{pmatrix} e & f \\ g & h \end{pmatrix} \Rightarrow u_n = gu_1 + hu_0$$

Application numérique : Calcul du nombre de Fibonacci F_{1000}

Programmation classique ⟶ Temps d'exécution ≈ **1h 10mn** :

```
Horloge
n = 1000 ;
a = FRACTION(1,1) ; b = FRACTION(1,1) ;
F0 = FRACTION(0,1) ; F1 = FRACTION(1,1) ;
u = F0 ; v = F1 ;
for i = 2:n
  x = F_MULTIPLICATION(a,u) ;
  y = F_MULTIPLICATION(b,v) ;
  w = F_ADDITION(x,y) ;
  u = v ;
  v = w ;
end
Fn = w ;
printf('\n F%i = ',n), ECRIRE(Fn)
Horloge
```

Programmation matricielle ⟶ Temps d'exécution ≈ **2mn** :

```
Horloge
n = 1000 ;
a = FRACTION(1,1) ; b = FRACTION(1,1) ;
F0 = FRACTION(0,1) ; F1 = FRACTION(1,1) ;
A = list(a,b,FRACTION(1,1),FRACTION(0,1),2,2) ;
An = M_PUISSANCE(A,n) ;
x = F_MULTIPLICATION(An(3),F1) ;
y = F_MULTIPLICATION(An(4),F0) ;
Fn = F_ADDITION(x,y) ;
printf('\n F%i = ',n), ECRIRE(Fn)
Horloge
```

On obtient un nombre entier à 209 chiffres :

F_{1000} = 43466557686937456435688527675040625802564660517371
78040248172908953655541794905189040387984007925516
92959225930803226347752096896232398733224711616429
96440906533187938298969649928516003704476137795166
849228875

Diminuons le temps d'exécution. Les calculs précédents sont effectués dans ℚ avec des fonctions matricielles qui font intervenir des nombres rationnels. En se plaçant dans ℕ les temps d'exécution seront nettement inférieurs.
Définissons de nouvelles fonctions matriciellles qui utilisent des entiers naturels.

//--- SOMME DE DEUX MATRICES ENTIERES

```
function C = M_addition(A,B)
  ma = A($-1) ; na = A($)
  mb = B($-1) ; nb = B($)
  if (ma == mb)&(na == nb) then
    m = ma ; n = na
  else
    C = list(0,0)
    return
  end
  C = list()
  for i = 1:m*n
    C($+1) = addition(A(i),B(i))
  end
  C($+1) = m ; C($+1) = n
endfunction
```

//--- PRODUIT DE DEUX MATRICES ENTIERES

```
function C = M_multiplication(A,B)
  ma = A($-1) ; na = A($)
  mb = B($-1) ; nb = B($)
  if na == mb then
    m = ma ; n = nb ; p = na
  else
    C = list(0,0)
    return
  end
  C = list()
  for i = 1:m
    for j = 1:n
      s = 0
      for k = 1:p
        t = multiplication(A((i-1)*na+k),B((k-1)*nb+j))
        s = addition(s,t)
      end
      C($+1) = s
    end
  end
  C($+1) = m ; C($+1) = n
endfunction
```

//--- MATRICE IDENTITE

```
function I = M_identite(n)
  I = list()
  for i = 1:n
    for j = 1:n
      if i == j then
        I($+1) = 1
      else
        I($+1) = 0
      end
    end
  end
  I($+1) = n ; I($+1) = n
endfunction
```

//--- PUISSANCE D'UNE MATRICE ENTIERE

```
function B = M_puissance(A,k)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then
    B = list(0,0) ; return
  end
  B = M_identite(n) ; identite = %t
  while k <> 0
    if modulo(k,2) <> 0 then
      if identite then
        B = A ; identite = %f
      else
        B = M_multiplication(B,A)
      end
    end
    k = floor(k/2)
    if k <> 0 then
      A = M_multiplication(A,A)
    end
  end
endfunction
```

Les nouveaux programmes sont :

Programmation classique ⟶ Temps d'exécution ≈ **2mn 30s** :

```
Horloge
n = 1000 ;
a = 1 ; b = 1 ;
F0 = 0 ; F1 = 1 ;
u = F0 ; v = F1 ;
for i = 2:n
  x = multiplication(a,u) ;
  y = multiplication(b,v) ;
  w = addition(x,y) ;
  u = v ;
  v = w ;
end
Fn = w ;
printf('\n F%i = ',n), ecrire(Fn)
Horloge
```

Programmation matricielle ⟶ Temps d'exécution ≈ **7s** :

```
Horloge
n = 1000 ;
a = 1 ; b = 1 ;
F0 = 0 ; F1 = 1 ;
A = list(a,b,1,0,2,2) ;
An = M_puissance(A,n) ;
x = multiplication(An(3),F1) ;
y = multiplication(An(4),F0) ;
Fn = addition(x,y) ;
printf('\n F%i = ',n), ecrire(Fn)
Horloge
```

Résultat :

F_{1000} = 43466557686937456435688527675040625802564660517371
78040248172908953655541794905189040387984007925516
92959225930803226347752096896232398733224711616429
96440906533187938298969649928516003704476137795166
849228875

Equation de Fermat

Comme pour la suite Fibonacci, l'équation de Fermat $x^2 - ny^2 = 1$ peut se mettre sous forme matricielle.
La solution (x_k,y_k) d'ordre k se déduit de la solution fondamentale (x_1,y_1) par les relations de récurrence :

$$\begin{cases} x_k = x_1 x_{k-1} + n y_1 y_{k-1} \\ y_k = y_1 x_{k-1} + x_1 y_{k-1} \end{cases}$$

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix} = \begin{pmatrix} x_1 & n y_1 \\ y_1 & x_1 \end{pmatrix} \begin{pmatrix} x_{k-1} \\ y_{k-1} \end{pmatrix} = A \begin{pmatrix} x_{k-1} \\ y_{k-1} \end{pmatrix} = A^k \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Programme de calcul de la solution d'ordre k d'une équation de **Fermat** :

```
function [x,y] = FERMAT(n,k)
  m = intsqrt(n)
  m2= multiplication(m,m)
  if comparaison(m2,n) == 0 then
    x = 1 ; y = 0
    return
  end
  fc = list()
  a = 0 ; b = 1 ; q = 0
  p = multiplication(m,2)
  while ~isequal(q,p)
    t = addition(m,a)
    [q,r] = division(t,b)
    t1 = soustraction(m,r)
    t2 = multiplication(t1,t1)
    t3 = soustraction(n,t2)
    [t4,r] = division(t3,b)
    a = t1 ; b = t4
    fc($+1) = q
  end
  T = length(fc) - 1
  for i = 1:T-1
    fc($+1) = fc(i+1)
  end
  if modulo(T,2) == 0 then
    K = T-1
  else
    K = 2*T-1
  end
  x = 1 ; y = 0
  for i = K+1:-1:1
    z = x
    x = multiplication(fc(i),x)
    x = addition(x,y)
    y = z
  end
  a11 = x ; a12 = multiplication(n,y)
  a21 = y ; a22 = x
  A = list(a11,a12,a21,a22,2,2)
  Ak = M_puissance(A,k)
  x = Ak(1) : y = Ak(3)
endfunction
```

Mathématique & Informatique

Application numérique : calcul du 1000^{e} nombre carré triangulaire

L'équation de Fermat $x^2 - 8y^2 = 1$ a pour solution fondamentale $(x_1,y_1) = (3,1)$.
La solution (x_{1000},y_{1000}) d'ordre 1000 conduit au nombre carré triangulaire $t_{1000} = y_{1000}^2$

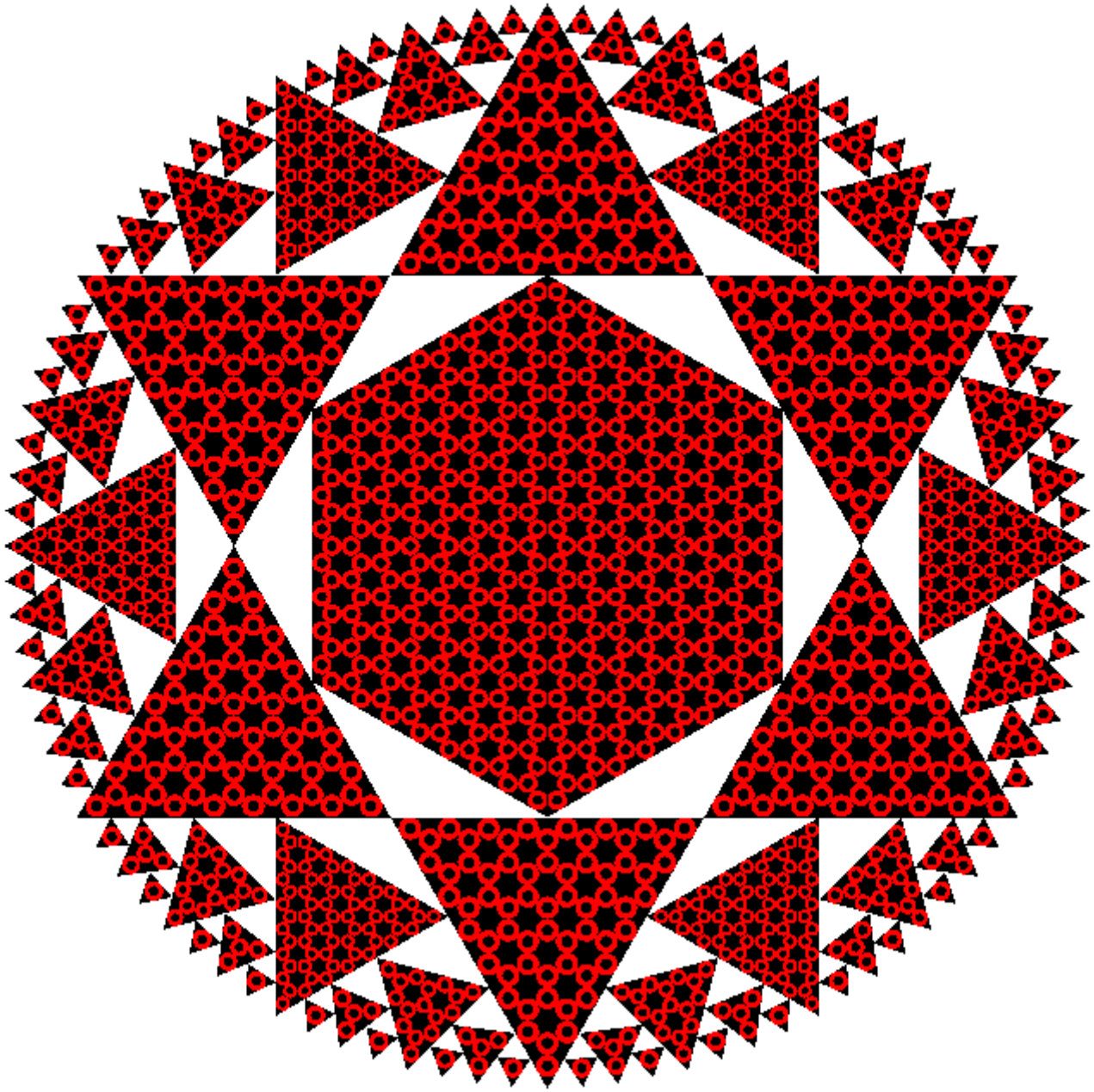
Programme :

```
exec("Scilab-Equations.sce",-1)
[x1000,y1000] = FERMAT(8,1000) ;
t1000 = puissance(y1000,2) ;
ecrire(t1000)
```

Solution :

```
39590535234784666605615849559208259715650623829209
91150303929884378432360965339379870000480420311419
98327908777283843450397303436305250560532461459817
65882146307037521335571079149701754049988543353500
63533843576509009175306794518782210687634159533553
49632952349167434238395612964884733639972925818810
01443619404174724416350495093422732400200427128825
27203600083052375965920245035800937469704826238285
09520359883414550556024714259918783947915616339535
43634963845963995458547814731105676969935039830339
88418123919019741949121422225678021660807381766387
37052597650704882791220715281013776363923532304730
95221955572363916799497958044454173419627786088372
48045986723252564806412204700914991432050055027227
62156230108121709738494511415053206214322453324598
82236185438603385299055207714827938460946486461780
33610072440757724625113100047498624294415670218661
25198523784038885643574604162881875204856501729560
90365932563363172718899505890588874374709499988825
52905057776632628703570767876515236316782725098447
73219612904461631090357760878417482680463537387540
75914038564117091794127359101438046206790859295795
16852697370458056036980311296155700318434115385037
11432373579650048945045772876494400281609291249240
58913551729965738214276984372566773363950838224484
81437989029089338254394584252480508279905949997582
54002506913969567958524783563469440577214033404870
04292123697035446841156213870769878335736681443757
22952027978370140509446975918130185858399847367514
73844608334383927943959203529365675128162080696683
50743546044905265151524923538496
```

Mathématique & Informatique



RÉDUCTION DES FORMES QUADRATIQUES

Soit $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$
Une forme quadratique q est un polynôme homogène du second degré par rapport à l'ensemble des variables x_i :

$$q(x) = \sum_{i=1}^{n} a_{ii} x_i^{\,2} + 2 \sum_{i<j} a_{ij} x_i x_j$$

On montre que l'on peut décomposer q en une somme de carrés :

$$q(x) = \sum_{i=1}^{r} k_i \, y_i^{\,2} \quad , \quad k_i \neq 0$$

où y_i est une forme linéaire en $x_1, x_2, \dots, x_n$.
Cette décomposition n'est pas unique mais elle admet toujours le même nombre P de carrés positifs et le même nombre N de carrés négatifs (Loi d'inertie de **Sylvester**). Le couple (P,N) est appelé la signature de q.
Notons X le vecteur colonne des x_i , la forme quadratique $q(x)$ s'exprime de manière matricielle :

$$q(x) = {}^t X \, A \, X$$

où $A = (a_{ij})$ est une matrice carrée symétrique réelle, c'est-à-dire vérifiant $a_{ij} = a_{ji}$ ou encore ${}^tA = A$
La ligne i de A correspond aux coefficients de $\frac{1}{2} \, \partial q / \partial x_i$
On établit que A est diagonalisable par une matrice régulière P telle que :

$${}^t P \, A \, P = D \quad \Rightarrow \quad A = {}^t P^{-1} \, D \, P^{-1}$$

où $D = (d_{ij})$ est une matrice diagonale, c'est-à-dire vérifiant $d_{ij} = 0$ pour $i \neq j$
Les matrices A et D sont dites congruentes (et non pas semblables). On déduit :

$$q(x) = {}^t X \, ({}^t P^{-1} \, D \, P^{-1}) \, X = {}^t (P^{-1} X) \, D \, (P^{-1} X)$$

En posant $P^{-1} = (p_{ij})$, on obtient :

$$q(x) = \sum_{i=1}^{r} d_{ii} \left(\sum_{j=1}^{n} p_{ij} x_j \right)^2$$

Le rang de q est le nombre r de carrés, donc le nombre d'éléments diagonaux non nuls de D.

Remarque
Une matrice symétrique a toutes ses valeurs propres réelles, mais n'est pas nécessairement régulière ; si elle admet la valeur propre 0, alors son déterminant est nul. Le rang de q est égal au nombre de valeurs propres non nulles.

Mathématique & Informatique

Les opérations élémentaires sur une matrice

Les opérations élémentaires sur une matrice M reviennent à multiplier cette matrice par des matrices élémentaires E déduites de la matrice identité I. Les multiplications matricielles s'effectuent à gauche pour les opérations sur les lignes et à droite pour les opérations sur les colonnes.
Pour simplifier, notons L_i la ligne i de M, C_j la colonne j de M, m_{ij} l'élément de la ligne i et de la colonne j.
La flèche $\longrightarrow$ désigne une opération de remplacement, la double flèche $\longleftrightarrow$ une opération de permutation.
Les différentes opérations élémentaires sont les suivantes :

- $L_i \longrightarrow L_i + \lambda L_k \quad \Leftrightarrow \quad M' = E_1\, M$ ($E_1 \equiv I$ où l'on a ajouté la valeur λ en ligne i, colonne $k \neq i$)
- $C_j \longrightarrow C_j + \lambda C_k \quad \Leftrightarrow \quad M' = M\, E_2$ ($E_2 \equiv I$ où l'on a ajouté la valeur λ en colonne j, ligne $k \neq j$)
- $L_i \longleftrightarrow L_k \quad \Leftrightarrow \quad M' = E_3\, M$ ($E_3 \equiv I$ où l'on a échangé les lignes i et $k \neq i$)
- $C_j \longleftrightarrow C_k \quad \Leftrightarrow \quad M' = M\, E_4$ ($E_4 \equiv I$ où l'on a échangé les colonnes j et $k \neq j$)
- $L_i \longrightarrow \lambda L_i \quad \Leftrightarrow \quad M' = E_5\, M$ ($E_5 \equiv I$ où l'on a multiplié la ligne i par la valeur $\lambda \neq 0$)
- $C_j \longrightarrow \lambda C_j \quad \Leftrightarrow \quad M' = M\, E_6$ ($E_6 \equiv I$ où l'on a multiplié la colonne j par la valeur $\lambda \neq 0$)

Les matrices élémentaires E sont inversibles puisqu'elles ont toutes un déterminant non nul :
$\det(E_1) = \det(E_2) = +1$; $\det(E_3) = \det(E_4) = -1$; $\det(E_5) = \det(E_6) = \lambda$
Les opérations E_1, E_2, E_3, E_4 correspondent à des matrices unimodulaires ($\det(E) = \pm 1$).
Soit d le déterminant de M et d' le déterminant de M'. Les opérations élémentaires E_1 et E_2 ne changent pas le déterminant ($d' = d$), les opérations élémentaires E_3 et E_4 inversent son signe ($d' = -d$), et les opérations élémentaires E_5 et E_6 le multiplient par λ ($d' = \lambda d$).
Exprimons les opérations élémentaires inverses équivalentes :

- $(L_i \longrightarrow L_i + \lambda L_k)^{-1} \equiv L_i \longrightarrow L_i - \lambda L_k$
- $(C_j \longrightarrow C_j + \lambda C_k)^{-1} \equiv C_j \longrightarrow C_j - \lambda C_k$
- $(L_i \longleftrightarrow L_k)^{-1} \equiv L_i \longleftrightarrow L_k$
- $(C_j \longleftrightarrow C_k)^{-1} \equiv C_j \longleftrightarrow C_k$
- $(L_i \longrightarrow \lambda L_i)^{-1} \equiv L_i \longrightarrow \lambda^{-1} L_i$
- $(C_j \longrightarrow \lambda C_j)^{-1} \equiv C_j \longrightarrow \lambda^{-1} C_j$

La méthode fondamentale des opérations élémentaires est à la base de nombreux algorithmes de l'algèbre linéaire, algorithmes qui servent également à la démonstration des théorèmes.

Note

En programmation, rappelons que pour permuter deux nombres, il en faut un troisième (temporaire).
Les instructions : x = y ; y = x n'échangent pas x et y (y est toujours le même).
Il faut écrire : t = x ; x = y ; y = t

On peut aussi adopter une syntaxe inspirée du langage *Pascal* pour distinguer les différentes instructions :
x = y est l'opération d'égalité booléenne entre les variables x et y (x égal y)
x := y est l'opération d'affectation de la variable y à la variable x (x prend pour valeur y)
x :=: y est l'opération d'échange des variables x et y (x échangé avec y)

Algorithme de calcul de D et P

Pour obtenir la matrice D et la matrice P, on construit la matrice augmentée (A,I) de dimension (n,2n), où I est la matrice identité d'ordre n, puis on diagonalise A en effectuant une suite d'opérations élémentaires sur les lignes et la même suite d'opérations élémentaires sur les colonnes. Finalement on aboutit à la matrice $(D,{}^tP)$.
Dans l'algorithme $D = {}^tPAP$ ci-dessous, notons M la matrice augmentée (A,I), L_i la ligne i de M, C_j la colonne j de M, m_{ij} l'élément situé à l'intersection de la ligne i et de la colonne j de M.

```
M = (A,I)
pour  k = 1  à  n
|     si  m_kk = 0  alors
|     |     si  ∃ m_ii ≠ 0, k < i ≤ n  alors
|     |     |     L_i ←→ L_k
|     |     |     C_i ←→ C_k
|     |     sinon si  ∃ m_ij ≠ 0, k ≤ i < j ≤ n  alors
|     |     |     L_i ⟶ L_i + L_j
|     |     |     C_i ⟶ C_i + C_j
|     |     |     L_i ←→ L_k
|     |     |     C_i ←→ C_k
|     |     sinon
|     |     |     sortir
|     |     fin
|     fin
|     pour  i = k+1  à  n
|     |     L_i ⟶ L_i - (m_ik /m_kk)L_k
|     fin
|     pour  j = k+1  à  n
|     |     C_j ⟶ C_j - (m_kj /m_kk)C_k
|     fin
fin
M ≡ (D,tP)
```

Remarque

Les matrices A et D ont le même déterminant puisque la matrice P est unimodulaire et $\det({}^tP) = \det(P)$.
La matrice P est inversible, donc ${}^tP$ également et on a : $({}^tP)^{-1} = {}^t(P^{-1})$.

Programme de calcul de D et P

Introduisons la fonction suivante : *function [D,P] = M_SYMETRIQUE(A)*
Pour une matrice symétrique A, dont les éléments sont des nombres rationnels, la fonction Scilab *M_SYMETRIQUE* renvoie la matrice diagonale D et la matrice régulière P, toutes deux à coefficients rationnels, telles que : $D = {}^tPAP$

```
function [D,P] = M_SYMETRIQUE(A)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then, D = list(0,0) ; P = list(0,0) ; return, end
  B = M_TRANSPOSEE(A)
  if M_COMPARAISON(A,B) <> 0 then, D = list(0,0) ; P = list(0,0) ; return, end
  c = 2*n
  M = list()
  for i = 1:n
    for j = 1:c
      if j <= n then
        M($+1) = A((i-1)*n+j)
      else
        if j == n+i then
          M($+1) = FRACTION(1,1)
        else
          M($+1) = FRACTION(0,1)
        end
      end
    end
  end
  for k = 1:n
    possible = %f
    if ~F_NUL(M((k-1)*c+k)) then
      possible = %t
    end
    if ~possible then
      for i = k+1:n
        if ~F_NUL(M((i-1)*c+i)) then
          for q = k:c
            t = M((k-1)*c+q)
            M((k-1)*c+q) = M((i-1)*c+q)
            M((i-1)*c+q) = t
          end
          for p = k:n
            t = M((p-1)*c+k)
            M((p-1)*c+k) = M((p-1)*c+i)
            M((p-1)*c+i) = t
          end
          possible = %t
          break
        end
      end
    end
```

```
  if ~possible then
    for i = k:n
      for j = i+1:n
        if ~F_NUL(M((i-1)*c+j)) then
          for q = k:c
            M((i-1)*c+q) = F_ADDITION(M((i-1)*c+q),M((j-1)*c+q))
          end
          for p = k:n
            M((p-1)*c+i) = F_ADDITION(M((p-1)*c+i),M((p-1)*c+j))
          end
          for q = k:c
            t = M((k-1)*c+q)
            M((k-1)*c+q) = M((i-1)*c+q)
            M((i-1)*c+q) = t
          end
          for p = k:n
            t = M((p-1)*c+k)
            M((p-1)*c+k) = M((p-1)*c+i)
            M((p-1)*c+i) = t
          end
          possible = %t
          break
        end
      end
      if possible then, break, end
    end
  end
  if ~possible then, break, end
  for i = k+1:n
    r = F_DIVISION(M((i-1)*c+k),M((k-1)*c+k))
    if F_NUL(r) then, continue, end
    for q = k:c
      t = F_MULTIPLICATION(r,M((k-1)*c+q))
      M((i-1)*c+q) = F_SOUSTRACTION(M((i-1)*c+q),t)
    end
  end
  for j = k+1:n
    r = F_DIVISION(M((k-1)*c+j),M((k-1)*c+k))
    if F_NUL(r) then, continue, end
    for p = k:n
      t = F_MULTIPLICATION(r,M((p-1)*c+k))
      M((p-1)*c+j) = F_SOUSTRACTION(M((p-1)*c+j),t)
    end
  end
end
```

```
  D = list()
  for i = 1:n
    for j = 1:n
      D($+1) = M((i-1)*c+j)
    end
  end
  D($+1) = n ; D($+1) = n
  P = list()
  for i = 1:n
    for j = 1:n
      P($+1) = M((j-1)*c+n+i)
    end
  end
  P($+1) = n ; P($+1) = n
endfunction
```

EXEMPLE

Transformer la forme quadratique suivante en une somme de carrés avec des coefficients rationnels :

$$q = \frac{1}{9}x^2 + \frac{13}{4}y^2 - 7z^2 + \frac{1}{2}t^2 + \frac{2}{17}xy - \frac{14}{5}xz - 6xt + \frac{42}{11}yz + \frac{11}{3}yt - 61zt$$

La matrice de la forme quadratique est :

$$A = \begin{pmatrix} \frac{1}{9} & \frac{1}{17} & -\frac{7}{5} & -3 \\ \frac{1}{17} & \frac{13}{4} & \frac{21}{11} & \frac{11}{6} \\ -\frac{7}{5} & \frac{21}{11} & -7 & -\frac{61}{2} \\ -3 & \frac{11}{6} & -\frac{61}{2} & \frac{1}{2} \end{pmatrix}$$

Exécutons le programme :

```
A = list(FRACTION(1,9),FRACTION(1,17),FRACTION(-7,5),FRACTION(-3,1),...
      FRACTION(1,17),FRACTION(13,4),FRACTION(21,11),FRACTION(11,6),...
      FRACTION(-7,5),FRACTION(21,11),FRACTION(-7,1),FRACTION(-61,2),...
      FRACTION(-3,1),FRACTION(11,6),FRACTION(-61,2),FRACTION(1,2),...
      4,4) ;
[D,P] = M_SYMETRIQUE(A) ;
Q = M_INVERSE(P) ;
printf('\n\n----------  A  ----------')
M_ECRIRE(A)
printf('\n\n----------  D  ----------')
M_ECRIRE(D)
printf('\n\n----------  P  ----------')
M_ECRIRE(P)
printf('\n\n----------  Q  ----------')
M_ECRIRE(Q)
```

On obtient :

$$D = \begin{pmatrix} \frac{1}{9} & 0 & 0 & 0 \\ 0 & \frac{3721}{1156} & 0 & 0 \\ 0 & 0 & -\frac{301910392}{11256025} & 0 \\ 0 & 0 & 0 & \frac{1134979495\,745}{1086877411\,2} \end{pmatrix}$$

$$P = \begin{pmatrix} 1 & -\frac{9}{17} & \frac{2667861}{204655} & -\frac{603916755}{86260112} \\ 0 & 1 & -\frac{168504}{204655} & \frac{36232559}{32347542} \\ 0 & 0 & 1 & -\frac{1600993295}{603820784} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$P^{-1} = \begin{pmatrix} 1 & \frac{9}{17} & -\frac{63}{5} & -27 \\ 0 & 1 & \frac{168504}{204655} & \frac{11866}{11163} \\ 0 & 0 & 1 & \frac{1600993295}{603820784} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

D'où la forme réduite :

$$\begin{aligned} q \;=\; & \frac{1}{9}\left(x + \frac{9}{17}y - \frac{63}{5}z - 27t\right)^2 \\ & + \frac{3721}{1156}\left(y + \frac{168504}{204655}z + \frac{11866}{11163}t\right)^2 \\ & - \frac{301910392}{11256025}\left(z + \frac{1600993295}{603820784}t\right)^2 \\ & + \frac{1134979495\,745}{1086877411\,2}(t)^2 \end{aligned}$$

Remarques :

- Le déterminant de A et D est identique, à savoir : Δ = - 226995899149 / 226599120
- En se plaçant dans $\mathbb{R}$, on peut mettre les coefficients de q à l'intérieur des carrés : $k(...)^2 = [k^{1/2}(\ldots)]^2$
  La forme quadratique s'écrit : $q = y_1^2 + y_2^2 - y_3^2 + y_4^2$, son rang est 4 et sa signature est (3,1)

Coniques & Quadriques

La signature d'une forme quadratique q(x) est un invariant qui ne dépend pas de la base choisie pour exprimer q(x). Autrement dit, la signature de q(x) est la même pour toutes les décompositions possibles de q(x) en carrés.
Elle permet notamment une classification des coniques et des quadriques.

Classification sommaire des coniques

Equation générale : $Ax^2 + A'y^2 + Bxy + Cx + C'y + D = 0$
Forme quadratique associée : $Ax^2 + A'y^2 + Bxy$
Matrice de la forme quadratique :

$$\begin{pmatrix} A & B/2 \\ B/2 & A' \end{pmatrix}$$

Signature de la forme quadratique :

- (2,0) ou (0,2) : ellipse
- (1,1) : hyperbole
- (1,0) ou (0,1) : parabole

Classification sommaire des quadriques

Equation générale : $Ax^2 + A'y^2 + A''z^2 + Bxy + B'xz + B''yz + Cx + C'y + C''z + D = 0$
Forme quadratique associée : $Ax^2 + A'y^2 + A''z^2 + Bxy + B'xz + B''yz$
Matrice de la forme quadratique :

$$\begin{pmatrix} A & B/2 & B'/2 \\ B/2 & A' & B''/2 \\ B'/2 & B''/2 & A'' \end{pmatrix}$$

Signature de la forme quadratique :

- (3,0) ou (0,3) : ellipsoïde
- (2,1) ou (1,2) : hyperboloïde à une ou deux nappes
- (2,0) ou (0,2) : paraboloïde elliptique
- (1,1) : paraboloïde hyperbolique

EXEMPLE

Quelle est la nature de la quadrique d'équation :

$$x^2 + \frac{23621}{39204}y^2 + \frac{23240}{13689}z^2 + \frac{21}{11}xy + \frac{34}{13}xz + \frac{30347}{11583}yz + \frac{1}{37}x - 221y - \frac{512}{491}z - 77 = 0$$

La forme quadratique associée est :

$$q = x^2 + \frac{23621}{39204}y^2 + \frac{23240}{13689}z^2 + \frac{21}{11}xy + \frac{34}{13}xz + \frac{30347}{11583}yz$$

Sa matrice est définie par :

$$A = \begin{pmatrix} 1 & \frac{21}{22} & \frac{17}{13} \\ \frac{21}{22} & \frac{23621}{39204} & \frac{30347}{23166} \\ \frac{17}{13} & \frac{30347}{23166} & \frac{23240}{13689} \end{pmatrix}$$

En appliquant la fonction *M_SYMETRIQUE* à A on obtient une matrice diagonale congruente D :

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\frac{25}{81} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

On en déduit que la signature de q est (1,1), et donc que l'équation de départ est celle d'un paraboloïde hyperbolique.

FORME CANONIQUE D'UNE MATRICE

Soit A une matrice de dimension (m,n) à coefficients rationnels.
Il existe une matrice carrée inversible P d'ordre m et une matrice carrée inversible Q d'ordre n, toutes les deux à coefficients rationnels, telles que :

$$PAQ = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 & & \\ 0 & 1 & & & 0 & & \\ \vdots & & \ddots & & \vdots & & 0 \\ \vdots & & & \ddots & \vdots & & \\ 0 & 0 & \cdots & \cdots & 1 & & \\ & & 0 & & & & 0 \end{pmatrix}$$

La matrice D = PAQ de dimension (m,n) est appelée forme canonique de A. Elle est unique.
Par contre, il n'y a pas unicité des matrices P et Q (par exemple : P' = kP et Q' = (1/k)Q $\Rightarrow$ D = P'AQ').
Le nombre de "1" sur la diagonale de D est égal au rang de A.

Dans le cas où la matrice A est une matrice carrée régulière, la matrice D est égale à la matrice identité.
On en déduit une nouvelle méthode pour calculer A^{-1} :

$PAQ = I \quad \Rightarrow \quad P = (AQ)^{-1} = Q^{-1}A^{-1}$ et $Q = (PA)^{-1} = A^{-1}P^{-1} \quad \Rightarrow \quad A^{-1} = QP$

Algorithme

Si A n'est pas nulle, on amène un coefficient non nul en position (1,1) en permutant les lignes et les colonnes adéquates.
On attribue à cet élément la valeur 1 en divisant tous les éléments de la 1re ligne ou de la 1re colonne.
Comme pour la méthode de **Gauss**, une série de combinaisons linéaires permet d'annuler tous les éléments de la 1ère ligne et de la 1ère colonne, sauf le premier élément.

$$\begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \vdots & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ 0 & \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{pmatrix}$$

En supprimant la 1re ligne et la 1re colonne de la matrice obtenue, on dispose d'une nouvelle matrice A de dimension (m-1,n-1) à laquelle on applique le même procédé.

Programme

```
function [D,P,Q] = M_CANONIQUE(A)
/
/
endfunction
```

La fonction *M_CANONIQUE* fournit les matrices D, P, Q telles que : D = PAQ

Application : système linéaire quelconque à coefficients rationnels

$\boxed{Ax = b}$ m équations à n inconnues

A = matrice rectangulaire de dimension (m,n)
x = vecteur colonne de dimension $(n,1)$
b = vecteur colonne de dimension $(m,1)$

1) Obtention de la forme canonique de A

$$D = PAQ = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}$$

D = matrice rectangulaire de dimension (m,n)
P = matrice carrée de dimension (m,m)
Q = matrice carrée de dimension (n,n)
I = matrice identité de dimension (r,r) où r est le rang du système linéaire

2) Transformation du système linéaire

$$P^{-1}DQ^{-1}x = b$$

$$DQ^{-1}x = Pb$$

$$\begin{pmatrix} R & S \\ 0 & 0 \end{pmatrix}\begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix}$$

$$\begin{cases} Ry + Sz = c \\ 0y + 0z = d \end{cases}$$

R = matrice carrée de dimension (r,r)
S = matrice rectangulaire de dimension $(r,n - r)$
y = vecteur colonne de dimension $(r,1)$
z = vecteur colonne de dimension $(n - r,1)$
c = vecteur colonne de dimension $(r,1)$ constitué des r premières lignes de Pb
d = vecteur colonne de dimension $(m - r,1)$ constitué des $m - r$ dernières lignes de Pb

3) Résolution du système linéaire

- $d \neq 0 \Rightarrow$ pas de solution, le système est impossible
- $d = 0 \Rightarrow$ indétermination d'ordre $n - r$
  z est indéterminé, sa valeur est quelconque
  y est solution du système de **Cramer** $Ry = c - Sz$
  $x = \begin{pmatrix} y \\ z \end{pmatrix}$ est solution du système linéaire $Ax = b$
  NB Si $r = n$, la solution est unique. Si $r < n$, il y a une infinité de solutions.

4) Intérêt

Avec cette méthode, les r premières inconnues du système linéaire sont toujours les inconnues principales et les $n - r$ dernières inconnues sont toujours les inconnues non principales auxquelles on peut donner la valeur que l'on veut, ce qui est en accord avec le fait que l'ensemble des solutions d'un système linéaire de rang r est un espace vectoriel de dimension $n - r$. En conclusion, parmi les n inconnues d'un système linéaire de rang r, il est possible de choisir à l'avance celles qui seront les inconnues principales et donc celles qui seront les inconnues non principales.

Algorithme : D = PAQ

D = A
P = I_m
Q = I_n
pour k = 1 à min(m,n)
| si $\exists\, d_{ij} \neq 0,\ k \leq i \leq m,\ k \leq j \leq n$ alors
| | $L_i \longleftrightarrow L_k$
| | $C_j \longleftrightarrow C_k$
| sinon
| | sortir
| fin
| si $n \leq m$ alors
| | $L_k \longrightarrow L_k / d_{kk}$
| sinon
| | $C_k \longrightarrow C_k / d_{kk}$
| fin
| pour i = k+1 à m
| | $L_i \longrightarrow L_i - d_{ik} L_k$
| fin
| pour j = k+1 à n
| | $C_j \longrightarrow C_j - d_{kj} C_k$
| fin
fin

```
function [D,P,Q] = M_CANONIQUE(A)
  m = A($-1) ; n = A($)
  D = A
  P = M_IDENTITE(m)
  Q = M_IDENTITE(n)
  for k = 1:min(m,n)
    zero = %t
    for i = k:m
      for j = k:n
        zero = F_NUL(D((i-1)*n+j))
        if ~zero then, break, end
      end
      if ~zero then, break, end
    end
    if zero then, break, end
    if i <> k then
      for q = k:n
        t = D((k-1)*n+q)
        D((k-1)*n+q) = D((i-1)*n+q)
        D((i-1)*n+q) = t
      end
      for q = 1:m
        t = P((k-1)*m+q)
        P((k-1)*m+q) = P((i-1)*m+q)
        P((i-1)*m+q) = t
      end
    end
    if j <> k then
      for p = k:m
        t = D((p-1)*n+k)
        D((p-1)*n+k) = D((p-1)*n+j)
        D((p-1)*n+j) = t
      end
      for p = 1:n
        t = Q((p-1)*n+k)
        Q((p-1)*n+k) = Q((p-1)*n+j)
        Q((p-1)*n+j) = t
      end
    end
    a = D((k-1)*n+k)
    if F_COMPARAISON(a,FRACTION(1,1)) <> 0 then
      if n <= m then
        for q = k:n
          D((k-1)*n+q) = F_DIVISION(D((k-1)*n+q),a)
        end
        for q = 1:m
          P((k-1)*m+q) = F_DIVISION(P((k-1)*m+q),a)
        end
      else
        for p = k:m
          D((p-1)*n+k) = F_DIVISION(D((p-1)*n+k),a)
        end
        for p = 1:n
          Q((p-1)*n+k) = F_DIVISION(Q((p-1)*n+k),a)
        end
      end
    end
```

```
    for i = k+1:m
      a = D((i-1)*n+k)
      if F_NUL(a) then, continue, end
      for q = k:n
        t = F_MULTIPLICATION(a,D((k-1)*n+q))
        D((i-1)*n+q) = F_SOUSTRACTION(D((i-1)*n+q),t)
      end
      for q = 1:m
        t = F_MULTIPLICATION(a,P((k-1)*m+q))
        P((i-1)*m+q) = F_SOUSTRACTION(P((i-1)*m+q),t)
      end
    end
    for j = k+1:n
      a = D((k-1)*n+j)
      if F_NUL(a) then, continue, end
      for p = k:m
        t = F_MULTIPLICATION(a,D((p-1)*n+k))
        D((p-1)*n+j) = F_SOUSTRACTION(D((p-1)*n+j),t)
      end
      for p = 1:n
        t = F_MULTIPLICATION(a,Q((p-1)*n+k))
        Q((p-1)*n+j) = F_SOUSTRACTION(Q((p-1)*n+j),t)
      end
    end
  end
endfunction
```

EXEMPLE

Mettre sous forme canonique la matrice A suivante :

$$A = \begin{pmatrix} 5 & 9 & 2 & 0 & 0 & 9 \\ \frac{16}{21} & \frac{30}{17} & -\frac{1}{32} & -\frac{7}{15} & -\frac{11}{10} & \frac{3}{2} \\ \frac{9}{14} & 15 & -\frac{47}{12} & -1 & \frac{24}{53} & -1 \\ -\frac{19}{27} & \frac{4}{23} & 13 & -7 & \frac{67}{43} & \frac{8}{13} \end{pmatrix}$$

On exécute le programme Scilab :

```
exec("Scilab-Matrices.sce",-1)

A = list(FRACTION(5,1),FRACTION(9,1),FRACTION(2,1),FRACTION(0,1),FRACTION(0,1),FRACTION(9,1),...
        FRACTION(16,21),FRACTION(30,17),FRACTION(-1,32),FRACTION(-7,15),FRACTION(-11,10),FRACTION(3,2),...
        FRACTION(9,14),FRACTION(15,1),FRACTION(-47,12),FRACTION(-1,1),FRACTION(24,53),FRACTION(-1,1),...
        FRACTION(-19,27),FRACTION(4,23),FRACTION(13,1),FRACTION(-7,1),FRACTION(67,43),FRACTION(8,13),...
        4,6) ;
[D,P,Q] = M_CANONIQUE(A) ;
```

On obtient les matrices D, P, Q telles que D = PAQ :

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{16}{105} & 1 & 0 & 0 \\ \frac{1225}{234} & -\frac{5491}{156} & 1 & 0 \\ -\frac{656804794}{71177157} & \frac{498877648}{7908573} & -\frac{44988586}{23725719} & 1 \end{pmatrix}$$

$$Q = \begin{pmatrix} \frac{1}{5} & -\frac{119}{26} & -\frac{29022}{114617} & -\frac{209986182}{4097600771} & \frac{1273637328554}{933843215709} & -\frac{1056642585}{4842619093} \\ 0 & \frac{595}{234} & \frac{38386}{343851} & \frac{63524206}{4097600771} & -\frac{1431697939126}{2801529647327} & \frac{8750744611}{4358357183} \\ 0 & 0 & \frac{14976}{114617} & \frac{239106528}{4097600771} & -\frac{10365464127696}{933843215709} & \frac{248906416}{4842619093} \\ 0 & 0 & 0 & -\frac{118628595}{4097600771} & -\frac{1857043659483}{933843215709} & \frac{1974369460}{4842619093} \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Mathématique & Informatique

FACTEURS INVARIANTS D'UNE MATRICE

Soit A une matrice de dimension (m,n) à coefficients entiers.
Il existe une matrice carrée inversible P d'ordre m et une matrice carrée inversible Q d'ordre n, toutes les deux à coefficients entiers, telles que :

$$PAQ = \begin{pmatrix} d_1 & 0 & \cdots & \cdots & 0 & \\ 0 & d_2 & & & 0 & \\ \vdots & & \ddots & & \vdots & 0 \\ \vdots & & & \ddots & \vdots & \\ 0 & 0 & \cdots & \cdots & d_r & \\ & & 0 & & & 0 \end{pmatrix}$$

Les d_i sont des entiers relatifs non nuls vérifiant la relation de divisibilité : $d_i \mid d_{i+1}$ $(i = 1, \ldots, r-1)$
On montre que la matrice diagonale D = PAQ est unique (au signe près).
Les matrices P et Q ne sont pas uniques, mais elles peuvent être choisies unimodulaires.
Les éléments $d_1, d_2, \ldots, d_r$ sont appelés les facteurs invariants de A.
Le premier élément de D est le pgcd des éléments de la matrice A : $|d_1| = \text{pgcd}(a_{ij})$

Remarque
Les matrices A et D ont le même rang r, c'est-à-dire le même nombre maximum de lignes ou de colonnes linéairement indépendantes ; elles sont dites équivalentes.
Les matrices P et Q ont un déterminant égal à ±1, par suite elles sont inversibles et P^{-1}, Q^{-1} ont des coefficients entiers.

Algorithme
Le principe est toujours le même : on applique une suite d'opérations élémentaires à la matrice A pour la transformer petit à petit en D. En effectuant des combinaisons linéaires en nombres entiers et des permutations, on reste dans $\mathbb{Z}$.

- les opérations sur les lignes sont recueillies dans une matrice P initialisée à I_m (matrice identité d'ordre m)
- les opérations sur les colonnes sont recueillies dans une matrice Q initialisée à I_n (matrice identité d'ordre n)

Programme

```
function [D,P,Q] = M_INVARIANTS(A)
//
//
endfunction
```

La fonction *M_INVARIANTS* travaille avec des entiers relatifs. Les résultats (c'est-à-dire les matrices D, P, Q) sont convertis en nombres rationnels pour rester compatible avec les autres fonctions matricielles.

Algorithme : D = PAQ

D = A
P = I_m
Q = I_n
pour k = 1 à min(m,n)
| $d_{uv} = Min\{|d_{ij}| \neq 0, k \leq i \leq m, k \leq j \leq n\}$
| si $d_{uv} \neq \varnothing$ alors
| | $L_u \longleftrightarrow L_k$ et $C_v \longleftrightarrow C_k$
| sinon
| | sortir
| fin
| répéter
| | pour i = k+1 à m
| | | $d_{ik} = qd_{kk} + r$
| | | $L_i \longrightarrow L_i - qL_k$
| | fin
| | pour j = k+1 à n
| | | $d_{kj} = qd_{kk} + r$
| | | $C_j \longrightarrow C_j - qC_k$
| | fin
| | $d_{uv} = Min\{|d_{ij}| \neq 0, j=k < i \leq m, i=k < j \leq n\}$
| | si $d_{uv} \neq \varnothing$ alors
| | | $L_u \longleftrightarrow L_k$ ou $C_v \longleftrightarrow C_k$
| | sinon
| | | si $\exists\, d_{ij}$, $k < i \leq m, k < j \leq n$: $d_{kk} \nmid d_{ij}$ alors
| | | | $C_k \longrightarrow C_k + C_j$
| | | sinon
| | | | sortir
| | | fin
| | fin
| fin
fin

```
function [D,P,Q] = M_INVARIANTS(A)
  function [D,P,Q] = permuter(D,P,Q,typ)
    if typ == 'l' then
      for c = k:n
        t = D((k-1)*n+c)
        D((k-1)*n+c) = D((u-1)*n+c)
        D((u-1)*n+c) = t
      end
      for c = 1:m
        t = P((k-1)*m+c)
        P((k-1)*m+c) = P((u-1)*m+c)
        P((u-1)*m+c) = t
      end
    end
    if typ == 'c' then
      for l = k:m
        t = D((l-1)*n+k)
        D((l-1)*n+k) = D((l-1)*n+v)
        D((l-1)*n+v) = t
      end
      for l = 1:n
        t = Q((l-1)*n+k)
        Q((l-1)*n+k) = Q((l-1)*n+v)
        Q((l-1)*n+v) = t
      end
    end
  endfunction
  m = A($-1) ; n = A($)
  D = list() ; P = list() ; Q = list()
  for i = 1:m
    for j = 1:n
      D($+1) = NUMERATEUR(A((i-1)*n+j))
    end
  end
  for i = 1:m
    for j = 1:m
      if i == j then
        P($+1) = ENTIER(1)
      else
        P($+1) = ENTIER(0)
      end
    end
  end
  for i = 1:n
    for j = 1:n
      if i == j then
        Q($+1) = ENTIER(1)
      else
        Q($+1) = ENTIER(0)
      end
    end
  end
```

```
for k = 1:min(m,n)
  d = []
  for i = k:m
    for j = k:n
      if ~NUL(D((i-1)*n+j)) then
        if isequal(d,[]) then
          u = i ; v = j
          d = ABSOLU(D((i-1)*n+j))
        elseif COMPARAISON(ABSOLU(D((i-1)*n+j)),d) < 0 then
          u = i ; v = j
          d = ABSOLU(D((i-1)*n+j))
        end
      end
    end
  end
  if ~isequal(d,[]) then
    if u <> k then, [D,P,Q] = permuter(D,P,Q,'l'), end
    if v <> k then, [D,P,Q] = permuter(D,P,Q,'c'), end
  else
    break
  end
  while %t
    for i = k+1:m
      if ~NUL(D((i-1)*n+k)) then
        [q,r] = DIVISION(D((i-1)*n+k),D((k-1)*n+k))
        for c = k:n
          t = MULTIPLICATION(q,D((k-1)*n+c))
          D((i-1)*n+c) = SOUSTRACTION(D((i-1)*n+c),t)
        end
        for c = 1:m
          t = MULTIPLICATION(q,P((k-1)*m+c))
          P((i-1)*m+c) = SOUSTRACTION(P((i-1)*m+c),t)
        end
      end
    end
    for j = k+1:n
      if ~NUL(D((k-1)*n+j)) then
        [q,r] = DIVISION(D((k-1)*n+j),D((k-1)*n+k))
        for l = k:m
          t = MULTIPLICATION(q,D((l-1)*n+k))
          D((l-1)*n+j) = SOUSTRACTION(D((l-1)*n+j),t)
        end
        for l = 1:n
          t = MULTIPLICATION(q,Q((l-1)*n+k))
          Q((l-1)*n+j) = SOUSTRACTION(Q((l-1)*n+j),t)
        end
      end
    end
```

```
    d = []
    for i = k+1:m
      if ~NUL(D((i-1)*n+k)) then
        if isequal(d,[]) then
          u = i ; v = k
          d = ABSOLU(D((i-1)*n+k))
        elseif COMPARAISON(ABSOLU(D((i-1)*n+k)),d) < 0 then
          u = i ; v = k
          d = ABSOLU(D((i-1)*n+k))
        end
      end
    end
    for j = k+1:n
      if ~NUL(D((k-1)*n+j)) then
        if isequal(d,[]) then
          u = k ; v = j
          d = ABSOLU(D((k-1)*n+j))
        elseif COMPARAISON(ABSOLU(D((k-1)*n+j)),d) < 0 then
          u = k ; v = j
          d = ABSOLU(D((k-1)*n+j))
        end
      end
    end
    if ~isequal(d,[]) then
      if v == k then, [D,P,Q] = permuter(D,P,Q,'l'), end
      if u == k then, [D,P,Q] = permuter(D,P,Q,'c'), end
    else
      iter = %f
      for i = k+1:m
        for j = k+1:n
          [q,r] = DIVISION(D((i-1)*n+j),D((k-1)*n+k))
          if ~NUL(r) then
            for l = k:m
              D((l-1)*n+k) = ADDITION(D((l-1)*n+k),D((l-1)*n+j))
            end
            for l = 1:n
              Q((l-1)*n+k) = ADDITION(Q((l-1)*n+k),Q((l-1)*n+j))
            end
            iter = %t
            break
          end
        end
        if iter then, break, end
      end
      if ~iter then, break, end
    end
  end
end
```

```
  for i = 1:m
    for j = 1:n
      D((i-1)*n+j) = list(D((i-1)*n+j),[1,1])
    end
  end
  D($+1) = m ; D($+1) = n
  for i = 1:m
    for j = 1:m
      P((i-1)*m+j) = list(P((i-1)*m+j),[1,1])
    end
  end
  P($+1) = m ; P($+1) = m
  for i = 1:n
    for j = 1:n
      Q((i-1)*n+j) = list(Q((i-1)*n+j),[1,1])
    end
  end
  Q($+1) = n ; Q($+1) = n
endfunction
```

Mathématique & Informatique

EXEMPLE 1

Quels sont les facteurs invariants de la matrice A suivante :

$$A = \begin{pmatrix} 8 & 8 & 2009 & 31 \\ 613 & 0 & 257 & 17 \\ 791 & -68 & 4 & -141 \\ -703 & 160 & 150 & -96 \end{pmatrix}$$

Le programme :

```
exec("Scilab-Matrices.sce",-1)

A = list(FRACTION(8,1),FRACTION(8,1),FRACTION(2009,1),FRACTION(31,1),...
        FRACTION(613,1),FRACTION(0,1),FRACTION(257,1),FRACTION(17,1),...
        FRACTION(791,1),FRACTION(-68,1),FRACTION(4,1),FRACTION(-141,1),...
        FRACTION(-703,1),FRACTION(160,1),FRACTION(150,1),FRACTION(-96,1),...
        4,4) ;
[D,P,Q] = M_INVARIANTS(A) ;
```

calcule les matrices D, P, Q telles que D = PAQ :

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -37551107864 \end{pmatrix}$$

$$P = \begin{pmatrix} 0 & 1 & -64 & 0 \\ -12 & -46694 & 2130379 & 23353 \\ 599037734999 & 2362987176589 1 & -10780766418400357 & -1181736745663747 \\ 673917450679 & 2658360569385381 & -12128362218733972 & -1329453836795013 \end{pmatrix}$$

$$Q = \begin{pmatrix} 0 & 7 & -719810042711 & 2402636854168852407663 \\ 0 & 19 & -1954427269696 & 6523636389460973218770 \\ 1 & 5200 & -534619588213347 & 1784494033128316277616684 \\ 0 & 29 & -2981764510128 & 9952760977823588128282 \end{pmatrix}$$

EXEMPLE 2

Quels sont les facteurs invariants de la matrice A suivante :

$$A = \begin{pmatrix} 710 & 497 & 568 & 497 \\ 4402 & 3124 & 3692 & 3124 \\ -142 & -142 & -284 & -142 \end{pmatrix}$$

Le programme :

```
exec("Scilab-Matrices.sce",-1)

A = list(FRACTION(710,1),FRACTION(497,1),FRACTION(568,1),FRACTION(497,1),...
        FRACTION(4402,1),FRACTION(3124,1),FRACTION(3692,1),FRACTION(3124,1),...
        FRACTION(-142,1),FRACTION(-142,1),FRACTION(-284,1),FRACTION(-142,1),...
        3,4) ;
[D,P,Q] = M_INVARIANTS(A) ;
```

calcule les matrices D, P, Q telles que D = PAQ :

$$D = \begin{pmatrix} -71 & 0 & 0 & 0 \\ 0 & 426 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$P = \begin{pmatrix} 1 & 0 & 4 \\ -2 & 0 & -7 \\ -6 & 1 & 1 \end{pmatrix}$$

$$Q = \begin{pmatrix} 0 & -1 & 2 & 0 \\ 1 & -2 & -4 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

EXEMPLE 3

Calculer le pgcd des 7 nombres :

$a1 = 37909408394210922470104077$
$a2 = 1262119108246645625376456$
$a3 = -\ 84223080952069776885286384$
$a4 = -\ 12790385222974513081054057$
$a5 = 25862698186873718383383796 07$
$a6 = 20901606819821195486381051$
$a7 = -\ 61053442301724856877917596$

On écrit :

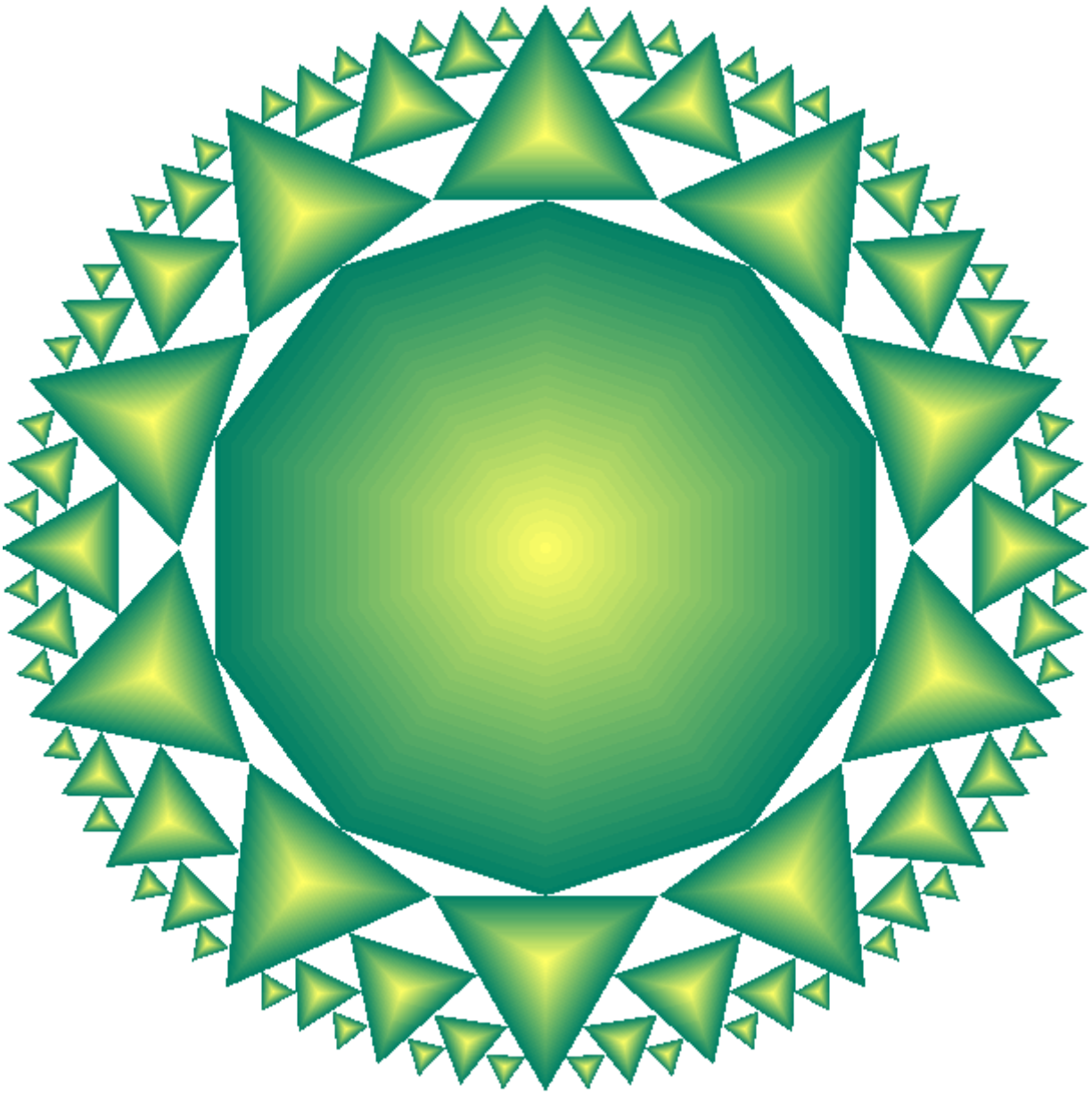
```
exec("Scilab-Matrices.sce",-1)

a1 = list([1,3,7,9,0,9,4,0,8,3,9,4,2,1,0,9,2,2,4,7,0,1,0,4,0,7,7],[1,1]) ;
a2 = list([1,1,2,6,2,1,1,9,1,0,8,2,4,6,6,4,5,6,2,5,3,7,6,4,5,6],[1,1]) ;
a3 = list([-1,8,4,2,2,3,0,8,0,9,5,2,0,6,9,7,7,6,8,8,5,2,8,6,3,8,4],[1,1]) ;
a4 = list([-1,1,2,7,9,0,3,8,5,2,2,2,9,7,4,5,1,3,0,8,1,0,5,4,0,5,7],[1,1]) ;
a5 = list([1,2,5,8,6,2,6,9,8,1,8,6,8,7,3,7,1,8,3,8,3,3,8,3,7,9,6,0,7],[1,1]) ;
a6 = list([1,2,0,9,0,1,6,0,6,8,1,9,8,2,1,1,9,5,4,8,6,3,8,1,0,5,1],[1,1]) ;
a7 = list([-1,6,1,0,5,3,4,4,2,3,0,1,7,2,4,8,5,6,8,7,7,9,1,7,5,9,6],[1,1]) ;
A = list(a1,a2,a3,a4,a5,a6,a7,1,7) ;
[D,P,Q] = M_INVARIANTS(A) ;
pgcd = ABSOLU(NUMERATEUR(D(1))) ;
ECRIRE(pgcd)
```

On obtient :

$\text{pgcd}(a_1,a_2,a_3,a_4,a_5,a_6,a_7) = 392488048668479537$

Mathématique & Informatique



Mathématique & Informatique

POLYNÔMES

Un polynôme à coefficients dans $\mathbb{Q}$ est une liste de nombres rationnels ordonnés du coefficient du monôme dominant au coefficient du terme constant :

$$P(x) = a_n x^n + \ldots + a_1 x + a_0 \quad \Leftrightarrow \quad P = \text{list}(a_n , \ldots , a_1 , a_0)$$

Les a_i sont des nombres rationnels représentés eux aussi par une liste :

$$a_i = n_i / d_i \quad \Leftrightarrow \quad a_i = \text{list}(n_i , d_i)$$

Le numérateur n_i et le dénominateur d_i sont des entiers relatifs assimilés à des vecteurs.

Note : Un polynôme non défini est identique à une liste vide : $P(x) = \varnothing \Leftrightarrow P = \text{list}()$

Pour afficher à l'écran la valeur d'un polynôme, on utilise la fonction *P_ECRIRE*

```
function P_ECRIRE(A)
  n = length(A)
  if n == 0 then, return, end
  printf('\n Polynôme de degré %i :',n-1)
  printf('\n ======================')
  for i = 1:n
    if F_NUL(A(i)) & n <> 1 then, continue, end
    printf('\n ')
    ECRIRE(A(i))
    k = length(A) - i
    if k > 1 then, printf('   x%i',k), end
    if k == 1 then, printf('   x'), end
    if k == 0 then, printf(' '), end
  end
endfunction
```

Note : Le degré d'un polynôme est égal à la longueur de la liste de ses coefficients moins 1.

Par exemple : $P(x) = x^2 - (3/29)x + 150$ s'écrit P = list(list([1,1],[1,1]),list([-1,3],[1,2,9]),list([1,1,5,0],[1,1]))
L'exécution de l'instruction P_ECRIRE(P) donne :

```
 Polynôme de degré 2 :
 ================
 1   x2
 - 3 / 29   x
 150
```

La fonction *P_SIMPLIFICATION* supprime les coefficients nuls en tête de liste, c'est dire les termes inutiles :

```
function B = P_SIMPLIFICATION(A)
  B = A
  while length(B) > 1
    if ~F_NUL(B(1)) then, break, end
    B(1) = null()
  end
endfunction
```

Ecrivons les fonctions Scilab d'addition, de soustraction, de multiplication et de division des polynômes.

```
//---  ADDITION DE DEUX POLYNOMES

function C = P_ADDITION(A,B)
  C = list()
  if length(A) == 0 | length(B) == 0 then, return, end
  A = P_SIMPLIFICATION(A)
  B = P_SIMPLIFICATION(B)
  na = length(A)
  nb = length(B)
  n = max(na,nb)
  for i = 1:n
    C($+1) = FRACTION(0,1)
    if n-i < na then
      C(i) = F_ADDITION(C(i),A(na-n+i))
    end
    if n-i < nb then
      C(i) = F_ADDITION(C(i),B(nb-n+i))
    end
  end
  C = P_SIMPLIFICATION(C)
endfunction


//---  SOUSTRACTION DE DEUX POLYNOMES

function C = P_SOUSTRACTION(A,B)
  C = list()
  if length(A) == 0 | length(B) == 0 then, return, end
  A = P_SIMPLIFICATION(A)
  B = P_SIMPLIFICATION(B)
  na = length(A)
  nb = length(B)
  n = max(na,nb)
  for i = 1:n
    C($+1) = FRACTION(0,1)
    if n-i < na then
      C(i) = F_ADDITION(C(i),A(na-n+i))
    end
    if n-i < nb then
      C(i) = F_SOUSTRACTION(C(i),B(nb-n+i))
    end
  end
  C = P_SIMPLIFICATION(C)
endfunction
```

```
//---  MULTIPLICATION DE DEUX POLYNOMES

function C = P_MULTIPLICATION(A,B)
  C = list()
  if length(A) == 0 | length(B) == 0 then, return, end
  A = P_SIMPLIFICATION(A)
  B = P_SIMPLIFICATION(B)
  na = length(A)
  nb = length(B)
  n = na + nb - 1
  for i = 1:n
    C($+1) = FRACTION(0,1)
  end
  for i = 1:na
    for j = 1:nb
      X = F_MULTIPLICATION(A(i),B(j))
      C(i+j-1) = F_ADDITION(C(i+j-1),X)
    end
  end
  C = P_SIMPLIFICATION(C)
endfunction

//---  DIVISION DE DEUX POLYNOMES

function [Q,R] = P_DIVISION(A,B)
  Q = list()
  R = list()
  if length(A) == 0 | length(B) == 0 then, return, end
  A = P_SIMPLIFICATION(A)
  B = P_SIMPLIFICATION(B)
  Q = P_CONSTANT(0,1)
  R = A
  if P_NUL(B) then, return, end
  while length(R) >= length(B) then
    P = list()
    n = length(R) - length(B) + 1
    for i = 1:n
      P($+1) = FRACTION(0,1)
    end
    P(1) = F_DIVISION(R(1),B(1))
    X = P_MULTIPLICATION(P,B)
    R = P_SOUSTRACTION(R,X)
    Q = P_ADDITION(Q,P)
    if length(P) == 1 then, break, end
  end
endfunction
```

Mathématique & Informatique

EXEMPLE 1

Effectuer la division exacte, selon les puissances décroissantes de x, des 2 polynômes A(x) et B(x) définis par :

$A(x) = 7x^{11} + 153245/121x^8 - 9587025404x^4 + 27/635479x^2 - 1/4x - 1234567890$
$B(x) = 1235/31x^5 - 394x^4 + x^3 - 12/17x^2 + x + 1$

On saisit le programme :

```
exec("Scilab-Polynomes.sce",-1)

A = list(list([1,7],[1,1]),list([0,0],[1,1]),list([0,0],[1,1]), ...
        list([1,1,5,3,2,4,5],[1,1,2,1]),list([0,0],[1,1]),list([0,0],[1,1]), ...
        list([0,0],[1,1]),list([-1,9,5,8,7,0,2,5,4,0,4],[1,1]),list([0,0],[1,1]), ...
        list([1,2,7],[1,6,3,5,4,7,9]),list([-1,1],[1,4]),list([-1,1,2,3,4,5,6,7,8,9,0],[1,1])) ;
B = list(list([1,1,2,3,5],[1,3,1]),list([-1,3,9,4],[1,1]),list([1,1],[1,1]), ...
        list([-1,1,2],[1,1,7]),list([1,1],[1,1]),list([1,1],[1,1])) ;
[Q,R] = P_DIVISION(A,B) ;

P_ECRIRE(A)
P_ECRIRE(B)
P_ECRIRE(Q)
P_ECRIRE(R)
```

La division du polynôme A(x) par le polynôme B(x) s'écrit A(x) = B(x)Q(x) + R(x), avec :

A(x) = 7 $\mathbf{x^{11}}$ + 153245/121 $\mathbf{x^8}$ - 9587025404 $\mathbf{x^4}$ + 27/635479 $\mathbf{x^2}$ - 1/4 **x** - 1234567890

B(x) = 1235/31 $\mathbf{x^5}$ - 394 $\mathbf{x^4}$ + $\mathbf{x^3}$ - 12/17 $\mathbf{x^2}$ + **x** + 1

Q(x) = 217 / 1235 $\mathbf{x^6}$
+ 2650438 / 1525225 $\mathbf{x^5}$
+ 32364141887 / 1883652875 $\mathbf{x^4}$
+ 965053301525239841 / 4785222345385625 $\mathbf{x^3}$
+ 11784768172720030977134 / 5909749596551246875 $\mathbf{x^2}$
+ 1189289940148442177038246 / 60318518609428015625 **x**
+ 298727606637999225628155322340 53 / 153232863082797883753671875

R(x) = - 145728032250985666119676957559382423 8 / 153232863082797883753671875 $\mathbf{x^4}$
- 280765703811840469578042865083 58 / 153232863082797883753671875 $\mathbf{x^3}$
+ 147586084236289321599921947128104047 93 / 127338194783298952176022618515625 $\mathbf{x^2}$
- 131576260758076380106494182396767 / 61293145233119153501468750 0 **x**
- 189206245215452478564798782003327803 / 153232863082797883753671875

EXEMPLE 2

Décomposer la fraction rationnelle $\dfrac{13x^7 - 7x^5 + 2014}{(x-1959)^5(x^2+x+1)^2}$ en éléments simples.

Solution

On sait qu'il existe des nombres rationnels $a, b, c, d, e, f, g, h, i$ tels que :

$$\frac{13x^7 - 7x^5 + 2014}{(x-1959)^5(x^2+x+1)^2} = \frac{a}{(x-1959)^5} + \frac{b}{(x-1959)^4} + \frac{c}{(x-1959)^3} + \frac{d}{(x-1959)^2} + \frac{e}{(x-1959)} + \frac{fx+g}{(x^2+x+1)^2} + \frac{hx+i}{(x^2+x+1)}$$

Réduisons au même dénominateur et procédons par identification :

$$\begin{aligned} 13x^7 - 7x^5 + 2014 = {} & a(x^2+x+1)^2 + b(x-1959)(x^2+x+1)^2 + c(x-1959)^2(x^2+x+1)^2 \\ & + d(x-1959)^3(x^2+x+1)^2 + e(x-1959)^4(x^2+x+1)^2 + fx(x-1959)^5 \\ & + g(x-1959)^5 + hx(x-1959)^5(x^2+x+1) + i(x-1959)^5(x^2+x+1) \end{aligned}$$

Posons :

$$\begin{aligned} P_0(x) &= 13x^7 - 7x^5 + 2014 \\ P_1(x) &= (x^2+x+1)^2 \\ P_2(x) &= (x-1959)(x^2+x+1)^2 \\ P_3(x) &= (x-1959)^2(x^2+x+1)^2 \\ P_4(x) &= (x-1959)^3(x^2+x+1)^2 \\ P_5(x) &= (x-1959)^4(x^2+x+1)^2 \\ P_6(x) &= x(x-1959)^5 \\ P_7(x) &= (x-1959)^5 \\ P_8(x) &= x(x-1959)^5(x^2+x+1) \\ P_9(x) &= (x-1959)^5(x^2+x+1) \end{aligned}$$

Les $n = 9$ inconnues $a, b, c, d, e, f, g, h, i$ sont les solutions du système linéaire suivant :

$$\begin{pmatrix} \alpha_{11} & \cdots & \cdots & \cdots & \alpha_{1j} & \cdots & \cdots & \cdots & \alpha_{19} \\ \vdots & & & & \vdots & & & & \vdots \\ \vdots & & & & \vdots & & & & \vdots \\ \vdots & & & & \vdots & & & & \vdots \\ \alpha_{i1} & \cdots & \cdots & \cdots & \alpha_{ij} & \cdots & \cdots & \cdots & \alpha_{i9} \\ \vdots & & & & \vdots & & & & \vdots \\ \vdots & & & & \vdots & & & & \vdots \\ \vdots & & & & \vdots & & & & \vdots \\ \alpha_{91} & \cdots & \cdots & \cdots & \alpha_{9j} & \cdots & \cdots & \cdots & \alpha_{99} \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \vdots \\ \vdots \\ \vdots \\ \beta_i \\ \vdots \\ \vdots \\ \vdots \\ \beta_9 \end{pmatrix}$$

où :

- α_{ij} est le coefficient du terme en x^{n-i} dans le polynôme $P_j(x)$
- β_i est le coefficient du terme en x^{n-i} dans le polynôme $P_0(x)$

Le programme de calcul des inconnues $a, b, c, d, e, f, g, h, i$ s'écrit :

```
funcprot(0) ;
exec("Scilab-Equations.sce",-1)

printf('\n------------------------------------------------------------------------')

function M = POLYNOME(P,m,n)
    M = list()
    for i = 1:length(P)
        Pi = P(i)
        for j = 1:n - length(Pi)
            M($+1) = FRACTION(0,1)
        end
        for j = 1:length(Pi)
            M($+1) = Pi(j)
        end
    end
    M($+1) = m
    M($+1) = n
endfunction

// Nombre n d'inconnues
n = 9 ;
// Polynômes Pk(x)
T0 = list(FRACTION(1,1),FRACTION(0,1)) ;
//
T11 = list(FRACTION(1,1),FRACTION(-1959,1)) ;
T12 = P_MULTIPLICATION(T11,T11) ;
T13 = P_MULTIPLICATION(T11,T12) ;
T14 = P_MULTIPLICATION(T11,T13) ;
T15 = P_MULTIPLICATION(T11,T14) ;
//
T21 = list(FRACTION(1,1),FRACTION(1,1),FRACTION(1,1)) ;
T22 = P_MULTIPLICATION(T21,T21) ;
//
P0 = list(FRACTION(13,1),FRACTION(0,1),FRACTION(-7,1),FRACTION(0,1),...
          FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(2014,1)) ;
P1 = T22 ;
P2 = P_MULTIPLICATION(T11,T22) ;
P3 = P_MULTIPLICATION(T12,T22) ;
P4 = P_MULTIPLICATION(T13,T22) ;
P5 = P_MULTIPLICATION(T14,T22) ;
P6 = P_MULTIPLICATION(T0,T15) ;
P7 = T15 ;
P8 = P_MULTIPLICATION(T0,P_MULTIPLICATION(T15,T21)) ;
P9 = P_MULTIPLICATION(T15,T21) ;
// Matrice A
M = POLYNOME(list(P1,P2,P3,P4,P5,P6,P7,P8,P9),n,n) ;
A = M_TRANSPOSEE(M) ;
// Vecteur b
M = POLYNOME(list(P0),1,n) ;
b = M_TRANSPOSEE(M) ;
// Système linéaire Ax = b
x = GAUSS(A,b) ;
for i = 1:n
    printf('\n x(%i) = ',i), ECRIRE(x(i))
end
```

On trouve :

$$a = \frac{14394094292747965620269 68}{14742843008881}$$

$$b = \frac{84666121980587464374462105 52}{56607224473462851721}$$

$$c = \frac{16600214701315351253213153116357}{21735141998451137744487 2161}$$

$$d = \frac{10849168506553082087424550782065339}{83455142358074924980380638913 4201}$$

$$e = -\frac{65493176091720485412638 77}{32043778625890116302659369677816326618 41}$$

$$f = -\frac{726009833870167779}{83455142358074924980380638913 4201}$$

$$g = -\frac{58456739393216777999}{83455142358074924980380638913 4201}$$

$$h = \frac{65493176091720485412638 77}{32043778625890116302659369677816326618 41}$$

$$i = \frac{51967302682069019614995 54}{32043778625890116302659369677816326618 41}$$

Ajoutons quelques fonctions fréquemment utilisées.

La fonction *P_PGCD* calcule le PGCD de deux polynômes.
Par exemple : $P(x) = (1/3)x^3 + (5/3)x^2 + 7x + 35$, $Q(x) = -x^4 - 5x^3 + (1/2)x + (5/2)$ $\Rightarrow$ $PGCD(P(x),Q(x)) = x + 5$

```
//---  PGCD DE DEUX POLYNOMES

function C = P_PGCD(A,B)
  C = A ; R = B
  while ~P_NUL(R)
    D = C ; C = R
    [Q,R] = P_DIVISION(D,C)
  end
  K = F_INVERSION(C(1))
  C = P_SCALAIRE(C,K)
endfunction
```

NB On convient de prendre la valeur 1 comme coefficient du monôme dominant du PGCD.

La fonction *P_PPCM* calcule le PPCM de deux polynômes.
Par exemple : $P(x) = (1/3)x^3 + (5/3)x^2 + 7x + 35$, $Q(x) = -x^4 - 5x^3 + (1/2)x + (5/2)$
$\Rightarrow$ $PPCM(P(x),Q(x)) = x^6 + 5x^5 + 21x^4 + (209/2)x^3 - (5/2)x^2 - (21/2)x - (105/2)$

```
//---  PPCM DE DEUX POLYNOMES

function C = P_PPCM(A,B)
  P = P_MULTIPLICATION(A,B)
  Q = P_PGCD(A,B)
  [C,R] = P_DIVISION(P,Q)
  K = F_INVERSION(C(1))
  C = P_SCALAIRE(C,K)
endfunction
```

NB Comme le PGCD, le PPCM est un polynôme unitaire.

Plus généralement, la fonction *P_PGCDPPCM* calcule le PGCD et le PPCM d'une liste de polynômes.

```
//---  PGCD ET PPCM DE PLUSIEURS POLYNOMES

function [PGCDL,PPCML] = P_PGCDPPCM(L)
  PGCDL = P_CONSTANT(0,1)
  PPCML = P_CONSTANT(1,1)
  for i = 1:length(L)
    PGCDL = P_PGCD(PGCDL,L(i))
    PPCML = P_PPCM(PPCML,L(i))
  end
endfunction
```

La fonction *P_EQUIVALENT* renvoie le polynôme B(x) équivalent au polynôme A(x) mais avec tous ses coefficients possédant le même dénominateur égal au ppcm des dénominateurs des coefficients du polynôme A(x).
Par exemple : $A(x) = (7/2)x^3 - (1/3)x + (2/15) \quad \Leftrightarrow \quad B(x) = (105/30)x^3 - (10/30)x + (4/30)$

```
//---  POLYNOME EQUIVALENT

function B = P_EQUIVALENT(A)
  L = list()
  for i = 1:length(A)
    L($+1) = DENOMINATEUR(A(i))
  end
  [pgcdl,ppcml] = PGCDPPCM(L)
  B = list()
  for i = 1:length(A)
    [a,r] = DIVISION(ppcml,DENOMINATEUR(A(i)))
    b = MULTIPLICATION(a,NUMERATEUR(A(i)))
    B($+1) = list(b,ppcml)
  end
endfunction
```

La fonction *P_COMPARAISON* compare terme à terme deux polynômes.
Par exemple : $A(x) = x^2 - 2 \quad \equiv \quad B(x) = (3/3)x^2 - (10/5) \quad \Rightarrow \quad s = 0$

```
//---  COMPARAISON DE DEUX POLYNOMES

function s = P_COMPARAISON(A,B)
  degA = P_DEGRE(A)
  degB = P_DEGRE(B)
  if degA == [] | degB == [] then, s = [], return, end
  if degA > degB then, s = +1, return, end
  if degA < degB then, s = -1, return, end
  C = P_SOUSTRACTION(A,B)
  s = F_SIGNE(C(1))
endfunction
```

```
//---   TEST POLYNOME NUL

function t = P_NUL(A)
  if length(A) == 0 then, t = %f, return, end
  t = P_COMPARAISON(A,P_CONSTANT(0,1)) == 0
endfunction


//---   DEGRE D'UN POLYNOME

function d = P_DEGRE(A)
  d = []
  A = P_SIMPLIFICATION(A)
  if length(A) == 0 then, return, end
  d = length(A) - 1
endfunction


//---   VALUATION D'UN POLYNOME

function v = P_VALUATION(A)
  v = []
  for i = length(A):-1:1
      if ~F_NUL(A(i)) then
          v = length(A) - i
          break
      end
  end
endfunction


//---   POLYNOME CONSTANT

function A = P_CONSTANT(a,b)
  A = list(FRACTION(a,b))
endfunction


//---   MONOME

function A = P_MONOME(a,n)
  A = list(a)
  for i = 1:n
    A($+1) = FRACTION(0,1)
  end
endfunction


//---   POLYNOME OPPOSE

function B = P_MOINS(A)
  B = list()
  for i = 1:length(A)
    B($+1) = F_MOINS(A(i))
  end
  B = P_SIMPLIFICATION(B)
endfunction
```

```
//---   POLYNOME VALEUR ABSOLUE

function B = P_ABSOLU(A)
  B = list()
  for i = 1:length(A)
    B($+1) = F_ABSOLU(A(i))
  end
  B = P_SIMPLIFICATION(B)
endfunction


//---   PRODUIT D'UN POLYNOME PAR UN SCALAIRE

function B = P_SCALAIRE(A,K)
  B = list()
  for i = 1:length(A)
    B($+1) = F_MULTIPLICATION(A(i),K)
  end
  B = P_SIMPLIFICATION(B)
endfunction


//---   POLYNOME NORMALISE

function B = P_NORMALISE(A)
  B = P_SIMPLIFICATION(A)
  if length(B) == 0 then, return, end
  if F_NUL(B(1)) then, return, end
  B = P_SCALAIRE(B,F_INVERSION(B(1)))
endfunction


//---   PUISSANCE D'UN POLYNOME

function B = P_PUISSANCE(A,k)
  B = list()
  if length(A) == 0 then, return, end
  B = P_CONSTANT(1,1)
  while k <> 0
    if modulo(k,2) <> 0 then
      B = P_MULTIPLICATION(B,A)
    end
    k = floor(k/2)
    if k <> 0 then
      A = P_MULTIPLICATION(A,A)
    end
  end
endfunction
```

Schéma de Horner

$$y = P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0 = (\ldots (a_n x + a_{n-1})x + \ldots a_1)x + a_0$$

Cette formule permet de calculer la valeur y que prend le polynôme P au point x, sans qu'il soit nécessaire de calculer les puissances de x. La fonction Scilab correspondante est :

```
function y = P_HORNER(P,x)
  n = length(P)
  if n == 0 then, y = [], return, end
  y = FRACTION(0,1)
  for i = 1:n
    z = F_MULTIPLICATION(y,x)
    y = F_ADDITION(z,P(i))
  end
endfunction
```

Exemple : quelle est la valeur du polynôme de **Laguerre** $L_4(x)$ en `x = 113795 / 53`

```
L4 = P_LAGUERRE(4) ;
x = FRACTION(113795,53) ;
y = P_HORNER(L4,x) ;
ECRIRE(y)
```

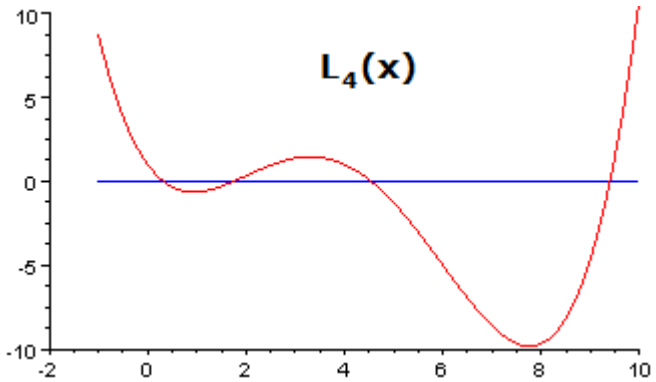
On obtient : `y = 55479152153405887243 / 63123848`

Graphique

La fonction *P_GRAPHIQUE(P,a,b)* fournit la courbe représentative du polynôme P(x) pour $x \in [a,b]$.

```
function P_GRAPHIQUE(P,a,b)
  nbpoints = 300
  x = linspace(a,b,nbpoints)
  y = zeros(1,nbpoints)
  for i = 1:length(P)
    y = y.*x + nombre_rationnel(P(i))
  end
  plot2d([a,b],[0,0],2)
  plot2d(x,y,5)
endfunction
```

Exemple : l'instruction *P_GRAPHIQUE(L4,-1,10)* trace la courbe de $L_4(x)$ pour x variant dans l'intervale [-1,10]



Mathématique & Informatique

Dérivée, Primitive, Intégrale d'un polynôme :

```
//---  DERIVEE D'UN POLYNOME

function B = P_DERIVEE(A)
  B = list()
  n = length(A)
  if n == 0 then, return, end
  if n == 1 then, B = P_CONSTANT(0,1), return, end
  for i = 1:n-1
    B($+1) = F_MULTIPLICATION(A(i),FRACTION(n-i,1))
  end
  B = P_SIMPLIFICATION(B)
endfunction


//---  PRIMITIVE D'UN POLYNOME

function B = P_PRIMITIVE(A)
  B = list()
  n = length(A)
  if n == 0 then, return, end
  for i = 1:n
    B($+1) = F_MULTIPLICATION(A(i),FRACTION(1,n-i+1))
  end
  B($+1) = FRACTION(0,1)
  B = P_SIMPLIFICATION(B)
endfunction


//---  INTEGRALE D'UN POLYNOME

function I = P_INTEGRALE(P,a,b)
  Q = P_PRIMITIVE(P)
  A = P_HORNER(Q,a)
  B = P_HORNER(Q,b)
  I = F_SOUSTRACTION(B,A)
endfunction
```

Exemple

Calculer : $$I = \int_{-\frac{157}{2}}^{+\frac{3226}{297}} \left(38\,x^5 + \frac{7}{4}x^3 - \frac{2}{15}x^2 - x + \frac{1}{21} \right) dx$$

```
P = list(FRACTION(38,1),FRACTION(0,1),FRACTION(7,4),FRACTION(-2,15),FRACTION(-1,1),FRACTION(1,21)) ;
I = P_INTEGRALE(P,FRACTION(-157,2),FRACTION(3226,297)) ;
ECRIRE(I)
```

$$I = -\frac{39059081924812578005692669014 13}{2635541871027183360}$$

EXEMPLE

Soit $P_{10}(x)$ le polynôme de **Legendre** d'ordre 10 :

$$P_{10}(x) = \frac{1}{256}\left(46189x^{10} - 109395x^8 + 90090x^6 - 30030x^4 + 3465x^2 - 63\right)$$

On considère l'intégrale définie :

$$I = \int_{-1}^{+1} P_{10}(x)^2\,dx$$

Calculons I de 2 manières différentes :

1) en utilisant la fonction standard *integrate* de Scilab

2) en utilisant la fonction *P_INTEGRALE* précédente

Avec la fonction *integrate* :

```
P2 = '((46189*x^10 - 109395*x^8 + 90090*x^6 - 30030*x^4 + 3465*x^2 - 63)/256)^2' ;
I = integrate(P2,'x',-1,+1) ;
printf('\n I = %.16f',I)
```

On obtient la valeur approchée :

$$I = 0.0952380952380947$$

Avec la fonction *P_INTEGRALE* :

```
exec("Scilab-Polynomes.sce",-1)
P = list(FRACTION(46189,256),FRACTION(0,1),FRACTION(-109395,256),FRACTION(0,1),...
         FRACTION(90090,256),FRACTION(0,1),FRACTION(-30030,256),FRACTION(0,1),...
         FRACTION(3465,256),FRACTION(0,1),FRACTION(-63,256)) ;
P2 = P_MULTIPLICATION(P,P) ;
I = P_INTEGRALE(P2,FRACTION(-1,1),FRACTION(1,1)) ;
printf('\n I = '), ECRIRE(I)
```

On obtient la valeur exacte :

$$I = \frac{2}{21}$$

Remarque

Si $P_n(x)$ est le polynôme de **Legendre** d'ordre *n*, on établit que : $\int_{-1}^{+1} P_n(x)^2\,dx = \frac{2}{2n+1}$

Algorithme d'Euclide étendu

L'équation de Bachet-Bézout AX + BY = C peut être résolue dans tout anneau principal, donc dans l'anneau des polynômes ℚ(x) en particulier.

```
//---  EUCLIDE :  A U + B V = PGCD(A,B)

function [U,V,P] = P_EUCLIDE(A,B)
  R0 = A ; R1 = B
  U0 = P_CONSTANT(1,1) ; U1 = P_CONSTANT(0,1)
  V0 = P_CONSTANT(0,1) ; V1 = P_CONSTANT(1,1)
  while ~P_NUL(R1)
    [Q,R] = P_DIVISION(R0,R1)
    T = P_MULTIPLICATION(Q,R1)
    R2 = P_SOUSTRACTION(R0,T)
    T = P_MULTIPLICATION(Q,U1)
    U2 = P_SOUSTRACTION(U0,T)
    T = P_MULTIPLICATION(Q,V1)
    V2 = P_SOUSTRACTION(V0,T)
    R0 = R1 ; R1 = R2
    U0 = U1 ; U1 = U2
    V0 = V1 ; V1 = V2
  end
  K = F_INVERSION(R0(1))
  U = P_SCALAIRE(U0,K)
  V = P_SCALAIRE(V0,K)
  P = P_SCALAIRE(R0,K)
endfunction

//---  BACHET :  A X + B Y = C

function [X,Y] = P_BACHET(A,B,C)
  [U,V,P] = P_EUCLIDE(A,B)
  [Q,R] = P_DIVISION(C,P)
  if P_NUL(R) then
    X = P_MULTIPLICATION(U,Q)
    Y = P_MULTIPLICATION(V,Q)
  else
    X = list()
    Y = list()
  end
endfunction
```

Un exemple simple : $A(x) = x^4 - 3x^2 + 2$, $B(x) = x^3 - 3x^2 - 2x + 6$, $C(x) = x^3 + x^2 - 2x - 2$
Trouver P(x) et Q(x) tels que : AP + BQ = C

Le programme s'écrit :

```
A = list(FRACTION(1,1),FRACTION(0,1),FRACTION(-3,1),FRACTION(0,1),FRACTION(2,1)) ;
B = list(FRACTION(1,1),FRACTION(-3,1),FRACTION(-2,1),FRACTION(6,1)) ;
C = list(FRACTION(1,1),FRACTION(1,1),FRACTION(-2,1),FRACTION(-2,1)) ;
[P,Q] = P_BACHET(A,B,C) ;
P_ECRIRE(P), P_ECRIRE(Q)
```

On trouve : $P(x) = (1/8)x + (1/8)$, $Q(x) = (-1/8)x^2 - (1/2)x - (3/8)$

EXEMPLE

Résoudre l'équation de Bachet-Bézout A(x)U(x) + B(x)V(x) = PGCD(A(x),B(x)) avec les 2 polynômes suivants :

$A(x) = 11x^3 + 55/24x^2 - 25/137x - 588$
$B(x) = 3x^3 - 1/33x^2 + 402x + 13/14$

Le programme :

```
exec("Scilab-Polynomes.sce",-1)

A = list(list([1,1,1],[1,1]),list([1,5,5],[1,2,4]),list([-1,2,5],[1,1,3,7]),list([-1,5,8,8],[1,1])) ;
B = list(list([1,3],[1,1]),list([-1,1],[1,3,3]),list([1,4,0,2],[1,1]),list([1,1,3],[1,1,4])) ;
[U,V,P] = P_EUCLIDE(A,B) ;

P_ECRIRE(P)
P_ECRIRE(U)
P_ECRIRE(V)
```

La solution :

PGCD(A(x),B(x)) = 1

U(x) = - 346603047465602743687872 / 27361050998228450590929134861 **x²**
+ 142409430200712749530288 / 27361050998228450590929134861 **x**
- 46532103172934520080525264 / 27361050998228450590929134861

V(x) = 1270877840707210060188864 / 27361050998228450590929134861 **x²**
- 244564544015811217453640 / 27361050998228450590929134861 **x**
+ 18774273856453616337806 2 / 27361050998228450590929134861

Remarques

- Il existe une infinité de polynômes R(x), S(x) solutions.
  Soit K(x) un polynôme non nul quelconque et :

  R(x) = U(x) + K(x)B(x)
  S(x) = V(x) – K(x)A(x)

  On a : A(x)R(x) + B(x)S(x) = A(x)U(x) + B(x)V(x) = PGCD(A(x),B(x))
  Les polynômes R(x) et S(x) sont de degré supérieur à U(x) et V(x).

- Lorsque le pgcd de 2 polynômes A(x) et B(x) à coefficients rationnels vaut 1, ils sont premiers entre eux, et par conséquent ils n'ont aucune racine commune dans $\mathbb{Q}$.

Polynômes de Wilkinson

Le polynôme de **Wilkinson** de degré *n* est le polynôme unitaire qui admet comme racines les *n* premiers entiers naturels non nuls :

$$Wn(x) = \prod_{i=1}^{n} (x - i)$$

Pour obtenir $W_n(x)$, on écrit :

```
function W = P_WILKINSON(n)
  W = P_CONSTANT(1,1)
  for i = 1:n
    W = P_MULTIPLICATION(W,list(FRACTION(1,1),FRACTION(-i,1)))
  end
endfunction
```

Les 10 premiers polynômes de **Wilkinson** sont :

$W_1(x) = x - 1$
$W_2(x) = x^2 - 3x + 2$
$W_3(x) = x^3 - 6x^2 + 11x - 6$
$W_4(x) = x^4 - 10x^3 + 35x^2 - 50x + 24$
$W_5(x) = x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120$
$W_6(x) = x^6 - 21x^5 + 175x^4 - 735x^3 + 1624x^2 - 1764x + 720$
$W_7(x) = x^7 - 28x^6 + 322x^5 - 1960x^4 + 6769x^3 - 13132x^2 + 13068x - 5040$
$W_8(x) = x^8 - 36x^7 + 546x^6 - 4536x^5 + 22449x^4 - 67284x^3 + 118124x^2 - 109584x + 40320$
$W_9(x) = x^9 - 45x^8 + 870x^7 - 9450x^6 + 63273x^5 - 269325x^4 + 723680x^3 - 1172700x^2 + 1026576x - 362880$
$W_{10}(x) = x^{10} - 55x^9 + 1320x^8 - 18150x^7 + 157773x^6 - 902055x^5 + 3416930x^4 - 8409500x^3 + 12753576x^2 - 10628640x + 3628800$

Par exemple, *P_ECRIRE(P_WILKINSON(25))* donne $W_{25}(x)$:

1 x^{25}
- 325 x^{24}
50050 x^{23}
- 4858750 x^{22}
333685495 x^{21}
- 17247104875 x^{20}
696829576300 x^{19}
- 22563937825000 x^{18}
595667304367135 x^{17}
- 12972753318542875 x^{16}
234961569422786050 x^{15}
- 3557372853474553750 x^{14}
45145946926994481865 x^{13}
- 480544558742733545125 x^{12}
4284218746244111474800 x^{11}
- 3188201437529851278250 x^{10}
19692810045111082024288 x^9
- 10013693045128413741100000 x^8
41444578032471158770368000 x^7
- 13746468217967926978680000 x^6
35770355645907606826362624 x^5
- 7087414531983767267719680 x^4
10233953060174467567257600 x^3
- 10048017154835116154880000 x^2
5919012881170120359936000 x
- 155112100433309859840000000

Mathématique & Informatique

Polynômes orthogonaux

Commençons par une fonction *P_LEGENDRE* qui calcule le polynôme de **Legendre** $P_n(x)$ de degré n.

Rappel

$$P_n(x) = \frac{1}{2^n\, n!}\frac{d^n}{dx^n}(x^2-1)^n$$

$P_n(x)$ est solution de l'équation différentielle : $(1-x^2)y'' - 2xy' + n(n+1)y = 0$
Orthogonalité pour le produit scalaire : $\langle P_p,P_q\rangle = \int_{[-1,+1]}P_p(x)P_q(x)dx = 0 \quad \forall\, p \neq q$
Racines : $P_n(x)$ possède n racines réelles distinctes dans l'intervalle $[-1,+1]$
Formule de récurrence : $nP_n(x) - (2n-1)xP_{n-1}(x) + (n-1)P_{n-2}(x) = 0$
Valeurs particulières : $P_0(x) = 1$; $P_1(x) = x$; $P_2(x) = (1/2)(3x^2 - 1)$; $P_3(x) = (1/2)(5x^3 - 3x)$

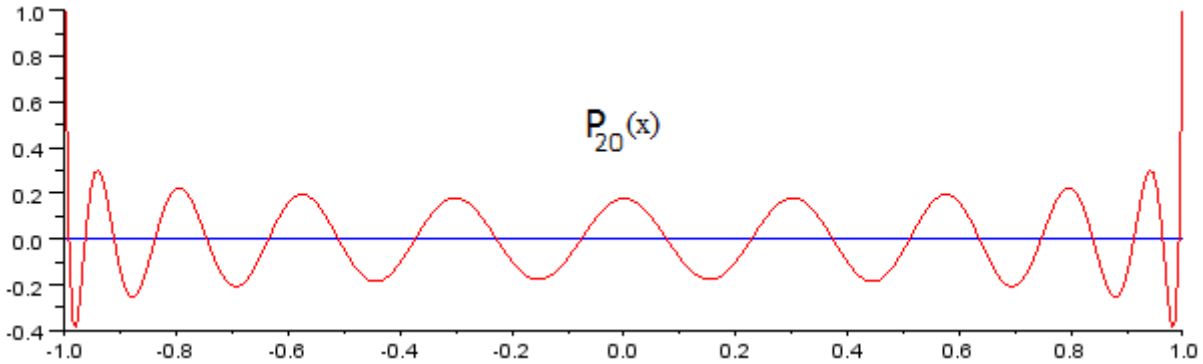
Fonction

```
function P = P_LEGENDRE(n)
  P0 = list(FRACTION(1,1))
  P1 = list(FRACTION(1,1),FRACTION(0,1))
  if n == 0 then, P = P0, return, end
  if n == 1 then, P = P1, return, end
  for i = 2:n
    A = P_MULTIPLICATION(list(FRACTION(2*i-1,1),FRACTION(0,1)),P1)
    B = P_MULTIPLICATION(list(FRACTION(-i+1,1)),P0)
    P = P_SCALAIRE(P_ADDITION(A,B),FRACTION(1,i))
    P0 = P1
    P1 = P
  end
  P = P_EQUIVALENT(P)
endfunction
```

Exemple

P20 = P_LEGENDRE(20) ; P_ECRIRE(P20) ; P_GRAPHIQUE(P20,-1,+1) donne le polynôme de **Legendre** $P_{20}(x)$

$$P_{20}(x) = \frac{1}{262144}(34461632205\,x^{20} - 167890003050\,x^{18} + 347123925225\,x^{16} - 396713057400\,x^{14} + 273491577450\,x^{12} - 116454478140\,x^{10} + 30117537450\,x^8 - 4461857400\,x^6 + 334639305\,x^4 - 9699690\,x^2 + 46189)$$



On écrit de même des fonctions *P_LAGUERRE*, *P_HERMITE*, *P_TCHEBYCHEV* pour obtenir les polynômes orthogonaux de **Laguerre** $L_n(x)$, d'**Hermite** $H_n(x)$, de **Tchebychev** $T_n(x)$ définis par les relations de récurrence :

- $nL_n(x) + (x-2n+1)L_{n-1}(x) + (n-1)L_{n-2}(x) = 0$; $L_0(x) = 1$; $L_1(x) = -x + 1$
- $H_n(x) - 2xH_{n-1}(x) + 2(n-1)H_{n-2}(x) = 0$; $H_0(x) = 1$; $H_1(x) = 2x$
- $T_n(x) - 2xT_{n-1}(x) + T_{n-2}(x) = 0$; $T_0(x) = 1$; $T_1(x) = x$

```
function L = P_LAGUERRE(n)
  L0 = list(FRACTION(1,1))
  L1 = list(FRACTION(-1,1),FRACTION(1,1))
  if n == 0 then, L = L0, return, end
  if n == 1 then, L = L1, return, end
  for i = 2:n
    A = P_MULTIPLICATION(list(FRACTION(-1,1),FRACTION(2*i-1,1)),L1)
    B = P_MULTIPLICATION(list(FRACTION(-i+1,1)),L0)
    L = P_SCALAIRE(P_ADDITION(A,B),FRACTION(1,i))
    L0 = L1
    L1 = L
  end
  L = P_EQUIVALENT(L)
endfunction
```

```
function H = P_HERMITE(n)
  H0 = list(FRACTION(1,1))
  H1 = list(FRACTION(2,1),FRACTION(0,1))
  if n == 0 then, H = H0, return, end
  if n == 1 then, H = H1, return, end
  for i = 2:n
    A = P_MULTIPLICATION(list(FRACTION(2,1),FRACTION(0,1)),H1)
    B = P_MULTIPLICATION(list(FRACTION(-2*i+2,1)),H0)
    H = P_ADDITION(A,B)
    H0 = H1
    H1 = H
  end
endfunction
```

```
function T = P_TCHEBYCHEV(n)
  T0 = list(FRACTION(1,1))
  T1 = list(FRACTION(1,1),FRACTION(0,1))
  if n == 0 then, T = T0, return, end
  if n == 1 then, T = T1, return, end
  for i = 2:n
    A = P_MULTIPLICATION(list(FRACTION(2,1),FRACTION(0,1)),T1)
    B = P_MULTIPLICATION(list(FRACTION(-1,1)),T0)
    T = P_ADDITION(A,B)
    T0 = T1
    T1 = T
  end
endfunction
```

Mathématique & Informatique

Polynômes de **Legendre**

$P0(x) = 1$

$P1(x) = \mathbf{x}$

$P2(x) = 1/2(3\mathbf{x}^2 - 1)$

$P3(x) = 1/2(5\mathbf{x}^3 - 3\mathbf{x})$

$P4(x) = 1/8(35\mathbf{x}^4 - 30\mathbf{x}^2 + 3)$

$P5(x) = 1/8(63\mathbf{x}^5 - 70\mathbf{x}^3 + 15\mathbf{x})$

$P6(x) = 1/16(231\mathbf{x}^6 - 315\mathbf{x}^4 + 105\mathbf{x}^2 - 5)$

$P7(x) = 1/16(429\mathbf{x}^7 - 693\mathbf{x}^5 + 315\mathbf{x}^3 - 35\mathbf{x})$

$P8(x) = 1/128(6435\mathbf{x}^8 - 12012\mathbf{x}^6 + 6930\mathbf{x}^4 - 1260\mathbf{x}^2 + 35)$

$P9(x) = 1/128(12155\mathbf{x}^9 - 25740\mathbf{x}^7 + 18018\mathbf{x}^5 - 4620\mathbf{x}^3 + 315\mathbf{x})$

$P10(x) = 1/256(46189\mathbf{x}^{10} - 109395\mathbf{x}^8 + 90090\mathbf{x}^6 - 30030\mathbf{x}^4 + 3465\mathbf{x}^2 - 63)$

Polynômes de **Laguerre**

$L0(x) = 1$

$L1(x) = -\mathbf{x} + 1$

$L2(x) = 1/2(\mathbf{x}^2 - 4\mathbf{x} + 2)$

$L3(x) = 1/6(-\mathbf{x}^3 + 9\mathbf{x}^2 - 18\mathbf{x} + 6)$

$L4(x) = 1/24(\mathbf{x}^4 - 16\mathbf{x}^3 + 72\mathbf{x}^2 - 96\mathbf{x} + 24)$

$L5(x) = 1/120(-\mathbf{x}^5 + 25\mathbf{x}^4 - 200\mathbf{x}^3 + 600\mathbf{x}^2 - 600\mathbf{x} + 120)$

$L6(x) = 1/720(\mathbf{x}^6 - 36\mathbf{x}^5 + 450\mathbf{x}^4 - 2400\mathbf{x}^3 + 5400\mathbf{x}^2 - 4320\mathbf{x} + 720)$

$L7(x) = 1/5040(-\mathbf{x}^7 + 49\mathbf{x}^6 - 882\mathbf{x}^5 + 7350\mathbf{x}^4 - 29400\mathbf{x}^3 + 52920\mathbf{x}^2 - 35280\mathbf{x} + 5040)$

$L8(x) = 1/40320(\mathbf{x}^8 - 64\mathbf{x}^7 + 1568\mathbf{x}^6 - 18816\mathbf{x}^5 + 117600\mathbf{x}^4 - 376320\mathbf{x}^3 + 564480\mathbf{x}^2 - 322560\mathbf{x} + 40320)$

$L9(x) = 1/362880(-\mathbf{x}^9 + 81\mathbf{x}^8 - 2592\mathbf{x}^7 + 42336\mathbf{x}^6 - 381024\mathbf{x}^5 + 1905120\mathbf{x}^4 - 5080320\mathbf{x}^3 + 6531840\mathbf{x}^2 - 3265920\mathbf{x} + 362880)$

$L10(x) = 1/3628800(\mathbf{x}^{10} - 100\mathbf{x}^9 + 4050\mathbf{x}^8 - 86400\mathbf{x}^7 + 1058400\mathbf{x}^6 - 7620480\mathbf{x}^5 + 31752000\mathbf{x}^4 - 72576000\mathbf{x}^3 + 81648000\mathbf{x}^2 - 36288000\mathbf{x} + 3628800)$

Mathématique & Informatique

Polynômes d'**Hermite**

$H0(x) = 1$

$H1(x) = 2\mathbf{x}$

$H2(x) = 4\mathbf{x^2} - 2$

$H3(x) = 8\mathbf{x^3} - 12\mathbf{x}$

$H4(x) = 16\mathbf{x^4} - 48\mathbf{x^2} + 12$

$H5(x) = 32\mathbf{x^5} - 160\mathbf{x^3} + 120\mathbf{x}$

$H6(x) = 64\mathbf{x^6} - 480\mathbf{x^4} + 720\mathbf{x^2} - 120$

$H7(x) = 128\mathbf{x^7} - 1344\mathbf{x^5} + 3360\mathbf{x^3} - 1680\mathbf{x}$

$H8(x) = 256\mathbf{x^8} - 3584\mathbf{x^6} + 13440\mathbf{x^4} - 13440\mathbf{x^2} + 1680$

$H9(x) = 512\mathbf{x^9} - 9216\mathbf{x^7} + 48384\mathbf{x^5} - 80640\mathbf{x^3} + 30240\mathbf{x}$

$H10(x) = 1024\mathbf{x^{10}} - 23040\mathbf{x^8} + 161280\mathbf{x^6} - 403200\mathbf{x^4} + 302400\mathbf{x^2} - 30240$

Polynômes de **Tchebychev**

$T0(x) = 1$

$T1(x) = \mathbf{x}$

$T2(x) = 2\mathbf{x^2} - 1$

$T3(x) = 4\mathbf{x^3} - 3\mathbf{x}$

$T4(x) = 8\mathbf{x^4} - 8\mathbf{x^2} + 1$

$T5(x) = 16\mathbf{x^5} - 20\mathbf{x^3} + 5\mathbf{x}$

$T6(x) = 32\mathbf{x^6} - 48\mathbf{x^4} + 18\mathbf{x^2} - 1$

$T7(x) = 64\mathbf{x^7} - 112\mathbf{x^5} + 56\mathbf{x^3} - 7\mathbf{x}$

$T8(x) = 128\mathbf{x^8} - 256\mathbf{x^6} + 160\mathbf{x^4} - 32\mathbf{x^2} + 1$

$T9(x) = 256\mathbf{x^9} - 576\mathbf{x^7} + 432\mathbf{x^5} - 120\mathbf{x^3} + 9\mathbf{x}$

$T10(x) = 512\mathbf{x^{10}} - 1280\mathbf{x^8} + 1120\mathbf{x^6} - 400\mathbf{x^4} + 50\mathbf{x^2} - 1$

Mathématique & Informatique

Le polynôme de **Tchebychev** $T_{100}(x)$:
==========

$$633825300114114700748351602688\ x^{100}$$
$$- 158456325028528675187087900672 00\ x^{98}$$
$$192128294097091018664344079564800\ x^{96}$$
$$- 150533508777102241427733505638 4000\ x^{94}$$
$$856747352688429553750811397324 8000\ x^{92}$$
$$- 377509938774080643368515422021 22240\ x^{90}$$
$$134036108580690866727917044786790400\ x^{88}$$
$$- 394078512785625681900511864396185600\ x^{86}$$
$$978503372439851812055958467641344000\ x^{84}$$
$$- 208245589519250513847806545677516 8000\ x^{82}$$
$$384213112663017198049203076775018 4960\ x^{80}$$
$$- 620078363644093128618734281209937 9200\ x^{78}$$
$$881673923306444917254762806095380 4800\ x^{76}$$
$$- 111086237021369054561276480874086 40000\ x^{74}$$
$$124602959383188461947677647359180 80000\ x^{72}$$
$$- 124896142817031258328731006529437 69600\ x^{70}$$
$$112211378312176521154719263678791 68000\ x^{68}$$
$$- 905802692399497218959782008009523 2000\ x^{66}$$
$$658179801895976129630329406226432 0000\ x^{64}$$
$$- 431088525218417114143841482440704 0000\ x^{62}$$
$$254746375371258363389376326029803 5200\ x^{60}$$
$$- 135895444366222815912958437936332 8000\ x^{58}$$
$$654531379770880870350000868032512000\ x^{56}$$
$$- 284578860769948204500000377405440000\ x^{54}$$
$$111631674825053695350740279623680000\ x^{52}$$
$$- 394729602181389866760217628749332 48\ x^{50}$$
$$125661060985499632739414395846656 00\ x^{48}$$
$$- 359578074052875661415675896791040 0\ x^{46}$$
$$923024074019658505866132324352000\ x^{44}$$
$$- 212040013118649088525828358144000\ x^{42}$$
$$434682026893230631477948134195 20\ x^{40}$$
$$- 792547875120897364546048491520 0\ x^{38}$$
$$128024162732075102774786785280 0\ x^{36}$$
$$- 182395347175955031090266112000\ x^{34}$$
$$227994183969943788862832640 00\ x^{32}$$
$$- 248538714833169492914208768 0\ x^{30}$$
$$234623135747458180159897600\ x^{28}$$
$$- 190234974930371497426944 00\ x^{26}$$
$$131210455968528728064 0000\ x^{24}$$
$$- 761119921128918220800 00\ x^{22}$$
$$3662889620432918937600\ x^{20}$$
$$- 143850563845029888000\ x^{18}$$
$$4517474603507712000\ x^{16}$$
$$- 110586893598720000\ x^{14}$$
$$2042087523840000\ x^{12}$$
$$- 27227833651200\ x^{10}$$
$$246628928000\ x^{8}$$
$$- 1386112000\ x^{6}$$
$$4165000\ x^{4}$$
$$- 5000\ x^{2}$$
$$1$$

Mathématique & Informatique

Polynômes trigonométriques

Le polynôme cosinus $C_n(x)$ est un polynôme à coefficients entiers qui admet (uniquement) les racines réelles suivantes :

$$x_{\pm k} = \pm cos\frac{k\pi}{n} \qquad \left(1 \leq k \leq E\left(\frac{n-1}{2}\right)\right)$$

En particulier, pour la plus grande racine positive : $C_n\left(cos\frac{\pi}{n}\right) = 0$

Le polynôme sinus $S_n(x)$ est un polynôme à coefficients entiers qui admet (uniquement) les racines réelles suivantes :

$$x_{\pm k} = \pm sin\frac{k\pi}{n} \qquad \left(1 \leq k \leq E\left(\frac{n-1}{2}\right)\right)$$

En particulier, pour la plus petite racine positive : $S_n\left(sin\frac{\pi}{n}\right) = 0$

Si l'on se conforme à la définition, les $C_n(x)$ et les $S_n(x)$ n'existent que pour $n \geq 3$ puisque l'inégalité $1 \leq E\left(\frac{n-1}{2}\right)$ n'est pas vérifiée pour n < 3. Par commodité dans les calculs, on prendra le polynôme nul pour $n = 0$, et le polynôme unité pour $n = 1$ ou 2. Les polynômes $C_n(x)$ et $S_n(x)$ non nuls sont de degré $2E\left(\frac{n-1}{2}\right)$. Ajoutons que les polynômes $C_n(x)$ n'ont aucune racine commune, de même pour les polynômes $S_n(x)$.

On a :

- $n = 0$: $C_0(x) = 0$
- $n = 1$: $C_1(x) = 1$
- $n = 2$: $C_2(x) = 1$
- $n \geq 3$: $$C_n(x) = 4^{E\left(\frac{n-1}{2}\right)} \prod_{k=1}^{E\left(\frac{n-1}{2}\right)} \left(x + \cos\frac{k\pi}{n}\right)\left(x - \cos\frac{k\pi}{n}\right)$$

et :

- $n = 0$: $S_0(x) = 0$
- $n = 1$: $S_1(x) = 1$
- $n = 2$: $S_2(x) = 1$
- $n \geq 3$: $$S_n(x) = 4^{E\left(\frac{n-1}{2}\right)} \prod_{k=1}^{E\left(\frac{n-1}{2}\right)} \left(x + \sin\frac{k\pi}{n}\right)\left(x - \sin\frac{k\pi}{n}\right)$$

Note

Les coefficients du polynôme $C_n(x)$ sont des multiples de 4 des nombres situés sur la n^e diagonale du triangle de **Pascal**. Les polynômes $C_n(x)$ et $S_n(x)$ s'apparentent aux polynômes de **Tchebychev**.

Mathématique & Informatique

On établit que $cos\frac{k\pi}{n}$ $\left(n \geq 3,\ 1 \leq k \leq E\left(\frac{n-1}{2}\right)\right)$ est solution de l'équation polynomiale :

$$C_n(x) = \sum_{k=0}^{E\left(\frac{n-1}{2}\right)} c_{2k}x^{2k} = 0 \qquad \text{avec} \qquad c_{2k} = (-1)^{E\left(\frac{n-1}{2}\right)-k}\, 4^k\, C_{n-1-E\left(\frac{n-1}{2}\right)+k}^{E\left(\frac{n-1}{2}\right)-k}$$

De même, on établit que $sin\frac{k\pi}{n}$ $\left(n \geq 3,\ 1 \leq k \leq E\left(\frac{n-1}{2}\right)\right)$ est solution de l'équation polynomiale :

$$S_n(x) = \sum_{k=0}^{E\left(\frac{n-1}{2}\right)} s_{2k}x^{2k} = 0 \qquad \text{avec} \qquad s_{2k} = \left[1 + (n \bmod 2)\left(\frac{n}{2k+1} - 1\right)\right] c_{2k}$$

Les formules se simplifient en étudiant la parité de n :

Si n est pair : $\quad c_{2k} = (-1)^{\frac{n-2}{2}-k}\, 4^k\, C_{\frac{n}{2}+k}^{2k+1}$ et $s_{2k} = c_{2k}$ $\quad \left(k = 0, \ldots, \frac{n-2}{2}\right)$

Si n est impair : $\quad c_{2k} = (-1)^{\frac{n-1}{2}-k}\, 4^k\, C_{\frac{n-1}{2}+k}^{2k}$ et $s_{2k} = \frac{n}{2k+1}c_{2k}$ $\quad \left(k = 0, \ldots, \frac{n-1}{2}\right)$

Propriété

En exprimant les valeurs de $C_n(0)$ et de $S_n(0)$, on obtient les relations :

$$\prod_{k=1}^{E\left(\frac{n-1}{2}\right)} cos\frac{k\pi}{n} = \begin{cases} \dfrac{\sqrt{\frac{n}{2}}}{2^{\frac{n-2}{2}}} & \text{si } n \text{ est pair} \\ \dfrac{1}{2^{\frac{n-1}{2}}} & \text{si } n \text{ est impair} \end{cases} \quad \text{et} \quad \prod_{k=1}^{E\left(\frac{n-1}{2}\right)} sin\frac{k\pi}{n} = \begin{cases} \dfrac{\sqrt{\frac{n}{2}}}{2^{\frac{n-2}{2}}} & \text{si } n \text{ est pair} \\ \dfrac{\sqrt{n}}{2^{\frac{n-1}{2}}} & \text{si } n \text{ est impair} \end{cases}$$

Mathématique & Informatique

Il est facile d'écrire une fonction Scilab *P_COSINUS* et une fonction Scilab *P_SINUS* qui calculent le polynôme $C_n(x)$ et le polynôme $S_n(x)$ lorsqu'on fournit le paramètre n :

```
function C = P_COSINUS(n)
    if n == 0 then, C = list(FRACTION(0,1)), return, end
    C = list()
    E = floor((n-1)/2)
    for k = 0:E
        u = (-1)^(E-k)
        v = puissance(4,k)
        w = binomiaux(n-1-E+k , E-k)
        C(0) = list([u,multiplication(v,w)],[1,1])
        if k <> E then, C(0) = FRACTION(0,1), end
    end
endfunction
```

```
function S = P_SINUS(n)
    if n == 0 then, S = list(FRACTION(0,1)), return, end
    S = list()
    E = floor((n-1)/2)
    for k = 0:E
        u = (-1)^(E-k)
        v = puissance(4,k)
        w = binomiaux(n-1-E+k , E-k)
        F1 = list([u,multiplication(v,w)],[1,1])
        F2 = FRACTION(2*k+1+modulo(n,2)*(n-2*k-1),2*k+1)
        S(0) = F_MULTIPLICATION(F1,F2)
        if k <> E then, S(0) = FRACTION(0,1), end
    end
endfunction
```

Les premiers polynômes cosinus sont :

$C_0(x) = 0$
$C_1(x) = 1$
$C_2(x) = 1$
$C_3(x) = 4x^2 - 1$
$C_4(x) = 4x^2 - 2$
$C_5(x) = 16x^4 - 12x^2 + 1$
$C_6(x) = 16x^4 - 16x^2 + 3$
$C_7(x) = 64x^6 - 80x^4 + 24x^2 - 1$
$C_8(x) = 64x^6 - 96x^4 + 40x^2 - 4$
$C_9(x) = 256x^8 - 448x^6 + 240x^4 - 40x^2 + 1$
$C_{10}(x) = 256x^8 - 512x^6 + 336x^4 - 80x^2 + 5$
$C_{11}(x) = 1024x^{10} - 2304x^8 + 1792x^6 - 560x^4 + 60x^2 - 1$
$C_{12}(x) = 1024x^{10} - 2560x^8 + 2304x^6 - 896x^4 + 140x^2 - 6$

Les premiers polynômes sinus sont :

$S_0(x) = 0$
$S_1(x) = 1$
$S_2(x) = 1$
$S_3(x) = 4x^2 - 3$
$S_4(x) = 4x^2 - 2$
$S_5(x) = 16x^4 - 20x^2 + 5$
$S_6(x) = 16x^4 - 16x^2 + 3$
$S_7(x) = 64x^6 - 112x^4 + 56x^2 - 7$
$S_8(x) = 64x^6 - 96x^4 + 40x^2 - 4$
$S_9(x) = 256x^8 - 576x^6 + 432x^4 - 120x^2 + 9$
$S_{10}(x) = 256x^8 - 512x^6 + 336x^4 - 80x^2 + 5$
$S_{11}(x) = 1024x^{10} - 2816x^8 + 2816x^6 - 1232x^4 + 220x^2 - 11$
$S_{12}(x) = 1024x^{10} - 2560x^8 + 2304x^6 - 896x^4 + 140x^2 - 6$

L'instruction *P_ECRIRE(P_COSINUS(51))* affiche $C_{51}(x)$:

1125899906842624 x^{50}
- 13792273858822144 x^{48}
79375943432404992 x^{46}
- 285257296710205440 x^{44}
717695219914506240 x^{42}
- 1343338226839977984 x^{40}
1940377438768857088 x^{38}
- 2214424252361211904 x^{36}
2027743138063319040 x^{34}
- 1504714154039050240 x^{32}
910168561467523072 x^{30}
- 449912868452696064 x^{28}
181695581490511872 x^{26}
- 59768283385036800 x^{24}
15922824531148800 x^{22}
- 3405715246940160 x^{20}
577755265105920 x^{18}
- 76467608616960 x^{16}
7724000870400 x^{14}
- 578029670400 x^{12}
30766095360 x^{10}
- 1098789120 x^{8}
24111360 x^{6}
- 280800 x^{4}
1300 x^{2}
- 1

L'instruction *P_ECRIRE(P_SINUS(51))* affiche $S_{51}(x)$:

1125899906842624 x^{50}
- 14355223812243456 x^{48}
86131342873460736 x^{46}
- 323291602938232832 x^{44}
851219911991623680 x^{42}
- 1670981696800948224 x^{40}
2537416650697736192 x^{38}
- 3052314510011400192 x^{36}
2954711429749407744 x^{34}
- 2325467328969441280 x^{32}
1497374084994957312 x^{30}
- 791226079003017216 x^{28}
343202765037633536 x^{26}
- 121927298105475072 x^{24}
35307132656025600 x^{22}
- 8271022742568960 x^{20}
1550816764231680 x^{18}
- 229402825850880 x^{16}
26261602959360 x^{14}
- 2267654860800 x^{12}
142642805760 x^{10}
- 6226471680 x^{8}
175668480 x^{6}
- 2864160 x^{4}
22100 x^{2}
- 51

Valeurs particulières des $C_n(x)$ et des $S_n(x)$

A partir de l'expression générale de $C_n(x)$ et de $S_n(x)$, on obtient les valeurs de ces polynômes en $x = 0$, $x = -1$, $x = +1$

- pour n pair :

  $C_n(0) = (-1)^{\frac{n-2}{2}} \frac{n}{2}$; $C_n(\pm 1) = \frac{n}{2}$

  $S_n(0) = (-1)^{\frac{n-2}{2}} \frac{n}{2}$; $S_n(\pm 1) = \frac{n}{2}$

- pour n impair :

  $C_n(0) = (-1)^{\frac{n-1}{2}}$; $C_n(\pm 1) = n$

  $S_n(0) = (-1)^{\frac{n-2}{2}} n$; $S_n(\pm 1) = 1$

Formules de récurrence des $C_n(x)$ et des $S_n(x)$

On recherche des relations du type :

$$C_{n+2}(x) + \alpha C_{n+1}(x) + \beta C_n(x) = 0 \quad \text{et} \quad S_{n+2}(x) + \gamma S_{n+1}(x) + \delta S_n(x) = 0$$

où α, β, γ, δ sont des polynômes de degré 2 en x.
A partir des formules donnant les coefficients c_{2k} de $C_n(x)$ et s_{2k} de $S_n(x)$, on établit :

- pour n pair :

  $C_{n+2}(x) - C_{n+1}(x) + C_n(x) = 0$

  $S_{n+2}(x) - S_{n+1}(x) - S_n(x) = 0$

- pour n impair :

  $C_{n+2}(x) - 4x^2 C_{n+1}(x) + C_n(x) = 0$

  $S_{n+2}(x) - 4(x^2 - 1)S_{n+1}(x) - S_n(x) = 0$

D'où :

$$C_{n+2}(x) - [1 + (n \bmod 2)(4x^2 - 1)]C_{n+1}(x) + C_n(x) = 0 \qquad (\forall\, n \geq 0)$$
$$C_1(x) = 1$$
$$C_0(x) = 0$$

Et :

$$S_{n+2}(x) - [1 + (n \bmod 2)(4x^2 - 5)]S_{n+1}(x) - S_n(x) = 0 \qquad (\forall\, n \geq 0)$$
$$S_1(x) = 1$$
$$S_0(x) = 0$$

Les fonctions Scilab de calcul par récurrence des $C_n(x)$ et des $S_n(x)$ sont les suivantes :

```
function C = P_COSINUS(n)
    if n < 2 then, C = list(FRACTION(n,1)), return, end
    C0 = list(FRACTION(0,1))
    C1 = list(FRACTION(1,1))
    C2 = list(FRACTION(4,1),FRACTION(0,1),FRACTION(0,1))
    for k = 0:n-2
        if modulo(k,2) == 0 then
            C = P_SOUSTRACTION(C1,C0)
        else
            C = P_SOUSTRACTION(P_MULTIPLICATION(C2,C1),C0)
        end
        C0 = C1
        C1 = C
    end
endfunction
```

```
function S = P_SINUS(n)
    if n < 2 then, S = list(FRACTION(n,1)), return, end
    S0 = list(FRACTION(0,1))
    S1 = list(FRACTION(1,1))
    S2 = list(FRACTION(4,1),FRACTION(0,1),FRACTION(-4,1))
    for k = 0:n-2
        if modulo(k,2) == 0 then
            S = P_ADDITION(S1,S0)
        else
            S = P_ADDITION(P_MULTIPLICATION(S2,S1),S0)
        end
        S0 = S1
        S1 = S
    end
endfunction
```

Lien avec les polynômes de **Tchebychev**

Soient $T_n(x)$ et $U_n(x)$ les polynômes de **Tchebychev** de première et de deuxième espèce.

On a :

- pour *n* pair : $C_n(x) = \frac{U_{n-1}(x)}{2x}$ et $S_n(x) = \frac{U_{n-1}(x)}{2x}$
- pour *n* impair : $C_n(x) = U_{n-1}(x)$ et $S_n(x) = \frac{T_n(x)}{x}$

De la relation $T_n(x) = U_n(x) - xU_{n-1}(x)$, on déduit :

- pour *n* pair : $T_n(x) = C_{n+1}(x) - 2x^2 C_n(x)$
- pour *n* impair : $T_n(x) = 2xC_{n+1}(x) - xC_n(x)$

Comme on connait les expressions algébriques des coefficients de $C_n(x)$ et de $C_{n+1}(x)$, il est possible de retrouver la formule du polynôme de **Tchebychev** $T_n(x)$:

$$T_n(x) = \sum_{k=0}^{E\left(\frac{n}{2}\right)} (-1)^k \left[\frac{n\, 2^{n-2k} (n-k-1)!}{2\,(k)!\,(n-2k)!} \right] x^{n-2k}$$

Cette expression permet d'écrire une fonction de calcul de $T_n(x)$ plus rapide que celle basée sur la relation existant entre les polynômes d'ordres successifs :

```
function T = P_TCHEBYCHEV(n)
    T = list()
    for k = 0:floor(n/2)
        s = (-1)^k
        a1 = entier(n)
        a2 = puissance(2,n-2*k)
        a3 = factorielle(n-k-1)
        a = multiplication(multiplication(a1,a2),a3)
        b1 = entier(2)
        b2 = factorielle(k)
        b3 = factorielle(n-2*k)
        b = multiplication(multiplication(b1,b2),b3)
        [q,r] = division(a,b)
        T($+1) = list([s,q],[1,1])
        T($+1) = FRACTION(0,1)
    end
    if modulo(n,2) == 0 then, T($) = null(), end
    if n == 0 then, T = list(FRACTION(1,1)), end
endfunction
```

Relations entre les $C_n(x)$ et les $S_n(x)$

En utilisant les relations de récurrence précédentes, on trouve :

- pour n pair :

  $C_n(x) = S_n(x)$

- pour n impair :

  $C_n(x) = S_{n+1}(x) + S_{n-1}(x) = C_{n+1}(x) + C_{n-1}(x)$

  $S_n(x) = C_{n+1}(x) - C_{n-1}(x) = S_{n+1}(x) - S_{n-1}(x)$

Equations différentielles des $C_n(x)$ et des $S_n(x)$

Les polynômes de **Tchebychev** satisfont les équations différentielles du second ordre suivantes :

$(1 - x^2)T_n''(x) - xT_n'(x) + n^2T_n(x) = 0$

$(1 - x^2)U_n''(x) - 3xU_n'(x) + n(n + 2)U_n(x) = 0$

Par suite :

- pour n pair :

  De : $U_{n-1}(x) = 2xC_n(x)$, on déduit :

  $(1 - x^2)xC_n''(x) + (2 - 5x^2)C_n'(x) + (n^2 - 4)xC_n(x) = 0$

  De : $U_{n-1}(x) = 2xS_n(x)$, on déduit :

  $(1 - x^2)xS_n''(x) + (2 - 5x^2)S_n'(x) + (n^2 - 4)xS_n(x) = 0$

- pour n impair :

  De : $U_{n-1}(x) = C_n(x)$, on déduit :

  $(1 - x^2)C_n''(x) - 3xC_n'(x) + (n^2 - 1)C_n(x) = 0$

  De : $T_n(x) = xS_n(x)$, on déduit :

  $(1 - x^2)xS_n''(x) + (2 - 3x^2)S_n'(x) + (n^2 - 1)xS_n(x) = 0$

Mathématique & Informatique

Orthogonalité des $C_n(x)$ *et* des $S_n(x)$

Soient $C_m(x)$, $C_n(x)$ deux polynômes cosinus et $S_m(x)$, $S_n(x)$ deux polynômes sinus ($m \neq n$).
Soit le produit scalaire défini par l'intégrale $\int_{-1}^{+1} x^2 \sqrt{1-x^2}\, f(x)\, g(x)\, dx$
Des relations $\int_{-1}^{+1} \sqrt{1-x^2}\, U_n^2(x)\, dx = \frac{\pi}{2}$ et $\int_{-1}^{+1} \sqrt{1-x^2}\, U_m(x)\, U_n(x)\, dx = 0$, on déduit :

n pair	$\int_{-1}^{+1} x^2 \sqrt{1-x^2}\, C_n^2(x)\, dx = \frac{\pi}{8}$
n impair	$\int_{-1}^{+1} x^2 \sqrt{1-x^2}\, C_n^2(x)\, dx = \frac{\pi}{4}$
m et n pairs	$\int_{-1}^{+1} x^2 \sqrt{1-x^2}\, C_m(x)\, C_n(x)\, dx = 0$
m et n impairs	$\int_{-1}^{+1} x^2 \sqrt{1-x^2}\, C_m(x)\, C_n(x)\, dx = \begin{cases} 0 & si\ m \neq n \pm 2 \\ +\frac{\pi}{8} & si\ m = n \pm 2 \end{cases}$
m et n de parité différente	$\int_{-1}^{+1} x^2 \sqrt{1-x^2}\, C_m(x)\, C_n(x)\, dx = \begin{cases} 0 & si\ m \neq n \pm 1 \\ +\frac{\pi}{8} & si\ m = n \pm 1 \end{cases}$

n pair	$\int_{-1}^{+1} x^2 \sqrt{1-x^2}\, S_n^2(x)\, dx = \frac{\pi}{8}$
n impair	$\int_{-1}^{+1} x^2 \sqrt{1-x^2}\, S_n^2(x)\, dx = \frac{\pi}{4}$
m et n pairs	$\int_{-1}^{+1} x^2 \sqrt{1-x^2}\, S_m(x)\, S_n(x)\, dx = 0$
m et n impairs	$\int_{-1}^{+1} x^2 \sqrt{1-x^2}\, S_m(x)\, S_n(x)\, dx = \begin{cases} 0 & si\ m \neq n \pm 2 \\ -\frac{\pi}{8} & si\ m = n \pm 2 \end{cases}$
m et n de parité différente	$\int_{-1}^{+1} x^2 \sqrt{1-x^2}\, S_m(x)\, S_n(x)\, dx = \begin{cases} 0 & si\ m \neq n \pm 1 \\ +\frac{\pi}{8} & si\ m = n - 1 \\ -\frac{\pi}{8} & si\ m = n + 1 \end{cases}$

Note

En intégrant par parties $I_{2p} = \int_{-1}^{+1} x^{2p} \sqrt{1-x^2}\, dx$, on trouve $I_{2p} = \frac{2p-1}{2p+2} I_{2p-2}$, d'où : $I_{2p} = \frac{(p+1)(2p)!}{2^{2p+1}\,(p+1)!^2}\,\pi$

Ainsi : $I_0 = \frac{\pi}{2}, I_2 = \frac{\pi}{8}, I_4 = \frac{\pi}{16}, I_6 = \frac{5\pi}{128}, I_8 = \frac{7\pi}{256}, I_{10} = \frac{21\pi}{1024}, \ldots$

Les polynômes tangente et cotangente

Il est clair que :

$$\frac{1}{cos^2\theta} = 1 + tan^2\theta$$

$$\frac{1}{sin^2\theta} = 1 + cotan^2\theta$$

En posant $X = \frac{1}{x}$ dans l'équation $\sum_{i=0}^{n} a_i x^i = 0$, on obtient l'équation inversée : $\sum_{i=0}^{n} a_{n-i} X^i = 0$.
Par suite, en remplaçant x^2 par $1 + x^2$ dans l'expression algébrique des polynômes $C_n(x)$ et $S_n(x)$ inversés, on obtient les polynômes tangente $V_n(x)$ et les polynômes cotangente $W_n(x)$ qui admettent respectivement les racines réelles $x_{\pm k} = \pm tan\frac{k\pi}{n}$ et $x_{\pm k} = \pm cotan\frac{k\pi}{n}$, l'indice k prenant les valeurs entières comprises entre 1 et $E\left(\frac{n-1}{2}\right)$.
En particulier, $V_n\left(tan\frac{\pi}{n}\right) = 0$ et $W_n\left(cotan\frac{\pi}{n}\right) = 0$.

$$V_n(x) = \sum_{k=0}^{E\left(\frac{n-1}{2}\right)} v_{2k}\left(1 + x^2\right)^k \qquad \text{avec} \qquad v_{2k} = c_{2\left(E\left(\frac{n-1}{2}\right) - k\right)}$$

$$W_n(x) = \sum_{k=0}^{E\left(\frac{n-1}{2}\right)} w_{2k}\left(1 + x^2\right)^k \qquad \text{avec} \qquad w_{2k} = s_{2\left(E\left(\frac{n-1}{2}\right) - k\right)}$$

Remarque

- pour n pair : $V_n(0) = (-1)^{\frac{n-2}{2}}\frac{n}{2}$; $W_n(0) = (-1)^{\frac{n-2}{2}}\frac{n}{2}$
- pour n impair : $V_n(0) = (-1)^{\frac{n-2}{2}} n$; $W_n(0) = (-1)^{\frac{n-1}{2}}$

Si n est pair, $V_n(x) = W_n(x)$ puisque $C_n(x) = S_n(x)$. Comme $cotan\,\theta = \frac{1}{tan\,\theta}$, on a : $V_n(x) = 0 \Leftrightarrow V_n\left(\frac{1}{x}\right) = 0$.
Il s'ensuit que la liste des coefficients des polynômes $V_n(x)$ et $W_n(x)$ est un palindrome (au signe près) lorsque n est pair.

Propriété

$$\prod_{k=1}^{E\left(\frac{n-1}{2}\right)} tan\frac{k\pi}{n} = \begin{cases} 1 & \text{si } n \text{ est pair} \\ \sqrt{n} & \text{si } n \text{ est impair} \end{cases} \qquad \text{et} \qquad \prod_{k=1}^{E\left(\frac{n-1}{2}\right)} cotan\frac{k\pi}{n} = \begin{cases} 1 & \text{si } n \text{ est pair} \\ \frac{1}{\sqrt{n}} & \text{si } n \text{ est impair} \end{cases}$$

Mathématique & Informatique

Pour calculer le polynôme $V_n(x)$, on calcule d'abord le polynôme $C_n(x)$, puis on effectue un changement de variable.
Pour calculer le polynôme $W_n(x)$, on calcule d'abord le polynôme $S_n(x)$, puis on effectue un changement de variable.
Avec le module *Scilab-Polynomes*, écrivons une fonction *P_TANGENTE* et une fonction *P_COTANGENTE* qui donnent l'expression du polynôme $V_n(x)$ et du polynôme $W_n(x)$ lorsqu'on fournit le paramètre n.
Si le monôme dominant de $V_n(x)$ (ou de $W_n(x)$) est négatif, on convient de changer tous les signes des coefficients.

```
function V = P_TANGENTE(n)
    C = P_COSINUS(n)
    V = list(FRACTION(0,1))
    P1 = list(FRACTION(1,1))
    P2 = list(FRACTION(1,1),FRACTION(0,1),FRACTION(1,1))
    for k = 0:floor((n-1)/2)
        if k == 0 then
            P = P1
        else
            P = P_MULTIPLICATION(P,P2)
        end
        V = P_ADDITION(V,P_SCALAIRE(P,C(2*k+1)))
    end
    if F_SIGNE(V(1)) < 0 then, V = P_MOINS(V), end
endfunction
```

```
function W = P_COTANGENTE(n)
    S = P_SINUS(n)
    W = list(FRACTION(0,1))
    P1 = list(FRACTION(1,1))
    P2 = list(FRACTION(1,1),FRACTION(0,1),FRACTION(1,1))
    for k = 0:floor((n-1)/2)
        if k == 0 then
            P = P1
        else
            P = P_MULTIPLICATION(P,P2)
        end
        W = P_ADDITION(W,P_SCALAIRE(P,S(2*k+1)))
    end
    if F_SIGNE(W(1)) < 0 then, W = P_MOINS(W), end
endfunction
```

Les premiers polynômes tangente sont :

$V_0(x) = 0$
$V_1(x) = 1$
$V_2(x) = 1$
$V_3(x) = x^2 - 3$
$V_4(x) = 2x^2 - 2$
$V_5(x) = x^4 - 10x^2 + 5$
$V_6(x) = 3x^4 - 10x^2 + 3$
$V_7(x) = x^6 - 21x^4 + 35x^2 - 7$
$V_8(x) = 4x^6 - 28x^4 + 28x^2 - 4$
$V_9(x) = x^8 - 36x^6 + 126x^4 - 84x^2 + 9$
$V_{10}(x) = 5x^8 - 60x^6 + 126x^4 - 60x^2 + 5$
$V_{11}(x) = x^{10} - 55x^8 + 330x^6 - 462x^4 + 165x^2 - 11$
$V_{12}(x) = 6x^{10} - 110x^8 + 396x^6 - 396x^4 + 110x^2 - 6$

Les premiers polynômes cotangente sont :

$W_0(x) = 0$
$W_1(x) = 1$
$W_2(x) = 1$
$W_3(x) = 3x^2 - 1$
$W_4(x) = 2x^2 - 2$
$W_5(x) = 5x^4 - 10x^2 + 1$
$W_6(x) = 3x^4 - 10x^2 + 3$
$W_7(x) = 7x^6 - 35x^4 + 21x^2 - 1$
$W_8(x) = 4x^6 - 28x^4 + 28x^2 - 4$
$W_9(x) = 9x^8 - 84x^6 + 126x^4 - 36x^2 + 1$
$W_{10}(x) = 5x^8 - 60x^6 + 126x^4 - 60x^2 + 5$
$W_{11}(x) = 11x^{10} - 165x^8 + 462x^6 - 330x^4 + 55x^2 - 1$
$W_{12}(x) = 6x^{10} - 110x^8 + 396x^6 - 396x^4 + 110x^2 - 6$

L'instruction *P_ECRIRE(P_TANGENTE(51))* affiche $V_{51}(x)$:

$1\ x^{50}$
$-\ 1275\ x^{48}$
$249900\ x^{46}$
$-\ 18009460\ x^{44}$
$636763050\ x^{42}$
$-\ 12777711870\ x^{40}$
$158753389900\ x^{38}$
$-\ 1292706174900\ x^{36}$
$7174519270695\ x^{34}$
$-\ 27900908274925\ x^{32}$
$77535155627160\ x^{30}$
$-\ 156077261327400\ x^{28}$
$229591913401900\ x^{26}$
$-\ 247959266474052\ x^{24}$
$196793068630200\ x^{22}$
$-\ 114456658306760\ x^{20}$
$48459472266975\ x^{18}$
$-\ 14771069086725\ x^{16}$
$3188675231420\ x^{14}$
$-\ 476260169700\ x^{12}$
$47626016970\ x^{10}$
$-\ 3042312350\ x^{8}$
$115775100\ x^{6}$
$-\ 2349060\ x^{4}$
$20825\ x^{2}$
$-\ 51$

L'instruction *P_ECRIRE(P_COTANGENTE(51))* affiche $W_{51}(x)$:

$51\ x^{50}$
$-\ 20825\ x^{48}$
$2349060\ x^{46}$
$-\ 115775100\ x^{44}$
$3042312350\ x^{42}$
$-\ 47626016970\ x^{40}$
$476260169700\ x^{38}$
$-\ 3188675231420\ x^{36}$
$14771069086725\ x^{34}$
$-\ 48459472266975\ x^{32}$
$114456658306760\ x^{30}$
$-\ 196793068630200\ x^{28}$
$247959266474052\ x^{26}$
$-\ 229591913401900\ x^{24}$
$156077261327400\ x^{22}$
$-\ 77535155627160\ x^{20}$
$27900908274925\ x^{18}$
$-\ 7174519270695\ x^{16}$
$1292706174900\ x^{14}$
$-\ 158753389900\ x^{12}$
$12777711870\ x^{10}$
$-\ 636763050\ x^{8}$
$18009460\ x^{6}$
$-\ 249900\ x^{4}$
$1275\ x^{2}$
$-\ 1$

Mathématique & Informatique

Polynômes de Bernoulli

On a déjà calculé les nombres de **Bernoulli** B_n
Ils sont liés aux polynômes de **Bernoulli** $B_n(x)$ par la relation :

$$B_n(x) = \sum_{k=0}^{n} C_n^k B_k \, x^{n-k}$$

L'identité précédente est vraie pour tout entier naturel n ($n = 0, 1, 2, 3, 4, 5, ...$). Le polynôme $B_n(x)$ est de degré n, son monôme dominant est x^n, et son terme constant est le nombre de **Bernoulli** B_n (nul pour n impair, sauf B_1).

$$B_n(x) = x^n - \frac{n}{2} x^{n-1} + \frac{n(n-1)}{12} x^{n-2} + \cdots + nB_{n-1}x + B_n$$

La fonction Scilab *P_BERNOULLI* calcule le polynôme de **Bernoulli** de degré n :

```
function B = P_BERNOULLI(n)
  B = list(FRACTION(1,1))
  if n == 0 then, return, end
  b0 = FRACTION(1,1)
  b1 = FRACTION(-1,2)
  b = list(b0,b1)
  B($+1) = FRACTION(-n,2)
  for m = 2:n
    if modulo(m,2) == 1 then
      B($+1) = FRACTION(0,1)
      continue
    end
    S = FRACTION(0,1)
    for k = [0,1,2:2:m-2]
      T1 = list([1,binomiaux(m+1,k)],[1,1])
      if k > 1 then, T2 = b(k/2+2), else, T2 = b(k+1), end
      T3 = F_MULTIPLICATION(T1,T2)
      S = F_ADDITION(S,T3)
    end
    T = FRACTION(-1,m+1)
    b($+1) = F_MULTIPLICATION(T,S)
    T = list([1, binomiaux(n,m)],[1,1])
    B($+1) = F_MULTIPLICATION(T,b($))
  end
endfunction
```

Par exemple, l'instruction *B50 = P_BERNOULLI(50)* calcule le polynôme de **Bernoulli** de degré 50
L'instruction *P_ECRIRE(B50)* affiche à la Console les coefficients du polynôme de **Bernoulli** $B_{50}(x)$
L'instruction *ECRIRE(B50($))* affiche à la Console le nombre de **Bernoulli** B_{50}

Les polynômes de **Bernoulli** apparaissent dans diverses formules mathématiques. Citons la formule de **Faulhaber** :

$$\sum_{k=0}^{n} k^p = \frac{B_{p+1}(n+1) - B_{p+1}(0)}{p+1}$$

Mathématique & Informatique

Les premiers polynômes de **Bernoulli** :

$B_0(x) = 1$
$B_1(x) = \mathbf{x} - 1/2$
$B_2(x) = \mathbf{x^2} - \mathbf{x} + 1/6$
$B_3(x) = \mathbf{x^3} - 3/2\ \mathbf{x^2} + 1/2\ \mathbf{x}$
$B_4(x) = \mathbf{x^4} - 2\ \mathbf{x^3} + \mathbf{x^2} - 1/30$
$B_5(x) = \mathbf{x^5} - 5/2\ \mathbf{x^4} + 5/3\ \mathbf{x^3} - 1/6\ \mathbf{x}$
$B_6(x) = \mathbf{x^6} - 3\ \mathbf{x^5} + 5/2\ \mathbf{x^4} - 1/2\ \mathbf{x^2} + 1/42$
$B_7(x) = \mathbf{x^7} - 7/2\ \mathbf{x^6} + 7/2\ \mathbf{x^5} - 7/6\ \mathbf{x^3} + 1/6\ \mathbf{x}$
$B_8(x) = \mathbf{x^8} - 4\ \mathbf{x^7} + 14/3\ \mathbf{x^6} - 7/3\ \mathbf{x^4} + 2/3\ \mathbf{x^2} - 1/30$
$B_9(x) = \mathbf{x^9} - 9/2\ \mathbf{x^8} + 6\ \mathbf{x^7} - 21/5\ \mathbf{x^5} + 2\ \mathbf{x^3} - 3/10\ \mathbf{x}$
$B_{10}(x) = \mathbf{x^{10}} - 5\ \mathbf{x^9} + 15/2\ \mathbf{x^8} - 7\ \mathbf{x^6} + 5\ \mathbf{x^4} - 3/2\ \mathbf{x^2} + 5/66$

Le polynôme de **Bernoulli** $B_{50}(x)$:
==========

$1\ \mathbf{x^{50}}$
$-25\ \mathbf{x^{49}}$
$1225/6\ \mathbf{x^{48}}$
$-23030/3\ \mathbf{x^{46}}$
$378350\ \mathbf{x^{44}}$
$-17895955\ \mathbf{x^{42}}$
$2334608675/3\ \mathbf{x^{40}}$
$-92183691110/3\ \mathbf{x^{38}}$
$3282459797050/3\ \mathbf{x^{36}}$
$-1187265708592985/34\ \mathbf{x^{34}}$
$5954542492816225/6\ \mathbf{x^{32}}$
$-74811626173909996/3\ \mathbf{x^{30}}$
$1648649367199981300/3\ \mathbf{x^{28}}$
$-136808278770228935730/13\ \mathbf{x^{26}}$
$173269701467076273150\ \mathbf{x^{24}}$
$-7268138892957127666460/3\ \mathbf{x^{22}}$
$85056098063910933337700/3\ \mathbf{x^{20}}$
$-27836090018381093355510835/102\ \mathbf{x^{18}}$
$42305783822167959822086 75/2\ \mathbf{x^{16}}$
$-50151723471623976337895 5190/39\ \mathbf{x^{14}}$
$17785011943274436164003 2450/3\ \mathbf{x^{12}}$
$-59465949218965178247453 5017/3\ \mathbf{x10}$
$45188702709574449786996 7025\ \mathbf{x^8}$
$-64100019841130258813257 9690\ \mathbf{x^6}$
$14613052234050848000310 04450/3\ \mathbf{x^4}$
$-19632911791492361901871 9464145/1326\ \mathbf{x^2}$
$49505720524107964821247 7525/66$

Autre méthode

Les polynômes de **Bernoulli** $B_n(x)$ peuvent être définis comme la suite de polynômes vérifiant les relations :

$$B_0(x) = 1$$
$$B_n'(x) = n\,B_{n-1}(x)$$
$$\int_0^1 B_n(x)\,dx = 0$$

Les nombres de **Bernoulli** B_n sont les termes constants de ces polynômes :

$$B_n = B_n(0)$$

En utilisant uniquement les fonctions polynomiales prédéfinies, écrivons une fonction Scilab *P_BERNOULLI(n)* qui retourne le polynôme de **Bernoulli** de degré *n* :

```
function B = P_BERNOULLI(n)
  B = list(FRACTION(1,1))
  for i = 1:n
    B = P_PRIMITIVE(B)
    B = P_SCALAIRE(B,FRACTION(i,1))
    P = P_PRIMITIVE(B)
    K = P_HORNER(P,FRACTION(1,1))
    B($) = F_MOINS(K)
  end
endfunction
```

Par exemple, le programme suivant affiche à la Console le polynôme $B_{20}(x)$ et le nombre de **Bernoulli** B_{20} :

```
PB20 = P_BERNOULLI(20) ;
P_ECRIRE(PB20)
NB20 = PB20($) ;
ECRIRE(NB20)
```

```
B20(x) :
====
x^20 - 10 x^19 + 95/3 x^18 - 323/2 x^16 + 6460/7 x^14 - 4199 x^12 + 41990/3 x^10
- 223193/7 x^8 + 45220 x^6 - 68723/2 x^4 + 219335/21 x^2 - 174611/330

B20 :
===
- 174611/330
```

Avec la fonction *L_BERNOULLI(n)*, on obtient la liste des nombres de **Bernoulli** de B_0 à B_n .

```
function L = L_BERNOULLI(n)
  L = list(FRACTION(1,1))
  B = L
  for i = 1:n
    B = P_PRIMITIVE(B)
    B = P_SCALAIRE(B,FRACTION(i,1))
    P = P_PRIMITIVE(B)
    K = P_HORNER(P,FRACTION(1,1))
    B($) = F_MOINS(K)
    L($+1) = B($)
  end
endfunction
```

Par exemple, l'exécution de l'instruction *AFFICHE(L_BERNOULLI(30))* fournit la liste suivante :

```
1
- 1 / 2
1 / 6
0
- 1 / 30
0
1 / 42
0
- 1 / 30
0
5 / 66
0
- 691 / 2730
0
7 / 6
0
- 3617 / 510
0
43867 / 798
0
- 174611 / 330
0
854513 / 138
0
- 236364091 / 2730
0
8553103 / 6
0
- 23749461029 / 870
0
8615841276005 / 14322
```

Polynômes d'Euler

Tout d'abord, introduisons les nombres sécants et les nombres tangents.
Ce sont des entiers naturels A_n qui figurent alternativement dans le développement en série de **sec** x et de **tan** x.

$$\sec x = A_0 + A_2 \frac{x^2}{2!} + A_4 \frac{x^4}{4!} + \cdots + A_{2p} \frac{x^{2p}}{(2p)!} + \cdots$$

$$\tan x = A_1 + A_3 \frac{x^3}{3!} + A_5 \frac{x^5}{5!} + \cdots + A_{2p+1} \frac{x^{2p+1}}{(2p+1)!} + \cdots$$

Les A_n d'indices pairs, qui sont dans **sec** x, sont appelés nombres sécants et les A_n d'indices impairs, qui sont dans **tan** x, sont appelés nombres tangents. On utilise le terme de nombres ZigZag pour désigner les A_n dans leur ensemble, les nombres sécants sont nommés nombres Zig et les nombres tangents nombres Zag.

Pour tout entier n non nul, on montre que :

$$2A_{n+1} = \sum_{k=0}^{n} C_n^k A_k A_{n-k}$$

En partant de $A_0 = A_1 = 1$, cette formule permet de calculer successivement A_2, A_3, A_4, A_5, ... On trouve :

A_n = 1, 1, 1, 2, 5, 16, 61, 272, 1385, 7936, 50521, 353792, 2702765, 22368256, 199360981, 1903757312, 19391512145, 209865342976, 2404879675441, 29088885112832, 370371188237525, ...

Remarque : Les nombres sécants s'identifient aux nombres d'**Euler** non nuls. On a : $A_{2p} = E_{2p}$

Le polynôme d'**Euler** de degré n s'exprime par :

$$E_n(x) = \sum_{k=0}^{n} C_n^k N_k x^{n-k}$$

où les N_k sont des nombres rationnels définis de la manière suivante :

- $N_0 = 1$
- $N_{2p-1} = (-1)^p A_{2p-1}/2^{2p-1} \quad , \quad p \neq 0$
- $N_{2p} = 0 \quad , \quad p \neq 0$

Remarque : Les nombres N_k sont liés aux nombres de **Bernoulli** : $N_k = -[2(2^{k+1} - 1)/(k+1)]B_{k+1}$

Comme les polynômes de **Bernoulli**, les polynômes d'**Euler** sont des polynômes unitaires :

$$E_n(x) = x^n - \frac{n}{2} x^{n-1} + \frac{n(n-1)(n-2)}{24} x^{n-3} + \cdots + nN_{k-1}x + N_k$$

Si n est pair, tous les coefficients de $E_n(x)$ sont des nombres entiers.

La fonction Scilab *P_EULER* calcule le polynôme d'**Euler** de degré *n* :

```
function E = P_EULER(n)
  E = list(FRACTION(1,1))
  if n == 0 then, return, end
  A = list(1,1)
  for m = 2:n
    S = 0
    for k = 0:m-1
      T1 = multiplication(A(k+1),A(m-k))
      T2 = binomiaux(m-1,k)
      T3 = multiplication(T1,T2)
      S = addition(S,T3)
    end
    [A($+1),r] = division(S,2)
  end
  for k = 1:n
    if modulo(k,2) == 0 then
      E($+1) = FRACTION(0,1)
    else
      T1 = list([1, binomiaux(n,k)],[1,1])
      T2 = list([(-1)^((k+1)/2),A(k+1)],[1,puissance(2,k)])
      E($+1) = F_MULTIPLICATION(T1,T2)
    end
  end
endfunction
```

Le programme :

```
for n = 0:10
  P_ECRIRE(P_EULER(n))
end
```

donne les premiers polynômes d'**Euler** :

$E_0(x) = 1$
$E_1(x) = \mathbf{x} - 1/2$
$E_2(x) = \mathbf{x}^2 - \mathbf{x}$
$E_3(x) = \mathbf{x}^3 - 3/2\,\mathbf{x}^2 + 1/4$
$E_4(x) = \mathbf{x}^4 - 2\,\mathbf{x}^3 + \mathbf{x}$
$E_5(x) = \mathbf{x}^5 - 5/2\,\mathbf{x}^4 + 5/2\,\mathbf{x}^2 - 1/2$
$E_6(x) = \mathbf{x}^6 - 3\,\mathbf{x}^5 + 5\,\mathbf{x}^3 - 3\,\mathbf{x}$
$E_7(x) = \mathbf{x}^7 - 7/2\,\mathbf{x}^6 + 35/4\,\mathbf{x}^4 - 21/2\,\mathbf{x}^2 + 17/8$
$E_8(x) = \mathbf{x}^8 - 4\,\mathbf{x}^7 + 14\,\mathbf{x}^5 - 28\,\mathbf{x}^3 + 17\,\mathbf{x}$
$E_9(x) = \mathbf{x}^9 - 9/2\,\mathbf{x}^8 + 21\,\mathbf{x}^6 - 63\,\mathbf{x}^4 + 153/2\,\mathbf{x}^2 - 31/2$
$E_{10}(x) = \mathbf{x}^{10} - 5\,\mathbf{x}^9 + 30\,\mathbf{x}^7 - 126\,\mathbf{x}^5 + 255\,\mathbf{x}^3 - 155\,\mathbf{x}$

Mathématique & Informatique

Le polynôme d'**Euler** $E_{50}(x)$:
==========

$1\ \mathbf{x^{50}}$
$-\ 25\ \mathbf{x^{49}}$
$4900\ \mathbf{x^{47}}$
$-\ 1059380\ \mathbf{x^{45}}$
$212254350\ \mathbf{x^{43}}$
$-\ 38834222350\ \mathbf{x^{41}}$
$6452858377700\ \mathbf{x^{39}}$
$-\ 968946646037300\ \mathbf{x^{37}}$
$130768837332170205\ \mathbf{x^{35}}$
$-\ 15767087198932562325\ \mathbf{x^{33}}$
$1687002170221670409800\ \mathbf{x^{31}}$
$-\ 158964022684942141759400\ \mathbf{x^{29}}$
$13078415422837985492668900\ \mathbf{x^{27}}$
$-\ 930234612624393650029914276\ \mathbf{x^{25}}$
$56551483209685300308384879400\ \mathbf{x^{23}}$
$-\ 2899310787060650668941380083400\ \mathbf{x^{21}}$
$123379872290632347583344524605925\ \mathbf{x^{19}}$
$-\ 4275340186759548736455590112221325\ \mathbf{x^{17}}$
$1178256476693475991323145851891025 80\ \mathbf{x^{15}}$
$-\ 250702915790950464543573223869745 1300\ \mathbf{x^{13}}$
$396263652260181169914549220622974119 50\ \mathbf{x^{11}}$
$-\ 441648925095810203465626019837090306350\ \mathbf{x^{9}}$
$322188421284530384147078257663167626 6100\ \mathbf{x^{7}}$
$-\ 13710695124170012066332070004375885813380\ \mathbf{x^{5}}$
$277836771708026973338146049244317829622 75\ \mathbf{x^{3}}$
$-\ 168904503412939657791756293891016696832 75\ \mathbf{x}$

Le polynôme d'**Euler** $E_{51}(x)$:
==========

$1\ \mathbf{x^{51}}$
$-\ 51 / 2\ \mathbf{x^{50}}$
$20825 / 4\ \mathbf{x^{48}}$
$-\ 1174530\ \mathbf{x^{46}}$
$492044175 / 2\ \mathbf{x^{44}}$
$-\ 47155841425\ \mathbf{x^{42}}$
$16454788863135 / 2\ \mathbf{x^{40}}$
$-\ 1300428393365850\ \mathbf{x^{38}}$
$741023411548964495 / 4\ \mathbf{x^{36}}$
$-\ 47301261596797686975 / 2\ \mathbf{x^{34}}$
$10754638835163148862475 / 4\ \mathbf{x^{32}}$
$-\ 27023883856440164099 0980\ \mathbf{x^{30}}$
$23821399520169187861646925\ \mathbf{x^{28}}$
$-\ 18246909709170798519817549 26\ \mathbf{x^{26}}$
$12017190182058126315531786 8725\ \mathbf{x^{24}}$
$-\ 67211295518224174598186538 29700\ \mathbf{x^{22}}$
$12584746973644499453501141509804 35 / 4\ \mathbf{x^{20}}$
$-\ 24226927724970776173248343969254175 / 2\ \mathbf{x^{18}}$
$15022770077841818889370109611610578 95 / 4\ \mathbf{x^{16}}$
$-\ 91327490752417669226587388695407154 50\ \mathbf{x^{14}}$
$33682410442115399442736683752952800157 5 / 2\ \mathbf{x^{12}}$
$-\ 22524095179886320376746927011691605623 85\ \mathbf{x^{10}}$
$41079023713777623978752477852053872392775 / 2\ \mathbf{x^{8}}$
$-\ 11654090855544510256382259503719502941373 0\ \mathbf{x^{6}}$
$14169675357109375640245448511460209310760 25 / 4\ \mathbf{x^{4}}$
$-\ 86141296740599225473795709884418515384702 5 / 2\ \mathbf{x^{2}}$
$34911752584973464926124521086412852527214 1 / 4$

Autre méthode

Les polynômes d'**Euler** $E_n(x)$ peuvent être définis comme la suite de polynômes vérifiant les relations :

$$E_0(x) = 1$$
$$E_n'(x) = n\,E_{n-1}(x)$$
$$E_n(0) + E_n(1) = 0$$

Les nombres d'**Euler** E_n , présents dans le développement en série de sec(x), sont tous positifs et donnés par :

$$E_n = \left| 2^n E_n\left(\frac{1}{2}\right) \right|$$

En utilisant uniquement les fonctions polynomiales prédéfinies, écrivons une fonction Scilab *P_EULER(n)* qui retourne le polynôme d'**Euler** de degré n :

```
function E = P_EULER(n)
  E = list(FRACTION(1,1))
  for i = 1:n
    E = P_PRIMITIVE(E)
    E = P_SCALAIRE(E,FRACTION(i,1))
    K = P_HORNER(E,FRACTION(1,1))
    E($) = F_DIVISION(K,FRACTION(-2,1))
  end
endfunction
```

Par exemple, le programme suivant affiche à la Console le polynôme $E_{20}(x)$ et le nombre d'**Euler** E_{20} :

```
PE20 = P_EULER(20) ;
P_ECRIRE(PE20)
K1 = F_PUISSANCE(FRACTION(2,1),20) ;
K2 = P_HORNER(PE20,FRACTION(1,2)) ;
NE20 = F_ABSOLU(F_MULTIPLICATION(K1,K2)) ;
ECRIRE(NE20)
```

```
E20(x) :
=====
x^20 - 10 x^19 + 285 x^17 - 7752 x^15 + 164730 x^13 - 2603380 x^11 + 29015090 x^9
- 211668360 x^7 + 900752361 x^5 - 1825305870 x^3 + 1109652905 x

E20 :
===
370371188237525
```

Avec la fonction *L_EULER(n)*, on obtient la liste des nombres d'**Euler** de E_0 à E_n .

```
function L = L_EULER(n)
  L = list(FRACTION(1,1))
  E = L
  for i = 1:n
    E = P_PRIMITIVE(E)
    E = P_SCALAIRE(E,FRACTION(i,1))
    K = P_HORNER(E,FRACTION(1,1))
    E($) = F_DIVISION(K,FRACTION(-2,1))
    K1 = F_PUISSANCE(FRACTION(2,1),i)
    K2 = P_HORNER(E,FRACTION(1,2))
    L($+1) = F_ABSOLU(F_MULTIPLICATION(K1,K2))
  end
endfunction
```

Par exemple, l'exécution de l'instruction *AFFICHE(L_EULER(30))* fournit la liste suivante :

```
1
0
1
0
5
0
61
0
1385
0
50521
0
2702765
0
199360981
0
19391512145
0
2404879675441
0
370371188237525
0
69348874393137901
0
15514534163557086905
0
4087072509293123892361
0
1252259641403629865468285
0
441543893249023104553682821
```

Polynôme de Lagrange

- Le problème

Par 2 points il passe une et une seule droite, par 3 points une et une seule parabole, et d'une manière générale par n+1 points un et un seul polynôme de degré n. Ce polynôme est appelé polynôme d'interpolation de **Lagrange**.

- La solution algébrique

Considérons les n+1 points : (x_0,y_0) , (x_1,y_1) , ... , (x_n,y_n) avec $x_i \neq x_j$
On recherche un polynôme $L(x)$ de degré n tel que : $L(x_i) = y_i$, $\forall i \quad (0 \leq i \leq n)$
Soient $L_i(x)$ les polynômes de degré n définis par : $L_i(x_j) = \delta_{ij}$ (symbole de **Kronecker** = 1 si $i = j$ et 0 sinon)
On connait les n racines de L_i : $L_i(x) = \lambda(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)$
et la valeur que prend L_i en x_i : $L_i(x_i) = \lambda(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n) = 1$
D'où :

$$L_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

$$L(x) = \sum_{i=0}^{n} y_i L_i(x)$$

Posons : $L(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$
La condition $L(x_i) = y_i$, $0 \leq i \leq n$, revient à résoudre un système linéaire de n+1 équations à n+1 inconnues a_i :

$$\begin{cases} (1)a_0 + (x_0)a_1 + \cdots + (x_0{}^{n-1})a_{n-1} + (x_0{}^{n})a_n = y_0 \\ \vdots \\ \vdots \\ \vdots \\ (1)a_0 + (x_n)a_1 + \cdots + (x_n{}^{n-1})a_{n-1} + (x_n{}^{n})a_n = y_n \end{cases}$$

Le déterminant $\Delta = \det(x_i^j)$, $0 \leq i,j \leq n$, de ce système est appelé déterminant de **Vandermonde**.
En effectuant n opérations élémentaires sur les colonnes : $C_{j+1} \rightarrow C_{j+1} - x_n C_j \quad (j = n, \dots , 1)$, puis en développant Δ par rapport à la dernière ligne, on obtient la relation de récurrence : $\Delta_{n+1} = (x_n - x_{n-1}) \dots (x_n - x_1)(x_n - x_0)\Delta_n$
On déduit :

$$\Delta = \prod_{j<i} (x_i - x_j)$$

Avec l'hypothèse $x_i \neq x_j$, on a $\Delta \neq 0$ et le système admet une solution unique en a_i.
Donc, il existe un et un seul polynôme $L(x)$ de degré n qui répond à la question.

- Cas particulier

Interpolation linéaire : (x_0,y_0) , $(x_1,y_1) \quad \Rightarrow \quad L(x) = [(y_1 - y_0)/(x_1 - x_0)]x + [(x_1y_0 - x_0y_1)/(x_1 - x_0)]$

La fonction Scilab *P_LAGRANGE* prend en paramètres d'entrée 2 matrices colonnes *X*, *Y* qui sont les coordonnées des points d'interpolation, et retourne en paramètre de sortie le polynôme *L* qui passe par tous les points.

```
function L = P_LAGRANGE(X,Y)
  L = list()
  l1 = X($-1) ; c1 = X($)
  l2 = Y($-1) ; c2 = Y($)
  if l1 <> l2 | c1 <> c2 then, return, end
  V = list()
  N = X($-1)
  for i = 1:N
    for j = 1:N
      if j == 1 then
        V($+1) = FRACTION(1,1)
      else
        V($+1) = F_MULTIPLICATION(V($),X(i))
      end
    end
  end
  V($+1) = N ; V($+1) = N
  a = GAUSS(V,Y)
  for i = length(a)-2:-1:1, L($+1) = a(i), end
endfunction
```

Note : La matrice *V* dont les éléments sont les x_i^j $(0 \le i,j \le n)$ est appelée matrice de **Vandermonde** d'ordre $n+1$

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} & x_1^n \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} & x_n^n \end{pmatrix}$$

$$X = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} \quad , \quad Y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

EXEMPLE

Déterminer les coefficients rationnels du polynôme de degré 7 qui passe par les 8 points suivants :

x_i	-83/12	-13/3	-29/9	-1	2/3	5	10	101/8
y_i	-16/3	34/11	-7/2	2	1/4	11	-1/5	-2

Notons X le vecteur des abscisses x_i et Y le vecteur des ordonnées y_i :

$$X = \begin{pmatrix} -83/12 \\ -13/3 \\ -29/9 \\ -1 \\ 2/3 \\ 5 \\ 10 \\ 101/8 \end{pmatrix} \quad , \quad Y = \begin{pmatrix} -16/3 \\ 34/11 \\ -7/2 \\ 2 \\ 1/4 \\ 11 \\ -1/5 \\ -2 \end{pmatrix}$$

Le programme de résolution s'écrit :

```
X = list(FRACTION(-83,12),FRACTION(-13,3),FRACTION(-29,9),FRACTION(-1,1),...
        FRACTION(2,3),FRACTION(5,1),FRACTION(10,1),FRACTION(101,8),8,1) ;
Y = list(FRACTION(-16,3),FRACTION(34,11),FRACTION(-7,2),FRACTION(2,1),...
        FRACTION(1,4),FRACTION(11,1),FRACTION(-1,5),FRACTION(-2,1),8,1) ;
L = P_LAGRANGE(X,Y) ;
P_ECRIRE(L)
```

Les coefficients a_i du polynôme de **Lagrange** $L(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ sont :

$$a_7 = \frac{7750352348614781223198072 3}{5993321665634532332824492021 25}$$

$$a_6 = -\frac{1889286627009698451400 23}{9640408831823918500572220 0}$$

$$a_5 = -\frac{3816870829230516683334433047}{5183413332440676612172533640 00}$$

$$a_4 = \frac{3968225162681346634521949385 79}{2739804190004357637862624924000}$$

$$a_3 = \frac{1079636110307985036917750774 23}{6186654622590484987220562800 0}$$

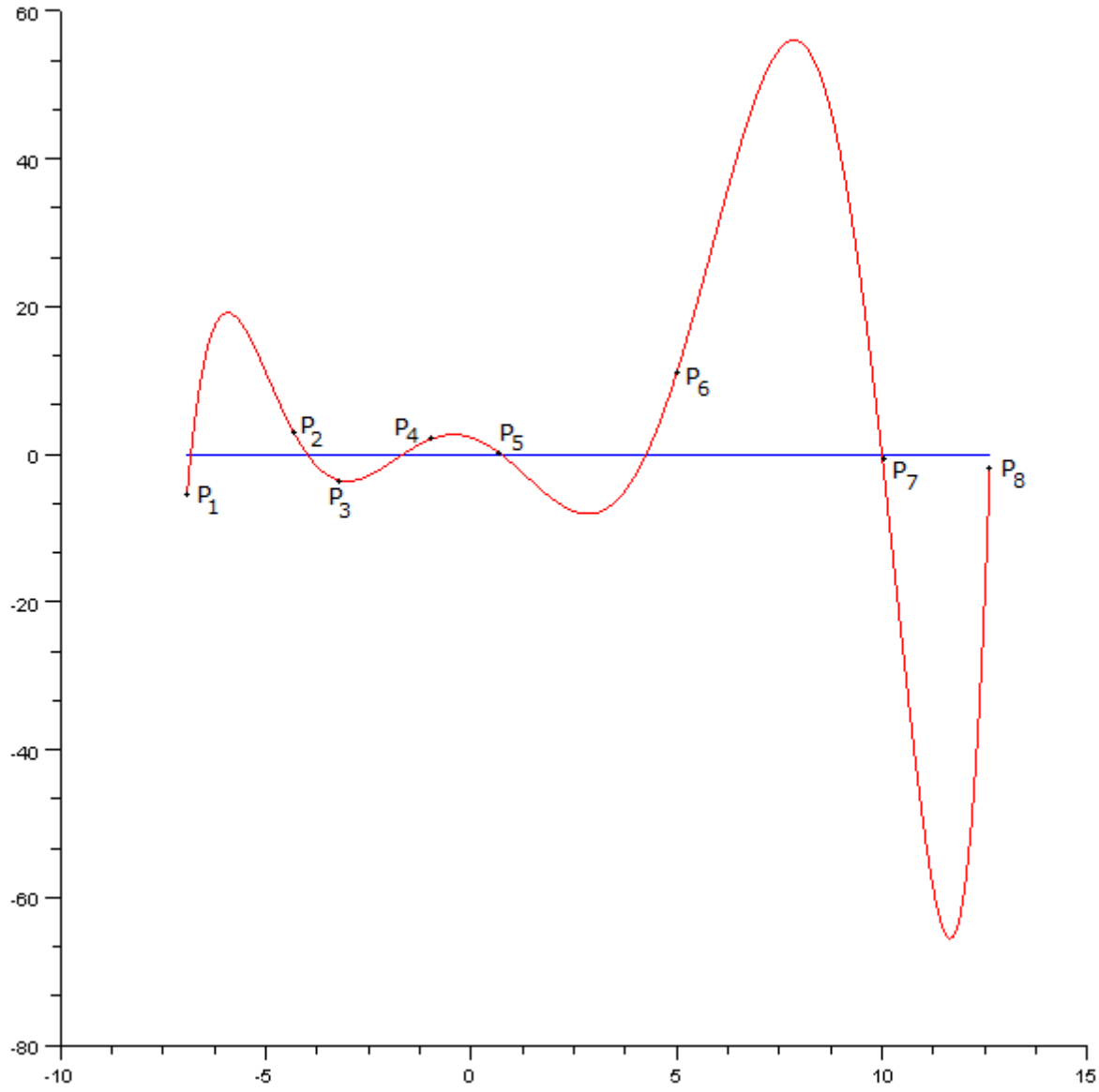
$$a_2 = -\frac{3564688751765949117067638130443}{1743511757275500315003488588000}$$

$$a_1 = -\frac{3718921473699079493850937935}{2073365332976270644869013456}$$

$$a_0 = \frac{5787369626800468358578177 83}{2543584791781233881304824 20}$$

Avec la fonction *plot2d* positionnons les points (x_i,y_i) et dessinons la courbe représentative de $L(x)$:

```
N = 8 ;
x = zeros(1,N) ;
y = zeros(1,N) ;
for i = 1:N
   x(i) = nombre_rationnel(X(i)) ;
   y(i) = nombre_rationnel(Y(i)) ;
end
plot2d(x,y,0)
P_GRAPHIQUE(L,min(x),max(x))
```



Polynôme d'ajustement

- Le problème

On recherche un polynôme $A(x) = a_n x^n + ... + a_1 x + a_0$ de degré n qui réalise la meilleure approximation au sens des moindres carrés d'un ensemble de N points $(x_i,y_i)_{\ 1 \le i \le N}$

- La solution

Il s'agit de minimiser la quantité :

$$\sum_{i=1}^{N} \left(a_n {x_i}^n + \cdots + a_1 x_i + a_0 - y_i \right)^2$$

Cette somme est une fonction e des a_k. Elle atteint un extremum lorsque toutes ses dérivées partielles s'annulent :

$$\frac{\partial e}{\partial a_k}(a_0, a_1, \cdots, a_n) = 0 \quad , \quad \forall k = 0, \cdots, n$$

Les n+1 équations précédentes reviennent à trouver la solution $a = (a_0 , a_1 , ... , a_n)$ du système linéaire suivant :

$$Sa = b$$

$$S = \begin{pmatrix} S_0 & S_1 & \cdots & S_n \\ S_1 & S_2 & \cdots & S_{n+1} \\ \vdots & \vdots & & \vdots \\ S_n & S_{n+1} & \cdots & S_{2n} \end{pmatrix} \qquad b = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{pmatrix}$$

$$S_p = \sum_{i=1}^{N} {x_i}^p \quad (0 \le p \le 2n) \qquad , \qquad b_q = \sum_{i=1}^{N} {x_i}^q y_i \quad (0 \le q \le n)$$

S est une matrice carrée symétrique d'ordre n+1. Elle est définie positive puisqu'on a : $S = {}^t M\, M$, avec :

$$M = \begin{pmatrix} 1 & x_1 & {x_1}^2 & \cdots & {x_1}^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_N & {x_N}^2 & \cdots & {x_N}^n \end{pmatrix}$$

Par suite, le noyau de S est réduit à 0, S est inversible, et le système linéaire $Sa = b$ admet une solution unique.
La fonction e est un polynôme de degré 2 à n+1 variables a_k ; ses dérivées partielles d'ordre ≥ 3 sont nulles.

Le développement de **Taylor** de e pour un écart $h = (h_0 , ... , h_n)$ autour de la solution $a = (a_0 , ... , a_n)$ s'écrit :

$$e(a+h) = e(a) + {}^t hSh = e(a) + \| Mh \|^2 \ge e(a)$$

Donc, e prend sa valeur minimale en a.

- Remarques

1. Lorsque A est du premier degré ($A(x) = a_1 x + a_0$) on obtient la droite des moindres carrés discrets.
2. Pour $n = N - 1$ on retrouve le polynôme d'interpolation de **Lagrange** qui passe par les N points ($e(a) = 0 \Rightarrow A = L$).

La fonction Scilab *P_AJUSTEMENT* prend en paramètres d'entrée 2 matrices colonnes *X, Y* qui sont les coordonnées des points d'interpolation, plus un entier *n* qui est le degré du polynôme d'ajustement, et retourne en paramètre de sortie le polynôme *A* qui passe "au mieux" par tous les points.

```
function A = P_AJUSTEMENT(X,Y,n)
  A = list()
  l1 = X($-1) ; c1 = X($)
  l2 = Y($-1) ; c2 = Y($)
  if l1 <> l2 | c1 <> c2 then, return, end
  S = M_VIDE(n+1,n+1)
  b = M_VIDE(n+1,1)
  N = X($-1)
  for i = 1:n+1
    for j = i:n+1
      sigma = FRACTION(0,1)
      for k = 1:N
        t = F_PUISSANCE(X(k),i+j-2)
        sigma = F_ADDITION(sigma,t)
      end
      S((i-1)*(n+1)+j) = sigma
      if i <> j then
        S((j-1)*(n+1)+i) = sigma
      end
    end
    sigma = FRACTION(0,1)
    for k = 1:N
      t = F_PUISSANCE(X(k),i-1)
      t = F_MULTIPLICATION(t,Y(k))
      sigma = F_ADDITION(sigma,t)
    end
    b(i) = sigma
  end
  a = GAUSS(S,b)
  for i = length(a)-2:-1:1, A($+1) = a(i), end
endfunction
```

EXEMPLE

Reprenons l'exemple précédent et recherchons le polynôme de degré 5 qui passe "au mieux" par les 8 points donnés.

```
X = list(FRACTION(-83,12),FRACTION(-13,3),FRACTION(-29,9),FRACTION(-1,1),...
        FRACTION(2,3),FRACTION(5,1),FRACTION(10,1),FRACTION(101,8),8,1) ;
Y = list(FRACTION(-16,3),FRACTION(34,11),FRACTION(-7,2),FRACTION(2,1),...
        FRACTION(1,4),FRACTION(11,1),FRACTION(-1,5),FRACTION(-2,1),8,1) ;
A = P_AJUSTEMENT(X,Y,5) ;
P_ECRIRE(A)
```

Les coefficients rationnels a_i du polynôme d'ajustement $A(x) = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ sont :

$$a_5 = \frac{1056947924276989893820759845696687398028400 1802}{1791605673935312398484327152155697299732849805 4225}$$

$$a_4 = -\frac{5737827189261505039669357125657903383635201 13199}{7166422695741249593937308608622789198931399221 6900}$$

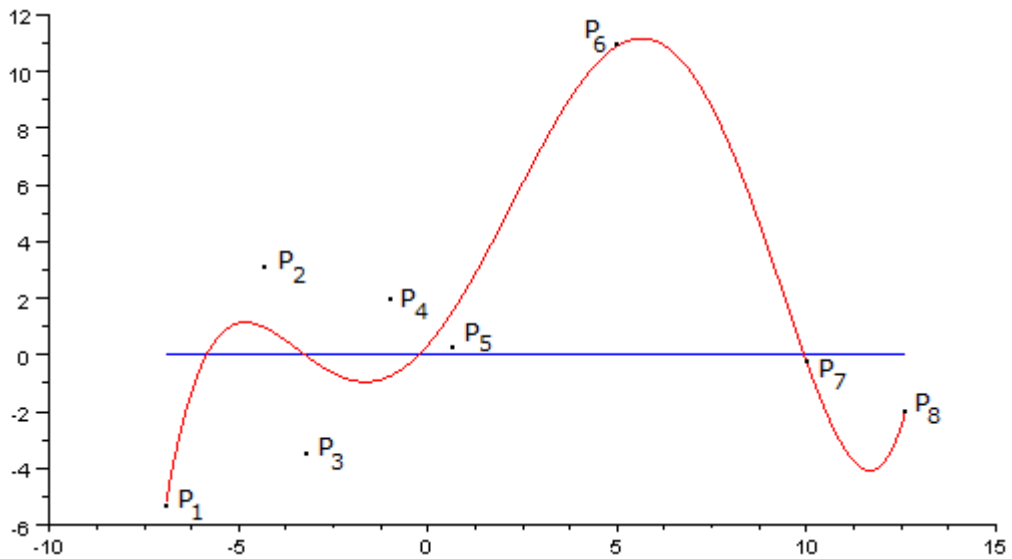
$$a_3 = -\frac{9803865034181235599125173690470521159362061 60683}{2605971889360454397795384948590105163247781535 1600}$$

$$a_2 = \frac{1528292340438634022286026002504137554175747661 2237}{3583211347870624796968654304311394599465699610 8450}$$

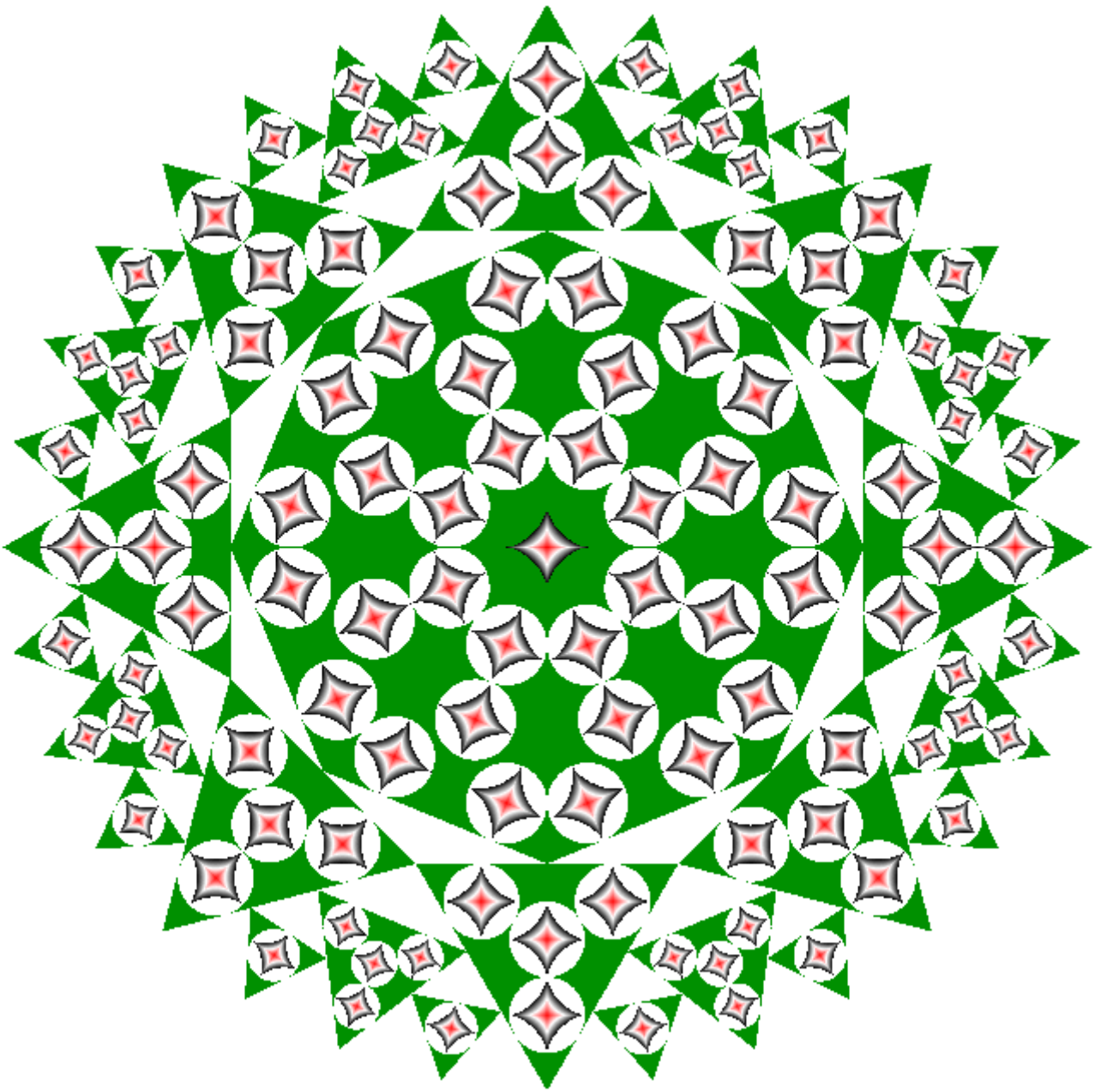
$$a_1 = \frac{4441333858787566824725309055196916679112787419 95911}{2866569078296499837574923443449115679572559688 67600}$$

$$a_0 = \frac{1006966725045530153233471434301660139306145567 5577}{2866569078296499837574923443449115679572559688 6760}$$

Représentation graphique de $A(x)$:



Mathématique & Informatique



Mathématique & Informatique

Racines rationnelles d'un polynôme à coefficients rationnels

Soit $A(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ un polynôme dont les coefficients a_k sont des nombres rationnels. L'équation $A(x) = 0$ peut aussi s'écrire $B(x) = 0$ où $B(x) = b_n x^n + b_{n-1}x^{n-1} + \ldots + b_1 x + b_0$ est un polynôme à coefficients entiers b_k. On suppose $a_n \neq 0$ et $a_0 \neq 0$, donc $b_n \neq 0$ et $b_0 \neq 0$; autrement dit $A(x)$ et $B(x)$ sont des polynômes de degré n qui n'admettent pas la racine triviale $x = 0$. Le cas de la racine nulle sera traité à part en se ramenant à un polynôme de degré inférieur.

$$\begin{aligned}
A(x) &= a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \\
&= \frac{p_n}{q_n}x^n + \frac{p_{n-1}}{q_{n-1}}x^{n-1} + \cdots + \frac{p_1}{q_1}x + \frac{p_0}{q_0} \\
&= \frac{1}{ppcm}\left(\frac{ppcm}{q_n}p_n x^n + \frac{ppcm}{q_{n-1}}p_{n-1}x^{n-1} + \cdots + \frac{ppcm}{q_1}p_1 x + \frac{ppcm}{q_0}p_0\right) \\
&= \frac{1}{ppcm}\left(b_n x^n + b_{n-1}x^{n-1} + \cdots + b_1 x + b_0\right) \\
&= \frac{B(x)}{ppcm}
\end{aligned}$$

avec :

$$\begin{aligned}
B(x) &= b_n x^n + b_{n-1}x^{n-1} + \cdots + b_1 x + b_0 \\
ppcm &= ppcm(q_n, q_{n-1}, \cdots, q_1, q_0) \\
b_k &= a_k \times ppcm \quad (k = 0, \cdots, n)
\end{aligned}$$

Le polynôme $B(x)$ possède les mêmes racines que $A(x)$, on l'appellera polynôme équivalent à $A(x)$.
Pour que $\frac{p}{q}$ irréductible soit racine de $B(x)$, il est nécessaire que p divise b_0 et que q divise b_n puisque :

$$\begin{aligned}
&B\left(\frac{p}{q}\right) = b_n\frac{p^n}{q^n} + b_{n-1}\frac{p^{n-1}}{q^{n-1}} + \cdots + b_1\frac{p}{q} + b_0 = 0 \\
&\Rightarrow b_n p^n + b_{n-1}p^{n-1}q + \cdots + b_1 pq^{n-1} + b_0 q^n = 0 \\
&\Rightarrow p\left(b_n p^{n-1} + \cdots + b_1 q^{n-1}\right) = -b_0 q^n \quad \text{et} \quad q\left(b_{n-1}p^{n-1} + \cdots + b_0 q^{n-1}\right) = -b_n p^n \\
&\Rightarrow p \mid b_0 \quad \text{et} \quad q \mid b_n
\end{aligned}$$

Comme l'ensemble des diviseurs de b_0 et l'ensemble des diviseurs de b_n sont finis, le nombre de racines possibles de $B(x)$ dans $\mathbb{Q}$ est fini.
En utilisant la fonction *diviseurs* décrite antérieurement dans ce document, on obtient l'ensemble D_0 des diviseurs positifs de b_0 et l'ensemble D_n des diviseurs positifs de b_n. Il reste à tester si un couple d'entiers naturels $(p,q) \in D_0 \times D_n$ vérifie $B\left(\pm\frac{p}{q}\right) = 0$, donc $A\left(\pm\frac{p}{q}\right) = 0$.
Pour calculer $B\left(\pm\frac{p}{q}\right)$, il convient d'effectuer les calculs avec la fonction *P_HORNER* du module *Scilab-Polynomes* qui donne la valeur exacte, plutôt que d'utiliser la fonction standard *horner* qui donne une approximation.

L'algorithme élémentaire précédent peut être amélioré.

Pour tout entier relatif α :

$$B\left(\frac{p}{q}\right) = B\left(\frac{p - \alpha q}{q} + \alpha\right) = B(\lambda + \alpha) = \sum_{k=0}^{n} b_k(\lambda + \alpha)^k = \sum_{k=0}^{n} b_k \left(\sum_{i=0}^{k} C_k^i \lambda^i \alpha^{k-i}\right)$$

$$= \sum_{k=0}^{n} \left(\sum_{i=k}^{n} C_i^k b_i \alpha^{i-k}\right) \lambda^k$$

$$= \sum_{k=0}^{n} c_k \lambda^k = c_0 + \sum_{k=1}^{n} c_k \lambda^k$$

$$= \sum_{i=0}^{n} b_i \alpha^i + \sum_{k=1}^{n} \frac{c_k (p - \alpha q)^k q^{n-k}}{q^n}$$

$$= B(\alpha) + \frac{(p - \alpha q)}{q^n} \sum_{k=1}^{n} c_k (p - \alpha q)^{k-1} q^{n-k}$$

D'où :

$$B\left(\frac{p}{q}\right) = 0 \quad \Rightarrow \quad (p - \alpha q) \sum_{k=1}^{n} c_k (p - \alpha q)^{k-1} q^{n-k} = -B(\alpha) q^n$$

Les coefficients $c_k = \sum_{i=k}^{n} C_i^k b_i \alpha^{i-k}$ sont des entiers relatifs. Posons $N = -\sum_{k=1}^{n} c_k (p - \alpha q)^{k-1} q^{n-k} \in \mathbb{Z}$:

$$B\left(\frac{p}{q}\right) = 0 \quad \Rightarrow \quad (p - \alpha q)\, N = B(\alpha) q^n$$

Par hypothèse p et q sont premiers entre eux. Le théorème de **Bachet - Bézout** permet d'écrire :

$\text{pgcd}(p,q) = 1 \Rightarrow \exists u, v:\ up + vq = 1 \Rightarrow u(p - \alpha q) + (\alpha u + v)q = 1 \Rightarrow \exists u, w:\ u(p - \alpha q) + wq = 1 \Rightarrow \text{pgcd}(p - \alpha q, q) = 1$

Les nombres $p - \alpha q$ et q n'ont aucun facteur premier commun ; il en est de même de $p - \alpha q$ et q^n.

Donc $p - \alpha q$ et q^n sont premiers entre eux, c'est - à - dire $\text{pgcd}(p - \alpha q, q^n) = 1$.

Appliquons le théorème de **Gauss** : $p - \alpha q \mid B(\alpha) q^n$ et $\text{pgcd}(p - \alpha q, q^n) = 1 \quad \Rightarrow \quad p - \alpha q \mid B(\alpha)$

Finalement, on obtient la propriété suivante :

$$B\left(\frac{p}{q}\right) = 0 \quad \Rightarrow \quad \forall\ \alpha \in \mathbb{Z} : \quad p - \alpha q \,\Big|\, B(\alpha)$$

En particulier, pour $\alpha = +1$ et $\alpha = -1$, on a :

$$B\left(\frac{p}{q}\right) = 0 \quad \Rightarrow \quad p - q \mid B(+1) \quad \text{et} \quad p + q \mid B(-1)$$

Ainsi $p - q$ divise $B(+1)$ et $p + q$ divise $B(-1)$ lorsque $\frac{p}{q}$ irréductible est racine de $B(x)$.

D'une manière générale, notons *D(e)* l'ensemble des diviseurs positifs d'un entier relatif *e*.
Si la fraction irréductible $\frac{p}{q}$ est une racine du polynôme $B(x) = b_n x^n + b_{n-1}x^{n-1} + \ldots + b_1 x + b_0 \in \mathbb{Z}(x)$, alors :

$|p| \in D(b_0)$
$|q| \in D(b_n)$
$|p - q| \in D(B(+1))$
$|p + q| \in D(B(-1))$

Les calculs dans $\mathbb{Q}$ sont beaucoup moins rapides que dans $\mathbb{Z}$ car les fractions sont systématiquement mises sous leur forme irréductible lors des opérations arithmétiques (recherche du pgcd entre le numérateur et le dénominateur avec l'algorithme d'**Euclide**). C'est pourquoi on décide de restreindre le nombre de tests $B\left(\pm \frac{p}{q}\right) = 0$. Pour ce faire, on prend les 2 ensembles D_1 et D_2 qui possèdent les plus petits cardinaux parmi $D(b_0)$, $D(b_n)$, $D(B(+1))$, $D(B(-1))$ et on applique les critères de divisibilité par p, q, $p - q$, $p + q$. Les 6 cas à envisager pour le produit cartésien $D_1 \times D_2$ sont :

① $\underline{D(b_0) \times D(b_n)}$

$a \in D(b_0), b \in D(b_n) \quad \Rightarrow \quad x_1 = \frac{p_1}{q_1} = \frac{a}{b}$, $x_2 = \frac{p_2}{q_2} = \frac{-a}{b}$

x_1 ou x_2 racine de $B(x) \quad \Rightarrow \quad p_1 - q_1 \mid B(+1)$ et $p_1 + q_1 \mid B(-1)$, ou $p_2 - q_2 \mid B(+1)$ et $p_2 + q_2 \mid B(-1)$

② $\underline{D(b_0) \times D(B(+1))}$

$a \in D(b_0), b \in D(B(+1)) \quad \Rightarrow \quad x_1 = \frac{p_1}{q_1} = \frac{a}{a-b}$, $x_2 = \frac{p_2}{q_2} = \frac{a}{a+b}$

x_1 ou x_2 racine de $B(x) \quad \Rightarrow \quad q_1 \mid b_n$ et $p_1 + q_1 \mid B(-1)$, ou $q_2 \mid b_n$ et $p_2 + q_2 \mid B(-1)$

③ $\underline{D(b_0) \times D(B(-1))}$

$a \in D(b_0), b \in D(B(-1)) \quad \Rightarrow \quad x_1 = \frac{p_1}{q_1} = \frac{a}{b-a}$, $x_2 = \frac{p_2}{q_2} = \frac{-a}{a+b}$

x_1 ou x_2 racine de $B(x) \quad \Rightarrow \quad q_1 \mid b_n$ et $p_1 - q_1 \mid B(+1)$, ou $q_2 \mid b_n$ et $p_2 - q_2 \mid B(+1)$

④ $\underline{D(b_n) \times D(B(+1))}$

$a \in D(b_n), b \in D(B(+1)) \quad \Rightarrow \quad x_1 = \frac{p_1}{q_1} = \frac{a+b}{a}$, $x_2 = \frac{p_2}{q_2} = \frac{a-b}{a}$

x_1 ou x_2 racine de $B(x) \quad \Rightarrow \quad p_1 \mid b_0$ et $p_1 + q_1 \mid B(-1)$, ou $p_2 \mid b_0$ et $p_2 + q_2 \mid B(-1)$

⑤ $\underline{D(b_n) \times D(B(-1))}$

$a \in D(b_n), b \in D(B(-1)) \quad \Rightarrow \quad x_1 = \frac{p_1}{q_1} = \frac{b-a}{a}$, $x_2 = \frac{p_2}{q_2} = \frac{a+b}{-a}$

x_1 ou x_2 racine de $B(x) \quad \Rightarrow \quad p_1 \mid b_0$ et $p_1 - q_1 \mid B(+1)$, ou $p_2 \mid b_0$ et $p_2 - q_2 \mid B(+1)$

⑥ $\underline{D(B(+1)) \times D(B(-1))}$

$a \in D(B(+1)), b \in D(B(-1)) \quad \Rightarrow \quad x_1 = \frac{p_1}{q_1} = \frac{\frac{a+b}{2}}{\frac{b-a}{2}}$, $x_2 = \frac{p_2}{q_2} = \frac{\frac{b-a}{2}}{\frac{a+b}{2}}$

x_1 ou x_2 racine de $B(x) \quad \Rightarrow \quad p_1 \mid b_0$ et $q_1 \mid b_n$, ou $p_2 \mid b_0$ et $q_2 \mid b_n$ (si a et b sont de même parité)

Après avoir constitué la liste (sans doublons) des rationnels $|x_k| = \left|\frac{p_k}{q_k}\right|$ satisfaisant aux conditions précédentes, il suffit d'effectuer les tests $B(\pm|x_k|) = 0$ pour trouver toutes les racines rationnelles de *B(x)*.
Ajoutons que la méthode suppose que *0*, *+1* et *-1* ne soient pas des racines de *B(x)*, donc de *A(x)*. Autrement dit, il est nécessaire que $A(0) \neq 0$, $A(+1) \neq 0$, $A(-1) \neq 0$. Dans le cas contraire, on divisera *A(x)* par x^m , ou $(x - 1)^m$, ou $(x + 1)^m$, l'entier *m* désignant l'ordre de multiplicité de la racine *0*, ou *+1*, ou *-1*, et on prendra pour *B(x)* le polynôme équivalent à ce nouveau polynôme dont les racines diffèrent de *0*, *+1*, *-1*.

Mathématique & Informatique

Exemple 1

Soit le polynôme de degré 6 :

$$A(x) = \frac{35}{3}x^6 - \frac{65443}{221}x^5 + \frac{1562679}{884}x^4 + \frac{4361749}{2652}x^3 - \frac{2424203}{884}x^2 - \frac{14595}{68}x - \frac{8723}{51}$$

En $x = 0, +1, -1$ le polynôme $A(x)$ vaut respectivement : $A(0) = -\frac{8723}{51}$, $A(+1) = 0$, $A(-1) = -\frac{1503614}{663}$

On établit que 1 est racine simple de $A(x)$ et on considère le polynôme $\frac{A(x)}{x-1}$ dont les racines diffèrent de 0, +1, -1.

$$\frac{A(x)}{x-1} = \frac{35}{3}x^5 - \frac{188594}{663}x^4 + \frac{3933661}{2652}x^3 + \frac{4147705}{1326}x^2 + \frac{78677}{204}x + \frac{8723}{51}$$

Le polynôme $B(x)$ à coefficients entiers qui admet les mêmes racines que $\frac{A(x)}{x-1}$ est :

$$B(x) = 30940x^5 - 754376x^4 + 3933661x^3 + 8295410x^2 + 1022801x + 453596$$

On a : $b_0 = 453596$, $b_n = 30940$, $B(+1) = 12982032$, $B(-1) = 3007228$.
Les ensembles de diviseurs $D(b_0)$, $D(b_n)$, $D(B(+1))$, $D(B(-1))$ se calculent avec la fonction *diviseurs* :

$D(b_0) = D(453596) = 1, 2, 4, 11, 13, \ldots, 34892, 41236, 113399, 226798, 453596$ (36 entiers)
$D(b_n) = D(30940) = 1, 2, 4, 5, 7, \ldots, 4420, 6188, 7735, 15470, 30940$ (48 entiers)
$D(B(+1)) = D(12982032) = 1, 2, 3, 4, 6, \ldots, 2163672, 3245508, 4327344, 6491016, 12982032$ (160 entiers)
$D(B(-1)) = D(3007228) = 1, 2, 4, 7, 14, \ldots, 214802, 429604, 751807, 1503614, 3007228$ (36 entiers)

Les plus petits cardinaux de ces 4 ensembles sont ceux de $D(b_0)$ et $D(B(-1))$. Par suite, plaçons nous dans le cas n° 3.
Toute racine rationnelle $\frac{p}{q}$ de $B(x)$ vérifie : $|p| \in D(453596)$ et $|p+q| \in D(3007228)$.
Le produit cartésien $D(453596) \times D(3007228)$ conduit à étudier $36 \times 36 \times 2 = 2592$ couples (p,q).
Avec la condition supplémentaire $q \mid 30940$ et $p - q \mid 12982032$, le nombre de rationnels $\left|\frac{p}{q}\right|$ possibles est ramené à 16 :

$$\left|\frac{p}{q}\right| = \frac{1}{2}, \frac{1}{5}, \frac{1}{13}, \frac{2}{1}, \frac{2}{5}, \frac{2}{65}, \frac{4}{5}, \frac{11}{7}, \frac{11}{10}, \frac{11}{13}, \frac{11}{17}, \frac{13}{1}, \frac{13}{14}, \frac{13}{85}, \frac{61}{65}, \frac{671}{442}$$

On effectue le test $B\left(\pm\left|\frac{p}{q}\right|\right) = 0$ pour chacune des valeurs précédentes et on obtient les racines 13 et $-\frac{671}{442}$
Donc, en rajoutant la racine 1 écartée au départ, les racines rationnelles du polynôme $A(x)$ sont $1,\ 13,\ -\frac{671}{442}$

Note La racine 13 a un ordre de multiplicité égal à 2, d'où : $A(x) = (x-1)(x-13)^2(x+\frac{671}{442})(\frac{35}{3}x^2 + \frac{7}{6}x + \frac{2}{3})$

Exemple 2

Soit le polynôme de degré 4 :

$$A(x) = x^4 - \frac{704}{299}x^3 - \frac{4119801067}{436792356}x^2 + \frac{240880424}{764386623}x - \frac{166270831945}{940783536}$$

On vérifie que $A(x)$ ne s'annule pas en 0, +1, -1. Le polynôme équivalent à $A(x)$ est :

$$B(x) = 12230185968\,x^4 - 28796156928\,x^3 - 115354429876\,x^2 + 3854086784\,x - 2161520815285$$

La fonction *diviseurs* fournit les ensembles de diviseurs positifs de b_0, b_n, $B(+1)$, $B(-1)$:

$D(b_0) = D(-2161520815285) = 1, 5, 13, \ldots, 432304163057, 2161520815285$ = ensemble de 64 entiers
$D(b_n) = D(12230185968) = 1, 2, 3, \ldots, 6115092984, 12230185968$ = ensemble de 1440 entiers
$D(B(+1)) = D(-2289587129337) = 1, 3, 71, 213, 10749235349, 32247706047, 763195709779, 2289587129337$
$D(B(-1)) = D(-2239702989049) = 1, 59, 37961067611, 2239702989049$

Les 2 ensembles qui ont le moins d'éléments sont $D(B(+1))$ et $D(B(-1))$: $Card(D(B(+1))) = 8$ et $Card(D(B(-1))) = 4$
On est donc dans le cas n° 6 avec $8 \times 4 \times 2 = 64$ couples (p,q) à examiner.
La condition $p \mid b_0$ et $q \mid b_n$ ramène à 2 le nombre de rationnels $\left|\frac{p}{q}\right|$ possibles : $\left|\frac{p}{q}\right| = \frac{1}{2}$ ou $\left|\frac{p}{q}\right| = \frac{65}{6}$
Comme $B\left(\pm\frac{1}{2}\right) \neq 0$ et $B\left(\pm\frac{65}{6}\right) \neq 0$, on conclut que le polynôme $A(x)$ n'admet pas de racines rationnelles.

La fonction de recherche des racines rationnelles d'un polynôme *A(x)* à coefficients rationnels est :

```
function R = P_RACINES(A)
    R = list()
    n = P_DEGRE(A)
    if n == [] then, return, end
    // Cas des racines 0, +1, -1
    deg = 0
    for r = [0,+1,-1]
        [A,m] = P_MULTIPLICITE(A,FRACTION(r,1))
        if m > 0 then, R($+1) = FRACTION(r,1), end
        deg = deg + m
        if deg == n then, return, end
    end
    // Calcul du polynôme équivalent B
    B = P_EQUIVALENT(A)
    for i = 1:length(B)
        B(i) = list(NUMERATEUR(B(i)),ENTIER(1))
    end
    // Nombres entiers b0, bn, B(+1), B(-1)
    n1 = NUMERATEUR(B($))
    n2 = NUMERATEUR(B(1))
    n3 = NUMERATEUR(P_HORNER(B,FRACTION(+1,1)))
    n4 = NUMERATEUR(P_HORNER(B,FRACTION(-1,1)))
    // Ensembles de diviseurs D(b0), D(bn), D(B(+1)), D(B(-1))
    E1 = diviseurs(abs(nombre_relatif(n1)))
    E2 = diviseurs(abs(nombre_relatif(n2)))
    E3 = diviseurs(abs(nombre_relatif(n3)))
    E4 = diviseurs(abs(nombre_relatif(n4)))
    // Tri des cardinaux
    T = [1,length(E1) ; 2,length(E2) ; 3,length(E3) ; 4,length(E4)]
    j = 1
    while j < 4
        i = j
        while i >= 1
            if T(i,2) > T(i+1,2) then
                t1 = T(i+1,1)
                t2 = T(i+1,2)
                T(i+1,1) = T(i,1)
                T(i+1,2) = T(i,2)
                T(i,1) = t1
                T(i,2) = t2
                i = i - 1
            else
                i = 0
            end
        end
        j = j + 1
    end
    if (T(1,1) == 1 & T(2,1) == 2) | (T(1,1) == 2 & T(2,1) == 1) then
        cas = 1 ; D1 = E1 ; D2 = E2
    elseif (T(1,1) == 1 & T(2,1) == 3) | (T(1,1) == 3 & T(2,1) == 1) then
        cas = 2 ; D1 = E1 ; D2 = E3
    elseif (T(1,1) == 1 & T(2,1) == 4) | (T(1,1) == 4 & T(2,1) == 1) then
        cas = 3 ; D1 = E1 ; D2 = E4
    elseif (T(1,1) == 2 & T(2,1) == 3) | (T(1,1) == 3 & T(2,1) == 2) then
        cas = 4 ; D1 = E2 ; D2 = E3
    elseif (T(1,1) == 2 & T(2,1) == 4) | (T(1,1) == 4 & T(2,1) == 2) then
        cas = 5 ; D1 = E2 ; D2 = E4
    elseif (T(1,1) == 3 & T(2,1) == 4) | (T(1,1) == 4 & T(2,1) == 3) then
        cas = 6 ; D1 = E3 ; D2 = E4
    end
```

```
// Liste L des rationnels possibles
L = list()
for i = 1:length(D1)
    for j = 1:length(D2)
        a = D1(i)
        b = D2(j)
        select cas
        case 1
            f1 = FRACTION(a,b)
            x1 = F_REDUCTION(f1)
            p1 = NUMERATEUR(x1)
            q1 = DENOMINATEUR(x1)
            f2 = FRACTION(-a,b)
            x2 = F_REDUCTION(f2)
            p2 = NUMERATEUR(x2)
            q2 = DENOMINATEUR(x2)
            [Q11,R11] = DIVISION(n3,SOUSTRACTION(p1,q1))
            [Q12,R12] = DIVISION(n4,ADDITION(p1,q1))
            [Q21,R21] = DIVISION(n3,SOUSTRACTION(p2,q2))
            [Q22,R22] = DIVISION(n4,ADDITION(p2,q2))
        case 2
            f1 = FRACTION(a,(a-b))
            x1 = F_REDUCTION(f1)
            p1 = NUMERATEUR(x1)
            q1 = DENOMINATEUR(x1)
            f2 = FRACTION(a,(a+b))
            x2 = F_REDUCTION(f2)
            p2 = NUMERATEUR(x2)
            q2 = DENOMINATEUR(x2)
            [Q11,R11] = DIVISION(n2,q1)
            [Q12,R12] = DIVISION(n4,ADDITION(p1,q1))
            [Q21,R21] = DIVISION(n2,q2)
            [Q22,R22] = DIVISION(n4,ADDITION(p2,q2))
        case 3
            f1 = FRACTION(a,(b-a))
            x1 = F_REDUCTION(f1)
            p1 = NUMERATEUR(x1)
            q1 = DENOMINATEUR(x1)
            f2 = FRACTION(-a,(a+b))
            x2 = F_REDUCTION(f2)
            p2 = NUMERATEUR(x2)
            q2 = DENOMINATEUR(x2)
            [Q11,R11] = DIVISION(n2,q1)
            [Q12,R12] = DIVISION(n3,SOUSTRACTION(p1,q1))
            [Q21,R21] = DIVISION(n2,q2)
            [Q22,R22] = DIVISION(n3,SOUSTRACTION(p2,q2))
        case 4
            f1 = FRACTION((a+b),a)
            x1 = F_REDUCTION(f1)
            p1 = NUMERATEUR(x1)
            q1 = DENOMINATEUR(x1)
            f2 = FRACTION((a-b),a)
            x2 = F_REDUCTION(f2)
            p2 = NUMERATEUR(x2)
            q2 = DENOMINATEUR(x2)
            [Q11,R11] = DIVISION(n1,p1)
            [Q12,R12] = DIVISION(n4,ADDITION(p1,q1))
            [Q21,R21] = DIVISION(n1,p2)
            [Q22,R22] = DIVISION(n4,ADDITION(p2,q2))
```

```
            case 5
                f1 = FRACTION((b-a),a)
                x1 = F_REDUCTION(f1)
                p1 = NUMERATEUR(x1)
                q1 = DENOMINATEUR(x1)
                f2 = FRACTION((a+b),-a)
                x2 = F_REDUCTION(f2)
                p2 = NUMERATEUR(x2)
                q2 = DENOMINATEUR(x2)
                [Q11,R11] = DIVISION(n1,p1)
                [Q12,R12] = DIVISION(n3,SOUSTRACTION(p1,q1))
                [Q21,R21] = DIVISION(n1,p2)
                [Q22,R22] = DIVISION(n3,SOUSTRACTION(p2,q2))
            case 6
                if modulo(a,2) == modulo(b,2) then
                    f1 = FRACTION((a+b)/2,(b-a)/2)
                    x1 = F_REDUCTION(f1)
                    p1 = NUMERATEUR(x1)
                    q1 = DENOMINATEUR(x1)
                    f2 = FRACTION((b-a)/2,(a+b)/2)
                    x2 = F_REDUCTION(f2)
                    p2 = NUMERATEUR(x2)
                    q2 = DENOMINATEUR(x2)
                    [Q11,R11] = DIVISION(n1,p1)
                    [Q12,R12] = DIVISION(n2,q1)
                    [Q21,R21] = DIVISION(n1,p2)
                    [Q22,R22] = DIVISION(n2,q2)
                else
                    R11 = [] ; R12 = [] ; R21 = [] ; R22 = []
                end
            end
            if NUL(R11) & NUL(R12) then
                doublon = %f
                for k = 1:length(L)
                    doublon = isequal(L(k),F_ABSOLU(x1))
                    if doublon then, break, end
                end
                if ~doublon then, L($+1) = F_ABSOLU(x1), end
            end
            if NUL(R21) & NUL(R22) then
                doublon = %f
                for k = 1:length(L)
                    doublon = isequal(L(k),F_ABSOLU(x2))
                    if doublon then, break, end
                end
                if ~doublon then, L($+1) = F_ABSOLU(x2), end
            end
        end
    end
    // Liste R des racines
    for k = 1:length(L)
        x1 = L(k)
        x2 = F_MOINS(L(k))
        y1 = P_HORNER(B,x1)
        y2 = P_HORNER(B,x2)
        if F_NUL(y1) then
            R($+1) = x1
            deg = deg + 1
        end
        if F_NUL(y2) then
            R($+1) = x2
            deg = deg + 1
        end
        if deg == n then, return, end
    end
endfunction
```

Avant d'utiliser la fonction *P_RACINES*, il est nécessaire de charger en mémoire le fichier des nombres premiers avec la commande *exec("Scilab-Premiers.sce",-1)* qui exécute les instructions suivantes :

```
global P
stacksize(2e7) ;
fd = mopen("PREMIERS",'rb') ;
P = mget(1e7) ;
mclose(fd) ;
```

La fonction *P_RACINES* calcule toutes les racines rationnelles d'un polynôme $A(x)$ à coefficients rationnels, mais elle ne fournit pas leur ordre de multiplicité. Pour trouver l'ordre de multiplicité m d'une racine rationnelle r, c'est-à-dire le plus grand entier m tel que $A(x) = (x - r)^m Q(x)$, on divise $A(x)$ par $x - r$ de manière itérative tant que le reste obtenu est nul. On a $m > 0$ si r est racine de $A(x)$ et $m = 0$ sinon. D'où la fonction *P_MULTIPLICITE* :

```
function [Q,m] = P_MULTIPLICITE(A,r)
    m = 0
    R = list(FRACTION(0,1))
    B = list(FRACTION(1,1),F_MOINS(r))
    while P_NUL(R)
        [Q,R] = P_DIVISION(A,B)
        if P_NUL(R) then
            m = m + 1
            A = Q
        else
            Q = A
        end
    end
endfunction
```

Mathématique & Informatique

EXEMPLE 1

Trouver les racines rationnelles (si elles existent) du polynôme suivant :

$$A(x) = \frac{6435}{128}x^9 - \frac{148005}{6657536}x^8 - \frac{3003}{32}x^7 + \frac{69069}{1664384}x^6 + \frac{3465}{64}x^5 - \frac{79695}{3328768}x^4 - \frac{315}{32}x^3 + \frac{7245}{1664384}x^2 + \frac{35}{128}x - \frac{805}{6657536}$$

Solution

```
exec("Scilab-Premiers.sce",-1)
exec("Scilab-Equations.sce",-1)
A = list(FRACTION(6435,128),FRACTION(- 148005,6657536),FRACTION(- 3003,32),FRACTION(69069,1664384), ...
         FRACTION(3465,64),FRACTION(- 79695,3328768),FRACTION(- 315,32),FRACTION(7245,1664384), ...
         FRACTION(35,128),FRACTION(- 805,6657536)) ;
R = P_RACINES(A) ;
// Ecriture du résultat
if length(R) == 0 then
    printf('\n Le polynôme ne possède pas de racine rationnelle')
else
    printf('\n Les racines rationnelles du polynôme sont :')
    for i = 1:length(R), printf('\n'), ECRIRE(R(i)), end
end
```

On obtient une seule racine rationnelle : $x = \frac{23}{52012}$

En fait, $A(x) = \left(x - \frac{23}{52012}\right) P_8(x)$ où $P_8(x)$ est le polynôme de **Legendre** d'ordre 8.

```
U = list(FRACTION(1,1),FRACTION(-23,52012)) ;
V = P_LEGENDRE(8) ;
A = P_MULTIPLICATION(U,V) ;
P_ECRIRE(A)
```

Le résultat précédent montre que le polynôme de **Legendre** d'ordre 8 n'a pas de racine rationnelle.

Remarque

Avec la fonction Scilab *roots* :

```
C = [-805/6657536,35/128,7245/1664384,-315/32,-79695/3328768, ...
     3465/64,69069/1664384,-3003/32,-148005/6657536,6435/128] ;
A = poly(C,'x','coeff') ;
R = roots(A) ;
for i = 1:length(R), printf('\n %.10f',R(i)), end
```

On obtient les 9 racines réelles de *A(x)* :

R_1 = 0.9602898565	R_2 = 0.7966664774	R_3 = - 0.9602898565
R_4 = - 0.7966664774	R_5 = - 0.5255324099	R_6 = 0.5255324099
R_7 = - 0.1834346425	R_8 = 0.1834346425	R_9 = 0.0004422056

La racine R_9 = 0.0004422056 est une bonne approximation de $\frac{23}{52012}$, mais on ne sait pas si R_9 est rationnelle ; a fortiori, on ne connait pas son représentant irréductible $\frac{p}{q}$. On ignore également que R_1 à R_8 sont irrationnelles.

EXEMPLE 2

Factoriser dans $\mathbb{Q}(x)$ le polynôme défini par :

$$A(x) = \frac{9}{2}x^7 + \frac{433269}{28168}x^6 + \frac{1464398}{73941}x^5 + \frac{362677957}{22083712}x^4 + \frac{195999511}{12619264}x^3 + \frac{51198265}{4732224}x^2 + \frac{77251}{28168}x - \frac{15}{503}$$

Solution

Pour un polynôme *A(x)* quelconque de $\mathbb{Q}(x)$, la fonction *P_FACTORISATION* ci-dessous affiche à l'écran les facteurs recherchés avec leur ordre de multiplicité :

```
function P_FACTORISATION(A)
    P_ECRIRE(A)
    R = P_RACINES(A)
    for i = 1:length(R)
        [A,m] = P_MULTIPLICITE(A,R(i))
        printf('\n\n Racine rationnelle '), ECRIRE(R(i)), printf(' ordre %i',m)
        P_ECRIRE(list(FRACTION(1,1),F_MOINS(R(i))))
    end
    printf('\n\n Racine(s) non rationnelle(s)')
    P_ECRIRE(A)
endfunction
```

Pour le polynôme *A(x)* de l'exemple donné, on écrit :

```
A = list(FRACTION(9,2), FRACTION(433269,28168),FRACTION(1464398,73941), ...
         FRACTION(362677957,22083712),FRACTION(195999511,12619264), ...
         FRACTION(51198265,4732224),FRACTION(77251,28168), FRACTION(- 15,503)) ;
P_FACTORISATION(A)
```

On obtient :

```
Racine rationnelle - 8/7 ordre 3
Polynôme de degré 1 :
================
1   x
8/7

Racine rationnelle 21/2012 ordre 1
Polynôme de degré 1 :
================
1   x
- 21/2012

Racine(s) non rationnelle(s)
Polynôme de degré 3 :
================
9/2   x3
7/3   x
245/128
```

D'où :

$$A(x) = \left(x + \frac{8}{7}\right)^3 \left(x - \frac{21}{2012}\right)\left(\frac{9}{2}x^3 + \frac{7}{3}x + \frac{245}{128}\right)$$

EXEMPLE 3

Montrer que les nombres $\sqrt[3]{2012 + 671\sqrt{5365}}$ et $\sqrt[3]{2012 - 671\sqrt{5365}}$ sont irrationnels, mais que le nombre $\sqrt[3]{2012 + 671\sqrt{5365}} + \sqrt[3]{2012 - 671\sqrt{5365}}$ est rationnel.

Solution

Ecrivons : $671\sqrt{5365} = \sqrt{2415542965}$
Posons : $a = 2012$, $b = 2415542965$, $u = \sqrt[3]{a + \sqrt{b}}$, $v = \sqrt[3]{a - \sqrt{b}}$

- D'une part :

$u^3 = a + \sqrt{b}$ et $v^3 = a - \sqrt{b} \Rightarrow u^6 - 2au^3 + (a^2 - b) = 0$ et $v^6 - 2av^3 + (a^2 - b) = 0$

On a : $-2a = -4024$
et : $a^2 - b = -2411494821$

Recherchons si le polynôme $A(x) = x^6 - 4024x^3 - 2411494821$ possède des racines rationnelles :

```
A = list(FRACTION(1,1),FRACTION(0,1), FRACTION(0,1), FRACTION(-4024,1), ...
        FRACTION(0,1),FRACTION(0,1), FRACTION(-2411494821,1)) ;
R = P_RACINES(A) ;
AFFICHE(R)
```

On trouve : R = ∅ $\Rightarrow$ $A(x)$ ne possède pas de racines rationnelles $\Rightarrow$ *u* et *v* sont irrationnels
Donc :

$\sqrt[3]{2012 + 671\sqrt{5365}}$ et $\sqrt[3]{2012 - 671\sqrt{5365}}$ sont des nombres irrationnels

- D'autre part :

$(u + v)^3 = u^3 + 3u^2v + 3uv^2 + v^3 = u^3 + 3uv(u + v) + v^3$

Posons : $x = u + v$, il vient : $x^3 - 3uvx - (u^3 + v^3) = 0$

On a : $-3uv = -3\sqrt[3]{(a + \sqrt{b})(a - \sqrt{b})} = -3\sqrt[3]{a^2 - b} = 4023$
et : $-(u^3 + v^3) = -(a + \sqrt{b} + a - \sqrt{b}) = -2a = -4024$

Il est évident que l'équation $x^3 + 4023x - 4024 = 0$ admet la racine $x = 1$ et que c'est la seule racine réelle puisque le discriminant $\Delta = 4(4023)^3 + 27(-4024)^2 = 260878640220$ est strictement positif.
Donc :

$$\sqrt[3]{2012 + 671\sqrt{5365}} + \sqrt[3]{2012 - 671\sqrt{5365}} = 1$$

La somme de 2 nombres rationnels est un nombre rationnel, par contre la somme de 2 nombres irrationnels n'est pas toujours un nombre irrationnel.

EXEMPLE 4

Rechercher les valeurs propres rationnelles des matrices *A* et *B* suivantes :

$$A = \begin{pmatrix} 4 & 7 & -\frac{5}{2} & \frac{1}{9} \\ \frac{21}{11} & -\frac{31}{13} & 10 & \frac{2}{17} \\ \frac{15}{14} & -1 & 0 & -\frac{5}{2} \\ \frac{56}{31} & \frac{19}{4} & \frac{1}{7} & \frac{17}{23} \end{pmatrix}$$

$$B = \begin{pmatrix} -\frac{2098741090}{30107} & \frac{25200938680}{30107} & -\frac{63026415015}{30107} & \frac{6004043860}{4301} \\ -\frac{5040187736}{90321} & \frac{60518355347}{90321} & -\frac{50450013990}{30107} & \frac{14417728030}{12903} \\ -\frac{4201761001}{90321} & \frac{16816671330}{30107} & -\frac{84112238543}{60214} & \frac{12018784730}{12903} \\ -\frac{1200808772}{30107} & \frac{14417728030}{30107} & -\frac{36056354190}{30107} & \frac{3434702564}{4301} \end{pmatrix}$$

Solution

Pour *A* :

```
A = list(FRACTION(4,1),FRACTION(7,1),FRACTION(-5,2),FRACTION(1,9), ...
    FRACTION(21,11),FRACTION(-31,13),FRACTION(10,1),FRACTION(2,17), ...
    FRACTION(15,14),FRACTION(-1,1),FRACTION(0,1),FRACTION(-5,2), ...
    FRACTION(56,31),FRACTION(19,4),FRACTION(1,7),FRACTION(17,23), ...
    4,4) ;
PA = M_POLYNOME(A) ;
RA = P_RACINES(PA) ;
AFFICHE(RA)
```

On trouve $RA = \varnothing$, ce qui signifie que le polynôme caractéristique de *A* ne possède pas de racines dans $\mathbb{Q}$, et donc que les valeurs propres de *A* sont irrationnelles ou complexes.

Pour *B* :

```
B = list(FRACTION(- 2098741090,30107), FRACTION(25200938680,30107),FRACTION(- 63026415015,30107), FRACTION(6004043860,4301), ...
    FRACTION(- 5040187736,90321), FRACTION(60518355347,90321), FRACTION(- 50450013990,30107), FRACTION(14417728030,12903), ...
    FRACTION(- 4201761001,90321), FRACTION(16816671330,30107), FRACTION(- 84112238543,60214), FRACTION(12018784730,12903), ...
    FRACTION(- 1200808772,30107), FRACTION(14417728030,30107),FRACTION(- 36056354190,30107), FRACTION(3434702564,4301), ...
    4,4) ;
PB = M_POLYNOME(B) ;
RB = P_RACINES(PB) ;
AFFICHE(RB)
```

On obtient l'expression exacte des valeurs propres de *B*, à savoir : $\lambda_1 = 0$, $\lambda_2 = -\frac{1}{10626}$, $\lambda_3 = \frac{151}{17}$, $\lambda_4 = 2012$

EXEMPLE 5

Résoudre dans $\mathbb{Q}$:

$$\begin{cases} a + b + c + d = -\dfrac{6931553}{58754} \\ a^2 + b^2 + c^2 + d^2 = \dfrac{55462141127121}{3452032516} \\ a^3 + b^3 + c^3 + d^3 = -\dfrac{410462989705137847157}{202820718445064} \\ a^4 + b^4 + c^4 + d^4 = \dfrac{3051538155161915106551169393}{11916528491521290256} \end{cases}$$

Solution

On étend à $\mathbb{Q}$ la fonction *resolution* définie antérieurement sur $\mathbb{Z}$:

```
function X = resolution(S)
    P = list()
    n = length(S)
    for p = 1:n
        if p == 1 then, P($+1) = FRACTION(1,1), end
        a = FRACTION(0,1)
        for k = 1:p-1
            a = F_ADDITION(a,F_MULTIPLICATION(P(k+1),S(p-k)))
        end
        P($+1) = F_MULTIPLICATION(F_ADDITION(a,S(p)),FRACTION(-1,p))
    end
    X = P_RACINES(P)
endfunction
```

Avec les données fournies, le programme d'appel est :

```
S1 = list([-1,6,9,3,1,5,5,3],[1,5,8,7,5,4]) ;
S2 = list([1,5,5,4,6,2,1,4,1,1,2,7,1,2,1],[1,3,4,5,2,0,3,2,5,1,6]) ;
S3 = list([-1,4,1,0,4,6,2,9,8,9,7,0,5,1,3,7,8,4,7,1,5,7],[1,2,0,2,8,2,0,7,1,8,4,4,5,0,6,4]) ;
S4 = list([1,3,0,5,1,5,3,8,1,5,5,1,6,1,9,1,5,1,0,6,5,5,1,1,6,9,3,9,3],[1,1,1,9,1,6,5,2,8,4,9,1,5,2,1,2,9,0,2,5,6]) ;
S = list(S1,S2,S3,S4) ;
// Résolution
X = resolution(S) ;
// Ecriture du résultat
for i = 1:length(X), printf('\n Nombre n° %i = ',i), ECRIRE(X(i)), end
```

L'exécution donne la solution unique :

$$a = \frac{7}{1013}$$
$$b = 8$$
$$c = \frac{15}{29}$$
$$d = -\frac{253}{2}$$

Mathématique & Informatique

Polynômes matriciels

Soit $P(x) = a_p x^p + \cdots + a_1 x + a_0$ un polynôme de degré p à coefficients dans $\mathbb{Q}$ et $A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$ une matrice carrée d'ordre n à éléments dans $\mathbb{Q}$. On définit la matrice $P(A)$ par :

$$P(A) = a_p A^p + \cdots + a_1 A + a_0 I$$

où I est la matrice identité d'ordre n.

En particulier, si l'on note $PC(x)$ le polynôme caractéristique de A, on a :

$PC(A) = 0$ (théorème de **Cayley-Hamilton**)

L'exponentielle d'une matrice carrée A est définie par la série :

$$e^A = I + A + \frac{A^2}{2!} + \cdots + \frac{A^k}{k!} + \cdots$$

Ecrivons une fonction Scilab qui calcule la matrice $B = P(A)$ lorsqu'on fournit le polynôme P et la matrice A :

```
function B = P_MATRICE(P,A)
    m = length(P)
    n = A($)
    M = M_IDENTITE(n)
    B = M_SCALAIRE(M,P(m))
    for i = 1:m-1
        M = M_MULTIPLICATION(M,A)
        B = M_ADDITION(B,M_SCALAIRE(M,P(m-i)))
    end
endfunction
```

Exemple

Pour calculer $P(A)$ où $P(x) = \frac{5}{7}x^5 + \frac{11}{8}x^2 - \frac{2}{3}x + \frac{21}{13}$ et $A = \begin{pmatrix} -\frac{1}{2} & \frac{2}{3} & \frac{3}{4} \\ \frac{4}{5} & -\frac{5}{6} & \frac{6}{7} \\ \frac{7}{8} & \frac{8}{9} & -\frac{9}{10} \end{pmatrix}$, on écrit :

```
exec("Scilab-Matrices.sce",-1)
P = list(FRACTION(5,7),FRACTION(0,1),FRACTION(0,1),...
        FRACTION(11,8),FRACTION(-2,3),FRACTION(21,13)) ;
A = list(FRACTION(-1,2),FRACTION(2,3),FRACTION(3,4),...
        FRACTION(4,5),FRACTION(-5,6),FRACTION(6,7),...
        FRACTION(7,8),FRACTION(8,9),FRACTION(-9,10),...
        3,3) ;
PA = P_MATRICE(P,A) ;
M_ECRIRE(PA)
```

On trouve :

$$P(A) = \begin{pmatrix} \frac{1018329769}{528353280} & -\frac{449298523}{1066867200} & \frac{4445011001}{4741632000} \\ -\frac{2185529}{11113200} & -\frac{992387621}{1733659200} & \frac{13123054753}{4148928000} \\ \frac{1997856239}{2438553600} & \frac{28167302927}{8001504000} & -\frac{862419096817}{462309120000} \end{pmatrix}$$

Décomposition de **Dunford**

Toute matrice carrée A se décompose de manière unique sous la forme $A = D + N$ avec D diagonalisable, N nilpotente, $DN = ND$. Cette décomposition est appelée décomposition de **Dunford**. L'algorithme de **Jordan-Chevalley** permet de calculer D (donc aussi N) à partir de A sans rechercher ses valeurs propres (⇨ **http://arxiv.org/pdf/1103.5020.pdf**) :

1) Définir la matrice A, calculer son polynôme caractéristique P et sa dérivée P',
   en déduire le polynôme $Q = \dfrac{P}{\text{pgcd}(P,P')}$ et sa dérivée Q'.
2) Utiliser l'algorithme d'**Euclide** étendu pour trouver deux polynômes U et V vérifiant : $PU + Q'V = 1$
3) Rechercher le plus petit entier r tel que $P \mid Q^r$, puis le plus petit entier s tel que $2^s \geq r$.
4) La matrice D s'obtient par le processus itératif suivant :
$$\begin{cases} D_0 = A \\ D_{i+1} = D_i - V(D_i)Q(D_i) & \text{pour } i = 0, \cdots, s-1 \\ D = D_s \end{cases}$$
5) La matrice N est donnée par :
$$N = A - D$$

La fonction Scilab correspondante est :

```
function [D, N]=JORDAN_CHEVALLEY(A)
    P = M_POLYNOME(A)
    PD = P_DERIVEE(P)
    [Q,R] = P_DIVISION(P,P_PGCD(P,PD))
    QD = P_DERIVEE(Q)
    P1 = P_CONSTANT(1,1)
    [U,V] = P_BACHET(P,QD,P1)
    Q1 = P1
    r = 0
    while %t
        r = r + 1
        Q1 = P_MULTIPLICATION(Q1,Q)
        [Q2,Q3] = P_DIVISION(Q1,P)
        if P_NUL(Q3) then, break, end
    end
    s = ceil(log2(r))
    D = A
    for i = 0:s-1
        M1 = P_MATRICE(V,D)
        M2 = P_MATRICE(Q,D)
        D = M_SOUSTRACTION(D,M_MULTIPLICATION(M1,M2))
    end
    N = M_SOUSTRACTION(A,D)
endfunction
```

La décomposition $A = D + N$ effectuée avec un algorithme faisant intervenir les valeurs propres de A est entachée d'erreurs d'arrondi dans le cas où ces valeurs propres sont irrationnelles puisqu'elles ne peuvent être représentées que de manière approximative dans un calculateur numérique. L'algorithme de **Jordan-Chevalley** fournit la solution exacte lorsque les éléments de la matrice à décomposer sont des nombres rationnels. L'intérêt de la décomposition de **Dunford** est de pouvoir calculer l'exponentielle de la matrice A. Si D a pour valeurs propres $\lambda_1, \ldots, \lambda_n$ alors il existe une matrice de passage inversible P telle que $D = P\text{diag}(\lambda_1, \ldots, \lambda_n)P^{-1}$. Supposons en outre que N soit nilpotente d'indice m, c'est-à-dire que $N^m = 0$, on a :

$$e^A = e^{D+N} = e^D e^N = e^{P\text{diag}(\lambda_1,\cdots,\lambda_n)P^{-1}} e^N = P\,\text{diag}(e^{\lambda_1},\cdots,e^{\lambda_n})P^{-1}\left(I + N + \frac{N^2}{2!} + \cdots + \frac{N^{m-1}}{(m-1)!}\right)$$

D'où, pour un système différentiel linéaire homogène à coefficients constants :

$$\frac{dX}{dt} = AX \Rightarrow X = e^{tA}X_0 = P\,\text{diag}(e^{\lambda_1 t},\cdots,e^{\lambda_n t})P^{-1}\left(I + tN + \frac{t^2}{2!}N^2 + \cdots + \frac{t^{m-1}}{(m-1)!}N^{m-1}\right)X_0$$

Exemple 1

Effectuer la décomposition de **Dunford** de la matrice suivante :

$$A=\begin{pmatrix} 5/6 & 1/3 & 1/2 \\ 2/3 & 1 & 5/6 \\ -5/6 & -1 & -5/6 \end{pmatrix}$$

Appliquons l'algorithme de **Jordan-Chevalley** :

```
exec("Scilab-Matrices.sce",-1)
A = list(FRACTION(5,6),FRACTION(1,3),FRACTION(1,2),...
        FRACTION(2,3),FRACTION(1,1),FRACTION(5,6),...
        FRACTION(-5,6),FRACTION(-1,1),FRACTION(-5,6),...
        3,3) ;
[D,N] = JORDAN_CHEVALLEY(A) ;
M_ECRIRE(D)
M_ECRIRE(N)
```

On obtient :

$$D=\begin{pmatrix} 1/3 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & 1/3 \end{pmatrix} \quad , \quad N=\begin{pmatrix} 1/2 & 1/3 & 1/2 \\ 2/3 & 2/3 & 5/6 \\ -5/6 & -1 & -7/6 \end{pmatrix}$$

Exemple 2

Effectuer la décomposition de **Dunford** de la matrice suivante :

$$A=\begin{pmatrix} 7 & 3 & 2 & 5 \\ 10 & 7 & 4 & 9 \\ 9 & 6 & 4 & 8 \\ -15 & -9 & -5 & -12 \end{pmatrix}$$

Appliquons l'algorithme de **Jordan-Chevalley** :

```
exec("Scilab-Matrices.sce",-1)
A = list(FRACTION(7,1),FRACTION(3,1),FRACTION(2,1),FRACTION(5,1),...
        FRACTION(10,1),FRACTION(7,1),FRACTION(4,1),FRACTION(9,1),...
        FRACTION(9,1),FRACTION(6,1),FRACTION(4,1),FRACTION(8,1),...
        FRACTION(-15,1),FRACTION(-9,1),FRACTION(-5,1),FRACTION(-12,1),...
        4,4) ;
[D,N] = JORDAN_CHEVALLEY(A) ;
M_ECRIRE(D)
M_ECRIRE(N)
```

On obtient :

$$D=\begin{pmatrix} 10 & 3 & 5 & 8 \\ 3 & 4 & 1 & 3 \\ 3 & 3 & 2 & 3 \\ -12 & -6 & -6 & -10 \end{pmatrix} \quad , \quad N=\begin{pmatrix} -3 & 0 & -3 & -3 \\ 7 & 3 & 3 & 6 \\ 6 & 3 & 2 & 5 \\ -3 & -3 & 1 & -2 \end{pmatrix}$$

Exemple 3

Montrer que la matrice $A=\begin{pmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&1&1&1&1&1\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\end{pmatrix}$ est diagonalisable.

Méthode avec recherche des valeurs propres

Pour que A soit diagonalisable dans $\mathbb{C}$ il faut et il suffit que la dimension du sous-espace propre associé à chaque valeur propre soit égale à l'ordre de multiplicité de cette valeur propre. Il est donc nécessaire de rechercher toutes les racines du polynôme caractéristique de A : $P(x)=x^7-3x^6+2x^5+x-1=(x-1)^2(x^5-x^4-x^3-x^2-x-1)=0$ en sachant que, mis à part la racine double évidente $x_1 = x_2 = 1$, les racines de $x^5-x^4-x^3-x^2-x-1=0$ (1 racine réelle x_3 et 4 racines complexes conjuguées deux à deux x_4 , x_5 , x_6 , x_7) sont toutes irrationnelles et ne peuvent pas s'exprimer avec des fonctions usuelles.

En saisissant *x = poly(0,'x') ; roots(x^5 - x^4 - x^3 - x^2 - x - 1)* à la console Scilab, on obtient les approximations suivantes :

$x_3 \approx$ 1.9659482
$x_4 \approx$ 0.1953766 + 0.8488536*i*
$x_5 \approx$ 0.1953766 − 0.8488536*i*
$x_6 \approx -$ 0.6783507 + 0.4585362*i*
$x_7 \approx -$ 0.6783507 − 0.4585362*i*

Ces 5 racines étant distinctes et toutes différentes de $x_1 = x_2 = 1$, les sous-espaces propres associés sont forcément de dimension 1, plus précisément on établit que le sous-espace vectoriel associé à la valeur propre x_k , $3 \le k \le 7$, est engendré par le vecteur propre $V_k = (0,0,x_k^4,x_k^3,x_k^2,x_k,1)$. Il reste à s'assurer que la dimension du sous-espace propre associé à la valeur propre double $x_1 = x_2 = 1$ est de dimension 2. Le système linéaire $(I-A)V = 0$ a pour solution $V = (\alpha,\beta,0,0,0,0,0)$ où α et β sont deux nombres complexes quelconques ; par suite le noyau de la matrice $(I-A)$ est un sous-espace vectoriel de dimension 2 engendré par les vecteurs propres $V_1 = (1,0,0,0,0,0,0)$ et $V_2 = (0,1,0,0,0,0,0)$. Donc A est diagonalisable.

Méthode sans recherche des valeurs propres

On effectue la décomposition exacte $A = D + N$ puis on vérifie la propriété : A diagonalisable $\Leftrightarrow$ N nulle

```
A = list(FRACTION(1,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),...
      FRACTION(0,1),FRACTION(1,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),...
      FRACTION(0,1),FRACTION(0,1),FRACTION(1,1),FRACTION(1,1),FRACTION(1,1),FRACTION(1,1),FRACTION(1,1),...
      FRACTION(0,1),FRACTION(0,1),FRACTION(1,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),...
      FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(1,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),...
      FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(1,1),FRACTION(0,1),FRACTION(0,1),...
      FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(1,1),FRACTION(0,1),...
      7,7) ;
[D,N] = JORDAN_CHEVALLEY(A) ;
```

Précisons les étapes de l'algorithme de **Jordan-Chevalley** :

$P(x)=x^7-3x^6+2x^5+x-1$

$Q(x)=x^6-2x^5+1$

$U(x)=\frac{5973}{4792}x^4-x^3-x^2-x-1$

$V(x)=-\frac{1991}{9584}x^6+\frac{1063}{2396}x^5-\frac{45}{4792}x^4-\frac{75}{4792}x^3-\frac{125}{4792}x^2-\frac{1007}{4792}x-\frac{2153}{9584}$

$r=2$
$s=1$
$D = A$
$N = 0$

Comme $N = 0$, A est diagonalisable.

Remarque

Soit la matrice $B = \begin{pmatrix} 1 & \varepsilon & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$, $\varepsilon \neq 0$.

Cette matrice n'est pas diagonalisable. En effet, bien que B et A aient le même polynôme caractéristique, le sous-espace propre associé à la valeur propre double $x_1 = x_2 = 1$ de B est de dimension 1 puisque les vecteurs V tels que $(I - B)V = 0$ sont de la forme $V = (\alpha,0,0,0,0,0,0)$, $\alpha \in \mathbb{C}$.
Il est facile de vérifier que la décomposition $B = D + N$ est :

$$\begin{pmatrix} 1 & \varepsilon & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & \varepsilon & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Ainsi $N \neq 0$ et par conséquent B n'est pas diagonalisable.

Exemple 4

Considérons :

- la matrice ordinaire $M=\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ de polynôme caractéristique $\det(xI_2-M)=x^2-(a+d)x+(ad-bc)$
- la matrice partitionnée $B=\left(\begin{array}{c:c} M & R \\ \hdashline 0 & M \end{array}\right)$ de polynôme caractéristique $\det(xI_4-B)=[\det(xI_2-M)]^2$
- la matrice partitionnée $A=\left(\begin{array}{c:c} B & 0 \\ \hdashline S & B \end{array}\right)$ de polynôme caractéristique $\det(xI_8-A)=[\det(xI_2-M)]^4$

Choisissons par exemple : $M=\begin{pmatrix} 29 & 12 \\ -3 & 18 \end{pmatrix}, R=\begin{pmatrix} 20 & 14 \\ -11 & -7 \end{pmatrix}, S=\begin{pmatrix} 13 & 12 & 6 & -9 \\ 15 & -8 & -1 & -13 \\ -21 & 17 & -10 & 4 \\ 19 & 12 & -2 & 23 \end{pmatrix}$

Le polynôme caractéristique de *M* est $x^2-47x+558$. Les valeurs propres de *M* sont $x_1=\frac{47+i\sqrt{23}}{2}$ et $x_2=\frac{47-i\sqrt{23}}{2}$

La matrice carrée *A* d'ordre 8 est la suivante :

$$A=\begin{pmatrix} 29 & 12 & 20 & 14 & 0 & 0 & 0 & 0 \\ -3 & 18 & -11 & -7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 29 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 18 & 0 & 0 & 0 & 0 \\ 13 & 12 & 6 & -9 & 29 & 12 & 20 & 14 \\ 15 & -8 & -1 & -13 & -3 & 18 & -11 & -7 \\ -21 & 17 & -10 & 4 & 0 & 0 & 29 & 12 \\ 19 & 12 & -2 & 23 & 0 & 0 & -3 & 18 \end{pmatrix}$$

Le polynôme caractéristique de *A* est : $P(x)=\left(x^2-47x+558\right)^4=\left(x-\frac{47+i\sqrt{23}}{2}\right)^4\left(x-\frac{47-i\sqrt{23}}{2}\right)^4$

Pour appliquer l'algorithme de **Jordan-Chevalley** à la matrice *A* on écrit :

```
A = list(FRACTION(29,1),FRACTION(12,1),FRACTION(20,1),FRACTION(14,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),...
      FRACTION(-3,1),FRACTION(18,1),FRACTION(-11,1),FRACTION(-7,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),...
      FRACTION(0,1),FRACTION(0,1),FRACTION(29,1),FRACTION(12,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),...
      FRACTION(0,1),FRACTION(0,1),FRACTION(-3,1),FRACTION(18,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),...
      FRACTION(13,1),FRACTION(12,1),FRACTION(6,1),FRACTION(-9,1),FRACTION(29,1),FRACTION(12,1),FRACTION(20,1),FRACTION(14,1),...
      FRACTION(15,1),FRACTION(-8,1),FRACTION(-1,1),FRACTION(-13,1),FRACTION(-3,1),FRACTION(18,1),FRACTION(-11,1),FRACTION(-7,1),...
      FRACTION(-21,1),FRACTION(17,1),FRACTION(-10,1),FRACTION(4,1),FRACTION(0,1),FRACTION(0,1),FRACTION(29,1),FRACTION(12,1),...
      FRACTION(19,1),FRACTION(12,1),FRACTION(-2,1),FRACTION(23,1),FRACTION(0,1),FRACTION(0,1),FRACTION(-3,1),FRACTION(18,1),...
      8,8) ;
[D,N] = JORDAN_CHEVALLEY(A) ;
```

Ci-dessous, les résultats intermédiaires de l'exécution de ce programme :

$P(x)=x^8-188x^7+15486x^6-730004x^5+21539329x^4-407342232x^3+4821782904x^2-32663329056x+96947540496$

$Q(x)=x^2-47x+558$

$U(x)=\frac{256}{279841}$

$V(x)=-\frac{128}{279841}x^7+\frac{21056}{279841}x^6-\frac{1487392}{279841}x^5+\frac{58486800}{279841}x^4-\frac{1382594312}{279841}x^3+\frac{19648839360}{279841}x^2-\frac{155440486656}{279841}x+\frac{528054682705}{279841}$

$r=4$

$s=2$

Finalement, on obtient la décomposition $A = D + N$:

$$A = \begin{pmatrix} 29 & 12 & 20 & 14 & 0 & 0 & 0 & 0 \\ -3 & 18 & -11 & -7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 29 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 18 & 0 & 0 & 0 & 0 \\ 13 & 12 & 6 & -9 & 29 & 12 & 20 & 14 \\ 15 & -8 & -1 & -13 & -3 & 18 & -11 & -7 \\ -21 & 17 & -10 & 4 & 0 & 0 & 29 & 12 \\ 19 & 12 & -2 & 23 & 0 & 0 & -3 & 18 \end{pmatrix}$$

$$D = \begin{pmatrix} 29 & 12 & 30/23 & -290/23 & 0 & 0 & 0 & 0 \\ -3 & 18 & -100/23 & -30/23 & 0 & 0 & 0 & 0 \\ 0 & 0 & 29 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 18 & 0 & 0 & 0 & 0 \\ 74138/529 & 114844/529 & 3703937/12167 & 10243407/12167 & 29 & 12 & 30/23 & -290/23 \\ -26963/529 & -64478/529 & 1592057/12167 & 2140133/12167 & -3 & 18 & -100/23 & -30/23 \\ -429/23 & 283/23 & -135277/529 & -291819/529 & 0 & 0 & 29 & 12 \\ 464/23 & 429/23 & 54480/529 & 125617/529 & 0 & 0 & -3 & 18 \end{pmatrix}$$

$$N = \begin{pmatrix} 0 & 0 & 430/23 & 612/23 & 0 & 0 & 0 & 0 \\ 0 & 0 & -153/23 & -131/23 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -67261/529 & -108496/529 & -3630935/12167 & -10352910/12167 & 0 & 0 & 430/23 & 612/23 \\ 34898/529 & 60246/529 & -1604224/12167 & -2298304/12167 & 0 & 0 & -153/23 & -131/23 \\ -54/23 & 108/23 & 129987/529 & 293935/529 & 0 & 0 & 0 & 0 \\ -27/23 & -153/23 & -55538/529 & -113450/529 & 0 & 0 & 0 & 0 \end{pmatrix}$$

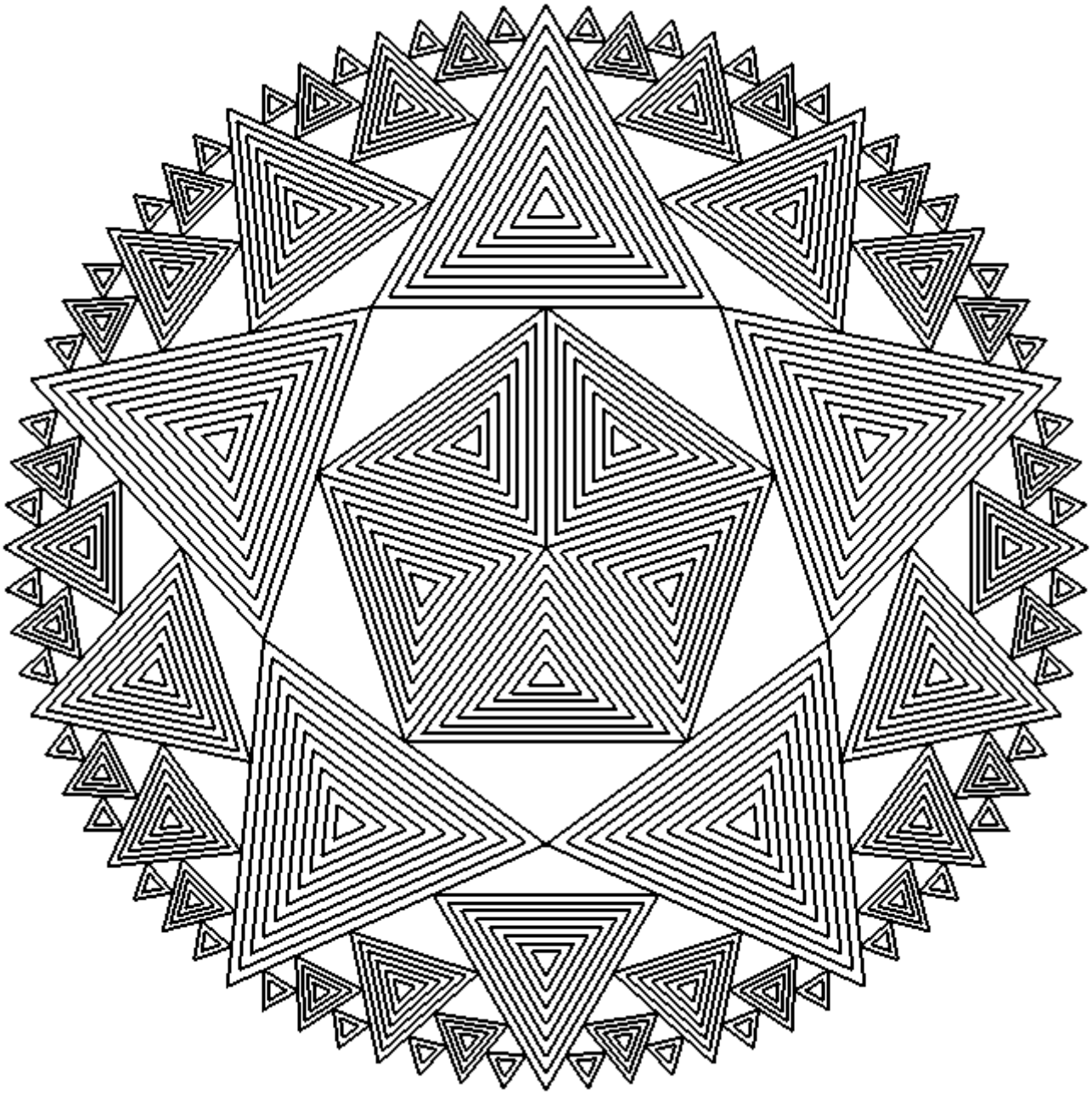
La matrice D est diagonalisable sous la forme :

$$\operatorname{diag}\left(\frac{47+i\sqrt{23}}{2}, \frac{47+i\sqrt{23}}{2}, \frac{47+i\sqrt{23}}{2}, \frac{47+i\sqrt{23}}{2}, \frac{47-i\sqrt{23}}{2}, \frac{47-i\sqrt{23}}{2}, \frac{47-i\sqrt{23}}{2}, \frac{47-i\sqrt{23}}{2}\right)$$

La matrice N est nilpotente d'indice 4 : $N^4 = 0$

Les matrices D et N commutent : $DN = ND$

Mathématique & Informatique



Mathématique & Informatique

Matrices polynomiales

Les matrices polynomiales étudiées sont des matrices de polynômes de $\mathbb{Q}(x)$.

Structure des données

Une matrice polynomiale rationnelle M de dimension (m,n) est représentée par une liste de mn polynômes $P_i(x)$:

$M = list(P_1, P_2, \ldots, P_{mn}, m, n)$

Chaque polynôme P_i est représenté par la liste des coefficients de ses monômes $a_i x^i$:

$P_i = list(a_k, a_{k-1}, \ldots, a_1, a_0)$

Chaque nombre rationnel a_i est représenté par la liste du numérateur et du dénominateur de sa fraction e_1/e_2 :

$a_i = list(e_1, e_2)$

Chaque entier relatif e_i est représenté par le vecteur signé de ses chiffres $\pm c_1 c_2 \ldots c_N$:

$e_i = [s, c_1, c_2, \ldots, c_N]$

Attribuons le préfixe MP aux fonctions matricielles polynomiales pour les différencier des fonctions matricielles ordinaires qui ont le préfixe M. Développons en langage de programmation Scilab les fonctions qui permettent d'effectuer les opérations habituelles sur les matrices polynomiales rationnelles : écriture, addition, soustraction, multiplication, multiplication par un polynôme, comparaison, ...

NB Ne pas confondre le nombre rationnel p/q ≡ list(p,q) avec le polynôme constant P(x) = p/q ≡ list(list(p,q)) ou avec la matrice à 1 seul élément M = p/q ≡ list(list(list(p,q)),1,1)

```
// ECRITURE D'UNE MATRICE

function MP_ECRIRE(A)
  m = A($-1) ; n = A($)
  printf('\n Matrice %i ligne(s) et %i colonne(s) :',m,n)
  printf('\n ====================================')
  for i = 1:m*n
    lig = floor((i-1)/n) + 1 ; col = modulo(i-1,n) + 1
    printf('\n (%i,%i) = ',lig,col), P_ECRIRE(A(i))
  end
endfunction

// MATRICE (OU VECTEUR) VIDE

function V = MP_VIDE(m,n)
  V = list()
  for i = 1:m*n
    V($+1) = list()
  end
  V($+1) = m ; V($+1) = n
endfunction
```

```
// MATRICE (OU VECTEUR) NULLE

function Z = MP_ZERO(m,n)
  Z = list()
  for i = 1:m*n
    Z($+1) = P_CONSTANT(0,1)
  end
  Z($+1) = m ; Z($+1) = n
endfunction

// MATRICE (OU VECTEUR) UNITE

function U = MP_UN(m,n)
  U = list()
  for i = 1:m*n
    U($+1) = P_CONSTANT(1,1)
  end
  U($+1) = m ; U($+1) = n
endfunction

// MATRICE IDENTITE

function I = MP_IDENTITE(n)
  I = list()
  for i = 1:n
    for j = 1:n
      if i == j then
        I($+1) = P_CONSTANT(1,1)
      else
        I($+1) = P_CONSTANT(0,1)
      end
    end
  end
  I($+1) = n ; I($+1) = n
endfunction

// SOMME DE DEUX MATRICES

function C = MP_ADDITION(A,B)
  ma = A($-1) ; na = A($)
  mb = B($-1) ; nb = B($)
  if (ma == mb)&(na == nb) then
    m = ma ; n = na
  else
    C = list(0,0)
    return
  end
  C = list()
  for i = 1:m*n
    C($+1) = P_ADDITION(A(i),B(i))
  end
  C($+1) = m ; C($+1) = n
endfunction
```

```
// DIFFERENCE DE DEUX MATRICES

function C = MP_SOUSTRACTION(A,B)
  ma = A($-1) ; na = A($)
  mb = B($-1) ; nb = B($)
  if (ma == mb)&(na == nb) then
    m = ma ; n = na
  else
    C = list(0,0)
    return
  end
  C = list()
  for i = 1:m*n
    C($+1) = P_SOUSTRACTION(A(i),B(i))
  end
  C($+1) = m ; C($+1) = n
endfunction

// OPPOSEE D'UNE MATRICE

function B = MP_MOINS(A)
  m = A($-1) ; n = A($)
  B = list()
  for i = 1:m*n
    B($+1) = P_MOINS(A(i))
  end
  B($+1) = m ; B($+1) = n
endfunction

// VALEUR ABSOLUE D'UNE MATRICE

function B = MP_ABSOLU(A)
  m = A($-1) ; n = A($)
  B = list()
  for i = 1:m*n
    B($+1) = P_ABSOLU(A(i))
  end
  B($+1) = m ; B($+1) = n
endfunction

// PRODUIT D'UNE MATRICE PAR UN POLYNOME

function B = MP_SCALAIRE(A,P)
  m = A($-1) ; n = A($)
  B = list()
  for i = 1:m*n
    B($+1) = P_MULTIPLICATION(A(i),P)
  end
  B($+1) = m ; B($+1) = n
endfunction
```

```
// PRODUIT DE DEUX MATRICES

function C = MP_MULTIPLICATION(A,B)
  ma = A($-1) ; na = A($)
  mb = B($-1) ; nb = B($)
  if na == mb then
    m = ma ; n = nb ; p = na
  else
    C = list(0,0)
    return
  end
  C = list()
  for i = 1:m
    for j = 1:n
      s = P_CONSTANT(0,1)
      for k = 1:p
        t = P_MULTIPLICATION(A((i-1)*na+k),B((k-1)*nb+j))
        s = P_ADDITION(s,t)
      end
      C($+1) = s
    end
  end
  C($+1) = m ; C($+1) = n
endfunction

//---  PUISSANCE D'UNE MATRICE

function B = MP_PUISSANCE(A,k)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then
    B = list(0,0) ; return
  end
  B = MP_IDENTITE(n) ; identite = %t
  while k <> 0
    if modulo(k,2) <> 0 then
      if identite then
        B = A ; identite = %f
      else
        B = MP_MULTIPLICATION(B,A)
      end
    end
    k = floor(k/2)
    if k <> 0 then
      A = MP_MULTIPLICATION(A,A)
    end
  end
endfunction
```

```
// TRANSPOSEE D'UNE MATRICE

function B = MP_TRANSPOSEE(A)
  m = A($-1) ; n = A($)
  B = list()
  for j = 1:n
    for i = 1:m
      B($+1) = A((i-1)*n+j)
    end
  end
  B($+1) = n ; B($+1) = m
endfunction

// TRACE D'UNE MATRICE

function P = MP_TRACE(A)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then
    P = list() ; return
  end
  P = P_CONSTANT(0,1)
  for i = 1:n
    P = P_ADDITION(P,A((i-1)*n+i))
  end
endfunction

// COMPARAISON DE DEUX MATRICES

function c = MP_COMPARAISON(A,B)
  ma = A($-1) ; na = A($)
  mb = B($-1) ; nb = B($)
  c = []
  if (ma == mb)&(na == nb) then
    m = ma ; n = na
  else
    return
  end
  for i = 1:m*n
    if i == 1 then
      c = P_COMPARAISON(A(1),B(1))
    else
      if P_COMPARAISON(A(i),B(i)) <> c then
        c = []
        break
      end
    end
  end
endfunction

// TEST MATRICE NULLE

function t = MP_NUL(A)
  if length(A) == 0 then, t = %f, return, end
  t = MP_COMPARAISON(A,MP_ZERO(A($-1),A($))) == 0
endfunction
```

EXEMPLE

Effectuer le produit des 2 matrices polynomiales suivantes :

$$A = \begin{pmatrix} x+1 & \frac{1}{6}x-7 \\ 29 & x^3-\frac{3}{5} \\ \frac{13}{15}x^2-6 & -\frac{6}{11}x^3-x \\ -\frac{2}{9}x+\frac{17}{2} & x^3-\frac{8}{21} \end{pmatrix}$$

$$B = \begin{pmatrix} 3x^3+2x^2-2 & 0 & x^3+x-\frac{11}{37} \\ \frac{1}{2}x^2-4x+\frac{9}{5} & x & \frac{15}{7}x^2-\frac{4}{3}x+\frac{1}{10} \end{pmatrix}$$

Définissons les matrices A et B par 2 listes de polynômes, puis utilisons la fonction *MP_MULTIPLICATION* :

```
exec("Scilab-Matrices.sce",-1)
a11 = list(FRACTION(1,1),FRACTION(1,1)) ;
a12 = list(FRACTION(1,6),FRACTION(-7,1)) ;
a21 = list(FRACTION(29,1)) ;
a22 = list(FRACTION(1,1),FRACTION(0,1),FRACTION(0,1),FRACTION(-3,5)) ;
a31 = list(FRACTION(13,15),FRACTION(0,1),FRACTION(-6,1)) ;
a32 = list(FRACTION(-6,11),FRACTION(0,1),FRACTION(-1,1),FRACTION(0,1)) ;
a41 = list(FRACTION(-2,9),FRACTION(17,2)) ;
a42 = list(FRACTION(1,1),FRACTION(0,1),FRACTION(0,1),FRACTION(-8,21)) ;
A = list(a11,a12,a21,a22,a31,a32,a41,a42,4,2) ; MP_ECRIRE(A)
b11 = list(FRACTION(3,1),FRACTION(2,1),FRACTION(0,1),FRACTION(-2,1)) ;
b12 = list(FRACTION(0,1)) ;
b13 = list(FRACTION(1,1),FRACTION(0,1),FRACTION(1,1),FRACTION(-11,37)) ;
b21 = list(FRACTION(1,2),FRACTION(-4,1),FRACTION(9,5)) ;
b22 = list(FRACTION(1,1),FRACTION(0,1)) ;
b23 = list(FRACTION(15,7),FRACTION(-4,3),FRACTION(1,10)) ;
B = list(b11,b12,b13,b21,b22,b23,2,3) ; MP_ECRIRE(B)
C = MP_MULTIPLICATION(A,B) ; MP_ECRIRE(C)
```

On obtient la matrice C = A × B :

$$C = \begin{pmatrix} 3x^4+\frac{61}{12}x^3-\frac{13}{6}x^2+\frac{263}{10}x-\frac{73}{5} & \frac{1}{6}x^2-7x & x^4+\frac{19}{14}x^3-\frac{128}{9}x^2+\frac{7439}{740}x-\frac{369}{370} \\ \frac{1}{2}x^5-4x^4+\frac{444}{5}x^3+\frac{577}{10}x^2+\frac{12}{5}x-\frac{1477}{25} & x^4-\frac{3}{5}x & \frac{15}{7}x^5-\frac{4}{3}x^4+\frac{291}{10}x^3-\frac{9}{7}x^2+\frac{149}{5}x-\frac{16061}{1850} \\ \frac{128}{55}x^5+\frac{646}{165}x^4-\frac{2143}{110}x^3-\frac{146}{15}x^2-\frac{9}{5}x+12 & -\frac{6}{11}x^4-x^2 & -\frac{349}{1155}x^5+\frac{8}{11}x^4-\frac{8467}{1155}x^3+\frac{199}{185}x^2-\frac{61}{10}x+\frac{66}{37} \\ \frac{1}{2}x^5-\frac{14}{3}x^4+\frac{2417}{90}x^3+\frac{353}{21}x^2+\frac{124}{63}x-\frac{619}{35} & x^4-\frac{8}{21}x & \frac{15}{7}x^5-\frac{14}{9}x^4+\frac{43}{5}x^3-\frac{458}{441}x^2+\frac{14101}{1554}x-\frac{19931}{7770} \end{pmatrix}$$

Polynôme minimal

C'est un polynôme de matrice.
Soit A une matrice carrée d'ordre n à coefficients rationnels et soit $P(x) = a_k x^k + a_{k-1}x^{k-1} + ... + a_1 x + a_0$ un polynôme de degré k à coefficients rationnels.
On définit la matrice P(A) par : $P(A) = a_k A^k + a_{k-1}A^{k-1} + ... + a_1 A + a_0 I$, où I est la matrice identité d'ordre n.
Il existe des polynômes P(x) pour lesquels P(A) = 0, par exemple le polynôme caractéristique $\Delta(x)$ de A :

$$\Delta(x) = \det(xI - A) \quad \Rightarrow \quad \Delta(A) = 0$$ (Théorème de **Cayley-Hamilton**)

Le polynôme unitaire m(x) de plus petit degré qui annule la matrice A est le polynôme minimal de A.

Propriétés :

- m(x) est unique
  $\exists$ m' : m'(A) = 0 , deg m' = deg m $\Rightarrow$ (m' - m)(A) = 0 , deg(m' - m) < deg m $\Rightarrow$ m non minimal
- m(x) divise tout polynôme P(x) admettant A comme zéro
  P(x) = q(x)m(x) + r(x) $\Rightarrow$ r(A) = 0 , deg r < deg m $\Rightarrow$ r = 0 $\Rightarrow$ m divise P

L'algorithme des facteurs invariants s'applique dans tout anneau euclidien (anneau commutatif unitaire intègre muni de la division euclidienne). Considérons une matrice M quelconque dont les coefficients sont des éléments de $\mathbb{Q}(x)$.
On obtient 2 matrices carrées P et Q, et une matrice diagonale D, toutes les 3 à coefficients dans $\mathbb{Q}(x)$, telles que :

$$D = PMQ$$

Prenons la matrice caractéristique M = xI - A. La recherche des facteurs invariants de M conduit à la matrice diagonale $D = Diag(d_1, d_2, ... , d_n)$ dont les éléments d_i sont des polynômes que l'on peut normer en prenant leur premier coefficient égal à 1. Les d_i sont les invariants de similitude de A.

On montre que le polynôme minimal m(x) de A est égal au dernier élément normé d_n de D et que le polynôme caractéristique $\Delta(x)$ de A est égal au produit des éléments normés $d_1 \times d_2 \times ... \times d_n$ de D.

Application

Deux matrices A et B sont dites semblables si elles représentent le même opérateur linéaire.
Dans ce cas, il existe une matrice inversible P (matrice de changement de base) telle que : $B = P^{-1}AP$
On dit aussi que B est obtenue à partir de A par une similitude.
La similitude des matrices est une relation d'équivalence :

1) A est semblable à A
$A = I^{-1}AI$
2) si A est semblable à B, alors B est semblable à A
$A = P^{-1}BP \quad \Rightarrow \quad B = PAP^{-1} = (P^{-1})^{-1}A(P^{-1})$
3) si A est semblable à B et B est semblable à C, alors A est semblable à C
$A = P^{-1}BP$, $B = Q^{-1}CQ \quad \Rightarrow \quad A = P^{-1}Q^{-1}CQP = (QP)^{-1}C(QP)$

Théorème :
Deux matrices carrées A et B sont semblables si et seulement si elles ont les mêmes invariants de similitude.

$$B = P^{-1}AP \quad \Leftrightarrow \quad D_A = D_B$$

Fonctions Scilab

Ajoutons de nouvelles fonctions matricielles polynomiales :

function [D,P,Q] = MP_INVARIANTS(M)
/
/
endfunction

La fonction *MP_INVARIANTS* prend en paramètre d'entrée une matrice rectangulaire M de m lignes et de n colonnes dont les éléments sont des polynômes de $\mathbb{Q}(x)$ et retourne en paramètres de sortie les matrices D, P, Q telles que :
D = PMQ où D est la matrice diagonale des facteurs invariants normés de M
Cette fonction est analogue à la fonction *M_INVARIANTS* décrite précédemment. La différence essentielle est que les calculs s'effectuent dans l'anneau des polynômes avec les fonctions élémentaires adéquates.

function B = MP_HORNER(P,A)
/
/
endfunction

La fonction *MP_HORNER* calcule la matrice P(A) où P est un polynôme de degré p et A une matrice carrée rationnelle d'ordre n.

function M = MP_CARACTERISTIQUE(A)
/
/
endfunction

La fonction *MP_CARACTERISTIQUE* crée la matrice polynomiale M = xI - A à partir d'une matrice carrée A dont les éléments sont des nombres rationnels.

function [PM,PC] = MP_MINIMAL(A)
/
/
endfunction

La fonction *MP_MINIMAL* fournit le polynôme minimal PM et également le polynôme caractéristique PC d'une matrice carrée A à coefficients rationnels.

function D = MP_DETERMINANT(M)
/
/
endfunction

La fonction *MP_DETERMINANT* retourne le polynôme D qui est le déterminant de la matrice polynomiale M.
En particulier, si A est une matrice rationnelle et M = xI - A est sa matrice caractéristique, alors le déterminant de M est le polynôme caractéristique de A (manière de calculer le polynôme caractéristique d'une matrice en se ramenant à la définition).

Programmes

```
//---  FACTEURS INVARIANTS D'UNE MATRICE POLYNOMIALE

function [D,P,Q] = MP_INVARIANTS(M)
  function [D,P,Q] = permuter(D,P,Q,typ)
    if typ == 'l' then
      for c = k:n
        t = D((k-1)*n+c)
        D((k-1)*n+c) = D((u-1)*n+c)
        D((u-1)*n+c) = t
      end
      for c = 1:m
        t = P((k-1)*m+c)
        P((k-1)*m+c) = P((u-1)*m+c)
        P((u-1)*m+c) = t
      end
    end
    if typ == 'c' then
      for l = k:m
        t = D((l-1)*n+k)
        D((l-1)*n+k) = D((l-1)*n+v)
        D((l-1)*n+v) = t
      end
      for l = 1:n
        t = Q((l-1)*n+k)
        Q((l-1)*n+k) = Q((l-1)*n+v)
        Q((l-1)*n+v) = t
      end
    end
  endfunction
  m = M($-1) ; n = M($)
  D = list() ; P = list() ; Q = list()
  for i = 1:m
    for j = 1:n
      D($+1) = M((i-1)*n+j)
    end
  end
  for i = 1:m
    for j = 1:m
      if i == j then
        P($+1) = P_CONSTANT(1,1)
      else
        P($+1) = P_CONSTANT(0,1)
      end
    end
  end
  for i = 1:n
    for j = 1:n
      if i == j then
        Q($+1) = P_CONSTANT(1,1)
      else
        Q($+1) = P_CONSTANT(0,1)
      end
    end
  end
```

```
for k = 1:min(m,n)
  d = []
  for i = k:m
    for j = k:n
      if ~P_NUL(D((i-1)*n+j)) then
        if isequal(d,[]) then
          u = i ; v = j
          d = P_ABSOLU(D((i-1)*n+j))
        elseif P_COMPARAISON(P_ABSOLU(D((i-1)*n+j)),d) < 0 then
          u = i ; v = j
          d = P_ABSOLU(D((i-1)*n+j))
        end
      end
    end
  end
  if ~isequal(d,[]) then
    if u <> k then, [D,P,Q] = permuter(D,P,Q,'l'), end
    if v <> k then, [D,P,Q] = permuter(D,P,Q,'c'), end
  else
    break
  end
  while %t
    for i = k+1:m
      if ~P_NUL(D((i-1)*n+k)) then
        [q,r] = P_DIVISION(D((i-1)*n+k),D((k-1)*n+k))
        for c = k:n
          t = P_MULTIPLICATION(q,D((k-1)*n+c))
          D((i-1)*n+c) = P_SOUSTRACTION(D((i-1)*n+c),t)
        end
        for c = 1:m
          t = P_MULTIPLICATION(q,P((k-1)*m+c))
          P((i-1)*m+c) = P_SOUSTRACTION(P((i-1)*m+c),t)
        end
      end
    end
    for j = k+1:n
      if ~P_NUL(D((k-1)*n+j)) then
        [q,r] = P_DIVISION(D((k-1)*n+j),D((k-1)*n+k))
        for l = k:m
          t = P_MULTIPLICATION(q,D((l-1)*n+k))
          D((l-1)*n+j) = P_SOUSTRACTION(D((l-1)*n+j),t)
        end
        for l = 1:n
          t = P_MULTIPLICATION(q,Q((l-1)*n+k))
          Q((l-1)*n+j) = P_SOUSTRACTION(Q((l-1)*n+j),t)
        end
      end
    end
```

```
    d = []
    for i = k+1:m
      if ~P_NUL(D((i-1)*n+k)) then
        if isequal(d,[]) then
          u = i ; v = k
          d = P_ABSOLU(D((i-1)*n+k))
        elseif P_COMPARAISON(P_ABSOLU(D((i-1)*n+k)),d) < 0 then
          u = i ; v = k
          d = P_ABSOLU(D((i-1)*n+k))
        end
      end
    end
    for j = k+1:n
      if ~P_NUL(D((k-1)*n+j)) then
        if isequal(d,[]) then
          u = k ; v = j
          d = P_ABSOLU(D((k-1)*n+j))
        elseif P_COMPARAISON(P_ABSOLU(D((k-1)*n+j)),d) < 0 then
          u = k ; v = j
          d = P_ABSOLU(D((k-1)*n+j))
        end
      end
    end
    if ~isequal(d,[]) then
      if v == k then, [D,P,Q] = permuter(D,P,Q,'l'), end
      if u == k then, [D,P,Q] = permuter(D,P,Q,'c'), end
    else
      iter = %f
      for i = k+1:m
        for j = k+1:n
          [q,r] = P_DIVISION(D((i-1)*n+j),D((k-1)*n+k))
          if P_COMPARAISON(r,P_CONSTANT(0,1)) <> 0 then
            for l = k:m
              D((l-1)*n+k) = P_ADDITION(D((l-1)*n+k),D((l-1)*n+j))
            end
            for l = 1:n
              Q((l-1)*n+k) = P_ADDITION(Q((l-1)*n+k),Q((l-1)*n+j))
            end
            iter = %t
            break
          end
        end
        if iter then, break, end
      end
      if ~iter then, break, end
    end
  end
end
```

```
  for i = 1:m
    d = D((i-1)*n+i)(1)
    if F_NUL(d) then, continue, end
    D((i-1)*n+i) = P_NORMALISE(D((i-1)*n+i))
    for j = 1:m
      P((i-1)*m+j) = P_SCALAIRE(P((i-1)*m+j),F_INVERSION(d))
    end
  end
  D($+1) = m ; D($+1) = n
  P($+1) = m ; P($+1) = m
  Q($+1) = n ; Q($+1) = n
endfunction


//---  MATRICE CARACTERISTIQUE

function M = MP_CARACTERISTIQUE(A)
  m = A($-1) ; n = A($)
  if (m <> n)|(m*n == 0) then, M = list(0,0), return, end
  M = list()
  for i = 1:n
    for j = 1:n
      if i == j then
        M($+1) = list(FRACTION(1,1),F_MOINS(A((i-1)*n+j)))
      else
        M($+1) = list(F_MOINS(A((i-1)*n+j)))
      end
    end
  end
  M($+1) = n ; M($+1) = n
endfunction


//---  POLYNOME MINIMAL ET POLYNOME CARACTERISTIQUE

function [PM,PC] = MP_MINIMAL(A)
  PM = list() ; PC = list()
  M = MP_CARACTERISTIQUE(A)
  if M($) == 0 then, return, end
  [D,P,Q] = MP_INVARIANTS(M)
  n = D($)
  PM = D(n*n)
  PC = P_CONSTANT(1,1)
  for i = 1:n
    PC = P_MULTIPLICATION(PC,D((i-1)*n+i))
  end
endfunction
```

```
//---  POLYNOME DE MATRICE

function B = MP_HORNER(P,A)
  m = A($-1) ; n = A($) ; p = length(P)
  if (m <> n)|(m*n == 0)|(p == 0) then
    B = list(0,0) ; return
  end
  B = M_ZERO(n,n)
  I = M_IDENTITE(n)
  for i = 1:p
    M1 = M_MULTIPLICATION(B,A)
    M2 = M_SCALAIRE(I,P(i))
    B = M_ADDITION(M1,M2)
  end
endfunction

//---  DETERMINANT D'UNE MATRICE

function D = MP_DETERMINANT(M)
  m = M($-1) ; n = M($)
  if (m <> n)|(m*n == 0) then, D = list() ; return, end
  s = 1
  P = P_CONSTANT(1,1)
  for k = 1:n
    p = 0
    for i = k:n
      if ~P_NUL(M((i-1)*n+k)) then, p = i ; break, end
    end
    if p == 0 then, D = P_CONSTANT(0,1) ; return, end
    if p <> k then
      s = -s
      for j = k:n
        t = M((k-1)*n+j)
        M((k-1)*n+j) = M((p-1)*n+j)
        M((p-1)*n+j) = t
      end
    end
    for i = k+1:n
      if ~P_NUL(M((i-1)*n+k)) then
        for j = k+1:n
          M((i-1)*n+j) = P_MULTIPLICATION(M((i-1)*n+j),M((k-1)*n+k))
          t = P_MULTIPLICATION(M((i-1)*n+k),M((k-1)*n+j))
          M((i-1)*n+j) = P_SOUSTRACTION(M((i-1)*n+j),t)
        end
        P = P_MULTIPLICATION(P,M((k-1)*n+k))
      end
    end
  end
  D = P_CONSTANT(s,1)
  for i = 1:n
    D = P_MULTIPLICATION(D,M((i-1)*n+i))
  end
  [D,R] = P_DIVISION(D,P)
endfunction
```

Mathématique & Informatique

Matrice carrée polynomiale inverse

L'inverse d'une matrice carrée polynomiale M est la matrice carrée M^{-1} vérifiant la double égalité :

$$MM^{-1} = M^{-1}M = I$$

Pour trouver M^{-1} on peut appliquer la méthode de **Jordan** et diagonaliser (M,I) en introduisant la notion de fraction rationnelle polynomiale $F(x) = P_1(x) / P_2(x)$ représentée par $F = list(P_1,P_2)$ où P_1 et P_2 sont 2 polynômes de $\mathbb{Q}(x)$.
Afin d'écrire F sous forme irréductible, il faut rechercher le pgcd de P_1 et P_2 et effectuer la simplification de la fraction.
La quantité de calcul et le temps d'exécution nécessaires pour aboutir à M^{-1} sont importants.
Une autre solution consiste à mettre l'inverse de la matrice M sous la forme :

$$M^{-1} = \frac{{}^t com(M)}{det(M)}$$

La méthode de **Le Verrier** donne le déterminant de M et également la transposée de la comatrice de M.
Son intérêt est la simplicité de l'algorithme qui ne fait appel qu'à des additions et à des multiplications matricielles.
Définissons une fonction Scilab *MP_INVERSE* qui fournit le déterminant D d'une matrice carrée polynomiale M ainsi que la transposée de la comatrice de M notée T :

```
function [D,T] = MP_INVERSE(M)
  m = M($-1) ; n = M($)
  if (m <> n)|(m*n == 0) then
    D = list() ; T = list(0,0)
    return
  end
  I = MP_IDENTITE(n)
  P = P_CONSTANT(1,1)
  TI = I
  for i = 1:n
    MI = MP_MULTIPLICATION(TI,M)
    P = P_SCALAIRE(MP_TRACE(MI),FRACTION(-1,i))
    if i == n then, break, end
    TI = MP_ADDITION(MI,MP_SCALAIRE(I,P))
  end
  D = P_MOINS(P) ; T = TI
endfunction
```

Remarque

Si l'on souhaite obtenir uniquement le déterminant D de M, on peut utiliser la fonction *MP_DETERMINANT* qui triangularise M par la méthode du pivot de **Gauss**.

Extension

On veut inverser une matrice carrée R dont les éléments sont des fractions rationnelles polynomiales :

$R = (P_{ij} / Q_{ij})$ où P_{ij} et Q_{ij} sont des polynômes de $\mathbb{Q}(x)$

Soit $Q = PPCM(Q_{ij})$ le polynôme obtenu avec la fonction *P_PGCDPPCM* appliquée à la liste des Q_{ij}
La matrice $M = QR$ est une matrice carrée polynomiale dont les éléments sont des polynômes de $\mathbb{Q}(x)$.
Par suite, la fonction *MP_INVERSE* donne M^{-1} :

$M^{-1} = (1/D)T$ avec $T = {}^t com(M)$ et $D = det(M)$

De la relation $M^{-1}M = M^{-1}(QR) = (QM^{-1})R = I$, on déduit :

$R^{-1} = QM^{-1} = (1/D)(QT)$

Mathématique & Informatique

EXEMPLE 1

Calculer le déterminant D de la matrice M :

$$M = \begin{pmatrix} 3x^3+2x^2-2 & 29 & x^3+x-\frac{11}{37} \\ \frac{1}{2}x^2-4x+\frac{9}{5} & x & \frac{15}{7}x^2-\frac{4}{3}x+\frac{1}{10} \\ \frac{13}{15}x^2-6 & x^3-\frac{3}{5} & -\frac{2}{9}x+\frac{17}{2} \end{pmatrix}$$

Programme :

```
m11 = list(FRACTION(3,1),FRACTION(2,1),FRACTION(0,1),FRACTION(-2,1)) ;
m12 = list(FRACTION(29,1)) ;
m13 = list(FRACTION(1,1),FRACTION(0,1),FRACTION(1,1),FRACTION(-11,37)) ;
m21 = list(FRACTION(1,2),FRACTION(-4,1),FRACTION(9,5)) ;
m22 = list(FRACTION(1,1),FRACTION(0,1)) ;
m23 = list(FRACTION(15,7),FRACTION(-4,3),FRACTION(1,10)) ;
m31 = list(FRACTION(13,15),FRACTION(0,1),FRACTION(-6,1)) ;
m32 = list(FRACTION(1,1),FRACTION(0,1),FRACTION(0,1),FRACTION(-3,5)) ;
m33 = list(FRACTION(-2,9),FRACTION(17,2)) ;
M = list(m11,m12,m13,m21,m22,m23,m31,m32,m33,3,3) ;
MP_ECRIRE(M)
D = MP_DETERMINANT(M) ;
P_ECRIRE(D)
```

Résultat :

$$D = -\frac{83}{14}x^8-\frac{30}{7}x^7+\frac{19}{5}x^6+\frac{2197}{777}x^5+\frac{2026571}{23310}x^4-\frac{26917}{1665}x^3-\frac{2656767}{5180}x^2+\frac{1119826}{925}x-\frac{852663}{1850}$$

Remarque

En faisant x = 0, on a : D(0) = - 852663/1850 qui est le déterminant de la matrice M(0)

$$M(0) = \begin{pmatrix} -2 & 29 & -\frac{11}{37} \\ \frac{9}{5} & 0 & \frac{1}{10} \\ -6 & -\frac{3}{5} & \frac{17}{2} \end{pmatrix}$$

On retrouve cette valeur en écrivant :

```
M0 = list(FRACTION(-2,1),FRACTION(29,1),FRACTION(-11,37),...
          FRACTION(9,5),FRACTION(0,1),FRACTION(1,10),...
          FRACTION(-6,1),FRACTION(-3,5),FRACTION(17,2),...
          3,3) ;
ECRIRE(M_DETERMINANT(M0))
```

EXEMPLE 2

Calculer l'inverse M^{-1} de la matrice M :

$$M=\begin{pmatrix} 3x^3+2x^2-2 & 29 & x^3+x-\frac{11}{37} \\ \frac{1}{2}x^2-4x+\frac{9}{5} & x & \frac{15}{7}x^2-\frac{4}{3}x+\frac{1}{10} \\ \frac{13}{15}x^2-6 & x^3-\frac{3}{5} & -\frac{2}{9}x+\frac{17}{2} \end{pmatrix}$$

Appliquons la fonction *MP_INVERSE* à la matrice M :

```
exec("Scilab-Matrices.sce",-1)
m11 = list(FRACTION(3,1),FRACTION(2,1),FRACTION(0,1),FRACTION(-2,1)) ;
m12 = list(FRACTION(29,1)) ;
m13 = list(FRACTION(1,1),FRACTION(0,1),FRACTION(1,1),FRACTION(-11,37)) ;
m21 = list(FRACTION(1,2),FRACTION(-4,1),FRACTION(9,5)) ;
m22 = list(FRACTION(1,1),FRACTION(0,1)) ;
m23 = list(FRACTION(15,7),FRACTION(-4,3),FRACTION(1,10)) ;
m31 = list(FRACTION(13,15),FRACTION(0,1),FRACTION(-6,1)) ;
m32 = list(FRACTION(1,1),FRACTION(0,1),FRACTION(0,1),FRACTION(-3,5)) ;
m33 = list(FRACTION(-2,9),FRACTION(17,2)) ;
M = list(m11,m12,m13,m21,m22,m23,m31,m32,m33,3,3) ;
[D,T] = MP_INVERSE(M) ;
P_ECRIRE(D), MP_ECRIRE(T)
```

On obtient : $M^{-1}=\frac{1}{D}\begin{pmatrix} P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 \\ P_7 & P_8 & P_9 \end{pmatrix}$

$$D=-\frac{83}{14}x^8-\frac{30}{7}x^7+\frac{19}{5}x^6+\frac{2197}{777}x^5+\frac{2026571}{23310}x^4-\frac{26917}{1665}x^3-\frac{2656767}{5180}x^2+\frac{1119826}{925}x-\frac{852663}{1850}$$

$$P_1=-\frac{15}{7}x^5+\frac{4}{3}x^4-\frac{1}{10}x^3+\frac{67}{63}x^2+\frac{77}{10}x+\frac{3}{50}$$

$$P_2=x^6+x^4-\frac{166}{185}x^3+\frac{263}{45}x-\frac{91139}{370}$$

$$P_3=-x^4+\frac{428}{7}x^2-\frac{4259}{111}x+\frac{29}{10}$$

$$P_4=\frac{13}{7}x^4-\frac{47}{45}x^3-\frac{112829}{6300}x^2+\frac{212}{5}x-\frac{159}{10}$$

$$P_5=-\frac{13}{15}x^5-\frac{2}{3}x^4+\frac{2717}{90}x^3+\frac{9578}{555}x^2+\frac{58}{9}x-\frac{695}{37}$$

$$P_6=-\frac{83}{14}x^5-\frac{30}{7}x^4+\frac{14}{3}x^3-\frac{163}{2590}x^2+\frac{179}{555}x-\frac{62}{185}$$

$$P_7=\frac{1}{2}x^5-4x^4+\frac{14}{15}x^3-\frac{3}{10}x^2+\frac{42}{5}x-\frac{27}{25}$$

$$P_8=-3x^6-2x^5+\frac{19}{5}x^3+\frac{79}{3}x^2-\frac{876}{5}$$

$$P_9=3x^4+2x^3-\frac{29}{2}x^2+114x-\frac{261}{5}$$

EXEMPLE 3

Trouver le polynôme minimal et le polynôme caractéristique de la matrice A suivante :

$$A = \begin{pmatrix} 2 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & -2 & 0 & 0 & 0 \\ 0 & 0 & -3 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -6 & -9 \\ 0 & 0 & 0 & 0 & 0 & 5 & -8 \end{pmatrix}$$

Le programme à exécuter est :

```
exec("Scilab-Matrices.sce",-1)
A = list(FRACTION(2,1),FRACTION(5,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),...
        FRACTION(0,1),FRACTION(2,1),FRACTION(0,2),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),...
        FRACTION(0,1),FRACTION(0,1),FRACTION(4,1),FRACTION(-2,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),...
        FRACTION(0,1),FRACTION(0,1),FRACTION(-3,1),FRACTION(5,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),...
        FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(7,1),FRACTION(0,1),FRACTION(0,1),...
        FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(-6,1),FRACTION(-9,1),...
        FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(0,1),FRACTION(5,1),FRACTION(-8,1),...
        7,7) ;
M_ECRIRE(A)
[PM,PC] = MP_MINIMAL(A) ;
P_ECRIRE(PM) , P_ECRIRE(PC)
```

Le polynôme minimal de A est :

$$PM(x) = x^5 + 3x^4 - 29x^3 - 603x^2 + 2584x - 2604$$

Le polynôme caractéristique de A est :

$$PC(x) = x^7 - 6x^6 - 42x^5 - 300x^4 + 7605x^3 - 34302x^2 + 59612x - 36456$$

Remarques

- On peut calculer PC(x) avec la méthode de **Le Verrier** et la fonction *M_POLYNOME*
  PC = M_POLYNOME(A) ; P_ECRIRE(PC)
  On obtient :
  $PC(x) = x^7 - 6x^6 - 42x^5 - 300x^4 + 7605x^3 - 34302x^2 + 59612x - 36456$

- On vérifie que PM(x) divise PC(x)
  [Q,R] = P_DIVISION(PC,PM) ; P_ECRIRE(Q), P_ECRIRE(R)
  On obtient :
  $Q(x) = x^2 - 9x + 14$, $R(x) = 0$

EXEMPLE 4

Vérifier que les 2 matrices A et B ci-dessous représentent le même opérateur linéaire :

$$A = \begin{pmatrix} 4 & 7 & -\frac{5}{2} & \frac{1}{9} \\ \frac{21}{11} & -\frac{31}{13} & 10 & \frac{2}{17} \\ \frac{15}{14} & -1 & 0 & -\frac{5}{2} \\ \frac{56}{31} & \frac{19}{4} & \frac{1}{7} & \frac{17}{23} \end{pmatrix}$$

$$B = \begin{pmatrix} -\frac{6390513}{248248} & -\frac{29985}{4774} & -\frac{27133727}{5123664} & \frac{6567530542\_7}{1747169424} \\ -\frac{444777}{17732} & -\frac{16613}{4774} & -\frac{9121631}{2561832} & \frac{3178088037\_5}{873584712} \\ \frac{7438903}{248248} & -\frac{3095}{4774} & \frac{8767523}{731952} & -\frac{4365710788\_1}{1747169424} \\ -\frac{4632099}{248248} & -\frac{6211}{4774} & -\frac{32516249}{5123664} & \frac{3424204908\_5}{1747169424} \end{pmatrix}$$

On écrit :

```
A = list(FRACTION(4,1),FRACTION(7,1),FRACTION(-5,2),FRACTION(1,9),...
        FRACTION(21,11),FRACTION(-31,13),FRACTION(10,1),FRACTION(2,17),...
        FRACTION(15,14),FRACTION(-1,1),FRACTION(0,1),FRACTION(-5,2),...
        FRACTION(56,31),FRACTION(19,4),FRACTION(1,7),FRACTION(17,23),...
        4,4) ;
B = list(FRACTION(-6390513,248248),FRACTION(-29985,4774),...
        FRACTION(-27133727,5123664),FRACTION(65675305427,1747169424),...
        FRACTION(-444777,17732),FRACTION(-16613,4774),...
        FRACTION(-9121631,2561832),FRACTION(31780880375,873584712),...
        FRACTION(7438903,248248),FRACTION(-3095,4774),...
        FRACTION(8767523,731952),FRACTION(-43657107881,1747169424),...
        FRACTION(-4632099,248248),FRACTION(-6211,4774),...
        FRACTION(-32516249,5123664),FRACTION(34242049085,1747169424),...
        4,4) ;
MA = MP_CARACTERISTIQUE(A) ;
[DA,PA,QA] = MP_INVARIANTS(MA) ;
MP_ECRIRE(DA)
MB = MP_CARACTERISTIQUE(B) ;
[DB,PB,QB] = MP_INVARIANTS(MB) ;
MP_ECRIRE(DB)
if MP_COMPARAISON(DA,DB) == 0 then
  printf('\n SEMBLABLES')
else
  printf('\n NON SEMBLABLES')
end
```

L'algorithme des facteurs invariants donne la même matrice diagonale $D = D_A = D_B$:

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & m \end{pmatrix}$$

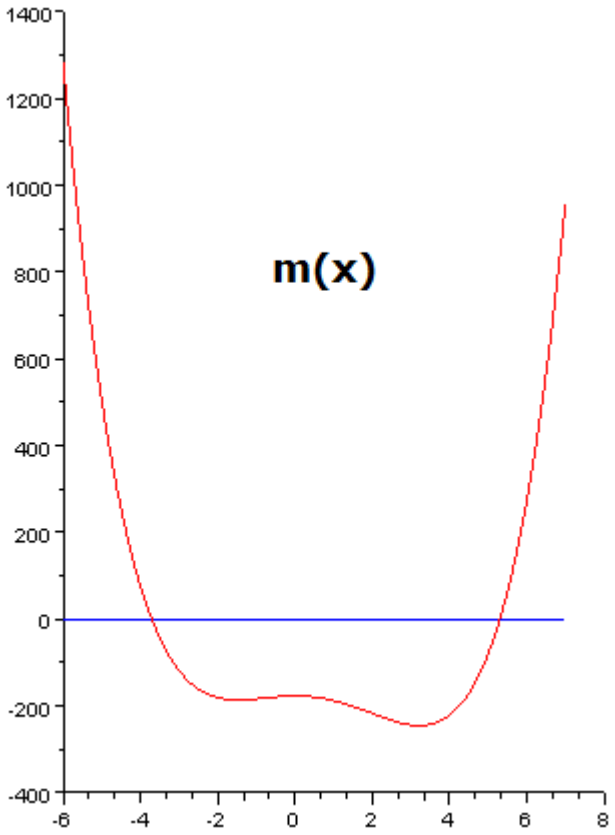
$$m = x^4 - \frac{704}{299}x^3 - \frac{4119801067}{436792356}x^2 + \frac{240880424}{764386623}x - \frac{166270831945}{940783536}$$

Donc, les matrices A et B ont les mêmes invariants de similitude. Elles sont semblables.

Remarques

Dans cet exemple, le polynôme minimal de A, ou de B, est égal au polynôme caractéristique : $m(x) = \Delta(x)$

La matrice de passage P telle que $B = P^{-1} A P$ est : $P = \begin{pmatrix} 1 & -1 & 0 & -1 \\ 2 & 0 & 1 & -1 \\ 0 & -1 & 1 & 3 \\ 0 & 0 & 1 & 1 \end{pmatrix}$



Mathématique & Informatique

Racines réelles d'un polynôme à coefficients rationnels

Introduction

On sait résoudre algébriquement $P(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0 = 0$ pour n = 1, 2, 3, 4
L'équation du premier degré (n = 1) est élémentaire.
Le principe de résolution de ax + b = 0 consiste à diviser 2 nombres : x = - b/a

$$3x - 4 = 0 \quad \Rightarrow \quad x = \frac{4}{3}$$

Les premières recherches pour trouver la solution d'une équation du second degré (n = 2) remontent à l'antiquité.
Le principe de résolution de $ax^2 + bx + c = 0$ consiste à regrouper les termes en x dans un carré, puis à étudier le signe du discriminant $\Delta = b^2 - 4ac$

$$5x^2 - 17x + 8 = 0 \quad \Rightarrow \quad x = \frac{17 \pm \sqrt{129}}{10}$$

Pour n = 3, l'équation a été étudiée par Gerolamo **Cardan**, au XVI^e siècle.
Le principe de résolution de $ax^3 + bx^2 + cx + d = 0$ consiste à mettre l'équation sous la forme simplifiée $X^3 + pX + q = 0$ avec le changement de variable x = X - b/3a, puis à effectuer un second changement de variable X = u + v si le discriminant $\Delta = 4p^3 + 27q^2$ est positif ou X = u cos v si le discriminant $\Delta = 4p^3 + 27q^2$ est négatif.

$$x^3 - 12x^2 + 45x - 49 = 0 \quad \Rightarrow \quad x = 4 + \sqrt[3]{\frac{-3 + \sqrt{5}}{2}} + \sqrt[3]{\frac{-3 - \sqrt{5}}{2}}$$

Pour n = 4, la découverte de la solution est attribuée à Ludovico **Ferrari**.
Le principe de résolution de $ax^4 + bx^3 + cx^2 + dx + e = 0$ consiste à introduire une variable auxiliaire y vérifiant une équation du troisième degré $y^3 + ry^2 + sy + t = 0$ dont la plus grande racine réelle permet de factoriser le polynôme de départ en 2 polynômes du second degré, puis à résoudre séparément chacune des 2 équations correspondantes.

$$x^4 + 4x^3 + 4x^2 + 12x - 7 = 0 \quad \Rightarrow \quad x = \frac{-2 - \sqrt{2} \pm \sqrt{2 + 12\sqrt{2}}}{2}$$

Au-delà de n = 4, les travaux de Niels **Abel** et d'Évariste **Galois** au XIX^e siècle ont montré l'impossibilité, en règle générale, d'exprimer les racines de l'équation P(x) = 0 à l'aide de radicaux.
Donc, sauf cas particuliers, une équation du type $a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 = 0$ ne peut être résolue que de manière numérique. Ajoutons que dans les applications concrètes, la formulation des racines à l'aide de radicaux (degré inférieur ou égal à 4) est insuffisante car il est nécessaire de disposer d'une valeur approchée des racines.
L'apparition des calculateurs électroniques au XX^e siècle a permis de mettre en œuvre des algorithmes donnant une approximation décimale précise des racines d'une équation polynomiale.

Mathématique & Informatique

Le problème

Il s'agit d'approcher avec précision les racines réelles d'un polynôme P à coefficients rationnels :

$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \quad \text{avec} \quad a_i \in \mathbb{Q}$
$P(x) = 0 \ ?$

Selon le même principe que celui utilisé pour la fonction *racines* qui calcule les racines entières d'un polynôme, écrivons une fonction *RACINES* qui permet d'obtenir les racines réelles de P(x) avec la précision que l'on veut.
La méthode consiste à séparer les racines, puis à les approximer par des grands nombres rationnels.

```
function x = RACINES(P,k,methode)
/
/
endfunction
```

Pour un polynôme **P** défini par ses coefficients, la fonction *RACINES* renvoie la liste **x** des approximations rationnelles des racines réelles de P avec une précision de **k** décimales en utilisant la méthode de **dichotomie** ou la méthode des tangentes de **Newton**.

On procède en 2 étapes :

1) on utilise la fonction Scilab *roots* pour obtenir une première approximation de chaque racine réelle de P(x)

2) on améliore le résultat précédent en appliquant la méthode de dichotomie ou la méthode des tangentes sur un intervalle où la racine est séparée

Séparation des racines

On suppose que les racines sont "suffisamment" éloignées les unes des autres et que la fonction *roots* fournit chaque racine r avec p chiffres exacts, sous la forme :

$$r = (\pm 0.d_1 d_2 \ldots d_p d_{p+1} \ldots d_n) \times 10^e$$

La racine réelle de P(x) appartient à l'intervalle I = [a,b], où a et b sont 2 nombres rationnels :

$$a = \frac{(\pm d_1 d_2 \ldots d_p - 1) \times 10^e}{10^p}$$

$$b = \frac{(\pm d_1 d_2 \ldots d_p + 1) \times 10^e}{10^p}$$

Par exemple, en prenant p = 10, on admet que la fonction *roots* donne 10 chiffres exacts d'une racine d'un polynôme. Evidemment, plus la précision finale demandée est grande, plus le temps de calcul est long.

Mathématique & Informatique

Méthode de dichotomie

L'algorithme de dichotomie est différent pour les 2 fonctions *racines* et *RACINES* : il s'effectue dans $\mathbb{Z}$ pour la première et dans $\mathbb{Q}$ pour la seconde.
Un des intérêts de la méthode de dichotomie est de pouvoir déterminer facilement la précision obtenue sur la racine.
Il suffit en effet que l'intervalle de recherche soit inférieur à ε pour que la racine soit connue à ε près.
La méthode de dichotomie ne suppose aucune hypothèse sur les dérivées de la fonction f à annuler, elle est donc générale et sûre. Rappelons cependant qu'elle ne convient pas si la fonction f ne change pas de signe dans l'intervalle [a,b] où se situe la racine r (cas d'une racine annulant f(x) et sa dérivée $f'(x)$). Autre désavantage de la méthode de dichotomie : elle nécessite un volume de calcul important pour obtenir une grande précision.

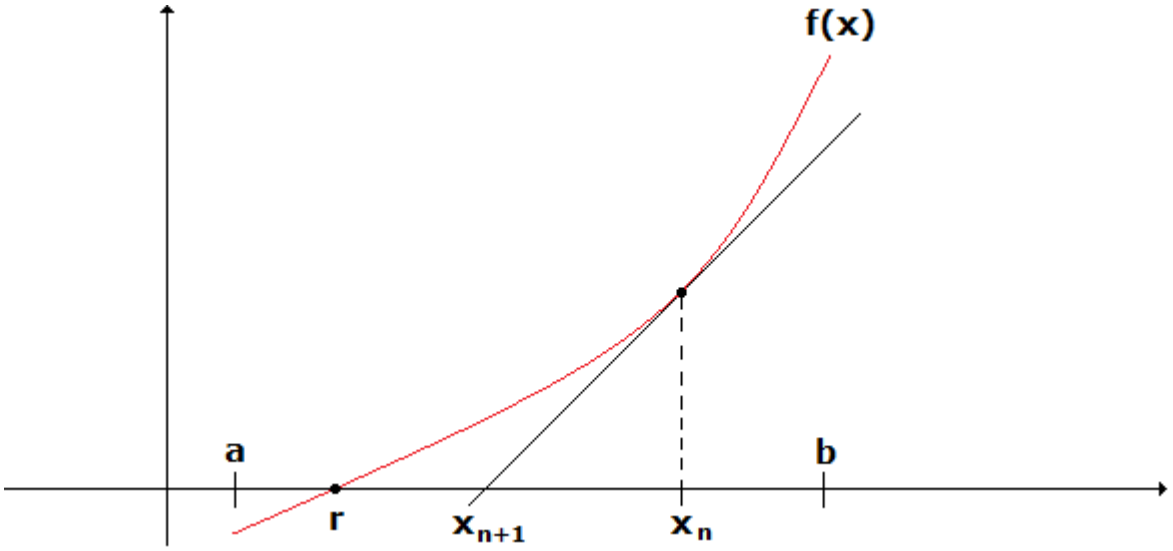
Méthode des tangentes de **Newton**

La fonction f possède une racine r séparée dans l'intervalle [a, b]

La suite $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ converge vers r

Initialisation : $x_0 = a$ si $f(a)f''(a) > 0$, $x_0 = b$ sinon

Convergence : $|r - x_{n+1}| \le \frac{M}{2m}|r - x_n|^2$ avec $M = \underset{[a,b]}{\text{Max}}|f''(x)|$ et $m = \underset{[a,b]}{\text{Min}}|f'(x)|$



La précision obtenue sur la racine est plus difficile à établir. Par exemple, les tests $|x_{n+1} - x_n| < \varepsilon$ et $|f(x_n)| < \varepsilon$ ne garantissent pas à eux seuls que x_n soit très proche de r sans hypothèses supplémentaires sur f.
La méthode de Newton est mise en œuvre lorsque $f'(x)$ et $f''(x)$ gardent des signes constants sur [a,b], ce qui est généralement le cas lorsque l'intervalle de séparation de la racine est étroit. Elle est d'autant plus efficace que la pente de la courbe, c'est-à-dire $f'(x)$, est élevée au voisinage de r. En particulier, si la constante M/2m est plus petite que 1 et si le terme x_n vérifie $|r - x_n| < 10^{-k}$, alors on a la majoration de l'erreur $|r - x_{n+1}| < 10^{-2k}$ au rang n+1 et il est clair que le nombre de décimales exactes double à chaque étape ; on dit que la convergence est quadratique.
On peut aussi commencer par une estimation grossière de la racine avec la méthode de dichotomie et affiner le résultat avec la méthode des tangentes.

Note

Il existe d'autres méthodes : méthode des cordes, méthode du point fixe, ... En fait, il n'y a pas de méthode idéale, chacune a ses avantages et ses inconvénients, et se révèle la mieux adaptée dans tel ou tel cas particulier.

Programmation des fonctions DICHOTOMIE et NEWTON

Le rationnel r est la valeur approchée à ε près de la racine de l'équation polynomiale P(x) = 0 dans l'intervalle [a,b]

```
function r = DICHOTOMIE(P,a,b,eps)
  m = b
  y1 = P_HORNER(P,a)
  loop = %t
  while loop
    y2 = P_HORNER(P,m)
    select F_SIGNE(y1)*F_SIGNE(y2)
      case 0
        if F_SIGNE(y1) == 0 then
          r = a
        else
          r = m
        end
        return
      case +1
        a = m
        y1 = y2
      case -1
        b = m
    end
    m = F_DIVISION(F_ADDITION(a,b),FRACTION(2,1))
    delta = F_ABSOLU(F_SOUSTRACTION(b,a))
    loop = F_COMPARAISON(delta,eps) >= 0
  end
  r = m
endfunction
```

```
function r = NEWTON(P,a,b,eps)
  DP = P_DERIVEE(P)
  D2P = P_DERIVEE(DP)
  ya = P_HORNER(P,a)
  yb = P_HORNER(P,b)
  ya2p = P_HORNER(D2P,a)
  if F_SIGNE(ya)*F_SIGNE(ya2p) >= 0 then
    x = a ; y = ya
  else
    x = b ; y = yb
  end
  loop = %t
  while loop
    yp = P_HORNER(DP,x)
    h = F_DIVISION(y,yp)
    x = F_SOUSTRACTION(x,h)
    y = P_HORNER(P,x)
    delta = F_ABSOLU(y)
    loop = F_COMPARAISON(delta,eps) >= 0
  end
  r = x
endfunction
```

Programmation de la fonction RACINES

```
function x = RACINES(P,k,methode)

  function Y = tri(X)
    Y = []
    for i = 1:length(X)
    if imag(X(i)) == 0 then
      Y = [Y ; real(X(i))]
     end
    end
    Y = unique(Y)
  endfunction

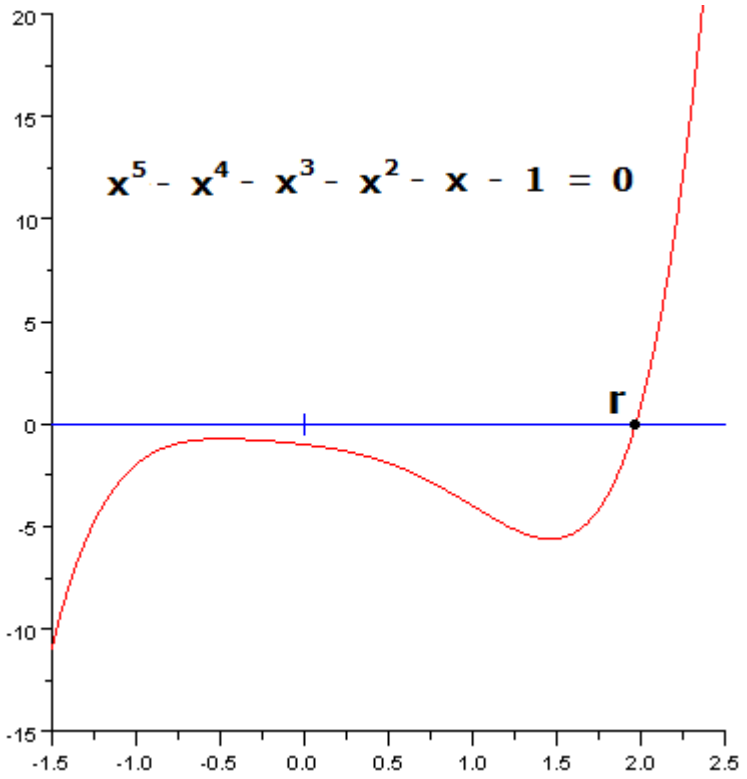
  function I = intervalle(X)
    I = list()
    for i = 1:length(X)
      if X(i) <> 0 then
        e = floor(log10(abs(X(i)))) + 1
        E = int(X(i)*10^(p-e))
        if e > = 0 then
          a = list([ENTIER(E-1),zeros(1,e)],[1,1,zeros(1,p)])
          b = list([ENTIER(E+1),zeros(1,e)],[1,1,zeros(1,p)])
        else
          a = list(ENTIER(E-1),[1,1,zeros(1,p-e)])
          b = list(ENTIER(E+1),[1,1,zeros(1,p-e)])
        end
        I($+1) = list(a,b)
      else
        I($+1) = list(FRACTION(0,1),FRACTION(0,1))
      end
    end
  endfunction

  p = 10
  P = P_SIMPLIFICATION(P)
  eps = list([1,1],[1,1,zeros(1,k)])
  x = poly(0,'x')
  polynome = 0
  for i = 1:length(P)
    polynome = polynome + nombre_rationnel(P(i))*x^(length(P)-i)
  end
  X = roots(polynome,'e')
  X = tri(X)
  I = intervalle(X)
  x = list()
  for i = 1:length(I)
    a = I(i)(1) ; b = I(i)(2)
    select methode
      case 'dichotomie', r = DICHOTOMIE(P,a,b,eps)
      case 'newton', r = NEWTON(P,a,b,eps)
      else, r = []
    end
    if r <> [] then, x($+1) = r, end
  end
endfunction
```

Mathématique & Informatique

EXEMPLE 1

Résolvons l'équation $x^5 - x^4 - x^3 - x^2 - x - 1 = 0$ à 10^{-54} près.



Exécutons le module *Scilab-Equations* pour pouvoir appeler la fonction *RACINES* :

exec("Scilab-Equations.sce",-1)

Le polynôme correspondant à l'équation P(x) = 0 est représenté par :

P = list(FRACTION(1,1),FRACTION(-1,1),FRACTION(-1,1),FRACTION(-1,1),FRACTION(-1,1),FRACTION(-1,1))

Calculons l'approximation rationnelle de la racine réelle de P avec la précision imposée en utilisant la méthode de dichotomie :

x = RACINES(P,54,'dichotomie')

Affichons l'expression décimale de cette racine :

ECRIRE(DECIMALE(x(1),54))

On obtient :

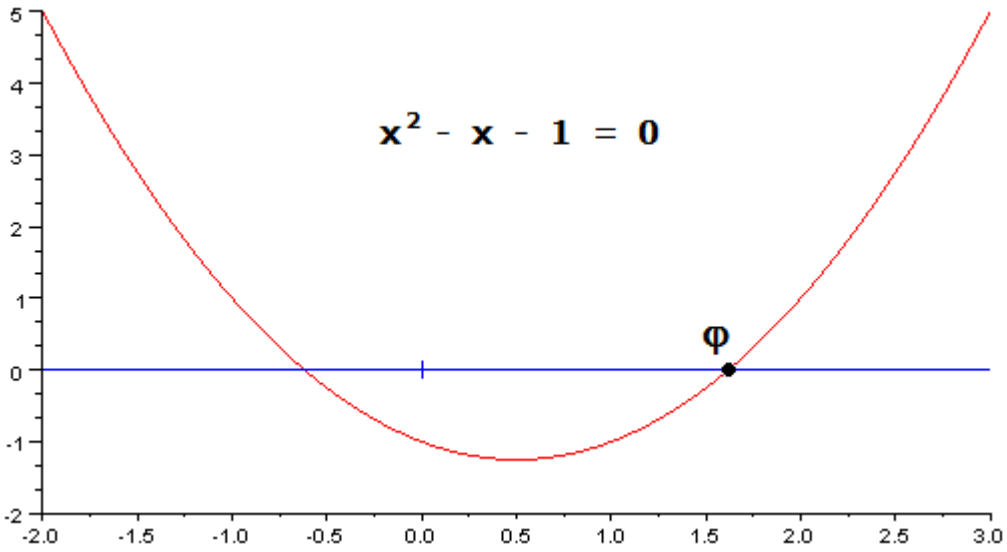
r = 1.965948236645485337189937375934401396151327177456861393

EXEMPLE 2

Calculons le nombre d'or avec une précision de 100 décimales.

Le nombre d'or φ est la racine positive de l'équation du second degré : $x^2 - x - 1 = 0$

```
x = linspace(-2,3,300) ;
y = x^2 - x - 1 ;
plot2d(x,y,5)
plot2d([-2,3],[0,0],2)
```



Pour obtenir les 100 premières décimales de φ, on exécute le programme :

```
exec("Scilab-Equations.sce",-1)
P = list(FRACTION(1,1),FRACTION(-1,1),FRACTION(-1,1)) ;
x = RACINES(P,100,'newton') ;
ECRIRE(DECIMALE(x(2),100))
```

Résultat :

φ = 1.6180339887498948482045868343656381177203091798057628621354486227052604628189024497072072041893911374

Mathématique & Informatique

EXEMPLE 3

Donnons une approximation de la valeur de $cos\frac{\pi}{7}$ avec 30 décimales exactes.

En général, on ne connait pas d'expression algébrique de $\cos\frac{\pi}{n}$. Par contre, on montre que l'on peut obtenir une équation équation polynomiale vérifiée par $\cos\frac{\pi}{n}$. Plus précisément, en alternant les signes des nombres situés sur la n^e diagonale du triangle de **Pascal**, on obtient les coefficients entiers d'un polynôme de degré $p = E\left(\frac{n-1}{2}\right)$ dont les p racines réelles distinctes sont $4cos^2\frac{k\pi}{n}$ $(k = 1, \dots, p)$.

⇨ **http://www.gecif.net/articles/mathematiques/trianglepascal/** (**Jean-Christophe Michel**)

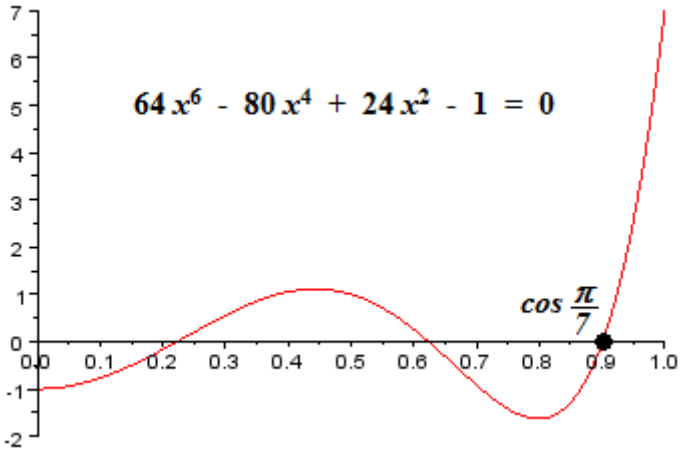
1									
1	1								
1	2	1							
1	3	3	1						
1	4	6	4	1					
1	5	10	10	5	1				
1	6	15	20	15	6	1			
1	7	21	35	35	21	7	1		
1	8	28	56	70	56	28	8	1	
1	9	36	84	126	126	84	36	9	1

a	*b*
	a+b

Par exemple, avec la 7e diagonale du triangle de **Pascal**, on obtient le polynôme : $x^3 - 5x^2 + 6x - 1 = 0$

Ce polynôme de degré 3 admet pour racines $x_1 = 4cos^2\frac{\pi}{7}$, $x_2 = 4cos^2\frac{2\pi}{7}$, $x_3 = 4cos^2\frac{3\pi}{7}$

Par suite $cos\frac{\pi}{7}$ est une racine réelle de : $64x^6 - 80x^4 + 24x^2 - 1 = 0$



Programme d'approximation de $cos\frac{\pi}{7}$ par dichotomie sur l'intervalle [0.85,0.95] avec une précision de 10^{-30} :

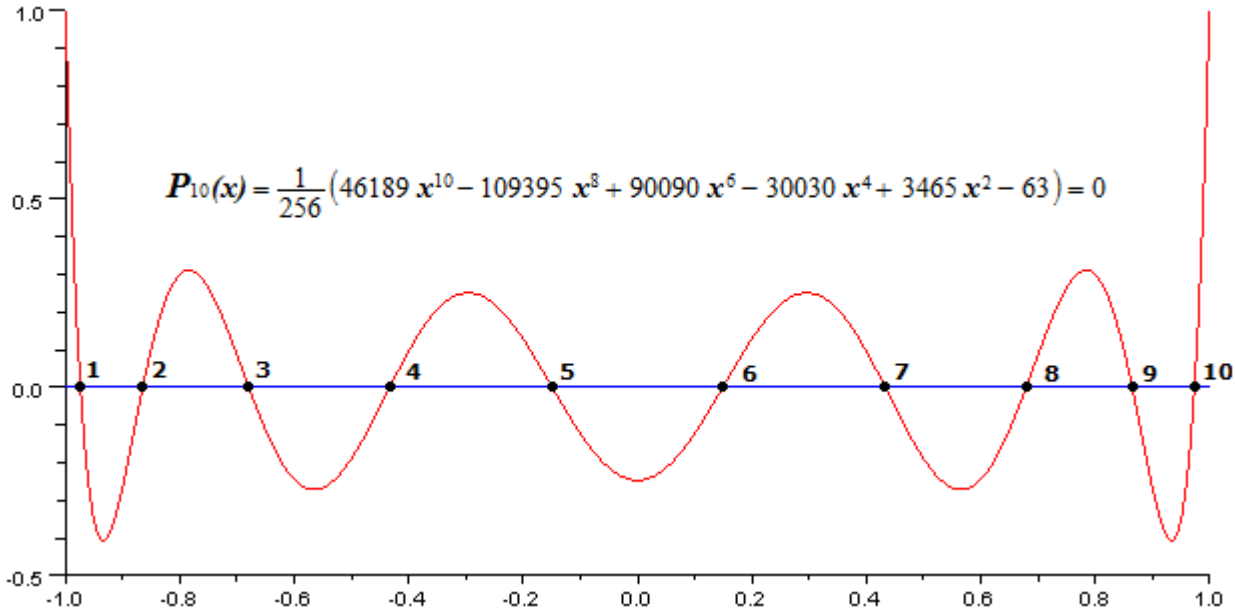
```
exec("Scilab-Equations.sce",-1)
P = list(FRACTION(64,1),FRACTION(0,1),FRACTION(-80,1), FRACTION(0,1), FRACTION(24,1), ...
         FRACTION(0,1),FRACTION(-1,1)) ;
a = list([1,8,5],[1,1,0,0]) ; b = list([1,9,5],[1,1,0,0]) ; eps = list([1,1],[1,1,zeros(1,30)]) ;
r = DICHOTOMIE(P,a,b,eps) ;
ECRIRE(DECIMALE(r,30))
```

Valeur obtenue : $cos\frac{\pi}{7} = 0.900968867902419126236102319507$

Remarque : il existe une équation algébrique plus simple qui admet $cos\frac{\pi}{7}$ comme racine : $8x^3 - 4x^2 - 4x + 1 = 0$

EXEMPLE 4

Quelles sont à 10^{-16} près les 10 racines du polynôme de **Legendre** de degré 10 ?



On écrit :

```
P = P_LEGENDRE(10) ;
K = list(P(1)(2),[1,1]) ;
Q = P_SCALAIRE(P,K) ;
x = RACINES(Q,16,'dichotomie') ;
for i = 1:length(x)
   printf('\n x%i =',i)
   ECRIRE(DECIMALE(x(i),16))
end
```

On obtient :

```
x1 = - 0.9739065285171717
x2 = - 0.8650633666889844
x3 = - 0.6794095682990244
x4 = - 0.4333953941292472
x5 = - 0.1488743389816312
x6 =   0.1488743389816312
x7 =   0.4333953941292472
x8 =   0.6794095682990244
x9 =   0.8650633666889844
x10 = 0.9739065285171717
```

EXEMPLE 5

Recherchons les solutions de l'équation du troisième degré :

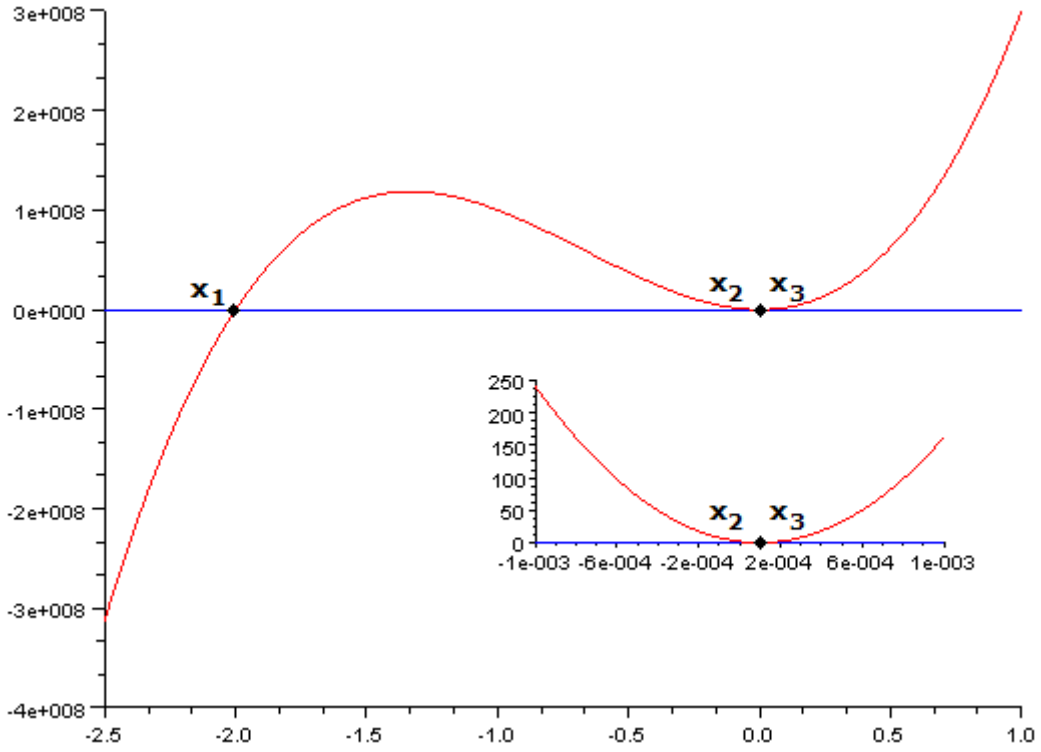
$$100000000\,x^3 + 199980000\,x^2 - 39999\,x + 2 = 0$$

Utilisons la fonction *RACINES* avec la méthode de **Newton** pour approximer les racines de P(x) = 0 :

```
exec("Scilab-Equations.sce",-1)
P = list(FRACTION(100000000,1),FRACTION(199980000,1),FRACTION(-39999,1),FRACTION(2,1)) ;
x = RACINES(P,16,'newton') ;
for i = 1:length(x)
   printf('\n x%i = ',i), ECRIRE(DECIMALE(x(i),16))
end
```

Pour chaque racine x_i , le test d'arrêt est : $|P(x_i)| < 10^{-16}$. On trouve :

```
x1 = - 2.0000000000000000
x2 =   0.0000999999995950
x3 =   0.0001000000003000
```



On vérifie que les solutions de cette équation sont exactement -2 (racine simple) et 1/10000 (racine double).
Les calculs n'ont pas permis d'établir avec certitude que l'équation possédait une racine double $x_2 = x_3 = 1/10000$.
La racine double ne peut pas être obtenue avec la méthode de dichotomie puisque P(x) ne change pas de signe dans l'intervalle de séparation. La dérivée P'(x) au voisinage de la racine double étant proche de 0 (courbe presque horizontale), la méthode de **Newton** converge très lentement vers la solution et on ne peut guère envisager une grande précision par ce procédé ...

EXEMPLE 6

Calculons avec une précision de 32 décimales les valeurs propres réelles de la matrice A suivante :

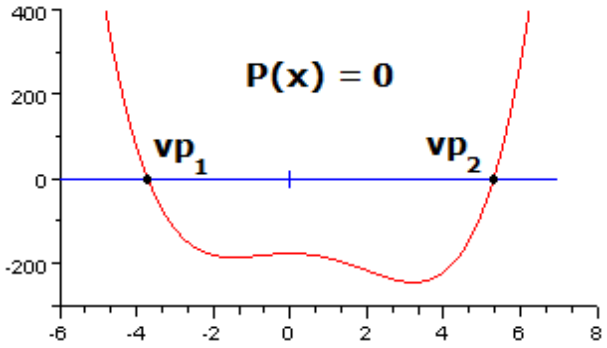
$$A = \begin{pmatrix} 4 & 7 & -\frac{5}{2} & \frac{1}{9} \\ \frac{21}{11} & -\frac{31}{13} & 10 & \frac{2}{17} \\ \frac{15}{14} & -1 & 0 & -\frac{5}{2} \\ \frac{56}{31} & \frac{19}{4} & \frac{1}{7} & \frac{17}{23} \end{pmatrix}$$

Le programme Scilab ci-dessous détermine l'expression exacte du polynôme caractéristique P de A en appelant la fonction *M_POLYNOME*. L'application de la fonction *RACINES* à P avec la précision k = 32 donne les racines rationnelles recherchées ; elles sont converties en nombres décimaux en utilisant la fonction *DECIMALE*.

```
exec("Scilab-Equations.sce",-1)
A = list(FRACTION(4,1),FRACTION(7,1),FRACTION(-5,2),FRACTION(1,9),...
         FRACTION(21,11),FRACTION(-31,13),FRACTION(10,1),FRACTION(2,17),...
         FRACTION(15,14),FRACTION(-1,1),FRACTION(0,1),FRACTION(-5,2),...
         FRACTION(56,31),FRACTION(19,4),FRACTION(1,7),FRACTION(17,23),...
         4,4) ;
P = M_POLYNOME(A) ; P_ECRIRE(P)
k = 32 ;
x = RACINES(P,k,'dichotomie') ;
for i = 1:length(x)
   vp = DECIMALE(x(i),k) ;
   printf('\n vp%i = ',i), ECRIRE(vp)
end
```

Les racines de $P(x) = x^4 - \frac{704}{299}x^3 - \frac{4119801067}{436792356}x^2 + \frac{240880424}{764386623}x - \frac{166270831945}{940783536} = 0$ sont :

$vp_1 = -\ 3.70151073172108172233853064291812$
$vp_2 = \ \ 5.30501511868573898050238674667618$



A propos des racines multiples

Soit le polynôme P(x) à coefficients rationnels :

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0 \qquad (a_i \in \mathbb{Q})$$
$$= a_n (x - x_1)^{\alpha 1} (x - x_2)^{\alpha 2} \ldots (x - x_m)^{\alpha m} \qquad (x_i \in \mathbb{C})$$

Le polynôme dérivé de P(x) est :

$$P'(x) = n a_n x^{n-1} + (n-1) a_{n-1} x^{n-2} + ... + a_1$$
$$= (x - x_1)^{\alpha 1 - 1} (x - x_2)^{\alpha 2 - 1} \ldots (x - x_m)^{\alpha m - 1} B(x)$$
$$= A(x)\, B(x)$$

Comme $B(x_k) \neq 0$, $1 \leq k \leq m \leq n$, on en déduit :

$$A(x) = pgcd(P(x), P'(x))$$

Notons Q(x) le polynôme ayant les mêmes racines que P(x) mais avec l'ordre de multiplicité $\alpha_i = 1$:

$$Q(x) = b_m (x - x_1)(x - x_2) \ldots (x - x_m)$$
$$= b_m x^m + b_{m-1} x^{m-1} + ... + b_1 x + b_0$$

On a :

$$Q(x) = \frac{P(x)}{pgcd(P(x), P'(x))}$$

En particulier, si Q(x) = P(x), c'est-à-dire si $pgcd(P(x), P'(x)) = 1$, le polynôme P(x) ne possède pas de racine multiple.

Application

Reprenons l'exemple 5 précédent :

$$P(x) = 100000000\,x^3 + 199980000\,x^2 - 39999\,x + 2 = 0$$

Recherchons si P(x) admet des racines multiples :

```
exec("Scilab-Polynomes.sce",-1)
P = list(FRACTION(100000000,1),FRACTION(199980000,1),FRACTION(-39999,1),FRACTION(2,1)) ;
PP = P_DERIVEE(P) ;
A = P_PGCD(P,PP) ;
[Q,R] = P_DIVISION(P,A) ;
P_ECRIRE(Q)
```

On obtient :

$$Q(x) = 100000000\,x^2 + 199990000\,x - 20000 = 0$$

Comme $Q(x) \neq P(x)$, le polynôme P(x) possède au moins une racine multiple, en l'occurrence P(x) a une racine double. Le calcul des racines (simples) de Q(x) donne :

```
x1 = -  2.000000000000
x2 =    0.000100000000
```

Mathématique & Informatique

A propos de la précision dans les calculs

En analyse numérique, la précision est un critère déterminant. Si elle est insuffisante, les solutions trouvées peuvent être très éloignées de la réalité, voire fausses ... Plus la précision demandée est grande, plus les ordinateurs doivent disposer de processeurs rapides et d'une importante capacité de calcul.
Prenons par exemple un polynôme $P(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$ dont on cherche les racines. La précision sur les a_i est parfois critique pour trouver les solutions de $P(x) = 0$. Ainsi, il arrive qu'une très petite variation sur un seul des coefficients a_i induit des différences significatives sur les résultats.

Illustration

⇨ http://www.enseignement.polytechnique.fr/profs/informatique/Eric.Schost/TD3/enonce.html
$W_{15}(x)$ est le polynôme de Wilkinson de degré 15. Ses coefficients sont des entiers parfaitement déterminés et ses racines sont par construction les entiers naturels de 1 à 15.

Polynôme $W_{15}(x)$:
==========
$1\ x^{15}$
$-\ 120\ x^{14}$
$6580\ x^{13}$
$-\ 218400\ x^{12}$
$4899622\ x^{11}$
$-\ 78558480\ x^{10}$
$928095740\ x^9$
$-\ 8207628000\ x^8$
$54631129553\ x^7$
$-\ 272803210680\ x^6$
$1009672107080\ x^5$
$-\ 2706813345600\ x^4$
$5056995703824\ x^3$
$-\ 6165817614720\ x^2$
$4339163001600\ x$
$-\ 1307674368000$

Considérons le polynôme $P(x) = W_{15}(x) + x^9$
P(x) diffère peu de $W_{15}(x)$ puisque seul son coefficient a_9 est supérieur d'une unité :

$a_9 = 928095740$ pour $W_{15}(x)$
$a_9 = 928095741$ pour $P(x)$

La variation de a_9 est d'environ 10^{-7} % entre les 2 polynômes, et pourtant les racines r_i de P(x) sont globalement différentes de celles de $W_{15}(x)$. Avec la fonction *roots* on trouve les valeurs approximatives suivantes :

r_1 =	1.	r_6 =	6.2765185 - 0.2869678i	r_{11} =	10.373766 + 1.8115162i
r_2 =	2.0000001	r_7 =	6.2765185 + 0.2869678i	r_{12} =	12.907505 - 1.5711101i
r_3 =	2.9999795	r_8 =	8.0712687 - 1.2382119i	r_{13} =	12.907505 + 1.5711101i
r_4 =	4.0010987	r_9 =	8.0712687 + 1.2382119i	r_{14} =	14.881009 - 0.4860167i
r_5 =	4.9787854	r_{10} =	10.373766 - 1.8115162i	r_{15} =	14.881009 + 0.4860167i

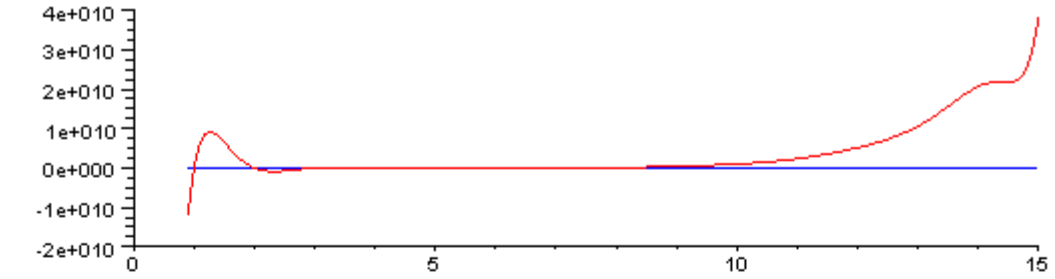
Les nombres r_1, r_2, r_3, r_4, r_5 sont très proches de 1, 2, 3, 4, 5 mais les racines réelles 6, 7, 8, 9, ... , 15 ont disparu !

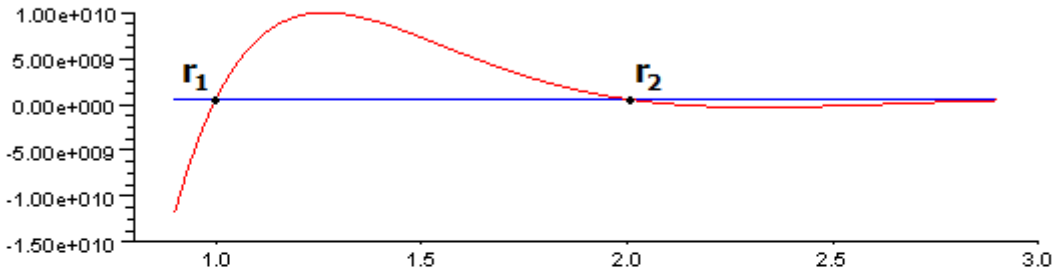
Conséquence

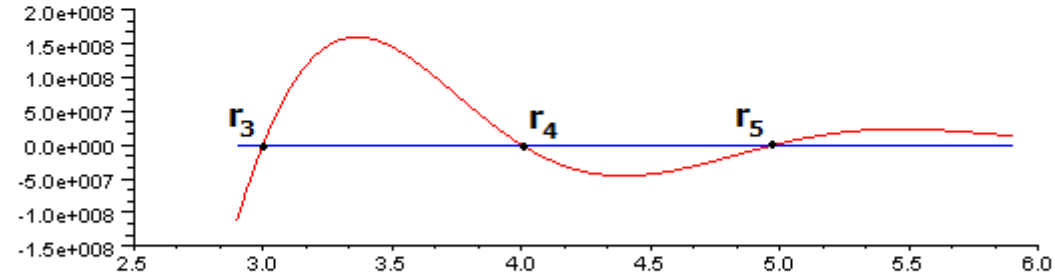
Lorsque les données nécessaires à un calcul sont issues de mesures physiques d'un phénomène de la nature, il faut s'assurer que leur précision est suffisante pour obtenir un résultat fiable.

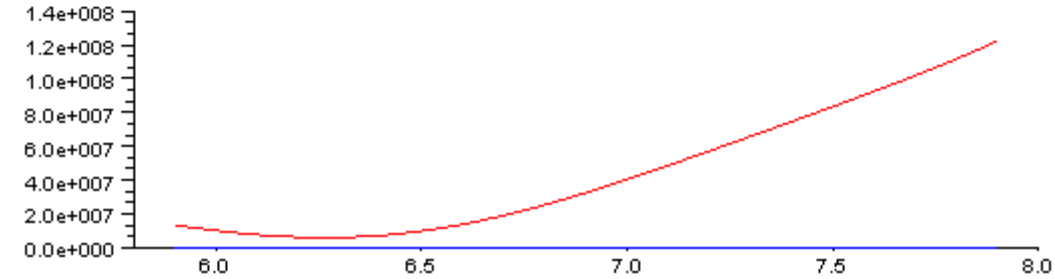
Mathématique & Informatique

```
W15 = P_WILKINSON(15) ; P_ECRIRE(W15)
X9 = P_MONOME(FRACTION(1,1),9) ;
P = P_ADDITION(W15,X9) ; P_ECRIRE(P)
x = poly(0,'x') ;
polynome = 0 ;
for i = 1:length(P)
   polynome = polynome + nombre_rationnel(P(i))*x^(length(P)-i) ;
end
printf('\n ========================= \n')
r = roots(polynome,'e')
subplot(4,1,1), P_GRAPHIQUE(P,0.9,15)
subplot(4,1,2), P_GRAPHIQUE(P,0.9,2.9)
subplot(4,1,3), P_GRAPHIQUE(P,2.9,5.9)
subplot(4,1,4), P_GRAPHIQUE(P,5.9,7.9)
```









Mathématique & Informatique

COURBES ELLIPTIQUES

D'une manière générale, une courbe elliptique est définie par l'ensemble des points (x,y) tels que :

$$y^2 + \alpha xy + \beta y = x^3 + \gamma x^2 + \delta x + \varepsilon$$

Sur $\mathbb{R}$, cette équation est celle d'une courbe plane. On l'exprime sous la forme simplifiée de **Weierstrass** :

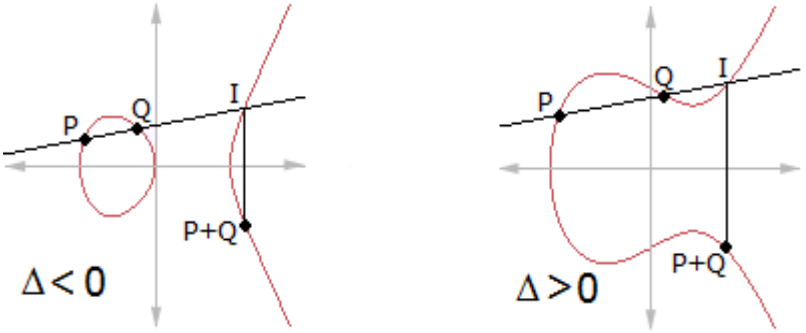
$$y^2 = x^3 + ax + b$$

où le discriminant de l'équation du troisième degré $x^3 + ax + b$ est non nul : $\Delta = 4a^3 + 27b^2 \neq 0$

Note : L'ellipse n'est pas une courbe elliptique.

Définition de l'addition de 2 points

Soient P et Q deux points distincts de la courbe, le point noté P+Q est le symétrique par rapport à l'axe des x du point I d'intersection de la droite (PQ) avec la courbe. Schématiquement :



Prenons : $P(x_p,y_p)$ et $Q(x_q,y_q)$. Les coordonnées de $P+Q(x_{p+q},y_{p+q})$ sont données par :

$$x_{p+q} = \left(\frac{y_p - y_q}{x_p - x_q}\right)^2 - x_p - x_q$$

$$y_{p+q} = \left(\frac{y_p - y_q}{x_p - x_q}\right)\left(x_p - x_{p+q}\right) - y_p$$

Si la droite (PQ) est parallèle à l'axe des ordonnées, c'est-à-dire si $x_p = x_q$, on convient que le point P+Q est à l'infini, et on le note 0_E ou plus simplement 0 (attention à ne pas le confondre avec l'origine des axes du repère).
Le point 0 est l'élément neutre pour l'addition des points : P+0 = 0+P = P
Le symétrique d'un point P de coordonnées (x_p,y_p) est le point noté -P de coordonnées $(x_p,-y_p)$
Dans le cas où P = Q, on prend la tangente à la courbe en P. Géométriquement, le point P+P = 2P s'obtient en traçant 2 droites comme précédemment. Les coordonnées du point $2P(x_{2p},y_{2p})$ s'expriment en fonction de celles de $P(x_p,y_p)$:

$$x_{2p} = \left(\frac{3x_p^{\,2} + a}{2y_p}\right)^2 - 2x_p$$

$$y_{2p} = \left(\frac{3x_p^{\,2} + a}{2y_p}\right)\left(x_p - x_{2p}\right) - y_p$$

Propriété

L'ensemble des points d'une courbe elliptique muni de la loi + est un groupe commutatif.

- Commutativité : $P + Q = Q + P$
- Associativité : $(P + Q) + R = P + (Q + R)$
- Elément neutre : $P + 0 = 0 + P = P$
- Symétrique : $P + (-P) = (-P) + P = 0$

L'associativité de la somme de 3 points peut être établie directement en calculant les coordonnées des points.
L'intérêt de cette propriété est de pouvoir appliquer les résultats de la théorie des groupes aux courbes elliptiques.

Théorème de **Siegel**

L'ensemble des points P de coordonnées entières est fini.

Exemple

Les courbes elliptiques ont servi à prouver le dernier théorème de **Fermat** : il n'existe pas d'entiers naturels x, y, z non nuls tels que $x^p + y^p = z^p$ pour $p \geq 3$. La démonstration repose sur la contradiction suivante : si l'équation de **Fermat** possédait une solution $a^p + b^p = c^p$, alors la courbe elliptique $y^2 = x(x - a^p)(x + b^p)$ ne vérifierait pas la conjecture de **Taniyama-Shimura-Weil** démontrée par Andrew **Wiles** en 1994.

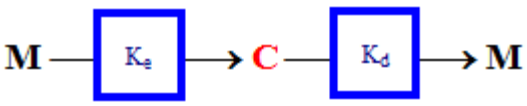
Mathématique & Informatique

CRYPTOLOGIE

Sécurité ⇒ *CAIN* = *Confidentialité, Authentification, Intégrité et Non-répudiation*.
Lorsque 2 personnes, traditionnellement Alice et Bob, souhaitent communiquer en toute confidentialité, elles s'échangent des messages chiffrés, c'est-à-dire incompréhensibles par une tierce personne.
La *cryptographie* consiste à transformer un message clair M en un message chiffré C, de sorte qu'il soit très difficile de retrouver M à partir de C, sans renseignements complémentaires.

$$\textit{Relation fondamentale} : \begin{cases} \boldsymbol{C = E(M)} \\ \boldsymbol{M = D(C)} \end{cases}$$

Les fonctions d'encodage E et de décodage D utilisent les paramètres respectifs K_e et K_d (Keys en anglais) appelés "clé de chiffrement" et "clé de déchiffrement". Les différentes méthodes de *cryptanalyse* (exhaustive, fréquentielle, différentielle ...) ne doivent pas permettre d'inverser directement E ou D, ni de découvrir K_e ou K_d .
Pour renforcer la sécurité, on augmente la taille de la clé : idéalement $|K| \geq |M|$ (système de **Vernam**).
En pratique, selon l'algorithme de chiffrement utilisé, la longueur d'une clé "sûre" peut varier de 128 à 2048 bits.



M : message clair
K_e : clé de chiffrement
C : message chiffré
K_d : clé de déchiffrement

Deux possibilités :

1) $K_e = K_d$
L'algorithme de chiffrement est dit **symétrique** (ou à clé secrète).
Principe
Alice et Bob disposent en commun d'une *clé secrète* : $K = K_e = K_d$
Bob utilise K pour envoyer des messages chiffrés à Alice qui les déchiffrera grâce à cette même clé K.
NB.
- Bob et Alice doivent convenir en secret d'une clé de chiffrement commune.
- Si Bob divulgue la clé, la confidentialité du message chiffré est perdue.

Exemple
Algorithme **A.E.S.** (Advanced Encryption Standard) : repose sur la transformation non linéaire de blocs.
⇨ Application : **connexion Wi-Fi**
Le protocole WEP utilise RC4 avec une clé $Hexa^{26}$; il peut être cassé en quelques secondes.
Son successeur WPA2 utilise AES avec une clé $Hexa^{32}$; il est actuellement considéré comme sûr.

2) $K_e \neq K_d$
L'algorithme de chiffrement est dit **asymétrique** (ou à clé publique).
Principe
Alice dispose en propre d'un couple *clé publique/clé privée* : K_e / K_d
Elle communique sa clé publique K_e à tous ceux qui souhaitent lui adresser des messages ; souvent K_e est publiée dans un annuaire. Par contre Alice garde secrète sa clé privée K_d .
Bob utilise K_e pour envoyer des messages chiffrés à Alice qui sera la seule à pouvoir les déchiffrer grâce à sa clé K_d .
NB.
- Bob et Alice n'ont pas besoin de convenir en secret d'une clé de chiffrement commune.
- Si Bob divulgue la clé, la confidentialité du message chiffré est conservée.

Exemple
Algorithme **R.S.A.** (**Rivest — Shamir — Adleman**) : repose sur l'arithmétique des grands nombres.
⇨ Application : **signature numérique**
Pour authentifier un document numérique N, on lui adjoint une donnée annexe nommée signature S.
L'origine et l'intégrité de N sont vérifiables à l'aide d'un algorithme utilisant N, S et une clé publique K :
$S = D_{Kprivée}[H(N)] \Rightarrow H(N) \equiv E_{Kpublique}(S)$ avec la fonction de hachage $H : \{0,1\}^* \longrightarrow \{0,1\}^m$

Mathématique & Informatique

Un nouvel algorithme pour factoriser des nombres entiers

Soit un entier n représenté par le vecteur de ses chiffres : $n = [n_k , \dots , n_i , \dots , n_1]$
Le but est de trouver 2 nombres u et v tels que : $n = u\ v$
Habituellement, on recherche un diviseur de n parmi les nombres premiers inférieurs à $\sqrt{n}$
Etudions une autre méthode qui évite de connaître ces nombres premiers. L'algorithme consiste à multiplier 2 nombres a et b en ajoutant un chiffre à chaque étape de sorte que le résultat final donne n.
La fonction "**diviseur**" suivante retourne 2 diviseurs u et v de n (si n est premier alors u = v = 0) :

```
fonction  [u,v] = diviseur(a,b)
|
|    u = 0 , v = 0
|
|    • On connaît les (i-1) derniers chiffres de a et b
|       a = [a(i-i) , ... , a2 , a1]
|       b = [b(i-i) , ... , b2 , b1]
|
|    • On crée la liste des i^e chiffres possibles
|       pour ai = 0 à 9
|       |     pour bi = 0 à 9
|       |     |     a' = [ai,a]
|       |     |     b' = [bi,b]
|       |     |     c' = a' × b'
|       |     |     si ci' = ni alors ajouter [ai,bi] à la liste
|       |     fin
|       fin
|
|    • On effectue des appels récursifs si n n'est pas factorisable
|       pour [ai,bi] dans la liste
|       |     si a'|n ou b'|n alors
|       |     |     n = dividende , u = diviseur , v = quotient
|       |     |     retour
|       |     sinon
|       |     |     [u,v] = diviseur(a',b')
|       |     |     si u ≠ 0 et v ≠ 0 alors retour
|       |     fin
|       fin
|
fin
```

Déduisons la fonction "**factorisation**" qui permet de décomposer un nombre entier quelconque en facteurs premiers.

```
fonction  factorisation(n)
|    a = [] , b = []
|    [u,v] = diviseur(a,b)
|    si u = v = 0 alors
|    |     facteur = n
|    sinon
|    |     factorisation(u)
|    |     factorisation(v)
|    fin
fin
```

L'algorithme précédent n'est pas limité par le nombre de chiffres de n.
Tel quel, il reste inapplicable aux grands nombres entiers en raison du temps de calcul.

Mathématique & Informatique

Programme Scilab de la fonction **diviseur**

```
function [u,v] = diviseur(a,b)
  u = 0 ; v = 0
  if comparaison(n,1) <= 0 then, return, end
  L = list()
  for ai = 0:9
    ap = [ai,a]
    for bi = 0:9
      bp = [bi,b]
      if (a == []) & (bi < ai) then, continue, end
      if comparaison(ap,r) > 0 & comparaison(bp,r) > 0 then, break, end
      cp = multiplication(ap,bp)
      i1 = Max - length(ap) + 1
      ni = n(i1)
      i2 = length(cp) - length(ap) + 1
      if i2 > 0 then, cpi = cp(i2), else, cpi = 0, end
      if cpi == ni then, L($+1) = [ai,bi], end
    end
  end
  for i = 1:length(L)
    ap = [L(i)(1),a] ; bp = [L(i)(2),b]
    if comparaison(ap,1) > 0 & comparaison(bp,1) > 0 then
      [qap,rap] = division(n,ap)
      [qbp,rbp] = division(n,bp)
      if nul(rap) then
        u = ap ; v = qap
        return
      end
      if nul(rbp) then
        u = bp ; v = qbp
        return
      end
    end
    if comparaison(ap,r) > 0 & comparaison([1,bp],r) > 0 then, continue, end
    if comparaison([1,ap],r) > 0 & comparaison(bp,r) > 0 then, continue, end
    [u,v] = diviseur(ap,bp)
    if ~nul(u) & ~nul(v) then, return, end
  end
endfunction
```

Remarques

1) Si n n'est pas premier, il existe un diviseur de n plus petit que $\sqrt{n}$
   Preuve : $n = u\,v$ et $u > \sqrt{n} \Rightarrow u\,v > v\,\sqrt{n} \Rightarrow n > v\,\sqrt{n} \Rightarrow v < \sqrt{n}$
2) L'algorithme n'examine pas le couple d'entiers (a,b) si la valeur (b,a) a déjà été testée.

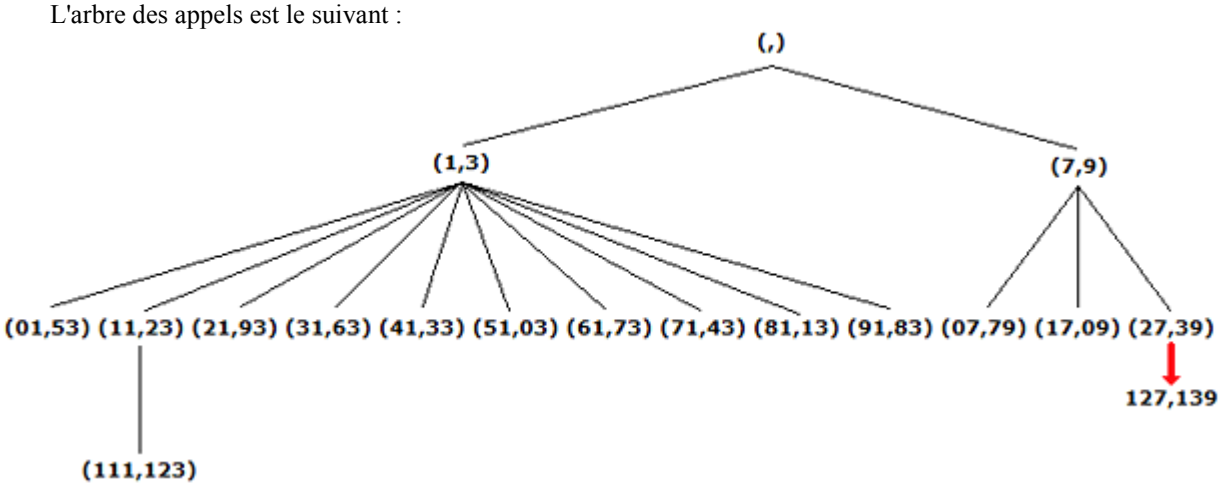
Mathématique & Informatique

Programme Scilab de la fonction **factorisation**

```
function factorisation(n)
  Max = length(n)
  r = intsqrt(n)
  a = [] ; b = []
  [u,v] = diviseur(a,b)
  if nul(u) & nul(v) then
    printf('\n Facteur : '), ecrire(n)
  else
    factorisation(u)
    factorisation(v)
  end
endfunction
```

Exemples

- Factorisation du nombre n = 17653
  On écrit : n = [1,7,6,5,3] ; factorisation(n)
  Les facteurs premiers de n s'affichent à l'écran : 127 et 139
  L'arbre des appels est le suivant :



- Factorisation du nombre n = 3195480257
  On écrit : n = [3,1,9,5,4,8,0,2,5,7] ; factorisation(n)
  Les facteurs premiers de n s'affichent à l'écran : 11003 et 290419

- Factorisation du nombre n = 23102009
  On écrit : n = [2,3,1,0,2,0,0,9] ; factorisation(n)
  Les facteurs premiers de n s'affichent à l'écran : 7, 29, 317, 359

- Factorisation du nombre n = 1299709
  On écrit : n = [1,2,9,9,7,0,9] ; factorisation(n)
  Les facteurs premiers de n s'affichent à l'écran : 1299709

Remarques

1) Si n possède k chiffres, alors le nombre de niveaux de l'arbre des appels est au plus k.
2) Lorsque l'instruction *factorisation(n)* retourne un seul facteur égal à n, le nombre n est premier.

Mathématique & Informatique

COMPLÉMENTS.

⇨ Chargement des modules

Afin de pouvoir utiliser les différentes fonctions incluses dans un module, il est impératif de le charger dans l'environnement Scilab. La commande à exécuter est :

exec("Nom_du_module.sce",-1)

Le paramètre -1 signifie que le texte du module ne s'affiche pas dans la fenêtre Console de Scilab lors du chargement. On ajoute souvent les 2 commandes annexes *funcprot(0)* et *lines(0)* pour, d'une part, pouvoir redéfinir une fonction sans qu'un message d'avertissement n'apparaisse à l'écran et, d'autre part, visualiser les résultats d'exécution sans arrêter le défilement des lignes (pas d'interruption avec message de continuation).

Note

Un module peut appeler un autre module lorsque les fonctions de l'un sont nécessaires à l'autre.

Scilab-Equations.sce
∨
Scilab-Matrices.sce
∨
Scilab-Polynomes.sce
∨
Scilab-Arithmetique.sce

Le module "**Scilab-Arithmetique.sce**" est le module de base ; il est utilisé par tous les autres modules.

⇨ Fenêtres graphiques

Si l'on exécute plusieurs instructions plot2d (ou plot3d) à la suite, les résultats s'affichent dans la même fenêtre graphique et les dessins se superposent. Pour obtenir des dessins distincts, il est nécessaire d'ouvrir des fenêtres graphiques différentes. Instructions de gestion des fenêtres graphiques :

scf() : ouvre une fenêtre graphique qui devient la fenêtre graphique courante
scf(n) : ouvre une fenêtre graphique et lui attribue le numéro *n*

clf() : efface le contenu de la fenêtre graphique courante
clf(n) : efface le contenu de la fenêtre graphique numéro *n*

xdel() : ferme la fenêtre graphique courante
xdel(n) : ferme la fenêtre graphique numéro *n*

NB
Par défaut, la première figure créée par Scilab porte le numéro 0 (Figure 0)
La fenêtre graphique courante est la dernière fenêtre ouverte. Pour que se soit la fenêtre graphique numéro *n* qui devienne la fenêtre graphique courante, on exécute l'instruction *scf(n)*.

Exemple
plot2d() ; plot3d() ⟶ dessins dans la même fenêtre numérotée 0
scf(1) ; plot2d() ; scf(2) ; plot3d() ⟶ dessins dans 2 fenêtres différentes numérotées 1 et 2

Mathématique & Informatique

⇨ Temps d'exécution

On souhaite souvent évaluer le temps d'exécution d'un programme, d'une fonction, d'un bloc d'instructions, ...

1) Horloge

La fonction *Horloge* affiche l'heure système :

```
function Horloge
  d = getdate() ; h = d(7) ; m = d(8) ; s = d(9) ;
  printf('\n ----- Horloge = %i:%i:%i',h,m,s)
endfunction
```

Le temps d'exécution est égal au temps écoulé entre 2 appels à Horloge en **heures:minutes:secondes**

2) Chronomètre

La fonction *Chronometre* affiche le temps CPU :

```
function Chronometre
  printf('\n ----- Temps CPU = %.3f secondes',timer())
endfunction
```

Premier appel à Chronometre : initialisation à zéro.
Deuxième, troisième, ... , appels à Chronomètre : temps écoulé entre 2 appels successifs.
Les temps d'exécution sont exprimés en **secondes**.

⇨ Calendrier perpétuel

La fonction *Calendrier* affiche la page de calendrier du mois *m* de l'année *a* :

```
function Calendrier(m,a)
  c = calendar(a,m)
  M = ['Janvier','Février','Mars','Avril','Mai','Juin','Juillet','Août','Septembre','Octobre','Novembre','Décembre']
  printf('\n -- ' + M(m) + ' ' + string(a) + ' --\n' + ' Lun Mar Mer Jeu Ven Sam Dim')
  for i = 1:6
    texte = ''
    for j = 1:7
      n = c(3)(i,j)
      if n == 0 then, x = '     ', elseif n >= 10 then, x = '  ' + string(n), else,  x = '   ' + string(n), end
      texte = texte + x
    end
    printf('\n' + texte)
  end
endfunction
```

Par exemple, la commande *Calendrier(2,1856)* donne :

```
 -- Février 1856 --
 Lun Mar Mer Jeu Ven Sam Dim
                   1   2   3
   4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29
```

Mathématique & Informatique

Calendrier grégorien perpétuel

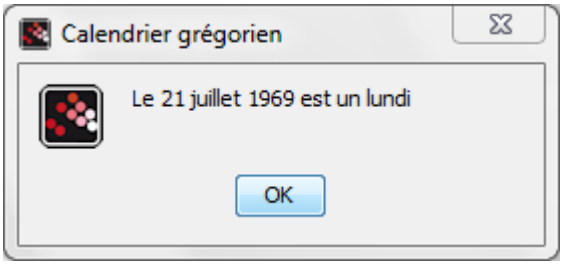
Pour rattraper le retard du calendrier julien – instauré par Jules César – sur le Soleil, le calendrier grégorien – instauré par le pape Grégoire XIII – passe du jeudi 4 octobre 1582 au vendredi 15 octobre 1582.
On recherche quel jour de la semaine (dimanche, lundi, mardi, ... , samedi) tombe une date donnée du calendrier grégorien.
Utilisons une formule due à Mike **Keith** (1990), similaire à la congruence de Christian **Zeller** (1883), et publiée dans *Journal of Recrational Mathematics* : ⇨ **http://www.cadaeic.net/calendar.htm**
Soit *a* l'année (à partir de 1582), *m* le mois (1 à 12), et *j* le jour (1 à 31) :

$$D = (a_1 + E(a_2/4) - E(a_2/100) + E(a_2/400) + E(23m/9) + j) \bmod 7$$
avec : $(a_1 = a + 2, a_2 = a$ si $m \geq 3)$ ou $(a_1 = a + 4, a_2 = a - 1$ si $m < 3)$

La valeur D = 0 correspond à un dimanche, D = 1 à un lundi, D = 2 à un mardi, ... , D = 6 à un samedi.
Programmons une fonction Scilab *calendrier* :

```
function calendrier
  // Lecture d'une date
  A = string([1582:3000])
  M = ['janvier','février','mars','avril','mai','juin','juillet','août','septembre','octobre','novembre','décembre']
  J = string([1:31])
  d = getdate()
  l1 = list('   Année',d(1)-1581,A)
  l2 = list('   Mois',d(2),M)
  l3 = list('   Jour',d(6),J)
  rep = x_choices('Date',list(l1,l2,l3))
  if rep == [] then, return, end
  a = rep(1) + 1581
  m = rep(2)
  j = rep(3)
  // Contrôle des données
  bissextile = (modulo(a,4) == 0 & modulo(a,100) <> 0) | (modulo(a,400) == 0)
  erreur1 = (m == 2) & (j > 28) & ~bissextile
  erreur2 = (m == 2) & (j > 29) & bissextile
  erreur3 = ((m == 4) | (m == 6) | (m == 9) | (m == 11)) & (j > 30)
  erreur4 = (a == 1582) & ((m < 10) | (m == 10) & (j < 15))
  if erreur1 | erreur2 | erreur3 | erreur4 then
    messagebox('Le ' + string(j) + ' ' + M(m) + ' ' + string(a) + ' n''existe pas','Calendrier grégorien','error')
    return
  end
  // Formule de Mike Keith
  if m >= 3 then, a1 = a + 2, a2 = a, else, a1 = a + 4, a2 = a - 1, end
  D = modulo(a1 + floor(a2/4) - floor(a2/100) + floor(a2/400) + floor(23*m/9) + j,7)
  // Affichage du résultat
  S = ['dimanche','lundi','mardi','mercredi','jeudi','vendredi','samedi']
  messagebox('Le ' + string(j) + ' ' + M(m) + ' ' + string(a) + ' est un ' + S(D+1),'Calendrier grégorien')
endfunction
```

Par exemple, recherchons quel jour de la semaine correspond au 21 juillet 1969 (premier pas de l'homme sur la Lune) :

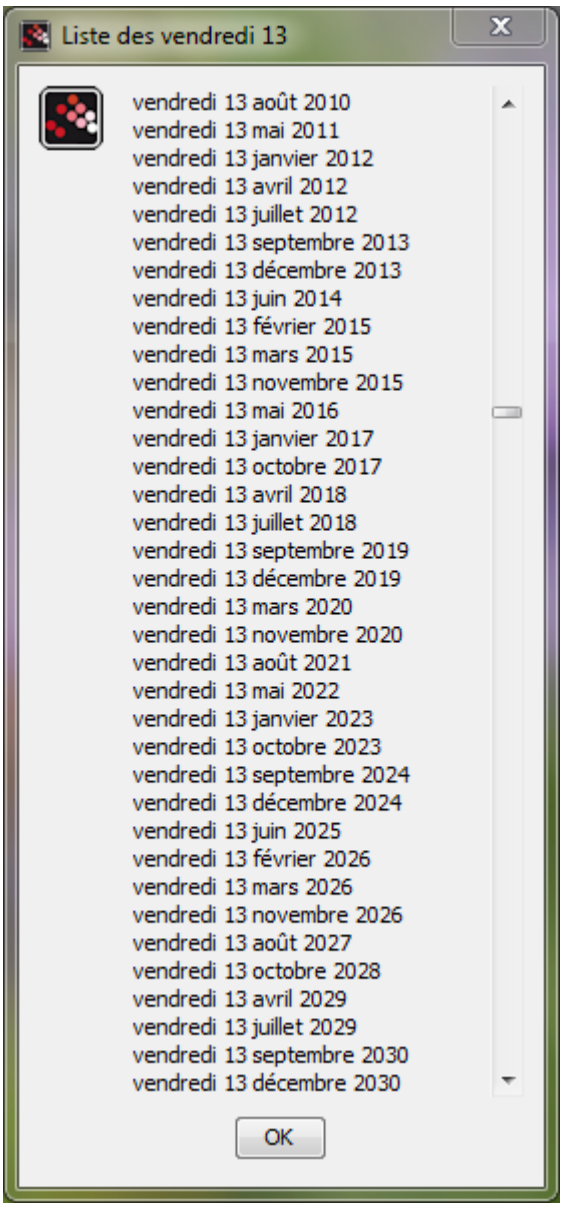


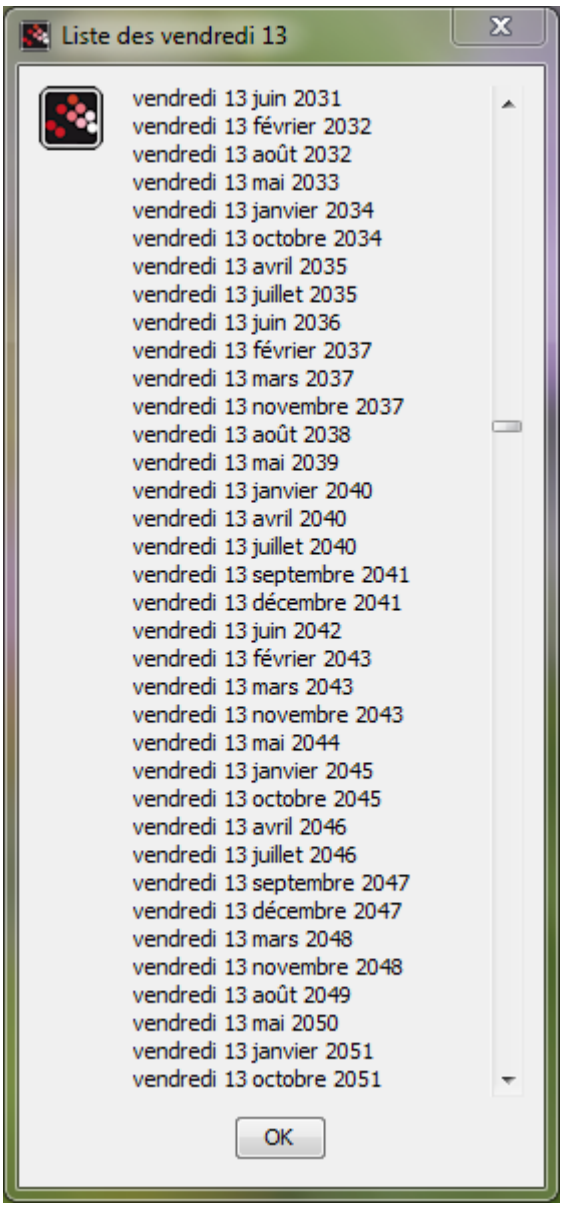
Mathématique & Informatique

Autre application de la formule de **Keith** : la liste des dates tombant un jour particulier de la semaine (jour ∈ {dimanche, lundi, mardi, mercredi, jeudi, vendredi, samedi}) et le j^e jour d'un mois ($j \in \{1, 2, 3, \dots, 29, 30, 31\}$).

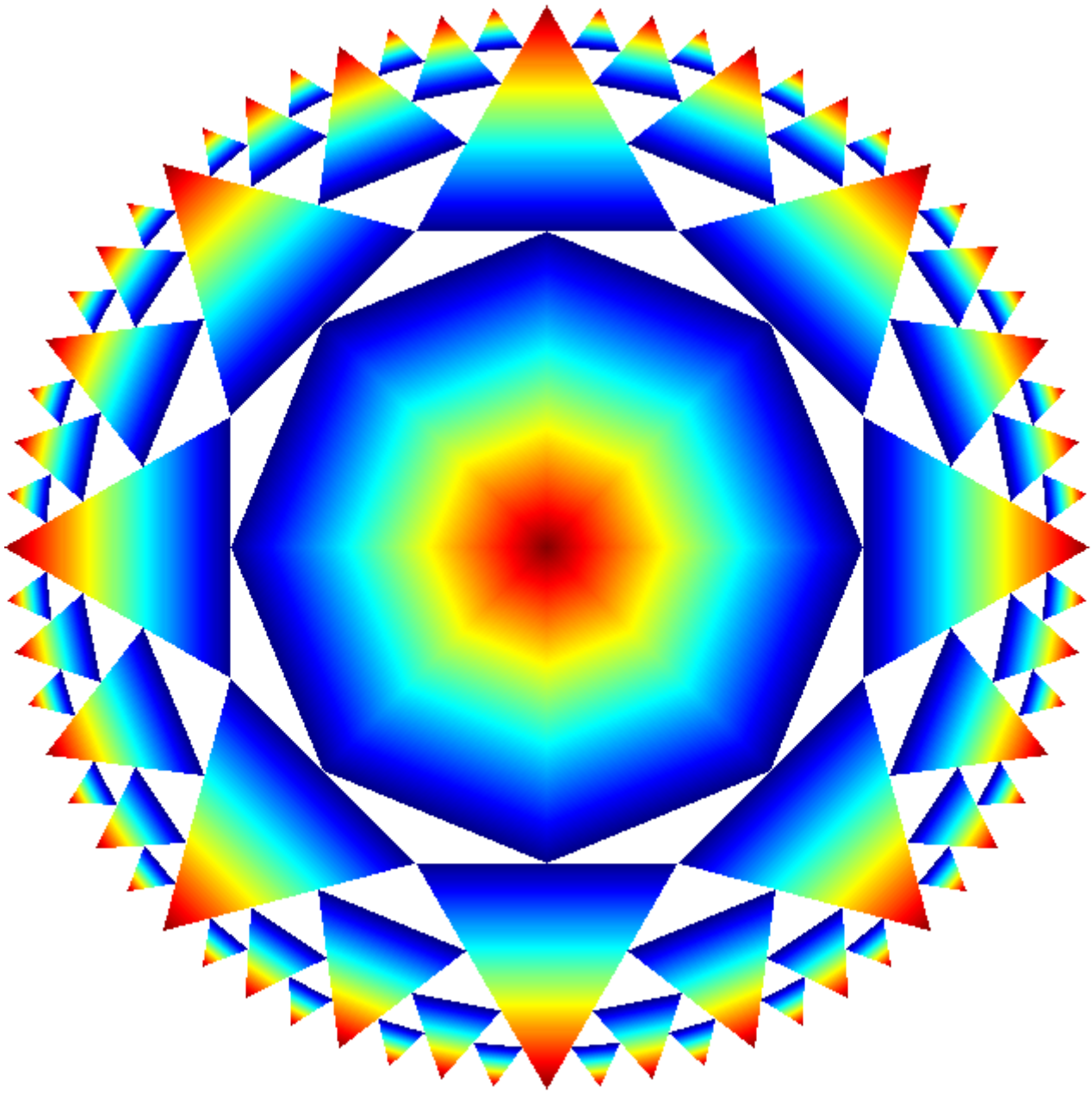
```
function liste_date(jour,j)
  liste = []
  M = ['janvier','février','mars','avril','mai','juin','juillet','août','septembre','octobre','novembre','décembre']
  J = ['dimanche','lundi','mardi','mercredi','jeudi','vendredi','samedi']
  d = find(J == jour) - 1
  if d == -1 | j < 1 | j > 31 then, return, end
  for a = 1583:3000
    for m = 1:12
      if m >= 3 then, a1 = a + 2, a2 = a, else, a1 = a + 4, a2 = a - 1, end
      D = modulo(a1 + floor(a2/4) - floor(a2/100) + floor(a2/400) + floor(23*m/9) + j,7)
      if D == d then, liste = [liste,jour + ' ' + string(j) + ' ' + M(m) + ' ' + string(a)], end
    end
  end
  messagebox(liste,'Liste des ' + jour + ' ' + string(j))
endfunction
```

Exemple : L'instruction *liste_date('vendredi',13)* donne la liste des Vendredi 13 (jusqu'en l'an 3000) :





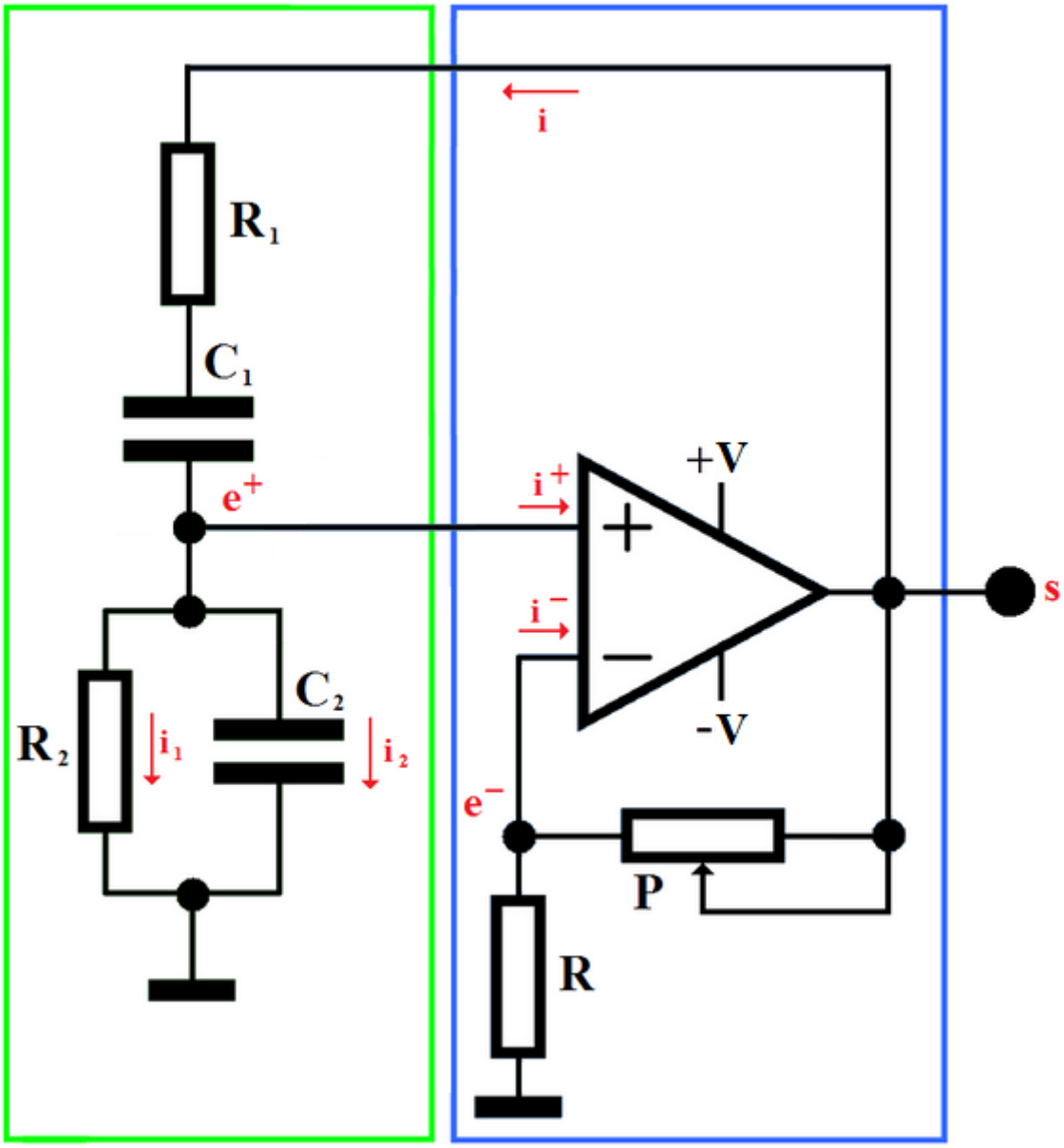
Mathématique & Informatique



Mathématique & Informatique

ÉLECTRONIQUE

Oscillateur à pont de Wien



Problème

Formuler l'équation différentielle donnant la tension de sortie s de l'AOP en fonction du temps t.
Ecrire un programme de simulation qui permet de visualiser la courbe s(t) lorsque le potentiomètre P varie de 0 à sa valeur maximale.

On prend : $R = 15\ k\Omega$, $R_1 = 10\ k\Omega$, $R_2 = 10\ k\Omega$, $P = 47\ k\Omega$, $C_1 = 10\ nF$, $C_2 = 10\ nF$, $V = 10\ V$

Solution

Le potentiomètre P assure la contre-réaction entre l'entrée inverseuse et la sortie de l'AOP. Dans ce type de montage, rappelons les 2 règles fondamentales de fonctionnement d'un amplificateur opérationnel idéal :

1) Le gain en boucle ouverte est très grand, de sorte que les tensions d'entrée sont maintenues pratiquement égales

$e^- = e^+ = e$

2) L'impédance d'entrée est très grande, de sorte qu'il n'entre (ou ne sort) pratiquement aucun courant par les entrées

$i^- = i^+ = 0$

Sur l'entrée – de l'AOP, écrivons l'équation du pont diviseur de tension :

$$e = \frac{R}{R+P}s$$

Sur l'entrée + de l'AOP, écrivons les 3 équations de branche et l'équation de nœud :

$$s - e = R_1 i + \frac{1}{C_1}\int i\,dt$$

$$e = R_2 i_1$$

$$e = \frac{1}{C_2}\int i_2\,dt$$

$$i = i_1 + i_2$$

On déduit :

$$i_1 = \frac{e}{R_2} \text{ et } i_2 = C_2\frac{de}{dt}$$

Par suite :

$$s - \frac{R}{R+P}s = R_1\left[\frac{1}{R_2}\left(\frac{R}{R+P}s\right) + C_2\frac{d}{dt}\left(\frac{R}{R+P}s\right)\right] + \frac{1}{C_1}\left[\frac{1}{R_2}\int\left(\frac{R}{R+P}s\right)dt + C_2\left(\frac{R}{R+P}s\right)\right]$$

Après simplification et dérivation, on obtient l'équation différentielle linéaire d'ordre 2 vérifiée par la tension s :

$$\frac{d^2s}{dt^2} + \left[\frac{R(R_1C_1 + R_2C_2) - R_2C_1P}{RR_1R_2C_1C_2}\right]\frac{ds}{dt} + \left(\frac{1}{R_1R_2C_1C_2}\right)s = 0$$

Dans le cas particulier où $R_1 = R_2 = R_0$ et $C_1 = C_2 = C_0$, l'équation devient :

$$\frac{d^2s}{dt^2} + \left(\frac{2R - P}{RR_0C_0}\right)\frac{ds}{dt} + \left(\frac{1}{R_0C_0}\right)^2 s = 0$$

Programme

Effectuons une simulation numérique avec la fonction Scilab *ode* :

```
function sp = f(t,s)
  sp(1) = s(2)
  sp(2) = (R2*C1*P - R*(R1*C1+R2*C2))/(R*R1*R2*C1*C2)*s(2) - 1/(R1*R2*C1*C2)*s(1)
endfunction

R = 15D3 ; R1 = 10D3 ;  R2 = 10D3 ; Pmax = 47D3 ; C1 = 10D-9 ; C2 = 10D-9 ; V = 10 ;
t0 = 0 ; t1 = 5D-3 ; n = 1000 ; t = linspace(t0, t1, n) ;

for P = 0:Pmax/100:Pmax
  s = ode([+V/2;0],0,t,f) ; s = s(1,:) ;
  for i = 1:n
    if s(i) < 0 then, s(i) = max(s(i),-V) ; end
    if s(i) > 0 then, s(i) = min(s(i),+V) ; end
  end
  clf()
  plot2d(t, s, 5, frameflag=5, rect=[t0,-3/2*V,t1,+3/2*V], axesflag=1)
  xpause(10e3)
end

R = 15D3 ; R1 = 10D3 ;  R2 = 10D3 ; Pmax = 47D3 ; C1 = 10D-9 ; C2 = 10D-9 ; V = 10 ;
t0 = 0 ; t1 = 25D-3 ; n = 1000 ; t = linspace(t0, t1, n) ;
while %t
  g = input('\n g = ?') ;
  if g == [] then, break, end
  P = (g-1)*R ;
  s = ode([+V/2;0],0,t,f) ; s = s(1,:) ;
  clf()
  plot2d([t0,t1], [0,0], 2)
  plot2d(t, s, 5, frameflag=5, rect=[t0,-V,t1,+V], axesflag=1)
end
```

Note

Cette simulation suppose que l'on fournisse une excitation de départ à l'oscillateur en chargeant l'un de ses condensateurs.
On choisit : s(0) = +V/2 (valeur de la tension de sortie à t = 0).
D'autre part, pour simplifier, on considère que la tension de saturation de l'amplificateur est égale à sa tension d'alimentation : $+V_{SAT} = +V$ et $-V_{SAT} = -V$

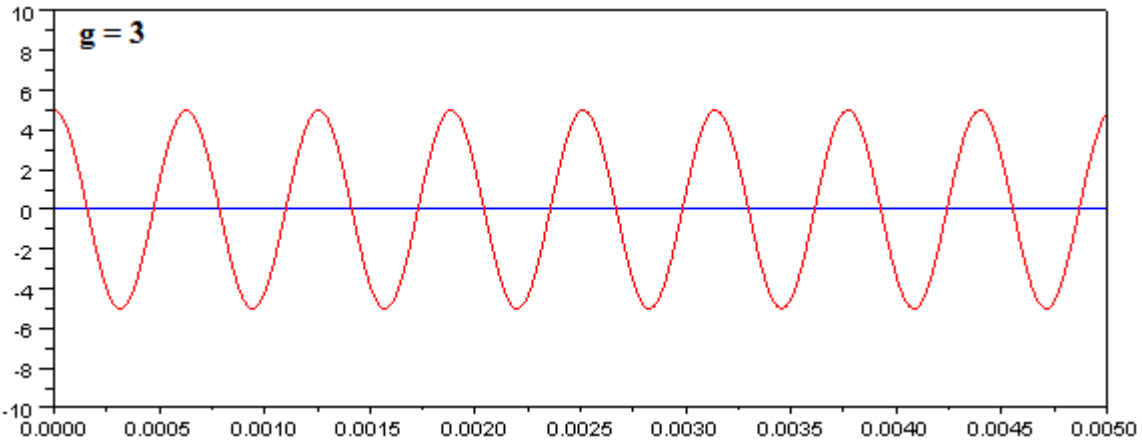
Cas particulier $R_1 = R_2 = R_0$, $C_1 = C_2 = C_0$

- Si P = 2R, l'équation différentielle se réduit à :

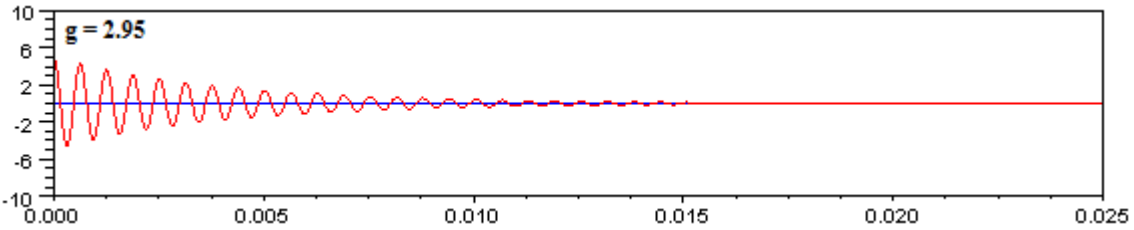
$$\frac{d^2 s}{dt^2} + \omega^2 s = 0 \qquad \text{avec} \qquad \omega = \frac{1}{R_0 C_0}$$

La solution est sinusoïdale de fréquence $f = 1/(2\pi R_0 C_0)$: $s = a \sin(\omega t + \varphi)$
Le gain en tension de l'AOP est :

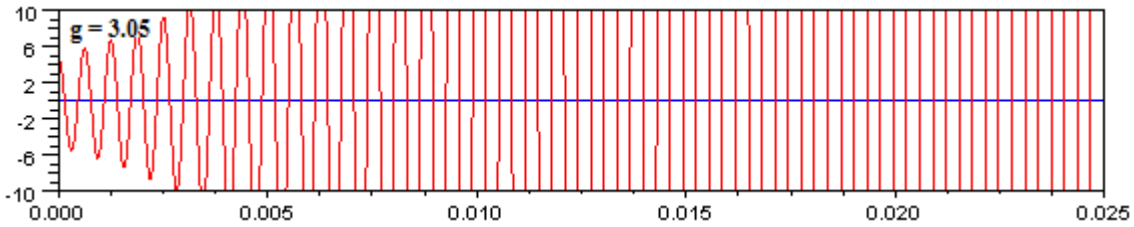
$$g = \frac{s}{e} = \frac{R+P}{R} = 1 + \frac{P}{R} = 3$$



- Si P < 2R (donc g < 3), on observe une oscillation initiale dont l'amplitude décroît exponentiellement ; après ce régime transitoire le signal de sortie est nul.



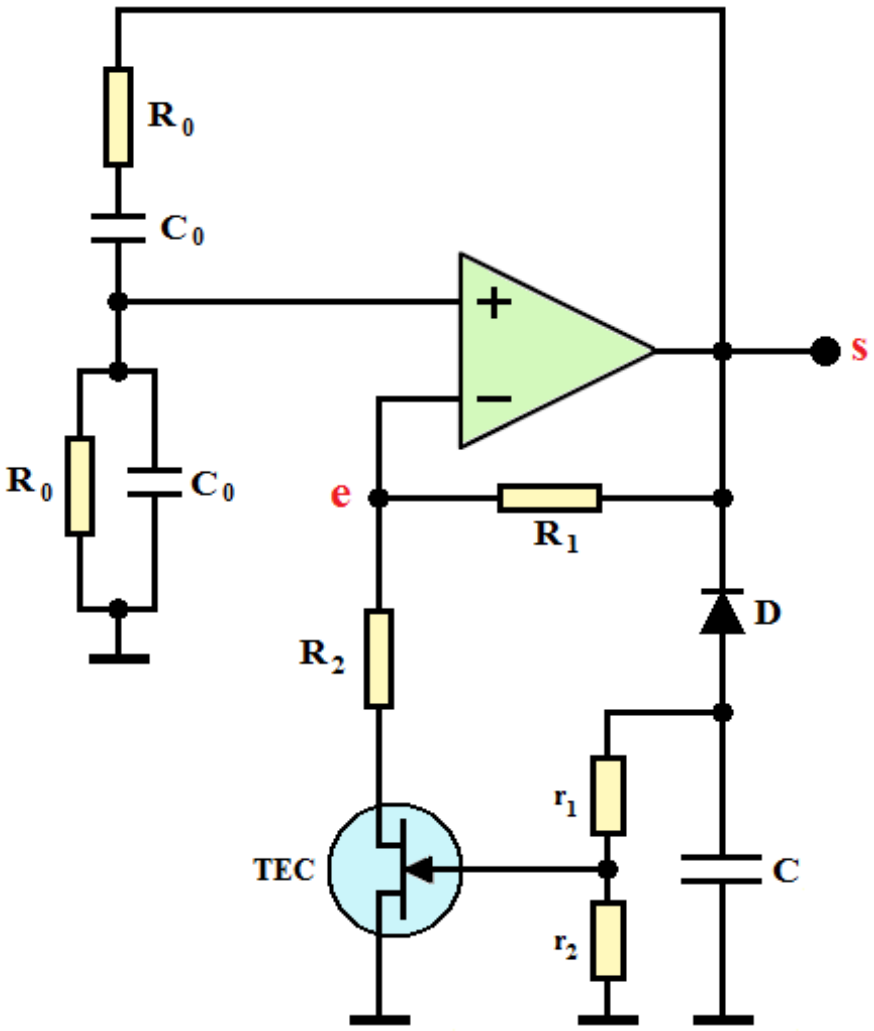
- Si P > 2R (donc g > 3), l'amplitude d'oscillation croît jusqu'à la saturation de l'amplificateur ; le signal de sortie devient carré.



Conclusion

Ce montage classique, appelé oscillateur à pont de **Wien**, est utilisé pour produire des sinusoïdes de faible distorsion. Il est limité aux basses fréquences (de 10 hertz à quelques centaines de kilohertz).
Théoriquement, un ajustement précis du potentiomètre permet de régler le gain de l'AOP à 3 exactement (s = 3e) pour obtenir un signal de sortie sinusoïdal. En pratique, c'est moins facile qu'il y paraît puisqu'on passe très rapidement de l'abscence de signal à un signal carré de saturation comme le confirme la simulation précédente. Afin de stabiliser l'oscillateur, on monte parfois 2 diodes tête-bêche et 2 résistances en parallèle avec le potentiomètre.
⇨ Jean **Auvray** : http://www.thehackademy.net/madchat/coding/electro/CSE016_Les_oscillateurs.pdf
Des montages plus perfectionnés, avec contrôle automatique de gain (CAG), fournissent des sinusoïdes dont la distorsion n'est que de quelques parties par million.

Exemple d'oscillateur à pont de Wien stabilisé en amplitude par un transistor à effet de champ



La tension de sortie s de l'oscillateur est redressée par la diode D et filtrée par le condensateur C. Elle pilote la grille d'un transistor TEC qui se comporte comme un élément résistif variable. De cette manière, on commande le gain de l'oscillateur : si l'amplitude de sortie augmente, la charge négative du condensateur C s'élève et la tension négative grille-source du TEC également via le pont diviseur constitué par r_1 et r_2 ; en conséquence le courant de drain chute proportionnellement à la transconductance du TEC et tout se passe comme si la résistance R_2 était plus grande, entrainant une diminution du gain g = s/e de l'amplificateur. Inversement, si l'amplitude de sortie diminue, le gain de l'amplificateur augmente. Cette boucle de contrôle automatique de gain permet de conserver une tension de sortie sinusoïdale.

Mathématique & Informatique

Transformées de Laplace

$$f(t) \supset F(p) = \int_0^{+\infty} f(t) e^{-pt}\, dt$$

La variable p est un nombre complexe : $p \in \mathbb{C}$.
En électricité il n'y a pas de temps négatif, donc on prend f(t) = 0 pour t < 0 (fonction causale).
Le calcul opérationnel ne fait aucune hypothèse sur la nature des tensions et des courants, par contre le calcul effectué avec des nombres complexes en remplaçant p par $j\omega$, où j désigne l'unité imaginaire, suppose que les tensions et les courants soient sinusoïdaux, c'est-à-dire de la forme : $a \sin(\omega t + \varphi)$
On adopte généralement les minuscules pour les fonctions temporelles et les majuscules pour les transformées.
Le symbole $\supset$ signifie "a pour transformée de **Laplace**", le symbole $\subset$ signifie "est la transformée de **Laplace** de".

Table des fonctions usuelles

$f(t)$	$F(p)$
$\delta(t)$	1
$u(t)$	$\frac{1}{p}$
t^n	$\frac{n!}{p^{n+1}}$
e^{at}	$\frac{1}{p-a}$
$\sin(\omega t)$	$\frac{\omega}{p^2+\omega^2}$
$\cos(\omega t)$	$\frac{p}{p^2+\omega^2}$
$\mathrm{sh}(\omega t)$	$\frac{\omega}{p^2-\omega^2}$
$\mathrm{ch}(\omega t)$	$\frac{p}{p^2-\omega^2}$
$\frac{1}{\sqrt{t}}$	$\sqrt{\frac{\pi}{p}}$
$\mathrm{Log}(t)$	$-\frac{\gamma + Log\, p}{p}$

Propriétés

$$f_1(t) + \lambda f_2(t) \supset F_1(p) + \lambda F_2(p)$$

$$f(t-a) \supset e^{-ap} F(p)$$

$$f(at) \supset \frac{1}{a} F\left(\frac{p}{a}\right)$$

$$e^{-at} f(t) \supset F(p+a)$$

$$f'(t) \supset F(p) - f(0)$$

$$\int_0^t f(x)\, dx \supset \frac{F(p)}{p}$$

$$\int_0^t f(x) g(t-x)\, dx \supset F(p) G(p)$$

$$\lim_{p \to \infty} pF(p) = f(0)$$

$$\lim_{p \to 0} pF(p) = f(+\infty)$$

Remarques

- Pour les fonctions usuelles, le calcul des transformées de **Laplace** se fait sans difficulté, sauf pour f(t) = $1/\sqrt{t}$ et f(t) = Log t où interviennent respectivement le nombre pi (π) et la constante d'**Euler** (γ).
  Dans ces 2 cas, on utilise la fonction eulérienne Gamma qui est liée à la transformée de **Laplace** :

$$\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t}\, dt$$

  Pour f(t) = $1/\sqrt{t}$, on effectue le changement de variable u = pt dans F(p), et on est ramené à l'intégrale :

$$\int_0^{+\infty} u^{-\frac{1}{2}} e^{-u}\, du = \Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$$

  Pour f(t) = Log t , on effectue le changement de variable u = pt dans F(p), puis en dérivant $\Gamma(x)$ sous le signe somme et en prenant x = 1, on est ramené à l'intégrale :

$$\int_0^{+\infty} e^{-u} \operatorname{Log} u\, du = \Gamma'(1)$$

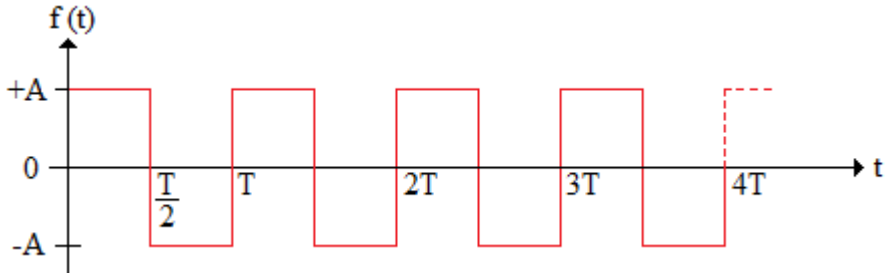
  La formule de **Weierstrass** (développement en produit infini de la fonction Gamma) s'écrit :

$$\frac{1}{\Gamma(x)} = x e^{\gamma x} \prod_{n=1}^{+\infty} \left(1 + \frac{x}{n}\right) e^{-\frac{x}{n}}$$

  La dérivée logarithmique de cette égalité prise en $x = 1$ donne :

$$-\frac{\Gamma'(1)}{\Gamma(1)} = \gamma$$

- Il existe des tables très complètes de transformées de **Laplace**. On y trouve la plupart des fonctions utilisées couramment en électronique. Par exemple, la fonction en créneaux :



  Soit x(t) la restriction de f(t) à l'intervalle [0,T[

  On a : $f(t) = x(t) + x(t-T) + x(t-2T) + x(t-3T) + \cdots$

  D'où : $F(p) = X(p) + e^{-pT} X(p) + e^{-2pT} X(p) + e^{-3pT} X(p) + \cdots = \dfrac{X(p)}{1 - e^{-pT}}$

  Comme : $X(p) = \displaystyle\int_0^{\frac{T}{2}} (+A) e^{-pt}\, dt + \int_{\frac{T}{2}}^{T} (-A) e^{-pt}\, dt = \frac{A}{p}\left(1 - e^{-p\frac{T}{2}}\right)^2$

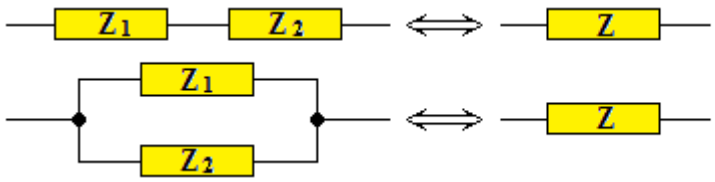
  On déduit la transformée de Laplace de f(t) : $F(p) = \dfrac{A}{p} th\left(\dfrac{Tp}{4}\right)$

Calcul opérationnel

Pour faciliter les calculs dans les circuits électriques, on introduit les transformées de **Laplace** des tensions u(t) et des courants i(t), ainsi que les impédances opérationnelles des différents composants R, L, C :

$$u(t) \supset U(p)$$
$$i(t) \supset I(p)$$
$$Z_R = R$$
$$Z_L = Lp$$
$$Z_C = \frac{1}{Cp}$$

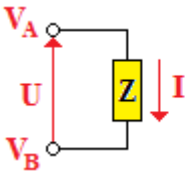
Association série et parallèle d'impédances :



$$Z = Z_1 + Z_2$$

$$\frac{1}{Z} = \frac{1}{Z_1} + \frac{1}{Z_2}$$

La loi d'**Ohm** généralisée :



$$U = V_A - V_B = Z\,I$$

Les lois de **Kirchhoff** sur les mailles ΣU = 0 et sur les nœuds ΣI = 0 s'appliquent également.
En fait, une équation différentielle de la variable t peut être ramenée à une équation algébrique de la variable p. Multiplier par p est équivalent à dériver et diviser par p est équivalent à intégrer. Si l'on arrive à résoudre l'équation en p, alors on revient à la solution en t en recherchant la transformée inverse dans une table.
Par exemple, considérons l'équation différentielle linéaire d'ordre n à coefficients constants :

$$a_n y^{(n)}(t) + \cdots + a_1 y'(t) + a_0 y(t) = f(t)$$

En prenant les transformées de **Laplace** des 2 membres, on obtient :

$$a_n \left[p^n Y(p) - \sum_{k=0}^{n-1} p^{n-k-1} y^{(k)}(0) \right] + \cdots + a_1 \left[pY(p) - y(0) \right] + a_0 Y(p) = F(p)$$

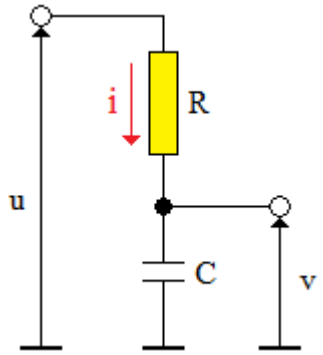
D'où :

$$Y(p) = \frac{F(p) + a_n \sum_{k=0}^{n-1} p^{n-k-1} y^{(k)}(0) + \cdots + a_1 y(0)}{a_n p^n + \cdots + a_1 p + a_0}$$

Si F(p) est une fraction rationnelle en p, on décompose Y(p) en éléments simples de première et de deuxième espèce, puis on utilise la table des transformées pour trouver l'expression de y(t).

Exemple 1

On effectue fréquemment la charge/décharge d'une capacité à tension constante :



A t = 0, la capacité C possède une tension initiale v_0 et elle est soumise à la tension constante u via la résistance R :

- $u > v_0 \Rightarrow$ la capacité C se charge à travers la résistance R
- $u < v_0 \Rightarrow$ la capacité C se décharge à travers la résistance R
- $u = v_0 \Rightarrow$ la capacité C reste au même potentiel

Entre 2 instants t_1 et t_2, la tension v aux bornes de C passe de v_1 à v_2. On a la relation :

$$\Delta t = t_2 - t_1 = RC\,Log\frac{u - v_1}{u - v_2}$$

Preuve

Equations temporelles :

$$u - v = Ri$$
$$i = C\frac{dv}{dt}$$

Transformées de **Laplace** :

$$U - V = RI$$
$$I = C(pV - v_0)$$

On déduit :

$$V = \frac{U + RCv_0}{1 + RCp}$$

Comme $u = Cte$, on a : $U = \frac{u}{p}$. D'où :

$$V = \frac{u + RCv_0 p}{p(1 + RCp)} = u\left(\frac{1}{p}\right) + (v_0 - u)\left(\frac{1}{p + \frac{1}{RC}}\right)$$

Revenons aux grandeurs temporelles en utilisant la table des transformées de **Laplace** :

$$v = u + (v_0 - u)e^{-\frac{t}{RC}}$$

Exprimons t :

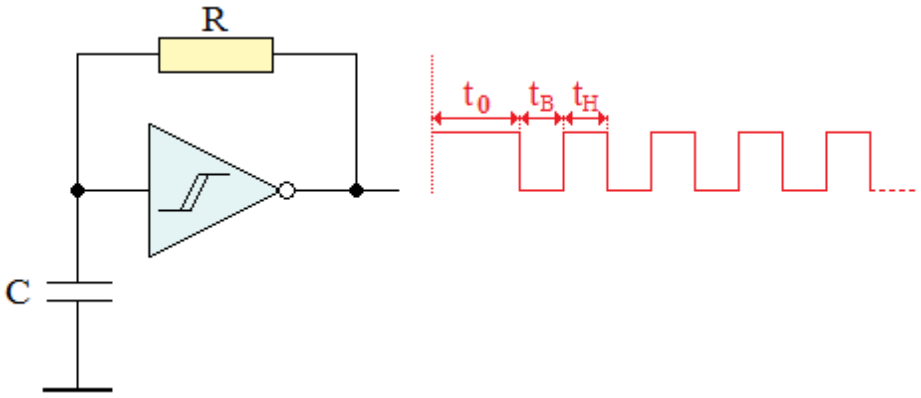
$$t = RC\,Log\frac{u - v_0}{u - v}$$

Posons $v = v_1$ à $t = t_1$ et $v = v_2$ à $t = t_2$. On obtient finalement :

$$\Delta t = t_2 - t_1 = RC\,Log\frac{u - v_0}{u - v_2} - RC\,Log\frac{u - v_0}{u - v_1} = RC\,Log\frac{u - v_1}{u - v_2}$$

Nota Bene : Cette relation, souvent employée, est vraie pour une charge ou une décharge capacitive.

Application



Le constructeur fournit les données suivantes pour la porte inverseuse à trigger de **Schmitt** :

- Tension d'alimentation : V = 5 volts
- Seuils de basculement haut et bas : $V_H = 2.9$ volts ; $V_B = 1.9$ volts

Calculons la période d'oscillation du multivibrateur astable ci-dessus en fonction de R et C.

1) Au départ la capacité est entièrement déchargée, l'entrée de la porte est à "0" (c'est à dire à 0 volt) et sa sortie est à "1" (c'est à dire à 5 volts).

La capacité se charge de 0 jusqu'à V_H en $t_0 = RC\ Log\frac{V - 0}{V - V_H} \approx 0.87\ RC$

2) La porte bascule à "0" : la capacité se décharge de V_H jusqu'à V_B en $t_B = RC\ Log\frac{0 - V_H}{0 - V_B} \approx 0.42\ RC$

3) La porte bascule à "1" : la capacité se charge de V_B jusqu'à V_H en $t_H = RC\ Log\frac{V - V_B}{V - V_H} \approx 0.39\ RC$

4) Le cycle recommence en 2)

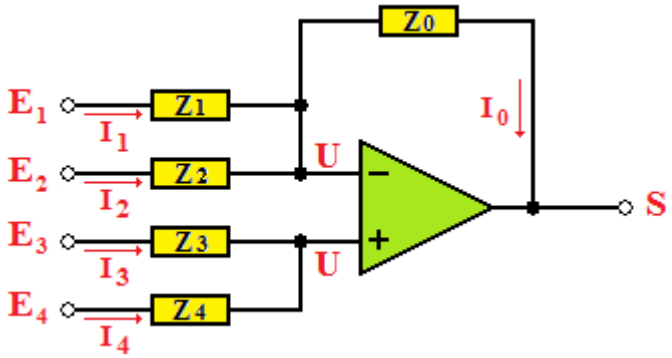
La période d'oscillation est donc : $T = t_B + t_H = RC\ Log\frac{V_H(V - V_B)}{V_B(V - V_H)} \approx 0.81\ RC$

Le rapport cyclique est : $\tau = \frac{t_H}{t_H + t_B} \approx \frac{1}{2}$

Mathématique & Informatique

Exemple 2

Dans le circuit électronique représenté ci-dessous, calculer la transformée de **Laplace** S(p) de la sortie en fonction des transformées de **Laplace** $E_1(p)$, $E_2(p)$, $E_3(p)$, $E_4(p)$ des entrées.



(1) $E_1 - U = Z_1 I_1$
(2) $E_2 - U = Z_2 I_2$
(3) $E_3 - U = Z_3 I_3$
(4) $E_4 - U = Z_4 I_4$
(5) $U - S = Z_0 I_0$
(6) $I_1 + I_2 = I_0$
(7) $I_3 + I_4 = 0$

L'expression de S s'obtient en résolvant le système d'équations linéaires (1) à (7) :

(5)&(6): $$S = U - Z_0 I_0 = U - Z_0 \left(I_1 + I_2 \right)$$

(1)&(2): $$S = U - Z_0 \left(\frac{E_1 - U}{Z_1} + \frac{E_2 - U}{Z_2} \right) = - \left(\frac{Z_0}{Z_1} \right) E_1 - \left(\frac{Z_0}{Z_2} \right) E_2 + \left(1 + \frac{Z_0}{Z_1} + \frac{Z_0}{Z_2} \right) U$$

(3)&(4)&(7): $$S = - \left(\frac{Z_0}{Z_1} \right) E_1 - \left(\frac{Z_0}{Z_2} \right) E_2 + \left(1 + \frac{Z_0}{Z_1} + \frac{Z_0}{Z_2} \right) \left(\frac{Z_4 E_3 + Z_3 E_4}{Z_3 + Z_4} \right)$$

$$S = - \left(\frac{Z_0}{Z_1} \right) E_1 - \left(\frac{Z_0}{Z_2} \right) E_2 + \left(1 + \frac{Z_0}{Z_1} + \frac{Z_0}{Z_2} \right) \left(\frac{Z_4}{Z_3 + Z_4} \right) E_3 + \left(1 + \frac{Z_0}{Z_1} + \frac{Z_0}{Z_2} \right) \left(\frac{Z_3}{Z_3 + Z_4} \right) E_4$$

S est une somme pondérée des tensions d'entrées E_1, E_2, E_3, E_4 . Ce schéma est appelé "amplificateur sommateur".
En attribuant à E_1, E_2, E_3, E_4, Z_0, Z_1, Z_2, Z_3, Z_4 des valeurs particulières, on retrouve les circuits fondamentaux :

E_1	E_2	E_3	E_4	Z_0	Z_1	Z_2	Z_3	Z_4	S
0	∅	E	∅	R_0	R_1	∞	0	∞	$\left(1 + \frac{R_0}{R_1}\right)E$ *amplificateur non inverseur*
E	∅	0	∅	R_0	R_1	∞	0	∞	$-\left(\frac{R_0}{R_1}\right)E$ *amplificateur inverseur*
E	∅	0	∅	$\frac{1}{Cp}$	R	∞	0	∞	$-\left(\frac{1}{RCp}\right)E$ *intégrateur*
E	∅	0	∅	R	$\frac{1}{Cp}$	∞	0	∞	$-(RCp)E$ *dérivateur*
E	∅	E	0	2R	2R	∞	$\frac{1}{Cp}$	R	$\left(\frac{RCp - 1}{RCp + 1}\right)E$ *déphaseur*

De nombreux circuits se ramènent au schéma de base précédent. Par exemple, pour l'oscillateur à pont de **Wien** :

- E_1 est reliée à la masse : $E_1 = 0$
- E_2 est non connectée : $E_2 = \varnothing$
- E_3 est reliée à la masse : $E_3 = 0$
- E_4 est reliée à la sortie S : $E_4 = S$
- Z_0 est un potentiomètre P : $Z_0 = P$
- Z_1 est une résistance R : $Z_1 = R$
- Z_2 est enlevée : $Z_2 = \infty$
- Z_3 est une résistance R_0 en parallèle avec une capacité C_0 : $Z_3 = R_0/(R_0C_0p + 1)$
- Z_4 est une résistance R_0 en série avec une capacité C_0 : $Z_4 = (R_0C_0p + 1)/(C_0p)$

La formule de l'amplificateur sommateur s'écrit dans ce cas :

$$S = \left(1 + \frac{Z_0}{Z_1}\right)\frac{Z_3}{Z_3 + Z_4}\ S$$

C'est-à-dire :

$$S = \left(1 + \frac{P}{R}\right)\frac{R_0C_0\,p}{R_0C_0\,p + \left(R_0C_0\,p + 1\right)^2}\ S$$

Cette relation se met sous la forme :

$$p^2S + \left(\frac{2R - P}{RR_0C_0}\right)pS + \left(\frac{1}{R_0C_0}\right)^2 S = 0$$

Equivalente à l'équation différentielle :

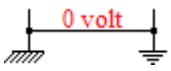
$$\frac{d^2s}{dt^2} + \left(\frac{2R - P}{RR_0C_0}\right)\frac{ds}{dt} + \left(\frac{1}{R_0C_0}\right)^2 s = 0$$

<u>La notion de masse électrique</u>

En pratique, rappelons que seule la différence de potentiel a une signification réelle (grandeur mesurable).
On exprime donc toutes les ddp (tensions) par rapport à un potentiel de référence appelé la "masse" et auquel on attribue conventionnellement la valeur de **0 volt**.
Sur une carte électronique, la masse est généralement un plan métallisé où sont raccordés de nombreux composants.

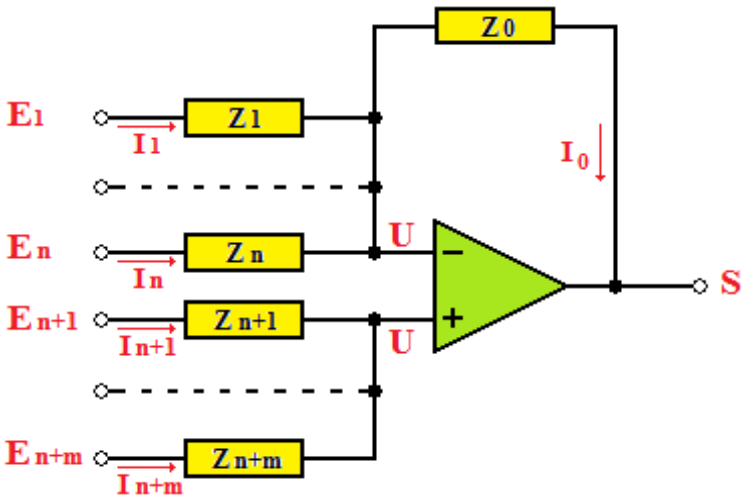
Sur les schémas, cette équipotentielle est représentée par l'un des symboles suivants : 

Ne pas confondre la "masse" avec la "terre" (le sol de notre planète) pour laquelle on utilise le symbole : 

Si on relie la masse à la terre, alors les 2 conducteurs sont au même potentiel de "0 V" : 

Souvent la masse n'est pas reliée à la terre (cas du circuit électrique d'une automobile isolée du sol par exemple).

Généralisation



$E_1, \ldots, E_n$ sont les transformées de **Laplace** des tensions appliquées à l'entrée – de l'AOP.
$E_{n+1}, \ldots, E_{n+m}$ sont les transformées de **Laplace** des tensions appliquées à l'entrée + de l'AOP.
S est la transformée de **Laplace** de la tension de sortie de l'AOP.
Z_0 est l'impédance opérationnelle placée dans la boucle de contre-réaction de l'AOP.
$Z_1, \ldots, Z_n$ sont les impédances opérationnelles reliées à l'entrée – de l'AOP.
$Z_{n+1}, \ldots, Z_{n+m}$ sont les impédances opérationnelles reliées à l'entrée + de l'AOP.

Le schéma conduit au système d'équations linéaires :

$E_1 - U = Z_1 I_1$
• • •
$E_n - U = Z_n I_n$
$E_{n+1} - U = Z_{n+1} I_{n+1}$
• • •
$E_{n+m} - U = Z_{n+m} I_{n+m}$
$U - S = Z_0 I_0$
$I_1 + \ldots + I_n = I_0$
$I_{n+1} + \ldots + I_{n+m} = 0$

En appliquant la même méthode de résolution que pour n = m = 2, on aboutit à la formule générale :

$$S = -\sum_{i=1}^{n} \left(\frac{Z_0}{Z_i}\right) E_i + \sum_{j=1}^{m} \left[\frac{\left(1 + \sum_{i=1}^{n} \frac{Z_0}{Z_i}\right) \prod_{k \neq j} Z_{n+k}}{\sum_{l=1}^{m} \prod_{k \neq l} Z_{n+k}} \right] E_{n+j}$$

Mathématique & Informatique

Programmation

Supposons que les transformées de **Laplace** E_k , $1 \le k \le n+m$, soient des fractions rationnelles de la variable p regroupées dans une liste E :

$$E = list(E_1, \dots , E_n, E_{n+1}, \dots , E_{n+m})$$

Notons Z la liste des fractions rationnelles de la variable p représentant les impédances opérationnelles Z_k , $0 \le k \le n+m$:

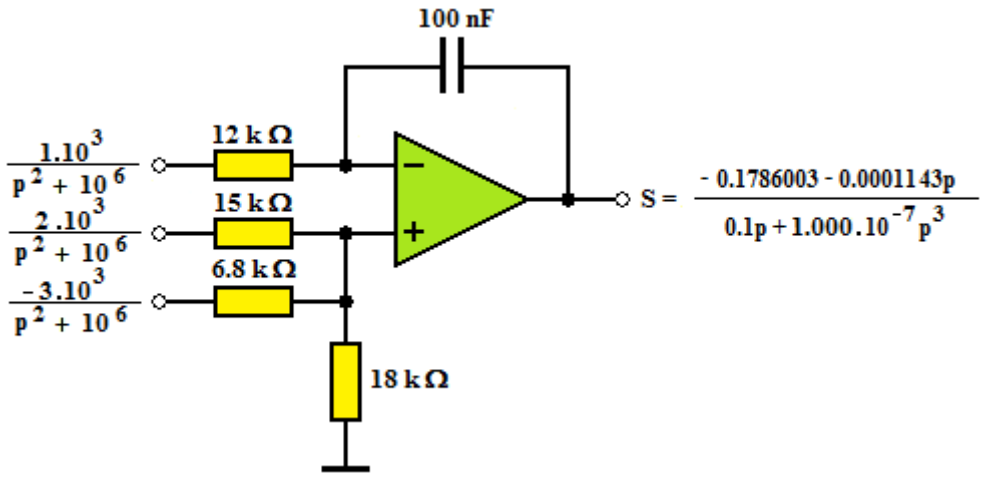
$$Z = list(Z_1, \dots , Z_n, Z_{n+1}, \dots , Z_{n+m}, Z_0)$$

La transformée de **Laplace** S de la tension sortie de l'AOP est une fraction rationnelle de la variable p.

Traduisons la formule obtenue par un programme Scilab qui calcule S en fonction de E et Z :

```
function S = AOP(E,Z,n,m)
  S1 = 0 ;  S2 = 0 ;  S3 = 0 ;  S4 = 0
  for i = 1:n
    S1 = S1 + 1/Z(i)
    S2 = S2 + E(i)/Z(i)
  end
  for j = 1:m
    PZ = 1
    for k = 1:m
      if k <> j then
        PZ = PZ*Z(n+k)
      end
    end
    S3 = S3 + PZ
    S4 = S4 + PZ*E(n+j)
  end
  S = - Z($)*S2 + ((1+Z($)*S1)/S3)*S4
endfunction
```

Exemple

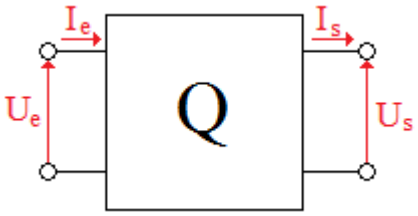



```
p = poly(0,'p') ;
S = AOP(list(1D3/(p^2+1D6),2D3/(p^2+1D6),-3D3/(p^2+1D6),0),list(12D3,15D3,6.8D3,18D3,1/(1D-7*p)),1,3)
```

Mathématique & Informatique

QUADRIPÔLES

Définition



On pose :

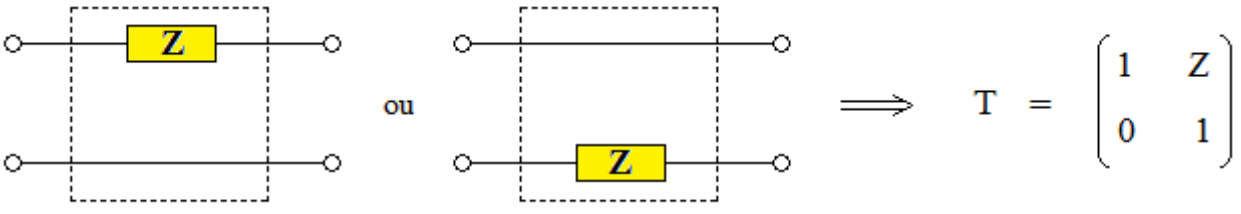
$$\begin{pmatrix} U_e \\ I_e \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} U_s \\ I_s \end{pmatrix} = T \begin{pmatrix} U_s \\ I_s \end{pmatrix}$$

Inversement :

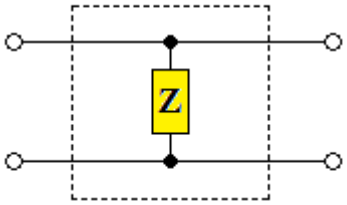
$$\begin{pmatrix} U_s \\ I_s \end{pmatrix} = T^{-1} \begin{pmatrix} U_e \\ I_e \end{pmatrix} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \begin{pmatrix} U_e \\ I_e \end{pmatrix}$$

La matrice T est appelée matrice de transfert du quadripôle Q.
Les éléments de T sont des fractions rationnelles de la variable p de la transformée de **Laplace** dans le cas général, ou des nombres complexes si l'on étudie des signaux sinusoïdaux (p = jω).

Quadripôles élémentaires



$$\Longrightarrow \quad T = \begin{pmatrix} 1 & Z \\ 0 & 1 \end{pmatrix}$$



$$\Longrightarrow \quad T = \begin{pmatrix} 1 & 0 \\ \frac{1}{Z} & 1 \end{pmatrix}$$

NB det T = 1

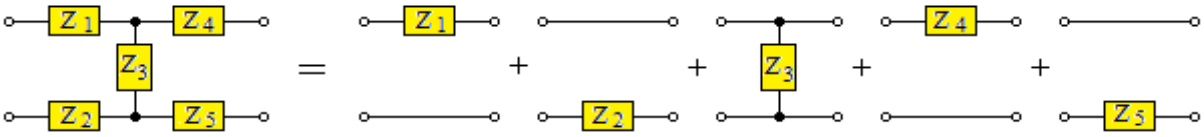
Mathématique & Informatique

Association de quadripôles



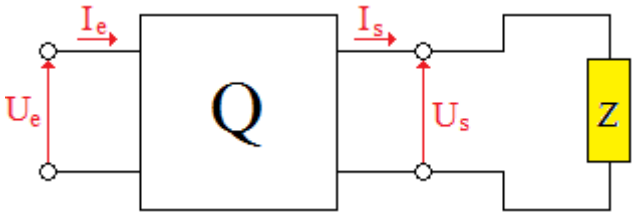
$$\begin{pmatrix} U_e \\ I_e \end{pmatrix} = T_1 \cdots T_n \begin{pmatrix} U_s \\ I_s \end{pmatrix} \quad \Leftrightarrow \quad \begin{pmatrix} U_s \\ I_s \end{pmatrix} = T_n^{-1} \cdots T_1^{-1} \begin{pmatrix} U_e \\ I_e \end{pmatrix}$$

Exemple :



$$T = \begin{pmatrix} 1 & Z_1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & Z_2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{1}{Z_3} & 1 \end{pmatrix} \begin{pmatrix} 1 & Z_4 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & Z_5 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1+\frac{Z_1+Z_2}{Z_3} & Z_1+Z_2+Z_4+Z_5+\frac{(Z_1+Z_2)(Z_4+Z_5)}{Z_3} \\ \frac{1}{Z_3} & 1+\frac{Z_4+Z_5}{Z_3} \end{pmatrix}$$

Impédance de charge



Soit $T = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ la matrice de transfert du quadripôle Q et soit Z son impédance de charge.

On a : $U_s = Z I_s$

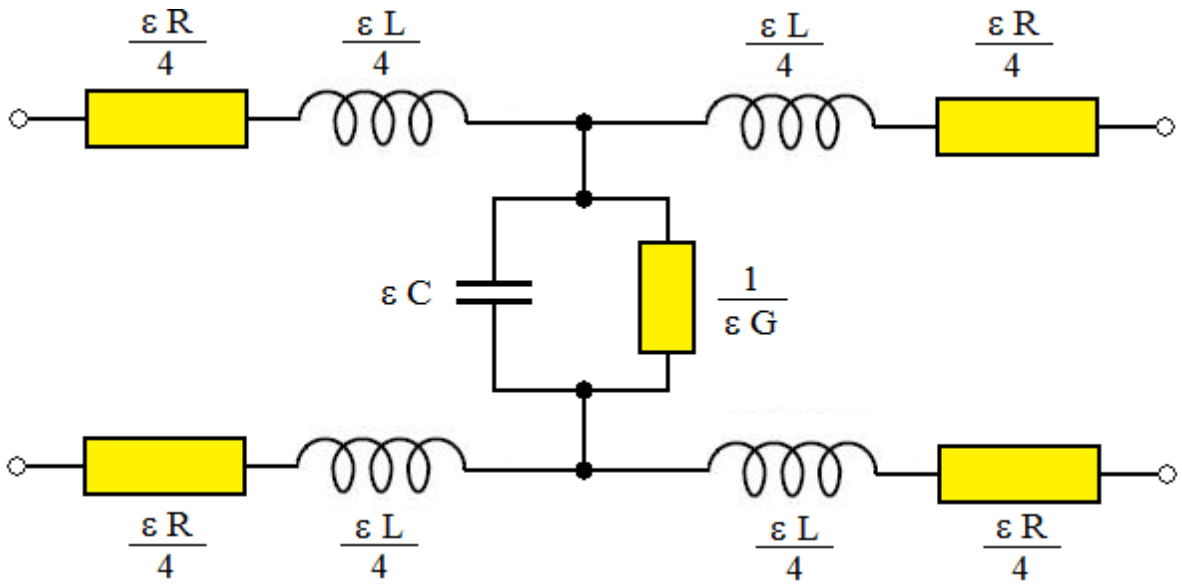
L'impédance d'entrée est : $Z_e = \dfrac{U_e}{I_e} = \dfrac{aZ + b}{cZ + d}$

L'atténuation est : $A = \dfrac{U_s}{U_e} = \dfrac{1}{a + \dfrac{b}{Z}}$

Mathématique & Informatique

Théorie des lignes électriques

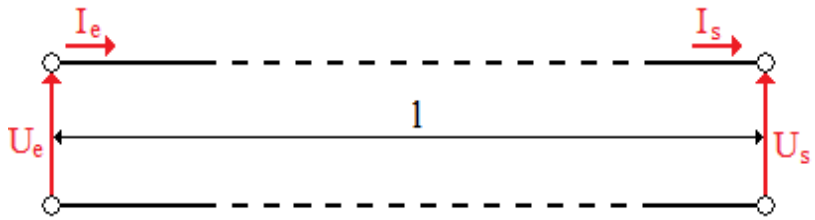
Une ligne électrique est constituée de 2 conducteurs alimentés par un courant alternatif de pulsation ω.
Un élément de ligne de longueur ε est modélisé par le réseau suivant :



R est la résistance linéique de la ligne
L est l'inductance linéique de la ligne
C est la capacité linéique de la ligne
G est la conductance linéique de la ligne

Note G est également appelée perditance et 1/G résistance d'isolement entre les 2 conducteurs

Considérons une ligne de longueur l = n ε composée de n cellules du type ci-dessus mises en cascade.



On montre que :

$$\begin{pmatrix} U_e \\ I_e \end{pmatrix} = \begin{pmatrix} \mathrm{ch}\gamma l & Z_c\,\mathrm{sh}\gamma l \\ \dfrac{\mathrm{sh}\gamma l}{Z_c} & \mathrm{ch}\gamma l \end{pmatrix} \begin{pmatrix} U_s \\ I_s \end{pmatrix}$$

$Z_c = \sqrt{\dfrac{R + jL\omega}{G + jC\omega}}$ est l'impédance caractéristique de la ligne

$\gamma = \sqrt{(R + jL\omega)(G + jC\omega)}$ est la constante de propagation de la ligne

Si l'on ferme la ligne sur une impédance Z, on a : $U_s = Z\, I_s$
On déduit l'impédance d'entrée de la ligne et son affaiblissement :

$$Z_e = Z_c \left(\frac{Z + Z_c \,\mathrm{th}\gamma l}{Z_c + Z \,\mathrm{th}\gamma l} \right)$$

$$A = \frac{1}{\mathrm{ch}\gamma l + \frac{Z_c}{Z}\mathrm{sh}\gamma l}$$

En particulier, si $Z = Z_c$:

$$Z_e = Z_c$$

$$A = \frac{1}{e^{\gamma l}}$$

Ecrivons une fonction Scilab qui calcule la matrice de transfert d'une ligne électrique de longueur l véhiculant un signal sinusoïdal de fréquence f :

```
function T = electric_line(R,L,C,G,l,f)
  omega = 2*%pi*f
  u = R + %i*L*omega
  v = G + %i*C*omega
  Zc = sqrt(u/v)
  Gamma = sqrt(u*v)
  ch = cosh(Gamma*l)
  sh = sinh(Gamma*l)
  T = [ch,sh*Zc;sh/Zc,ch]
endfunction
```

Application numérique

Une ligne téléphonique aérienne est caractérisée par les constantes linéiques suivantes :

- R = 6 mΩ/m
- L = 2.3 µH/m
- C = 5 pF/m
- G = 1.2 nS/m

Calculons la matrice de transfert d'un signal sinusoïdal de fréquence f = 800 Hz transmis sur une longueur l = 1 km :

```
T = electric_line(6e-3,2.3e-6,5e-12,1.2e-9,1e3,800)
```

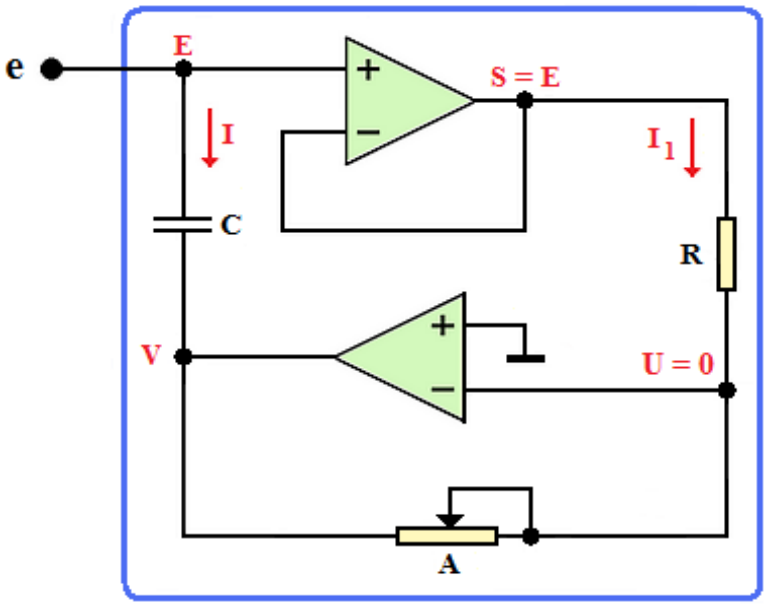
On obtient :

$$T = \begin{pmatrix} 0.9998583 + 0.0000823\mathrm{i} & 5.9993994 + 11.56068\mathrm{i} \\ 0.0000012 + 0.0000251\mathrm{i} & 0.9998583 + 0.0000823\mathrm{i} \end{pmatrix}$$

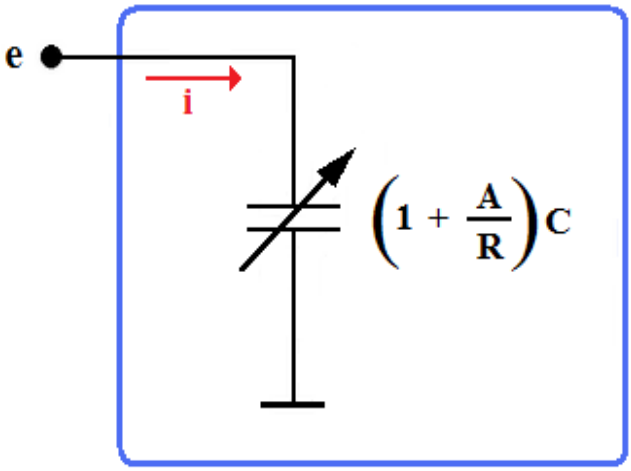
Mathématique & Informatique

Simulateur de capacité

Montrons que le circuit suivant :



est équivalent à une capacité variable :



Calcul

Soit :

E, V les transformées de Laplace des tensions e(t) et v(t)
I, I_1 les transformées de Laplace des courants i(t) et $i_1(t)$

On a les équations suivantes :

$E - V = (1/Cp)I$ (1)
$E = RI_1$ (2)
$V = - AI_1$ (3)

On déduit :

$$E + A(E/R) = (1/Cp)I$$

Et :

$$E = \frac{1}{\left(1 + \frac{A}{R}\right)C\,p}\,I$$

C'est-à-dire :

$$e = \frac{1}{\left(1 + \frac{A}{R}\right)C}\int i\,dt$$

Cette équation est celle d'une capacité de valeur (1 + A/R)C soumise à la ddp e et traversée par le courant i.

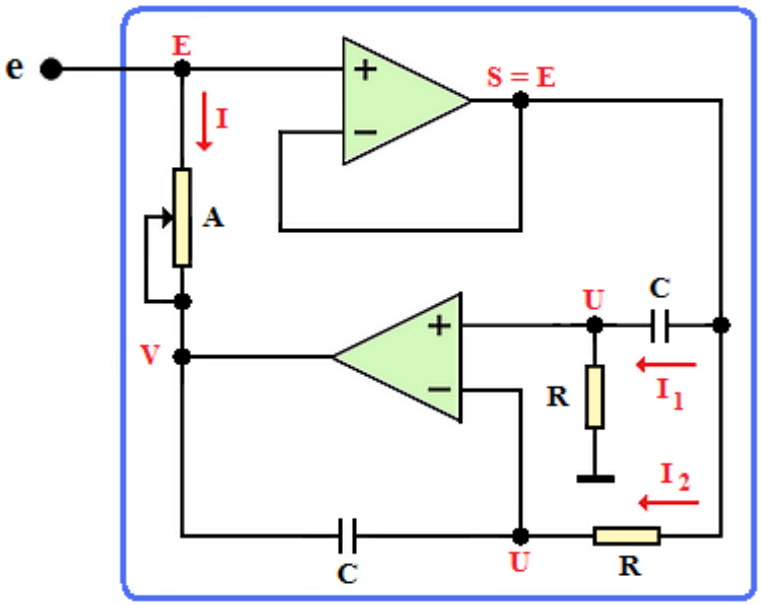
Conclusion

Le circuit est appelé multiplicateur de capacité ; il permet de multiplier la capacité C par le coefficient k = 1 + A/R
Par exemple, avec C = 10 μF, R = 1 kΩ, A = 1 MΩ, on peut simuler une capacité non polarisée de 10 000 μF sous un volume réduit.

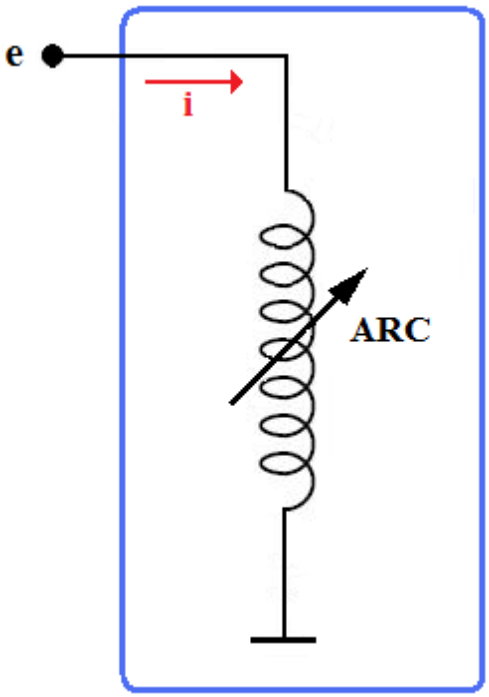
Mathématique & Informatique

Simulateur d'inductance

Montrons que le circuit suivant :



est équivalent à une self variable :



Calcul

E, U, V désignent les transformées de Laplace des tensions.
I, I_1, I_2 désignent les transformées de Laplace des courants.

On a les équations suivantes :

$$E - V = AI \qquad (1)$$
$$U - V = (1/Cp)I_2 \qquad (2)$$
$$E - U = RI_2 \qquad (3)$$
$$E - U = (1/Cp)I_1 \qquad (4)$$
$$U = RI_1 \qquad (5)$$

On déduit :

(2) & (3) : $E - V = RI_2 + (1/Cp)I_2$ (6)
(3) & (5) : $E = RI_1 + RI_2$ (7)
(4) & (5) : $E = RI_1 + (1/Cp)I_1$ (8)
(1) & (6) : $AI = RI_2 + (1/Cp)I_2$ (9)
(7) & (8) : $E = (E - RI_2) + (1/Cp)(E/R - I_2) = E + (1/RCp)E - [RI_2 + (1/Cp)I_2]$ (10)
(9) & (10) : $E = E + (1/RCp)E - AI$ (11)

La dernière équation donne :

$$E = ARCpI$$

En revenant aux grandeurs temporelles :

$$e = ARC\,\frac{di}{dt}$$

En posant L = ARC, cette relation s'écrit :

$$e = L\,\frac{di}{dt}$$

On reconnaît la formule d'une self L soumise à une ddp e et traversée par un courant i.

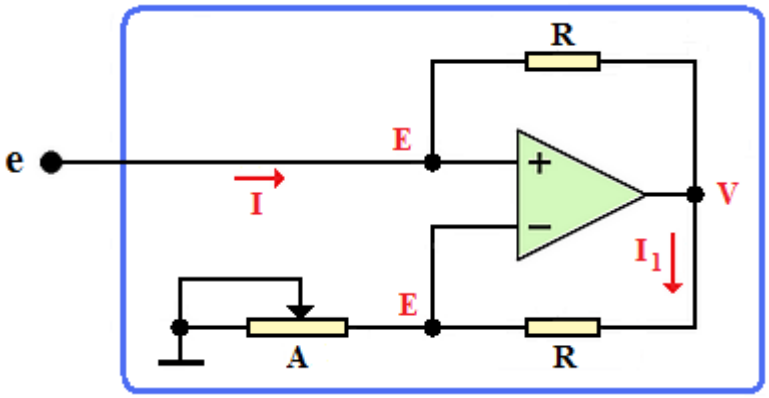
Conclusion

Avec des AOPs on peut simuler des selfs sans bobinage possédant des inductances considérables (1000 H).
L'intérêt est d'éviter la fabrication d'une bobine encombrante (voire irréalisable) dans un circuit.
Remarquons que la self L est théoriquement pure (facteur résistif nul $\Rightarrow$ facteur de qualité infini).
L'inconvénient est que les amplificateurs opérationnels ordinaires ont une bande passante faible (gain × BP = Cte).
Par contre, ils sont bien adaptés aux basses et très basses fréquences.
Ce type de montage entre dans le cadre des circuits **gyrateurs** qui font l'objet d'études particulières.

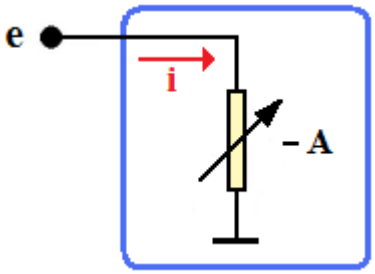
Mathématique & Informatique

Simulateur de résistance

Montrons que le circuit suivant :



est équivalent à une résistance négative variable :



Calcul

E, V sont les transformées de Laplace des tensions e(t) et v(t)
I, I_1 sont les transformées de Laplace des courants i(t) et $i_1(t)$

On a :

$$E - V = RI \qquad (1)$$
$$V - E = RI_1 \qquad (2)$$
$$E = AI_1 \qquad (3)$$

On déduit : $I = - I_1$

D'où : $E = - AI$

Donc : $e = -A\,i$

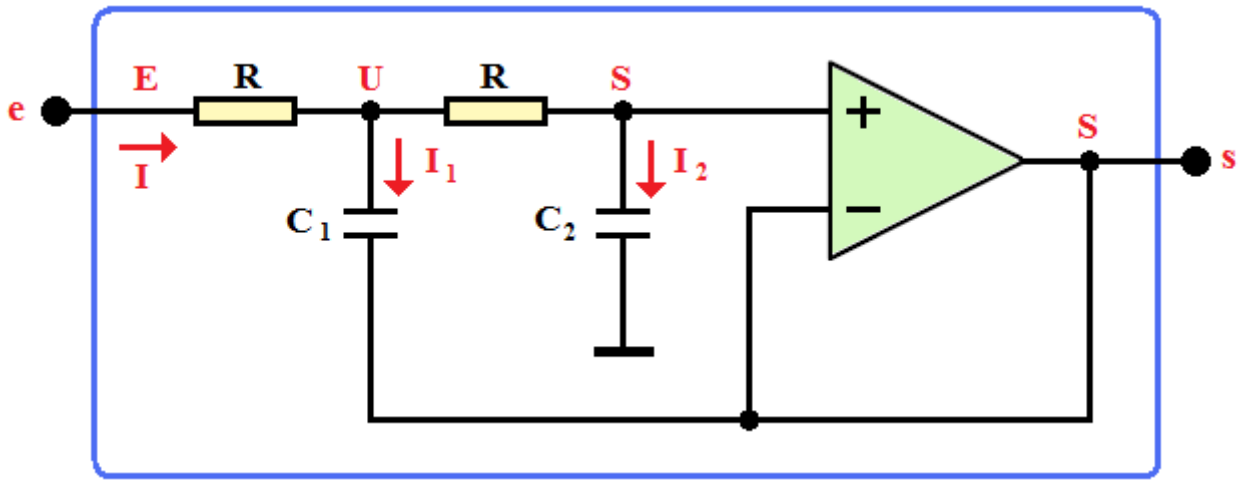
Tout se passe comme si une résistance - A était soumise à la ddp e et traversée par le courant i.

Conclusion

Avec un AOP, il est possible de simuler une résistance négative bien qu'un tel composant passif n'existe pas.

Mathématique & Informatique

Filtre passe-bas d'ordre 2



Problème

1) Calculer les caractéristiques de ce filtre actif : fonction de transfert, fréquence de coupure, atténuation

2) Représenter la courbe du gain g en décibels en fonction de la pulsation ω lorsque $e(t) = a \sin(\omega t + \varphi)$

3) Tracer la courbe s(t) lorsque e(t) est la fonction de **Heaviside** u(t) et lorsque e(t) est la fonction de **Dirac** $\delta(t)$

On prend : $R = 10\ k\Omega$, $C_1 = 20\ nF$, $C_2 = 10\ nF$ (filtre de **Butterworth**)

Rappel

Un signal complexe est composé d'une multitude de fréquences (voir transformée de **Fourier** du signal).
Le but d'un filtre est de séparer les fréquences utiles des fréquences indésirables. On réalise des filtres passe-bas, passe-haut, passe-bande et coupe-bande. Un filtre passif n'utilise que des composants passifs (résistances, condensateurs, selfs), alors qu'un filtre actif utilise en plus des composants actifs (diodes, transistors, AOPs).

Fonction de transfert du filtre

C'est le rapport de la transformée de Laplace de la sortie sur la transformée de Laplace de l'entrée :

$$H = \frac{S}{E}$$

On a :

$E - U = RI$ (1)
$U - S = (1/C_1 p) I_1$ (2)
$U - S = RI_2$ (3)
$S = (1/C_2 p) I_2$ (4)
$I = I_1 + I_2$ (5)

D'où :

(1) : $E = U + RI$
(5) : $E = U + RI_1 + RI_2$
(3) : $E = S + RI_1 + 2RI_2$
(2) & (3) : $E = S + R^2C_1pI_2 + 2RI_2$
(4) : $E = S + (R^2C_1p + 2R)C_2pS$

La dernière égalité donne S/E en fonction de R, C_1, C_2 :

$$H = \frac{1}{R^2C_1C_2p^2 + 2RC_2p + 1}$$

Fréquence de coupure du filtre

C'est la fréquence $f_c = \omega_c/2\pi$ telle que :

$$H = \frac{1}{\frac{p^2}{\omega_c^2} + \frac{1}{Q}\frac{p}{\omega_c} + 1}$$

On obtient immédiatement :

$$f_c = \frac{1}{2\pi R\sqrt{C_1C_2}}$$

Le facteur de qualité du filtre est :

$$Q = \frac{1}{2}\sqrt{\frac{C_1}{C_2}}$$

Atténuation du filtre

Supposons e et s sinusoïdales. Le rapport s/e s'obtient en remplaçant p par $j\omega$ dans H :

$$\frac{s}{e} = \frac{1}{(1 - R^2C_1C_2\omega^2) + j(2RC_2\omega)}$$

Le gain G du filtre est égal au module de s/e :

$$G = \left|\frac{s}{e}\right| = \frac{1}{\sqrt{(1 - R^2C_1C_2\omega^2)^2 + (2RC_2\omega)^2}}$$

En décibels :

$$g = 20\log_{10}G = -10\log_{10}\left[1 + 2R^2C_2(2C_2 - C_1)\omega^2 + R^4C_1^2C_2^2\omega^4\right]$$

En particulier, à la pulsation $\omega = \omega_c$ on a G = Q, et :

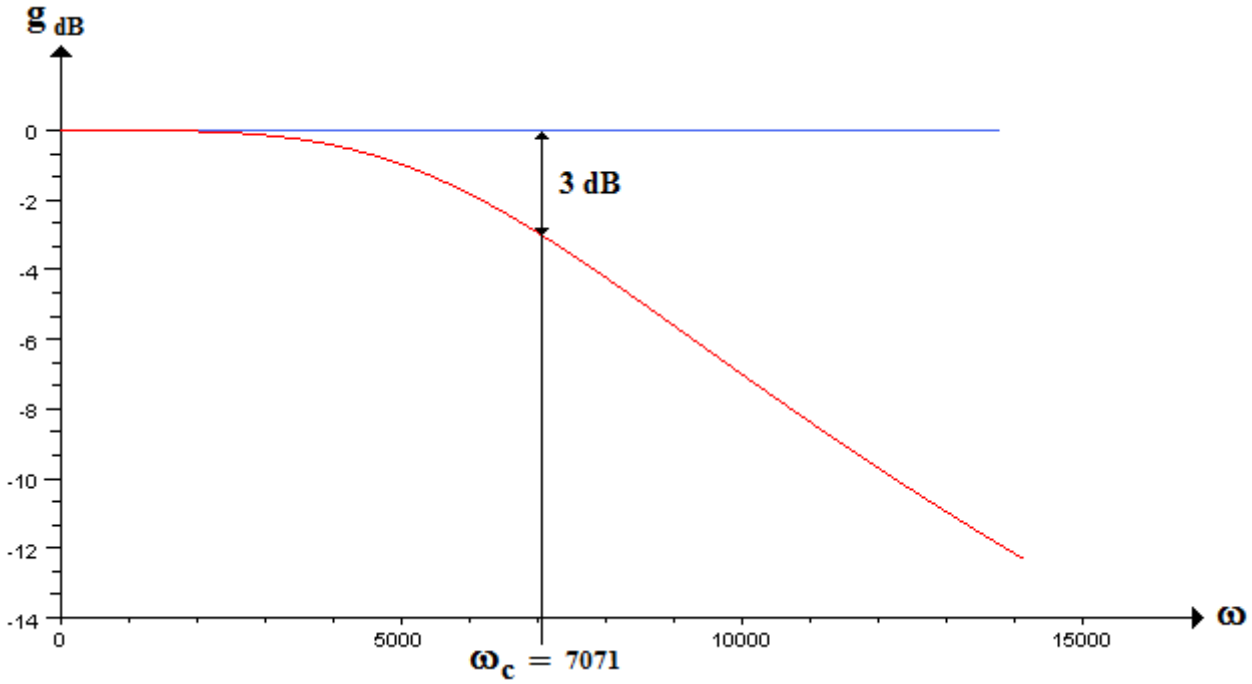
$$g_{\omega_c} = 20\log_{10} Q = -10\log_{10}\left(\frac{4C_2}{C_1}\right)$$

La pente limite du filtre pour un doublement de la fréquence (1 octave) est donnée par :

$$\lim_{\omega\to\infty}(g_{2\omega} - g_{\omega}) = -10\log_{10} 2^4 = -12dB/octave$$

Avec les valeurs choisies pour R, C_1, C_2, traçons la courbe g = f(ω) :

```
R = 10D3 ; C1 = 20D-9 ; C2 = 10D-9 ;
omegac = 1/(R*sqrt(C1*C2)) ;
omega = linspace(0, 2*omegac, 300) ;
gain = - 10*log10(1 + 2*R^2*C2*(2*C2-C1)*omega.^2 + R^4*C1^2*C2^2*omega.^4) ;
scf(0) ; plot2d(omega, gain, 5) ; plot2d(omegac, 0, -1)
```



Equation différentielle du filtre

L'expression de la fonction de transfert fournit l'équation polynomiale en S :

$$R^2C_1C_2p^2S + 2RC_2pS + S = E$$

Elle est équivalente à l'équation différentielle en s :

$$\frac{d^2s}{dt^2} + \frac{2}{RC_1}\frac{ds}{dt} + \frac{1}{R^2C_1C_2}s = \frac{e}{R^2C_1C_2}$$

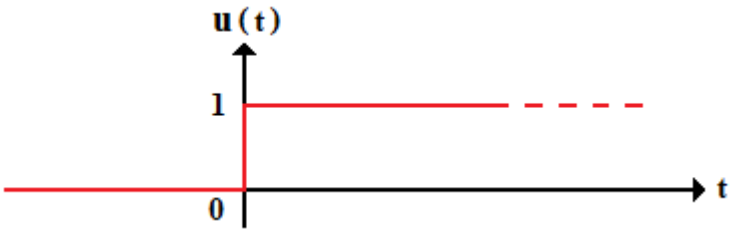
Le programme Scilab de résolution est le suivant :

```
function sp = f(t,s)
   y = e(t)
   sp(1) = s(2)
   sp(2) = y/(R*R*C1*C2) - (2/(R*C1))*s(2) - (1/(R*R*C1*C2))*s(1)
endfunction

s = ode([0;0],t0,t,f)
```

Réponse à un échelon unité

Un échelon unité u(t) — voir fonction de **Heaviside** — est une tension constante de 1 volt :



La transformée de Laplace de u(t) est :

$$\int_0^{+\infty} e^{-pt}\,dt = -\frac{1}{p}\left[e^{-pt}\right]_0^{+\infty} = \frac{1}{p}$$

Si E = 1/p alors S = H/p. La réponse indicielle permet de connaître la fonction de transfert divisée par p du circuit.

Programmons la fonction *echelon(t)* :

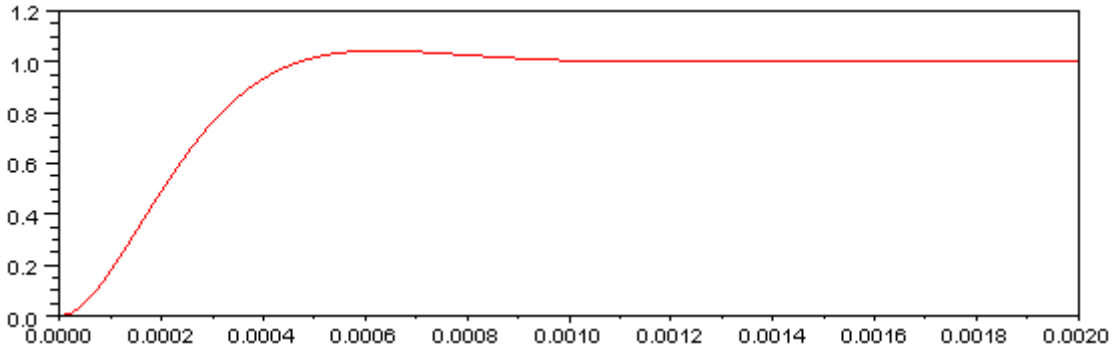
```
function y = echelon(t)
   if t < 0 then, y = 0, else, y = 1, end
endfunctio
```

Pour obtenir la courbe s(t) lorsque e(t) = u(t), on écrit :

```
function sp = f(t,s)
   y = echelon(t)
   sp(1) = s(2)
   sp(2) = y/(R*R*C1*C2) - (2/(R*C1))*s(2) - (1/(R*R*C1*C2))*s(1)
endfunction

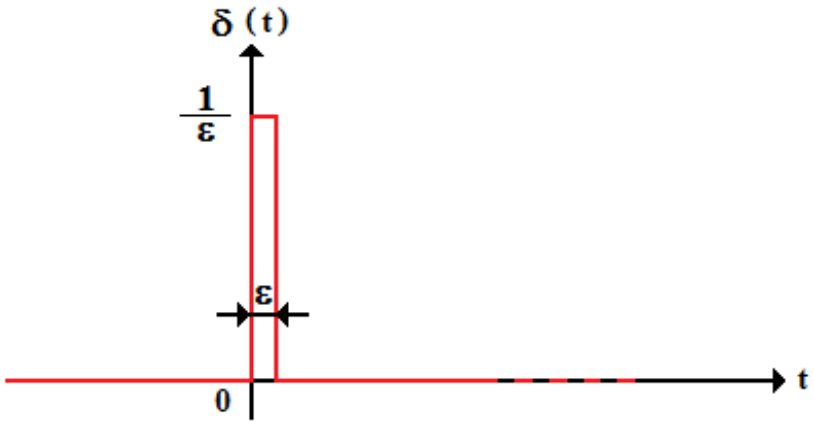
R = 10D3 ; C1 = 20D-9 ; C2 = 10D-9 ;
t0 = 0 ; t1 = 20D-4 ; n = 100 ; t = linspace(t0, t1, n) ;
s = ode([0;0],t0,t,f) ;
scf(1) ; plot2d(t, s(1,:), 5, frameflag=5, rect=[t0,0,t1,1.2], axesflag=1)
```

Résultat :



Réponse à un Dirac

Un Dirac δ(t) — voir fonction de **Dirac** — est une très brève impulsion de très grande amplitude (percussion) :



La transformée de Laplace de δ(t) est :

$$\lim_{\varepsilon \to 0} \int_0^{\varepsilon} \frac{1}{\varepsilon} e^{-pt} dt = \lim_{\varepsilon \to 0} \left(\frac{1}{\varepsilon} \left[-\frac{1}{p} e^{-pt} \right]_0^{\varepsilon} \right) = \lim_{\varepsilon \to 0} \left(\frac{1 - e^{-p\varepsilon}}{p\varepsilon} \right) = 1$$

Si E = 1 alors S = H. La réponse percussionnelle permet de connaître la fonction de transfert du circuit.

Programmons la fonction *dirac(t)* :

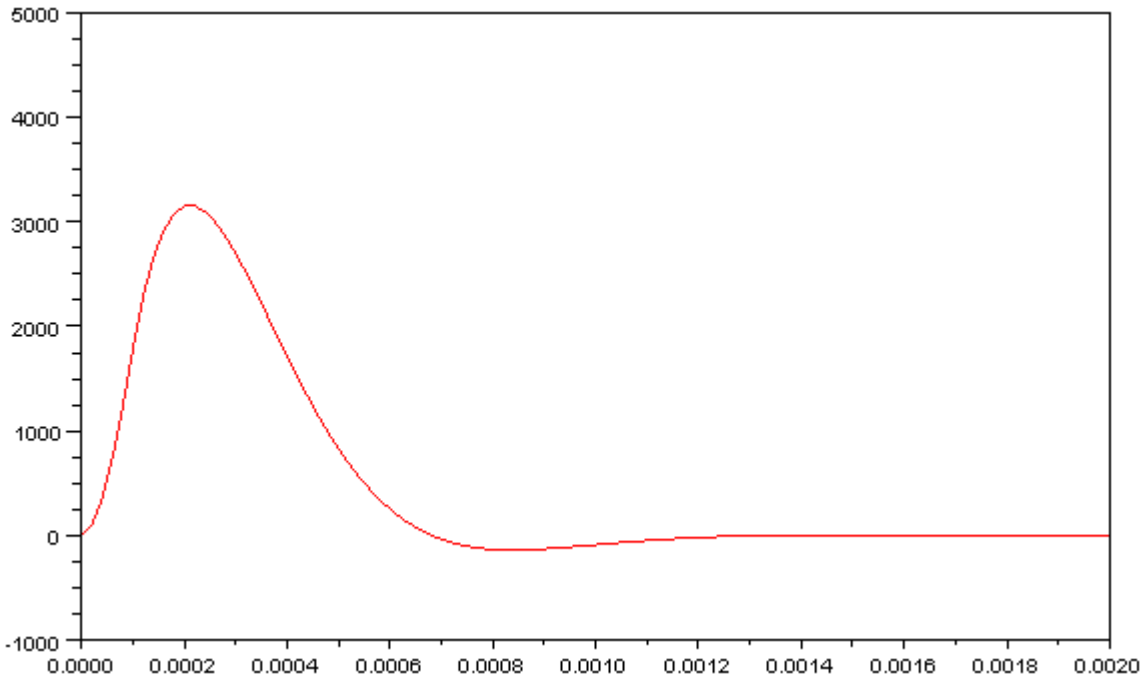
```
function y = dirac(t)
  eps = 1/10000
  if t < 0 | t > eps then
    y = 0
  else
    y = 1/eps
  end
endfunction
```

Pour obtenir la courbe s(t) lorsque e(t) = δ(t), on écrit :

```
function sp = f(t,s)
  y = dirac(t)
  sp(1) = s(2)
  sp(2) = y/(R*R*C1*C2) - (2/(R*C1))*s(2) - (1/(R*R*C1*C2))*s(1)
endfunction

R = 10D3 ; C1 = 20D-9 ; C2 = 10D-9 ;
t0 = 0 ; t1 = 20D-4 ; n = 100 ; t = linspace(t0, t1, n) ;
s = ode([0;0],t0,t,f) ;
scf(2) ; plot2d(t, s(1,:), 5, frameflag=5, rect=[t0,-1000,t1,5000], axesflag=1)
```

Résultat :



En associant en série 2, 3, ... filtres d'ordre 2 on obtient un filtre d'ordre 4, 6, ... avec une atténuation croissante de -24 dB/octave, -36 dB/octave, ...
D'une manière générale, la fonction de transfert d'un filtre passe-bas d'ordre n est H = N(p)/D(p) où D(p) est un polynôme de degré n en p. Certains polynômes permettent de définir des filtres possédant des propriétés particulières. Ainsi, on construit des filtres de **Butterworth**, de **Tchebychev**, de **Cauer**, de **Bessel**, de **Gauss**, de **Legendre**, ...
Documentation sur Internet :

- ⇨ Thierry **Dutoit** : http://tcts.fpms.ac.be/cours/1005-01b/filtanalog.pdf
- ⇨ Sylvain **Larribe** : http://sylvain.larribe.free.fr/CNAM/2004-2005/CNAM_2005_Filtrage.pdf

Il existe des tables donnant les fonctions de transmission normalisées (inverses des fonctions de transfert) de ces différents filtres. On les utilise pour concevoir le filtre dont on a besoin. On fait également appel à des programmes spécialisés pour le calcul des filtres analogiques, notamment dans le cas des filtres de Cauer qui font intervenir les fonctions elliptiques.
Consulter la partie "Traitement du Signal" de Scilab (fonctions *analpf*, *zpbutt*, *zpch1*, *zpch2*, *zpell*, *syslin*, *bode*).

Mathématique & Informatique

FILTRAGE ANALOGIQUE

Dans le cas général, la fonction de transfert d'un filtre analogique s'exprime par :

$$H(p) = \frac{N(p)}{D(p)} = K\,\frac{p^m + \cdots + a_1 p + a_0}{p^n + \cdots + b_1 p + b_0} = K\,\frac{(p - z_1)\cdots(p - z_m)}{(p - p_1)\cdots(p - p_n)}$$

L'ordre du filtre est le degré n ($\geq m$) de D(p)
Les zéros du filtre sont les racines de N(p)
Les pôles du filtre sont les racines de D(p)
Le gain du filtre est la constante K

Note En anglais on utilise la lettre "s" (au lieu de "p") pour désigner la variable de la transformée de Laplace.

La fonction Scilab *analpf*

Cette fonction permet de calculer la fonction de transfert d'un filtre passe-bas :

[hs, pols, zers, gain] = analpf(n, type, [r_1, r_2], omega)

Les paramètres d'entrée

- *n* est l'ordre du filtre
  n = 1, 2, 3, 4, ...

- *type* est le nom du filtre
  'butt' = filtre de Butterworth
  'cheb1' = filtre de Tchebychev 1
  'cheb2' = filtre de Tchebychev 2
  'ellip' = filtre elliptique (filtre de Cauer)

- *[r_1, r_2]* sont 2 coefficients indiquant la variation du gain dans la bande passante ($1-r_1 < gain_{BP} < 1$) et dans la bande coupée ($0 < gain_{BC} < r_2$). Ces 2 atténuations sont souvent exprimées en décibels : A_1 dB et A_2 dB.
  On a les relations :

$$-20\log_{10}(1 - r_1) = A_1 \quad \Rightarrow \quad r_1 = 1 - 10^{-\frac{A_1}{20}}$$

$$-20\log_{10}(r_2) = A_2 \quad \Rightarrow \quad r_2 = 10^{-\frac{A_2}{20}}$$

- *omega* est la pulsation de coupure ω_c exprimée en radians par seconde (rad/s)
  Elle est liée à la fréquence de coupure f_c exprimée en Hertz (Hz) par la relation : $\omega_c = 2\pi f_c$
  Pour obtenir une fonction de transfert normalisée on prend **ω_c = 1 rad/s**

Les paramètres de sortie

- *hs* est la fonction de transfert du filtre (fraction rationnelle de la variable s)

- *pols* est le vecteur des racines dans $\mathbb{C}$ du dénominateur de hs

- *zers* est le vecteur des racines dans $\mathbb{C}$ du numérateur de hs

- *gain* est le gain du filtre : hs ≡ gain * poly(zers,'s') / poly(pols,'s')

Exemple 1

Quelle est la fonction de transfert normalisée et quels sont les pôles et les zéros d'un filtre passe-bas de Butterworth d'ordre 4 ?

On écrit sous Scilab :

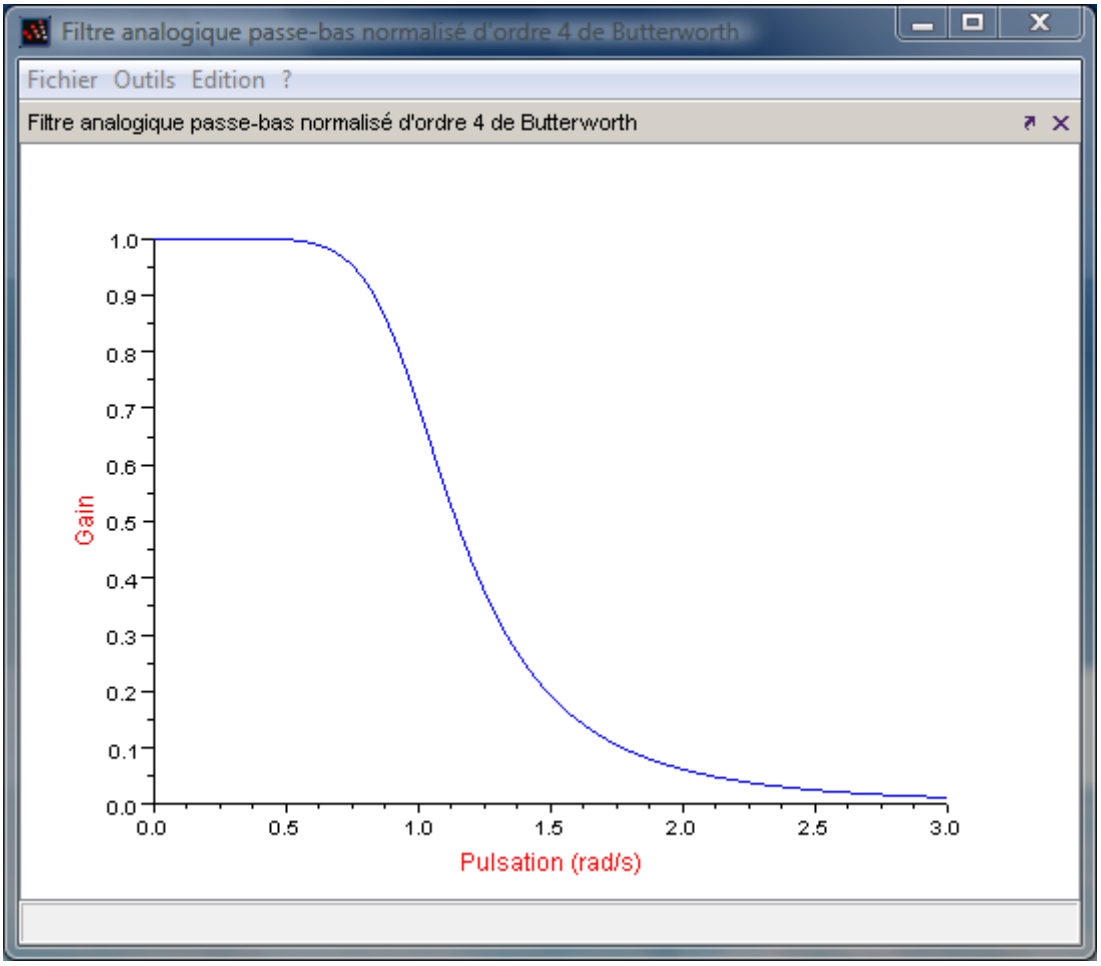
```
[hs1,pols1,zers1,gain1] = analpf(4,'butt',[0,0],1)
```

On obtient :

$$hs1 = \frac{1}{1 + 2.6131259s + 3.4142136s^2 + 2.6131259s^3 + s^4}$$

p1 = - 0.3826834 + 0.9238795i z1 = [ ]
p2 = - 0.9238795 + 0.3826834i z2 = [ ]
p3 = - 0.9238795 - 0.3826834i z3 = [ ]
p4 = - 0.3826834 - 0.9238795i z4 = [ ]

Courbe de réponse :



Exemple 2

Quelle est la fonction de transfert normalisée et quels sont les pôles et les zéros d'un filtre passe-bas de Tchebychev 1, d'ordre 4, et d'atténuation maximale 1 dB dans la bande passante ?

On écrit sous Scilab :

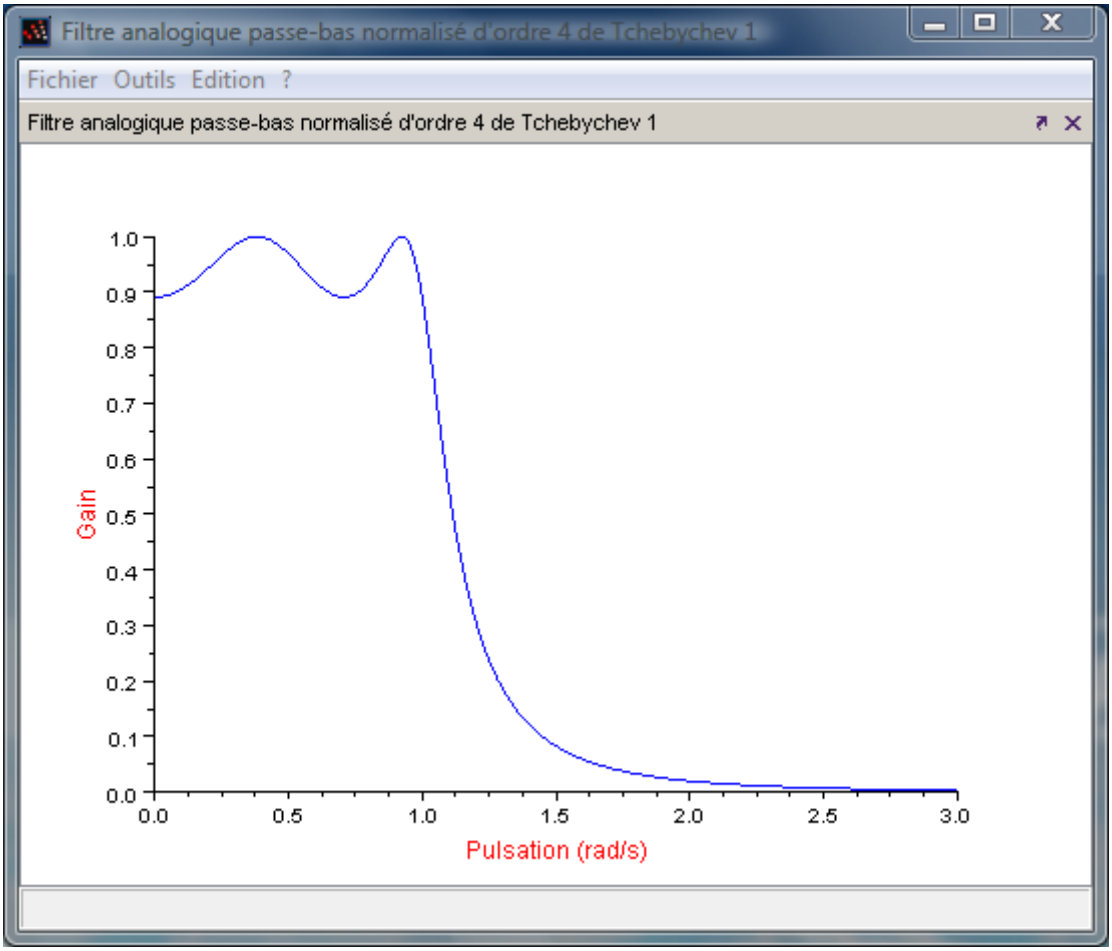
```
[hs2,pols2,zers2,gain2] = analpf(4,'cheb1',[1-10^(-1/20),0],1)
```

On obtient :

$$hs2 = \frac{0.2456533}{0.2756276 + 0.7426194s + 1.4539248s^2 + 0.9528114s^3 + s^4}$$

p1 = - 0.139536 + 0.9833792i z1 = [ ]
p2 = - 0.3368697 + 0.4073290i z2 = [ ]
p3 = - 0.3368697 - 0.4073290i z3 = [ ]
p4 = - 0.139536 - 0.9833792i z4 = [ ]

Courbe de réponse :



Exemple 3

Quelle est la fonction de transfert normalisée et quels sont les pôles et les zéros d'un filtre passe-bas de Tchebychev 2, d'ordre 4, et d'atténuation minimale 20 dB dans la bande coupée ?

On écrit sous Scilab :

[hs3,pols3,zers3,gain3]=analpf(4,'cheb2',[0,10^(-20/20)],1)

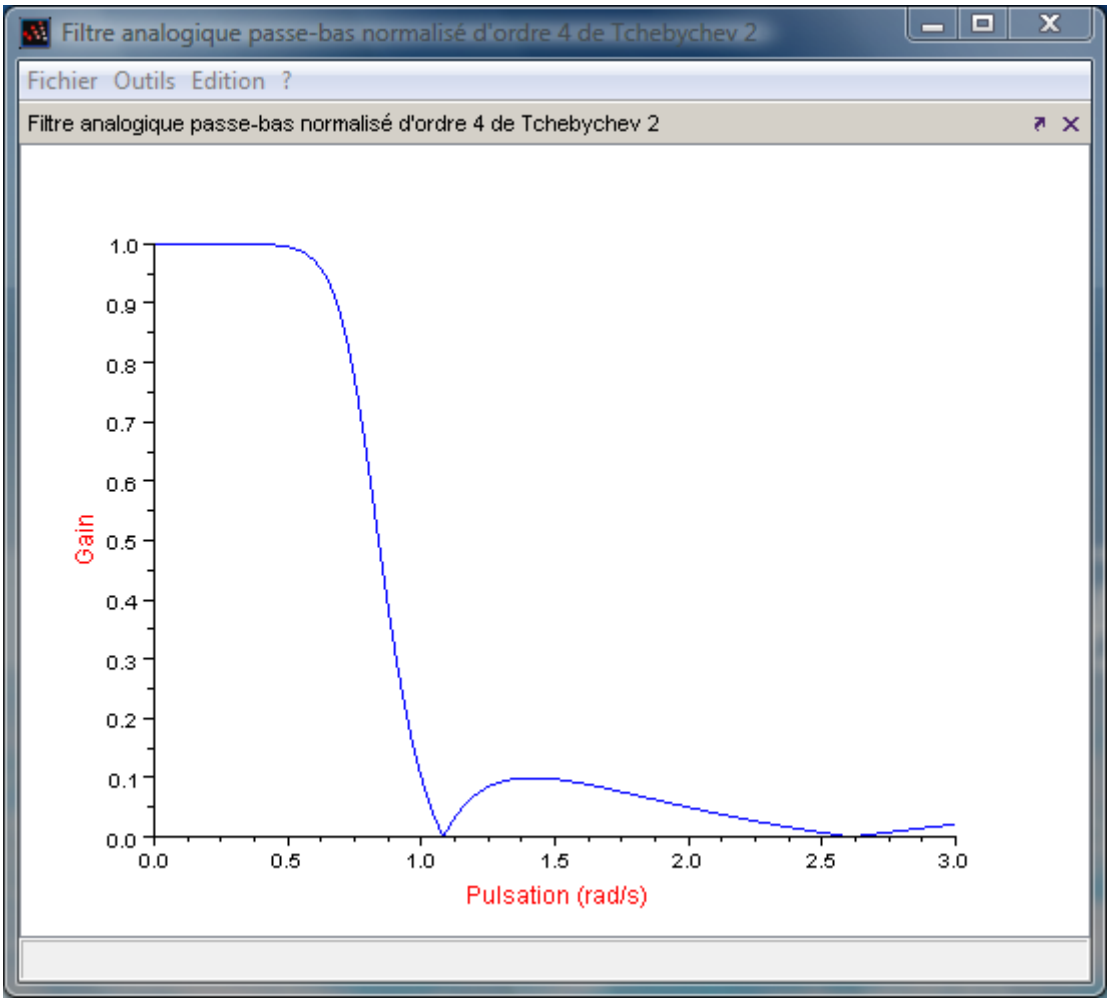
On obtient :

$$hs3 = \frac{0.8 + 0.8s^2 + 0.1s^4}{0.8 + 1.7144706s + 2.6371308s^2 + 2.2614733s^3 + s^4}$$

p1 = - 0.2056459 - 0.7829114i
p2 = - 0.9250908 - 0.6042623i
p3 = - 0.9250908 + 0.6042623i
p4 = - 0.2056459 + 0.7829114i

z1 = 1.0823922i
z2 = 2.6131259i
z3 = - 2.6131259i
z4 = - 1.0823922i

Courbe de réponse :



Exemple 4

Quelle est la fonction de transfert normalisée et quels sont les pôles et les zéros d'un filtre passe-bas de Cauer, d'ordre 4, d'atténuation maximale 1 dB dans la bande passante et d'atténuation minimale 20 dB dans la bande coupée ?

On écrit sous Scilab :

```
[hs4,pols4,zers4,gain4]=analpf(4,'ellip',[1-10^(-1/20),10^(-20/20)],1)
```

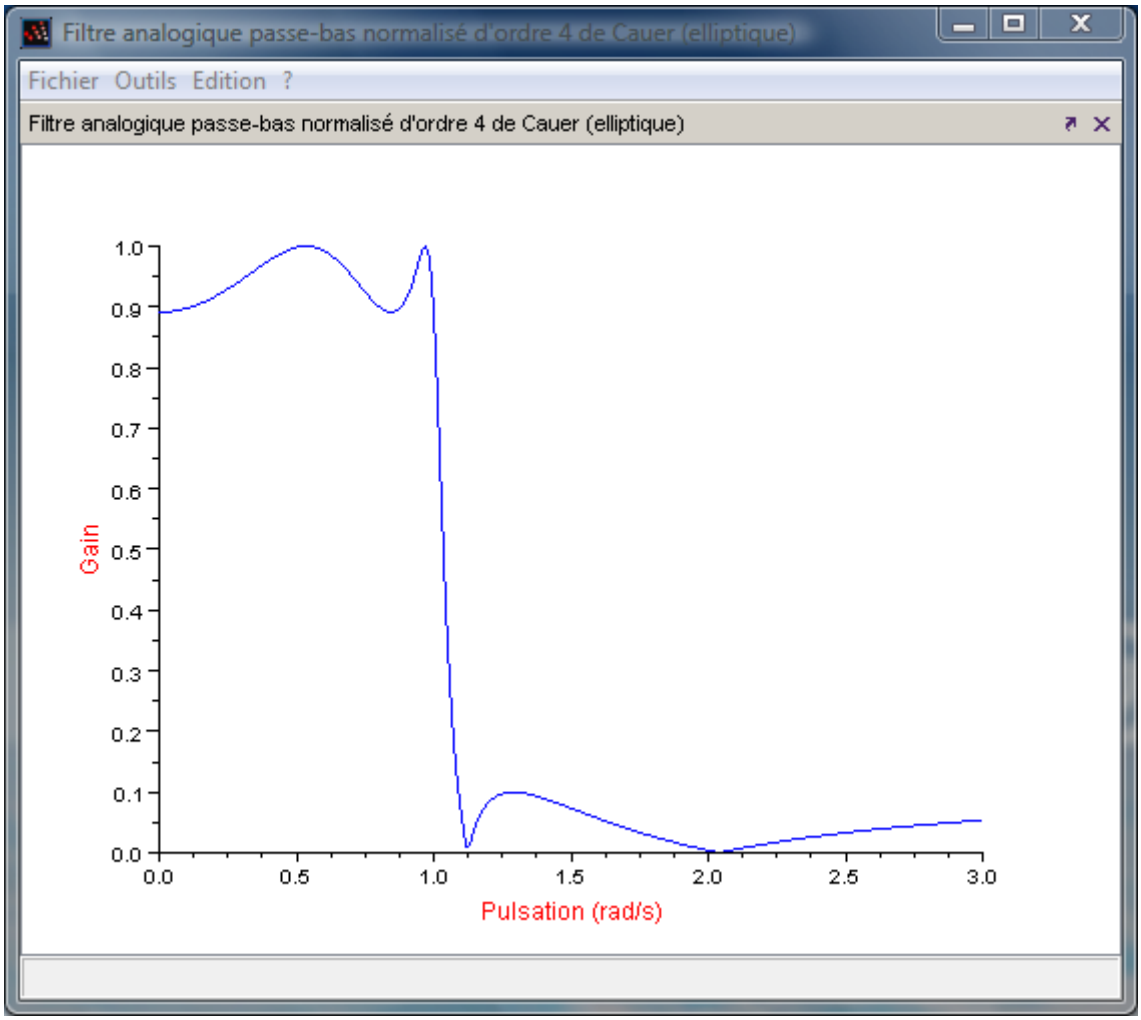
On obtient :

$$hs4 = \frac{0.5255875 + 0.5421963s^2 + 0.1000000s^4}{0.5897189 + 0.8688099s + 1.6765067s^2 + 0.9037750s^3 + s^4}$$

p1 = - 0.4002754 + 0.6509016i
p2 = - 0.0516121 + 1.00365i
p3 = - 0.4002754 - 0.6509016i
p4 = - 0.0516121 - 1.00365i

z1 = 2.0390907i
z2 = 1.1243097i
z3 = - 2.0390907i
z4 = - 1.1243097i

Courbe de réponse :



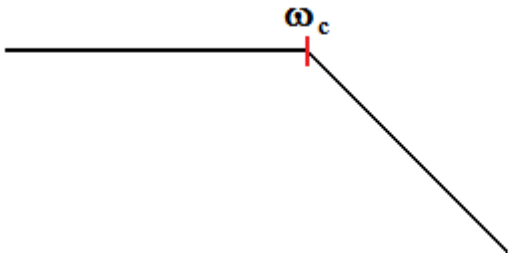
Mathématique & Informatique

Méthode pratique pour concevoir un filtre analogique

1) Calculer avec *analpf* la fonction de transfert h(p) du filtre passe-bas normalisé (ω_c = 1 rad/s) pour le type de filtre requis (Butterworth, Tchebychev, Cauer)

2) Effectuer la transformation permettant d'obtenir la fonction de transfert H(p) du filtre à réaliser :

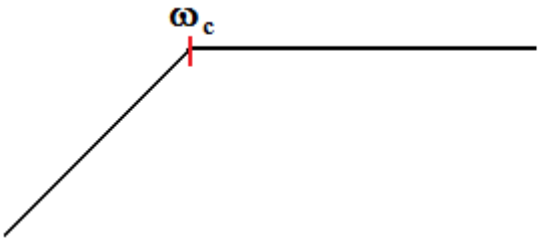
a) Passe-bas



ω_c = pulsation de coupure

Remplacer p par p/ω_c dans h

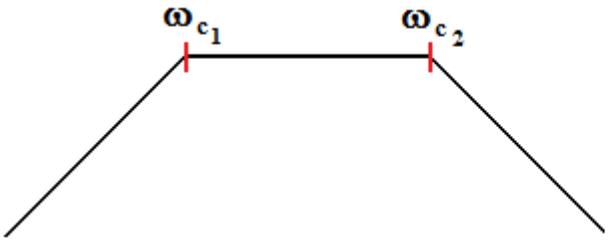
b) Passe-haut



ω_c = pulsation de coupure

Remplacer p par ω_c/p dans h

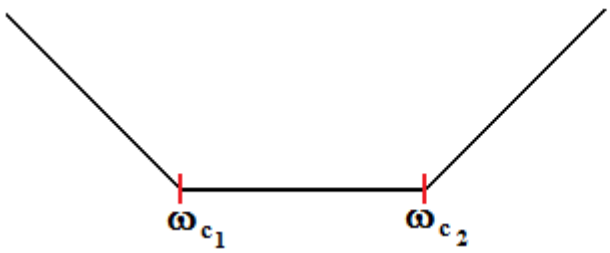
c) Passe-bande



$B = \omega_{c_2} - \omega_{c_1}$ = largeur de bande
$\omega_c^2 = \omega_{c_2} \times \omega_{c_1}$ = pulsation centrale

Remplacer p par $(p^2 + \omega_c^2) / (Bp)$ dans h

d) Coupe-bande



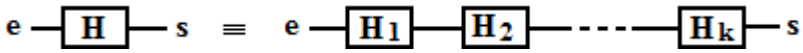
$B = \omega_{c_2} - \omega_{c_1}$ = largeur de bande
$\omega_c^2 = \omega_{c_2} \times \omega_{c_1}$ = pulsation centrale

Remplacer p par $(Bp) / (p^2 + \omega_c^2)$ dans h

3) Décomposer la fraction rationnelle H(p) en un produit de fractions rationnelles H_i(numérique) de sorte que :

$$H = H_1 \times H_2 \times \ \ldots \ \times H_k \quad \text{avec} \quad H_i(p) = N_i(p)/D_i(p)$$

Le filtre de fonction de transfert H est la mise en série de k filtres de fonctions de transfert H_1 , H_2 , ... , H_k



4) Pour chaque H_i(numérique) choisir un circuit RCAO (Résistances + Condensateurs + Amplificateurs Opérationnels) qui possède une fonction de transfert H_i(symbolique) de même expression polynomiale.

5) Identifier les coefficients de H_i(numérique) avec ceux de H_i(symbolique).
Résoudre les équations correspondantes pour déterminer les valeurs des résistances et des condensateurs.
Les choix éventuels doivent conduire à des valeurs proches des valeurs standard pour R et C. Typiquement :

$1\ \Omega < R < 1M\Omega$
$1\ pF < C < 100\ \mu F$

Note

Le système d'équations obtenu en 5) n'est pas linéaire et compte généralement plus d'inconnues que d'équations.
On arrive à le résoudre algébriquement en attribuant des valeurs judicieuses à certaines inconnues.
On peut aussi utiliser la fonction Scilab *fsolve*.

Vocabulaire

On rencontre parfois le terme de "fréquence angulaire de coupure" pour désigner ω_c en rad/s, et même par abus de langage "fréquence de coupure", à ne pas confondre avec la fréquence de coupure f_c exprimée en Hz.

Exemple 1

Reprenons l'exemple du filtre passe-bas d'ordre 2 précédent (schéma de **Sallen et Key**) et recherchons les valeurs à donner à R, C_1, C_2 pour obtenir un filtre passe-bas de **Butterworth** de pulsation de coupure ω_c = 7071 rad/s.
Pour cela, utilisons la fonction Scilab *analpf* :

```
h = analpf(2,'butt',[0,0],1)
```

On obtient la fonction de transfert normalisée du filtre passe-bas recherché :

$$h = \frac{1}{p^2 + \lambda p + 1} = \frac{1}{p^2 + 1.4142136\ p + 1}$$

La fonction de transfert du filtre passe-bas pour ω_c = 7071 rad/s se déduit en remplaçant p par p/ω_c :

$$H = \frac{1}{\frac{1}{\omega_c^2}p^2 + \frac{\lambda}{\omega_c}p + 1} = \frac{1}{2.10^{-8}\,p^2 + 2.10^{-4}\,p + 1}$$

D'autre part, on a vu que le circuit étudié conduisait à la fonction de transfert :

$$H = \frac{1}{R^2 C_1 C_2 p^2 + 2RC_2 p + 1}$$

En identifiant les 2 expressions, on doit avoir :

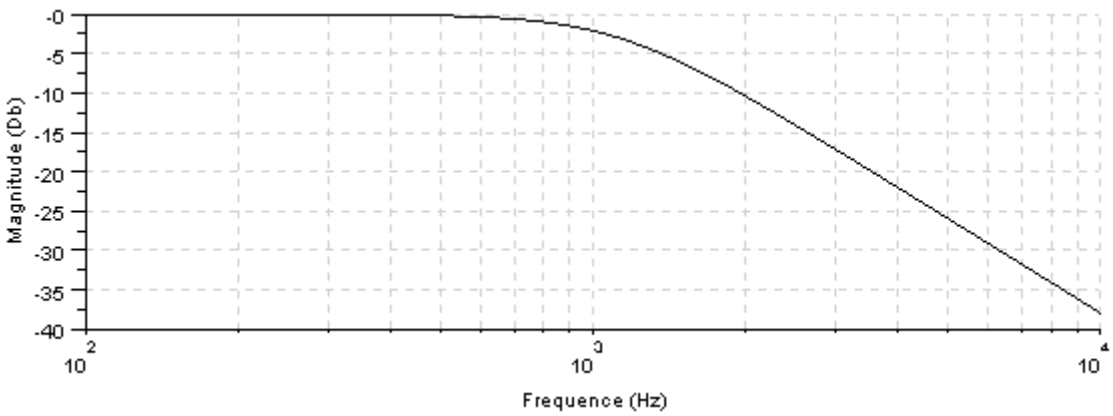
$R^2C_1C_2 = 1/\omega_c^2 = 2.10^{-8}$ et $2RC_2 = \lambda/\omega_c = 2.10^{-4}$

Choisissons R = 10 kΩ, il vient :

$C_2 = 2.10^{-4} / 2.10^4 = 10$ nF et $C_1 = 2.10^{-8} / [(10^4)^2\ 10^{-8}] = 20$ nF

Pour représenter la courbe de réponse du filtre, utilisons les fonctions *syslin* et *bode* :

```
p = poly(0,'p') ;
H = 1/(2D-8*p^2 + 2D-4*p + 1) ;
elatf = syslin('c',H(2),H(3)) ;
bode(elatf,1D2,1D4)
```



Exemple 2

Réalisons un filtre passe-haut de **Tchebychev** de fréquence de coupure f_c = 2 kHz.
Le filtre sera d'ordre 6 avec une ondulation maximale de 0.1 dB dans la bande passante.

Soit h la fonction de transfert du filtre passe-bas normalisé d'ordre 6 de Tchebychev 1 :

$$h = \frac{k}{p^6 + \alpha p^5 + \beta p^4 + \gamma p^3 + \delta p^2 + \varepsilon p + \varsigma}$$

Notons p_1, p_2, p_3, p_1^*, p_2^*, p_3^* les 6 pôles de ce filtre. On écrit h sous la forme :

$$h = \frac{k}{(p-p_1)(p-p_1^*)(p-p_2)(p-p_2^*)(p-p_3)(p-p_3^*)}$$

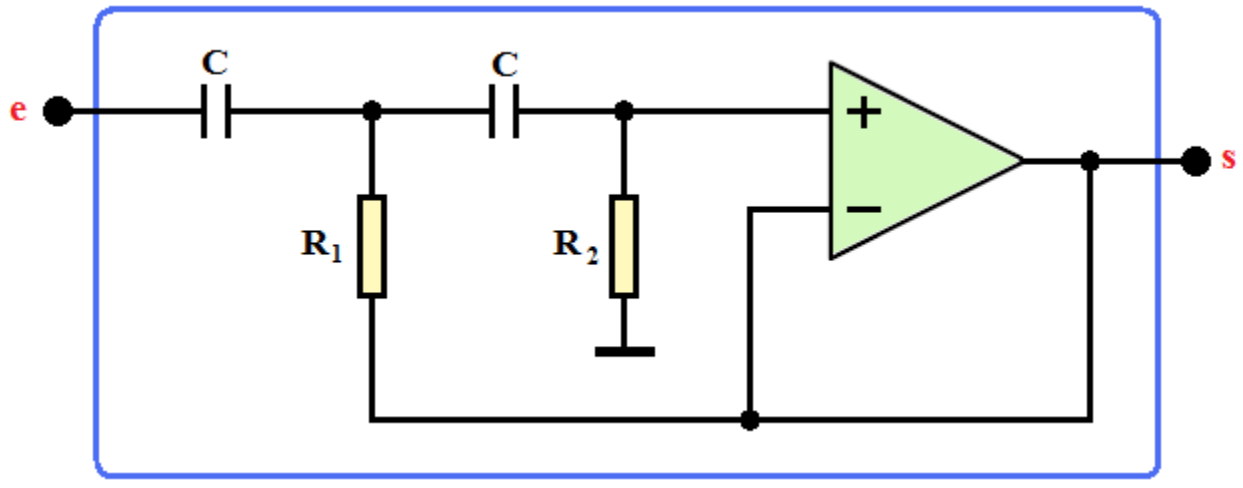
$$= \frac{k}{[p^2 - 2\mathrm{Re}(p_1)p + |p_1|^2][p^2 - 2\mathrm{Re}(p_2)p + |p_2|^2][p^2 - 2\mathrm{Re}(p_3)p + |p_3|^2]}$$

La fréquence de coupure f_c (Hz) correspond à une pulsation $\omega_c = 2\pi f_c$ (rad/s)
Pour obtenir la fonction de transfert H du filtre passe-haut recherché, on remplace p par ω_c/p dans h. On obtient :

$$H = \frac{k\,p^6}{[|p_1|^2 p^2 - 2\mathrm{Re}(p_1)\omega_c p + \omega_c^2][|p_2|^2 p^2 - 2\mathrm{Re}(p_2)\omega_c p + \omega_c^2][|p_3|^2 p^2 - 2\mathrm{Re}(p_3)\omega_c p + \omega_c^2]}$$

$$= \frac{\frac{|p_1|^2}{\omega_c^2}p^2}{\frac{|p_1|^2}{\omega_c^2}p^2 - \frac{2\mathrm{Re}(p_1)}{\omega_c}p + 1} \times \frac{\frac{|p_2|^2}{\omega_c^2}p^2}{\frac{|p_2|^2}{\omega_c^2}p^2 - \frac{2\mathrm{Re}(p_2)}{\omega_c}p + 1} \times \frac{\frac{|p_3|^2}{\omega_c^2}p^2}{\frac{|p_3|^2}{\omega_c^2}p^2 - \frac{2\mathrm{Re}(p_3)}{\omega_c}p + 1} \times \frac{k}{|p_1|^2|p_2|^2|p_3|^2}$$

La fonction de transfert H est le produit de 4 fonctions de transfert H_1, H_2, H_3 et H_4
D'une part, prenons la structure élémentaire du filtre passe-haut d'ordre 2 de **Sallen et Key** :

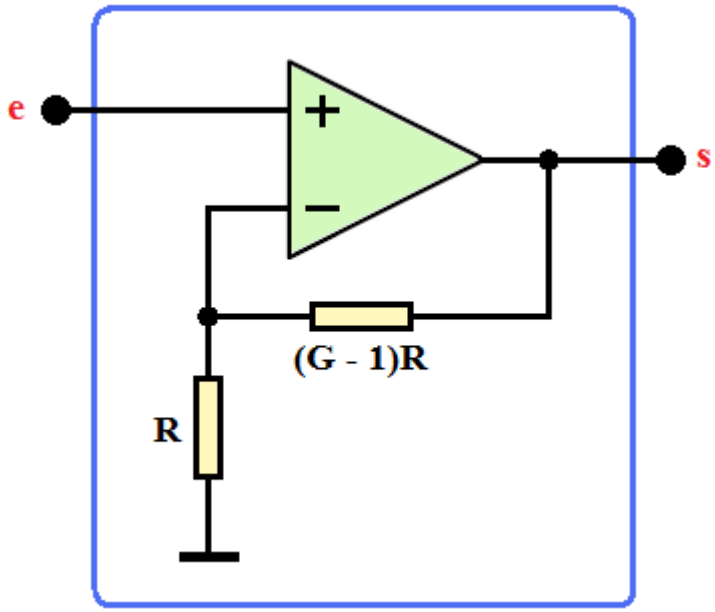


Comme pour le filtre passe-bas précédemment étudié, on établit que la fonction de transfert de ce circuit est :

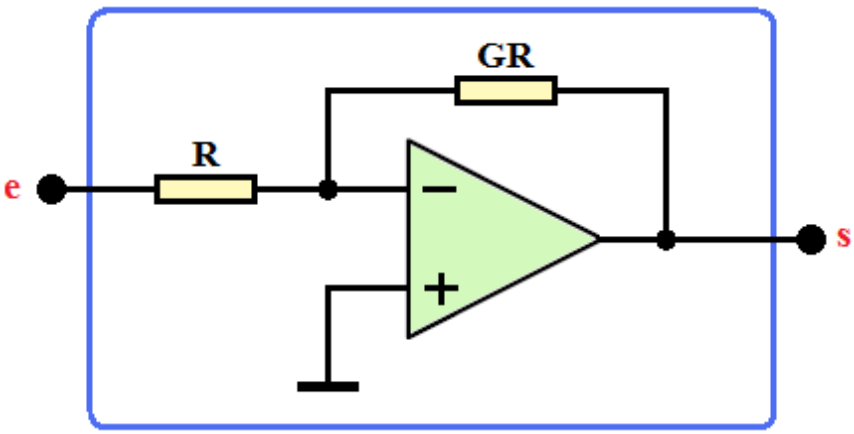
$$H = \frac{R_1 R_2 C^2 p^2}{R_1 R_2 C^2 p^2 + 2 R_1 C p + 1}$$

D'autre part, pour fixer le gain en tension du filtre, utilisons l'un des 2 montages fondamentaux :

- AOP non inverseur (H = G ≥ 1, très grande impédance d'entrée, faible impédance de sortie)



- AOP inverseur (H = - G ≤ 0, impédance d'entrée égale à R, faible impédance de sortie)



Conclusion.

Le filtre passe-haut recherché est la mise en série de 3 filtres passe-haut élémentaires de **Sallen et Key** (fonctions de transfert H_1, H_2, H_3), plus éventuellement un amplificateur de signal (fonction de transfert H_4)

Calcul des résistances et des condensateurs

Filtre n°1

On attribue une valeur arbitraire à C
L'identification $2R_1C = -2Re(p_1)/\omega_c$ donne : $R_1 = -Re(p_1)/(C\omega_c)$
L'identification $R_1R_2C^2 = |p_1|^2/\omega_c^2$ donne : $R_2 = |p_1|^2/(R_1C^2\omega_c^2)$

Filtre n°2

On attribue une valeur arbitraire à C'
L'identification $2R_1'C' = -2Re(p_2)/\omega_c$ donne : $R_1' = -Re(p_2)/(C'\omega_c)$
L'identification $R_1'R_2'C'^2 = |p_2|^2/\omega_c^2$ donne : $R_2' = |p_2|^2/(R_1'C'^2\omega_c^2)$

Filtre n°3

On attribue une valeur arbitraire à C''
L'identification $2R_1''C'' = -2Re(p_3)/\omega_c$ donne : $R_1'' = -Re(p_3)/(C''\omega_c)$
L'identification $R_1''R_2''C''^2 = |p_3|^2/\omega_c^2$ donne : $R_2'' = |p_3|^2/(R_1''C''^2\omega_c^2)$

Amplificateur

$G = k/(|p_1|^2\,|p_2|^2\,|p_3|^2)$
On attribue une valeur arbitraire à R et on remplace la résistance de contre-réaction par un potentiomètre de valeur standard.

Remarques

- Dans le schéma final, les filtres seront ordonnés dans le sens des facteurs de qualité (coefficients de surtension) croissants, c'est-à-dire dans le sens des $Q = \frac{1}{2}(R_2/R_1)^{\frac{1}{2}}$ croissants.
- On peut réaliser le filtre passe-haut d'ordre 6 sans le dernier AOP, c'est-à-dire avec 3 AOPs uniquement.
  Le schéma à 4 AOPs présente l'avantage de pouvoir régler le gain global indépendamment des 3 filtres élémentaires.
- Il est facile de généraliser le calcul précédent à un filtre d'ordre 2n constitué de n filtres élémentaires d'ordre 2.

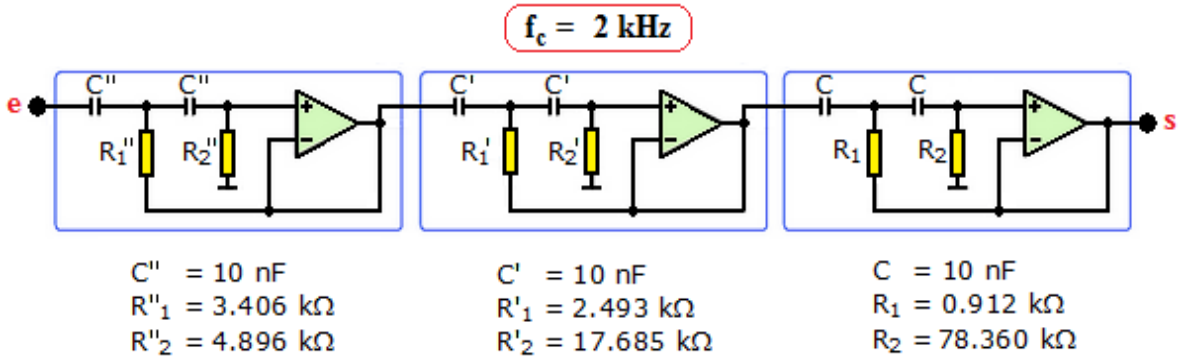
Finalement, on est conduit au programme suivant :

```
funcprot(0) ; lines(0)

//---------- Données
fc = 2D3 ;
A = 0.1 ;
//---------- Fonction de transfert normalisée
[h,pols,zers,k] = analpf(6,'cheb1',[1-10^(-A/20),0],1) ;
p1 = pols(1) ;
p2 = pols(2) ;
p3 = pols(3) ;
//---------- Grahique
fr = 0:0.01:3 ;
hf = freq(h(2),h(3),%i*fr) ;
hm = abs(hf) ;
scf(0) ;
f = gcf() ;
f.figure_name = "Filtre analogique passe-bas normalisé d''ordre 6 de Tchebychev" ;
a = gca() ;
a.axes_visible = "on" ; a.box="off" ;
a.x_location = "origin" ; a.y_location = "left" ;
a.x_label.text = "Pulsation (rad/s)" ; a.y_label.text = "Gain" ;
a.x_label.font_foreground = 5 ; a.y_label.font_foreground = 5 ;
a.x_label.font_size = 2 ; a.y_label.font_size = 2 ;
plot2d(fr,hm,2)
//---------- Transformation en filtre passe-haut
omegac = 2*%pi*fc ;
p = poly(0,'p') ;
N = k*p^6 ;
D1 = abs(p1)^2*p^2 - 2*real(p1)*omegac*p + omegac^2 ;
D2 = abs(p2)^2*p^2 - 2*real(p2)*omegac*p + omegac^2 ;
D3 = abs(p3)^2*p^2 - 2*real(p3)*omegac*p + omegac^2 ;
D = D1*D2*D3 ;
H = N/D ;
//---------- Diagramme de Bode
scf(1) ;
elatf = syslin('c',H(2),H(3)) ;
bode(elatf,500,100D3)
title('Filtre analogique passe-haut d''ordre 6 de Tchebychev      Fréquence de coupure :   2 kHz')
//---------- Calcul des résistances et des condensateurs
C = 10D-9
R1 = -real(p1)/(C*omegac)
R2 = abs(p1)^2/(R1*C^2*omegac^2)
Cp = 10D-9
Rp1 = -real(p2)/(Cp*omegac)
Rp2 = abs(p2)^2/(Rp1*Cp^2*omegac^2)
Cs = 10D-9
Rs1 = -real(p3)/(Cs*omegac)
Rs2 = abs(p3)^2/(Rs1*Cs^2*omegac^2)
//---------- Gain
G = k/(abs(p1)^2*abs(p2)^2*abs(p3)^2)
```

Mathématique & Informatique

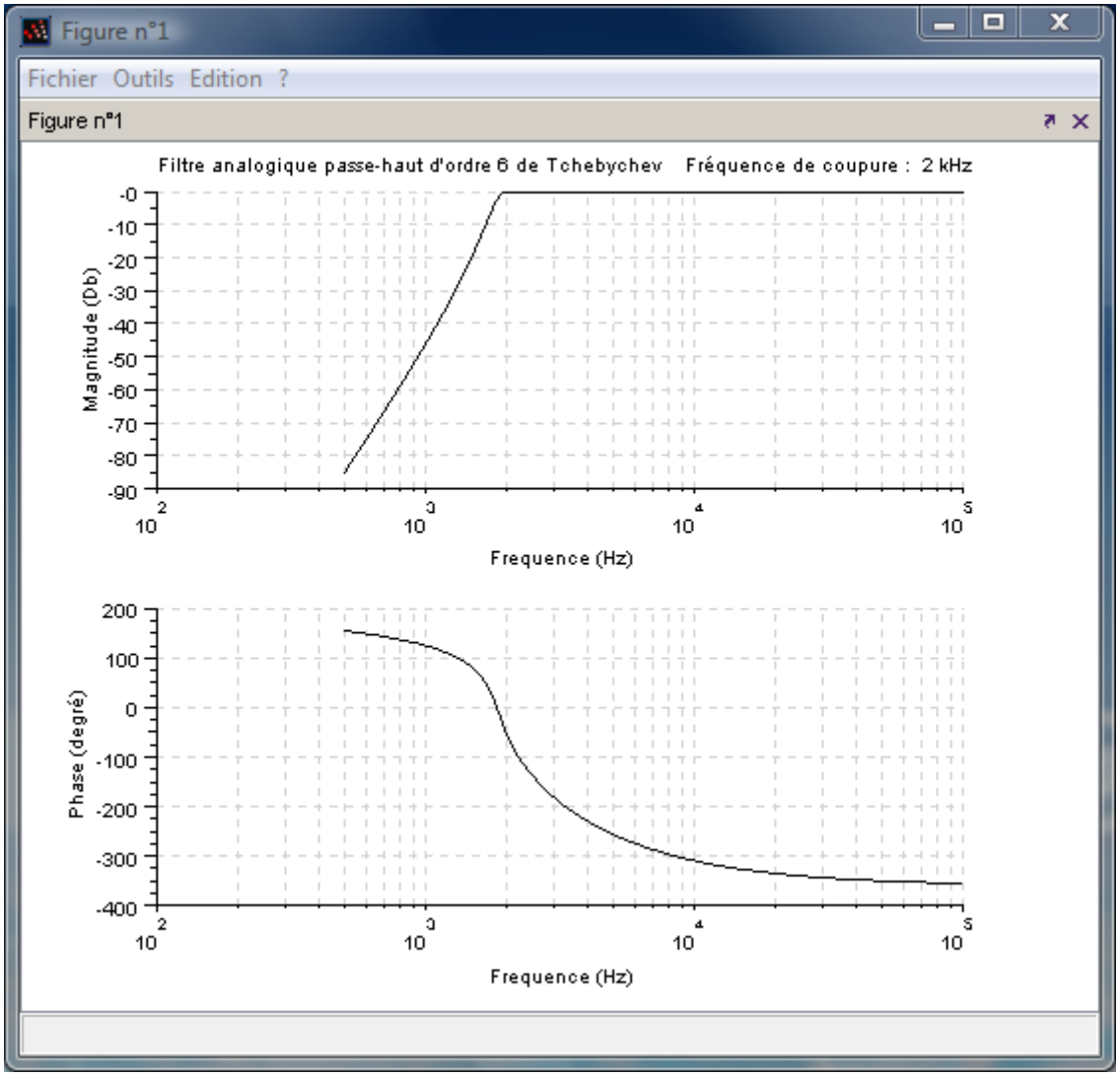
Résultat :



C'' = 10 nF
R''_1 = 3.406 kΩ
R''_2 = 4.896 kΩ

C' = 10 nF
R'_1 = 2.493 kΩ
R'_2 = 17.685 kΩ

C = 10 nF
R_1 = 0.912 kΩ
R_2 = 78.360 kΩ



Note G = 0.9885531 ≈ 1

Exemple 3

Réalisation d'un filtre passe-bande d'ordre 12 pour la plage de fréquence 2 kHz - 20 kHz.
On met en série 2 filtres de **Tchebychev** d'ordre 6 et d'ondulation maximale 0.1 dB dans la bande passante :
un premier filtre passe-bas suivi d'un deuxième filtre passe-haut.
Dans l'exemple 2, on a déjà calculé le filtre passe-haut de fréquence de coupure $f_c = 2$ kHz. Il reste à réaliser le filtre passe-bas de fréquence de coupure $f_c = 20$ kHz.
Reprenons les calculs en remplaçant p par p/ω_c dans h. On obtient :

$$H = \frac{k\omega_c^6}{[p^2 - 2\mathrm{Re}(p_1)\omega_c p + |p_1|^2\omega_c^2][p^2 - 2\mathrm{Re}(p_2)\omega_c p + |p_2|^2\omega_c^2][p^2 - 2\mathrm{Re}(p_3)\omega_c p + |p_3|^2\omega_c^2]}$$

$$= \frac{1}{\frac{1}{|p_1|^2\omega_c^2}p^2 - \frac{2\mathrm{Re}(p_1)}{|p_1|^2\omega_c}p + 1} \times \frac{1}{\frac{1}{|p_2|^2\omega_c^2}p^2 - \frac{2\mathrm{Re}(p_2)}{|p_2|^2\omega_c}p + 1} \times \frac{1}{\frac{1}{|p_3|^2\omega_c^2}p^2 - \frac{2\mathrm{Re}(p_3)}{|p_3|^2\omega_c}p + 1} \times \frac{k}{|p_1|^2|p_2|^2|p_3|^2}$$

On a : $H = H_1 \times H_2 \times H_3 \times H_4$
Comparons H_1, H_2, H_3 avec la fonction de transfert du filtre passe-bas d'ordre 2 de **Sallen et Key** :

$$H = \frac{1}{R^2C_1C_2p^2 + 2RC_2p + 1}$$

Filtre n°1

On attribue une valeur arbitraire à R
L'identification $2RC_2 = -2\mathrm{Re}(p_1)/(|p_1|^2\omega_c)$ donne : $C_2 = -\mathrm{Re}(p_1)/(|p_1|^2\omega_c R)$
L'identification $R^2C_1C_2 = 1/(|p_1|^2\omega_c^2)$ donne : $C_1 = 1/(|p_1|^2\omega_c^2R^2C_2)$

Filtre n°2

On attribue une valeur arbitraire à R'
L'identification $2R'C'_2 = -2\mathrm{Re}(p_2)/(|p_2|^2\omega_c)$ donne : $C'_2 = -\mathrm{Re}(p_2)/(|p_2|^2\omega_c R')$
L'identification $R'^2C'_1C'_2 = 1/(|p_2|^2\omega_c^2)$ donne : $C'_1 = 1/(|p_2|^2\omega_c^2R'^2C'_2)$

Filtre n°3

On attribue une valeur arbitraire à R''
L'identification $2R''C''_2 = -2\mathrm{Re}(p_3)/(|p_3|^2\omega_c)$ donne : $C''_2 = -\mathrm{Re}(p_3)/(|p_3|^2\omega_c R'')$
L'identification $R''^2C''_1C''_2 = 1/(|p_3|^2\omega_c^2)$ donne : $C''_1 = 1/(|p_3|^2\omega_c^2R''^2C''_2)$

Amplificateur

$G = k/(|p_1|^2\,|p_2|^2\,|p_3|^2)$

Remarque

Dans le schéma final, les filtres seront ordonnés dans le sens des facteurs de qualité (coefficients de surtension) croissants, c'est-à-dire dans le sens des $Q = \frac{1}{2}(C_1/C_2)^{\frac{1}{2}}$ croissants.

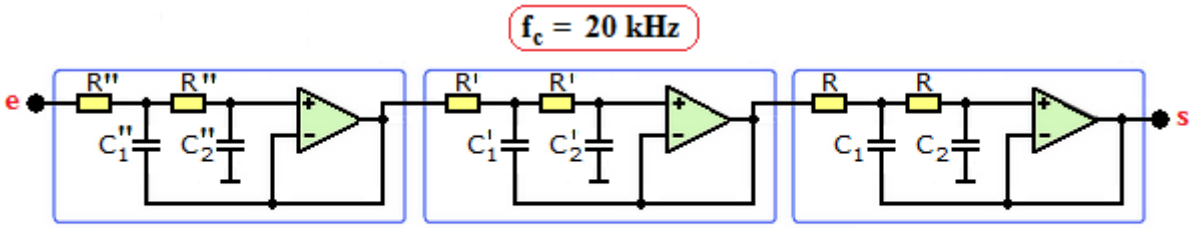
Mathématique & Informatique

Programme de calcul du filtre passe-bas 20 kHz :

```
//---------- Données
fc = 20D3 ;
A = 0.1 ;
//---------- Fonction de transfert normalisée
[h,pols,zers,k] = analpf(6,'cheb1',[1-10^(-A/20),0],1) ;
p1 = pols(1) ;
p2 = pols(2) ;
p3 = pols(3) ;
//---------- Transformation en filtre passe-bas
omegac = 2*%pi*fc ;
p = poly(0,'p') ;
N = k*omegac^6 ;
D1 = p^2 - 2*real(p1)*omegac*p + abs(p1)^2*omegac^2 ;
D2 = p^2 - 2*real(p2)*omegac*p + abs(p2)^2*omegac^2 ;
D3 = p^2 - 2*real(p3)*omegac*p + abs(p3)^2*omegac^2 ;
D = D1*D2*D3 ;
H = N/D ;
//---------- Diagramme de Bode
scf(2) ;
elatf = syslin('c',H(2),H(3)) ;
bode(elatf,500,100D3)
title('Filtre analogique passe-bas d''ordre 6 de Tchebychev      Fréquence de coupure :   20 kHz')
//---------- Calcul des résistances et des condensateurs
R = 1D3
C2 = -real(p1)/(abs(p1)^2*omegac*R)
C1 = 1/(abs(p1)^2*omegac^2*R^2*C2)
Rp = 1D3
Cp2 = -real(p2)/(abs(p2)^2*omegac*Rp)
Cp1 = 1/(abs(p2)^2*omegac^2*Rp^2*Cp2)
Rs = 1D3
Cs2 = -real(p3)/(abs(p3)^2*omegac*Rs)
Cs1 = 1/(abs(p3)^2*omegac^2*Rs^2*Cs2)
//---------- Gain
G = k/(abs(p1)^2*abs(p2)^2*abs(p3)^2)
```

Mathématique & Informatique

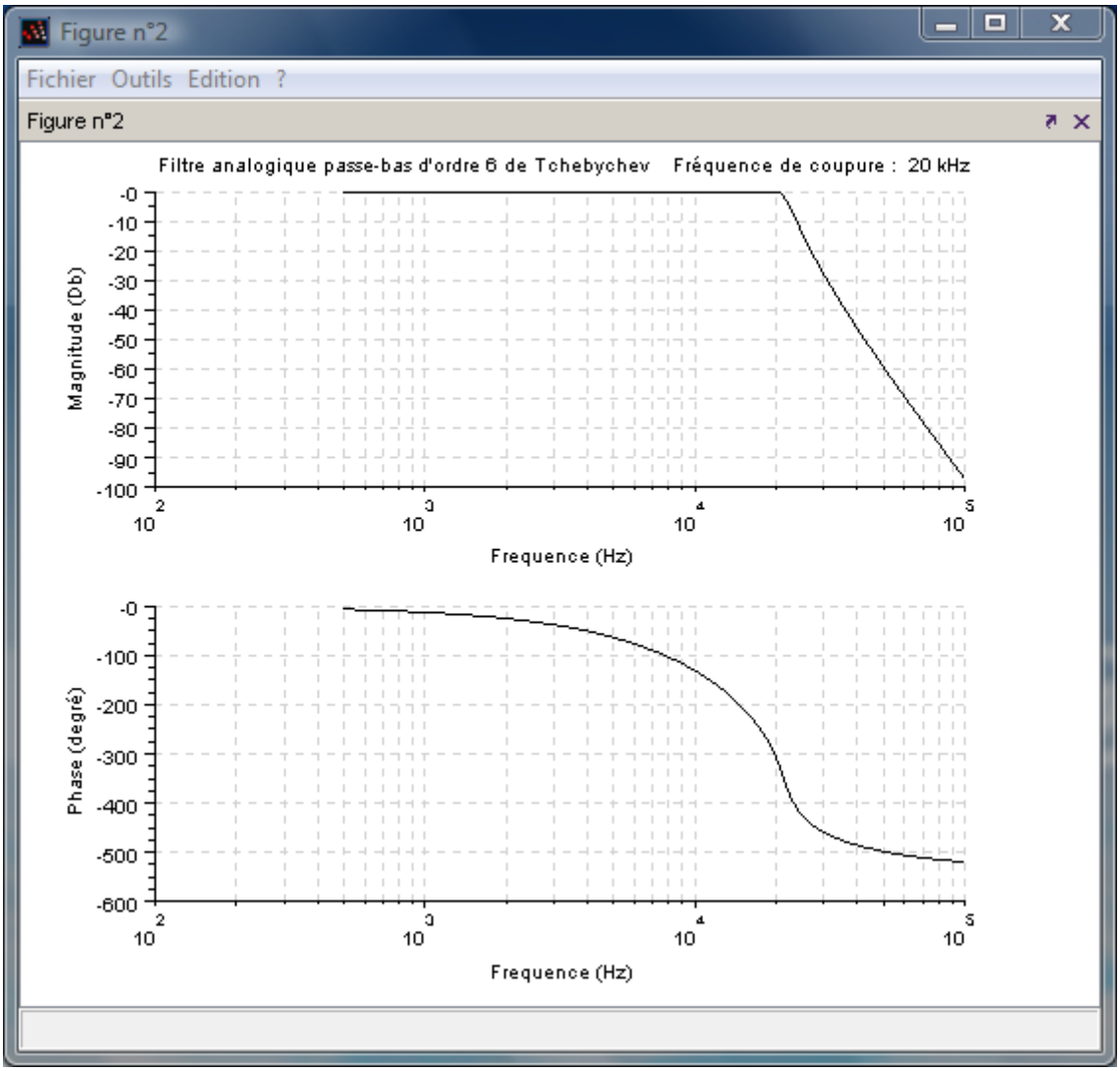
Résultats :



$R'' = 1\ k\Omega$
$C''_1 = 18.59\ nF$
$C''_2 = 12.93\ nF$

$R' = 1\ k\Omega$
$C'_1 = 25.40\ nF$
$C'_2 = 3.581\ nF$

$R = 1\ k\Omega$
$C_1 = 69.38\ nF$
$C_2 = 0.808\ nF$

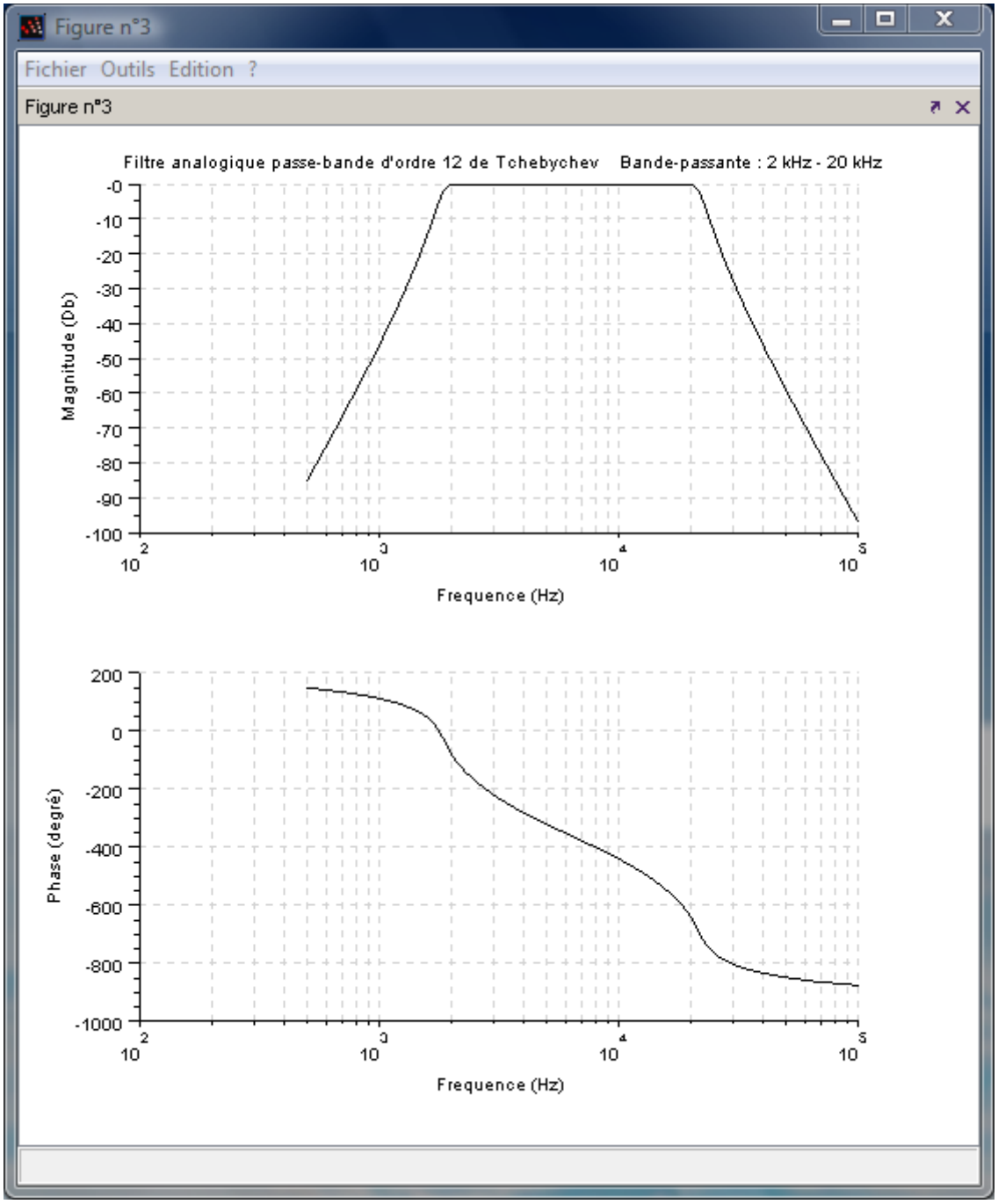


Note G = 0.9885531 ≈ 1

Mathématique & Informatique

Programme de calcul du filtre passe-bande 2 kHz - 20 kHz :

```
//---------- Données
fc1 = 2D3 ;
fc2 = 20D3 ;
A = 0.1 ;
//---------- Fonction de transfert normalisée
[h,pols,zers,k] = analpf(6,'cheb1',[1-10^(-A/20),0],1) ;
p1 = pols(1) ;
p2 = pols(2) ;
p3 = pols(3) ;
//---------- Transformation en filtre passe-haut
omegac1 = 2*%pi*fc1 ;
p = poly(0,'p') ;
N = k*p^6 ;
D1 = abs(p1)^2*p^2 - 2*real(p1)*omegac1*p + omegac1^2 ;
D2 = abs(p2)^2*p^2 - 2*real(p2)*omegac1*p + omegac1^2 ;
D3 = abs(p3)^2*p^2 - 2*real(p3)*omegac1*p + omegac1^2 ;
D = D1*D2*D3 ;
H1 = N/D ;
//---------- Transformation en filtre passe-bas
omegac2 = 2*%pi*fc2 ;
p = poly(0,'p') ;
N = k*omegac2^6 ;
D1 = p^2 - 2*real(p1)*omegac2*p + abs(p1)^2*omegac2^2 ;
D2 = p^2 - 2*real(p2)*omegac2*p + abs(p2)^2*omegac2^2 ;
D3 = p^2 - 2*real(p3)*omegac2*p + abs(p3)^2*omegac2^2 ;
D = D1*D2*D3 ;
H2 = N/D ;
//---------- Fonction de transfert du filtre passe-bande
H = H1*H2 ;
//---------- Diagramme de Bode
scf(3) ;
elatf = syslin('c',H(2),H(3)) ;
bode(elatf,500,100D3)
title('Filtre analogique passe-bande d''ordre 12 de Tchebychev      Bande-passante : 2 kHz - 20 kHz')
```

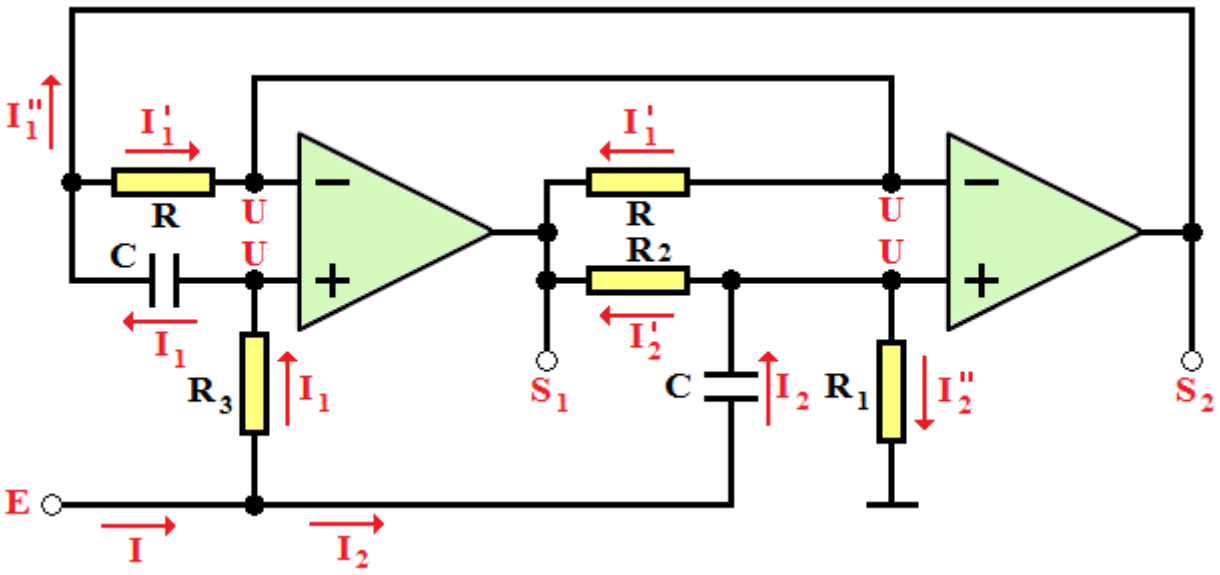

Figure n°3
Fichier Outils Edition ?
Figure n°3
Filtre analogique passe-bande d'ordre 12 de Tchebychev Bande-passante : 2 kHz - 20 kHz
Magnitude (Db)
-0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
Frequence (Hz)
Phase (degré)
200
0
-200
-400
-600
-800
-1000
Frequence (Hz)

Exemple 4

Réalisation d'un filtre passe-bas de **Cauer** de fréquence de coupure 1000 Hz.
Pour obtenir une forte pente du filtre à la fréquence de coupure, il faut accepter en contrepartie une variation du gain dans la bande passante. On construit un filtre avec les caractéristiques suivantes :

- type elliptique
- ordre 6
- atténuation dans la bande passante : 1 dB maximum
- atténuation dans la bande coupée : 36 dB minimum

Le filtre passe-bas d'ordre 6 est composé de 3 cellules d'ordre 2 mises en cascade.
Chaque cellule est un réjecteur de fréquence construit autour de 2 AOPs.
Précisons le schéma détaillé d'une cellule :



La fonction de transfert est $H = H_1$ ou $H = H_2$ suivant le cas. Elle s'exprime par :

$$H = \frac{p^2 + a_0}{p^2 + b_1 p + b_0} = \frac{p^2 + \omega_z^2}{p^2 + \frac{\omega_p}{Q_p} p + \omega_p^2}$$

où ω_z , ω_p , Q_p sont des fonctions des R_i et des C_i
On suppose $\omega_z \neq \omega_p$
Le paramètre Q_p est le facteur de qualité ou coefficient de surtension du filtre.

Note
On peut remarquer que le 1er AOP est un montage analogue à une self, et que le 2e AOP est un montage analogue à une capacité.

Calcul des fonctions de transfert H_1 et H_2

Le montage avec contre-réaction des 2 AOPs maintient leurs 4 entrées approximativement au même potentiel U.

Equations de branche

(1) $E - U = R_3 I_1$
(2) $U - S_2 = (1/Cp) I_1$
(3) $S_2 - U = R I_1'$
(4) $U - S_1 = R I_1'$
(5) $E - U = (1/Cp) I_2$
(6) $U - S_1 = R_2 I_2'$
(7) $U = R_1 I_2''$

Equations de nœud

(8) $I = I_1 + I_2$
(9) $I_1 = I_1' + I_1''$
(10) $I_2 = I_2' + I_2''$

Ces équations constituent un système linéaire de 10 équations à 10 inconnues I, I_1, I_1', I_1'', I_2, I_2', I_2'', U, S_1, S_2.
Il s'agit d'un système de **Cramer** que l'on peut résoudre par substitution ou par la méthode des déterminants.
On obtient alors les expressions de I, I_1, I_1', I_1'', I_2, I_2', I_2'', U, S_1, S_2 en fonction de R, C, R_1, R_2, R_3, E, p.
Dans le cas présent, les déterminants d'ordre 10 sont creux, c'est-à-dire qu'ils comportent de nombreux zéros, donc leur calcul est envisageable "à la main". Pour des circuits complexes, il est préférable de recourir à un logiciel de calcul formel.
Comme on cherche uniquement à exprimer S_1 et S_2 en fonction de R, C, R_1, R_2, R_3, E, p, simplifions le système des équations (1) à (10) en éliminant les inconnues I, I_1, I_1', I_1'', I_2, I_2', I_2'' et U. Dans un premier temps, ne gardons qu'un seul courant I_1 :

I_1' en fonction de I_1

(2) & (3) $(1/Cp) I_1 = - R I_1' \Rightarrow I_1' = - (1/RCp) I_1$

I_1'' en fonction de I_1

(9) $I_1'' = I_1 - I_1' \Rightarrow I_1'' = (1 + 1/RCp) I_1$

I_2 en fonction de I_1

(1) & (5) $R_3 I_1 = (1/Cp) I_2 \Rightarrow I_2 = R_3 Cp I_1$

I_2' en fonction de I_1

(4) & (6) $R I_1' = R_2 I_2' \Rightarrow I_2' = - (1/R_2 Cp) I_1$

I_2'' en fonction de I_1

(10) $I_2'' = I_2 - I_2' \Rightarrow I_2'' = (R_3 Cp + 1/R_2 Cp) I_1$

I en fonction de I_1

(8) $I = I_1 + I_2 \Rightarrow I = (1 + R_3 Cp) I_1$

Dans un deuxième temps, établissons une relation entre S_1 et I_1, puis entre S_2 et I_1 :

(4) & (7) $\quad S_1 = R_1 I_2'' - R I_1' = (R_1 R_3 Cp + R_1/R_2 Cp + 1/Cp) I_1$

(3) & (7) $\quad S_2 = R_1 I_2'' + R I_1' = (R_1 R_3 Cp + R_1/R_2 Cp - 1/Cp) I_1$

Il reste à formuler une équation en S_1 et E, et une équation en S_2 et E :

(1) & (4) $\quad E - S_1 = R_3 I_1 + R I_1' = (R_3 - 1/Cp) I_1 = [(R_3 - 1/Cp)/(R_1 R_3 Cp + R_1/R_2 Cp + 1/Cp)] S_1$

(1) & (3) $\quad E - S_2 = R_3 I_1 - R I_1' = (R_3 + 1/Cp) I_1 = [(R_3 + 1/Cp)/(R_1 R_3 Cp + R_1/R_2 Cp - 1/Cp)] S_2$

Après simplifications, on aboutit à :

$$H_1 = \frac{S_1}{E} = \frac{p^2 + \dfrac{R_1 + R_2}{R_1 R_2 R_3 C^2}}{p^2 + \dfrac{1}{R_1 C} p + \dfrac{1}{R_2 R_3 C^2}}$$

$$H_2 = \frac{S_2}{E} = \frac{p^2 + \dfrac{R_1 - R_2}{R_1 R_2 R_3 C^2}}{p^2 + \dfrac{1}{R_1 C} p + \dfrac{1}{R_2 R_3 C^2}}$$

Dans les 2 cas :

$$\omega_p^2 = \frac{1}{R_2 R_3 C^2}$$

$$Q_p = R_1 C \omega_p$$

Pour ω_z , distinguons :

- H_1
  $\omega_z^2 = (R_1 + R_2)/(R_1 R_2 R_3 C^2) = \omega_p^2 + (R_2/R_1)\omega_p^2$
  D'où :
  $$\omega_z = \omega_p \sqrt{1 + \frac{R_2}{R_1}}$$
- H_2
  $\omega_z^2 = (R_1 - R_2)/(R_1 R_2 R_3 C^2) = \omega_p^2 - (R_2/R_1)\omega_p^2$
  D'où :
  $$\omega_z = \omega_p \sqrt{1 - \frac{R_2}{R_1}}$$

<u>Conclusion</u>

- Si $\omega_z > \omega_p$, on prend $H = H_1$ pour pouvoir calculer R_2 en fonction de R_1 : $\quad R_2 = (\omega_z^2/\omega_p^2 - 1)R_1$
- Si $\omega_z < \omega_p$, on prend $H = H_2$ pour pouvoir calculer R_2 en fonction de R_1 : $\quad R_2 = (1 - \omega_z^2/\omega_p^2)R_1$
- Si $\omega_z = \omega_p$, alors $R_2 = 0$ et $\omega_p = \infty$. Le circuit de départ ne convient pas, on utilise un autre montage ...

Méthode de calcul

$$\omega_z = \sqrt{a_0}$$

$$\omega_p = \sqrt{b_0}$$

$$Q_p = \frac{\sqrt{b_0}}{b_1}$$

$$Choisir\ C\ et\ R$$

$$R_1 = \frac{Q_p}{C\omega_p}$$

$$si\ \ \omega_z > \omega_p\ \ alors\ \ H = H_1\ \ et\ \ R_2 = (\frac{\omega_z^2}{\omega_p^2} - 1)R_1$$

$$si\ \ \omega_z < \omega_p\ \ alors\ \ H = H_2\ \ et\ \ R_2 = (1 - \frac{\omega_z^2}{\omega_p^2})R_1$$

$$R_3 = \frac{1}{R_2 C^2 \omega_p^2}$$

Réglage du filtre

La fonction de transfert H du filtre dépend des 3 paramètres ω_z , ω_p , Q_p . On ajuste :

- ω_z avec R_2
- ω_p avec R_3
- Q_p avec R_1

Le programme Scilab :

```
funcprot(0) ; lines(0)

//---------- Données
fc = 1000 ;
n = 6 ;
A1 = 1 ;
A2 = 36 ;

//---------- Pulsation de coupure
omegac = 2*%pi*fc ;
//---------- Fonction de transfert
[h,pols,zers,g] = analpf(n,'ellip',[1-10^(-A1/20),10^(-A2/20)],omegac) ;
//---------- Courbe de réponse
scf(1) ;
fr = 0:3*omegac ;
hf = freq(h(2),h(3),%i*fr) ;
hm = abs(hf) ;
f = gcf() ;
f.figure_name = 'Filtre analogique passe-bas d''ordre ' + string(n)' + ' de Cauer' ...
                                + '      Pulsation de coupure :   ' + string(omegac) + ' rad/s' ;
a = gca() ;
a.axes_visible = "on" ; a.box="off" ;
a.x_location = "origin" ; a.y_location = "left" ;
a.x_label.text = "Pulsation (rad/s)" ; a.y_label.text = "Gain" ;
a.x_label.font_foreground = 5 ; a.y_label.font_foreground = 5 ;
a.x_label.font_size = 2 ; a.y_label.font_size = 2 ;
plot2d(fr,hm,2)

//---------- Filtre passe-bas normalisé : omegac = 1
[h,pols,zers,g] = analpf(n,'ellip',[1-10^(-A1/20),10^(-A2/20)],1) ;
//---------- Transformation en filtre passe-bas spécifique : omegac = 2*pi*fc
p = poly(0,'p') ;
m = length(zers) ;
n = length(pols) ;
N = 1 ;
for k = 1:m
   N = N*(p - omegac*zers(k)) ;
end
D = 1 ;
for k = 1:n
   D = D*(p - omegac*pols(k)) ;
end
H = g*omegac^(n-m)*real(N)/real(D) ;
//---------- Diagramme de Bode
scf(2) ;
elatf = syslin('c',H(2),H(3)) ;
bode(elatf,1D2,1D4)
title('Filtre analogique passe-bas d''ordre ' + string(n)' + ' de Cauer' ...
          + '      Fréquence de coupure :   ' + string(fc) + ' Hz')
```

```
//---------- Filtres réjecteurs à 2 AOPs
for i = 1:n/2
  a0 = omegac^2*abs(zers(i))^2 ;
  b0 = omegac^2*abs(pols(i))^2 ;
  b1 = - 2*omegac*real(pols(i)) ;
  omegaz = sqrt(a0) ;
  omegap = sqrt(b0) ;
  Qp = sqrt(b0)/b1 ;
  //----- Choix de la capacité et de la résistance
  C = 10D-9 ; R = 10D3 ;
  //----- Calcul des résistances
  R1 = 1/(C*b1) ;
  if a0 > b0 then
    filtre = 'H = H1' ;
    R2 = (a0/b0-1)*R1 ;
  else
    filtre = 'H = H2' ;
    R2 = (1-a0/b0)*R1 ;
  end
  R3 = 1/(R2*C^2*b0) ;
  //----- Résultats
  printf('\n=====================\n')
  filtre
  omegaz, omegap, Qp
  C
  R, R1, R2, R3
end
```

La valeur des composants :

Filtre réjecteur numéro 1

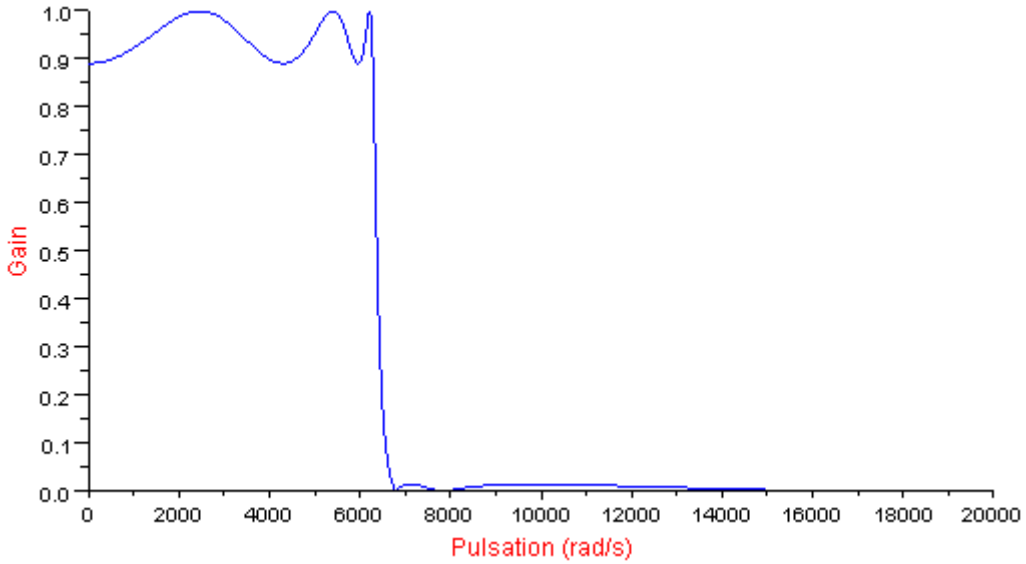
H = H1
omegaz = 17208.488 ; omegap = 3438.0605 ; Qp = 0.8332862
C = 1.000D-08
R = 10000 ; R1 = 24237.1 ; R2 = 582973.11 ; R3 = 1451.1906

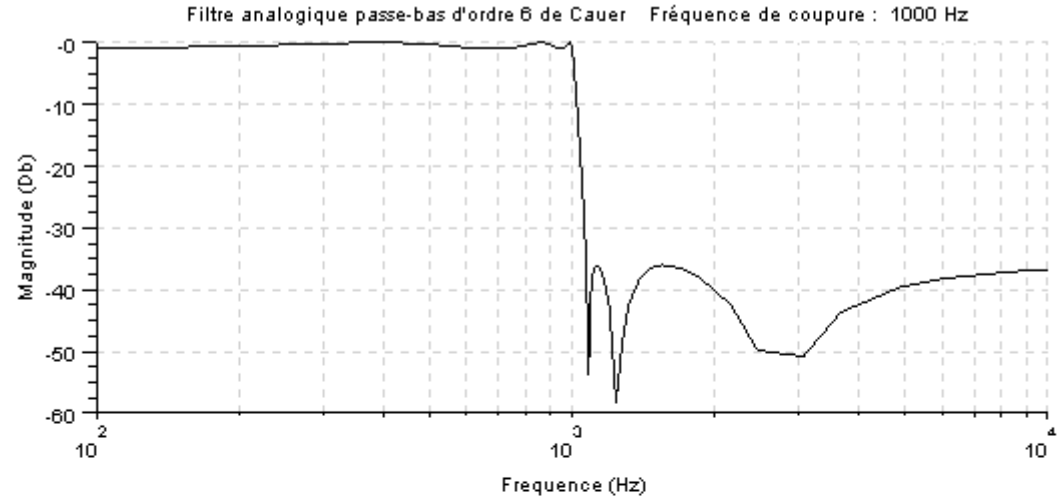
Filtre réjecteur numéro 2

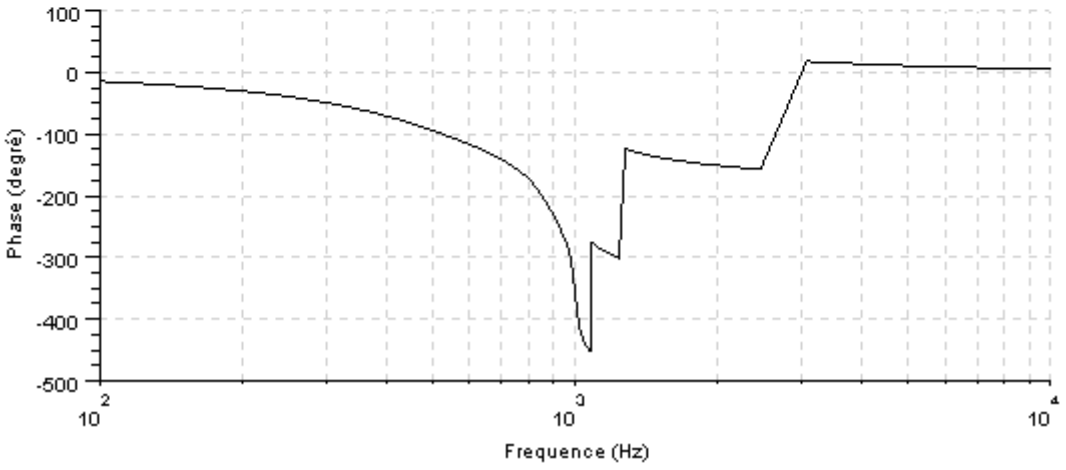
H = H1
omegaz = 7830.2192 ; omegap = 5656.544 ; Qp = 4.1357667
C = 1.000D-08
R = 10000 ; R1 = 73114.727 ; R2 = 66989.22 ; R3 = 4665.4415

Filtre réjecteur numéro 3

H = H1
omegaz = 6800.3689 ; omegap = 6283.2509 ; Qp = 25.289027
C = 1.000D-08
R = 10000 ; R1 = 402483.17 ; R2 = 68975.755 ; R3 = 3672.271

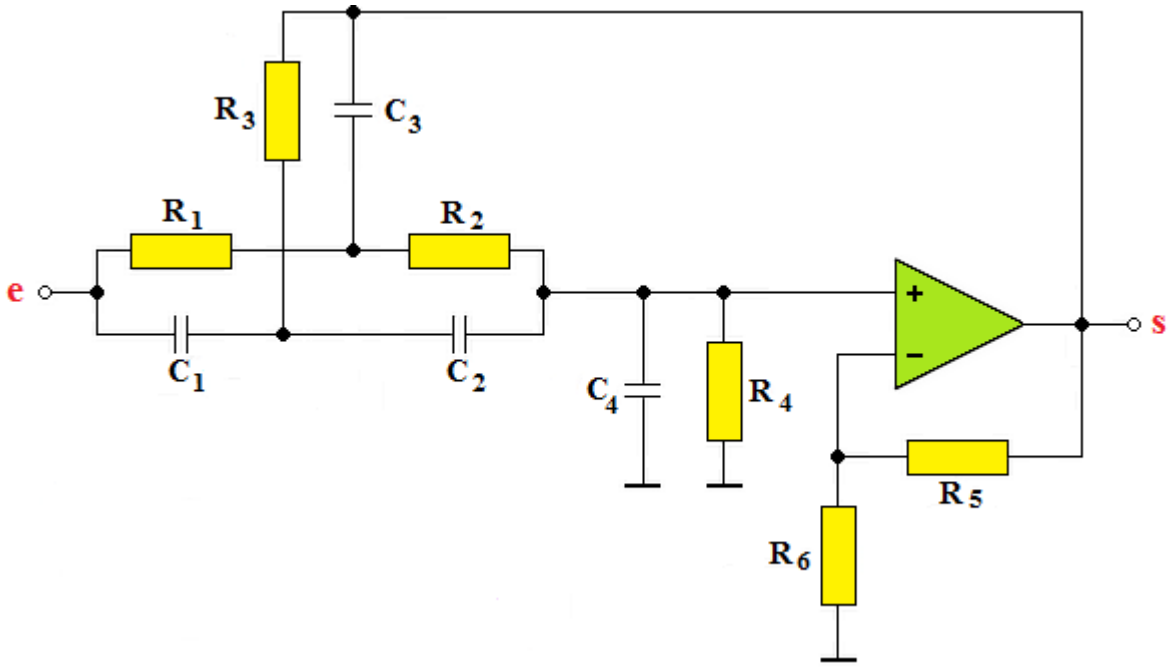

Gain
Pulsation (rad/s)


Filtre analogique passe-bas d'ordre 6 de Cauer Fréquence de coupure : 1000 Hz
Magnitude (Db)
Frequence (Hz)


Phase (degré)
Frequence (Hz)

Autre solution

Le filtre réjecteur peut être réalisé avec un circuit ne comportant qu'un seul AOP associé à un pont en double T :



La fonctionde transferts'écrit : $H = K\dfrac{p^2 + a_0}{p^2 + b_1 p + b_0} = K\dfrac{p^2 + \omega_z^2}{p^2 + \dfrac{\omega_p}{Q_p}p + \omega_p^2}$

où K, ω_z , ω_p , Q_p sont des fonctions des R_i et des C_i
Le gain en continu (p = 0) est $G = 20\log_{10}(K\omega_z^2/\omega_p^2)$ décibels.
Les formules de ce filtre réjecteur :

$$R_S = R_1 + R_2$$

$$C_S = \frac{C_1 C_2}{C_1 + C_2}$$

$$K = \frac{1 + R_5/R_6}{1 + C_4/C_S}$$

$$\omega_z^2 = \frac{1}{R_1 R_2 C_3 C_S} = \frac{1}{R_3 R_S C_1 C_2}$$

$$\omega_p = \omega_z \sqrt{(1 + R_S/R_4)(1 + C_4/C_S)}$$

$$k = \frac{1}{2\sqrt{(1 + C_2/C_1)(1 + C_2/C_3)}}$$

$$Q_p = \left[\frac{k(1 + C_4/C_S)}{k R_S C_S \omega_z^2 - (R_5/R_6)\omega_z + k/(R_4 C_S)}\right]\omega_p$$

Le programme Scilab de calcul :

```
//---------- Filtres réjecteurs à 1 AOP
for i = 1:n/2
  a0 = omegac^2*abs(zers(i))^2 ;
  b0 = omegac^2*abs(pols(i))^2 ;
  b1 = - 2*omegac*real(pols(i)) ;
  omegaz = sqrt(a0) ;
  omegap = sqrt(b0) ;
  Qp = sqrt(b0)/b1 ;
  //----- Choix des capacités
  C1 = 10D-9 ;
  C2 = 10D-9 ;
  C3 = 10D-9 ;
  //----- Choix des résistances
  R6 = 10D3 ;
  //----- Calcul des capacités
  CS = (C1*C2)/(C1+C2) ;
  C4 = ceil(a0/b0)*CS ;
  //----- Calcul des résistances
  k = 1/(2*sqrt((1+C2/C1)*(1+C2/C3))) ;
  R1 = 1/(2*k*(C2+C3)*sqrt(a0)) ;
  R2 = (1+C2/C1)/(R1*C2*C3*a0) ;
  RS = R1+R2 ;
  R3 = 1/(RS*C2*C3*a0) ;
  R4 = RS/((1+C4/CS)*(b0/a0)-1) ;
  R5 = k*(1/(R4*CS*sqrt(a0))+RS*C4*sqrt(a0)-(1+C4/CS)*b1/sqrt(a0))*R6 ;
  //----- Calcul du gain
  K = (1+R5/R6)/(1+C4/CS) ;
  //----- Résultats
  printf('\n====================\n')
  omegaz, omegap, Qp
  C1, C2, C3, C4
  R1, R2, R3, R4, R5, R6
  K
end
```

La valeur des composants :

Filtre réjecteur numéro 1

omegaz = 17208.488 ; omegap = 3438.0605 ; Qp = 0.8332862
C1 = 1.000D-08 ; C2 = 1.000D-08 ; C3 = 1.000D-08 ; C4 = 1.000D-07 ;
R1 = 5811.0858 ; R2 = 11622.172 ; R3 = 1937.0286 ; R4 = 224312.69 ; R5 = 81445.735 ; R6 = 10000.
K = 0.3386879

Filtre réjecteur numéro 2

omegaz = 7830.2192 ; omegap = 5656.544 ; Qp = 4.1357667
C1 = 1.000D-08 ; C2 = 1.000D-08 ; C3 = 1.000D-08 ; C4 = 1.000D-08 ;
R1 = 12771.035 ; R2 = 25542.069 ; R3 = 4257.0115 ; R4 = 67741.055 ; R5 = 7132.6025 ; R6 = 10000.
K = 0.5710867

Filtre réjecteur numéro 3

omegaz = 6800.3689 ; omegap = 6283.2509 ; Qp = 25.289027
C1 = 1.000D-08 ; C2 = 1.000D-08 ; C3 = 1.000D-08 ; C4 = 1.000D-08 ;
R1 = 14705.085 ; R2 = 29410.169 ; R3 = 4901.6949 ; R4 = 28259.234 ; R5 = 9827.8001 ; R6 = 10000.
K = 0.6609267

Réglage d'un filtre :

La fonction de transfert H du filtre dépend des 3 paramètres ω_z , ω_p , Q_p . On ajuste :

- ω_z avec R_1, R_2, R_3
- ω_p avec R_4
- Q_p avec R_5

Exemple 5

Calculer la fonction de transfert d'un filtre passe-bande d'ordre 8 possédant les caractéristiques suivantes :

- fréquence inférieure de coupure : 1000 Hz
- fréquence supérieure de coupure : 2000 Hz
- atténuation maximale dans la bande passante : 1 dB
- atténuation minimale dans la bande coupée : 40 dB

On utilisera les approximations de **Butterworth**, de **Tchebychev**, de **Cauer**.
Représenter le diagramme de **Bode** (gain, déphasage) dans l'intervalle 500 Hz – 5000 Hz pour chaque type de filtre.

Programme :

```
funcprot(0) ; lines(0)

//-----   Données
f1 = 1000 ;
f2 = 2000 ;
fmin = 500 ;
fmax = 5000 ;
r1 = 1-10^(-1/20) ;
r2 = 10^(-40/20) ;

function H = transformation(pols,zers,gain)
  omega1 = 2*%pi*f1
  omega2 = 2*%pi*f2
  B = omega2 - omega1
  omega0 = omega2 * omega1
  s = poly(0,'s')
  m = length(zers)
  num = 1
  for k = 1:m
    num = num*(s^2 - zers(k)*B*s + omega0)
  end
  n = length(pols)
  den = 1
  for k = 1:n
    den = den*(s^2 - pols(k)*B*s + omega0)
  end
  K = gain*B^(n-m)
  N = real(K*num*s^(n-m))
  D = real(den)
  H = N / D
endfunction

function graphique(i,text,H)
  scf(i)
  elatf = syslin('c',H(2),H(3))
  bode(elatf,fmin,fmax)
  title(text)
endfunction
```

Mathématique & Informatique

```
//-----  Passe-bande de Butterworth
[h1,pols1,zers1,gain1] = analpf(4,'butt',[0,0],1) ;
H1 = transformation(pols1,zers1,gain1)
graphique(1,'Filtre analogique passe-bande d''ordre 8 de Butterworth',H1)

//-----  Passe-bande de Tchebychev 1
[h2,pols2,zers2,gain2] = analpf(4,'cheb1',[r1,0],1) ;
H2 = transformation(pols2,zers2,gain2)
graphique(2,'Filtre analogique passe-bande d''ordre 8 de Tchebychev 1',H2)

//-----  Passe-bande de Tchebychev 2
[h3,pols3,zers3,gain3] = analpf(4,'cheb2',[0,r2],1) ;
H3 = transformation(pols3,zers3,gain3)
graphique(3,'Filtre analogique passe-bande d''ordre 8 de Tchebychev 2',H3)

//-----  Passe-bande de Cauer
[h4,pols4,zers4,gain4] = analpf(4,'ellip',[r1,r2],1) ;
H4 = transformation(pols4,zers4,gain4)
graphique(4,'Filtre analogique passe-bande d''ordre 8 de Cauer (elliptique)',H4)
```

Résultats :

Filtre passe-bande de Butterworth

$$H1=\frac{1.559D{+}15s^4}{3.887D{+}31+8.082D{+}27s+2.809D{+}24s^2+3.583D{+}20s^3+6.025D{+}16s^4+4.537D{+}12s^5+4.506D{+}08s^6+16418.754s^7+s^8}$$

Filtre passe-bande de Tchebychev 1

$$H2=\frac{3.829D{+}14s^4}{3.887D{+}31+2.947D{+}27s+2.327D{+}24s^2+1.265D{+}20s^3+4.690D{+}16s^4+1.602D{+}12s^5+3.732D{+}08s^6+5986.6905s^7+s^8}$$

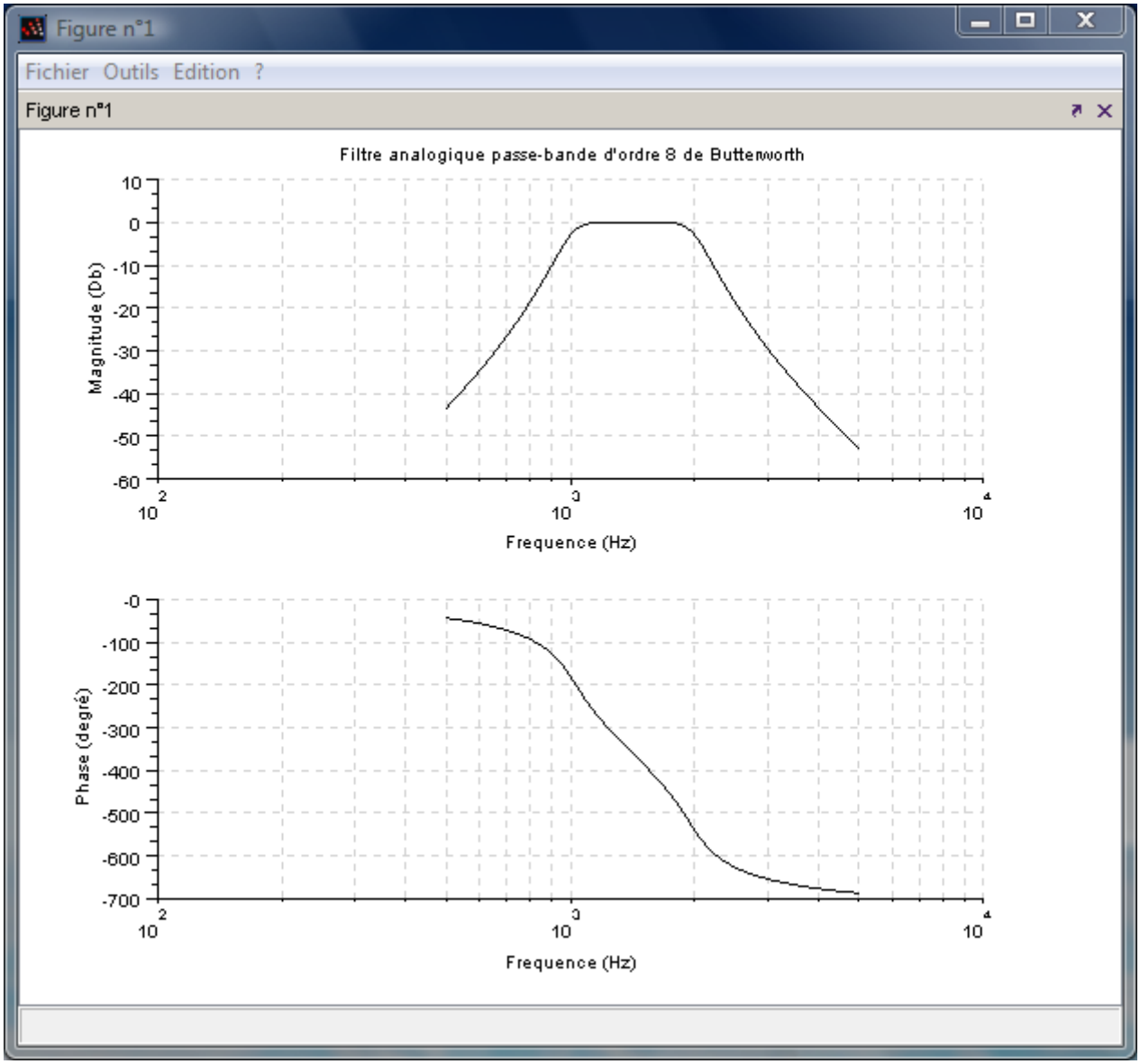
Filtre passe-bande de Tchebychev 2

$$H3=\frac{3.887D{+}29+3.938D{+}22s^2+9.975D{+}14s^4+6316546.8s^6+0.01s^8}{3.887D{+}31+4.180D{+}27s+2.194D{+}24s^2+1.660D{+}20s^3+4.323D{+}16s^4+2.102D{+}12s^5+3.519D{+}08s^6+8491.0609s^7+s^8}$$

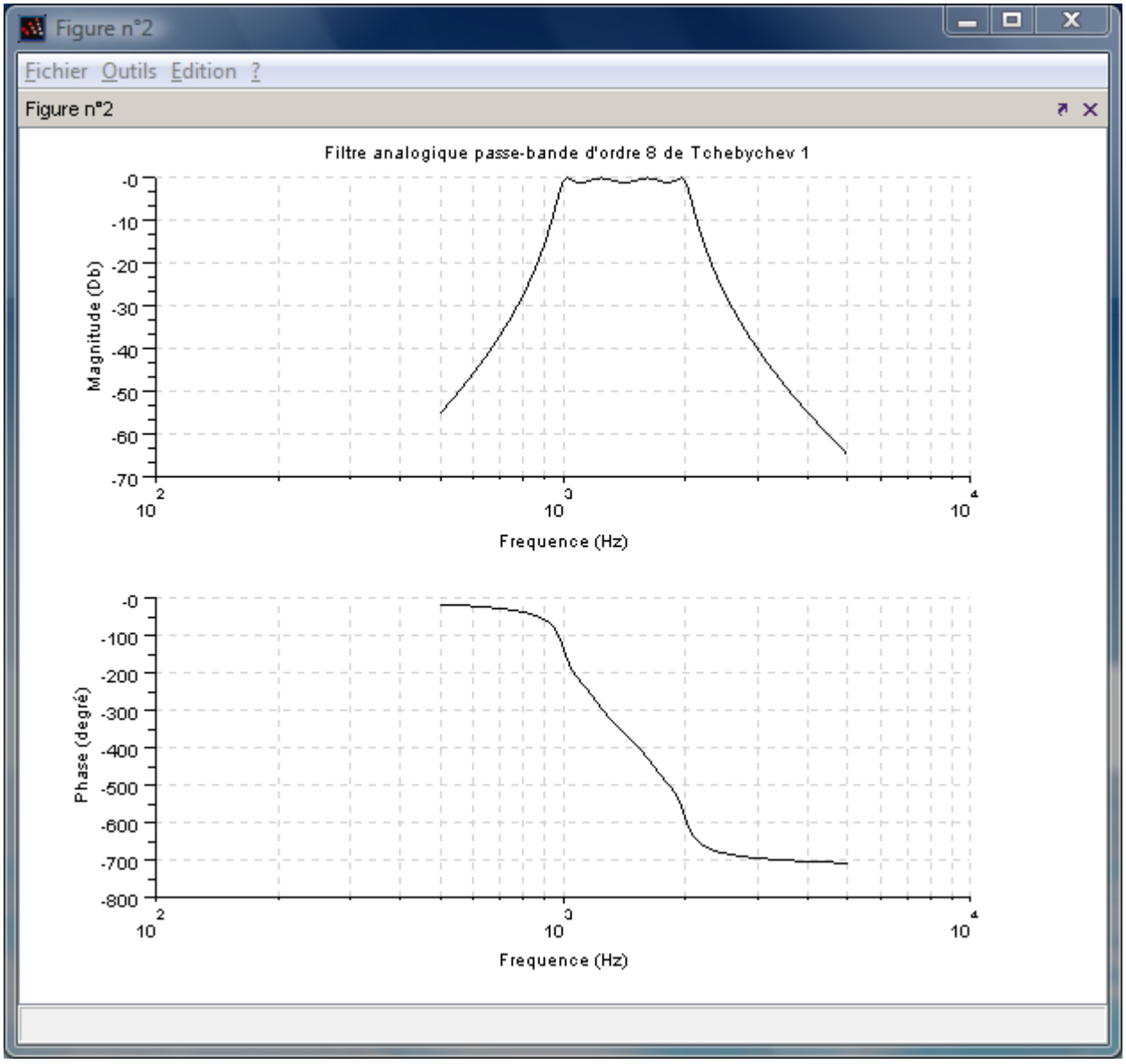
Filtre passe-bande de Cauer (elliptique)

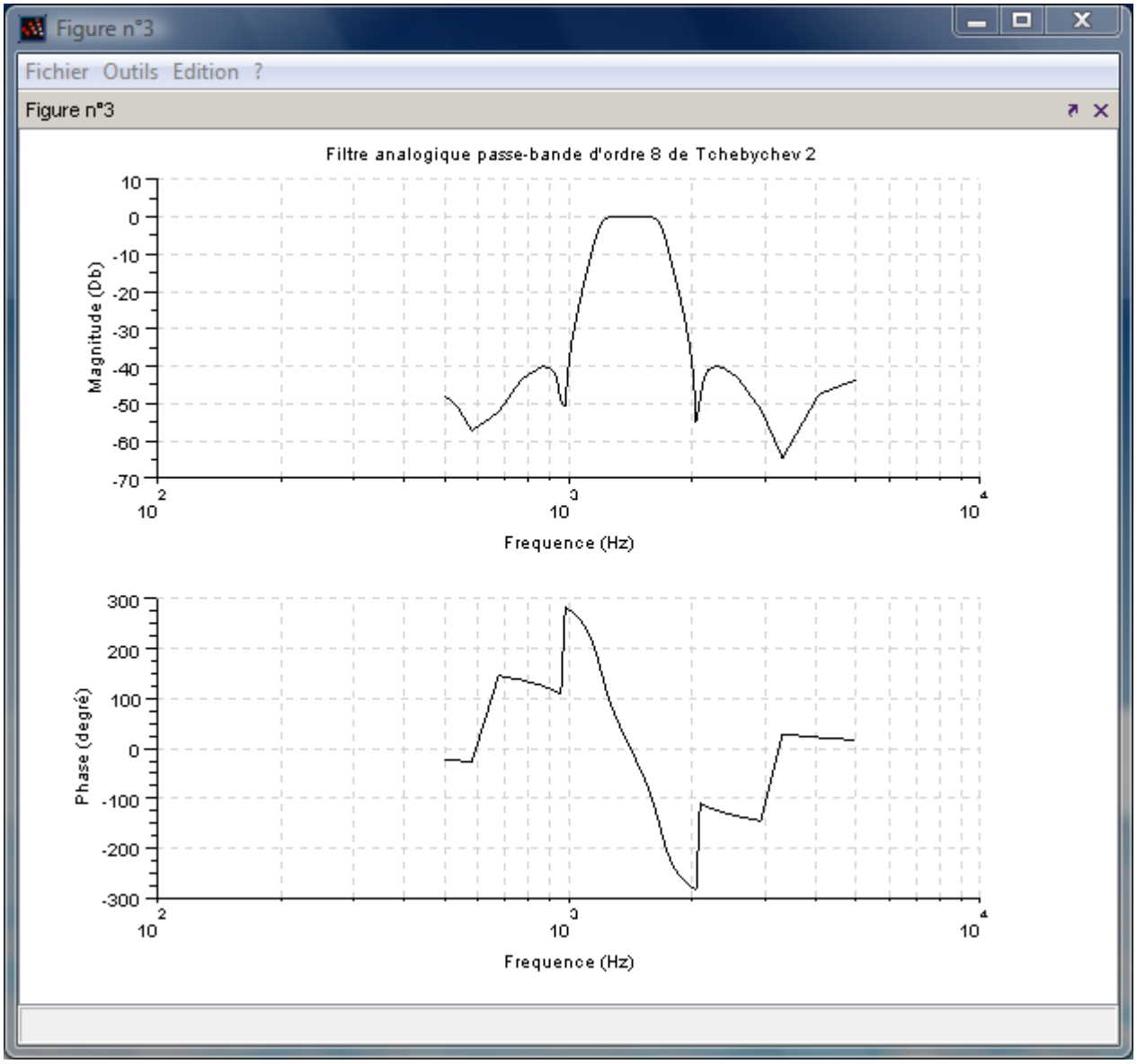
$$H4=\frac{3.887D{+}29+5.665D{+}22s^2+1.812D{+}15s^4+9087262.2s^6+0.01s^8}{3.887D{+}31+2.905D{+}27s+2.341D{+}24s^2+1.261D{+}20s^3+4.740D{+}16s^4+1.597D{+}12s^5+3.756D{+}08s^6+5900.814s^7+s^8}$$

Mathématique & Informatique

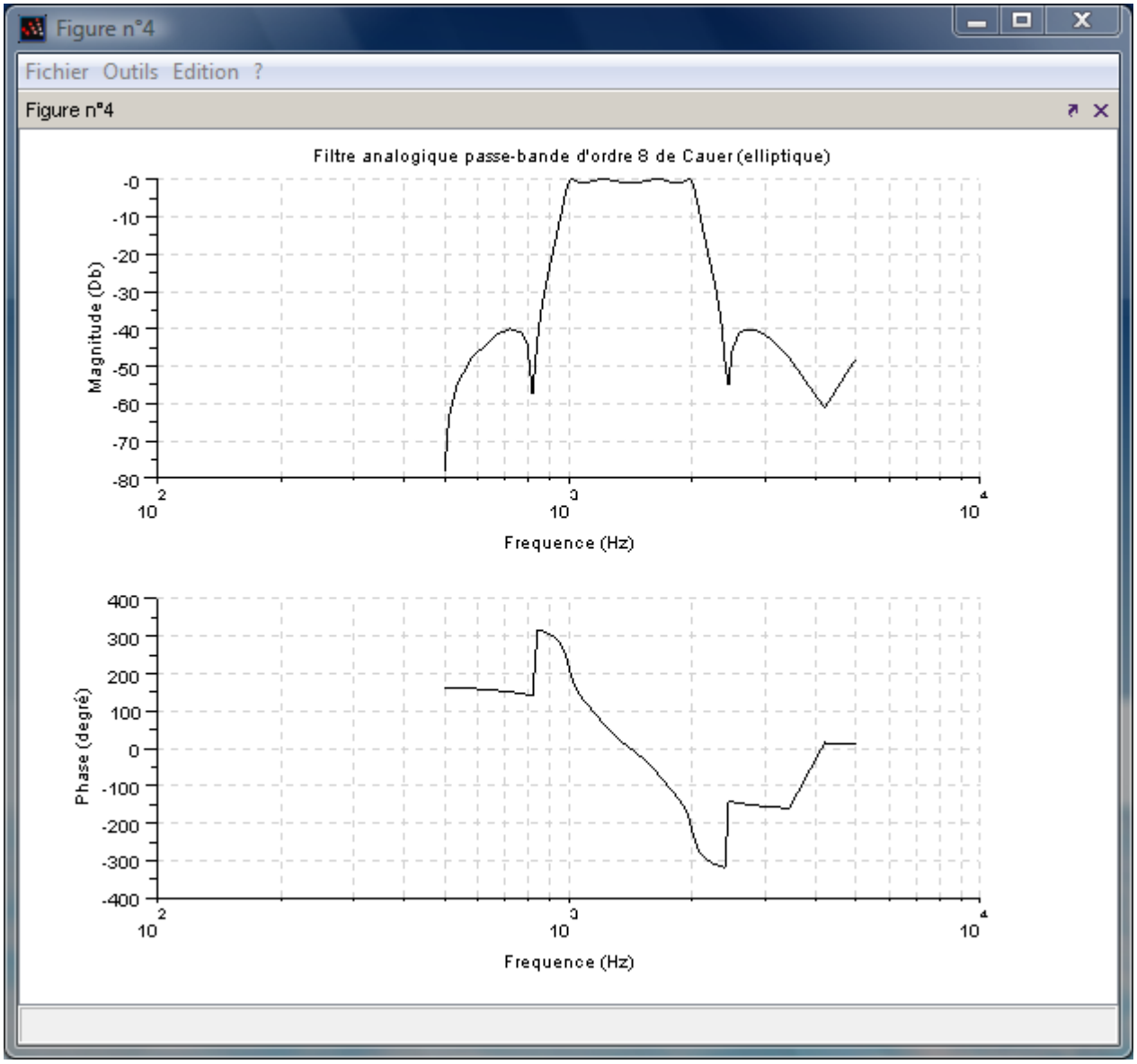


Mathématique & Informatique



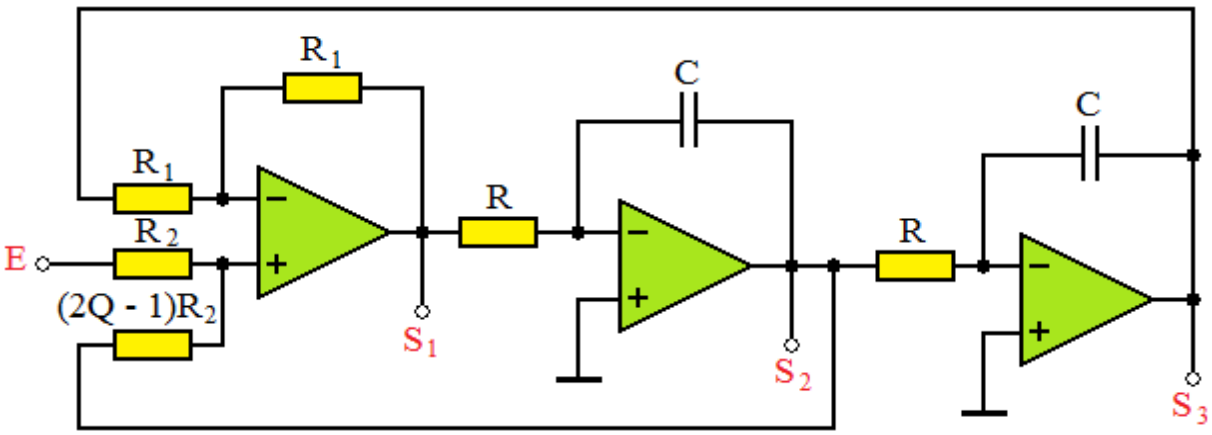

Figure n°3
Fichier Outils Edition ?
Figure n°3
Filtre analogique passe-bande d'ordre 8 de Tchebychev 2
Magnitude (Db)
10
0
-10
-20
-30
-40
-50
-60
-70
10^2
10^3
10^4
Frequence (Hz)
Phase (degré)
300
200
100
0
-100
-200
-300
10^2
10^3
10^4
Frequence (Hz)

Mathématique & Informatique

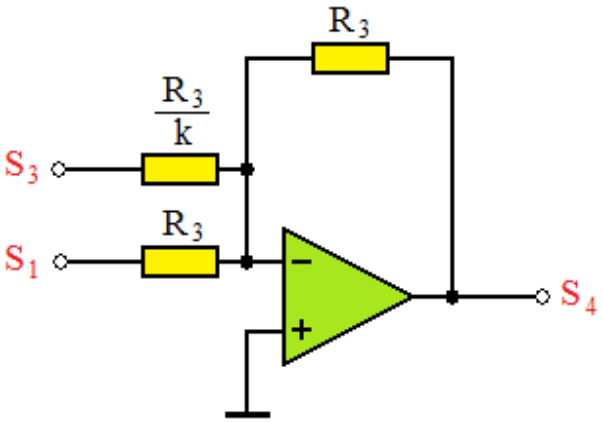


Filtres à capacités commutées

La cellule biquadratique KHN (**Kerwin - Huelsman - Newcomb**) est composée d'un sommateur et de 2 intégrateurs :



Les 3 sorties S_1 , S_2 , S_3 des 3 AOPs correspondent respectivement à un filtre passe-haut, à un filtre passe-bande, et à un filtre passe-bas. A l'aide d'un 4^e AOP monté en sommateur on obtient un filtre coupe-bande :



Chaque AOP fournit une équation :

$$\text{AOP}_1 \text{ (sommateur)} : \quad S_1 = \left(\frac{2Q-1}{Q}\right)E + \left(\frac{1}{Q}\right)S_2 - S_3$$

$$\text{AOP}_2 \text{ (intégrateur)} : \quad S_2 = -\left(\frac{1}{RCp}\right)S_1$$

$$\text{AOP}_3 \text{ (intégrateur)} : \quad S_3 = -\left(\frac{1}{RCp}\right)S_2$$

$$\text{AOP}_4 \text{ (sommateur)} : \quad S_4 = -S_1 - k\,S_3$$

Ces 4 équations à 4 inconnues S_1 , S_2 , S_3 , S_4 constituent un système de **Cramer** qui admet une solution unique.
Exprimons la première équation en fonction de S_1 :

$$S_1 = \left(\frac{2Q-1}{Q}\right)E - \left(\frac{1}{QRCp}\right)S_1 - \left(\frac{1}{R^2C^2p^2}\right)S_1$$

On obtient :

$$S_1 = \left[\frac{2Q-1}{Q} \times \frac{p^2}{p^2 + \frac{1}{QRC}p + \frac{1}{R^2C^2}}\right]E$$

On déduit successivement S_2 , S_3 et S_4
Posons :

$$\omega_c = \frac{1}{RC} \qquad \text{(pulsation de coupure)}$$

$$K = 2 - \frac{1}{Q} \qquad \text{(gain)}$$

Regroupons dans un tableau les différentes fonctions de transfert :

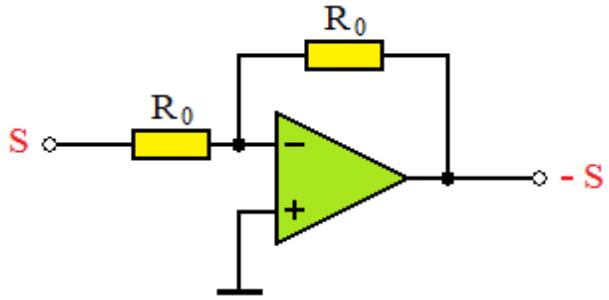
La cellule biquadratique KHN	
Filtre passe-haut	$H_1 = \frac{S_1}{E} = K \times \frac{p^2}{p^2 + \frac{\omega_c}{Q}p + \omega_c^2}$
Filtre passe-bande	$H_2 = \frac{S_2}{E} = -K\omega_c \times \frac{p}{p^2 + \frac{\omega_c}{Q}p + \omega_c^2}$
Filtre passe-bas	$H_3 = \frac{S_3}{E} = K\omega_c^2 \times \frac{1}{p^2 + \frac{\omega_c}{Q}p + \omega_c^2}$
Filtre coupe-bande	$H_4 = \frac{S_4}{E} = -K \times \frac{p^2 + k\omega_c^2}{p^2 + \frac{\omega_c}{Q}p + \omega_c^2}$

Remarques

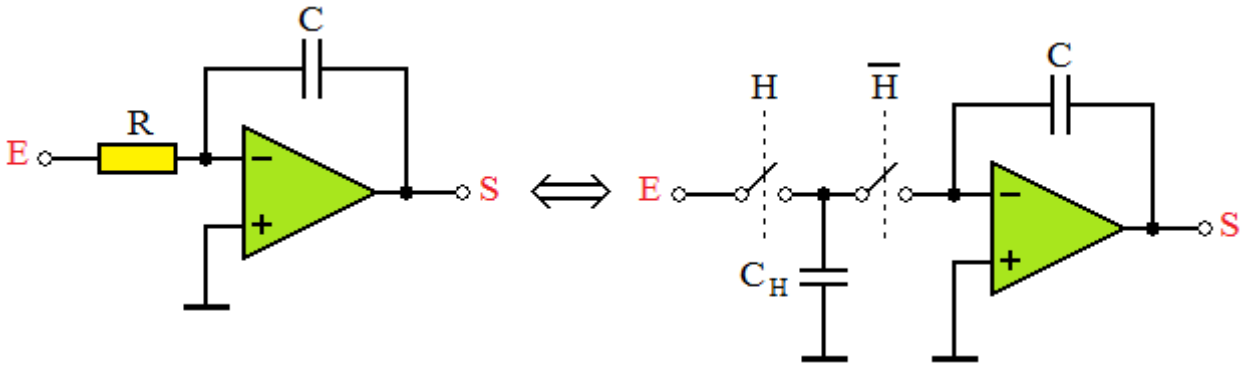
- Les résistances R_1 , R_2 , R_3 sont quelconques. On prendra des valeurs standard.
- Le facteur de qualité Q doit être supérieur à $\frac{1}{2}$
- Les réglages des paramètres de la cellule s'effectuent de la manière suivante :

  ω_c avec R et C
  Q avec $R_Q = (2Q - 1)R_2$
  k avec $R_k = R_3/k$

- Les sorties S_2 et S_4 des filtres passe-bande et coupe-bande sont en opposition de phase par rapport aux sorties S_1 et S_3 des filtres passe-haut et passe-bas.
  On peut inverser une sortie avec un AOP monté en amplificateur de gain -1 :



Pour que la cellule KHN fonctionne effectivement à la fréquence de coupure souhaitée, les valeurs des composants utilisés doivent être proches des valeurs théoriques.
Une résistance fixe est généralement fabriquée avec une tolérance de quelques pour cent et sa valeur varie avec la température, c'est pourquoi on remplace chaque intégrateur "classique" par un intégrateur à capacité commutée.
Le principe est le suivant :



La résistance R est remplacée par la capacité commutée C_H . On produit un signal carré d'horloge H de fréquence F_H et son inverse $\overline{H}$ pour commander 2 interrupteurs analogiques. On montre que les deux montages sont équivalents si :

$$RC_HF_H = 1$$

La fréquence F_H doit être supérieure au double de la plus haute fréquence F_E contenue dans le signal d'entrée E :

$$F_H > 2F_E$$ (Théorème de **Nyquist – Shannon**)

L'intérêt de cette transformation est de pouvoir simuler avec précision une résistance variable R en produisant des signaux carrés de fréquence stable. De plus, les 2 capacités C_H et C peuvent être implantées dans un circuit intégré.
En conclusion, la cellule KHN à capacités commutées permet de réaliser un filtre universel d'ordre 2 dont on peut régler la fréquence de coupure avec un signal d'horloge. Il existe des circuits intégrés spécialisés qui regroupent un
à plusieurs filtres à capacités commutées. Ils fonctionnent pour des fréquences maximales inférieures au Mégahertz.
Exemple : le MAX7418 est un filtre passe-bas elliptique d'ordre 5 à capacités commutées (1 Khz < f_c < 30 Khz)

Exemple 1

Utilisons la cellule KHN à capacités commutées pour réaliser un filtre passe-bas aux caractéristiques suivantes :

- fréquence de coupure : 1000 Hz
- type elliptique
- ordre 10
- atténuation dans la bande passante : 0.1 dB maximum
- atténuation dans la bande coupée : 40 dB minimum

Le programme est similaire à celui de l'exemple 4 précédent, seul le calcul des composants diffère :

```
//---------- Données
fc = 1000 ;
n = 10 ;
A1 = 0.1 ;
A2 = 40 ;

//---------- Pulsation de coupure
omegac = 2*%pi*fc ;
//---------- Fonction de transfert
[h,pols,zers,g] = analpf(n,'ellip',[1-10^(-A1/20),10^(-A2/20)],omegac) ;
//---------- Courbe de réponse
scf(1) ;
fr = 0:3*omegac ;
hf = freq(h(2),h(3),%i*fr) ;
hm = abs(hf) ;
f = gcf() ;
f.figure_name = 'Filtre analogique passe-bas d''ordre ' + string(n)' + ' de Cauer' ...
                                + '      Pulsation de coupure :   ' + string(omegac) + ' rad/s' ;
a = gca() ;
a.axes_visible = "on" ; a.box="off" ;
a.x_location = "origin" ; a.y_location = "left" ;
a.x_label.text = "Pulsation (rad/s)" ; a.y_label.text = "Gain" ;
a.x_label.font_foreground = 5 ; a.y_label.font_foreground = 5 ;
a.x_label.font_size = 2 ; a.y_label.font_size = 2 ;
plot2d(fr,hm,2)

//---------- Transformation en filtre passe-bas de fréquence de coupure omegac = 2*pi*fc
p = poly(0,'p') ;
m = length(zers) ;
n = length(pols) ;
N = 1 ;
for k = 1:m
   N = N*(p - omegac*zers(k)) ;
end
D = 1 ;
for k = 1:n
   D = D*(p - omegac*pols(k)) ;
end
H = g*omegac^(n-m)*real(N)/real(D) ;
//---------- Diagramme de Bode
scf(2) ;
elatf = syslin('c',H(2),H(3)) ;
bode(elatf,1D2,1D4)
title('Filtre analogique passe-bas d''ordre ' + string(n)' + ' de Cauer' ...
          + '      Fréquence de coupure :   ' + string(fc) + ' Hz')
```

```
//---------- Filtres réjecteurs KHN
for i = 1:n/2
  a0 = omegac^2*abs(zers(i))^2 ;
  b0 = omegac^2*abs(pols(i))^2 ;
  b1 = - 2*omegac*real(pols(i)) ;
  omegaz = sqrt(a0) ;
  omegap = sqrt(b0) ;
  Qp = sqrt(b0)/b1 ;
  //----- Choix des capacités et des résistances
  C = 1 ; CH = 1 ; R1 = 1 ; R2 = 1 ; R3 = 1 ;
  //----- Calcul des résistances
  RQ = (2*Qp - 1)*R2 ;
  Rk = R3/(omegaz/omegap)^2 ;
  //----- Fréquence d'horloge
  FH = (C/CH)*omegap ;
  //----- Résultats
  printf('\n==========   Filtre réjecteur numéro %i\n',i)
  omegaz, omegap, Qp
  C, CH
  R1, R2, R3, RQ, Rk
  FH
end
```

NB

Dans le programme on prend $C = C_H = R_1 = R_2 = R_3 = 1$, ce qui signifie que les valeurs obtenues pour R_Q , R_k et F_H peuvent être multipliées par une constante λ à condition de multiplier aussi R_2 par λ, R_3 par λ, C par λ.

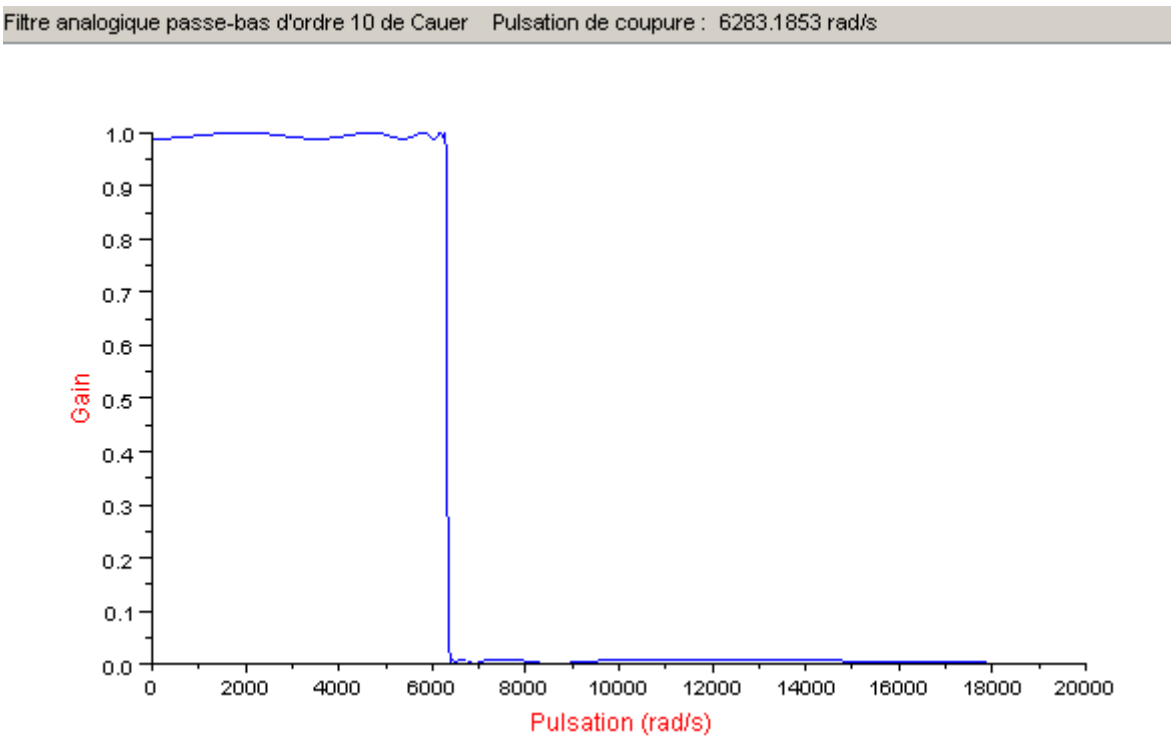
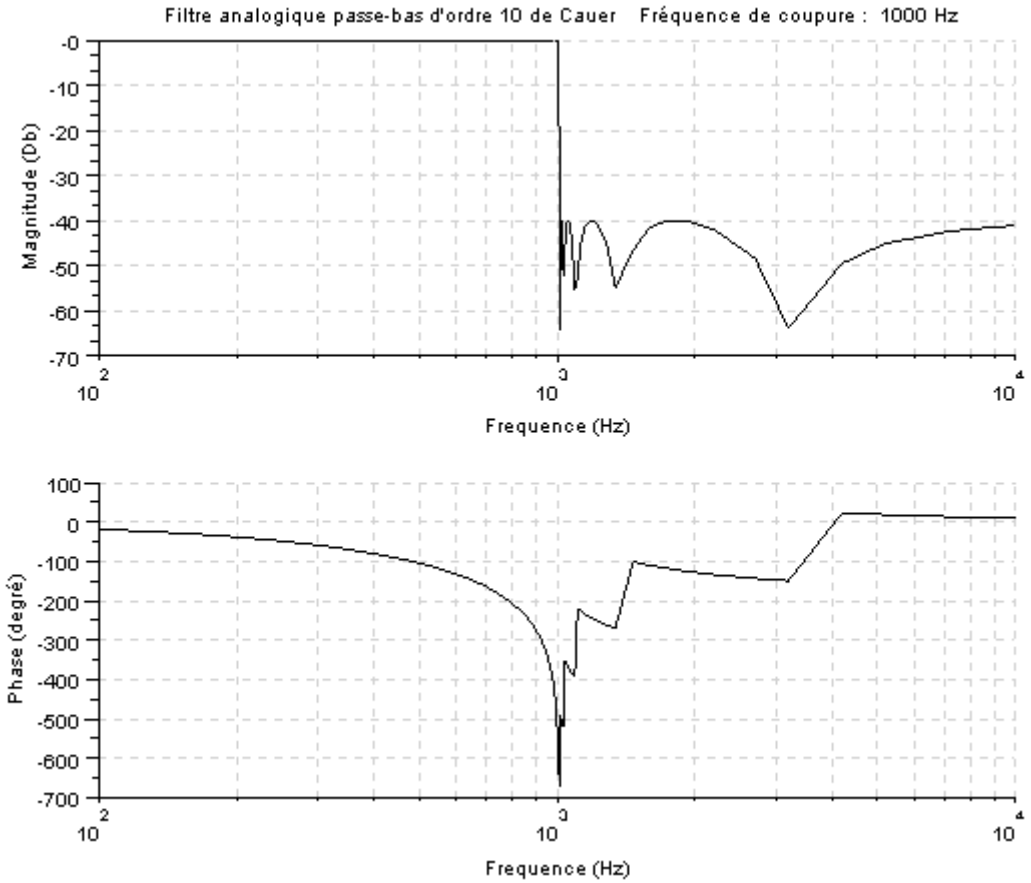


Diagramme de Bode :



Valeurs des composants :

Filtre réjecteur numéro 1
omegaz = 20853.851 ; omegap = 4004.4407 ; Qp = 0.6342786
C = 1.000D-09 ; CH = 1.000D-09
R1 = 10000 ; R2 = 10000 ; R3 = 100000 ; RQ = 2685.5722 ; Rk = 3687.3233
FH = 4004.4407

Filtre réjecteur numéro 2
omegaz = 8609.1204 ; omegap = 5422.6487 ; Qp = 1.9350969
C = 1.000D-09 ; CH = 1.000D-09
R1 = 10000 ; R2 = 1000 ; R3 = 10000 ; RQ = 2870.1938 ; Rk = 3967.3944
FH = 5422.6487

Filtre réjecteur numéro 3
omegaz = 6908.101 ; omegap = 6053.0368 ; Qp = 6.7236631
C = 1.000D-09 ; CH = 1.000D-09
R1 = 10000 ; R2 = 1000 ; R3 = 10000 ; RQ = 12447.326 ; Rk = 7677.6669
FH = 6053.0368

Filtre réjecteur numéro 4
omegaz = 6493.9128 ; omegap = 6248.5611 ; Qp = 24.054629
C = 1.000D-09 ; CH = 1.000D-09
R1 = 10000 ; R2 = 1000 ; R3 = 10000 ; RQ = 47109.258 ; Rk = 9258.6388
FH = 6248.5611

Filtre réjecteur numéro 5
omegaz = 6388.8955 ; omegap = 6301.2861 ; Qp = 115.19782
C = 1.000D-09 ; CH = 1.000D-09
R1 = 10000 ; R2 = 1000 ; R3 = 10000 ; RQ = 229395.64 ; Rk = 9727.6252
FH = 6301.2861

Exemple 2

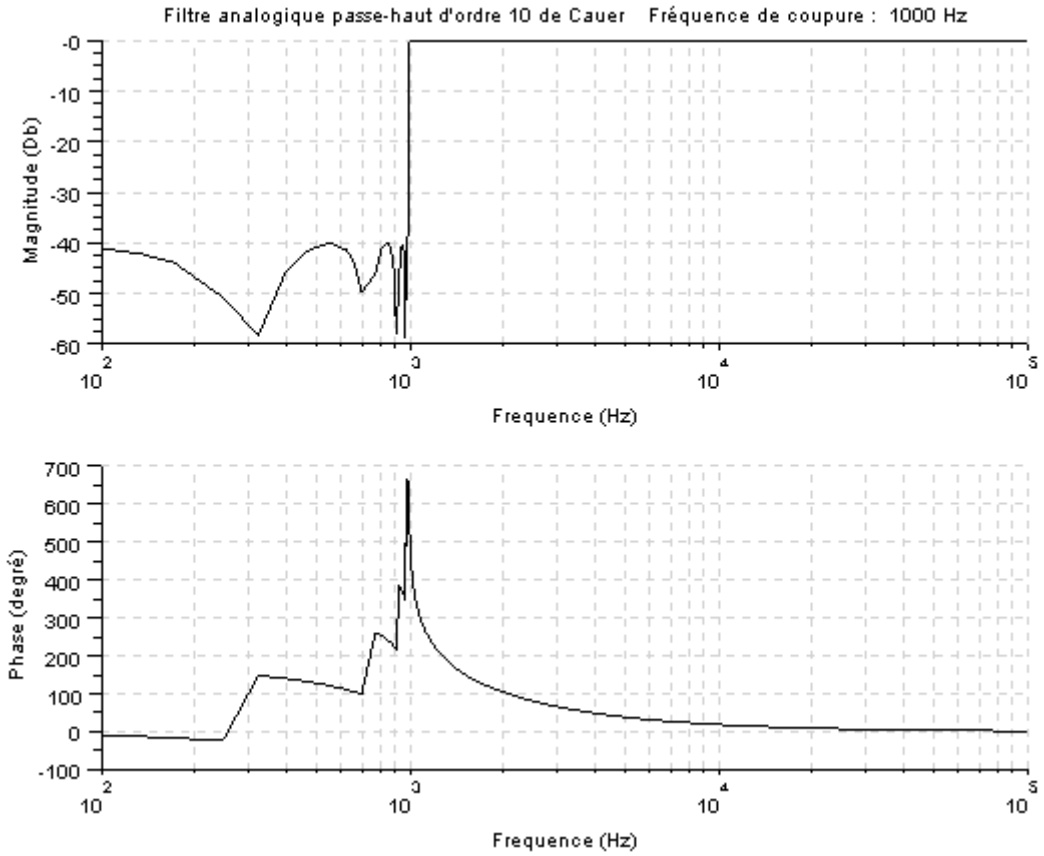
Réalisation d'un filtre passe-haut ayant les mêmes caractéristiques que le filtre passe-bas de l'exemple 1.

```
//---------- Données
fc = 1000 ;
n = 10 ;
A1 = 0.1 ;
A2 = 40 ;

//---------- Pulsation de coupure
omegac = 2*%pi*fc ;
//---------- Filtre passe-bas normalisé : omegac = 1
[h,pols,zers,g] = analpf(n,'ellip',[1-10^(-A1/20),10^(-A2/20)],1) ;
//---------- Transformation en filtre passe-haut de fréquence de coupure omegac = 2*pi*fc
p = poly(0,'p') ;
m = length(zers) ;
n = length(pols) ;
N = 1 ;
for k = 1:m
   N = N*(omegac - zers(k)*p) ;
end
D = 1 ;
for k = 1:n
   D = D*(omegac - pols(k)*p) ;
end
H = g*p^(n-m)*real(N)/real(D) ;
//---------- Diagramme de Bode
scf(3) ;
elatf = syslin('c',H(2),H(3)) ;
bode(elatf,1D2,1D5)
title('Filtre analogique passe-haut d''ordre ' + string(n)' + ' de Cauer' ...
            + '      Fréquence de coupure :   ' + string(fc) + ' Hz')

//---------- Filtres réjecteurs KHN
for i = 1:n/2
   a0 = omegac^2/abs(zers(i))^2 ;
   b0 = omegac^2/abs(pols(i))^2 ;
   b1 = - 2*omegac*real(pols(i))/abs(pols(i))^2 ;
   omegaz = sqrt(a0) ;
   omegap = sqrt(b0) ;
   Qp = sqrt(b0)/b1 ;
   //----- Choix des capacités et des résistances
   C = 1 ; CH = 1 ; R1 = 1 ; R2 = 1 ; R3 = 1 ;
   //----- Calcul des résistances
   RQ = (2*Qp - 1)*R2 ;
   Rk = R3/(omegaz/omegap)^2 ;
   //----- Fréquence d'horloge
   FH = (C/CH)*omegap ;
   //----- Résultats
   printf('\n==========   Filtre réjecteur numéro %i\n',i)
   omegaz, omegap, Qp
   C, CH
   R1, R2, R3, RQ, Rk
   FH
end
```

Diagramme de Bode :



Valeurs des composants :

Filtre réjecteur numéro 1
omegaz = 1893.0997 ; omegap = 9858.6596 ; Qp = 0.6342786
C = 3.300D-08 ; CH = 1.000D-09
R1 = 10000 ; R2 = 10000 ; R3 = 1000 ; RQ = 2685.5722 ; Rk = 27119.944
FH = 325335.77

Filtre réjecteur numéro 2
omegaz = 4585.6505 ; omegap = 7280.2831 ; Qp = 1.9350969
C = 3.300D-08 ; CH = 1.000D-09
R1 = 10000 ; R2 = 1000 ; R3 = 10000 ; RQ = 2870.1938 ; Rk = 25205.46
FH = 240249.34

Filtre réjecteur numéro 3
omegaz = 5714.8003 ; omegap = 6522.0845 ; Qp = 6.7236631
C = 3.300D-08 ; CH = 1.000D-09
R1 = 10000 ; R2 = 1000 ; R3 = 10000 ; RQ = 12447.326 ; Rk = 13024.79
FH = 215228.79

Filtre réjecteur numéro 4
omegaz = 6079.2959 ; omegap = 6318.0013 ; Qp = 24.054629
C = 3.300D-08 ; CH = 1.000D-09
R1 = 10000 ; R2 = 1000 ; R3 = 10000 ; RQ = 47109.258 ; Rk = 10800.724
FH = 208494.04

Filtre réjecteur numéro 5
omegaz = 6179.2242 ; omegap = 6265.1365 ; Qp = 115.19782
C = 3.300D-08 ; CH = 1.000D-09
R1 = 10000 ; R2 = 1000 ; R3 = 10000 ; RQ = 229395.64 ; Rk = 10280.001
FH = 206749.51

Exemple 3

A partir des exemples 1 et 2, réalisons un filtre passe-bande elliptique à capacités commutées de fréquence centrale f_c = 2010 Hz et de bande passante B = 500 Hz.
Le filtre passe-bande sera obtenu en plaçant en série un filtre passe-bas KHN et un filtre passe-haut KHN d'ordres 8.

Le programme Scilab est le suivant :

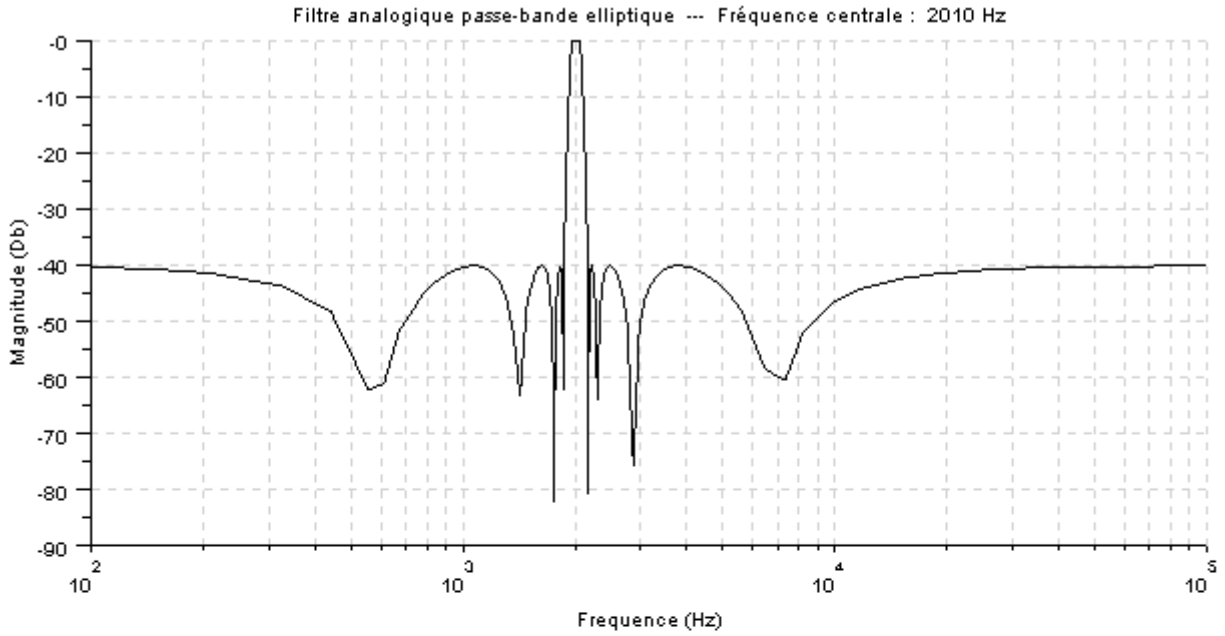
```
//---------- Données
fc = 2010 ;
B = 500 ;
n = 8 ;
A1 = 0.1 ;
A2 = 40 ;

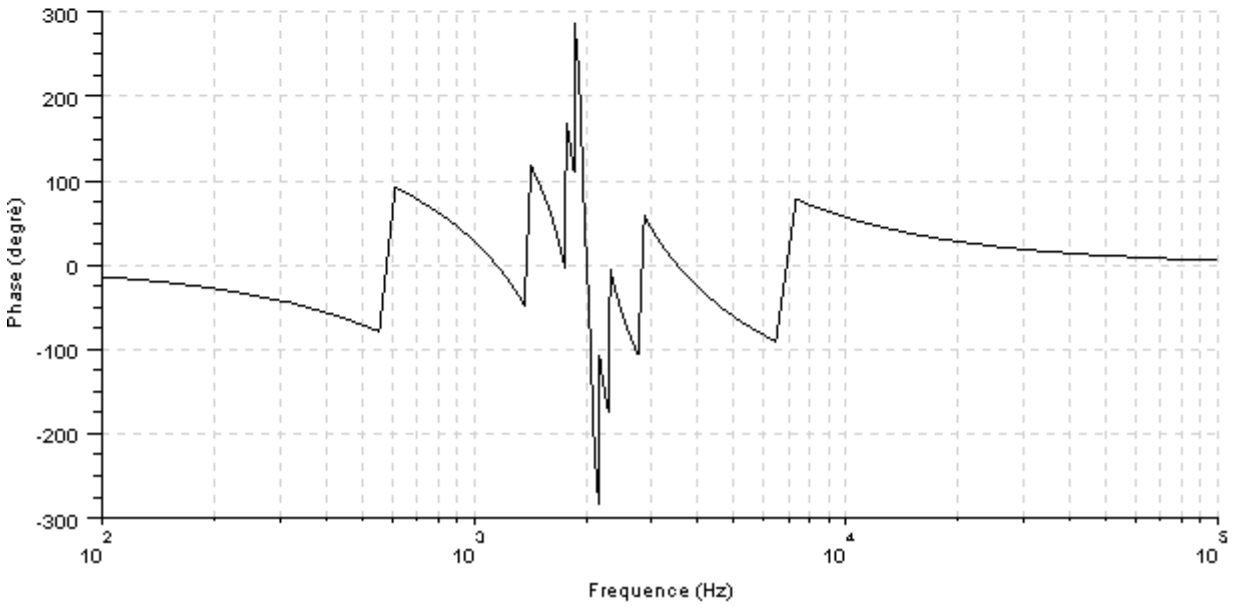
//---------- Pulsation centrale
omegac = 2*%pi*fc ;
//---------- Pulsations de coupure
omegac1 = (-B + sqrt(B^2 + 4*omegac^2))/2 ;
omegac2 = (+B + sqrt(B^2 + 4*omegac^2))/2 ;
//---------- Filtre passe-bas normalisé : omegac = 1
[h,pols,zers,g] = analpf(n,'ellip',[1-10^(-A1/20),10^(-A2/20)],1) ;

//========== Filtre passe-haut elliptique
//---------- Transformation en filtre passe-haut de pulsation de coupure omegac1
p = poly(0,'p') ;
m = length(zers) ;
n = length(pols) ;
N = 1 ;
for k = 1:m
   N = N*(omegac1 - zers(k)*p) ;
end
D = 1 ;
for k = 1:n
   D = D*(omegac1 - pols(k)*p) ;
end
H1 = g*p^(n-m)*real(N)/real(D) ;
//---------- Filtres réjecteurs KHN
printf('\n\n====   FILTRE PASSE-HAUT ELLIPTIQUE   ====')
for i = 1:n/2
   a0 = omegac1^2/abs(zers(i))^2 ;
   b0 = omegac1^2/abs(pols(i))^2 ;
   b1 = - 2*omegac1*real(pols(i))/abs(pols(i))^2 ;
   omegaz = sqrt(a0) ;
   omegap = sqrt(b0) ;
   Qp = sqrt(b0)/b1 ;
   //----- Choix des capacités et des résistances
   C = 1 ; CH = 1 ; R1 = 1 ; R2 = 1 ; R3 = 1 ;
   //----- Calcul des résistances
   RQ = (2*Qp - 1)*R2 ;
   Rk = R3/(omegaz/omegap)^2 ;
   //----- Fréquence d'horloge
   FH = (C/CH)*omegap ;
   //----- Résultats
   printf('\n\n--->   Filtre réjecteur numéro %i\n',i)
   printf('\n omegaz = %f ; omegap = %f ; Qp = %f', omegaz, omegap, Qp)
   printf('\n RQ/R2 = %f ; Rk/R3 = %f', RQ, Rk)
   printf('\n FH/(C/CH) = %f', FH)
end
```

```
//========== Filtre passe-bas elliptique
//---------- Transformation en filtre passe-bas de pulsation de coupure omegac2
p = poly(0,'p') ;
m = length(zers) ;
n = length(pols) ;
N = 1 ;
for k = 1:m
   N = N*(p - omegac2*zers(k)) ;
end
D = 1 ;
for k = 1:n
   D = D*(p - omegac2*pols(k)) ;
end
H2 = g*omegac2^(n-m)*real(N)/real(D) ;
printf('\n\n====   FILTRE PASSE-BAS ELLIPTIQUE   ====')
for i = 1:n/2
   a0 = omegac2^2*abs(zers(i))^2 ;
   b0 = omegac2^2*abs(pols(i))^2 ;
   b1 = - 2*omegac2*real(pols(i)) ;
   omegaz = sqrt(a0) ;
   omegap = sqrt(b0) ;
   Qp = sqrt(b0)/b1 ;
   //----- Choix des capacités et des résistances
   C = 1 ; CH = 1 ; R1 = 1 ; R2 = 1 ; R3 = 1 ;
   //----- Calcul des résistances
   RQ = (2*Qp - 1)*R2 ;
   Rk = R3/(omegaz/omegap)^2 ;
   //----- Fréquence d'horloge
   FH = (C/CH)*omegap ;
   //----- Résultats
   printf('\n\n--->   Filtre réjecteur numéro %i\n',i)
   printf('\n omegaz = %f ; omegap = %f ; Qp = %f', omegaz, omegap, Qp)
   printf('\n RQ/R2 = %f ; Rk/R3 = %f', RQ, Rk)
   printf('\n FH/(C/CH) = %f', FH)
end

//========== Filtre passe-bande elliptique
H = H1*H2 ;
//---------- Diagramme de Bode
scf(4) ;
elatf = syslin('c',H(2),H(3)) ;
bode(elatf,1D2,1D5)
title('Filtre analogique passe-bande elliptique   ---   Fréquence centrale :   ' + string(fc) + ' Hz')
```


Filtre analogique passe-bande elliptique --- Fréquence centrale : 2010 Hz
Magnitude (Db)
-0
-10
-20
-30
-40
-50
-60
-70
-80
-90
10^2
10^3
10^4
10^5
Frequence (Hz)


Phase (degré)
300
200
100
0
-100
-200
-300
10^2
10^3
10^4
10^5
Frequence (Hz)

Valeurs des composants :

==== FILTRE PASSE-BAS ELLIPTIQUE ====

---> Filtre réjecteur numéro 1

omegaz = 43581.220430 ; omegap = 8366.175972 ; Qp = 0.634325
RQ/R2 = 0.268649 ; Rk/R3 = 0.036852
FH/(C/CH) = 8366.175972

---> Filtre réjecteur numéro 2

omegaz = 17997.952403 ; omegap = 11327.097952 ; Qp = 1.938314
RQ/R2 = 2.876628 ; Rk/R3 = 0.396087
FH/(C/CH) = 11327.097952

---> Filtre réjecteur numéro 3

omegaz = 14460.918499 ; omegap = 12635.289168 ; Qp = 6.870092
RQ/R2 = 12.740183 ; Rk/R3 = 0.763447
FH/(C/CH) = 12635.289168

---> Filtre réjecteur numéro 4

omegaz = 13657.250825 ; omegap = 13012.202523 ; Qp = 32.876447
RQ/R2 = 64.752895 ; Rk/R3 = 0.907768
FH/(C/CH) = 13012.202523

==== FILTRE PASSE-HAUT ELLIPTIQUE ====

---> Filtre réjecteur numéro 1

omegaz = 3659.758800 ; omegap = 19064.475274 ; Qp = 0.634325
RQ/R2 = 0.268649 ; Rk/R3 = 27.135935
FH/(C/CH) = 19064.475274

---> Filtre réjecteur numéro 2

omegaz = 8861.938925 ; omegap = 14080.990174 ; Qp = 1.938314
RQ/R2 = 2.876628 ; Rk/R3 = 2.524695
FH/(C/CH) = 14080.990174

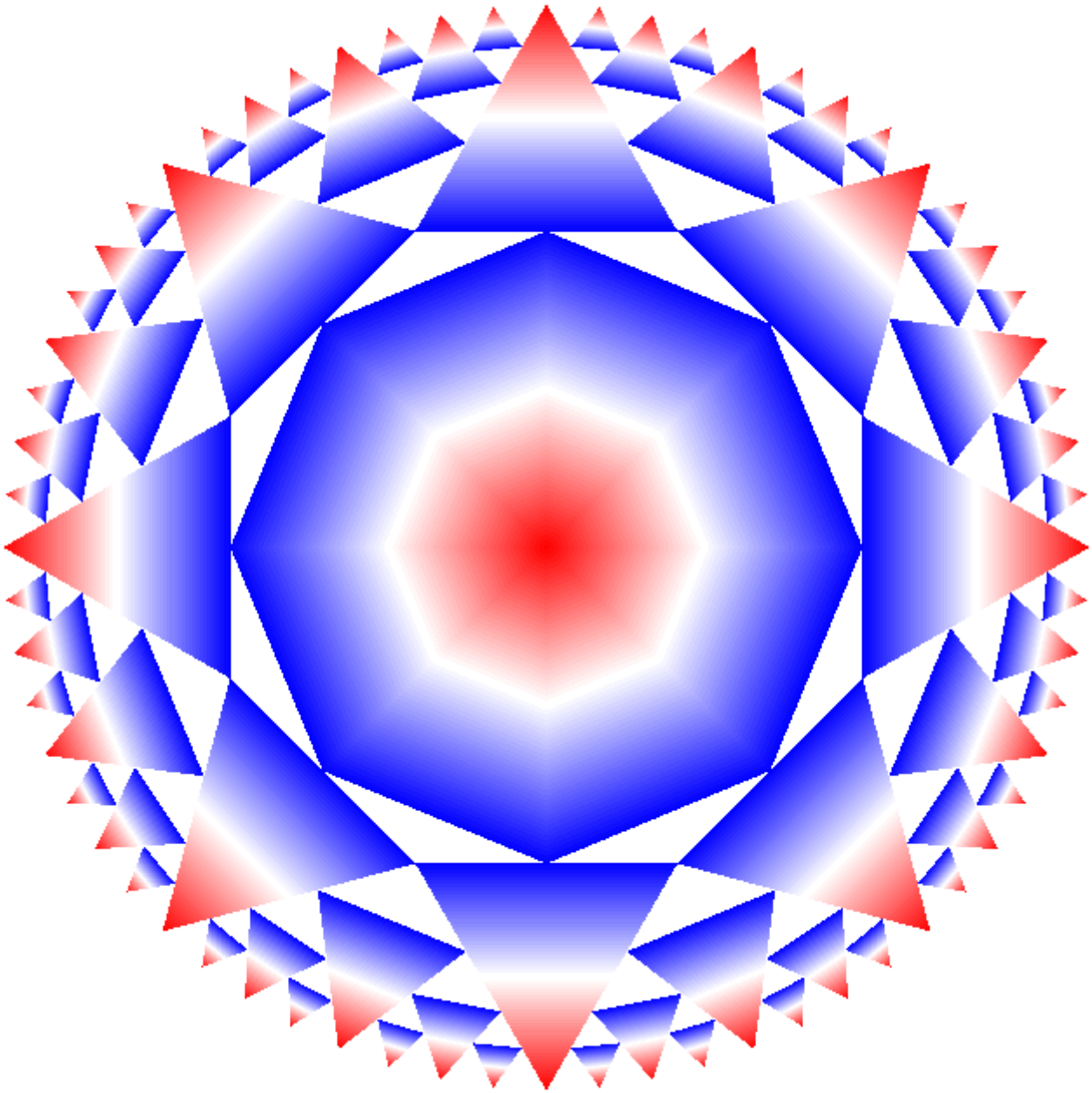
---> Filtre réjecteur numéro 3

omegaz = 11029.503760 ; omegap = 12623.118699 ; Qp = 6.870092
RQ/R2 = 12.740183 ; Rk/R3 = 1.309849
FH/(C/CH) = 12623.118699

---> Filtre réjecteur numéro 4

omegaz = 11678.540360 ; omegap = 12257.475603 ; Qp = 32.876447
RQ/R2 = 64.752895 ; Rk/R3 = 1.101603
FH/(C/CH) = 12257.475603

Mathématique & Informatique



Mathématique & Informatique

RAPPELS MATHÉMATIQUES SUCCINCTS SUR LE TRAITEMENT DU SIGNAL

De la série à l'intégrale de **Fourier** (biographie @ http://joseph.fourier.free.fr/)

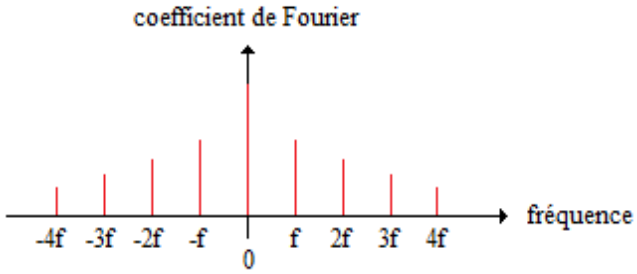
- Un signal périodique x(t) de fréquence f = 1/T est composé d'un ensemble de signaux sinusoïdaux représentant les fréquences multiples de f. Le signal x(t) s'écrit sous la forme d'une série de **Fourier** :

$$x(t) = \sum_{n=-\infty}^{n=+\infty} c_n e^{j2\pi nft}$$

La fréquence nf est caractérisée dans x(t) par un coefficient c_n appelé coefficient de **Fourier** de x(t) :

$$c_n = \frac{1}{T}\int_0^T x(t)e^{-j2\pi nft}\,dt$$

Le module du nombre complexe c_n indique l'importance de la fréquence nf dans le signal x(t).
Le spectre de x(t) est discret :



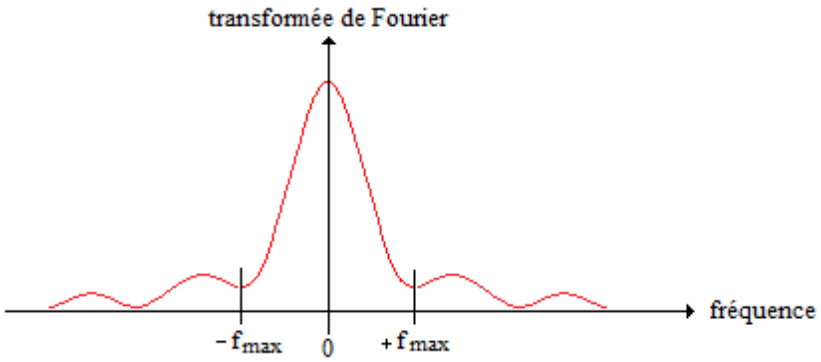
- Un signal non périodique x(t) est composé d'un ensemble de signaux sinusoïdaux représentant toutes les fréquences f possibles. Le signal x(t) s'écrit sous la forme d'une intégrale de **Fourier** :

$$x(t) = \int_{-\infty}^{+\infty} X(f)e^{j2\pi ft}\,df$$

La fréquence f est caractérisée dans x(t) par un coefficient X(f) appelé transformée de **Fourier** de x(t) :

$$X(f) = \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft}\,dt$$

Le module du nombre complexe X(f) indique l'importance de la fréquence f dans le signal x(t).
Le spectre de x(t) est continu :



Mathématique & Informatique

Conventions d'écriture

1) L'unité imaginaire est notée i en Mathématique (i pour imaginaire) et j en Physique (j pour ne pas la confondre avec l'intensité du courant électrique i(t)). En programmation Scilab, cette constante prédéfinie s'écrit %i.
2) Les fréquences négatives n'ont pas de signification en tant que telles. Dans les formules, elles sont une commodité de calcul associée à l'emploi des exponentielles complexes : $e^{j\theta} = \cos\theta + j\sin\theta$

Transformée de **Fourier** discrète

La définition intégrale de X(f) ne permet pas son calcul formel dans le cas d'un signal x(t) quelconque.
En échantillonnant x(t) à la fréquence $f_e = 1/T_e$ et en limitant le calcul de l'intégrale à un intervalle de temps $[0,(N-1)T_e]$, on obtient l'approximation suivante pour une fréquence $f_k = (k/N)f_e$, $0 \le k \le N-1$:

$$X(f_k) \approx T_e \sum_{n=0}^{N-1} x(nT_e)\, e^{-j2\pi\frac{nk}{N}} = T_e\, X(k)$$

Notons x(0), ... , x(N-1) la suite des N valeurs prises par le signal x(t) aux instants d'échantillonnage 0, ... , $(N-1)T_e$
La suite des N termes X(0), ... , X(N-1) est la transformée de **Fourier** discrète (TFD) de la suite des N échantillons x(0), ... , x(N-1). L'algorithme de calcul optimisé des nombres complexes X(k) est appelé transformée de **Fourier** rapide (FFT = Fast Fourier Transform). Il est dû à James **Cooley** et John **Tukey** et nécessite un nombre d'opérations de l'ordre de $N\log_2 N$ ⇨ G. **Baudoin** et J.-F**. Bercher** : http://www.esiee.fr/~bercherj/New/polys/poly_tfd.pdf

Reconstitution d'un signal analogique

En réduisant le spectre d'un signal quelconque x(t) a un domaine fréquentiel borné $[-f_{max} , +f_{max}]$ et en échantillonnant x(t) à une fréquence $f_e > 2f_{max}$ (théorème de **Shannon**), on montre que la fonction x(t) peut se mettre sous la forme :

$$x(t) = \sum_{n=-\infty}^{n=+\infty} x(nT_e)\,\mathrm{sinc}\left[\pi\left(\frac{t}{T_e} - n\right)\right]$$

Dans cette expression, $T_e = 1/f_e$ est la période d'échantillonnage et sinc(u) = sin(u)/u est la fonction sinus cardinal.
On peut donc reconstituer x(t) uniquement à partir des valeurs $x(nT_e)$ prises aux différents instants d'échantillonnage. Théoriquement, la série comporte une infinité de termes. En pratique, on ne calcule qu'une somme finie d'entre eux puisqu'on ne dispose que d'un nombre fini d'échantillons. On introduit donc une erreur de troncature qui s'ajoute aux imprécisions sur la mesure et sur la conversion numérique des valeurs échantillonnées.

Les fonctions Scilab

Soit un vecteur *a* de N nombres complexes *a* (1), ... , *a* (N). La fonction Scilab *fft* retourne le vecteur *F = fft(a)* constitué des N nombres complexes F(1), ... , F(N) définis par :

$$F(k) = \sum_{n=1}^{N} a(n)\, e^{-j2\pi\frac{(n-1)(k-1)}{N}} \quad , \quad 1 \le k \le N$$

Soit un vecteur *a* de N nombres complexes *a* (1), ... , *a* (N). La fonction Scilab *sinc* retourne le vecteur *S = sinc(a)* constitué des N nombres complexes S(1), ... , S(N) définis par :

$$S(k) = \frac{\sin(a(k))}{a(k)} \quad , \quad 1 \le k \le N$$

Mathématique & Informatique

Série de **Fourier** sous forme réelle

Soit un signal x(t) périodique, de période T = 1/f. Sous certaines conditions peu restrictives (théorème de **Dirichlet**), la fonction x(t) est développable en série trigonométrique :

$$x(t) = \sum_{n=-\infty}^{n=+\infty} c_n \, e^{\,j\,2\pi n f t} = c_0 + \sum_{n=1}^{\infty} (c_n + c_{-n}) \cos 2\pi n f t + j \sum_{n=1}^{\infty} (c_n - c_{-n}) \sin 2\pi n f t$$

avec :

$$c_0 = \frac{1}{T} \int_0^T x(t) dt$$

$$c_n + c_{-n} = \frac{2}{T} \int_0^T x(t) \cos 2\pi n f t \, dt$$

$$c_n - c_{-n} = -j \frac{2}{T} \int_0^T x(t) \sin 2\pi n f t \, dt$$

Posons :

$$a_n = c_n + c_{-n} \quad ; \quad b_n = j(c_n - c_{-n})$$

Le signal x(t) s'écrit :

$$x(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos 2\pi n f t + b_n \sin 2\pi n f t)$$

avec :

$$a_n = \frac{2}{T} \int_0^T x(t) \cos 2\pi n f t \, dt \quad ; \quad b_n = \frac{2}{T} \int_0^T x(t) \sin 2\pi n f t \, dt$$

Dans l'expression des coefficients a_n et b_n , l'intégrale peut être calculée sur un intervalle quelconque de largeur T, puisque toute fonction φ de période T possède la propriété suivante :

$$\int_0^T \varphi(t) dt = \int_\lambda^{\lambda+T} \varphi(t) dt \quad , \forall \lambda$$

Si la fonction x(t) est impaire, tous les termes a_n sont nuls ; de même si x(t) est paire, tous les termes b_n sont nuls.
Le terme constant $a_0/2$ de la série de **Fourier** correspond à la valeur moyenne du signal x(t) sur une période :

$$\frac{a_0}{2} = \frac{1}{T} \int_0^T x(t) dt$$

Transformées de **Fourier** cosinus et sinus

On définit la transformée de **Fourier** cosinus et la transformée de **Fourier** sinus d'un signal x(t) par :

$$X_C(f) = \int_0^{+\infty} x(t) \cos 2\pi f t \, dt$$

$$X_S(f) = \int_0^{+\infty} x(t) \sin 2\pi f t \, dt$$

Les intégrales $X_C(f)$ et $X_S(f)$ sont des nombres réels. Il existe des tables donnant les transformées de **Fourier** cosinus et les transformées de **Fourier** sinus de différentes fonctions x(t). Par exemple :

$x(t)$	$X_C(f)$	$X_S(f)$
$\frac{1}{t^p} \quad (0<p<1)$	$\frac{\pi^p}{2^{2-p} \; f^{1-p} \; \Gamma(p) \cos(p\pi/2)}$	$\frac{\pi^p}{2^{2-p} \; f^{1-p} \; \Gamma(p) \sin(p\pi/2)}$
$\frac{\sin \lambda t}{t}$	$\pi/2 \quad \text{si} \quad f < \lambda/2\pi$ $\pi/4 \quad \text{si} \quad f = \lambda/2\pi$ $0 \quad \text{si} \quad f > \lambda/2\pi$	$\text{Argcoth}\left(\frac{2\pi f}{\lambda}\right)$

Pour exprimer la transformée de **Fourier** d'un signal x(t), on écrit :

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j 2\pi f t} \, dt = \int_{-\infty}^{+\infty} x(t) \cos 2\pi f t \, dt - j \int_{-\infty}^{+\infty} x(t) \sin 2\pi f t \, dt$$

$$= \int_0^{+\infty} [x(-t) + x(t)] \cos 2\pi f t \, dt + j \int_0^{+\infty} [x(-t) - x(t)] \sin 2\pi f t \, dt$$

Posons :

a(t) = x(-t) + x(t)
b(t) = x(-t) - x(t)

Soit $A_C(f)$ la transformée de **Fourier** cosinus de a(t), et $B_S(f)$ la transformée de **Fourier** sinus de b(t), on obtient :

$$X(f) = A_C(f) + j B_S(f)$$

En général X(f) est un nombre complexe, c'est un réel si x(t) est paire et un imaginaire pur si x(t) est impaire.

Remarque sur la symétrie du spectre d'un signal quelconque

Pour les signaux périodiques, on utilise les coefficients de **Fourier** :

$$c_n = \frac{1}{T}\int_0^T x(t)\, e^{-j2\pi n f t}\, dt = \frac{1}{T}\int_0^T x(t)\cos(2\pi n f t)dt - j\frac{1}{T}\int_0^T x(t)\sin(2\pi n f t)dt$$

$$c_{-n} = \frac{1}{T}\int_0^T x(t)\, e^{+j2\pi n f t}\, dt = \frac{1}{T}\int_0^T x(t)\cos(2\pi n f t)dt + j\frac{1}{T}\int_0^T x(t)\sin(2\pi n f t)dt$$

Les nombres complexes c_n et c_{-n} sont conjugués, donc leurs modules sont identiques. Par suite, le spectre de raies représentant $|c_n|$ en fonction de f est symétrique par rapport à l'origine.
Pour les signaux non périodiques, on utilise les transformées de **Fourier** :

$$X(f) = \int_{-\infty}^{+\infty} x(t)\, e^{-j2\pi f t}\, dt = \int_{-\infty}^{+\infty} x(t)\cos(2\pi f t)dt - j\int_{-\infty}^{+\infty} x(t)\sin(2\pi f t)dt$$

$$X(-f) = \int_{-\infty}^{+\infty} x(t)\, e^{+j2\pi f t}\, dt = \int_{-\infty}^{+\infty} x(t)\cos(2\pi f t)dt + j\int_{-\infty}^{+\infty} x(t)\sin(2\pi f t)dt$$

Les nombres complexes X(f) et X(-f) sont conjugués, donc leurs modules sont égaux et le spectre du signal x(t), c'est-à-dire le graphe représentant |X(f)| en fonction de f, est symétrique par rapport à l'origine.
De même, la courbe représentant le module $|X(f_k)|$ de la transformée de **Fourier** discrète en fonction de la fréquence f_k est symétrique par rapport à son centre. Prouvons ce résultat.
Le signal x(t) est échantillonné aux instants 0, T_e, $2T_e$, ... , $(N-1)T_e$ où T_e est la période d'échantillonnage.
Pour une fréquence $f_k = (k/N)f_e$, k prenant les valeurs entières 0, 1, 2, ... , N-1 et $f_e = 1/T_e$ étant la fréquence d'échantillonnage, l'algorithme FFT permet de calculer :

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j\,2\,\pi\,\frac{n\,k}{N}}$$

Les valeurs x(0), x(1), x(2), ... , x(N-1) sont les valeurs du signal x(t) aux N instants d'échantillonnage.
Le spectre de x(t) est la courbe donnant $|X(f_k)| = T_e\,|X(k)|$ en fonction de f_k.
Pour montrer que cette courbe est symétrique par rapport à son centre, il suffit d'établir que :

$$\left|X(f_k)\right| = \left|X(f_{N-k})\right| \quad \Leftrightarrow \quad \left|X(k)\right| = \left|X(N-k)\right| \qquad \text{pour} \quad k = 1, \ldots, E\left(\frac{N}{2}\right)$$

On a :

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j\,2\,\pi\,\frac{nk}{N}} = \sum_{n=0}^{N-1} x(n)\cos\left(2\pi\,\frac{n\,k}{N}\right) - j\sum_{n=0}^{N-1} x(n)\sin\left(2\pi\,\frac{n\,k}{N}\right)$$

$$X(N-k) = \sum_{n=0}^{N-1} x(n)\, e^{-j\,2\,\pi\,\frac{n(N-k)}{N}} = \sum_{n=0}^{N-1} x(n)\cos\left(2\pi\,\frac{n\,k}{N}\right) + j\sum_{n=0}^{N-1} x(n)\sin\left(2\pi\,\frac{n\,k}{N}\right)$$

D'où :

$$X(k) = \overline{X(N-k)} \qquad \Rightarrow \qquad \left|\,X(k)\,\right| = \left|\,X(N-k)\,\right|$$

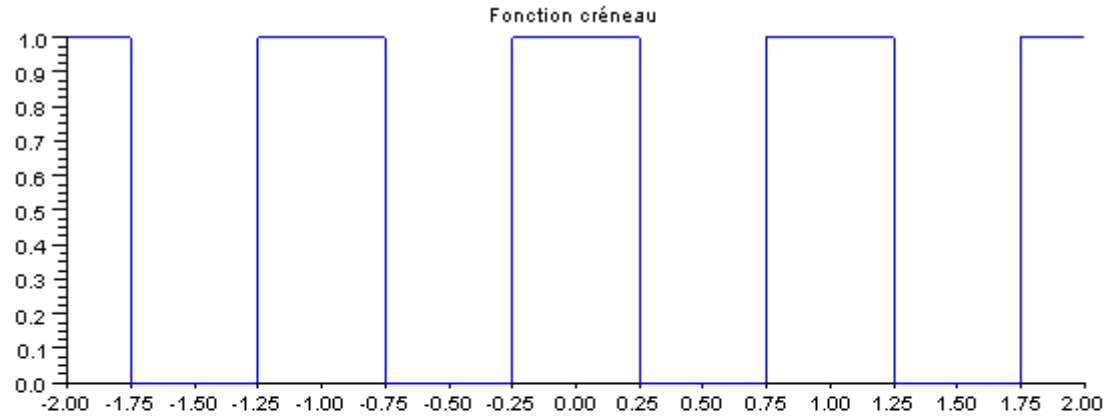
Fonction créneau

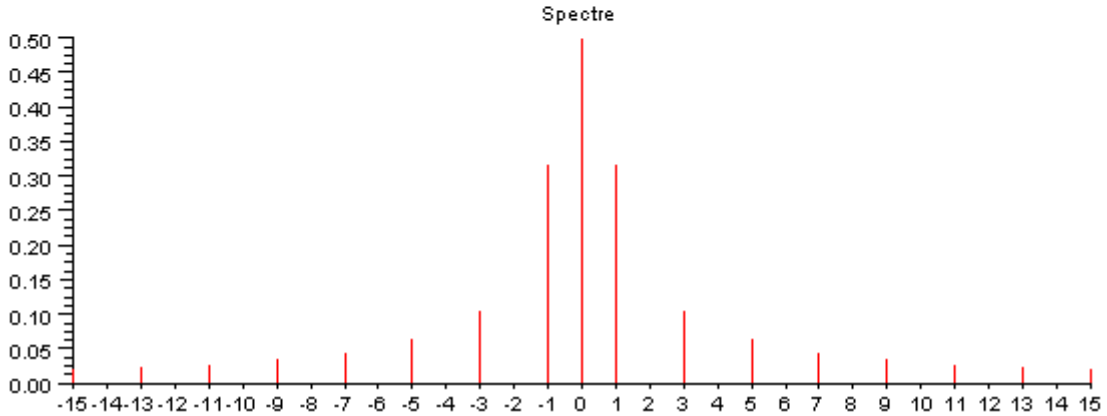
La fonction créneau de période T est définie par : x(t) = A si t ∈ [-T/4 , +T/4], 0 si t ∈ [-T/2 , -T/4[∪]+T/4 , +T/2]
Calculons son coefficient de **Fourier** :

$$c_n = \frac{1}{T}\int_0^T x(t)e^{-j2\pi nft}\,dt = \frac{1}{T}\int_{-\frac{T}{4}}^{+\frac{T}{4}} Ae^{-j2\pi nft}\,dt = -\frac{A}{j2\pi n}\left[e^{-j2\pi nft}\right]_{-\frac{T}{4}}^{+\frac{T}{4}} = \frac{A}{2}\frac{\sin\left(n\frac{\pi}{2}\right)}{n\frac{\pi}{2}} = \frac{A}{2}\mathrm{sinc}\left(n\frac{\pi}{2}\right)$$

Représentons la fonction x(t), le temps en abscisse et l'amplitude en ordonnée, puis le spectre de x(t), la fréquence en abscisse et le module du coefficient de **Fourier** en ordonnée.

```
// Fonction créneau
A = 1 ;
T = 1 ;
t = -2*T:T/1000:+2*T ;
tp = t - floor(t/T)*T ;
x = A*bool2s(tp <= T/4 | tp >= 3*T/4) ;
subplot(2,1,1) ; xtitle('Fonction créneau')
plot2d(t,x,2,nax=[0,17,3,11])
// Coefficient de Fourier
n = (-15:15) ;
cn = A/2*sinc(n*%pi/2) ;
subplot(2,1,2) ; xtitle('Spectre')
plot2d([n;n]*(1/T),[zeros(1,length(n));abs(cn)],5*ones(1,length(n)),nax=[0,31,3,11])
```





Mathématique & Informatique

Fonction porte

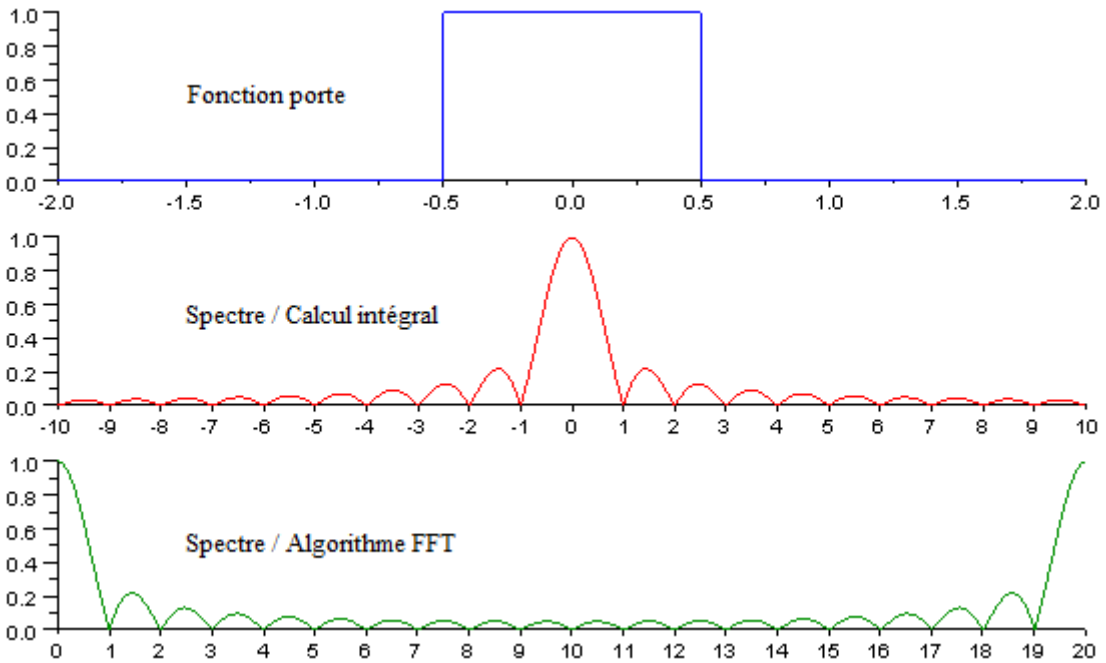
La fonction porte est définie par : x(t) = A si t ∈ [-T/2 , +T/2] , 0 sinon.
Calculons sa transformée de **Fourier** :

$$X(f) = \int_{-\infty}^{+\infty} x(t)e^{-j2\pi ft}\,dt = \int_{-\frac{T}{2}}^{+\frac{T}{2}} Ae^{-j2\pi ft}\,dt = -\frac{A}{j2\pi f}\left[e^{-j2\pi ft}\right]_{-\frac{T}{2}}^{+\frac{T}{2}} = AT\frac{\sin(\pi fT)}{\pi fT} = AT\,\mathrm{sinc}(\pi fT)$$

Représentons le spectre de x(t), la fréquence f en abscisse et le module de X(f) en ordonnée.
Cette courbe peut être obtenue de 2 manières :

1) en utilisant l'expression de X(f) issue du calcul intégral
2) en appliquant l'algorithme FFT au calcul de la transformée de **Fourier** discrète $X(f_k)$

```
// Fonction porte
A = 1 ;
T = 1 ;
t = -2*T:T/1000:+2*T ;
x = A*bool2s(t >= -T/2 & t <= +T/2) ;
subplot(3,1,1) ; xtitle('Fonction porte') ; plot2d(t,x,2,nax=[1,9,1,6])
// Transformée de Fourier
fmax = 10/T ;
f = -fmax:fmax/1000:fmax ;
Xf = A*T*sinc(%pi*f*T) ;
subplot(3,1,2) ; xtitle('Spectre / Calcul intégral') ; plot2d(f,abs(Xf),5,nax=[0,21,1,6])
// Transformée de Fourier Rapide
N = 2000 ;
fe = 2*fmax ;
Te = 1/fe ;
t = (0:N-1)*Te ;
x = A*bool2s(t > 0 & t <= T) ;
X = fft(x) ;
f = (0:N-1)/N*fe ;
Xf = Te*X ;
subplot(3,1,3) ; xtitle('Spectre / Algorithme FFT') ; plot2d(f,abs(Xf),13,nax=[0,21,1,6])
```

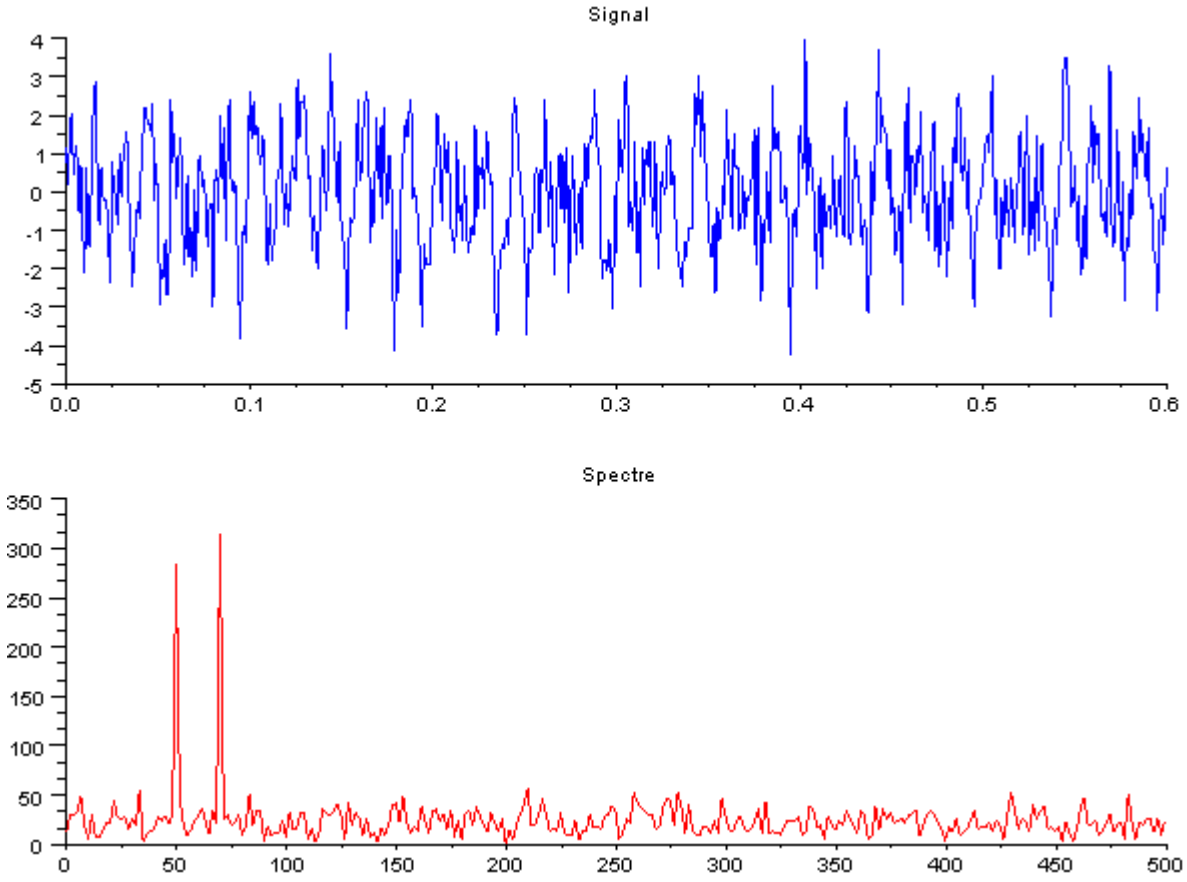


Exemple 1 : Spectre d'un signal

Pour illustrer l'utilisation de la fonction *fft* prenons l'exemple fournit par Scilab dans le dossier "Traitement du Signal"

```
// Composantes fréquentielles d'un signal
// -----------------------------------------------
// Construisons et échantillonnons à 1000 Hz un signal bruité contenant les 2 fréquences pures 50 Hz et 70 Hz
fe = 1000 ;   // Fréquence d'échantillonnage
t = 0:1/fe:0.6 ;   // Vecteur des instants d'échantillonnage
N = size(t,'*') ;   // Nombre d'échantillons
grand('setsd',0) ;   // Initialisation du générateur de nombres aléatoires
s = sin(2*%pi*50*t) + sin(2*%pi*70*t+%pi/4) + grand(1,N,'nor',0,1) ;   // Signal s(t)
subplot(2,1,1) ; xtitle('Signal') ; plot2d(t,s,2)   // Représentation graphique du signal s(t)
S = fft(s) ;   // Transformée de Fourier Rapide du signal s(t)
f = fe*(0:N/2)/N ;   // Vecteur des fréquences réduit aux N/2 premiers points (réponse symétrique de la fonction fft)
n = size(f,'*') ;   // Nombre de fréquences échantillonnées
subplot(2,1,2) ; xtitle('Spectre') ; plot2d(f,abs(S(1:n)),5)   // Représentation graphique du spectre de s(t)
```

On obtient les 2 graphiques suivants :



Le spectre de s(t) laisse clairement apparaître la prédominance des fréquences 50 Hz et 70 Hz dans le signal s(t).

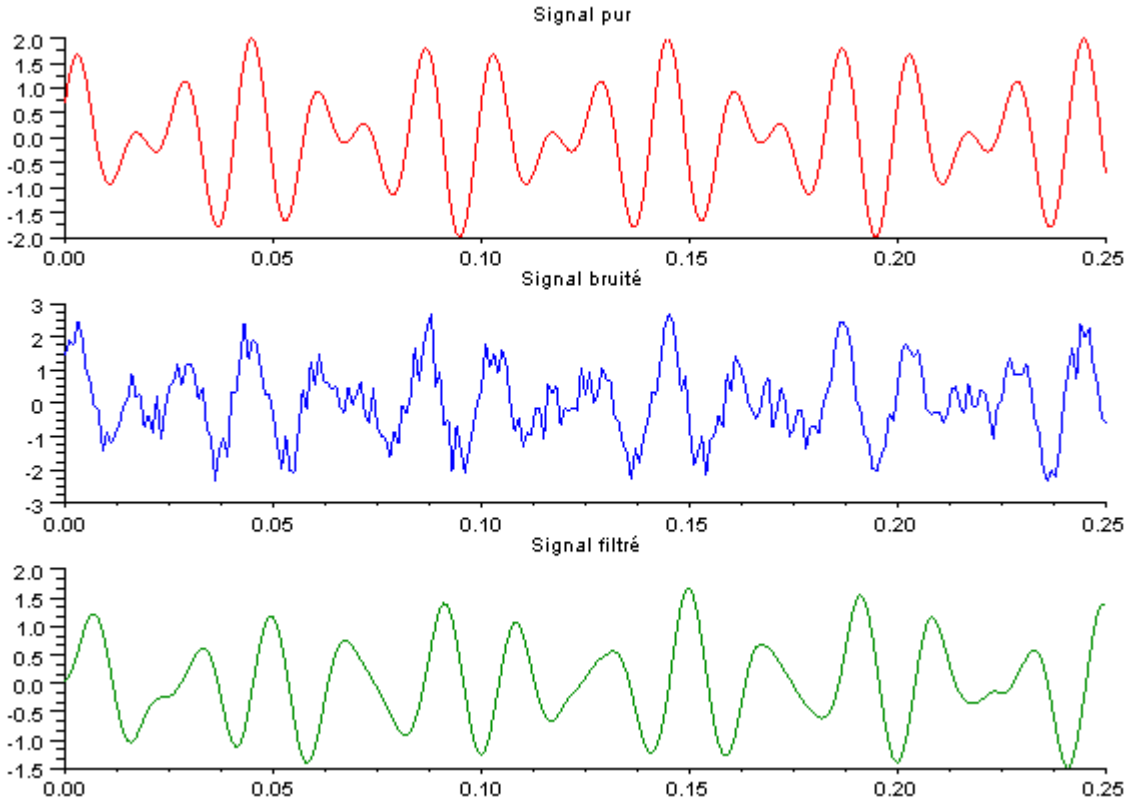
Exemple 2 : Filtrage numérique d'un signal

La fonction *iir* permet de créer un filtre numérique et la fonction *flts* est utilisée pour filtrer un signal échantillonné.

```
// Représentons un signal s1(t) composé des 2 fréquences pures 50 Hz et 70 Hz
t = linspace(0,0.25,500) ;  // Discrétisation temporelle
s1 = sin(2*%pi*50*t) + sin(2*%pi*70*t+%pi/4) ;  // Signal s1(t)
subplot(3,1,1) ; xtitle('Signal pur') ; plot2d(t,s1,5)  // Représentation graphique du signal pur s1(t)

// Construisons et échantillonnons à 1000 Hz un signal bruité s2(t) contenant les 2 fréquences pures 50 Hz et 70 Hz
fe = 1000 ;  // Fréquence d'échantillonnage
t = 0:1/fe:0.25 ;  // Vecteur des instants d'échantillonnage
N = size(t,'*') ;  // Nombre d'échantillons
grand('setsd',0) ;  // Initialisation du générateur de nombres aléatoires
s2 = sin(2*%pi*50*t) + sin(2*%pi*70*t+%pi/4) + 0.5*grand(1,N,'nor',0,1) ;  // Signal s2(t)
subplot(3,1,2) ; xtitle('Signal bruité') ; plot2d(t,s2,2)  // Représentation graphique du signal bruité s2(t)

// Filtrons le signal s2(t) avec un filtre elliptique passe-bande afin d'obtenir un signal s3(t)
ordre = 2 ;  // Ordre du filtre
fe = 1000 ;  // Fréquence d'échantillonnage (Hz)
f1 = 10 ;  // Fréquence de coupure inférieure (Hz)
f2 = 100 ;  // Fréquence de coupure supérieure (Hz)
A1 = 1 ; // Atténuation dans la bande passante (dB)
A2 = 40 ; // Atténuation dans la bande coupée (dB)
hz = iir(ordre,'bp','ellip',[f1/fe/2,f2/fe/2],[1-10^(-A1/20),10^(-A2/20)]) ;  // Fonction de transfert du filtre
sys = tf2ss(hz) ; // Transformation du filtre en système linéaire
s3 = flts(s2,sys) ; // Signal s3(t)
subplot(3,1,3) ; xtitle('Signal filtré') ; plot2d(t,s3,13)  // Représentation graphique du signal filtré s3(t)
```



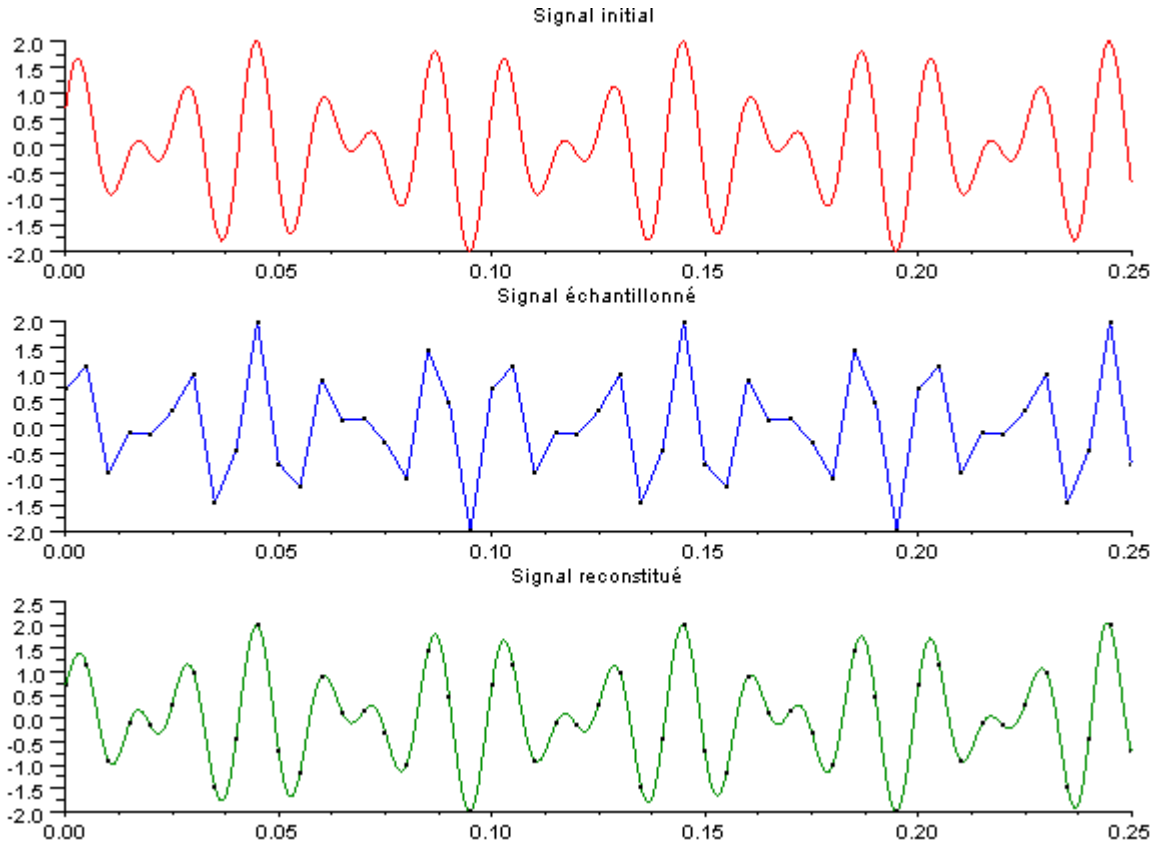
Exemple 3 : Reconstitution d'un signal

Application de la règle de **Shannon** et utilisation de la fonction *sinc*.

```
// Représentons un signal s1(t) composé des 2 fréquences pures 50 Hz et 70 Hz
t = linspace(0,0.25,500) ;   // Discrétisation temporelle
s1 = sin(2*%pi*50*t) + sin(2*%pi*70*t+%pi/4) ;   // Signal s1(t)
subplot(3,1,1) ; xtitle('Signal initial') ; plot2d(t,s1,5)   // Graphique du signal s1(t)

// Echantillonnons s1(t) à la fréquence de 200 Hz
fe = 200 ;   // Fréquence d'échantillonnage
te = 0:1/fe:0.25 ;   // Vecteur des instants d'échantillonnage
ne = size(te,'*') ;   // Nombre de points d'échantillonnage
s2 = sin(2*%pi*50*te) + sin(2*%pi*70*te+%pi/4) ;   // Signal s2(t)
subplot(3,1,2) ; xtitle('Signal échantillonné') ; plot2d(te,s2,2) ; plot2d(te,s2,0)   // Graphique du signal s2(t)

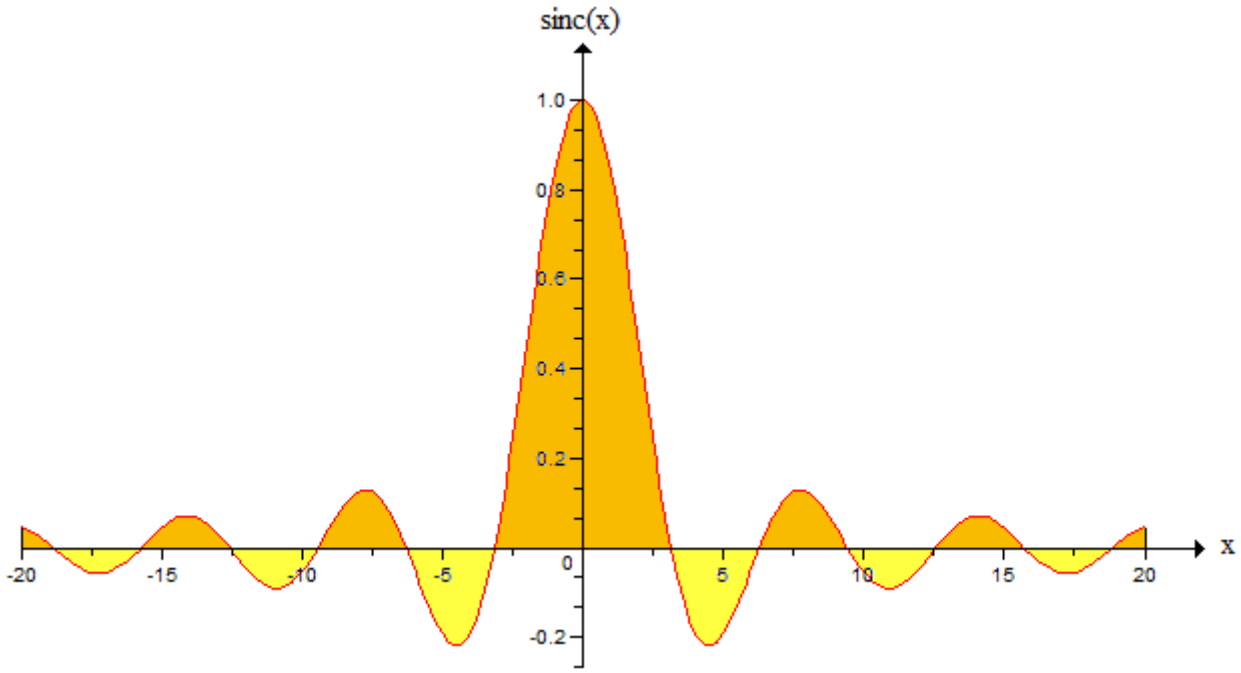
// Reconstituons le signal s1(t) à partir de s2(t)
fr = 1500 ;   // Fréquence de reconstitution
tr = 0:1/fr:0.25 ;   // Vecteur des instants de reconstitution
nr = size(tr,'*') ;   // Nombre de points de reconstitution
s3 = zeros(1,nr) ;   // Initialisation du vecteur s3
for i = 1:nr
  for j = 1:ne
    s3(i) = s3(i) + s2(j)*sinc(%pi*(tr(i)*fe-j+1)) ;   // Signal s3(t)
  end
end
subplot(3,1,3) ; xtitle('Signal reconstitué') ; plot2d(tr,s3,13) ; plot2d(te,s2,0)   // Graphique du signal s3(t)
```



La fonction sinus cardinal

$$y = \operatorname{sinc}(x) = \frac{\sin x}{x} = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n+1)!} = \prod_{n=1}^{\infty} \left(1 - \frac{x^2}{n^2 \pi^2}\right)$$

Traçons la courbe y = sinc(x) pour x ∈ [-20,+20] : *x = linspace(-20,+20,300) ; plot2d(x,sinc(x),5)*



Transformée de Laplace de la fonction sinus cardinal.

Par définition : $S(p) = \int_0^{\infty} \operatorname{sinc}(t)\, e^{-pt}\, dt = \int_0^{\infty} \frac{\sin t}{t} e^{-pt}\, dt$

En dérivant sous le signe somme et en intégrant 2 fois par parties : $S'(p) = -\frac{1}{1+p^2} \Rightarrow S(p) = Arccotan\, p + Cte$

Lorsque p tend vers l'infini : $S(\infty) = 0 \Rightarrow Cte = 0 \Rightarrow S(p) = Arccotan\, p$

En particulier, pour p = 0 : $S(0) = \int_0^{\infty} \frac{\sin t}{t}\, dt = Arccotan\, 0 = \frac{\pi}{2} \Rightarrow \int_{-\infty}^{+\infty} \operatorname{sinc}(x)\, dx = \pi$

Transformée de Fourier de la fonction sinus cardinal.

On a : $S(f) = \int_{-\infty}^{+\infty} \operatorname{sinc}(t)\, e^{-j2\pi f t} dt = 2\int_0^{\infty} \frac{\sin t \cos 2\pi f t}{t}\, dt = \int_0^{\infty} \frac{\sin(1-2\pi f)t}{t}\, dt + \int_0^{\infty} \frac{\sin(1+2\pi f)t}{t}\, dt$

De l'égalité : $\int_0^{\infty} \frac{\sin t}{t}\, dt = \frac{\pi}{2}$, on déduit : $\int_0^{\infty} \frac{\sin \lambda t}{t}\, dt = \begin{cases} \pi/2 & \text{si } \lambda > 0 \\ 0 & \text{si } \lambda = 0 \\ -\pi/2 & \text{si } \lambda < 0 \end{cases}$

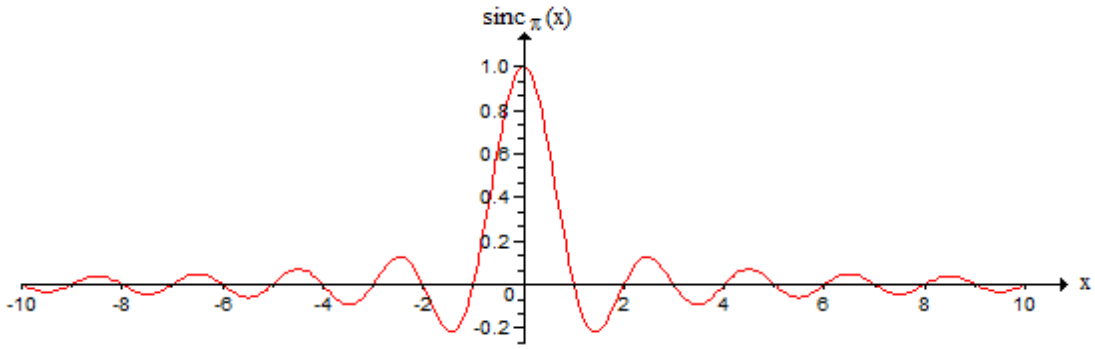
Par suite : $S(f) = \begin{cases} \pi & \text{si } |f| < \frac{1}{2\pi} \\ \pi/2 & \text{si } |f| = \frac{1}{2\pi} \\ 0 & \text{si } |f| > \frac{1}{2\pi} \end{cases} = \pi\, rect_{1/\pi}(f)$

Mathématique & Informatique

La fonction sinus cardinal normalisée

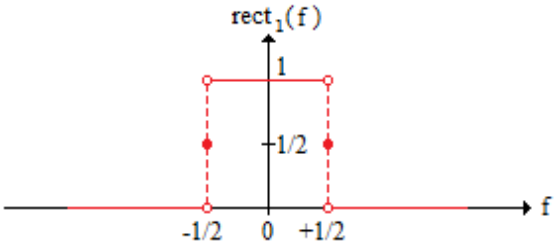
Elle est définie par : $y = \mathrm{sinc}_\pi(x) = \dfrac{\sin(\pi x)}{\pi x} = \mathrm{sinc}(\pi x)$,d'où la valeur de l'intégrale : $\int_{-\infty}^{+\infty} \mathrm{sinc}_\pi(x)\,dx = 1$

Le graphe (-10 ≤ x ≤ +10) s'obtient en écrivant : *x = linspace(-10,+10,300) ; plot2d(x,sinc(%pi*x),5)*



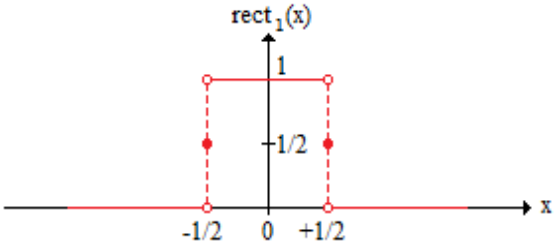
La transformée de **Fourier** de la fonction sinus cardinal normalisée est la fonction porte normalisée :

$$rect_1(f) = \begin{cases} 1 & \text{si } |f| < 1/2 \\ 1/2 & \text{si } |f| = 1/2 \\ 0 & \text{si } |f| > 1/2 \end{cases}$$

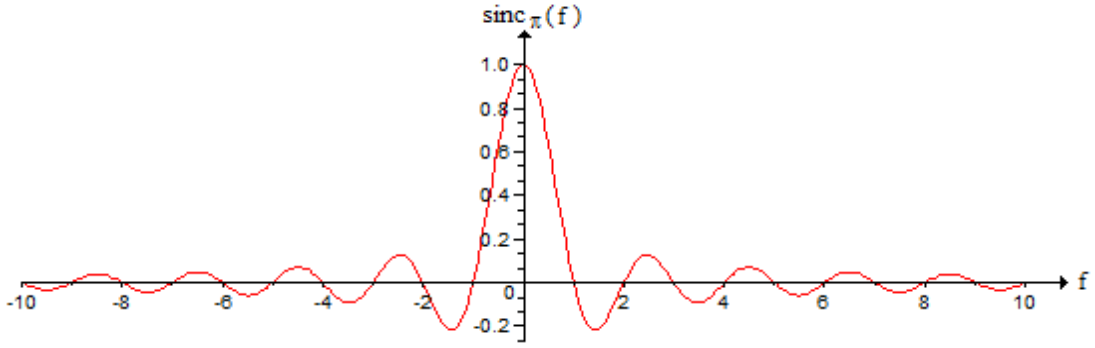


Inversement, soit la fonction porte normalisée définie par :

$$rect_1(x) = \begin{cases} 1 & \text{si } |x| < 1/2 \\ 1/2 & \text{si } |x| = 1/2 \\ 0 & \text{si } |x| > 1/2 \end{cases}$$



La transformée de **Fourier** de la fonction porte normalisée est la fonction sinus cardinal normalisée :



Mathématique & Informatique

La fonction sinus cardinal 3D

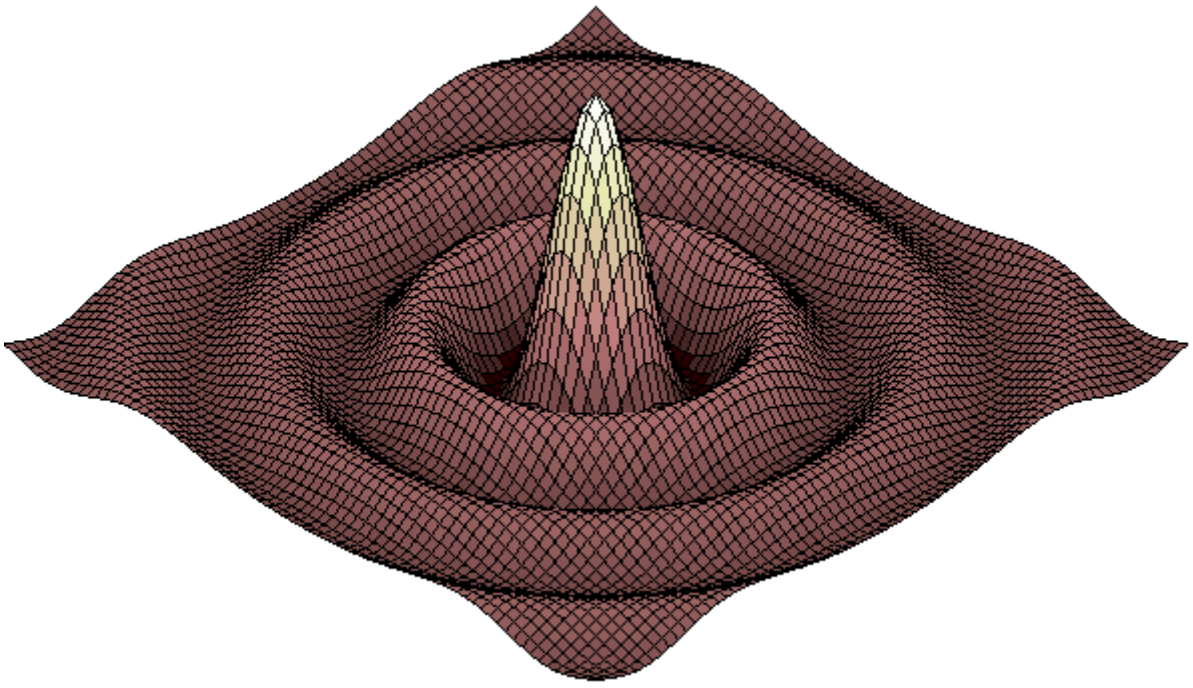
$$z = \mathrm{sinc}_{3d}(x,y) = \frac{\sin\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}} = \mathrm{sinc}\left(\sqrt{x^2+y^2}\right)$$

On a :

$$\frac{\sin\left(\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}} = 1 + \sum_{n=1}^{\infty} (-1)^n \frac{(x^2+y^2)^n}{(2n+1)!} \quad \Rightarrow \quad \lim_{x \to 0\,,\ y \to 0} z(x,y) = 1$$

Le prolongement z(0,0) = sinc(0) = 1 assure la continuité de la fonction sinc_{3d} sur $\mathbb{R}^2$.
Programmons une représentation graphique de cette surface dans l'espace :

```
function z = sinc3d(x,y)
if x <> 0 | y <> 0 then
z = sin(sqrt(x^2 + y^2))/sqrt(x^2 + y^2)
else
z = 1
end
endfunction
xset("colormap", pinkcolormap(32)) ;
x = linspace(-18,+18,73) ;
y = linspace(-18,+18,73) ;
fplot3d1(x, y, sinc3d, theta=45, alpha=87, flag=[1,2,0])
```

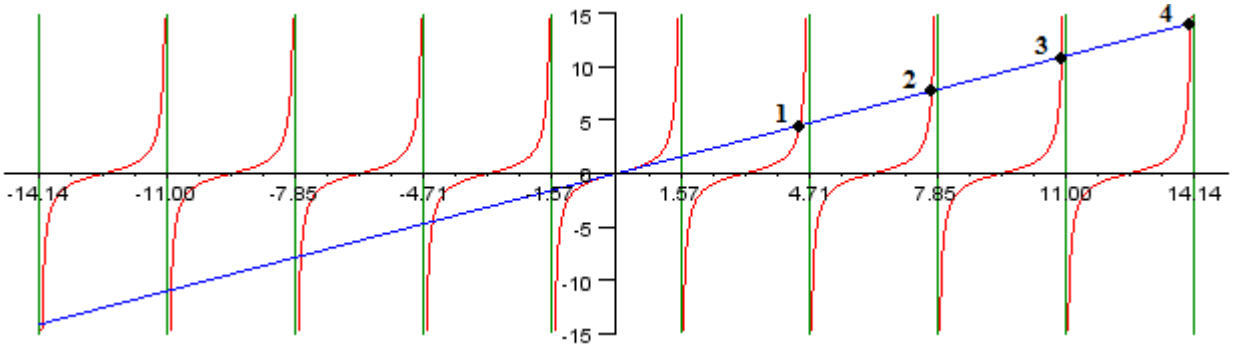


Mathématique & Informatique

Les extremums de la fonction $z(x,y) = \dfrac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$ vérifient : $\dfrac{\partial z}{\partial x} = \dfrac{\partial z}{\partial y} = 0 \Rightarrow \tan(\sqrt{x^2+y^2}) = \sqrt{x^2+y^2}$

Les lieux géométriques de ces points sont des cercles de rayon r_k , situés dans le plan d'équation $z = z_k = \sin(r_k)/r_k$ et centrés en $(0, 0, z_k)$, où r_k est la k^e racine positive de l'équation tan(r) = r dans l'intervalle $]-\pi/2 + k\pi , +\pi/2 + k\pi[$.
Effectuons une représentation graphique de l'équation tan(r) = r pour k = 1, 2, 3, 4 :

```
r = linspace(-%pi/2 + 0.068,+%pi/2 - 0.068,300) ;
a = gca() ; a.x_location = "origin" ; a.y_location = "origin" ;
for k = -4:4, plot2d(r + k*%pi, tan(r), 5), end
for k = -9:2:9, plot2d([k*%pi/2,k*%pi/2], [-15,+15], 13), end
plot2d([-9*%pi/2,+9*%pi/2], [-9*%pi/2,+9*%pi/2], 2, nax=[3,10,0,7]) ;
```

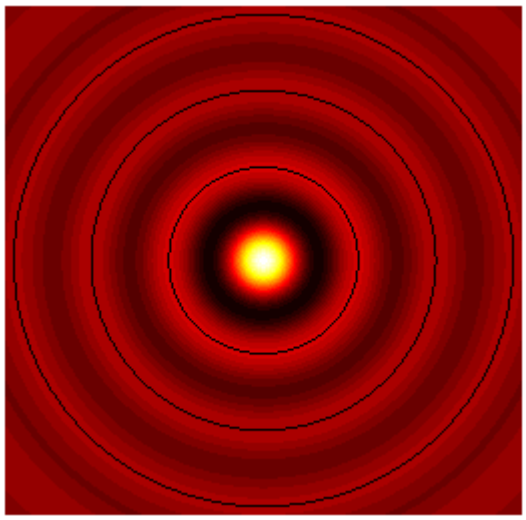


On peut calculer les racines r_k avec la fonction Scilab *fsolve*, à condition de partir de la bonne valeur initiale r_0.
Par exemple : $r_k = fsolve(r_0, f)$ avec $r_0 = \pi/2 + (k - 1)\pi + \varepsilon$ et f(r) = tan(r) - r
Une autre solution consiste à utiliser l'équivalence : $\tan(r) = r \Leftrightarrow r = \text{Arc tan}(r)$. La dérivée de la fonction Arc tan(r) étant égale à $1/(1+r^2)$, donc strictement inférieure à 1, le point fixe r_k est la limite de la suite récurrente définie par : $r_0 = k\pi$, $r_{n+1} = k\pi + \text{Arc tan}(r_n)$. Calculons r_1 à r_{10} avec une précision de 10^{-10} :

```
eps = 1e-10 ;
for k = 1:10
  r0 = k*%pi ;
  delta = eps ;
  while delta >= eps
    rk = k*%pi + atan(r0) ;
    delta = abs(rk - r0) ;
    r0 = rk ;
  end
  printf('\n r%i = %.10f ',k,rk)
end
```

r1 = 4.4934094579
r2 = 7.7252518369
r3 = 10.9041216594
r4 = 14.0661939128
r5 = 17.2207552719
r6 = 20.3713029593
r7 = 23.5194524987
r8 = 26.6660542588
r9 = 29.8115987909
r10 = 32.9563890398

Projection 2D obtenue avec *Sfgrayplot*



Les rayons r_k de rang impair $(r_1, r_3, r_5, r_7, r_9)$ correspondent à des minima de la surface, alors que les rayons r_k de rang pair $(r_2, r_4, r_6, r_8, r_{10})$ correspondent à des maxima. Le rayon $r_k \sim (2k+1)\pi/2$ quand l'indice $k \to +\infty$

Mathématique & Informatique

FRACTIONS CONTINUES

Conversion des fractions ordinaires en fractions continues

Le développement en fraction continue d'un nombre rationnel p/q (p et q étant 2 entiers relatifs) est fini :

$$\frac{p}{q} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ddots + \cfrac{1}{a_n}}} = \left[a_0, a_1, a_2, \cdots, a_n\right]$$

Le premier élément a_0 est la partie entière du rapport p/q : $a_0 \geq 0$ si $p/q \geq 0$ ou $a_0 < 0$ si $p/q < 0$
Les termes $a_i > 0$ ($i \neq 0$) sont les quotients entiers qui apparaissent dans les divisions successives lors de la recherche du pgcd de p et q par l'algorithme d'**Euclide** ; ils sont uniques et appelés quotients incomplets de la fraction continue.
Ecrivons une fonction Scilab *fraction_continue* qui prend en paramètre d'entrée une fraction ordinaire fo constituée d'un nombre quelconque de chiffres, et qui retourne en paramètre de sortie la fraction continue fc équivalente à fo.
Cette fonction est intégrée dans le module *Scilab-Fractions_continues*.

```
function fc = fraction_continue(fo)
  fc = list()
  u = fo(1) ; v = fo(2)
  while ~NUL(v)
    [q,r] = PARTIE_ENTIERE(u,v)
    fc($+1) = q
    u = v ; v = r
  end
endfunction
```

Exemple

```
exec('Scilab-Fractions_continues.sce',-1)
p = [2,1,0,7,2,0,1,0,1,2,3,4,5,0,9,8,5,7,7,6,5,4,3,2,1] ;
q = [5,3,6,5,4,8,9,4,1,5,6,0,9,1,1,5,5,8,1,2,7] ;
fo = list([1,p],[1,q]) ;
fc = fraction_continue(fo) ;
printf('\n '), ECRIRE(fo)
printf('\n '), FC_ECRIRE(fc)
```

$$\frac{21072010123450985776432 1}{53654894156091155812 7}$$

$$\Big| \Big|$$

[3927,3,10,2,4,9,1,72,15,7,3,2,1,1,8,4,2,4,1,1,1,2,1,3,1,8,2,1,1,3,1,1,2,2,2,1,6,1,2,1,4,2,2,10]

EXEMPLE

Développement en fraction continue des nombres de **Bernoulli** non nuls de B0 à B50.
Le nombre de Bernoulli B_k est le terme constant du polynôme de Bernoulli $P_k(x)$ dont les coefficients rationnels sont fournis par la fonction Scilab *P_BERNOULLI*. Les nombres de Bernoulli d'indice impair sont nuls à partir de B_3.

```
exec('Scilab-Fractions_continues.sce',-1)
exec('Scilab-Polynomes.sce',-1)
for k = [0,1,2:2:50]
  Pk = P_BERNOULLI(k) ;
  Bk = Pk($) ;
  fc = fraction_continue(Bk) ;
  printf('\n B%i = ',k)
  printf('\n '), ECRIRE(Bk)
  printf('\n '), FC_ECRIRE(fc)
end
```

La liste des fractions continues recherchées s'affiche à la Console :

```
B0 = 1 = [1]
B1 = - 1 / 2 = [-1, 2]
B2 = 1 / 6 = [0, 6]
B4 = - 1 / 30 = [-1, 1, 29]
B6 = 1 / 42 = [0, 42]
B8 = - 1 / 30 = [-1, 1, 29]
B10 = 5 / 66 = [0, 13, 5]
B12 = - 691 / 2730 = [-1, 1, 2, 1, 19, 3, 11]
B14 = 7 / 6 = [1, 6]
B16 = - 3617 / 510 = [-8, 1, 9, 1, 5, 1, 2, 2]
B18 = 43867 / 798 = [54, 1, 33, 1, 2, 3, 2]
B20 = - 174611 / 330 = [-530, 1, 7, 20, 2]
B22 = 854513 / 138 = [6192, 8, 8, 2]
B24 = - 236364091 / 2730 = [-86581, 1, 2, 1, 19, 3, 11]
B26 = 8553103 / 6 = [1425517, 6]
B28 = - 23749461029 / 870 = [-27298232, 1, 13, 1, 2, 1, 14]
B30 = 8615841276005 / 14322 = [601580873, 1, 9, 15, 2, 7, 6]
B32 = - 7709321041217 / 510 = [-15116315768, 1, 9, 1, 5, 1, 2, 2]
B34 = 2577687858367 / 6 = [429614643061, 6]
B36 = - 26315271553053477373 / 1919190 = [-13711655205089, 1, 2, 197, 1, 1, 1, 44, 4, 1, 4]
B38 = 2929993913841559 / 6 = [488332318973593, 6]
B40 = - 261082718496449122051 / 13530 = [-19296579341940069, 1, 5, 1, 2, 1, 2, 11, 16]
B42 = 1520097643918070802691 / 1806 = [841693047573682615, 1806]
B44 = - 27833269579301024235023 / 690 = [-40338071854059455414, 1, 12, 53]
B46 = 596451111593912163277961 / 282 = [2115074863808199160560, 6, 1, 7, 5]
B48 = - 5609403368997817686249127547 / 46410 = [-120866265222965259346028, 1, 2, 4, 1, 6, 7, 1, 7, 1, 5]
B50 = 4950572052410796482124775 25 / 66 = [75008667460769643668557 20, 13, 5]
```

Conversion des fractions continues en fractions ordinaires

Soit fc la liste des éléments a_i d'une fraction continue. La fonction Scilab *fraction_ordinaire* calcule la fraction ordinaire fo équivalente à fc.

```
function fo = fraction_ordinaire(fc)
  p = ENTIER(1) ; q = ENTIER(0)
  for i = length(fc):-1:1
    u = p
    v = MULTIPLICATION(fc(i),p)
    p = ADDITION(v,q)
    q = u
  end
  fo = list(p,q)
endfunction
```

Exemple

Convertir la fraction continue [1, 2, 3, 4, 5, 6, ... , 47, 48, 49, 50] en nombre rationnel.
Le programme Scilab est le suivant :

```
exec('Scilab-Fractions_continues.sce',-1)
fc = list() ;
for i = 1:50
  fc($+1) = ENTIER(i) ;
end
fo = fraction_ordinaire(fc) ;
printf('\n '), FC_ECRIRE(fc)
printf('\n '), ECRIRE(fo)
```

On obtient la valeur recherchée :

[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50]

||

$$\frac{67958941007729162096557840697158925276672441401340713037951251}{47420026812410519916441151369724483594838921505749248182936966650}$$

Fractions réduites

Par définition :

$$\frac{p_k}{q_k} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ddots + \cfrac{1}{a_k}}} = [a_0, a_1, a_2, \cdots, a_k] \quad , \quad 0 \le k \le n$$

En particulier, pour un nombre rationnel p/q :

$$\frac{p_0}{q_0} = E\left(\frac{p}{q}\right) \text{ et } \frac{p_n}{q_n} = \frac{p}{q}$$

Relations de récurrence

$$\begin{cases} p_k = a_k p_{k-1} + p_{k-2} & , k \ge 1 \\ q_k = a_k q_{k-1} + q_{k-2} & , k \ge 1 \end{cases}$$

avec : $p_{-1} = 1$; $q_{-1} = 0$; $p_0 = a_0$; $q_0 = 1$

Propriétés

$$\begin{cases} p_k q_{k-1} - p_{k-1} q_k = (-1)^{k-1} & , k \ge 1 \\ p_k q_{k-2} - p_{k-2} q_k = (-1)^k a_k & , k \ge 2 \end{cases}$$

Programme de calcul

Supposons que la variable fc contienne la liste des n+1 termes a_i d'une fraction continue.
La fonction Scilab *reduite_ordinaire* retourne la valeur de la fraction ordinaire fr = p_k/q_k ($0 \le k \le n$) équivalente à la réduite d'ordre k de fc.

```
function fr = reduite_ordinaire(fc,k)
  if k >= length(fc) then
    fr = list() ; return
  end
  pim2 = [1,1] ; qim2 = [0,0]
  pim1 = fc(1) ; qim1 = [1,1]
  for i = 1:k
    pi = ADDITION(MULTIPLICATION(fc(i+1),pim1),pim2)
    qi = ADDITION(MULTIPLICATION(fc(i+1),qim1),qim2)
    pim2 = pim1 ; qim2 = qim1
    pim1 = pi ; qim1 = qi
  end
  fr = list(pim1,qim1)
endfunction
```

Mathématique & Informatique

EXEMPLE 1

L'exponentielle e est un nombre transcendant, son développement en fraction continue est infini et non périodique.
Euler obtint l'expression suivante :

$$e = [2,1,2,1,1,4,1,\ldots,1,2p,1,\ldots]$$

On en déduit facilement un programme d'approximation de e avec d = 100 décimales exactes :

```
d = 100 ;
pkm2 = 1 ; qkm2 = 0 ;
pkm1 = 2 ; qkm1 = 1 ;
fc = list([1,2]) ;
k = 0 ; a = 0 ;
while length(qkm1) <= ceil(d/2)
  k = k + 1 ;
  if modulo(k+1,3) == 0 then
    a = addition(a,2) ;
    ak = a ;
  else
    ak = 1 ;
  end
  fc($+1) = [1,ak] ;
  pk = addition(multiplication(ak,pkm1),pkm2) ;
  qk = addition(multiplication(ak,qkm1),qkm2) ;
  pkm2 = pkm1 ; qkm2 = qkm1 ;
  pkm1 = pk ; qkm1 = qk ;
end
e = list([1,pk],[1,qk]) ;
printf('\n e = '), FC_ECRIRE(fc)
printf('\n e = '), ECRIRE(e)
printf('\n e = '), ECRIRE(DECIMALE(e,d))
```

Développement en fraction continue de l'exponentielle :

e =
[2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1, 1, 10, 1, 1, 12, 1, 1, 14, 1, 1, 16, 1, 1, 18, 1, 1, 20, 1, 1, 22, 1, 1, 24, 1, 1, 26, 1, 1, 28, 1, 1, 30, 1, 1, 32, 1, 1, 34, 1, 1, 36, 1, 1, 38, 1, 1, 40, 1, 1, 42, 1, 1, 44, 1, 1, 46, 1, 1, 48, 1, 1, 50, 1, 1, 52, 1, 1, 54, 1, 1, 56, 1, 1, 58, 1, 1, 60,...]

Approximation rationnelle de e :

$$e = \frac{3122681731663446268852622205448790639106028937172031}{1148770410400620419199349985370106606649752990544640}$$

Approximation décimale de e à 10^{-100} près :

e = 2.7182818284590452353602874713526624977572470936999595749669676277240766303535475945713821785251664274

Remarque
D'une manière analogue on peut développer le nombre π en fraction continue généralisée (formule de **Brouncker**), mais la convergence est lente et le temps de calcul excessif si l'on souhaite obtenir une grande précision.

EXEMPLE 2

A partir des approximations décimales de π et γ déjà obtenues par d'autres méthodes, calculons les premiers termes du développement en fraction continue de ces 2 constantes.

Nombre π

```
Pi = list([1,3,1,4,1,5,9,2,6,5,3,5,8,9,7,9,3,2,3,8,4,6,2,6,4,3,3,8,3,2,7,9,5,0,2,8,8,4,1,9,7,1,6,9,3,9,9,3,7,5,1,0,...
           5,8,2,0,9,7,4,9,4,4,5,9,2,3,0,7,8,1,6,4,0,6,2,8,6,2,0,8,9,9,8,6,2,8,0,3,4,8,2,5,3,4,2,1,1,7,0,6,7,9],...
           [1,1,zeros(1,100)]) ;
fc = fraction_continue(Pi) ;
printf('\n Pi = '), FC_ECRIRE(fc)
```

Soit :

π =
[3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, 2, 2, 2, 1, 84, 2, 1, 1, 15, 3, 13, 1, 4, 2, 6, 6, 99, 1, 2, 2, 6, 3, 5, 1, 1, 6, 8, 1, 7, 1, 2, 3, 7, 1, 2, 1, 1, 12, 1, 1, 1, 3, 1, 1, 8, 1, 1, 2, 1, 6, 1, 1, 5, 2, 2, 3, 1, 2, 4, 4, 16, 1, 161, 45, 1, 22, 1, 2, 2, 1, 4, 1, 2, 24, 1, 2, 1, 3, 1, 2, 4, 2, 5, 1, 1, 1, 1, 1, 1, 1, 2, 9, 3, 1, 1, 16, 1, 4, 3, 1, 11, 3, 6, 8, 1, 1, 1, 8, 3, 2, 1, 1, 5, 1, 1, 1, 7, 33, 2, 2, 1, 5, 2, 4, 1, 1, 1, 2, 1, 1, 1, 22, 18, 16, 11, 1, 1, 1, 1, 2, 2, 10, 1, 1, 21, 3, 2, 4, 3, 3, 1, 1, 1, 1, 11, 1, 7, 1, 43, 2, 1, 1, 2, 5, 3, 1, 4, 1, 78, 2, 1, 21, 3, 9, 4, 11, 2, 1, 2, 2, 2, 1, 4, ...]

Brève histoire du nombre π : les Hébreux et les Babyloniens utilisaient la valeur 3 (π ≈ [3]), **Archimède** trouva le majorant 22/7 (π ≈ [3,7]), et Adrien **Metius** (XVII^{e} siècle) découvrit la fraction 355/113 (π ≈ [3,7,15,1]) qui donne les 6 premières décimales de π.

Nombre γ

```
Gamma = list([1,0,5,7,7,2,1,5,6,6,4,9,0,1,5,3,2,8,6,0,6,0,6,5,1,2,0,9,0,0,8,2,4,0,2,4,3,1,0,4,2,1,5,9,3,3,5,9,3,9,9,2,...
              3,5,9,8,8,0,5,7,6,7,2,3,4,8,8,4,8,6,7,7,2,6,7,7,7,6,6,4,6,7,0,9,3,6,9,4,7,0,6,3,2,9,1,7,4,6,7,4,9,5],...
              [1,1,zeros(1,100)]) ;
fc = fraction_continue(Gamma) ;
printf('\n Gamma = '), FC_ECRIRE(fc)
```

Soit :

γ =
[0, 1, 1, 2, 1, 2, 1, 4, 3, 13, 5, 1, 1, 8, 1, 2, 4, 1, 1, 40, 1, 11, 3, 7, 1, 7, 1, 1, 5, 1, 49, 4, 1, 65, 1, 4, 7, 11, 1, 399, 2, 1, 3, 2, 1, 2, 1, 5, 3, 2, 1, 10, 1, 1, 1, 1, 2, 1, 1, 3, 1, 4, 1, 1, 2, 5, 1, 3, 6, 2, 1, 2, 1, 1, 1, 2, 1, 3, 16, 8, 1, 1, 2, 16, 6, 1, 2, 2, 1, 7, 2, 1, 1, 1, 3, 1, 2, 1, 2, 13, 5, 1, 1, 1, 6, 1, 2, 1, 1, 16, 1, 7, 1, 2, 1, 10, 1, 20, 1, 1, 1, 1, 1, 1, 6, 5, 1, 5, 6, 2, 2, 3, 15, 1, 1, 1, 4, 2, 3, 3, 2, 1, 13, 1, 1, 1877, 4, 3, 2, 11, 1, 1, 1, 2, 4, 1, 2, 12, 1, 7, 1, 245, 23, 1, 34, 1, 1, 3, 7, 1, 1, 9, 2, 1, 1, 13, 48, 1, 1, 1, 1, 16, 1, 10, 1, 4, 2, 1, 8, 3, 1, 1, 1, 1, 1, 1, 1, 16, 1, 3, 3, 22, 4, ...]

La réduite d'ordre 8 est égale à [0,1,1,2,1,2,1,4,3] = $\frac{228}{395}$ ≈ 0.577215 ; elle fournit 6 décimales exactes de γ.

Remarque
Un nombre est transcendant s'il n'est solution d'aucune équation polynomiale à coefficients entiers.
Il est généralement difficile d'établir la transcendance d'un nombre réel. Pour π la preuve a été apportée par **Lindemann** en 1882 à partir de l'identité $e^{i\pi} = -1$. Il reste le cas de γ dont on ignore encore s'il est irrationnel.

Mathématique & Informatique

Développement en fraction continue des nombres quadratiques

Un nombre quadratique x est défini par :

$$x = a + b\sqrt{n}$$

avec a , $b \in \mathbb{Q}$ et $n \in \mathbb{N}$

On suppose b non nul et n non carré parfait.
x est irrationnel, donc son développement en fraction continue est infini.
Comme x est solution de l'équation du second degré $x^2 - 2ax + (a^2 - b^2n) = 0$ et que le développement en fraction continue de la solution d'une équation du second degré à coefficients rationnels est périodique à partir d'un certain rang r d'après le théorème de **Lagrange**, on a :

$$x = \left[a_0, a_1, \cdots, a_{r-1}, a_r, \cdots, a_{r+T-1}, a_r, \cdots, a_{r+T-1}, \cdots\right]$$

NB Le rang r et la période T peuvent être de grands entiers

Précisons le principe de calcul des a_i
En exprimant $a = \alpha_1/\alpha_2$ et $b = \beta_1/\beta_2$, puis en réduisant au même dénominateur, le réel x se met sous la forme :

$$x = \frac{A + \sqrt{N}}{B}$$

avec A $\in \mathbb{Z}$, B $\in \mathbb{Z}^*$ et N $\in \mathbb{N}^*$

Soit M la partie entière de la racine carrée de N, on écrit :

$$x = \frac{A + M - M + \sqrt{N}}{B} = \frac{(A + M) + (\sqrt{N} - M)}{B}$$

Effectuons la division euclidienne de A + M par B : A + M = BQ + R
On obtient :

$$x = Q + \frac{\sqrt{N} - (M - R)}{B} = Q + \frac{1}{x'}$$

où Q est la partie entière de x

Appliquons à x' l'identité remarquable : $(u - v)(u + v) = u^2 - v^2$

$$x' = \frac{B}{\sqrt{N} - (M - R)} = \frac{B(M - R) + B\sqrt{N}}{N - (M - R)^2} = \frac{A' + \sqrt{N'}}{B'}$$

x' est un nouveau nombre quadratique que l'on peut décomposer selon la même méthode.
En répétant le processus de décomposition, les quotients Q successifs sont les valeurs des éléments a_i du développement en fraction continue de x.

Programmons une fonction Scilab *fraction_quadratique* qui fournit la fraction continue réduite fc d'ordre *k* d'un nombre quadratique défini par les 3 paramètres *a*, *b*, *n*.

```
function fc = fraction_quadratique(a,b,n,k)
  fc = list()
  A = MULTIPLICATION(a(1),b(2))
  B = MULTIPLICATION(a(2),b(2))
  C = MULTIPLICATION(a(2),b(1))
  N = MULTIPLICATION(n,MULTIPLICATION(C,C))
  if SIGNE(N) <= 0 then, return, end
  if SIGNE(C) < 0 then, A = MOINS(A), B = MOINS(B), end
  for i = 0:k
    if i <= 1 then, M = [1,intsqrt(VALEUR(N))], end
    A = ADDITION(A,M)
    [Q,R] = PARTIE_ENTIERE(A,B)
    fc($+1) = Q
    if i == 0 then
      U = SOUSTRACTION(M,R)
      V = MULTIPLICATION(B,U)
      W = SOUSTRACTION(N,MULTIPLICATION(U,U))
      if SIGNE(B) < 0 then, V = MOINS(V), W = MOINS(W), end
      P = PGCD(PGCD(V,W),B)
      [C,R] = DIVISION(B,P)
      [A,R] = DIVISION(V,P)
      [B,R] = DIVISION(W,P)
      N = MULTIPLICATION(N,MULTIPLICATION(C,C))
    else
      U = SOUSTRACTION(M,R)
      W = SOUSTRACTION(N,MULTIPLICATION(U,U))
      A = U
      [B,R] = DIVISION(W,B)
    end
  end
endfunction
```

En faisant *fo = fraction_ordinaire(fc)* on obtient un nombre rationnel fo = p/q qui est une approximation de *x*, d'autant plus précise que fc comporte un grand nombre d'éléments :

$$x = a + b\sqrt{n} \approx \frac{p}{q}$$

Il reste à évaluer l'erreur commise en remplaçant *x* par p/q .
On montre que le nombre rationnel p_k/q_k équivalent à la réduite d'ordre k est alternativement inférieur et supérieur à *x* lorsque *k* varie de 0 à l'infini. Donc, l'intervalle d'extrémités p_k/q_k et p_{k+1}/q_{k+1} contient x. Il s'ensuit que p_k/q_k s'approche de *x* avec un écart inférieur ou égal à la largeur de cet intervalle, ce qui se traduit par l'estimation :

$$\left| x - \frac{p_k}{q_k} \right| \leq \left| \frac{p_{k+1}}{q_{k+1}} - \frac{p_k}{q_k} \right| = \frac{1}{q_k\, q_{k+1}} < \frac{1}{{q_k}^2}$$

Ajoutons que la fraction p_k/q_k est irréductible et réalise la meilleure approximation rationnelle de x pour un nombre de chiffres donné, ce qui signifie que toute autre fraction λ/μ au dénominateur inférieur ou égal à q_k ($\mu \leq q_k$) s'écarte plus du nombre x que la fraction p_k/q_k . Malgré que la convergence soit assez lente en général (il existe des méthodes plus rapides pour calculer une approximation de *x*), on peut atteindre la précision que l'on veut.

Mathématique & Informatique

EXEMPLE 1

Calculer le développement en fraction continue d'ordre 50 puis l'approximation rationnelle et décimale correspondante de chacun des 2 nombres quadratiques *x* et *y* ci-dessous :

$$x = \frac{3187947861}{900482} - \frac{637}{1959}\sqrt{47368695}$$

$$y = -\frac{3187947861}{900482} + \frac{637}{1959}\sqrt{47368695}$$

Programme Scliab :

```
exec('Scilab-Fractions_continues.sce',-1)
a = FRACTION(3187947861,900482) ;
b = FRACTION(-637,1959) ;
n = ENTIER(47368695) ;
k = 50 ;
fc = fraction_quadratique(a,b,n,k) ;
fo = fraction_ordinaire(fc) ;
x = DECIMALE(fo,2*length(fo(2))-4) ;
//---------------------------------------------
printf('\n x = '), FC_ECRIRE(fc)
printf('\n x = '), ECRIRE(fo)
printf('\n x = '), ECRIRE(x)
```

```
exec('Scilab-Fractions_continues.sce',-1)
a = FRACTION(-3187947861,900482) ;
b = FRACTION(637,1959) ;
n = ENTIER(47368695) ;
k = 50 ;
fc = fraction_quadratique(a,b,n,k) ;
fo = fraction_ordinaire(fc) ;
y = DECIMALE(fo,2*length(fo(2))-4) ;
//---------------------------------------------
printf('\n y = '), FC_ECRIRE(fc)
printf('\n y = '), ECRIRE(fo)
printf('\n y = '), ECRIRE(y)
```

Résultat :

$$x \approx [1302, 3, 6, 3, 3, 1, 1, 12, 2, 14, 4, 1, 9, 1, 10, 3, 10, 4, 10, 2, 1, 1, 1, 1, 2, ...$$
$$3, 362, 1, 2, 4, 1, 26, 4, 3, 2, 19, 4, 2, 4, 1, 1, 1, 47, 1, 1, 1, 749, 5, 4, 1, 2]$$

$$x \approx \frac{2184749282\,4303021750\,7992535587\,6367}{1677586918\,6289928300\,3904620426\,2}$$

$$x \approx 1302.31659\,4251812337\,3054734690\,3711025996\,1093403568\,3696423093\,72316$$

$$y \approx [-1303, 1, 2, 6, 3, 3, 1, 1, 12, 2, 14, 4, 1, 9, 1, 10, 3, 10, 4, 10, 2, 1, 1, 1, 1, ...$$
$$2, 3, 362, 1, 2, 4, 1, 26, 4, 3, 2, 19, 4, 2, 4, 1, 1, 1, 47, 1, 1, 1, 749, 5, 4, 1]$$

$$y \approx -\frac{7781304281\,4985672316\,1018449892\,631}{5974971305\,6286187057\,5743086663}$$

$$y \approx -1302.31659\,4251812337\,3054734690\,3711025996\,1093403568\,3696423093\,723$$

EXEMPLE 2

Vérifions que le développement en fraction continue de $\sqrt{\frac{73871}{50061}}$ est périodique.

Transformons tout d'abord la racine carrée du nombre rationnel en nombre quadratique :

$$\sqrt{\frac{73871}{50061}} = \frac{1}{50061}\sqrt{50061 \times 73871}$$

Utilisons ensuite la fonction *fraction_quadratique* :

```
a = FRACTION(0,1) ;
b = FRACTION(1,50061) ;
n = ENTIER(50061*73871) ;
k = 1000 ;
fc = fraction_quadratique(a,b,n,k) ;
printf('\n '), FC_ECRIRE(fc)
```

La réduite d'ordre 1000 du développement en fraction continue de $\sqrt{\frac{73871}{50061}}$ est :

[1, **4, 1, 1, 1, 10, 3, 12, 1, 10, 14, 1, 1, 5, 6, 1, 1, 1, 2, 2, 1, 17, 2, 6, 6, 1, 6, 2, 1, 2, 2, 1, 1, 1, 2, 48, 1, 3, 1, 1, 4, 4, 1, 335, 5, 1, 31, 2, 6, 45, 4, 11, 1, 1, 1, 1, 1, 2, 1, 7, 13, 3, 1, 1, 1, 2, 2, 1, 15, 4, 1, 7, 1, 1, 5, 17, 1, 67, 1, 3, 3, 11, 2, 2, 3, 2, 1, 2, 6, 3, 1, 2, 3, 2, 1, 8, 1, 8, 3, 10, 1, 2, 1, 5, 1, 3, 8, 1, 1, 16, 3, 4, 2, 1, 1, 2, 2, 7, 1, 9, 3, 1, 1, 1, 2, 1, 2, 4, 13, 5, 1, 2, 6, 1, 1, 1, 127, 1, 1, 28, 1, 1, 1, 2, 3, 5, 8, 1, 22, 1, 2, 1, 2, 3, 4, 1, 4, 1, 5, 1, 2, 1, 10, 1, 2, 4, 1, 1, 1, 1, 110, 1, 1, 1, 5527, 1, 1, 1, 110, 1, 1, 1, 1, 4, 2, 1, 10, 1, 2, 1, 5, 1, 4, 1, 4, 3, 2, 1, 2, 1, 22, 1, 8, 5, 3, 2, 1, 1, 1, 28, 1, 1, 127, 1, 1, 1, 6, 2, 1, 5, 13, 4, 2, 1, 2, 1, 1, 1, 3, 9, 1, 7, 2, 2, 1, 1, 2, 4, 3, 16, 1, 1, 8, 3, 1, 5, 1, 2, 1, 10, 3, 8, 1, 8, 1, 2, 3, 2, 1, 3, 6, 2, 1, 2, 3, 2, 2, 11, 3, 3, 1, 67, 1, 17, 5, 1, 1, 7, 1, 4, 15, 1, 2, 2, 1, 1, 1, 3, 13, 7, 1, 2, 1, 1, 1, 1, 1, 11, 4, 45, 6, 2, 31, 1, 5, 335, 1, 4, 4, 1, 1, 3, 1, 48, 2, 1, 1, 1, 2, 2, 1, 2, 6, 1, 6, 6, 2, 17, 1, 2, 2, 1, 1, 1, 6, 5, 1, 1, 14, 10, 1, 12, 3, 10, 1, 1, 1, 4, 2**, 4, 1, 1, 1, 10, 3, 12, 1, 10, 14, 1, 1, 5, 6, 1, 1, 1, 2, 2, 1, 17, 2, 6, 6, 1, 6, 2, 1, 2, 2, 1, 1, 1, 2, 48, 1, 3, 1, 1, 4, 4, 1, 335, 5, 1, 31, 2, 6, 45, 4, 11, 1, 1, 1, 1, 1, 2, 1, 7, 13, 3, 1, 1, 1, 2, 2, 1, 15, 4, 1, 7, 1, 1, 5, 17, 1, 67, 1, 3, 3, 11, 2, 2, 3, 2, 1, 2, 6, 3, 1, 2, 3, 2, 1, 8, 1, 8, 3, 10, 1, 2, 1, 5, 1, 3, 8, 1, 1, 16, 3, 4, 2, 1, 1, 2, 2, 7, 1, 9, 3, 1, 1, 1, 2, 1, 2, 4, 13, 5, 1, 2, 6, 1, 1, 1, 127, 1, 1, 28, 1, 1, 1, 2, 3, 5, 8, 1, 22, 1, 2, 1, 2, 3, 4, 1, 4, 1, 5, 1, 2, 1, 10, 1, 2, 4, 1, 1, 1, 1, 110, 1, 1, 1, 5527, 1, 1, 1, 110, 1, 1, 1, 1, 4, 2, 1, 10, 1, 2, 1, 5, 1, 4, 1, 4, 3, 2, 1, 2, 1, 22, 1, 8, 5, 3, 2, 1, 1, 1, 28, 1, 1, 127, 1, 1, 1, 6, 2, 1, 5, 13, 4, 2, 1, 2, 1, 1, 1, 3, 9, 1, 7, 2, 2, 1, 1, 2, 4, 3, 16, 1, 1, 8, 3, 1, 5, 1, 2, 1, 10, 3, 8, 1, 8, 1, 2, 3, 2, 1, 3, 6, 2, 1, 2, 3, 2, 2, 11, 3, 3, 1, 67, 1, 17, 5, 1, 1, 7, 1, 4, 15, 1, 2, 2, 1, 1, 1, 3, 13, 7, 1, 2, 1, 1, 1, 1, 1, 11, 4, 45, 6, 2, 31, 1, 5]

Le développement est périodique à partir du rang 1, la période est de 348 éléments, et le dernier élément de la période (2) est égal au double du premier élément du développement (1) mais sans être la première occurrence de cette propriété comme cela est toujours le cas lorsqu'on effectue le développement en fraction continue de la racine carrée d'un nombre entier. De plus, les éléments d'une période forment un palindrome, c'est-à-dire que la suite des nombres (4, 1, 1, 1, 10, 3, ... , 5527, ... , 3, 10, 1, 1, 1, 4) peut se lire indifféremment de la gauche vers la droite ou de la droite vers la gauche, autrement dit la suite est symétrique par rapport à sa valeur médiane (5527).

EXEMPLE 3

Soit p/q un nombre rationnel supérieur à 1 (p > q > 0). Le problème consiste à trouver 2 entiers X et Y tels que :

$$X^2 - \frac{p}{q} Y^2 = 1$$

Posons :

$$\omega = \sqrt{\frac{p}{q}} = \frac{1}{q} \sqrt{p\,q}$$

Le développement en fraction continue du nombre quadratique ω est périodique à partir du rang 1 :

$$\omega = [a_0, a_1, \cdots, a_T, a_1, \cdots, a_T, \cdots]$$

Soit $X/Y = p_{\lambda T - 1}/q_{\lambda T - 1}$ la fraction ordinaire équivalente à la réduite d'ordre λT-1 si T est paire et $X/Y = p_{2\lambda T - 1}/q_{2\lambda T - 1}$ la fraction ordinaire équivalente à la réduite d'ordre 2λT-1 si T est impaire (λ est un entier naturel non nul).
Le couple d'entiers (X,Y) vérifie la relation : $X^2 - (p/q)Y^2 = 1$.
L'équation possède donc une infinité de solutions ; pour λ = 1, on obtient la plus petite d'entre elles.

Exemples simples :

- p = 5, q = 3 ⇒ ω = [1, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, ...] ⇒ T = 2, X = 4, Y = 3 ⇒ $4^2 - (5/3)3^2 = 1$
- p = 5, q = 2 ⇒ ω = [1, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 2, ...] ⇒ T = 3, X = 19, Y = 12 ⇒ $19^2 - (5/2)12^2 = 1$

Appelons *FERMAT* la fonction Scilab qui recherche le plus petit couple (X,Y).
Dans le programme, le développement en fraction continue de $(p/q)^{1/2}$ n'excède pas l'ordre k = 1000 afin de limiter le temps de calcul. Si k est insuffisamment grand pour obtenir X et Y, alors la fonction *FERMAT* retourne X = Y = ∅.
Le cas q = 1 correspond à l'équation de **Fermat** classique $X^2 - pY^2 = 1$ où p est un entier.

Application numérique

Trouver les 2 plus « petits » entiers X et Y vérifiant l'équation suivante : $X^2 - \frac{73871}{50061} Y^2 = 1$

On écrit :

```
p = 73871 , q = 50061
[X,Y] = FERMAT(p,q) ;
printf('\n X = '), ecrire(X)
printf('\n Y = '), ecrire(Y)
```

On obtient :

X = 31560988232690664545372258268485141351243643715547132431459932448535445218525101005658950542171690138859020453579423657589822391084816195809924203003049127386239635526873989430

Y = 25981448779038803900358813298139055003240266571931705102573924131054014695725876632222117682561305682103936106105849529881785131964782804640177753036542360856840180372690760253

Remarque
La solution de l'équation de **Fermat** $x^2 - Ny^2 = 1$ avec N = pq donne la solution de l'équation $X^2 - (p/q)Y^2 = a^2$.
Il suffit d'écrire : $x^2 - pqy^2 = 1 \Rightarrow (ax)^2 - (p/q)(aqy)^2 = a^2$ et de prendre X = ax et Y = aqy.

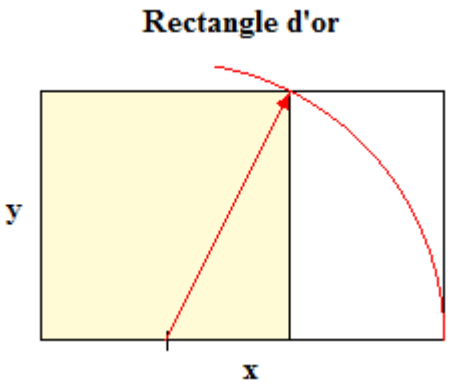
```
function [X,Y] = FERMAT(p,q)
  p = entier(p)
  q = entier(q)
  k = 1000
  T = 0
  A = 0
  B = q
  N = multiplication(p,q)
  M = intsqrt(N)
  fc = list()
  for i = 0:k
    [Q,R] = division(addition(A,M),B)
    fc($+1) = Q
    A = soustraction(M,R)
    C = soustraction(N,multiplication(A,A))
    [B,R] = division(C,B)
    if i == 0 then
      AP = A
      BP = B
    elseif isequal(list(A,B),list(AP,BP)) then
      T = i
      break
    end
  end
  printf('\n T = %i',T)
  if T == 0 then
    X = []
    Y = []
    return
  end
  for i = 1:T-1
    fc($+1) = fc(i+1)
  end
  if modulo(T,2) == 0 then
    K = T - 1
  else
    K = 2*T - 1
  end
  X = 1
  Y = 0
  for i = K+1:-1:1
    t = X
    X = multiplication(fc(i),X)
    X = addition(X,Y)
    Y = t
  end
endfunction
```

EXEMPLE 4

Le nombre d'or φ, censé donner des proportions harmonieuses, est tel que le rapport de 2 grandeurs x et y, de la plus grande x avec la plus petite y, est égal au rapport de leur somme x + y avec la plus grande x.

Ainsi :

$$\varphi = \frac{x}{y} = \frac{x+y}{x}$$



D'où :

$$\varphi^2 - \varphi - 1 = 0 \quad \Leftrightarrow \quad \varphi = 1 + \frac{1}{\varphi}$$

Donc :

$$\varphi = \frac{1+\sqrt{5}}{2} = [1,1,1,1,\ldots]$$

φ est un nombre algébrique irrationnel.
Déduisons de la formule précédente les 100 premières décimales de φ.
La fraction continue illlimitée constituée uniquement de 1 fournit des nombres rationnels p_k/q_k qui approchent φ avec une précision croissante. Pour obtenir d = 100 décimales de φ, on calcule p_k et q_k avec les relations de récurrence sur le numérateur et sur le dénominateur des fractions réduites et on arrête les itérations lorsque $q_k^2 \geq 10^{100}$, inégalité assurée dès que q_k possède plus de 50 chiffres.
Dans le cas présent : $p_k = p_{k-1} + p_{k-2}$ avec $p_{-1} = 1$ et $p_0 = 1$, et $q_k = q_{k-1} + q_{k-2}$ avec $q_{-1} = 0$ et $q_0 = 1$. Par conséquent p_k/q_k ($k \geq 0$) est égal au rapport f_{k+2}/f_{k+1} de 2 nombres de **Fibonacci** successifs, ce qui conduit au programme :

```
d = 100 ;
x = 1 ; y = 1 ;
while length(y) <= ceil(d/2)
   z = addition(x,y) ;
   y = x ; x = z ;
end
phi = list([1,x],[1,y]) ;
printf('\n phi = '), ECRIRE(phi)
phi = DECIMALE(phi,d) ;
printf('\n phi = '), ECRIRE(phi)
```

L'exécution de ce programme donne l'approximation rationnelle suivante du nombre d'or :

$$\varphi = \frac{168083057059453008835412295811648513482449585399521}{103881042195729914708510518382775401680142036775841}$$

et l'approximation décimale à 10^{-100} près :

φ = 1.6180339887498948482045868343656381177203091798057628621354486227052604628189024497072072041893911374

Fractions continues généralisées

On appelle ainsi une fraction continue dont les numérateurs ne sont pas des 1 mais des nombres quelconques b_i (réels, complexes) et les dénominateurs ne sont pas nécessairement des entiers mais des nombres quelconques a_i :

$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \ddots + \cfrac{b_i}{a_i + \ddots}}} = \left[a_0 ; \frac{b_1}{a_1}, \frac{b_2}{a_2}, \cdots, \frac{b_i}{a_i}, \cdots \right]$$

Exemple :

$$e = 2 + \cfrac{2}{2 + \cfrac{3}{3 + \cfrac{4}{4 + \cfrac{5}{5 + \cfrac{6}{6 + \cdots}}}}} = 2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \ddots + \cfrac{1}{1 + \cfrac{1}{2p + \cfrac{1}{1 + \cdots}}}}}}$$

Relations de récurrence permettant de calculer la réduite d'ordre k :

$$\frac{p_k}{q_k} = \left[a_0 ; \frac{b_1}{a_1}, \frac{b_2}{a_2}, \cdots, \frac{b_k}{a_k} \right]$$

$$\begin{cases} p_k = a_k p_{k-1} + b_k p_{k-2} & , k \geq 1 \\ q_k = a_k q_{k-1} + b_k q_{k-2} & , k \geq 1 \end{cases}$$

avec : $p_{-1} = 1 ; q_{-1} = 0 ; p_0 = a_0 ; q_0 = 1$

Propriétés des réduites :

$$\begin{cases} p_k q_{k-1} - p_{k-1} q_k = (-1)^{k-1} b_1 b_2 \cdots b_k & , k \geq 1 \\ p_k q_{k-2} - p_{k-2} q_k = (-1)^k b_1 b_2 \cdots b_{k-1} a_k & , k \geq 2 \end{cases}$$

Approximation d'un nombre réel x :

$$x = \left[a_0 ; \frac{b_1}{a_1}, \frac{b_2}{a_2}, \cdots, \frac{b_i}{a_i}, \cdots \right] , a_i \geq b_i > 0 \quad \Rightarrow \quad \left| x - \frac{p_k}{q_k} \right| \leq \frac{b_1 b_2 \cdots b_{k+1}}{q_k q_{k+1}}$$

Mathématique & Informatique

Critères d'irrationalité

- Critère 1

a_i et b_i sont des rationnels strictement positifs : a_i et $b_i \in \mathbb{Q}^+$

Si $a_i \geq b_i > 0$ pour i suffisamment grand, alors : $x = \left[a_0 ; \frac{b_1}{a_1}, \frac{b_2}{a_2}, \cdots, \frac{b_i}{a_i}, \cdots \right]$ est irrationnel.

En particulier, une fraction continue ordinaire infinie représente toujours un nombre irrationnel puisque : $a_i \geq b_i = 1 > 0$

- Critère 2

a_i et b_i sont des entiers relatifs : a_i et $b_i \in \mathbb{Z}$

Si $|a_i| > |b_i| + 1$ pour i suffisamment grand, alors : $x = \left[a_0 ; \frac{b_1}{a_1}, \frac{b_2}{a_2}, \cdots, \frac{b_i}{a_i}, \cdots \right]$ est irrationnel.

Par exemple : $\tan\frac{m}{n} = \left[0 ; \frac{m}{n}, -\frac{m^2}{3n}, -\frac{m^2}{5n}, -\frac{m^2}{7n}, \cdots, -\frac{m^2}{(2i+1)n}, \cdots \right]$ est irrationnel puisque $(2i + 1)|n| > |m|^2 + 1$ dès que $i > \mathrm{E}\left(\frac{|m|^2 - |n| + 1}{2|n|} \right)$.

Programme de calcul

Supposons que la variable *fc* contienne la liste des n+1 termes a_0 et b_i/a_i $(1 \leq i \leq n)$ d'une fraction continue généralisée. La fonction Scilab *reduite_generalisee* retourne la valeur de la fraction ordinaire $fr = p_k/q_k$ $(0 \leq k \leq n)$ équivalente à la réduite d'ordre k de *fc*.

```
function fr = reduite_generalisee(fc,k)
  if k >= length(fc) then
    fr = list() ; return
  end
  pim2 = [1,1] ; qim2 = [0,0]
  pim1 = fc(1) ; qim1 = [1,1]
  for i = 1:k
    fi = fc(i+1)
    pi = ADDITION(MULTIPLICATION(fi(2),pim1),MULTIPLICATION(fi(1),pim2))
    qi = ADDITION(MULTIPLICATION(fi(2),qim1),MULTIPLICATION(fi(1),qim2))
    pim2 = pim1 ; qim2 = qim1
    pim1 = pi ; qim1 = qi
  end
  fr = list(pim1,qim1)
endfunction
```

Mathématique & Informatique

EXEMPLE 1

Calculer l'approximation rationnelle et l'approximation décimale correspondant à la réduite d'ordre 50 de la fraction continue généralisée ci-dessous qui converge vers le nombre π :

$$\pi = 3 + \cfrac{1}{6 + \cfrac{9}{6 + \ddots + \cfrac{(2i-1)^2}{6 + \ddots}}} = \left[3; \frac{1}{6}, \frac{9}{6}, \cdots, \frac{(2i-1)^2}{6}, \cdots \right]$$

Le programme Scilab est le suivant :

```
exec("Scilab-Fractions_continues.sce",-1)
fc = list(ENTIER(3)) ;
for i = 1:50
    fc($+1) = FRACTION((2*i-1)^2,6) ;
end
printf('\n pi = '), FC_ECRIRE(fc)
fr = reduite_generalisee (fc,50) ;
printf('\n pi = '), ECRIRE(fr)
D = DECIMALE(fr,10) ;
printf('\n pi = '), ECRIRE(D)
```

Résultat affiché à l'écran :

pi = [3 ;1/6, 9/6, 25/6, 49/6, 81/6, 121/6, 169/6, 225/6, 289/6, 361/6, 441/6, 529/6, 625/6, 729/6, 841/6, 961/6, 1089/6, 1225/6, 1369/6, 1521/6, 1681/6, 1849/6, 2025/6, 2209/6, 2401/6, 2601/6, 2809/6, 3025/6, 3249/6, 3481/6, 3721/6, 3721/6, 3969/6, 4225/6, 4489/6, 4761/6, 5041/6, 5329/6, 5625/6, 5929/6, 6241/6, 6561/6, 6889/6, 7225/6, 7569/6, 7921/6, 8281/6, 8649/6, 9025/6, 9409/6, 9801/6]

$$pi = \frac{4410320603767014767055406471947281639659506336848278377214232570344419324114257 8125}{1403849491185600766985365392706013188008089189864143338414552776955544004223632 8125}$$

pi = 3.1415907698

En conclusion, la convergence est très lente puisqu'une fraction continue d'ordre élevé (50) ne permet d'obtenir que 5 décimales exactes de π, soit moins que le nombre de **Metius** $\frac{355}{113} = 3.14159292$ qui en donne 6.

Le développement en fraction continue généralisée peut être utile pour étudier les propriétés d'un nombre.
Dans le développement en fraction continue illimitée de π, les termes $(2i\text{-}1)^2/6$ constituent une suite croissante qui rend impossible toute périodicité, donc π n'est solution d'aucune équation du second degré à coefficients entiers.

Mathématique & Informatique

EXEMPLE 2

Vérifier numériquement, avec précision, la formule de **Ramanujan** suivante :

$$1 + \frac{1}{1.3} + \frac{1}{1.3.5} + \frac{1}{1.3.5.7} + \frac{1}{1.3.5.7.9} + \frac{1}{1.3.5.7.9.11} + \cdots + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{2}{1 + \cfrac{3}{1 + \cfrac{4}{1 + \cfrac{5}{1 + \cdots}}}}}} = \sqrt{\frac{e\,\pi}{2}}$$

Prouver l'exactitude de cette formule.

Notons : $S_n = \sum_{p=0}^{n} \left[\prod_{i=0}^{p} (2i + 1) \right]^{-1} = \dfrac{1 + \sum_{p=1}^{n} \prod_{i=1}^{p} [2(n-i)+3]}{\prod_{i=1}^{n} (2i + 1)}$ la somme partielle de la série

et : $F_n = \left[0 ; \frac{1}{1}, \frac{1}{1}, \frac{2}{1}, \frac{3}{1}, \frac{4}{1}, \frac{5}{1}, \cdots, \frac{j}{1}, \cdots, \frac{n}{1} \right]$ la réduite de la fraction continue

On a : $\lim_{n \infty} (S_n + F_n) = \sqrt{\frac{e\,\pi}{2}}$, d'où : $\varepsilon = \left| \sqrt{\frac{e\,\pi}{2}} - (S_n + F_n) \right| \underset{n \infty}{\to} 0$

Pour éviter d'extraire la racine carrée de $\frac{e\,\pi}{2}$, on calcule le nombre η défini par :

$$\eta = \left| \frac{e\,\pi}{2} - (S_n + F_n)^2 \right| \underset{n \infty}{\to} 0$$

Posons : $\mathrm{A} = \frac{e\,\pi}{2}$, $\mathrm{B} = (S_n + F_n)^2$

Il vient : $|\mathrm{A} - \mathrm{B}| = \eta \;\Rightarrow\; \left| \sqrt{\mathrm{A}} - \sqrt{\mathrm{B}} \right| = \varepsilon = \dfrac{\eta}{\left| 2\sqrt{\mathrm{A}} \pm \varepsilon \right|}$

Donc : $\varepsilon \approx \dfrac{\eta}{\sqrt{2e\pi}} \approx 0.24\eta$ lorsque n est grand.

Prenons *e* et π avec 50 décimales exactes, limitons le calcul de la série à l'ordre *n1* = 100 et celui de la fraction continue à l'ordre *n2* = 200, puis écrivons un programme pour obtenir la valeur correspondante de η :

```
funcprot(0) ; lines(0) ;
exec("Scilab-Fractions_continues.sce",-1)

// Constantes e et pi
e = [2,7,1,8,2,8,1,8,2,8,4,5,9,0,4,5,2,3,5,3,6,0,2,8,7,4,7,1,3,5,2,6,6,2,4,9,7,7,5,7,2,4,7,0,9,3,6,9,9,9,5] ;
pi = [3,1,4,1,5,9,2,6,5,3,5,8,9,7,9,3,2,3,8,4,6,2,6,4,3,3,8,3,2,7,9,5,0,2,8,8,4,1,9,7,1,6,9,3,9,9,3,7,5,1,0] ;

// Calcul de S
n1 = 100 ;
X = ENTIER(1) ; N = ENTIER(1) ; D = ENTIER(1) ;
for i = 1:n1
    X = MULTIPLICATION(X,ENTIER(2*(n1-i)+3)) ;
    N = ADDITION(N,X);
    D = MULTIPLICATION(D,ENTIER(2*i+1)) ;
end
S = list(N,D) ;
printf('\n ==> S = '), ECRIRE(S)

// Calcul de F
n2 = 200 ;
F = list(ENTIER(0),FRACTION(1,1)) ;
for j = 1:n2
    F($+1) = FRACTION(j,1) ;
end
F = reduite_generalisee(F,length(F)-1) ;
printf('\n ==> F = '), ECRIRE(F)

// Calcul de A = (e * pi)/2
N = [1,multiplication(e,pi)] ;
D = [1,2,zeros(1,100)] ;
A = list(N,D) ;
printf('\n ==> A = '), ECRIRE(A)

// Calcul de B = (S + F)^2
a = NUMERATEUR(S) ; b = DENOMINATEUR(S) ;
c = NUMERATEUR(F) ; d = DENOMINATEUR(F) ;
N = ADDITION(MULTIPLICATION(a,d),MULTIPLICATION(b,c)) ;
N = MULTIPLICATION(N,N) ;
D = MULTIPLICATION(b,d) ;
D = MULTIPLICATION(D,D) ;
B = list(N,D) ;
printf('\n ==> B = '), ECRIRE(B)

// Calcul de eta = |A - B|
a = NUMERATEUR(A) ; b = DENOMINATEUR(A) ;
c = NUMERATEUR(B) ; d = DENOMINATEUR(B) ;
N = SOUSTRACTION(MULTIPLICATION(a,d),MULTIPLICATION(b,c)) ;
N = ABSOLU(N) ;
D = MULTIPLICATION(b,d) ;
eta = list(N,D) ;
eta = DECIMALE(eta,50) ;
printf('\n ==> eta = '), ECRIRE(eta)
```

On trouve : eta = 0.00000000000794740872903422143394112425431788196866
On en déduit : $\eta \approx 7.95\ 10^{-12} \Rightarrow \varepsilon \approx 1.91\ 10^{-12}$
La petite valeur de ε, de l'ordre de $2\ 10^{-12}$, est en accord avec la formule.

Mathématique & Informatique

Démontrons la formule de **Ramanujan**.
Notons $S_\infty = \lim_{n\,\infty} S_n$ la série et $F_\infty = \lim_{n\,\infty} F_n$ la fraction continue.

Etape 1

Partons de l'équation différentielle linéaire du premier ordre : $y' = xy + 1$

L'intégrale générale est de la forme : $y(x) = e^{\frac{x^2}{2}}\left(\int_0^x e^{-\frac{u^2}{2}}\,du + \mathrm{C}\right)$ où C est une constante.

La série entière $y(x) = \sum_{n=0}^{\infty} a_n\, x^n$, solution générale, a pour coefficients : $a_{2p} = \frac{a_0}{\prod_{i=1}^{p}(2i)}$ et $a_{2p+1} = \frac{1}{\prod_{i=0}^{p}(2i+1)}$

Par suite, si l'on prend la solution particulière qui s'annule en $x = 0$ ($y(0) = 0$), on a en $x = 1$:

$$y(1) = \sqrt{e}\int_0^1 e^{-\frac{u^2}{2}}\,du = \sum_{p=0}^{\infty}\left[\prod_{i=0}^{p}(2i+1)\right]^{-1} = S_\infty$$

Etape 2

Utilisons l'expression du développement en fraction continue de la fonction d'erreur *erf*(x) (**Laplace**) :

$$erf(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,dt = 1 - \frac{e^{-x^2}}{\sqrt{\pi}}\cdot\cfrac{1}{x + \cfrac{1}{2x + \cfrac{2}{x + \cfrac{3}{2x + \cfrac{4}{x + \cdots}}}}}$$

On en déduit :

$$S_\infty = \sqrt{2e}\int_0^{\frac{1}{\sqrt{2}}} e^{-t^2}\,dt = \sqrt{\frac{e\pi}{2}}\,erf\left(\frac{1}{\sqrt{2}}\right)$$

$$= \sqrt{\frac{e\pi}{2}} - \frac{1}{\sqrt{2}}\cdot\cfrac{1}{\cfrac{1}{\sqrt{2}} + \cfrac{1}{\sqrt{2} + \cfrac{2}{\cfrac{1}{\sqrt{2}} + \cfrac{3}{\sqrt{2} + \cfrac{4}{\cfrac{1}{\sqrt{2}} + \cdots}}}}}} = \sqrt{\frac{e\pi}{2}} - \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{2}{1 + \cfrac{3}{1 + \cfrac{4}{1 + \cdots}}}}}$$

$$= \sqrt{\frac{e\pi}{2}} - F_\infty$$

D'où : $S_\infty + F_\infty = \sqrt{\frac{e\pi}{2}}$, c'est-à-dire :

$$1 + \frac{1}{1.3} + \frac{1}{1.3.5} + \frac{1}{1.3.5.7} + \frac{1}{1.3.5.7.9} + \cdots + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{2}{1 + \cfrac{3}{1 + \cfrac{4}{1 + \cdots}}}}} = \sqrt{\frac{e\pi}{2}}$$

GRANDS NOMBRES COMPLEXES

Considérons le sous-corps $\mathbb{Q}[i]$ des nombres complexes de la forme :

$$z = x + i\,y = \frac{a}{b} + i\frac{c}{d}$$

où *a, b, c, d* sont des entiers relatifs.
Les entiers de **Gauss** $\mathbb{Z}[i] = \{p + iq / p, q \in \mathbb{Z}\}$ font partie de $\mathbb{Q}[i]$
Adoptons la représentation suivante :

$$z = \text{list}(a, b, c, d)$$

où *a, b, c, d* sont les vecteurs de chiffres équivalents aux numérateurs et dénominateurs des fractions constituant z.
Programmons les différentes fonctions Scilab permettant d'effectuer les opérations habituelles sur ces complexes :

```
//---  CONVERSION_1

     function z = COMPLEXE(a,b,c,d)
          z = list(ENTIER(a),ENTIER(b),ENTIER(c),ENTIER(d))
     endfunction

//---  CONVERSION_2

     function c = nombre_complexe(z)
       c = nombre_rationnel(REELLE(z)) + %i*nombre_rationnel(IMAGINAIRE(z))
     endfunction

//---  PARTIE REELLE

     function x = REELLE(z)
       x = list(z(1),z(2))
     endfunction

//---  PARTIE IMAGINAIRE

     function y = IMAGINAIRE(z)
       y = list(z(3),z(4))
     endfunction

//---  REDUCTION

     function zr = C_REDUCTION(z)
       x = F_REDUCTION(REELLE(z))
       y = F_REDUCTION(IMAGINAIRE(z))
       zr = list(x(1),x(2),y(1),y(2))
     endfunction
```

//--- COMPARAISON : z1 = z2

```
function s = C_COMPARAISON(z1,z2)
  s1 = F_COMPARAISON(REELLE(z1),REELLE(z2))
  s2 = F_COMPARAISON(IMAGINAIRE(z1),IMAGINAIRE(z2))
  if s1 == s2 then, s = s1, else, s = [], end
endfunction
```

//--- TEST : z = 0

```
function t = C_NUL(z)
  if length(z) == 0 then, t = %f, return, end
  t = C_COMPARAISON(z,COMPLEXE(0,1,0,1)) == 0
endfunction
```

//--- ENTIER : z = entier de Gauss

```
function t = C_ENTIER(z)
  zr = C_REDUCTION(z)
  t = isequal(VALEUR(zr(2)),1) & isequal(VALEUR(zr(4)),1)
endfunction
```

//--- MOINS : zm = - z

```
function zm = C_MOINS(z)
  zm = list(MOINS(z(1)),z(2),MOINS(z(3)),z(4))
endfunction
```

//--- CONJUGUE : zc = x - iy

```
function zc = C_CONJUGUE(z)
  zc = list(z(1),z(2),MOINS(z(3)),z(4))
endfunction
```

//--- ABSOLU : za = |x| + i|y|

```
function za = C_ABSOLU(z)
  za = list(ABSOLU(z(1)),ABSOLU(z(2)),ABSOLU(z(3)),ABSOLU(z(4)))
endfunction
```

//--- NORME(*) : Nz = x^2 + y^2

```
function Nz = C_NORME(z)
  x2 = F_MULTIPLICATION(REELLE(z),REELLE(z))
  y2 = F_MULTIPLICATION(IMAGINAIRE(z),IMAGINAIRE(z))
  Nz = F_ADDITION(x2,y2)
endfunction
```

(*) la "norme" d'un nombre complexe est le carré de son module (terminologie de **Gauss**).

//--- ADDITION : z = z1 + z2

```
function z = C_ADDITION(z1,z2)
  x = F_ADDITION(REELLE(z1),REELLE(z2))
  y = F_ADDITION(IMAGINAIRE(z1),IMAGINAIRE(z2))
  z = list(x(1),x(2),y(1),y(2))
endfunction
```

//--- SOUSTRACTION : z = z1 - z2

```
function z = C_SOUSTRACTION(z1,z2)
  x = F_SOUSTRACTION(REELLE(z1),REELLE(z2))
  y = F_SOUSTRACTION(IMAGINAIRE(z1),IMAGINAIRE(z2))
  z = list(x(1),x(2),y(1),y(2))
endfunction
```

//--- MULTIPLICATION : z = z1 * z2

```
function z = C_MULTIPLICATION(z1,z2)
  a = F_MULTIPLICATION(REELLE(z1),REELLE(z2))
  b = F_MULTIPLICATION(IMAGINAIRE(z1),IMAGINAIRE(z2))
  c = F_MULTIPLICATION(REELLE(z1),IMAGINAIRE(z2))
  d = F_MULTIPLICATION(IMAGINAIRE(z1),REELLE(z2))
  x = F_SOUSTRACTION(a,b)
  y = F_ADDITION(c,d)
  z = list(x(1),x(2),y(1),y(2))
endfunction
```

//--- DIVISION : z = z1 / z2

```
function z = C_DIVISION(z1,z2)
  F = C_NORME(z2)
  Z = C_MULTIPLICATION(z1,C_CONJUGUE(z2))
  x = F_DIVISION(REELLE(Z),F)
  y = F_DIVISION(IMAGINAIRE(Z),F)
  z = list(x(1),x(2),y(1),y(2))
endfunction
```

//--- INVERSION : zi = 1/z

```
function zi = C_INVERSION(z)
  F = C_NORME(z)
  x = F_DIVISION(REELLE(z),F)
  y = F_MOINS(F_DIVISION(IMAGINAIRE(z),F))
  zi = list(x(1),x(2),y(1),y(2))
endfunction
```

//--- PUISSANCE : zn = z ^ n

```
function zn = C_PUISSANCE(z,n)
  zn = COMPLEXE(1,1,0,1)
  while n <> 0
    if modulo(n,2) <> 0 then
      zn = C_MULTIPLICATION(zn,z)
    end
    n = floor(n/2)
    if n <> 0 then
      z = C_MULTIPLICATION(z,z)
    end
  end
endfunction
```

Mathématique & Informatique

EXEMPLE

Quelle est la valeur exacte du nombre complexe suivant :

$$z = \frac{\left(-31 + i\,\frac{57}{13}\right)^{21}}{\frac{4092010}{37536877} - i\,\frac{62402587183479}{107017421}}$$

Posons $z = z_1 / z_2 = x + i\,y$ et exécutons les instructions ci-dessous :

```
z1 = C_PUISSANCE(COMPLEXE(-31,1,57,13),21) ;
z2 = COMPLEXE(4092010,37536877,-62402587183479,107017421) ;
z = C_DIVISION(z1,z2) ;
x = DECIMALE(REELLE(z),50) ;
y = DECIMALE(IMAGINAIRE(z),50) ;
printf('\n z = '), ECRIRE(z)
printf('\n x = '), ECRIRE(x)
printf('\n y = '), ECRIRE(y)
```

On obtient :

$$z = -\frac{230868761736889790642332230643861404037010125596032876267337471953329581457013183488}{27665325509283210504359206905925808762439992641336867595193}$$

$$+ i\,\frac{918753082699418343394243493365128149487645156181277640218041750715682692674396230553621281019681406400818027631300455831443264614816256682116861}$$

$$x \approx -83450585730830063710960 0.89265650714821928171797830220809138755926018405$$

$$y \approx 43172418401732927629826 18.87681312690534502416408476515125129498797474007$$

Remarque

Le calcul usuel avec Scilab donne une valeur approchée du nombre z :

```
z1 = (-31 + %i*57/13)^21 ;
z2 = 4092010/37536877 - %i*62402587183479/107017421 ;
z = z1/z2 ;
x = real(z) ;
y = imag(z) ;
printf('\n z = %.15e + i%.15e',x,y)
```

Soit :

```
z = -8.345058573083003e+024 + i4.317241840117330e+025
```

Mathématique & Informatique

Systèmes d'équations linéaires à coefficients complexes

Il s'agit de résoudre dans $\mathbb{C}$ un système linéaire de n équations à n inconnues de la forme :

$$\begin{cases} a_{11}z_1 + \ldots + a_{1n}z_n = b_1 \\ \vdots \qquad\qquad \vdots \qquad \vdots \\ a_{n1}z_1 + \ldots + a_{nn}z_n = b_n \end{cases} \Leftrightarrow \quad Az = b$$

où les coefficients a_{ij} et b_i ($1 \le i \le n$, $1 \le j \le n$) sont des nombres complexes appartenant à $\mathbb{Q}$[i].

On suppose que le déterminant Δ du système est non nul. Par suite, le vecteur solution $z = (z_1 , \ldots , z_n)$ est unique et ses composantes z_i sont des nombres complexes dont les parties réelles et imaginaires sont des nombres rationnels.

Pour calculer z nous utiliserons la méthode de triangularisation de **Gauss**, valable dans n'importe quel corps K et en particulier dans le corps $\mathbb{C}$ des nombres complexes.

On écrira une fonction Scilab nommée *C_GAUSS* qui retourne les z_i lorsqu'on fournit la liste des a_{ij} et la liste des b_i.

Exemples

- Résoudre dans $\mathbb{C}$:

$$\begin{cases} (1+i)z - it = 2 + i \\ (2+i)z + (2-i)t = 2i \end{cases}$$

Voir sur **YouTube** (**http://www.youtube.com/watch?v=X65g1bjEHfU**) la résolution détaillée de ce système.

- Quelle est la solution exacte (z_1 , z_2 , z_3 , z_4 , z_5) du système linéaire suivant :

$$\begin{cases} \left(\frac{3}{5} - 6i\right)z_1 + \left(\frac{2}{7} + \frac{13}{2}i\right)z_2 - \left(\frac{4}{15} + \frac{11}{8}i\right)z_3 - (9+i)z_4 - \left(6 + \frac{17}{3}i\right)z_5 = \frac{3}{4} + \frac{4}{9}i \\ \left(\frac{8}{3} - \frac{6}{11}i\right)z_1 - \left(\frac{1}{4} - 10i\right)z_2 - 3iz_3 - \left(\frac{9}{2} + 7i\right)z_5 = \frac{1}{3} - \frac{1}{2}i \\ \left(-\frac{2}{7} - 5i\right)z_3 - \left(7 + \frac{1}{7}i\right)z_4 - 19z_5 = i \\ (1-i)z_1 + \left(\frac{13}{17} + \frac{7}{2}i\right)z_2 - 7z_3 - z_4 - \frac{4}{13}z_5 = 0 \\ (1+i)z_1 + (1-i)z_2 + \left(\frac{3}{11} + \frac{11}{8}i\right)z_3 + (1+i)z_4 + \left(\frac{6}{5} + \frac{8}{3}i\right)z_5 = 8 + 10i \end{cases}$$

Application

La fonction *C_GAUSS* permet de résoudre le système d'équations linéaires résultant de l'application des lois de **Kirchhoff** à un réseau électrique constitué d'impédances R, L, C soumises à des tensions sinusoïdales de même pulsation.
Rappelons qu'une résistance de R ohms a pour impédance complexe $Z_R = R$, une inductance de L henrys a pour impédance complexe $Z_L = jL\omega$ et un condensateur de C farads a pour impédance complexe $Z_C = \frac{1}{jC\omega}$

(j est l'unité imaginaire ; ω désigne la pulsation du signal sinusoïdal).

```
function z = C_GAUSS(A,b)
  if (b($) <> 1)|(b($-1) <> A($-1))|(A($-1) <> A($)) then
    z = list(0,0)
    return
  end
  n = A($)
  z = list()
  for i = 1:n
    z($+1) = []
  end
  z($+1) = n
  z($+1) = 1
  for k = 1:n
    p = 0
    for i = k:n
      if ~C_NUL(A((i-1)*n+k)) then
        p = i
        break
      end
    end
    if p == 0 then
      printf('\n Matrice singuliere')
      z = list(0,0)
      return
    end
    if p <> k then
      for j = k:n
        t = A((k-1)*n+j)
        A((k-1)*n+j) = A((p-1)*n+j)
        A((p-1)*n+j) = t
      end
      t = b(k)
      b(k) = b(p)
      b(p) = t
    end
    for i = k+1:n
      if ~C_NUL(A((i-1)*n+k)) then
        t = C_DIVISION(A((i-1)*n+k),A((k-1)*n+k))
        for j = k+1:n
          u = C_MULTIPLICATION(t,A((k-1)*n+j))
          A((i-1)*n+j) = C_SOUSTRACTION(A((i-1)*n+j),u)
        end
        v = C_MULTIPLICATION(t,b(k))
        b(i) = C_SOUSTRACTION(b(i),v)
      end
    end
  end
  for i = n:-1:1
    t = COMPLEXE(0,1,0,1)
    for j = i+1:n
      u = C_MULTIPLICATION(A((i-1)*n+j),z(j))
      t = C_ADDITION(t,u)
    end
    v = C_SOUSTRACTION(b(i),t)
    z(i) = C_DIVISION(v,A((i-1)*n+i))
  end
endfunction
```

EXEMPLES

Pour le premier système linéaire de 2 équations à 2 inconnues, on écrit :

```
exec("Scilab-Equations.sce",-1)
A = list(COMPLEXE(1,1,1,1), COMPLEXE(0,1,-1,1), COMPLEXE(2,1,1,1), COMPLEXE(2,1,-1,1),2,2) ;
b = list(COMPLEXE(2,1,1,1), COMPLEXE(0,1,2,1),2,1) ;
//------------------------------------------------------------------------------------------------------------------------------------------ --
z = C_GAUSS(A,b) ;
for i = 1:2
   printf('\n z%i = ',i), ECRIRE(z(i))
end
```

On obtient : $\begin{cases} (1+i)z_1 - iz_2 = 2+i \\ (2+i)z_1 + (2-i)z_2 = 2i \end{cases} \Rightarrow \begin{array}{l} z_1 = \dfrac{6}{13} - i\dfrac{9}{13} \\ z_2 = -\dfrac{16}{13} + i\dfrac{11}{13} \end{array}$

Note

L'instruction Scilab $z = A \setminus b$ fournit une solution approchée du système linéaire $Az = b$ dans $\mathbb{C}$. Ainsi, dans l'exemple ci-dessus, si l'on saisit l'instruction z = [1 + %i , - %i ; 2 + %i , 2 - %i] \ [2 + %i ; 2*%i], les composantes z_1 et z_2 du vecteur colonne z s'affichent dans la fenêtre Scilab : 0.4615385 - 0.6923077i et - 1.2307692 + 0.8461538i.

Pour le deuxième système linéaire de 5 équations à 5 inconnues, on écrit :

```
exec("Scilab-Equations.sce",-1)
A = list(COMPLEXE(3,5,-6,1), COMPLEXE(2,7,13,2), COMPLEXE(-4,15,-11,8), COMPLEXE(-9,1,-1,1),COMPLEXE(-6,1,-17,3), ...
        COMPLEXE(8,3,-6,11), COMPLEXE(-1,4,10,1), COMPLEXE(0,1,-3,1), COMPLEXE(0,1,0,1),COMPLEXE(-9,2,-7,1),...
        COMPLEXE(0,1,0,1),COMPLEXE(0,1,0,1),COMPLEXE(-2,7,-5,1), COMPLEXE(-7,1,-1,7), COMPLEXE(-19,1,0,1), ...
        COMPLEXE(1,1,-1,1), COMPLEXE(13,17,7,2), COMPLEXE(-7,1,0,1), COMPLEXE(-1,1,0,1),COMPLEXE(-4,13,0,1), ...
        COMPLEXE(1,1,1,1), COMPLEXE(1,1,-1,1), COMPLEXE(3,11,11,8), COMPLEXE(1,1,1,1),COMPLEXE(6,5,8,3), ...
        5,5) ;
b = list(COMPLEXE(3,4,4,9), COMPLEXE(1,3,-1,2), COMPLEXE(0,1,1,1), COMPLEXE(0,1,0,1),COMPLEXE(8,1,10,1),5,1) ;
//------------------------------------------------------------------------------------------------------------------------------------------
z = C_GAUSS(A,b) ;
for i = 1:5
   printf('\n z%i = ',i), ECRIRE(z(i))
end
```

Résultat :

$$z_1 = \frac{27874057021218616614562437754}{3872487495918991488733620337} + i\frac{28971964931662293967496226990 9}{10842964988573176168454136943 6}$$

$$z_2 = \frac{65413184073331000397416857 89}{11617462487756974466200861011} + i\frac{42848220459604455171180308156}{11617462487756974466200861011}$$

$$z_3 = -\frac{11434709413921319609522853502}{27107412471432940421135342359} + i\frac{60178791502079959195107316498}{81322237414298821263406027077}$$

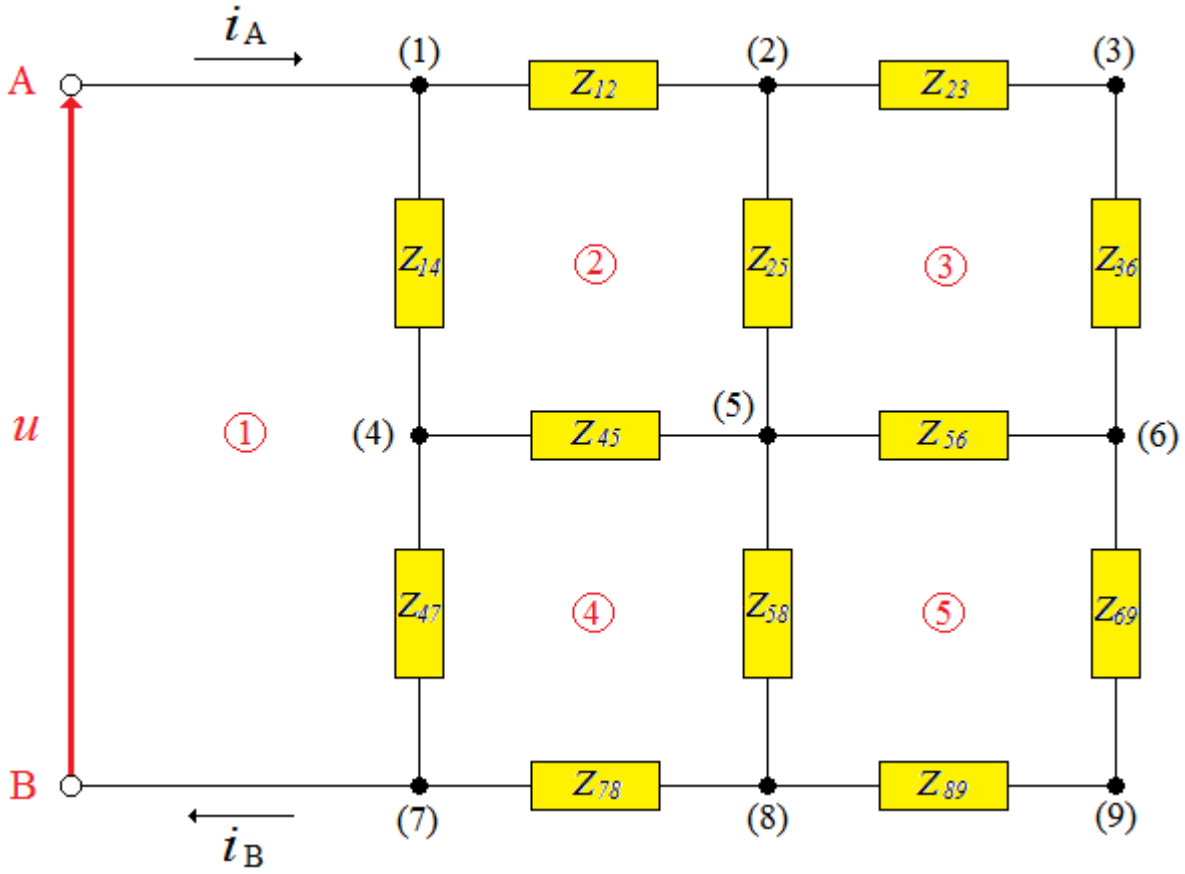
$$z_4 = \frac{49485578247343036413583348 9}{14853376696675583792402927 32} - i\frac{24770160239406630743336839103}{44560130090026751377208781 96}$$

$$z_5 = \frac{19811337556719807971658576 05}{54214824942865880842270684718} + i\frac{56728455347259263462552316490}{27107412471432940421135342359}$$

Mathématique & Informatique

APPLICATION

Une tension sinusoïdale $u = 10 \sin(10000t)$ alimente le réseau électrique schématisé ci-dessous :



Les divers composants Z_{ij} sont les suivants :

- $Z_{12} = Z_{14} = Z_{23} = Z_{36} = Z_{47} = Z_{69} = Z_{78} = Z_{89}$ = résistance de 12 kΩ
- Z_{45} = inductance de 15 mH
- Z_{56} = inductance de 47 mH
- Z_{25} = condensateur de 100 nF
- Z_{58} = condensateur de 220 nF

Calculer la valeur de l'impédance complexe $\underline{Z} = R + jX$ (R résistance, X réactance) vue des bornes A et B du réseau. Vérifier numériquement que la puissance complexe $\underline{S}$ dans le réseau d'impédance $\underline{Z}$ est égale à la somme des puissances complexes $\underline{S_{ij}}$ dans les différents dipôles d'impédances $\underline{Z_{ij}}$ (théorème de **Boucherot**).

A propos de l'utilisation des nombres complexes en électricité

1) Communément, dans l'expression d'une grandeur électrique complexe, le symbole j désigne l'unité imaginaire ($j^2 = -1$), à ne pas confondre avec la racine cubique de l'unité ($j^3 = 1$) ou avec la densité de courant ($j = di/ds$). En sciences physiques, le symbole i est habituellement réservé à un courant électrique, c'est pourquoi un nombre complexe s'écrit $a + jb$ alors qu'il s'écrit $a + ib$ en mathématique. En programmation informatique Scilab il n'y a pas de risque de confusion puisque l'unité imaginaire est considérée comme une constante spéciale associée à l'identificateur *%i* ; un nombre complexe s'écrit *a + %i*b* où *a* et *b* sont 2 nombres réels standards.
2) En régime alternatif sinusoïdal (*), une grandeur électrique complexe G est conventionnellement notée $\underline{G}$ (ou $\widehat{G}$). Les valeurs complexes de la tension, du courant, de l'impédance et de la puissance électrique sont notées $\underline{U}$, $\underline{I}$, $\underline{Z}$, $\underline{S}$ (ou $\widehat{U}$, $\widehat{I}$, $\widehat{Z}$, $\widehat{S}$). La tension complexe $\underline{U}$ et le courant complexe $\underline{I}$ s'expriment sous la forme exponentielle $r\,e^{j\theta}$: $\underline{U} = U_m\, e^{j(\omega t + \varphi_u)}$ et $\underline{I} = I_m\, e^{j(\omega t + \varphi_i)}$. La tension instantanée u et le courant instantané i sont définis par une fonction sinus ou par une fonction cosinus (au choix). Correspondance entre les grandeurs réelles et complexes :

$$u = U_m \sin(\omega t + \varphi_u) \text{ et } i = I_m \sin(\omega t + \varphi_i) \Leftrightarrow u = \mathrm{Im}(\underline{U}) \text{ et } i = \mathrm{Im}(\underline{I})$$
$$u = U_m \cos(\omega t + \varphi_u) \text{ et } i = I_m \cos(\omega t + \varphi_i) \Leftrightarrow u = \mathrm{Re}(\underline{U}) \text{ et } i = \mathrm{Re}(\underline{I})$$

Mathématique & Informatique

(*) Synthèse des notations usuelles et des formules essentielles en régime alternatif sinusoïdal

Dans $\mathbb{R}$:

- La tension électrique instantanée u est de la forme :

$$\boxed{u = U_m \sin(\omega t + \varphi_u)}$$

U_m est la tension maximale en volts (V)
ω est la pulsation en radians par seconde (rad/s)
t est le temps en secondes (s)
φ_u est la phase à l'origine en radians (rad)

- Le courant électrique instantané i est de la forme :

$$\boxed{i = I_m \sin(\omega t + \varphi_i)}$$

I_m est le courant maximal en ampères (A)
ω est la pulsation en radians par seconde (rad/s)
t est le temps en secondes (s)
φ_i est la phase à l'origine en radians (rad)

- La puissance électrique instantanée p est définie par :

$$\boxed{p = ui}$$ | u en volts (V), i en ampères (A), p en watts (W)

Pulsation ω, période T, fréquence f d'un signal sont liées par la relation : $\boxed{\omega = \dfrac{2\pi}{T} = 2\pi f}$

Le déphasage φ entre la tension et le courant est : $\boxed{\varphi = \varphi_u - \varphi_i}$

La tension efficace U, le courant efficace I et la puissance moyenne P sont donnés par :

$$\boxed{U = U_{\text{RMS}} = \sqrt{\frac{1}{T}\int_0^T u^2 dt} = \frac{U_m}{\sqrt{2}} \ , \ I = I_{\text{RMS}} = \sqrt{\frac{1}{T}\int_0^T i^2 dt} = \frac{I_m}{\sqrt{2}} \ , \ P = \frac{1}{T}\int_0^T p\, dt = UI\cos\varphi}$$

Dans $\mathbb{C}$:

- La tension complexe $\underline{U}$ est définie par : $\boxed{\underline{U} = U_m \, \mathrm{e}^{j(\omega t + \varphi_u)}}$
- Le courant complexe $\underline{I}$ est défini par : $\boxed{\underline{I} = I_m \, \mathrm{e}^{j(\omega t + \varphi_i)}}$
- L'impédance complexe $\underline{Z}$ est définie par : $\boxed{\underline{Z} = \dfrac{\underline{U}}{\underline{I}}}$

$$\underline{Z} = R + jX = Z\mathrm{e}^{j\varphi}$$

R est la résistance en ohms (Ω) : $R = \mathrm{Re}(\underline{Z}) = \dfrac{U}{I}\cos\varphi$
X est la réactance en ohms (Ω) : $X = \mathrm{Im}(\underline{Z}) = \dfrac{U}{I}\sin\varphi$
Z est l'impédance réelle en ohms (Ω) : $Z = |\underline{Z}| = \sqrt{R^2 + X^2} = \dfrac{U}{I}$
φ est le déphasage tension - courant en radians (rad) : $\varphi = \mathrm{Arg}(\underline{Z}) = \mathrm{Arctan}\left(\dfrac{X}{R}\right)$

- La puissance complexe $\underline{S}$ est définie par : $\boxed{\underline{S} = \dfrac{1}{2}\,\underline{U}\,\underline{I}^*}$ où $\underline{I}^*$ désigne le conjugué de $\underline{I}$

$$\underline{S} = P + jQ = S\mathrm{e}^{j\varphi}$$

P est la puissance active en watts (W) : $P = \mathrm{Re}(\underline{S}) = UI\cos\varphi$
Q est la puissance réactive en volts - ampères - réactifs (VAR) : $Q = \mathrm{Im}(\underline{S}) = UI\sin\varphi$
S est la puissance apparente en volts - ampères (VA) : $S = |\underline{S}| = \sqrt{P^2 + Q^2} = UI$
φ est le déphasage tension - courant en radians (rad) : $\varphi = \mathrm{Arg}(\underline{S}) = \mathrm{Arctan}\left(\dfrac{Q}{P}\right)$

Solution

La méthode de résolution est analogue à celle utilisée dans un problème précédent. Ici, on est dans le cas général d'impédances Z_{ij} au lieu de simples résistances r_{ij}. On aboutit à un système linéaire de 14 équations à 14 inconnues dans $\mathbb{C}$:

$$\begin{pmatrix}
0 & 0 & 0 & \underline{Z_{14}} & 0 & 0 & 0 & 0 & \underline{Z_{47}} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \underline{Z_{12}} & -\underline{Z_{14}} & 0 & \underline{Z_{25}} & 0 & -\underline{Z_{45}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \underline{Z_{23}} & -\underline{Z_{25}} & \underline{Z_{36}} & 0 & 0 & -\underline{Z_{56}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \underline{Z_{45}} & -\underline{Z_{47}} & 0 & \underline{Z_{58}} & 0 & -\underline{Z_{78}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \underline{Z_{56}} & -\underline{Z_{58}} & \underline{Z_{69}} & 0 & -\underline{Z_{89}} \\
1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
\underline{I_A} \\ \underline{I_B} \\ \underline{I_{12}} \\ \underline{I_{14}} \\ \underline{I_{23}} \\ \underline{I_{25}} \\ \underline{I_{36}} \\ \underline{I_{45}} \\ \underline{I_{47}} \\ \underline{I_{56}} \\ \underline{I_{58}} \\ \underline{I_{69}} \\ \underline{I_{78}} \\ \underline{I_{89}}
\end{pmatrix}
=
\begin{pmatrix}
\underline{U} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{pmatrix}$$

A la pulsation $\omega = 10\,000$ rad/s, les valeurs des impédances complexes sont :

- $\underline{Z_{12}} = \underline{Z_{14}} = \underline{Z_{23}} = \underline{Z_{36}} = \underline{Z_{47}} = \underline{Z_{69}} = \underline{Z_{78}} = \underline{Z_{89}} = 12.\ 10^3 = 12000\ \Omega$
- $\underline{Z_{45}} = j \times 15.\ 10^{-3} \times 10000 = j150\ \Omega$
- $\underline{Z_{56}} = j \times 47.\ 10^{-3} \times 10000 = j470\ \Omega$
- $\underline{Z_{25}} = \dfrac{1}{j \times 100.10^{-9} \times 10000} = -j1000\ \ \Omega$
- $\underline{Z_{58}} = \dfrac{1}{j \times 220.10^{-9} \times 10000} = -j\dfrac{5000}{11}\ \ \Omega$

La tension complexe $\underline{U} = 10\,\mathrm{e}^{\,j10000t}$ représente la tension instantanée $u = 10\sin(10000t)$.

Les courants complexes $\underline{I_A} = \underline{I_B} = \underline{I}$ et $\underline{I_{ij}}$ sont remplacés par de nouvelles inconnues z_k permettant de se ramener à un système linéaire indépendant du temps t (le terme $\mathrm{e}^{j\omega t}$ n'apparaît plus dans les équations) :

$$\begin{pmatrix}
\underline{I_A} \\ \underline{I_B} \\ \underline{I_{12}} \\ \underline{I_{14}} \\ \underline{I_{23}} \\ \underline{I_{25}} \\ \underline{I_{36}} \\ \underline{I_{45}} \\ \underline{I_{47}} \\ \underline{I_{56}} \\ \underline{I_{58}} \\ \underline{I_{69}} \\ \underline{I_{78}} \\ \underline{I_{89}}
\end{pmatrix}
= \underline{U}
\begin{pmatrix}
z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \\ z_8 \\ z_9 \\ z_{10} \\ z_{11} \\ z_{12} \\ z_{13} \\ z_{14}
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
0 & 0 & 0 & \underline{Z_{14}} & 0 & 0 & 0 & 0 & \underline{Z_{47}} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \underline{Z_{12}} & -\underline{Z_{14}} & 0 & \underline{Z_{25}} & 0 & -\underline{Z_{45}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \underline{Z_{23}} & -\underline{Z_{25}} & \underline{Z_{36}} & 0 & 0 & -\underline{Z_{56}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \underline{Z_{45}} & -\underline{Z_{47}} & 0 & \underline{Z_{58}} & 0 & -\underline{Z_{78}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \underline{Z_{56}} & -\underline{Z_{58}} & \underline{Z_{69}} & 0 & -\underline{Z_{89}} \\
1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \\ z_8 \\ z_9 \\ z_{10} \\ z_{11} \\ z_{12} \\ z_{13} \\ z_{14}
\end{pmatrix}
=
\begin{pmatrix}
1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{pmatrix}$$

Mathématique & Informatique

L'expression de l'impédance complexe $\underline{Z}$ se déduit de la loi d'**Ohm** généralisée $\underline{U} = \underline{Z}\,\underline{I}$ et de la relation algébrique $\underline{I} = \underline{U}\,z_1$:

$$\underline{Z} = \frac{1}{z_1}$$

Le programme de calcul de $\underline{Z}$ s'écrit :

```
exec("Scilab-Equations.sce",-1)
A = list(0,0,0,[1,4],0,0,0,0,[4,7],0,0,0,0,0, ...
         0,0,[1,2],-[1,4],0,[2,5],0,-[4,5],0,0,0,0,0,0, ...
         0,0,0,0,[2,3],-[2,5],[3,6],0,0,-[5,6],0,0,0,0, ...
         0,0,0,0,0,0,0,[4,5],-[4,7],0,[5,8],0,-[7,8],0, ...
         0,0,0,0,0,0,0,0,0,[5,6],-[5,8],[6,9],0,-[8,9], ...
         1,0,-1,-1,0,0,0,0,0,0,0,0,0,0, ...
         0,0,1,0,-1,-1,0,0,0,0,0,0,0,0, ...
         0,0,0,0,1,0,-1,0,0,0,0,0,0,0, ...
         0,0,0,1,0,0,0,-1,-1,0,0,0,0,0, ...
         0,0,0,0,0,1,0,1,0,-1,-1,0,0,0, ...
         0,0,0,0,0,0,1,0,0,1,0,-1,0,0, ...
         0,-1,0,0,0,0,0,0,1,0,0,0,-1,0, ...
         0,0,0,0,0,0,0,0,0,0,1,0,1,-1, ...
         0,0,0,0,0,0,0,0,0,0,0,1,0,1, ...
         14,14) ;
b = list(1,0,0,0,0,0,0,0,0,0,0,0,0,0, ...
         14,1) ;
for n = 1:14^2
    if A(n) == 0 then, A(n) = COMPLEXE(0,1,0,1) ; continue, end
    if A(n) == 1 then, A(n) = COMPLEXE(1,1,0,1) ; continue, end
    if A(n) == -1 then, A(n) = COMPLEXE(-1,1,0,1) ; continue, end
    I = A(n) ;
    i = abs(I(1)) ;
    j = abs(I(2)) ;
    select [i,j]
        case [2,5] then
            Zij = COMPLEXE(0,1,-1000,1) ;
        case [4,5] then
            Zij = COMPLEXE(0,1,150,1) ;
        case [5,6] then
            Zij = COMPLEXE(0,1,470,1) ;
        case [5,8] then
            Zij = COMPLEXE(0,1,-5000,11) ;
        else
            Zij = COMPLEXE(12000,1,0,1) ;
    end
    s = COMPLEXE(I(1),abs(I(1)),0,1) ;
    A(n) = C_MULTIPLICATION(s,Zij) ;
end
for n = 1:14
    b(n) = COMPLEXE(b(n),1,0,1) ;
end
//-------------------------------------------------------------------------
z = C_GAUSS(A,b) ;
Z = C_INVERSION(z(1)) ;
printf('\n Z = '), ECRIRE(Z)
```

L'exécution de ce programme donne l'impédance complexe équivalente recherchée :

$$\underline{Z} = \frac{546857931246615000}{45476734353689} - j\frac{16454716968933000}{45476734353689}$$

Mathématique & Informatique

Soit $\underline{S}$ la puissance complexe dans le réseau d'impédance $\underline{Z}$ et $\underline{S_{ij}}$ la puissance complexe dans le dipôle d'impédance $\underline{Z_{ij}}$:

$$\underline{S} = \frac{1}{2}\,\underline{U}\,\underline{I}^{*} = \frac{1}{2}\,\underline{U}\left(\frac{\underline{U}}{\underline{Z}}\right)^{*} = \frac{1}{2}\,\underline{U}\,\underline{U}^{*}\left(\frac{1}{\underline{Z}}\right)^{*} = \frac{1}{2}\,|\underline{U}|^{2}\,z_1{}^{*}$$

$$\underline{S_{ij}} = \frac{1}{2}\,\underline{U_{ij}}\,\underline{I_{ij}}^{*} = \frac{1}{2}\,\underline{Z_{ij}}\,\underline{I_{ij}}\,\underline{I_{ij}}^{*} = \frac{1}{2}\,\left|\underline{I_{ij}}\right|^{2}\,\underline{Z_{ij}} = \frac{1}{2}\,|\underline{U}\,z_k|^{2}\,\underline{Z_{ij}} = \frac{1}{2}\,|\underline{U}|^{2}\,|z_k|^{2}\,\underline{Z_{ij}}$$

D'une part, $\underline{S}$ s'obtient en écrivant :

```
z1c = C_CONJUGUE(z(1)) ;
S = C_MULTIPLICATION(COMPLEXE(50,1,0,1),z1c) ;
printf('\n S = '), ECRIRE(S)
```

On trouve :

$$\underline{S} = \frac{12152398472147}{2925299021887656} - j\,\frac{1828301885437}{14626495109438280}$$

D'autre part, $\sum \underline{S_{ij}}$ s'obtient en écrivant :

```
sigma = COMPLEXE(0,1,0,1) ;
for k = 3:14
    Nzk = C_NORME(z(k)) ;
    zk2 = list(Nzk(1),Nzk(2),[0,0],[1,1]) ;
    select k
    case 6 then, cplx = C_MULTIPLICATION(zk2,COMPLEXE(0,1,-1000,1)) ;
    case 8 then, cplx = C_MULTIPLICATION(zk2,COMPLEXE(0,1,150,1)) ;
    case 10 then, cplx = C_MULTIPLICATION(zk2,COMPLEXE(0,1,470,1)) ;
    case 11 then, cplx = C_MULTIPLICATION(zk2,COMPLEXE(0,1,-5000,11)) ;
    else, cplx = C_MULTIPLICATION(zk2,COMPLEXE(12000,1,0,1)) ;
    end
    Sij = C_MULTIPLICATION(COMPLEXE(50,1,0,1),cplx) ;
    sigma = C_ADDITION(sigma,Sij) ;
    printf('\n Sij = '), ECRIRE(Sij)
end
printf('\n sigma = '), ECRIRE(sigma)
```

On trouve :

$$\sum \underline{S_{ij}} = \underline{S_{12}} + \underline{S_{14}} + \underline{S_{23}} + \underline{S_{25}} + \underline{S_{36}} + \underline{S_{45}} + \underline{S_{47}} + \underline{S_{56}} + \underline{S_{58}} + \underline{S_{69}} + \underline{S_{78}} + \underline{S_{89}}$$

$$= \frac{72406476034217}{70207176525303744} + \frac{40761288326093}{39003986958502080} + \frac{628051206649}{351035882626518720} - j\,\frac{10047948805801}{117011960875506240}$$

$$+ \frac{628051206649}{351035882626518720} + j\,\frac{4326627}{650066449308368} + \frac{40507401742253}{39003986958502080} + j\,\frac{13538115}{650066449308368}$$

$$- j\,\frac{916336386251}{23402392175101248} + \frac{129963846649}{351035882626518720} + \frac{72664239152969}{70207176525303744} + \frac{129963846649}{351035882626518720}$$

$$= \frac{12152398472147}{2925299021887656} - j\,\frac{1828301885437}{14626495109438280}$$

On a bien $\underline{S} = \sum \underline{S_{ij}}$

Mathématique & Informatique

Développement en fraction continue des nombres complexes

Etudions les nombres complexes z dont la partie réelle x et la partie imaginaire y sont des nombres rationnels :

$$z = x + i\,y$$

avec $x, y \in \mathbb{Q}$

Exprimons x et y sous la forme d'un rapport de 2 nombres entiers :

$$z = \frac{x_1}{x_2} + i\frac{y_1}{y_2} = \frac{x_1 y_2 + i\,x_2 y_1}{x_2 y_2} = \frac{A + i\,B}{C}$$

avec A, B, C $\in \mathbb{Z}$

Soit a_0 la partie entière de A/C et b_0 la partie entière de B/C.
Convenons que la partie entière d'un nombre complexe est constituée de la partie entière de sa partie réelle et de la partie entière de sa partie imaginaire :

$$E(z) = E(x) + i\,E(y) = a_0 + i\,b_0$$

Par analogie avec les nombres quadratiques, décomposons le nombre z :

$$z = (a_0 + i\,b_0) + \frac{1}{z'}$$

$$z' = \frac{1}{\left(\frac{A}{C} - a_0\right) + i\left(\frac{B}{C} - b_0\right)} = \frac{\left(\frac{A}{C} - a_0\right) - i\left(\frac{B}{C} - b_0\right)}{\left(\frac{A}{C} - a_0\right)^2 + \left(\frac{B}{C} - b_0\right)^2} = \frac{A' + iB'}{C'} = x' + i\,y'$$

Notons : $x + i\,y = x^{(0)} + i\,y^{(0)} = z^{(0)}$; $x' + i\,y' = x^{(1)} + i\,y^{(1)} = z^{(1)}$; $x'' + i\,y'' = x^{(2)} + i\,y^{(2)} = z^{(2)}$; etc, ... Il vient :

$$x + i\,y = (a_0 + i\,b_0) + \frac{1}{x^{(1)} + i\,y^{(1)}} = (a_0 + i\,b_0) + \frac{1}{(a_1 + i\,b_1) + \frac{1}{x^{(2)} + i\,y^{(2)}}} = \cdots$$

$$= (a_0 + i\,b_0) + \cfrac{1}{(a_1 + i\,b_1) + \cfrac{1}{(a_2 + i\,b_2) + \ddots + \cfrac{1}{x^{(n)} + i\,y^{(n)}}}}$$

Remarque

Les nombres complexes $a_k + i\,b_k$ (k > 0) sont des entiers de **Gauss** non nuls dont la partie réelle est positive ($a_k \geq 0$) et la partie imaginaire négative ($b_k \leq 0$). Le corps des nombres complexes étant commutatif, on écrira indifféremment $a_k + i\,b_k$ ou $a_k + b_k\,i$.

Deux cas peuvent se présenter :

1) Le développement est fini : on aboutit à un couple $(x^{(n)}, y^{(n)}) \in \mathbb{Z}^2$
Dans ce cas, le nombre complexe z se met immédiatement sous la forme :

$$z = \left[a_0 + ib_0, a_1 + ib_1, \cdots, a_n + ib_n\right]$$
$$= a_0 + ib_0 + \cfrac{1}{a_1 + ib_1 + \ddots + \cfrac{1}{a_n + ib_n}}$$

2) Le développement est infini : on aboutit à un couple $(x^{(n)}, y^{(n)}) \in \mathbb{Q}^2 - \mathbb{Z}^2$ qui est déjà apparu dans le calcul de la fraction continue, plus précisément :

$$\exists\, r > 0 \;/\; x^{(n)} + i\, y^{(n)} = x^{(r+3)} + i\, y^{(r+3)} = x^{(r)} + i\, y^{(r)}$$

Les 3 parties entières successives du développement en fraction continue du nombre complexe $z^{(r)} = x^{(r)} + i\, y^{(r)}$ sont égales à -i. On a :

$$z^{(r)} = -i + \cfrac{1}{-i + \cfrac{1}{-i + \cfrac{1}{z^{(r)}}}}$$

Par suite, la fraction continue de z est illimitée et tous ses termes sont égaux à -i à partir du rang r :

$$z = \left[\, a_0 + ib_0,\ a_1 + ib_1,\ \cdots,\ a_{r-1} + ib_{r-1}, -i, -i, -i, \cdots \right]$$

Si l'on souhaite obtenir un développement fini équivalent à z, on ne prend pas la valeur -i de la partie entière du nombre complexe $z^{(r+3)}$ au rang r+3, mais la valeur 1 - i :

$$a_{r+3} + i\, b_{r+3} = 1 - i$$

La fraction continue n'est plus périodique au rang r.
En appliquant ce procédé à chaque nouvelle périodicité, on est ramené au cas 1)

Remarque

Le développement en fraction continue d'un nombre complexe n'est pas unique.

Fonctions Scilab

Soit z un nombre complexe à coefficients rationnels, la fonction *fraction_complexe* renvoie la variable fc (liste de nombres complexes) égale au développement en fraction continue complexe de z à l'ordre k maximum.

```
function fc = fraction_complexe(z,k)
  fc = list()
  Z = list()
  for i = 0:k
    printf('\n z%i = ',i), ECRIRE(z)
    [Q1,R1] = PARTIE_ENTIERE(z(1),z(2))
    [Q2,R2] = PARTIE_ENTIERE(z(3),z(4))
    Z($+1) = z
    if i >= 3 & C_COMPARAISON(z,Z(1)) == 0 then
      Q1 = ADDITION(Q1,ENTIER(1))
      R1 = SOUSTRACTION(R1,z(2))
    end
    if i >= 3 then, Z(1) = null(), end
    fc($+1) = list(Q1,ENTIER(1),Q2,ENTIER(1))
    if NUL(R1) & NUL(R2) then, break, end
    u = F_SOUSTRACTION(REELLE(z),list(Q1,[1,1]))
    v = F_SOUSTRACTION(IMAGINAIRE(z),list(Q2,[1,1]))
    z = C_INVERSION(list(u(1),u(2),v(1),v(2)))
  end
endfunction
```

Soit fc une fraction continue complexe, la fonction *reduite_complexe* renvoie la variable fr (nombre complexe) égale à la réduite d'ordre k de fc.

```
function fr = reduite_complexe(fc,k)
  if k >= length(fc) then
    fr = list() ; return
  end
  pim2 = COMPLEXE(1,1,0,1) ; qim2 = COMPLEXE(0,1,0,1)
  pim1 = fc(1) ; qim1 = COMPLEXE(1,1,0,1)
  for i = 1:k
    fi = fc(i+1)
    pi = C_ADDITION(C_MULTIPLICATION(fi,pim1),pim2)
    qi = C_ADDITION(C_MULTIPLICATION(fi,qim1),qim2)
    pim2 = pim1 ; qim2 = qim1
    pim1 = pi ; qim1 = qi
  end
  fr = C_DIVISION(pim1,qim1)
endfunction
```

Mathématique & Informatique

EXEMPLE 1

Développer en fraction continue les 2 nombres complexes suivants :

$$z_1 = -\frac{565521805017353}{43796948603271} + i\,\frac{117452125914449}{14598982867757}$$

$$z_2 = \frac{654841}{64484} + i\,\frac{65165}{165151}$$

Programme de calcul pour développer en fraction continue un nombre complexe quelconque :

```
while %t
  a = input('a = ?') ; if a == [] then, break, end
  b = input('b = ?') ; if b == [] then, break, end
  c = input('c = ?') ; if c == [] then, break, end
  d = input('d = ?') ; if d == [] then, break, end
  k = input('k = ?') ; if k == [] then, break, end
  z = COMPLEXE(a,b,c,d) ;
  fz = fraction_complexe(z,k) ;
  printf('\n '), ECRIRE(z), printf(' = '), FC_ECRIRE(fz)
end
```

$a = -565521805017353,\quad b = 43796948603271,\quad c = 117452125914449,\quad d = 14598982867757,\quad k = 100$

$$z_1 = -\frac{565521805017353}{43796948603271} + i\,\frac{117452125914449}{14598982867757} = [-13+i8, 9-i5, -i3, 2-i6, -i, 1-i, 6-i6, 17-i, 4-i15, 1-i, 7-i19]$$

$$= -13+8i+\cfrac{1}{9-5i+\cfrac{1}{-3i+\cfrac{1}{2-6i+\cfrac{1}{-i+\cfrac{1}{1-i+\cfrac{1}{6-6i+\cfrac{1}{17-i+\cfrac{1}{4-15i+\cfrac{1}{1-i+\cfrac{1}{7-19i}}}}}}}}}}$$

$a = 654841,\quad b = 64484,\quad c = 65165,\quad d = 165151,\quad k = 100$

$$z_2 = \frac{654841}{64484} + i\,\frac{65165}{165151}$$

$$= [10, -i3, -i, 1-i, 1-i4, -i2, -i, 1-i, 2-i3, 4-i2, -i, -i, -i, 1-i, -1-i2, -i2, 1-i, -i2, -i, 1-i, 1-i, -i,$$
$$2-i, 1-i4, -i, 1-i, 1-i, -i, 1-i, -i2, -i, 1-i, -i, -i, -i, 1-i, -1-i2, 1-i, 1-i, -i, 1-i, -i, -i, -i, 1-i, -1-i2,$$
$$-i, -i, -i, 1-i, -1-i3, -i, 1-i, -i2, -i, 1-i, -i2, -i2, -i2, -i2, -i2, -i, 3-i, 1-i, -i2, -i, 1-i, 2-i2]$$

Mathématique & Informatique

EXEMPLE 2

- Développer en fraction continue le grand nombre complexe :

$$z = \frac{A + i\,B}{C} \qquad avec \ :$$

$$A = 4370558304961542821137577945872421236353067245243435639$$

$$B = -\ 327469242209610831377294582282424229825864776800381764 88$$

$$C = 4528685388263201585489962786799598337477540612064977$$

Programme :

a = [1,4,3,7,0,5,5,8,3,0,4,9,6,1,5,4,2,8,2,1,1,3,7,5,7,7,9,4,5,8,7,2,4,2,1,2,3,6,3,5,3,0,6,7,2,4,5,2,4,3,4,3,5,6,3,9] ;
b = [1,4,5,2,8,6,8,5,3,8,8,2,6,3,2,0,1,5,8,5,4,8,9,9,6,2,7,8,6,7,9,9,5,9,8,3,3,7,4,7,7,5,4,0,6,1,2,0,6,4,9,7,7] ;
c = [-1,3,2,7,4,6,9,2,4,2,2,0,9,6,1,0,8,3,1,3,7,7,2,9,4,5,8,2,2,8,2,4,2,4,2,2,9,8,2,5,8,6,4,7,7,6,8,0,0,3,8,1,7,6,4,8,8] ;
d = [1,4,5,2,8,6,8,5,3,8,8,2,6,3,2,0,1,5,8,5,4,8,9,9,6,2,7,8,6,7,9,9,5,9,8,3,3,7,4,7,7,5,4,0,6,1,2,0,6,4,9,7,7] ;
z = list(a,b,c,d) ;
fz = fraction_complexe(z,100) ;
FC_ECRIRE(fz)

Résultat :

```
[965 - i7232, - i, 12 - i, 57 - i21, 63 - i2, 1 - i, 1 - i, 489 - i15, 10 - i32, 1 - i2, 5 - i88,
- i13, 41 - i9, - i6, - i6, - i6, 111 - i27, 2 - i, 3 - i7, 17 - i3, 63 - i12, 9 - i10, 15 - i61]
```

Ce développement nécessite un temps de calcul d'environ 15 mn avec un micro-ordinateur.

- Développer en fraction continue le nombre complexe :

$$z = \frac{568}{547} + i\,\frac{61}{13}$$

Appelons la fonction *fraction_complexe* :

fz = fraction_complexe(COMPLEXE(568,547,61,13),1000) ;
FC_ECRIRE(fz)

On obtient une fraction continue à 262 termes :

```
[1 + i4, - i2, - i2, 2 - i3, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2,
- i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2,
- i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2,
- i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2,
- i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2,
- i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2,
- i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2,
- i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2,
- i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2,
- i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2,
- i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2,
- i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2, - i2,
- i2, - i2, - i2, - i2, - i2, - i2, - i, 104 - i, 1 - i, - i2, - i2, 1 - i]
```

Un petit nombre complexe peut donner lieu à un long développement en fraction continue.

Annexe : fonction d'écriture d'une fraction continue

Une fraction continue ordinaire est une liste d'entiers relatifs qui sont des vecteurs de chiffres.
Une fraction continue généralisée est une liste de nombres rationnels qui sont des listes de 2 vecteurs de chiffres.
Une fraction continue complexe est une liste de nombres complexes qui sont des listes de 4 vecteurs de chiffres.
La fonction *FC_ECRIRE* affiche à l'écran la valeur d'une fraction continue sous la forme [a_0 , a_1 , a_2 , ... , a_n] s'il s'agit d'une fraction continue ordinaire, ou sous la forme [a_0 ; b_1/a_1 , b_2/a_2 , ... , b_n/a_n] s'il s'agit d'une fraction continue généralisée, ou encore sous la forme [$a_0 + i\,b_0$, $a_1 + i\,b_1$, ... , $a_n + i\,b_n$] s'il s'agit d'une fraction continue complexe.

```
function FC_ECRIRE(fc)
  printf('[')
  for i = 1:length(fc)
    e = fc(i)
    if typeof(e) == "constant" then
      if i > 1 then, printf(', '), end
      ECRIRE(e)
    end
    if typeof(e) == "list" then
      if length(e) == 2 then
        if i == 2 then, printf(' ; '), end
        if i > 2 then, printf(', '), end
        ECRIRE(e)
      end
      if length(e) == 4 then
        if i > 1 then, printf(', '), end
        ECRIRE(e)
      end
    end
  end
  printf(']')
endfunction
```

Exemple

Fraction continue ordinaire :

```
fc1 = list(ENTIER(-18), ENTIER(1), ENTIER(2), ENTIER(9)) ;
FC_ECRIRE(fc1)
```

affiche à la Console : [- 18, 1, 2, 9]

Fraction continue généralisée :

```
fc2 = list(ENTIER(0), FRACTION(1,2), FRACTION(-7,65)) ;
FC_ECRIRE(fc2)
```

affiche à la Console : [0 ; 1/2, - 7/65]

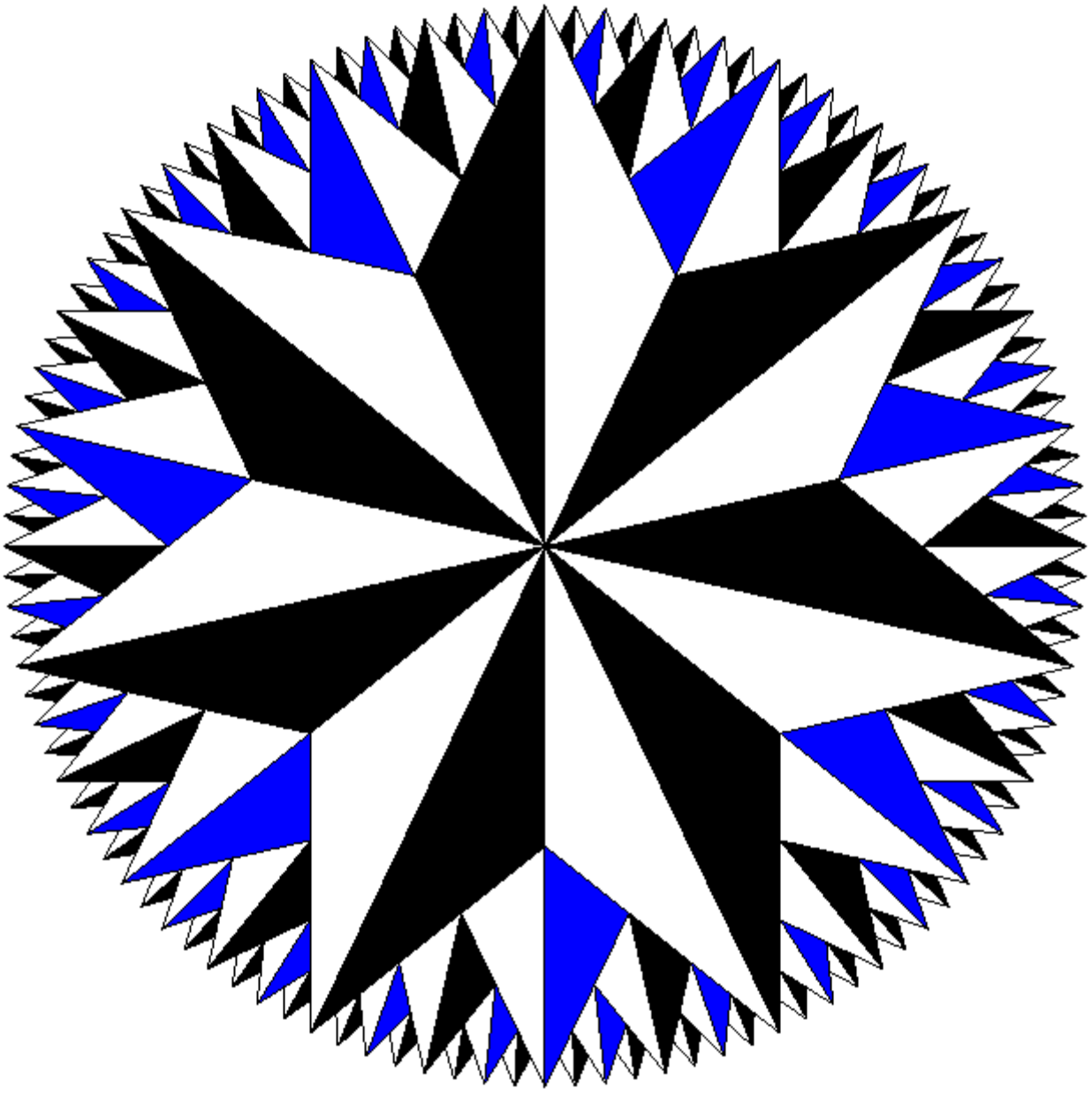
Fraction continue complexe :

```
fc3 = list(COMPLEXE(-1,1,3,1),COMPLEXE(2,1,-1,1),COMPLEXE(21,1,-13,1)) ;
FC_ECRIRE(fc3)
```

affiche à la Console : [- 1 + i3, 2 - i, 21 - i13]

Mathématique & Informatique

FRACTALES

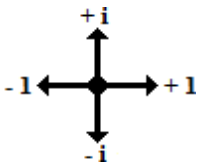


Mathématique & Informatique

Lire le mémoire à l'adresse : ⇨ **http://www.3dfractals.com/docs/Master_Thesis_Lajoie.pdf** (Josiane **Lajoie)**

Courbe de Hilbert

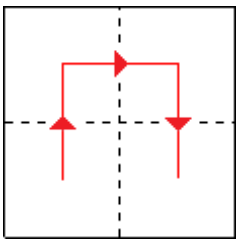
Cette courbe fractale conduit à une représentation cartésienne paramétrée (x(t),y(t)) qui remplit entièrement le carré [0,1] × [0,1]. Plus exactement, elle permet de définir une application continue surjective de [0,1] dans [0,1] × [0,1]. Pour étudier plus facilement la courbe de **Hilbert** d'ordre n, changeons d'échelle et plaçons nous dans le carré de côté $[0,2^n]$. On peut assimiler cette courbe plane à une suite de déplacements élémentaires représentés par des nombres complexes $z_k \in \{ -1 , +1 , -i , +i \}$:



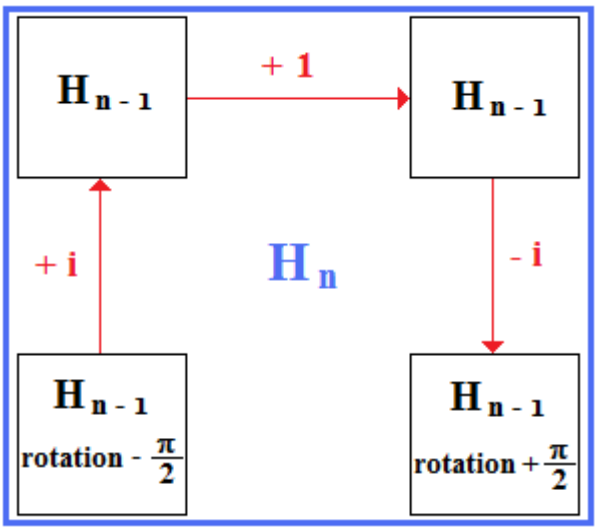
Ainsi, la courbe de **Hilbert** H_n d'ordre n est un vecteur complexe de dimension $4^n - 1$ dont les éléments z_k sont soit l'unité réelle au signe près, soit l'unité imaginaire au signe près :

$$H_n = (z_1 , z_2 , z_3 , \ldots , z_{4^n - 1})$$

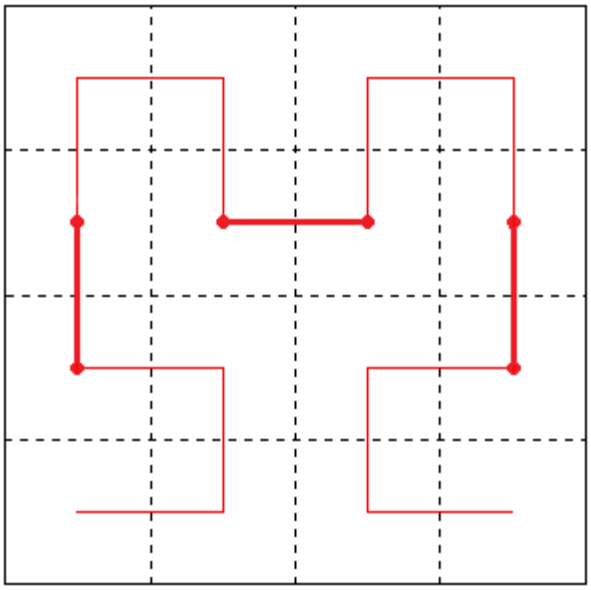
La première courbe de **Hilbert** est $H_1 = (i , 1 , -i)$; elle correspond au dessin de base :



Le schéma récurrent liant H_n à H_{n-1} est le suivant :



Pour n = 2, on obtient la courbe de **Hilbert** H_2 :



Remarquons qu'effectuer une rotation d'angle - π/2 sur H_{n-1} revient à permuter ses parties réelles et imaginaires et qu'effectuer une rotation d'angle + π/2 sur H_{n-1} revient à permuter ses parties réelles et imaginaires en inversant leurs signes. On déduit un algorithme itératif très simple pour calculer H_n :

```
function H = Hilbert(n)
    H = []
    for k = 1:n
        HP = imag(H) + %i*real(H)
        H = [HP,%i,H,1,H,-%i,-HP]
    end
endfunction
```

De proche en proche, on trouve :

Hilbert(1) = (i , 1 , -i)

Hilbert(2) = (1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1)

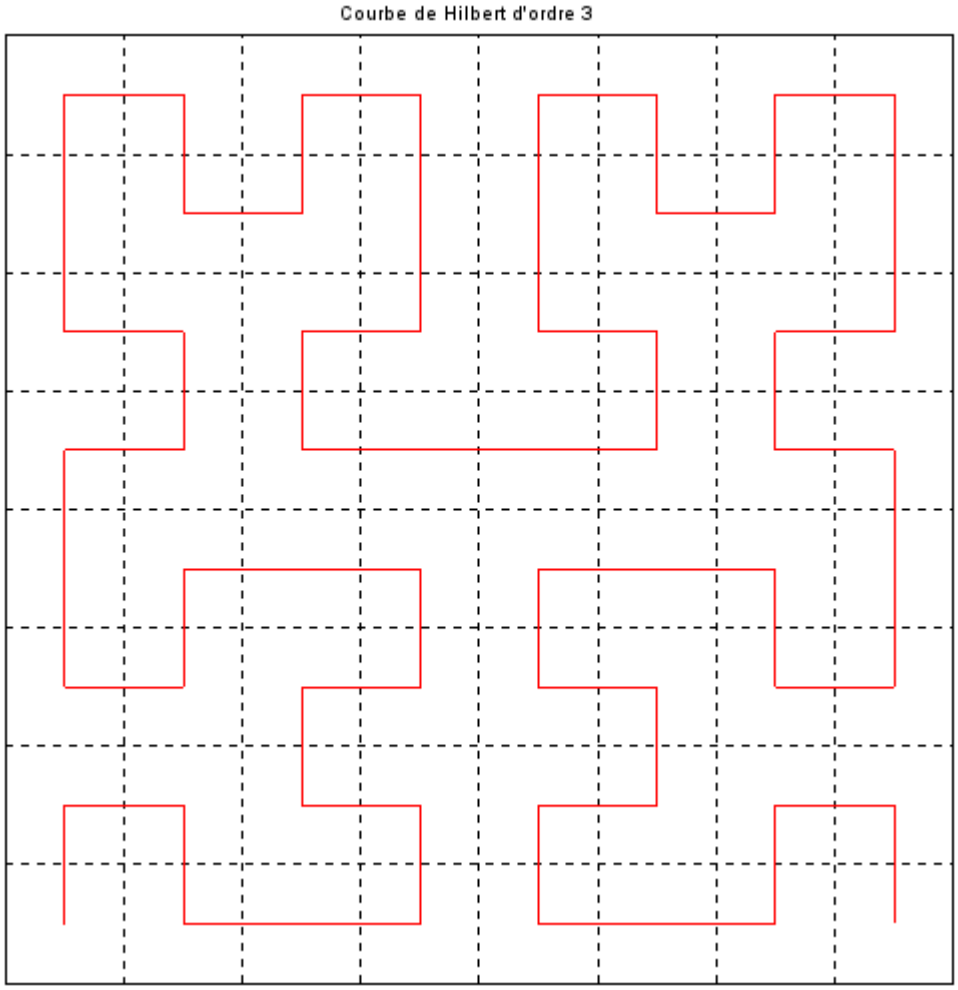
Hilbert(3) = (i , 1 , -i , 1 , 1 , i , -1 , i , 1 , i , -1 , -1 , -i , -1 , i , i , 1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i ,
1 , 1 , 1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1 , -i , -i , -1 , i , -1 , -1 , -i , 1 , -i , -1 , -i , 1 , 1 , i , 1 , -i)

Hilbert(4) = (1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1 , 1 , i , 1 , -i , 1 , 1 , i , -1 , i , 1 , i , -1 , -1 , -i ,
-1 , i , i , i , 1 , -i , 1 , 1 , i , -1 , i , 1 , i , -1 , -1 , -i , -1 , i , -1 , -1 , -i , 1 , -i , -i , -1 , i , -1 , -i , -1 , i , i , 1 , i ,
-1 , i , i , 1 , -i , 1 , 1 , i , -1 , i , 1 , i , -1 , -1 , -i , -1 , i , i , 1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1 , 1 , 1 , i ,
1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1 , -i , -i , -1 , i , -1 , -1 , -i , 1 , -i , -1 , -i , 1 , 1 , i , 1 , -i , 1 , i ,
1 , -i , 1 , 1 , i , -1 , i , 1 , i , -1 , -1 , -i , -1 , i , i , 1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1 , 1 , 1 , i ,
-1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1 , -i , -i , -1 , i , -1 , -1 , -i , 1 , -i , -1 , -i , 1 , 1 , i , 1 , -i , -i , -1 , -i ,
1 , -i , -i , -1 , i , -1 , -i , -1 , i , i , 1 , i , -1 , -1 , -i , -1 , i , -1 , -1 , -i , 1 , -i , -1 , -i , 1 , 1 , i , 1 , -i , -i , -i , -1 ,
i , -1 , -1 , -i , 1 , -i , -1 , -i , 1 , 1 , i , 1 , -i , 1 , 1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1)

Graphique

Il suffit de diviser les éléments de H_n par 2^n pour se ramener au domaine carré $[0,1] \times [0,1]$. Si l'on souhaite placer la courbe de **Hilbert** d'ordre n au centre d'un quadrillage comportant 4^n petits carrés, on part du milieu du premier petit carré, c'est-à-dire du point d'affixe $z = (1/2 + i/2)/2^n$, puis on trace les $4^n - 1$ segments de la ligne brisée, les uns à la suite des autres, dans l'ordre croissant des éléments du vecteur H_n.
La longueur de la courbe de **Hilbert** d'ordre n est : $L = (4^n - 1) \times 1/2^n = 2^n - 1/2^n$
Le programme ci-dessous dessine la courbe de **Hilbert** d'ordre 3 (par exemple) :

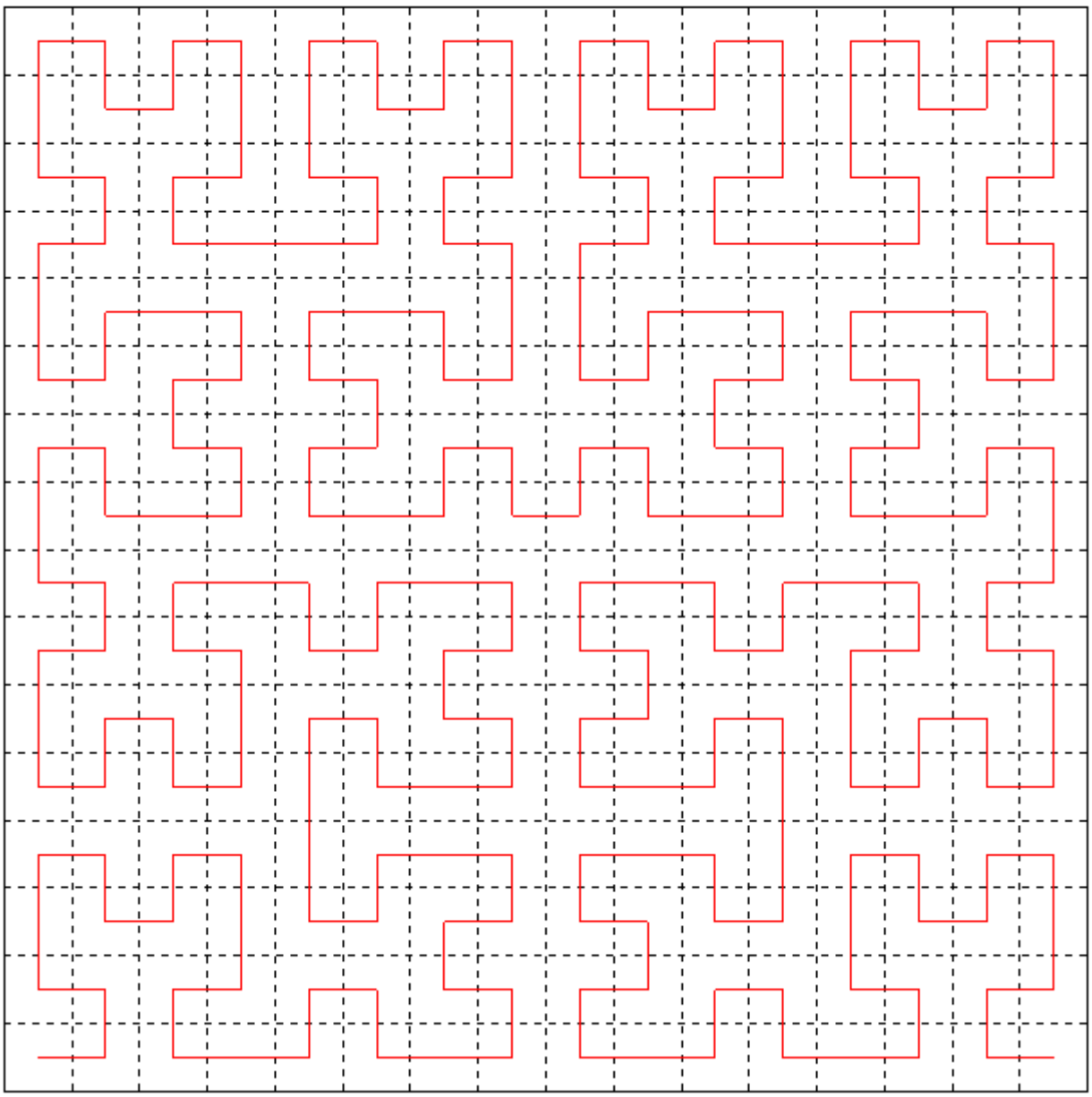
```
n = 3 ;
xtitle('Courbe de Hilbert d''ordre '+ string(n))
plot2d(%inf,%inf,frameflag=3,rect=[0,0,1,1],nax=[0,2^n+1,0,2^n+1],axesflag=2)
xgrid(1)
drawlater()
H = Hilbert(n) ;
z = (1/2 + %i/2)/2^n;
for k = 1:length(H)
    seg = [z ; z + H(k)/2^n] ;
    xsegs(real(seg),imag(seg),5)
    z = seg(2) ;
end
drawnow()
```



Mathématique & Informatique

Hilbert(4) = (1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1 , 1 , i , 1 , -i , 1 , 1 , i , -1 , i , 1 , i , -1 , -1 , -i , -1 , i , i , i ,
1 , -i , 1 , 1 , i , -1 , i , 1 , i , -1 , -1 , -i , -1 , i , -1 , -1 , -i , 1 , -i , -i , -1 , i , -1 , -i , -1 , i , i , 1 , i , -1 , i , i , 1 , -i , 1 , 1 , i ,
-1 , i , 1 , i , -1 , -1 , -i , -1 , i , i , 1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1 , 1 , 1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i ,
-1 , -i , 1 , -i , -i , -1 , i , -1 , -1 , -i , 1 , -i , -1 , -i , 1 , 1 , i , 1 , -i , 1 , i , 1 , -i , 1 , 1 , i , -1 , i , 1 , i , -1 , -1 , -i , -1 , i , i , 1 ,
i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1 , 1 , 1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1 , -i , -i , -1 , i , -1 , -1 , -i ,
1 , -i , -1 , -i , 1 , 1 , i , 1 , -i , -i , -1 , -i , 1 , -i , -i , -1 , i , -1 , -i , -1 , i , i , 1 , i , -1 , -1 , -i , 1 , -i , -1 , -i ,
1 , 1 , i , 1 , -i , -i , -i , -1 , i , -1 , -1 , -i , 1 , -i , -1 , -i , 1 , 1 , i , 1 , -i , 1 , 1 , i , -1 , i , i , 1 , -i , 1 , i , 1 , -i , -i , -1 , -i , 1)

Courbe de Hilbert d'ordre 4

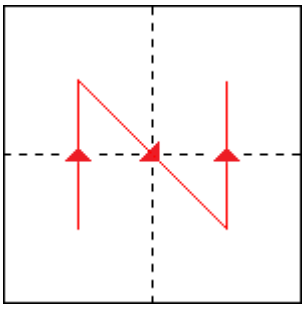


Courbe de Lebesgue

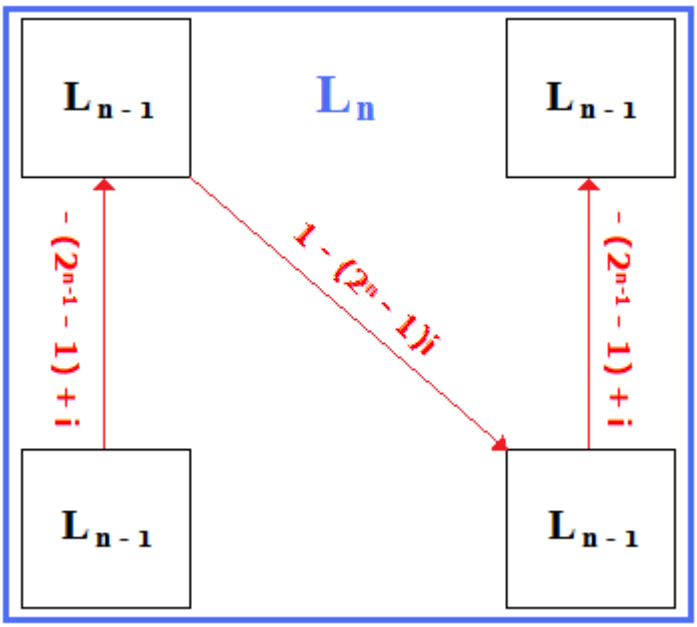
Cette courbe fractale est une ligne brisée similaire à la courbe de **Hilbert** mais elle possède en plus la propriété d'être dérivable en "presque" tous ses points. Comme précédemment, on peut représenter la courbe de **Lebesgue** d'ordre n par un vecteur complexe de dimension $4^n - 1$ (dont les éléments z_k ne sont pas nécessairement ± 1 ou ± i cette fois) :

$$L_n = (z_1 , z_2 , z_3 , \dots , z_{4^n-1})$$

On part de la courbe d'ordre 1 définie par $L_1 = (i , 1 - i , i)$:



Puis on applique le schéma récurrent suivant :



On déduit la fonction de calcul de L_n :

```
function L = Lebesgue(n)
    L = []
    for k = 1:n
        L = [L,-(2^(k-1)-1)+%i,L,1-(2^k-1)*%i,L,-(2^(k-1)-1)+%i,L]
    end
endfunction
```

De proche en proche, on trouve :

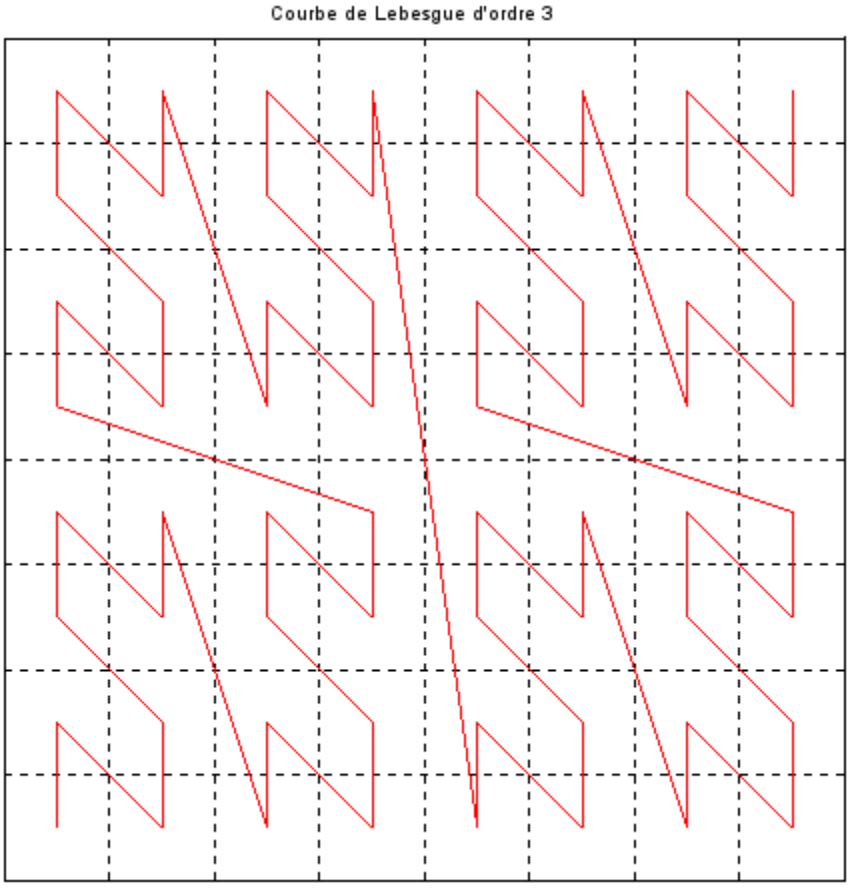
Lebesgue(1) = (i , 1 - i , i)

Lebesgue(2) = (i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i)

Lebesgue(3) = (i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , -3 + i , i , 1 - i , i ,
-1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 7i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i ,
1 - i , i , -1 + i , i , 1 - i , i , -3 + i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i)

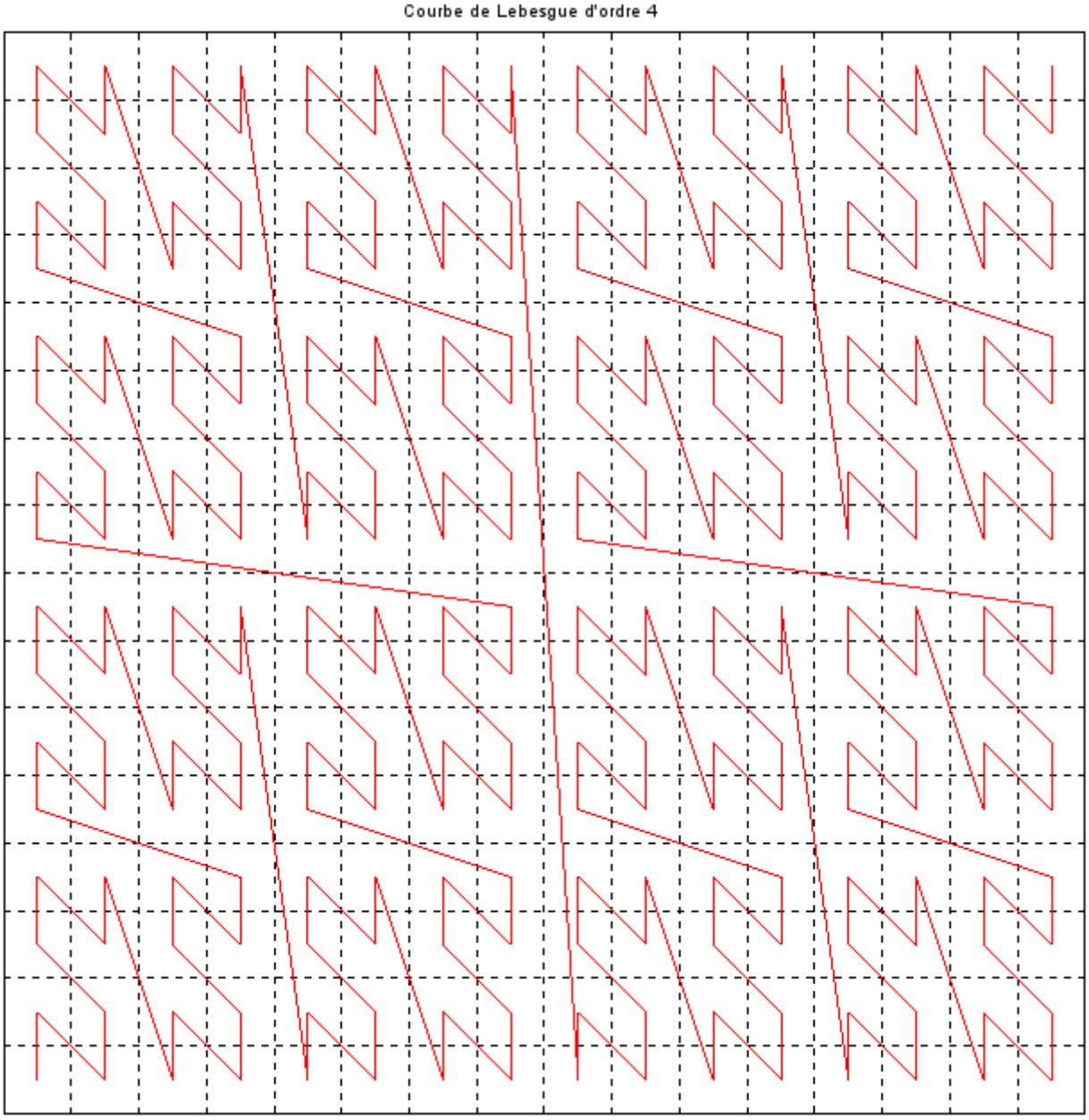
Le programme ci-dessous dessine la courbe de **Lebesgue** d'ordre 3 (par exemple) :

```
n = 3 ;
xtitle('Courbe de Lebesgue d''ordre '+ string(n))
plot2d(%inf,%inf,frameflag=3,rect=[0,0,1,1],nax=[0,2^n+1,0,2^n+1],axesflag=2)
xgrid(1)
drawlater()
L = Lebesgue(n) ;
z = (1/2 + %i/2)/2^n ;
for k = 1:length(L)
    seg = [z ; z + L(k)/2^n] ;
    xsegs(real(seg),imag(seg),5)
    z = seg(2) ;
end
drawnow()
```



Mathématique & Informatique

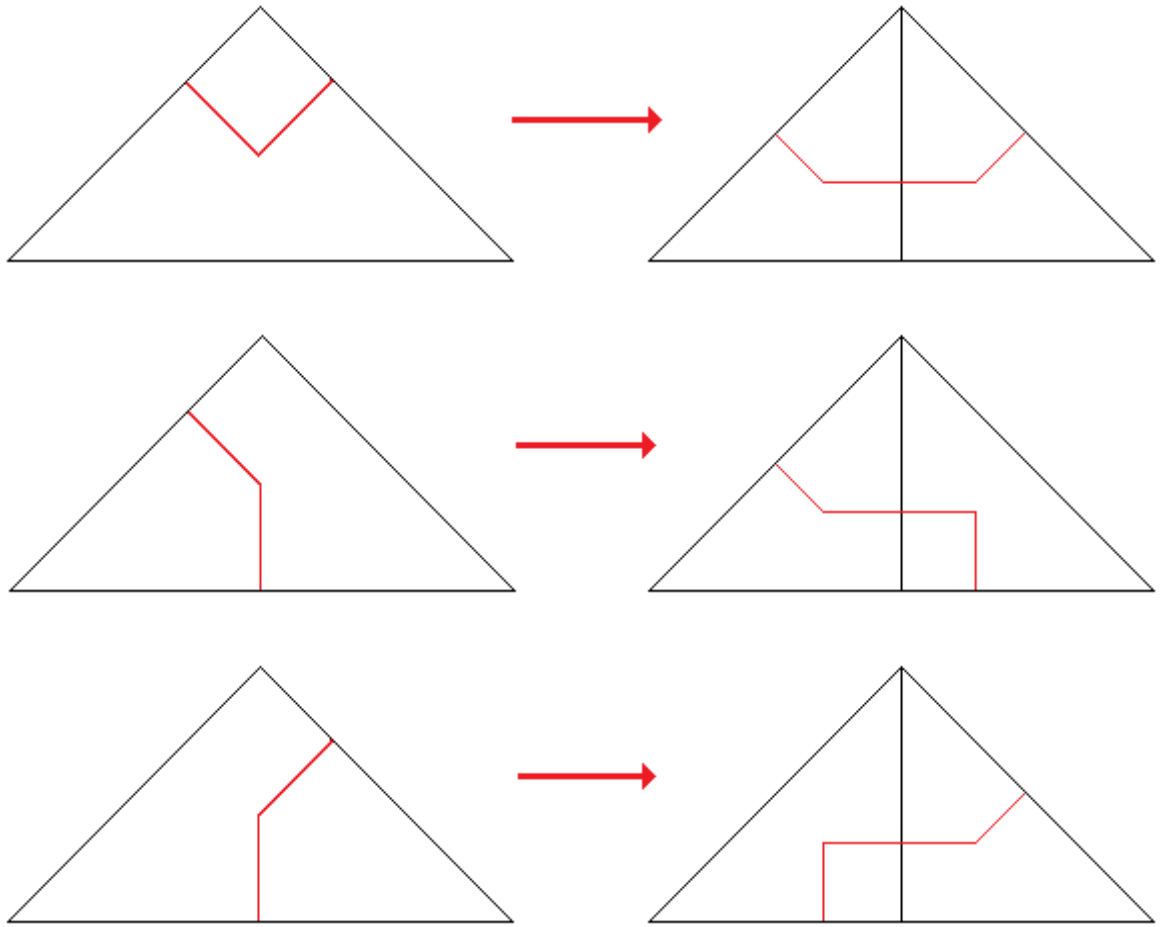
Lebesgue(4) = (i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , -3 + i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 7i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , -3 + i , i , 1 - i , i , -1 + i , i , 1 - i , i , -7 + i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , -3 + i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 7i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , -3 + i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 15i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , -3 + i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 7i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , -3 + i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , -7 + i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , -3 + i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 7i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i , -3 + i , i , 1 - i , i , -1 + i , i , 1 - i , i , 1 - 3i , i , 1 - i , i , -1 + i , i , 1 - i , i)



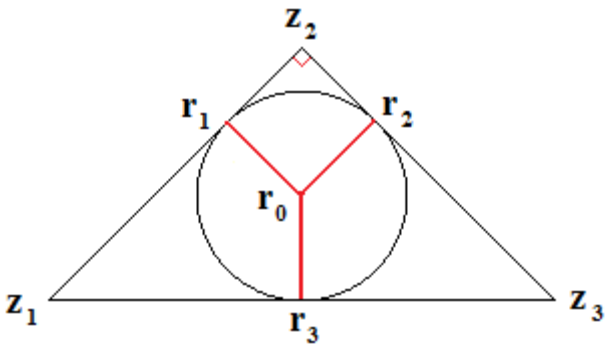
Mathématique & Informatique

Courbe de Sierpinski

Cette courbe fractale se construit en itérant (à l'infini) le processus de division suivant :



Le triangle rectangle isocèle initial est divisé en 2 triangles rectangles isocèles similaires.
Dans chaque nouveau triangle, on trace 2 rayons du cercle inscrit perpendiculairement à 2 côtés, en se référant à l'un des 3 cas ci-dessus. Appelons z_1, z_2, z_3, r_0, r_1, r_2, r_3 les affixes des points caractéristiques d'un triangle :



Notons T le vecteur colonne des 3 affixes des sommets du triangle rectangle : $T = [z_1 ; z_2 ; z_3]$
Notons r le vecteur colonne indicateur des 3 rayons du cercle inscrit dans le triangle rectangle : $r = [i_1 ; i_2 ; i_3]$
$i_k = 1$ si le rayon d'extrémités r_0 et r_k figure sur le schéma du triangle T et 0 sinon.

On établit que dans un triangle rectangle isocèle dont l'hypoténuse vaut c, les côtés valent $c/\sqrt{2}$, la hauteur vaut c/2, le rayon du cercle inscrit vaut $c/(2+2\sqrt{2})$. L'affixe r_0 du centre du cercle inscrit et les affixes r_1, r_2, r_3 des extrémités des rayons se déduisent des affixes z_1, z_2, z_3 des sommets du triangle rectangle isocèle par les formules suivantes :

$$\begin{cases} r_0 = (1-1/\sqrt{2})\,z_1 + (\sqrt{2}-1)\,z_2 + (1-1/\sqrt{2})\,z_3 \\ r_1 = (1-1/\sqrt{2})\,z_1 + (1/\sqrt{2})\,z_2 \\ r_2 = (1/\sqrt{2})\,z_2 + (1-1/\sqrt{2})\,z_3 \\ r_3 = (1/2)\,z_1 + (1/2)\,z_3 \end{cases}$$

Nous avons tous les éléments pour programmer une fonction *Sierpinski* qui dessine la courbe d'ordre n (≥ 0) :

```
function Sierpinski(T,r,n)
    if n > 0 then
        z = (T(1) + T(3))/2
        if r(1) == 0 then
            Sierpinski([T(3);z;T(2)],[0;1;1],n-1)
            Sierpinski([T(2);z;T(1)],[1;1;0],n-1)
        end
        if r(2) == 0 then
            Sierpinski([T(3);z;T(2)],[1;1;0],n-1)
            Sierpinski([T(2);z;T(1)],[1;0;1],n-1)
        end
        if r(3) == 0 then
            Sierpinski([T(3);z;T(2)],[0;1;1],n-1)
            Sierpinski([T(2);z;T(1)],[1;0;1],n-1)
        end
    else
        r0 = (1-1/sqrt(2))*T(1) + (sqrt(2)-1)*T(2) + (1-1/sqrt(2))*T(3)
        r1 = (1-1/sqrt(2))*T(1) + (1/sqrt(2))*T(2)
        r2 = (1/sqrt(2))*T(2) + (1-1/sqrt(2))*T(3)
        r3 = (1/2)*T(1) + (1/2)*T(3)
        S1 = [r0;r1] ; S2 = [r0;r2] ; S3 = [r0;r3]
        if r(1) == 1 then, xsegs(real(S1),imag(S1),5), end
        if r(2) == 1 then, xsegs(real(S2),imag(S2),5), end
        if r(3) == 1 then, xsegs(real(S3),imag(S3),5), end
    end
endfunction
```

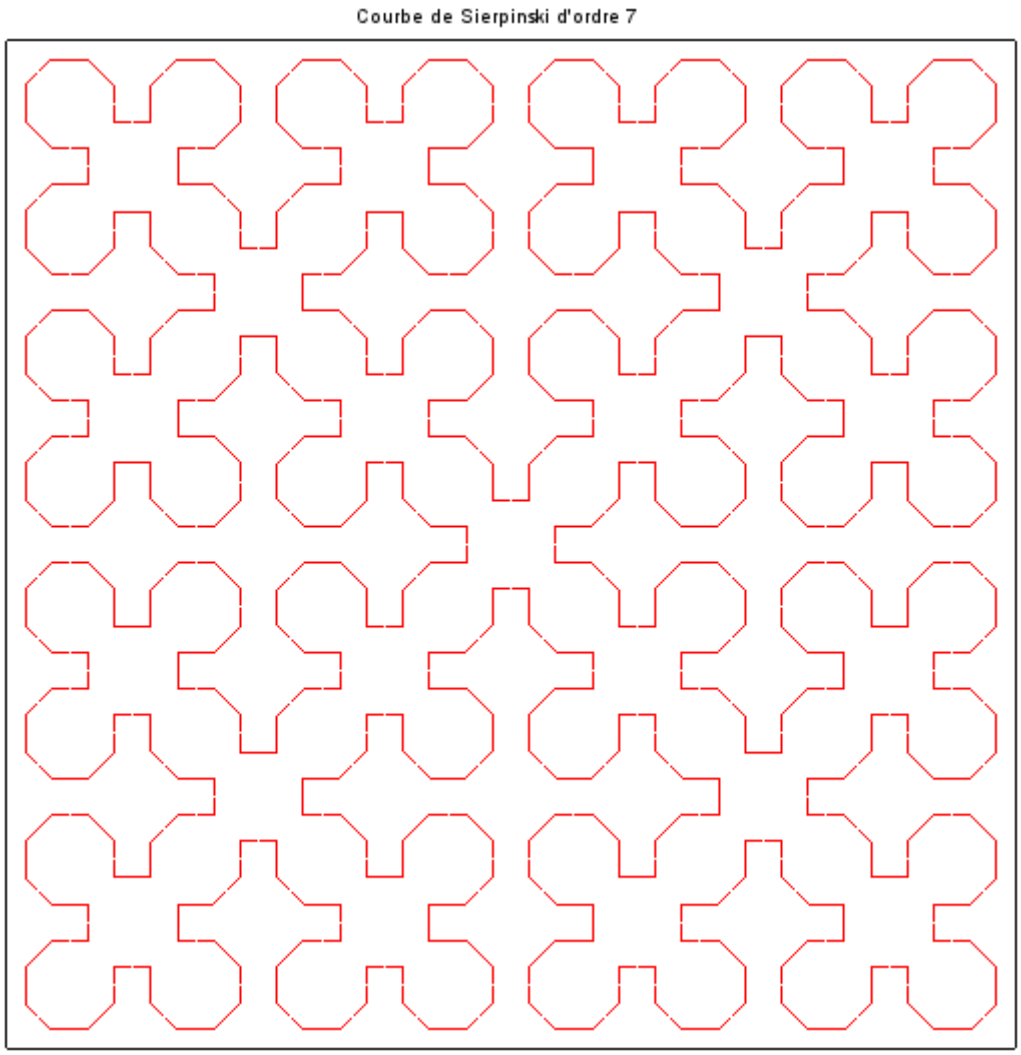
Longueur de la courbe de Sierpinski

Calculons la longueur L de la courbe fermée de **Sierpinski** inscrite dans un carré de côté c dont les diagonales délimitent 4 triangles rectangles isocèles. A l'ordre n, le carré initial est divisé en 4×2^n petits triangles rectangles d'hypoténuse $c/(\sqrt{2})^n$ et de rayon inscrit $c/(2+2\sqrt{2})(\sqrt{2})^n$. Par suite :

$$L = 4\times 2^n \times 2 \times \frac{c}{(2+2\sqrt{2})(\sqrt{2})^n} = \frac{2^{\frac{n}{2}+2}}{1+\sqrt{2}}\,c = 2^{\frac{n}{2}+2}\,(\sqrt{2}-1)\,c$$

Considérons le carré unité [0,1] × [0,1] dont les affixes des sommets sont les complexes $z_1 = 0$, $z_2 = i$, $z_3 = 1+i$, $z_4 = 1$, et dont le centre est le complexe $z_0 = (1+i)/2$. La courbe de **Sierpinski** d'ordre 7 (par exemple) s'obtient en appelant la fonction *Sierpinski* pour les 4 triangles rectangles isocèles constituant le carré de départ :

```
n = 7 ;
drawlater()
xtitle('Courbe de Sierpinski d''ordre '+ string(n))
plot2d(%inf,%inf,frameflag=3,rect=[0,0,1,1],axesflag=2)
z0 = (1+%i)/2 ; z1 = 0 ; z2 = %i ; z3 = 1+%i ; z4 = 1 ;
Sierpinski([z1;z0;z4],[1;1;0],n) ;
Sierpinski([z4;z0;z3],[1;1;0],n) ;
Sierpinski([z3;z0;z2],[1;1;0],n) ;
Sierpinski([z2;z0;z1],[1;1;0],n) ;
drawnow()
```

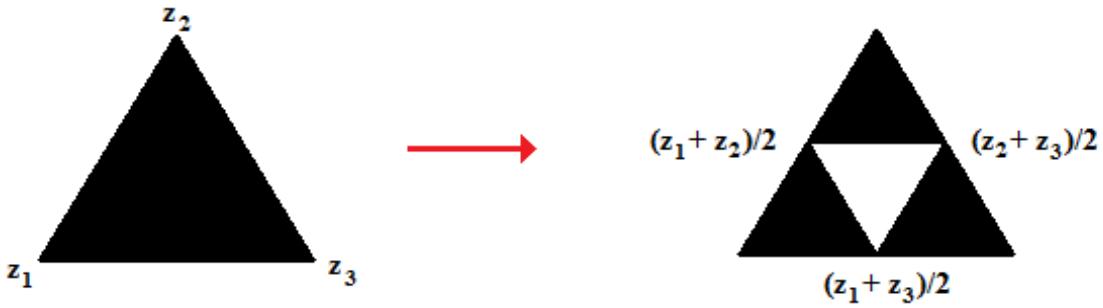


Remarque

Lorsque n tend vers l'infini, la courbe fractale de **Sierpinski** remplit le carré [0,1] × [0,1]. C'est un exemple de courbe de longueur infinie qui couvre entièrement une surface, comme la courbe de **Hilbert** ou celle de **Lebesgue**.

Polygones de Sierpinski

Commençons par le triangle de **Sierpinski**.
Cette fractale se construit en itérant (à l'infini) le processus de coloration suivant :



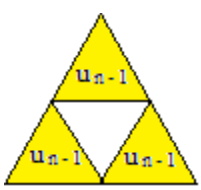
Les complexes z_1 , z_2 , z_3 sont les affixes des sommets du triangle représenté par T = [z_1 ; z_2 ; z_3]
La fonction récursive *Triangle* calcule la transformation d'ordre *n* :

```
function Triangle(T,n)
    if n > 0 then
        Triangle([T(1);(T(1)+T(2))/2;(T(1)+T(3))/2],n-1)
        Triangle([(T(1)+T(2))/2;T(2);(T(2)+T(3))/2],n-1)
        Triangle([(T(1)+T(3))/2;(T(2)+T(3))/2;T(3)],n-1)
    else
        xfpolys(real(T),imag(T),1)
    end
endfunction
```

Le programme ci-dessous permet d'obtenir le triangle d'ordre 5 (par exemple) à partir des 3 affixes 0, $e^{i\pi/3}$, 1 :

```
n = 5 ;
T = [0 ; exp(%i*%pi/3) ; 1] ;
f = scf(1) ;
xtitle('Triangle de Sierpinski')
plot2d(%inf, %inf, frameflag = 3, rect = [0,0,+1,+1], axesflag = 0)
drawlater()
Triangle(T,n)
drawnow()
```

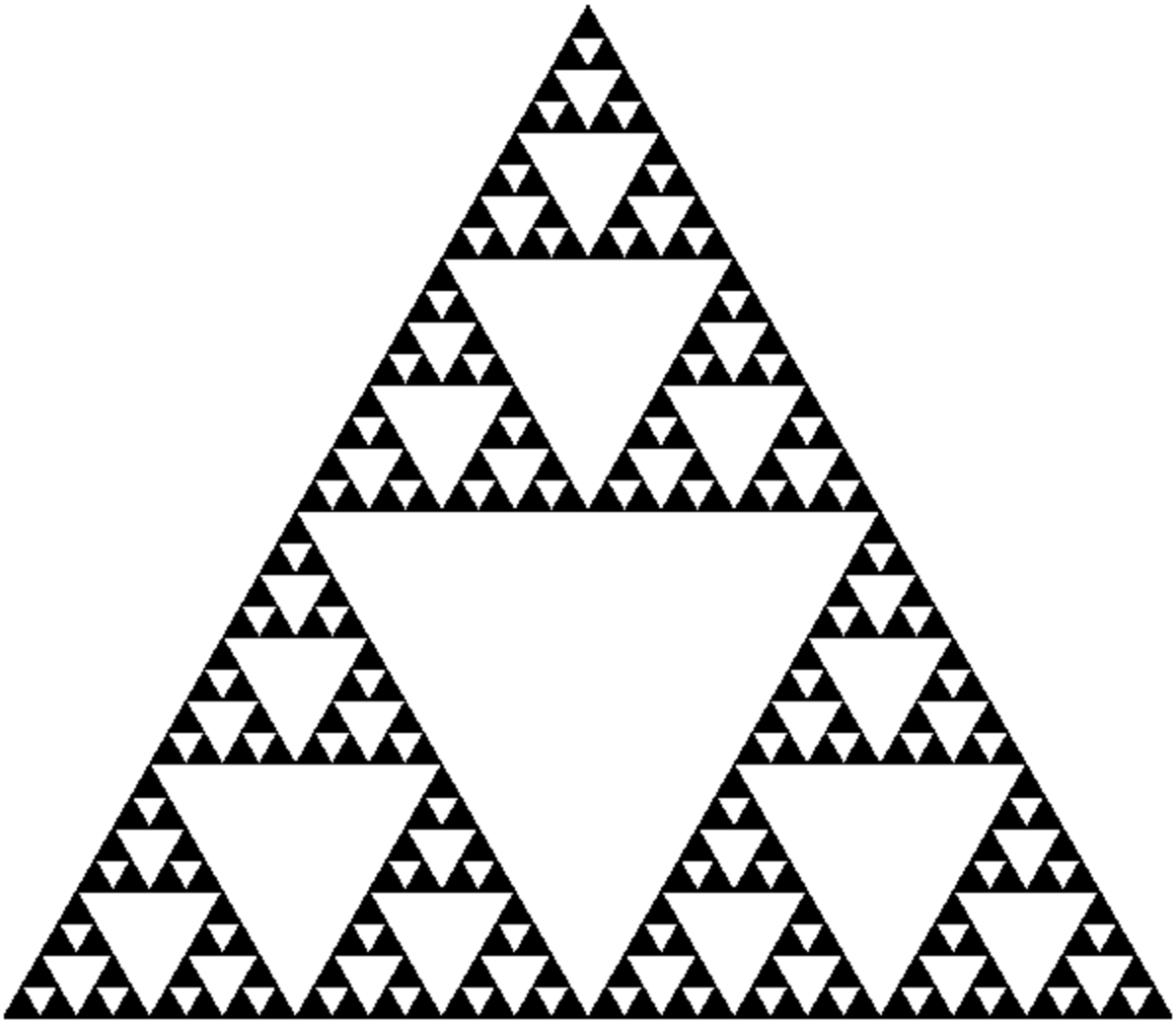
Remarque



Notons u_n le nombre total de triangles (de différentes dimensions) qui figurent dans le triangle **Sierpinski** d'ordre *n*. Avec 3 triangles d'ordre $n-1$ on construit un triangle d'ordre *n*. On obtient $3 \times u_{n-1}$ triangles formant un grand triangle plus un triangle central vide. D'où : $u_n = 3u_{n-1} + 2$. La suite u_n est une progression arithmético-géométrique de raisons 3 et 2 ; son terme général s'exprime en fonction de *n* : $u_n = 2.3^{n-1} - 1$.
Ainsi : $u_1 = 1$, $u_2 = 5$, $u_3 = 17$, $u_4 = 53$, $u_5 = 161$, …

Mathématique & Informatique

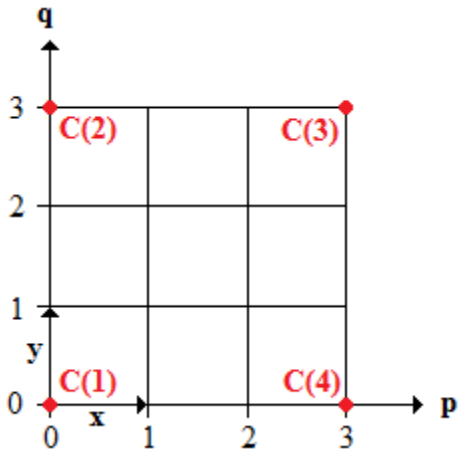
Triangle de **Sierpinski**



Passons au carré de **Sierpinski** (également appelé tapis de **Sierpinski**).
Un carré entièrement coloré est divisé en 9 petits carrés identiques. Le petit carré central devient incolore.



Notons C le vecteur colonne des affixes des sommets du carré numéroté dans le sens horaire en partant du coin inférieur gauche C(1). Chaque petit carré de ce carré initial est repéré à l'aide de 2 indices p et q, plus 2 déplacements élémentaires x et y. Le coin inférieur gauche d'un petit carré a pour affixe le nombre complexe $z_1 = C(1) + px + qy$ avec la condition : $(p,q) \in \{0,1,2\}\times\{0,1,2\}$. En tournant dans le sens horaire, les affixes des autres sommets sont respectivement $z_2 = z_1 + y$, $z_3 = z_2 + x$, $z_4 = z_3 - y$. Le petit carré central, celui correspondant à p = q = 1, n'est pas examiné.



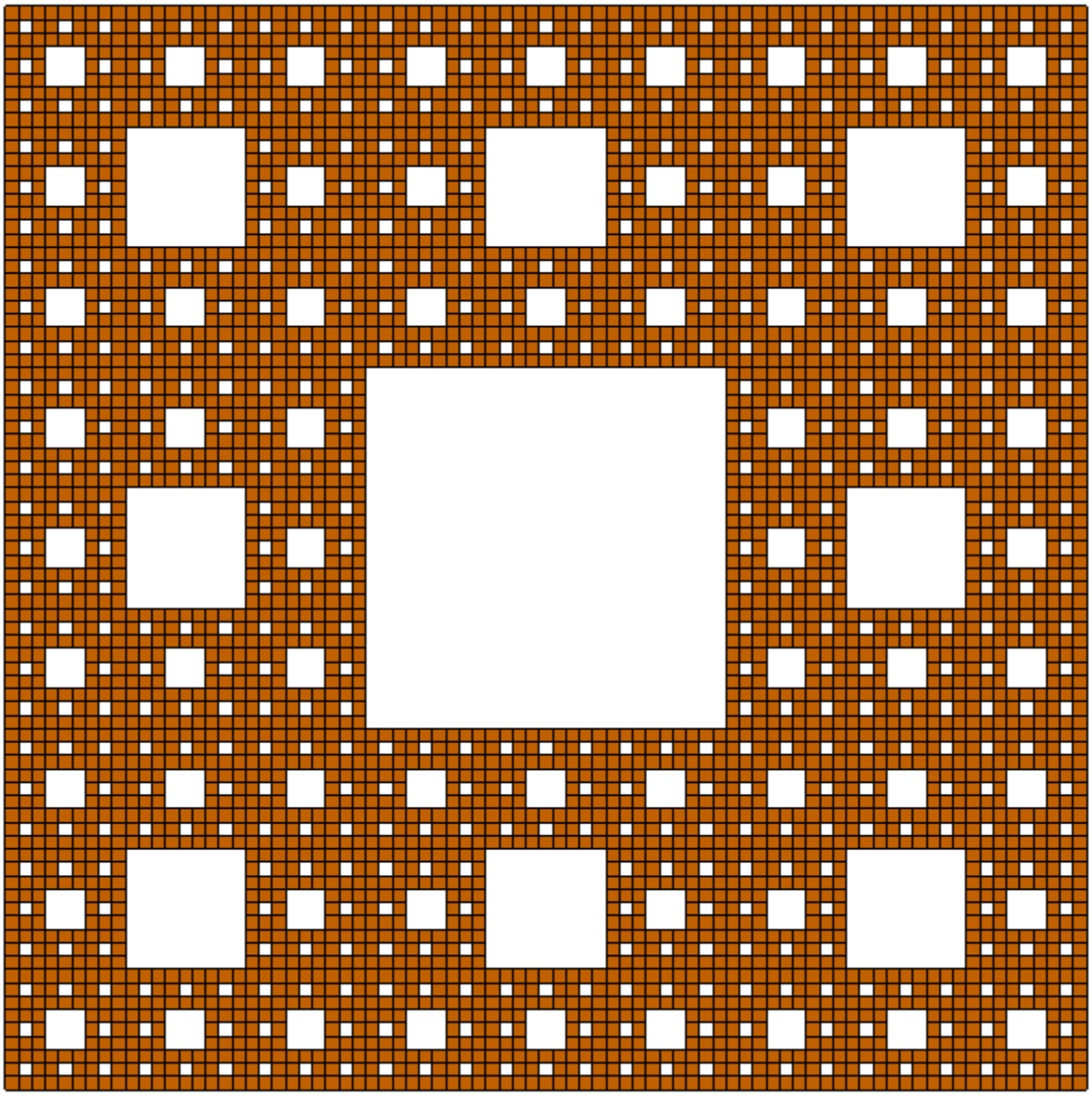
Ecrivons une fonction *Carre* qui itère n fois le processus que l'on vient de décrire :

```
function Carre(C,n)
    if n > 0 then
        x = (C(4)-C(1))/3
        y = (C(2)-C(1))/3
        z = C(1)
        for p = 0:2
            for q = 0:2
                if p == 1 & q == 1 then, continue, end
                z1 = z + p*x + q*y
                z2 = z1 + y
                z3 = z2 + x
                z4 = z3 - y
                Carre([z1;z2;z3;z4],n-1)
            end
        end
    else
        xfpolys(real(C),imag(C),27)
    end
endfunction
```

En appelant cette fonction avec le carré [0,1]×[0,1], c'est-à-dire avec C = [0 ; i ; 1+i ; 1], et n = 4 (par exemple), on dessine le carré de **Sierpinski** d'ordre 4 :

```
n = 4 ;
C = [0 ; %i ; 1+%i ; 1] ;
f = scf(2) ;
xtitle('Carré de Sierpinski')
plot2d(%inf, %inf, frameflag = 3, rect = [0,0,+1,+1], axesflag = 0)
drawlater()
Carre(C,n)
drawnow()
```

Tapis de **Sierpinski**



Mathématique & Informatique

L'éponge de **Menger** est à l'espace à 3 dimensions ce que le tapis (carré) de **Sierpinski** est au plan.
Pour représenter cette fractale à l'ordre n, le programme graphique fait appel à 2 fonctions : une fonction récursive *Menger* qui divise le cube de départ en N cubes plus petits, et une fonction *Cube* qui dessine les faces d'un cube dont le centre est le point C de coordonnées [x,y,z] = [C(1),C(2),C(3)] et dont les arêtes ont pour longueur a.

```
function Menger(C,a,p)
  if p > 0 then
    for i = -1:+1
      for j = -1:+1
        for k = -1:+1
          x = C(1) + i*a/3
          y = C(2) + j*a/3
          z = C(3) + k*a/3
          if i == 0 & j == 0 | ...
              i == 0 & k == 0 | ...
              j == 0 & k == 0 then
              Cube([x,y,z],a/3,8)
          else
             Menger([x,y,z],a/3,p-1)
          end
        end
      end
    end
  else
    Cube(C,a,27)
  end
endfunction
```

```
function Cube(C,a,q)
  nf = 6 ; na = 12 ; ns = 8 ; nsf = 4
  S = (a/2)*[1,1,1;-1,1,1;-1,-1,1;1,-1,1;1,1,-1;-1,1,-1;-1,-1,-1;1,-1,-1]
  F = [1,4,3,2;1,2,6,5;2,3,7,6;3,4,8,7;4,1,5,8;5,6,7,8]
  xf = zeros(nsf,nf) ; yf = zeros(nsf,nf) ; zf = zeros(nsf,nf) ; sf = zeros(nsf,3)
  for i = 1:nf
    for j = 1:nsf
        sf(j,:) = S(F(i,j),:)
    end
    xf(:,i) = sf(:,1) ; yf(:,i) = sf(:,2) ; zf(:,i) = sf(:,3)
  end
  plot3d(C(1)+xf, C(2)+yf, C(3)+zf, theta = 50, alpha = 65, flag = [q,4,0])
endfunction
```

Le paramètre p (fonction *Menger*) est l'ordre du cube, le paramètre q (fonction *Cube*) est la couleur du cube.
Ouvrons une fenêtre graphique subdivisée en 4 parties. Appelons la fonction *Menger* pour le cube unité centré à l'origine, C = [0,0,0] et a = 1, avec les valeurs successives n = 0, 1, 2, 3. On obtient les fractales d'ordre 0 à 3 :

```
drawlater()
for n = 0:3
    subplot(2,2,n+1)
    xtitle('n = ' + string(n))
    Menger([0,0,0],1,n)
end
drawnow()
```

Remarque

Les complexes sont utilisés en 2D. En 3D, les calculs sont effectués directement avec les coordonnées cartésiennes.
Le nombre N d'appels à la fonction *Cube* est une progression arithmético-géométrique de raisons 20 et 7. Donc :

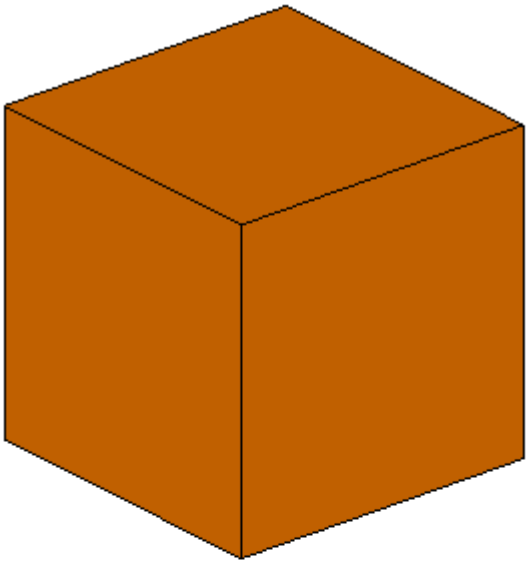
$$N = 20^n + 7\,\frac{(20^n - 1)}{(20 - 1)} = \frac{26 \times 20^n - 7}{19}$$

Pour n = 0 : N = 1 ; pour n = 1 : N = 27 ; pour n = 2 : N = 547 ; pour n = 3 : N = 10947 .
L'éponge de **Menger** d'ordre n est constituée de 20^n petits cubes élémentaires d'arêtes $a/3^n$.
Sur les graphiques, ces petits cubes sont en marron (couleur n° 27) et l'espace "intercube" est en blanc (couleur n° 8).

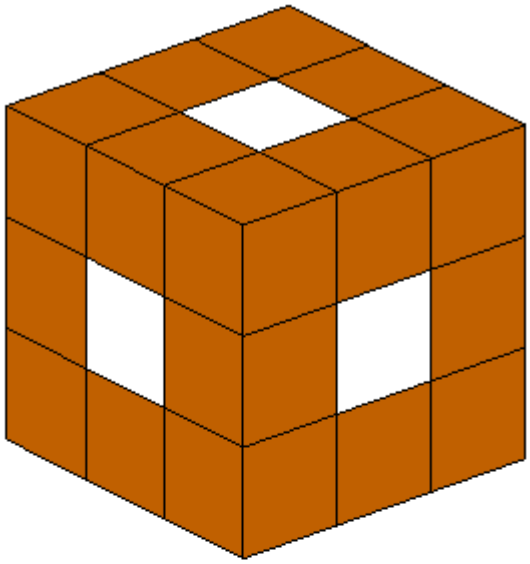
Mathématique & Informatique

Eponge de **Menger**

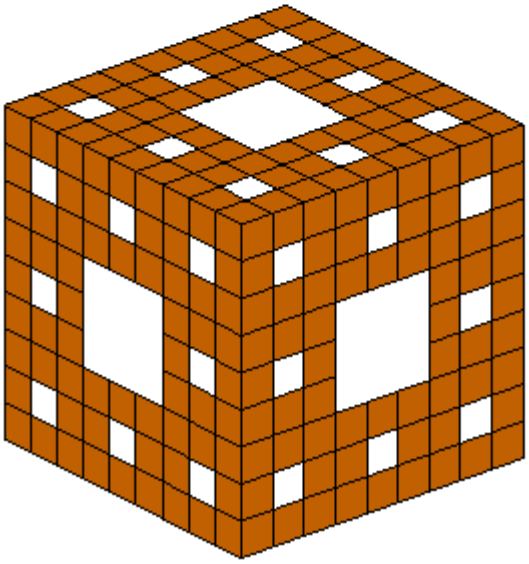
n = 0



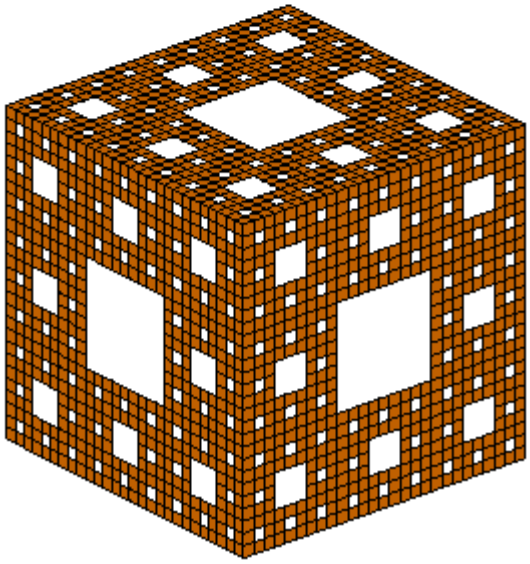
n = 1



n = 2



n = 3



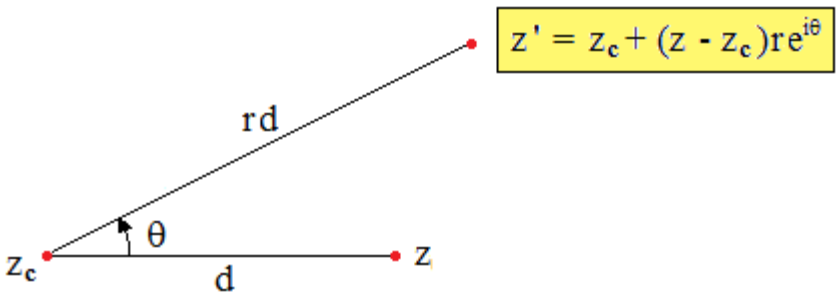
Mathématique & Informatique

Le procédé de **Sierpinski** se généralise à un polygone régulier à p côtés (p ≥ 5) que l'on divise en p polygones plus petits à p côtés. Les petits polygones périphériques seront coloriés alors que l'espace complémentaire restera vide.
On montre que le rapport d'homothétie d'un des petits polygones isométriques avec le grand polygone est égal à :

$$r = \frac{1}{1 + \dfrac{\sin\left[\left(2\,E\left(\dfrac{p}{4}\right)+1\right)\dfrac{\pi}{p}\right]}{\sin\left(\dfrac{\pi}{p}\right)}}$$

NB La fonction E(x) désigne la partie entière de x

Rappelons un résultat important sur les similitudes directes de rapport r et d'angle θ dans $\mathbb{C}$:



Le nombre complexe z_c est l'affixe du centre C de la similitude composée d'une homothétie de rapport r et d'une rotation d'angle θ. Le point M' d'affixe z' est l'image du point M d'affixe z par la similitude de paramètres C, r, θ.
La distance du point M au point C est d = |z - z_c| ; la distance du point M' au point C est d' = |z' - z_c| = rd.

Soient Z_1 , Z_2 , Z_3 , ... , Z_p les affixes des sommets du grand polygone numéroté dans le sens horaire, et soient $z_{k,1}$, $z_{k,2}$, $z_{k,3}$, ... , $z_{k,p}$ celles du k^e petit polygone périphérique (1 ≤ k ≤ p) numéroté également dans le sens horaire.
Le sommet $z_{k,1}$ est égal à Z_k, le sommet $z_{k,2}$ s'obtient par une homothétie de rapport r, et le sommet $z_{k,j}$ (3 ≤ j ≤ p) par une rotation d'angle ((p-2)/p)π centrée en $z_{k,j-1}$.
On a :

$$z_{k,1} = Z_k$$

$$z_{k,2} = Z_k + r(Z_{k+1} - Z_k)$$

$$z_{k,j} = z_{k,j-1} + (z_{k,j-2} - z_{k,j-1})\,e^{i\left(\frac{p-2}{p}\right)\pi} \qquad (3 \le j \le p)$$

NB Par convention : $Z_{p+1} = Z_1$

Des formules précédentes, on déduit la fonction récursive *Polygone* qui calcule la transformation d'ordre n :

```
function Polygone(Z,n)
    if n > 0 then
        Z = [Z;Z(1)]
        for k = 1:p
            z = zeros(p,1)
            z(1) = Z(k)
            z(2) = Z(k) + r*(Z(k+1)-Z(k))
            for j = 3:p
                z(j) = z(j-1) + (z(j-2)-z(j-1))*exp(%i*(p-2)/p*%pi)
            end
            Polygone(z,n-1)
        end
    else
        xfpolys(real(Z),imag(Z),2)
    end
endfunction
```

Partons du polygone régulier à p côtés contenu dans le cercle de centre 0 et de rayon 1.
Les affixes Z_k ($1 \le k \le p$) de ses sommets sont données par :

$$Z_k = e^{i\left(\frac{\pi}{2} - \frac{2(k-1)\pi}{p}\right)}$$

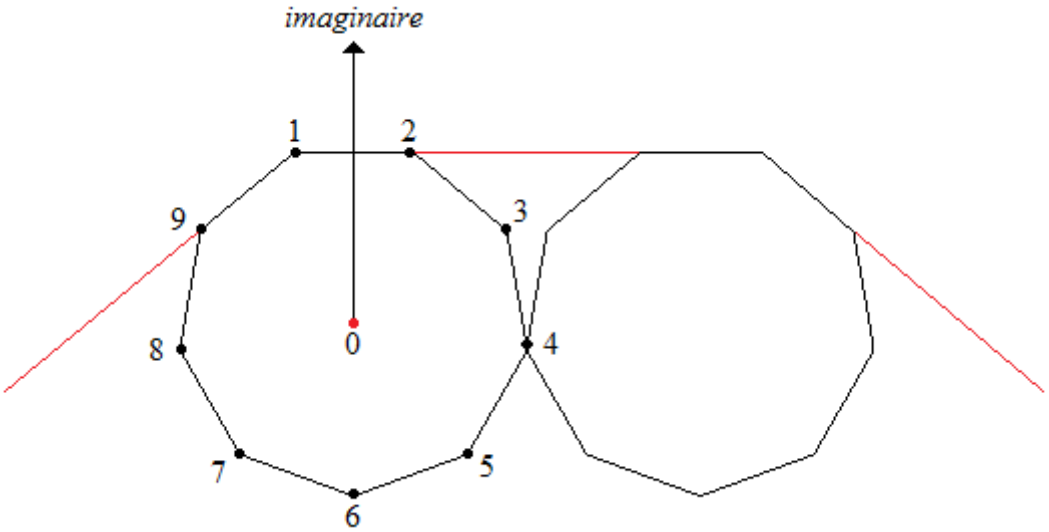
Par exemple, pour n = 4 et p = 6, on obtient l'hexagone de **Sierpinski** d'ordre 4 :

```
n = 4 ;
p = 6 ;
r = 1/(1 + sin((2*floor(p/4)+1)*%pi/p)/sin(%pi/p)) ;
Z = exp(%i*(%pi/2 - 2*([1:p]-1)*%pi/p)).' ;
f = scf(3) ;
xtitle('Polygone (' + string(p) + ' côtés) de Sierpinski')
plot2d(%inf, %inf, frameflag = 3, rect = [-1,-1,+1,+1], axesflag = 0)
drawlater()
Polygone(Z,n)
drawnow()
```

Mathématique & Informatique

Le calcul de r

Soit le petit polygone régulier à p côtés (p = 9 sur le schéma) inscrit dans le cercle unité.
Numérotons ses sommets s_k ($1 \le k \le p$) dans le sens horaire de la manière suivante :



Les sommets s_k ont pour affixes $e^{i\left(\frac{\pi}{2}+\frac{\pi}{p}-(k-1)\frac{2\pi}{p}\right)}$ où k est le numéro de sommet (k = 1, 2, 3, … , p).
Deux petits polygones successifs sont jointifs par le sommet d'abscisse maximale, c'est-à-dire par le sommet k tel que $\cos[\pi/2 - (2k-3)\pi/p]$ soit maximal, autrement dit pour la valeur k vérifiant $\pi/2 - (2k-3)\pi/p = 0$, donc $k = (p+6)/4$
Comme k doit être entier, on prend l'arrondi entier le plus proche de (p+6)/4
Avec la fonction *round(x)* de Scilab, on a : $k = round[(p+6)/4]$
En introduisant la fonction *floor(x)* qui est la partie entière inférieure de x, on a : $k = floor[(p+8)/4] = floor(p/4) + 2$
Du pentagone à l'heptagone : k = 3, de l'octogone à l'hendéagone : k = 4, du dodécagone au pentadécagone : k = 5, ...
Lorsque p est un multiple de 4, $p = 4\lambda$ (octogone, dodécagone, ...), les 2 petits polygones possèdent un côté commun puisque les sommets $\lambda+1$ et $\lambda+2$ ont le même cosinus.
Le petit polygone (en noir) a pour côté : $c = 2\sin(\pi/p)$
Le grand polygone (en rouge) a pour côté : $c/2 + 2\cos[\pi/2 - (2k-3)\pi/p] + c/2 = 2\sin(\pi/p) + 2\sin[(2k-3)\pi/p]$
Adoptons la notation mathématique E(x) pour désigner la partie entière de x. On a : $k = E(p/4) + 2$
Le rapport d'homothétie entre le petit polygone et le grand polygone est défini par :

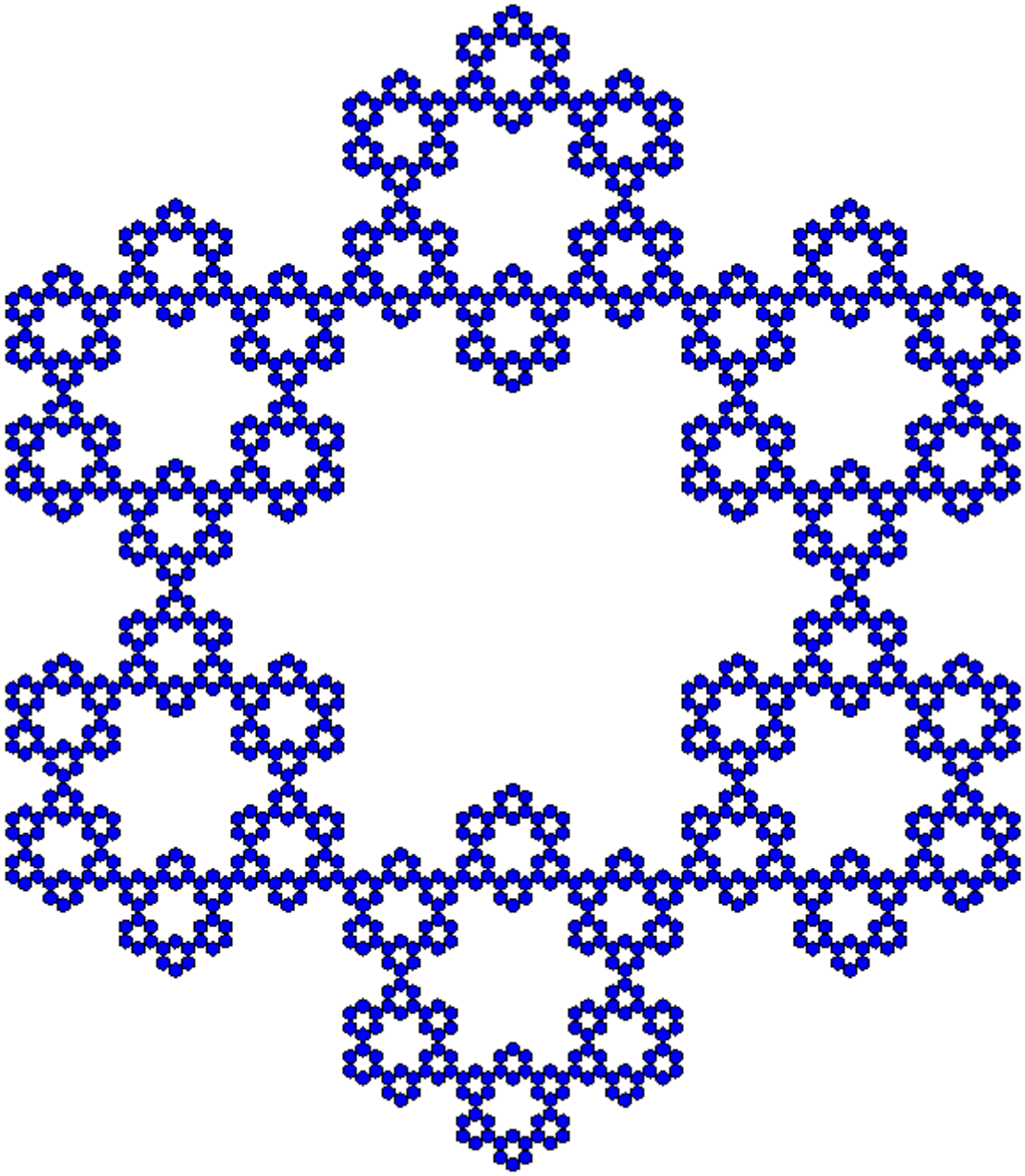
$$r = \frac{2\sin\left(\frac{\pi}{p}\right)}{2\sin\left(\frac{\pi}{p}\right) + 2\sin\left[(2k-3)\frac{\pi}{p}\right]} = \frac{1}{1 + \frac{\sin\left[(2k-3)\frac{\pi}{p}\right]}{\sin\left(\frac{\pi}{p}\right)}} = \frac{1}{1 + \frac{\sin\left[\left(2E\left(\frac{p}{4}\right)+1\right)\frac{\pi}{p}\right]}{\sin\left(\frac{\pi}{p}\right)}}$$

L'expression de r peut être simplifiée dans des cas particuliers. Ainsi, pour p = 6 (hexagone) on trouve r = 1/3, et pour p = 5 (pentagone) on met r sous la forme $r = 1/\varphi^2$ où $\varphi = (1+\sqrt{5})/2$ est le nombre d'or.

C.Q.F.D.

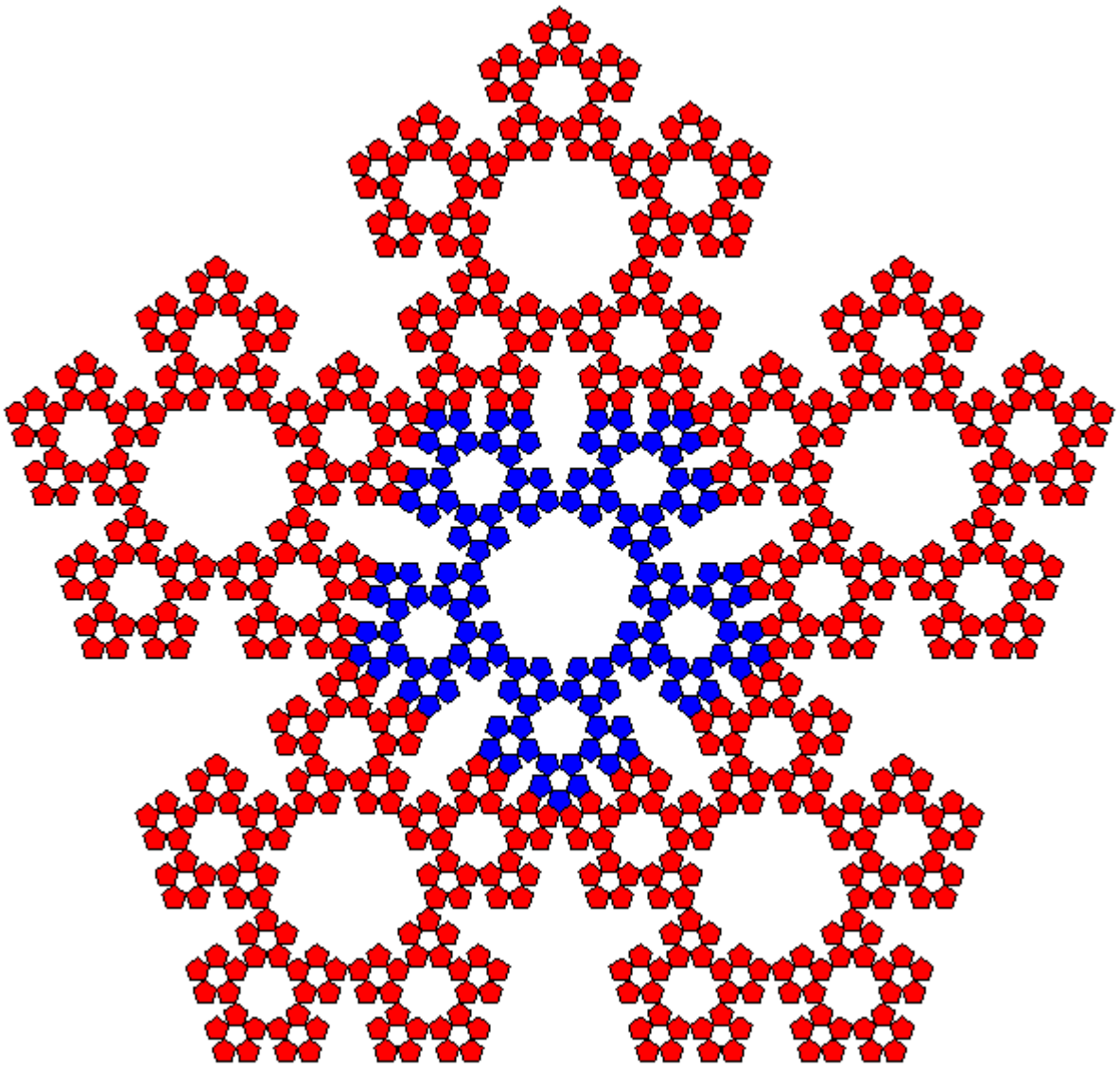
Mathématique & Informatique

Hexagone de **Sierpinski**



Mathématique & Informatique

Pentagone de **Sierpinski**



Courbes de Lévy et de Heighway

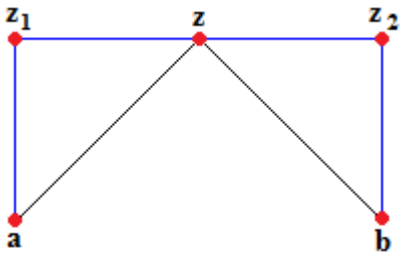
Ces fractales se construisent en itérant (à l'infini) le processus de triangulation suivant :

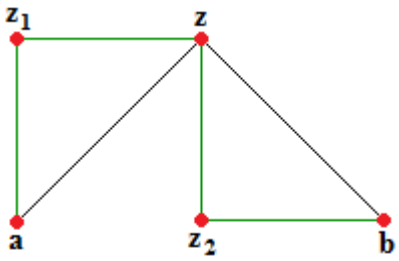


Notons S = [a ; b] le segment d'affixes a et b, et z l'affixe telle que T = [a ; z ; b] soit un triangle rectangle isocèle. Le complexe z s'obtient par une similitude de centre a, de rapport $1/\sqrt{2}$, et d'angle $\pi/4$. Donc :

$$z = a + \frac{(b-a)}{\sqrt{2}} e^{i\frac{\pi}{4}} = \frac{(1-i)}{2} a + \frac{(1+i)}{2} b = \frac{1}{2} \begin{pmatrix} 1-i & 1+i \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

Pour calculer la transformation d'ordre n, on a le choix de l'orientation du segment d'extrémités a et z et du segment d'extrémités b et z, d'où la possibilité de 2 fonctions récursives différentes nommées *Levy* et *Heighway* :






```
function Levy(S,n)
    if n > 0 then
        z = [1-%i,1+%i]/2*S
        Levy([S(1);z],n-1)
        Levy([z;S(2)],n-1)
    else
        xsegs(real(S),imag(S),2)
    end
endfunction
```

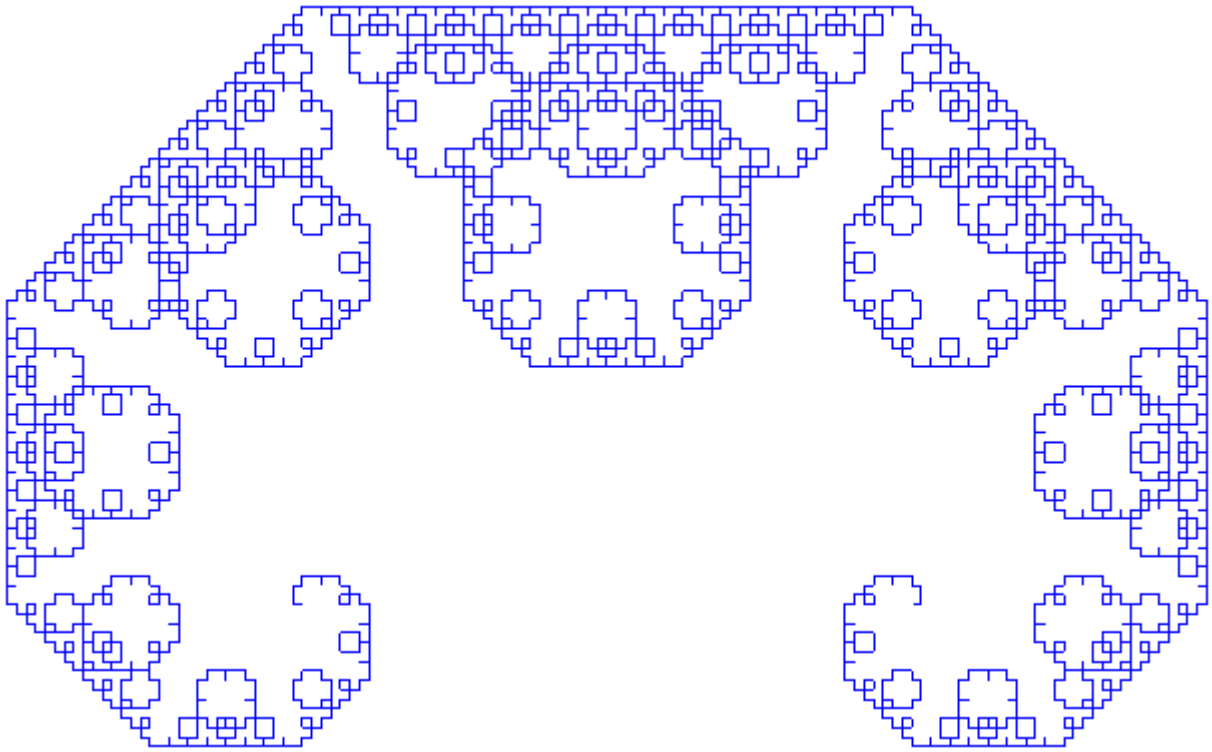
```
function Heighway(S,n)
    if n > 0 then
        z = [1-%i,1+%i]/2*S
        Heighway([S(1);z],n-1)
        Heighway([S(2);z],n-1)
    else
        xsegs(real(S),imag(S),13)
    end
endfunction
```

Appelons les fonctions *Levy* et *Heighway* avec n = 12 (par exemple) en partant du segment horizontal S = [0 ; 1] :

```
n = 12 ;
S = [0;1] ;
f = scf(1) ;
drawlater()
xtitle('Courbe de Levy')
plot2d(%inf, %inf, frameflag = 3, rect = [-0.5,-0.5,+1.5,+1], axesflag = 0)
Levy(S,n)
drawnow()
f = scf(2) ;
drawlater()
xtitle('Courbe de Heighway')
plot2d(%inf, %inf, frameflag = 3, rect = [-0.5,-0.5,+1.5,+1], axesflag = 0)
Heighway(S,n)
drawnow ()
```

Mathématique & Informatique

Courbe de **Lévy**



Courbe de **Heighway**



Flocon de Von Koch

Cette fractale se construit en itérant (à l'infini) le processus de segmentation suivant :



Les complexes a et b sont les affixes des extrémités du segment représenté par le vecteur S = [a ; b]
Le segment S est divisé en 3 parties égales et on trace au milieu un triangle équilatéral T = [z_1 ; z_2 ; z_3]
Les complexes z_1 , z_2 , z_3 se déduisent de a et b par les formules :

$$z_1 = a + \frac{(b-a)}{3} \quad ; \quad z_2 = z_1 + \frac{(b-a)}{3} e^{i\frac{\pi}{3}} \quad ; \quad z_3 = b - \frac{(b-a)}{3}$$

Sous forme matricielle :

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} 2/3 & 1/3 \\ (3-i\sqrt{3})/6 & (3+i\sqrt{3})/6 \\ 1/3 & 2/3 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

La fonction récursive *VonKoch* dessine la courbe d'ordre n composée de 4^n petits segments :

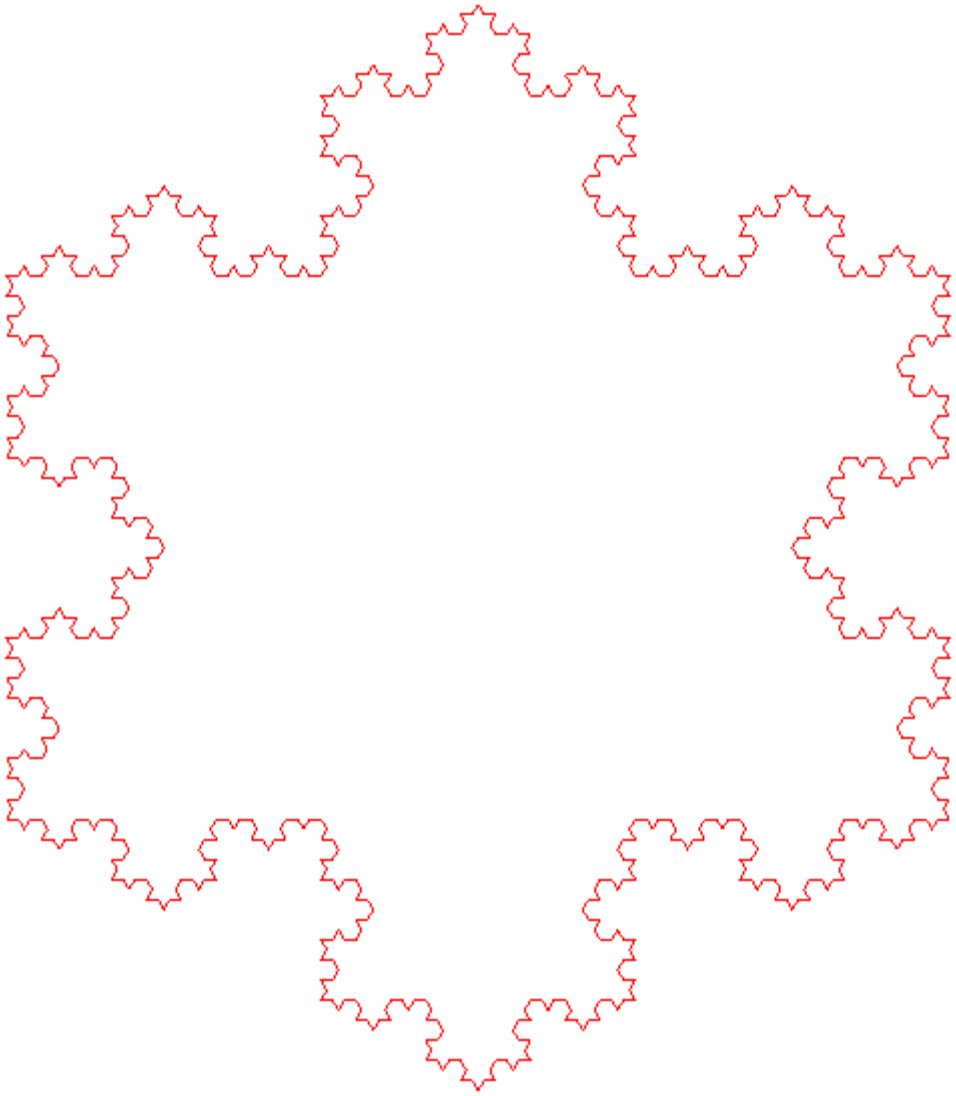
```
function VonKoch(S,n)
    if n > 0 then
        T = [2/3,1/3;(3-%i*sqrt(3))/6,(3+%i*sqrt(3))/6;1/3,2/3]*S
        VonKoch([S(1);T(1)],n-1)
        VonKoch([T(1);T(2)],n-1)
        VonKoch([T(2);T(3)],n-1)
        VonKoch([T(3);S(2)],n-1)
    else
        xsegs(real(S),imag(S),5)
    end
endfunction
```

Partons du triangle équilatéral dont les affixes des sommets sont les 3 nombres complexes $z_1 = 0$, $z_2 = e^{i\pi/3}$, $z_3 = 1$
On obtient le flocon d'ordre 4 (par exemple) en écrivant :

```
n = 4 ;
z1 = 0 ; z2 = exp(%i*%pi/3) ; z3 = 1 ;
drawlater()
xtitle('Flocon de Von Koch')
plot2d(%inf, %inf, frameflag = 3, rect = [0,-0.5,+1,+1], axesflag = 0)
VonKoch([z1;z2],n)
VonKoch([z2;z3],n)
VonKoch([z3;z1],n)
drawnow()
```

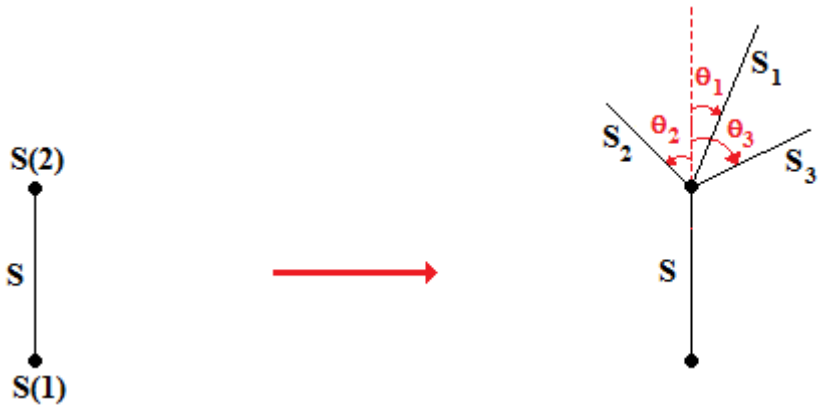
Mathématique & Informatique

Flocon de **Von Koch**



Fougère de Barnsley

Cette fractale est le résultat du processus de développement suivant :



On ajoute à un segment initial S trois segments supplémentaires S_1, S_2, S_3, puis on reproduit (à l'infini) ce procédé pour chaque nouveau segment créé. Le segment S est assimilé au vecteur des affixes de ses extrémités $S(1)$ et $S(2)$. Le segment S_p (p = 1, 2, 3) se calcule à partir du segment S par une similitude de centre $S(1)$, de rapport r_p et d'angle θ_p (compté positivement dans le sens trigonométrique), suivie d'une translation d'affixe $S(2) - S(1)$. Pour k = 1, 2 :

$$S_p(k) = S(1) + [S(k) - S(1)]r_p\, e^{i\theta_p} + [S(2) - S(1)] = S(2) + [S(k) - S(1)]r_p\, e^{i\theta_p}$$

Ecrivons une fonction récursive *Barnsley* qui dessine la fougère d'ordre n composée de $(3^{n+1} - 1)/2$ petits segments :

```
function Barnsley(S,n,c)
    xsegs(real(S),imag(S),13)
    if n > 0 then
        S1 = S(2) + (S-S(1))*r1*exp(%i*c*theta1)
        S2 = S(2) + (S-S(1))*r2*exp(%i*theta2)
        S3 = S(2) + (S-S(1))*r3*exp(%i*theta3)
        Barnsley(S1,n-1,c)
        Barnsley(S2,n-1,+1)
        Barnsley(S3,n-1,-1)
    end
endfunction
```

Le segment de départ est un trait vertical de longueur 1 représenté par le vecteur colonne $S = [0 ; i]$.
Les 3 similitudes directes sont définies en attribuant des valeurs aux coefficients r_p et θ_p. Prenons :

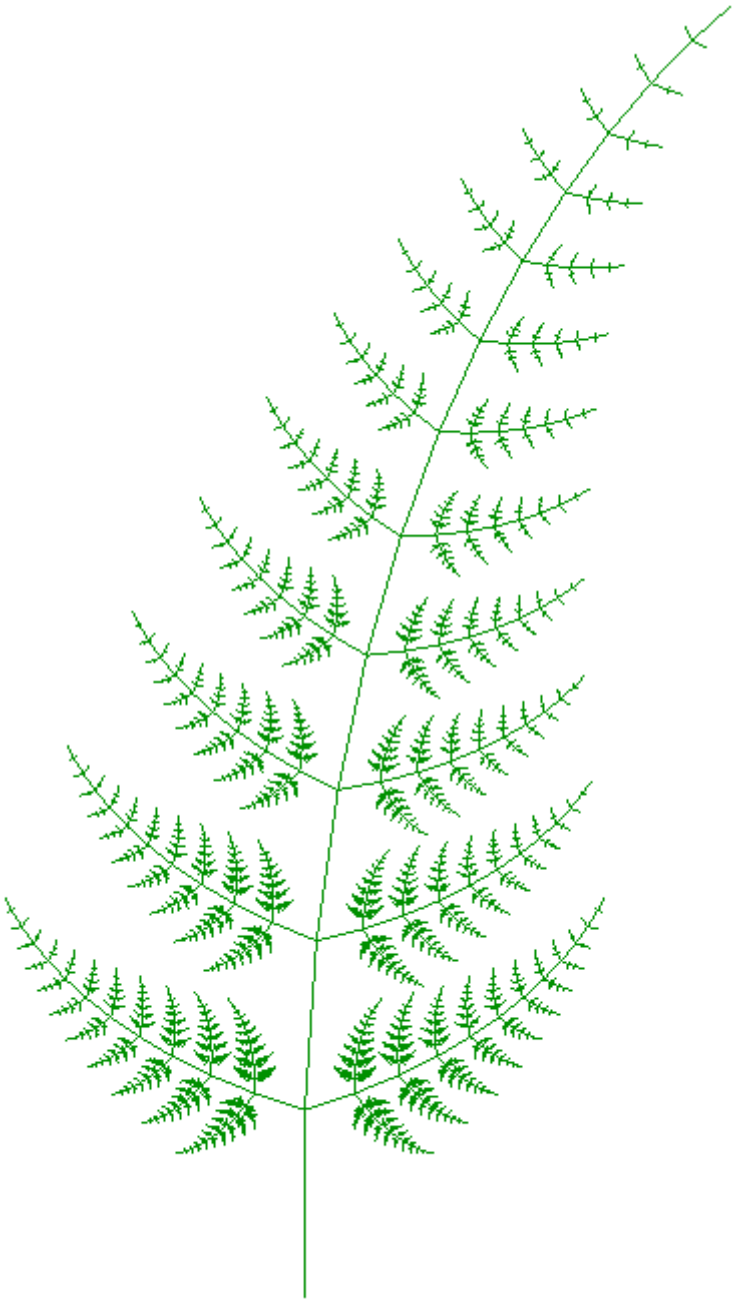
$r_1 = 0.90$, $\theta_1 = -0.07$; $r_2 = 0.28$, $\theta_2 = +1.25$; $r_3 = 0.28$, $\theta_3 = -1.25$

Le paramètre $c = \pm 1$ permet de choisir la courbure des branches de la fougère afin qu'elles soient toutes dirigées dans le même sens. Si le premier appel à la fonction *Barnsley* est effectué avec $c = +1$ alors le tronc de la fougère est incliné vers la droite ; si c'est avec $c = -1$ alors il est incliné vers la gauche.
Le programme de dessin de la fougère d'ordre 12 (par exemple) est :

```
n = 12 ;
r1 = 0.90 ; theta1 = - 0.07 ;
r2 = 0.28 ; theta2 = + 1.25 ;
r3 = 0.28 ; theta3 = - 1.25 ;
drawlater()
xtitle('Fougère de Barnsley')
plot2d(%inf, %inf, frameflag = 3, rect = [-2.0,0,+2.5,+7.0], axesflag = 0)
Barnsley([0;%i],n,1)
drawnow()
```

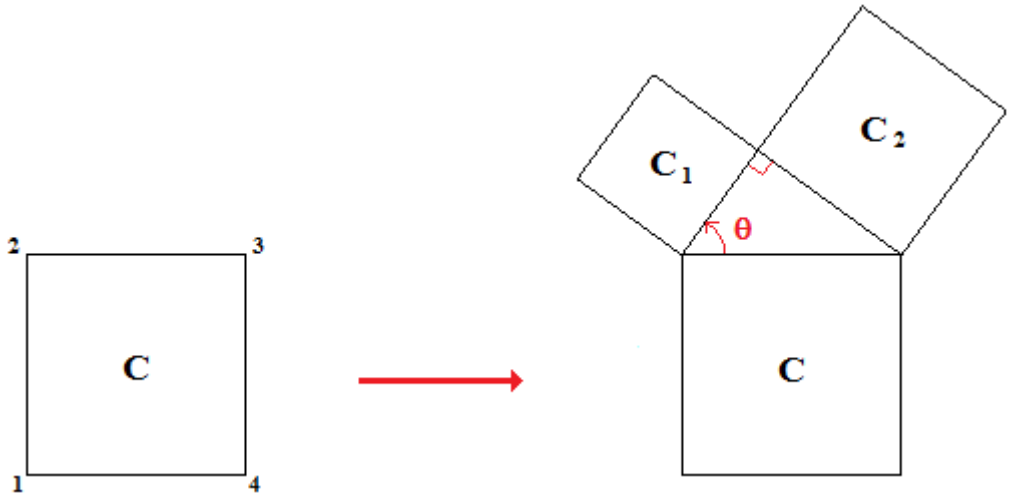
Mathématique & Informatique

Fougère de **Barnsley**



Arbre de Pythagore

Cette fractale s'obtient en itérant (à l'infini) le processus de construction suivant :



A partir d'un carré représenté par un vecteur colonne C constitué de 4 affixes C(1), C(2), C(3), C(4) numérotées dans le sens horaire du coin inférieur gauche au coin inférieur droit, on construit 2 autres carrés plus petits, un à gauche C_1 et l'autre à droite C_2, décalés d'un angle θ et π/2 - θ respectivement par rapport au côté supérieur du carré initial, de sorte que les 3 carrés C, C_1, C_2 forment au centre un triangle rectangle.
Le résultat est l'illustration géométrique du théorème de **Pythagore** : la somme des aires des 2 petits carrés est égale à l'aire du grand carré, ce qui s'écrit algébriquement $a^2 + b^2 = c^2$.
Le premier petit carré C_1 (celui de gauche) se calcule à partir de C par une similitude de centre C(1), de rapport cosθ et d'angle + θ (dans le sens trigonométrique), suivie d'une translation d'affixe C(2) - C(1). D'où, pour $1 \le k \le 4$:

$$C_1(k) = C(1) + [C(k) - C(1)]\cos(\theta)e^{+i\theta} + [C(2) - C(1)] = C(2) + [C(k) - C(1)]\cos(\theta)e^{i\theta}$$

Le deuxième petit carré C_2 (celui de droite) se calcule à partir de C par une similitude de centre C(4), de rapport sinθ et d'angle - (π/2 - θ) (dans le sens trigonométrique), suivie d'une translation d'affixe C(3) - C(4). D'où, pour $1 \le k \le 4$:

$$C_2(k) = C(4) + [C(k) - C(4)]\sin(\theta)e^{-i(\pi/2 - \theta)} + [C(3) - C(4)] = C(3) + [C(k) - C(4)]\sin(\theta)e^{-i(\pi/2 - \theta)}$$

Ecrivons une fonction récursive *Pythagore* qui dessine l'arbre d'ordre n composé de $2^{n+1} - 1$ petits carrés :

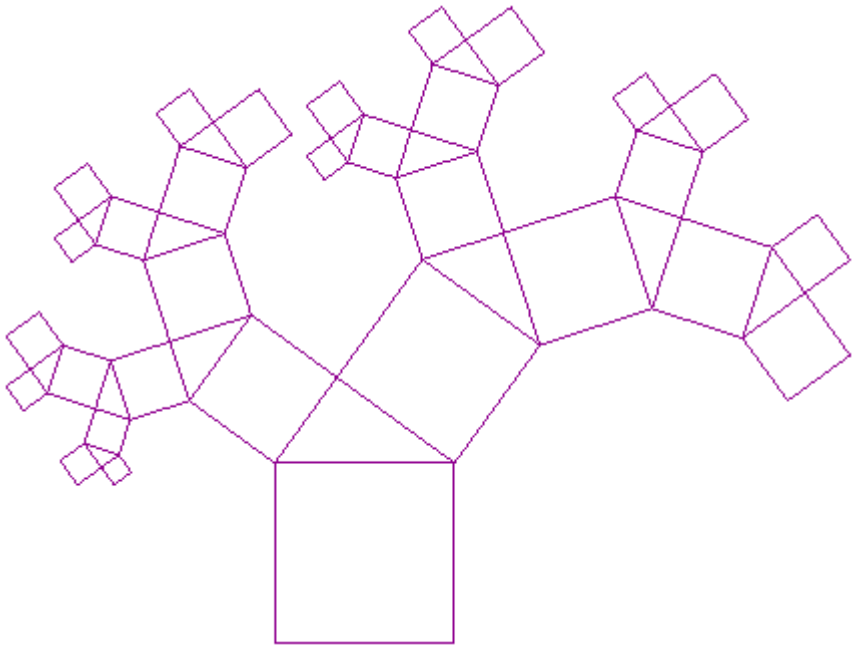
```
function Pythagore(C,n)
    xpoly(real(C),imag(C),'lines',1)
    if n > 0 then
        C1 = C(2) + (C-C(1))*cos(theta)*exp(%i*theta)
        C2 = C(3) + (C-C(4))*sin(theta)*exp(-%i*(%pi/2 - theta))
        Pythagore(C1,n-1)
        Pythagore(C2,n-1)
    end
endfunction
```

Le carré initial est le carré unité équivalent au vecteur [0;i;1+i;1] de $\mathbb{C}^4$. On choisit un angle θ (< π/2) et un niveau de récursivité n (≥ 0), puis on appelle la fonction *Pythagore* pour dessiner l'arbre d'ordre n. Exemple : θ = 3π/10 et n = 4

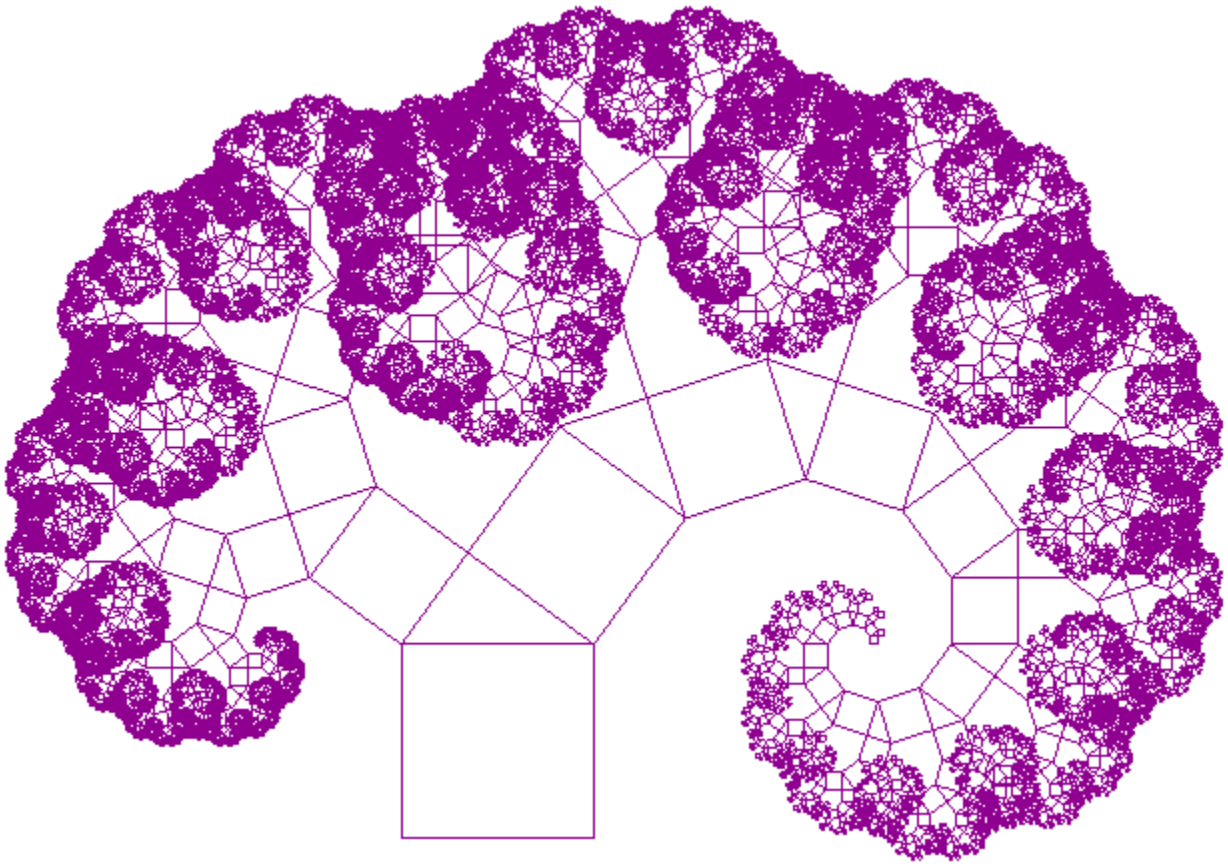
```
n = 4 ;
theta = 3*%pi/10 ;
drawlater()
xset("color",22)
xtitle('Arbre de Pythagore')
plot2d(%inf, %inf, frameflag = 3, rect = [-3,-1,+5,+5], axesflag = 0)
Pythagore([0;%i;1+%i;1],n)
drawnow()
```

Mathématique & Informatique

Arbre de **Pythagore** $\theta = 3\pi/10$ et n = 4

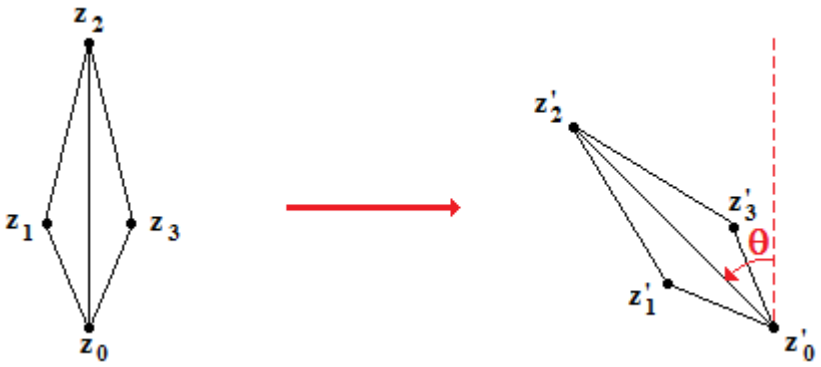


Arbre de **Pythagore** $\theta = 3\pi/10$ et n = 15



ÉTOILE

Cette fractale se construit en itérant (à l'infini) le processus de rotation suivant :



Soit b ($b \geq 2$) le nombre de branches de premier niveau de l'étoile et n ($n \geq 1$) le nombre de niveaux de rotation. Soient z_0 , z_1 , z_2 , z_3 les affixes des sommets du quadrilatère qui est la réunion des 2 triangles isométriques notés $T_1 = [z_0 ; z_2 ; z_3]$ et $T_2 = [z_0 ; z_1 ; z_2]$. Représentons ce quadrilatère par une matrice complexe de dimension (3,2) définie par $B = [T_1 , T_2] = [z_0 , z_0 ; z_2 , z_1 ; z_3 , z_2]$. Les 4 nombres complexes z_0 , z_1 , z_2 , z_3 ont pour expressions :

$$z_0 = 0 , \quad z_1 = r e^{i\left(\frac{\pi}{2} + \frac{\pi}{b}\right)} , \quad z_2 = R i , \quad z_3 = r e^{i\left(\frac{\pi}{2} - \frac{\pi}{b}\right)}$$

L'étoile est inscrite dans le cercle de centre 0 et de rayon R.

On prend $r \leq R$. Dans le cas particulier $r = \dfrac{R}{2\cos\left(\frac{\pi}{b}\right)}$ avec $b > 2$, la branche B est un losange.

Les affixes z'_k se calculent en multipliant les affixes z_k par le nombre complexe $e^{i\theta}$, où θ est l'angle de rotation.

Au niveau p, on a :

$$\theta = \frac{\pi}{2^{p-2} b} \qquad p = 1, \ldots, n$$

A la q^e rotation de niveau p, on a :

<u>$p = 1$</u>

$$z'_k = z_k e^{i \frac{2(q-1)\pi}{b}} \qquad k = 0,1,2,3 \ ; \ \ q=1,2,\ldots,b$$

<u>$p > 1$</u>

$$z'_k = z_k e^{i \frac{(2q-1)\pi}{2^{p-2} b}} \qquad k = 0,1,2,3 \ ; \ p = 2, \ldots, n \ ; \ q = 1, 2, \ldots, 2^{p-2} b$$

Mathématique & Informatique

Programmation

L'étoile d'ordre n est la réunion des étoiles obtenues aux différents niveaux p de rotation ($1 \le p \le n$).
Pour réaliser la bonne figure, il faut remplir le pixmap associé à l'image sur l'écran dans l'ordre des p décroissants, c'est-à-dire avec une boucle de la variable p du type $p = n{:}{-}1{:}1$ de pas -1.

Les 2 formules donnant z'_k en fonction de z_k sont réunies en une seule boucle de la variable q du type $q = 1{:}min(p,2){:}2^{p-1}b$ de pas 1 ou 2 :

- le pas de boucle min(p,2) dépend de p

(pas = 1 si $p = 1$ et pas = 2 si $p \ge 2$)

- l'indice maximal q_{max} n'atteint pas la limite supérieure $2^{p-1}b$ lorsque $p \ge 2$

($q_{max} = b = 2^{p-1}b$ si $p = 1$ mais $q_{max} = 2^{p-1}b - 1 < 2^{p-1}b$ si $p \ge 2$)

- l'indice minimal q_{min} peut être pris égal à 1 lorsque $p = 1$ puisque $e^{i2\pi} = e^{i0}$

($q = 1{:}1{:}b \quad \Leftrightarrow \quad q = 0{:}1{:}b{-}1$)

On ajoute une matrice C de dimension (n,2) dont les lignes $C(p,:) = [C(p,1),C(p,2)]$ sont les numéros des couleurs utilisées à chaque niveau de rotation pour remplir les 2 triangles T_1 et T_2.

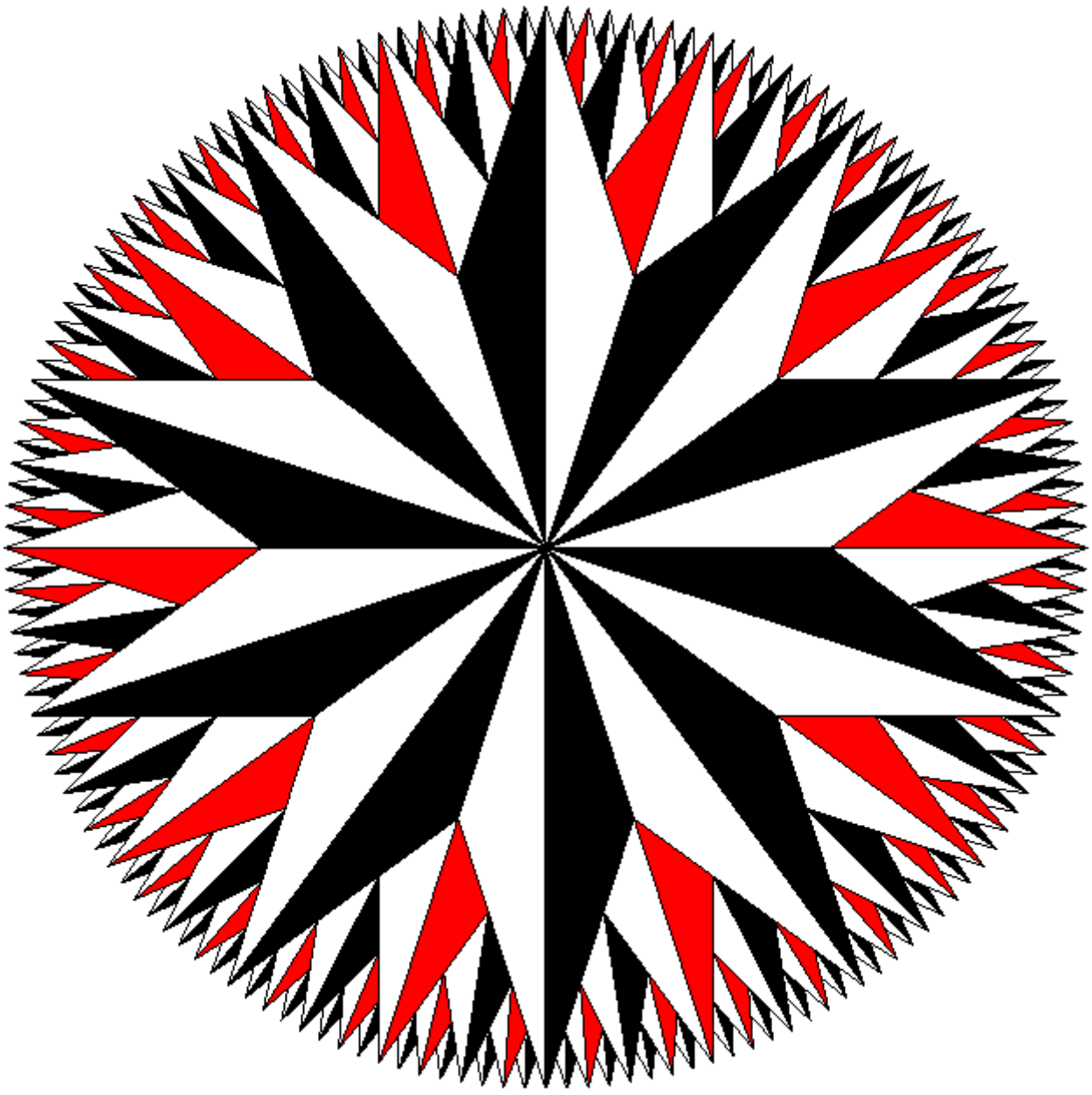
Finalement, on aboutit à une fonction *Star* très concise :

```
function Star(B,C,b,n)
  for p = n:-1:1
    for q = 1:min(p,2):2^(p-1)*b
      Z = B*exp(%i*q*%pi/2^(p-2)/b)
      xfpolys(real(Z), imag(Z), C(p,:))
    end
  end
endfunction
```

Par exemple, le programme ci-dessous affiche une étoile d'ordre 5 à 10 branches avec les couleurs blanc, noir, rouge :

```
n = 5 ; b = 10 ;
R = 1 ; r = R/(2*cos(%pi/b)) ;
z0 = 0 ; z1 = r*exp(%i*(%pi/2 + %pi/b)) ; z2 = R*%i ; z3 = r*exp(%i*(%pi/2 - %pi/b)) ;
B = [z0,z0;z2,z1;z3,z2] ;
C = ones(ceil(n/2),1).*.[8,1;8,5] ;
f1 = scf(1) ;
drawlater()
plot2d(%inf, %inf, frameflag = 3, rect = [-R,-R,R,R], axesflag = 0)
Star(B,C,b,n) ;
drawnow()
```

Mathématique & Informatique



Mathématique & Informatique

Les étoiles n = 3, b ∈ {2, 3, 4, 5, 6, 7, 8, 9, 10}, R = 1, r = 1/3

b = 2



b = 3



b = 4



b = 5



b = 6



b = 7



b = 8



b = 9

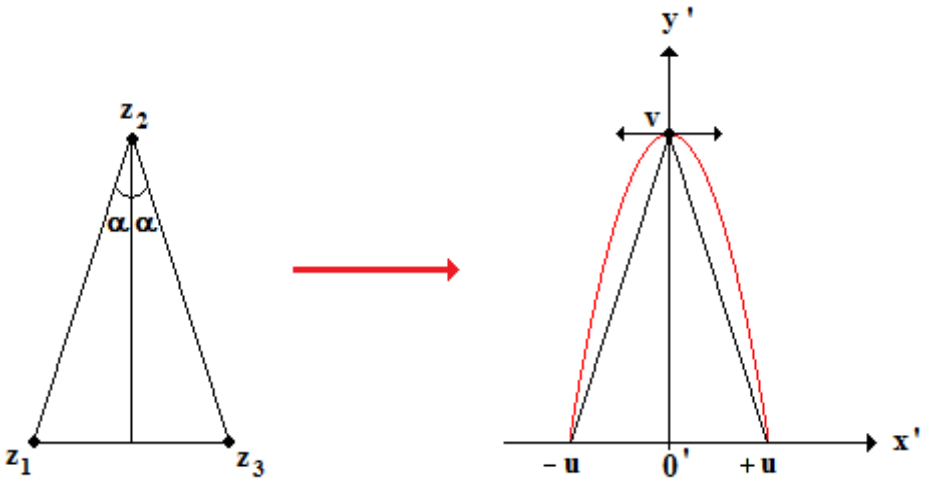


b = 10



Etoile parabolique

Cette étoile se déduit de l'étoile ordinaire en arrondissant les pointes de ses branches. Chaque extrémité triangulaire est remplacée par un arc de parabole conformément au schéma ci-dessous :



Soit $y' = ax'^2 + bx' + c$ l'équation de la parabole dans le repère (0',x',y'). La courbe passe par les points de coordonnées (0,v) et (u,0) avec une tangente horizontale au point d'abscisse nulle. Les coefficients a, b, c doivent vérifier :

$a(0)^2 + b(0)^2 + c = v$
$a(u)^2 + b(u) + c = 0$
$2a(0) + b = 0$

On trouve : $a = -v/u^2$, $b = 0$, $c = v$; donc : $y' = -(v/u^2)x'^2 + v$

Pour un triangle isocèle quelconque (z_1,z_2,z_3) du plan, on utilise l'équation de la parabole que l'on vient de définir en considérant un repère relatif (0',x',y') centré au milieu de la base du triangle et un axe des abscisses colinéaire à cette base. L'affixe z d'un point de la courbe (tracée en rouge sur le graphique) par rapport au repère absolu (0,x,y) s'obtient par une translation d'affixe $z_{0'} = (z_1 + z_3)/2$ suivie d'une rotation d'angle $\theta = \mathrm{Arg}(z_3 - z_1)$.
Soient x' et y' les coordonnées cartésiennes d'un point de la parabole dans le repère relatif. L'affixe z de ce point est :

$$z = z_{0'} + (x' + iy')e^{i\theta}$$

Dans le cas présent, on a :

$$z_1 = re^{i\left(\frac{\pi}{2}+\frac{\pi}{b}\right)} \quad , \quad z_2 = Ri \quad , \quad z_3 = re^{i\left(\frac{\pi}{2}-\frac{\pi}{b}\right)}$$

D'où :

$$u = r\sin\frac{\pi}{b} \quad , \quad v = R - r\cos\frac{\pi}{b} \quad , \quad z_{0'} = ir\cos\frac{\pi}{b} \quad , \quad \theta = 0$$

Le programme Scilab est donné page suivante.

```
//-------------------------------------------------
//                  Etoile parabolique
//-------------------------------------------------

funcprot(0) ; lines(0)

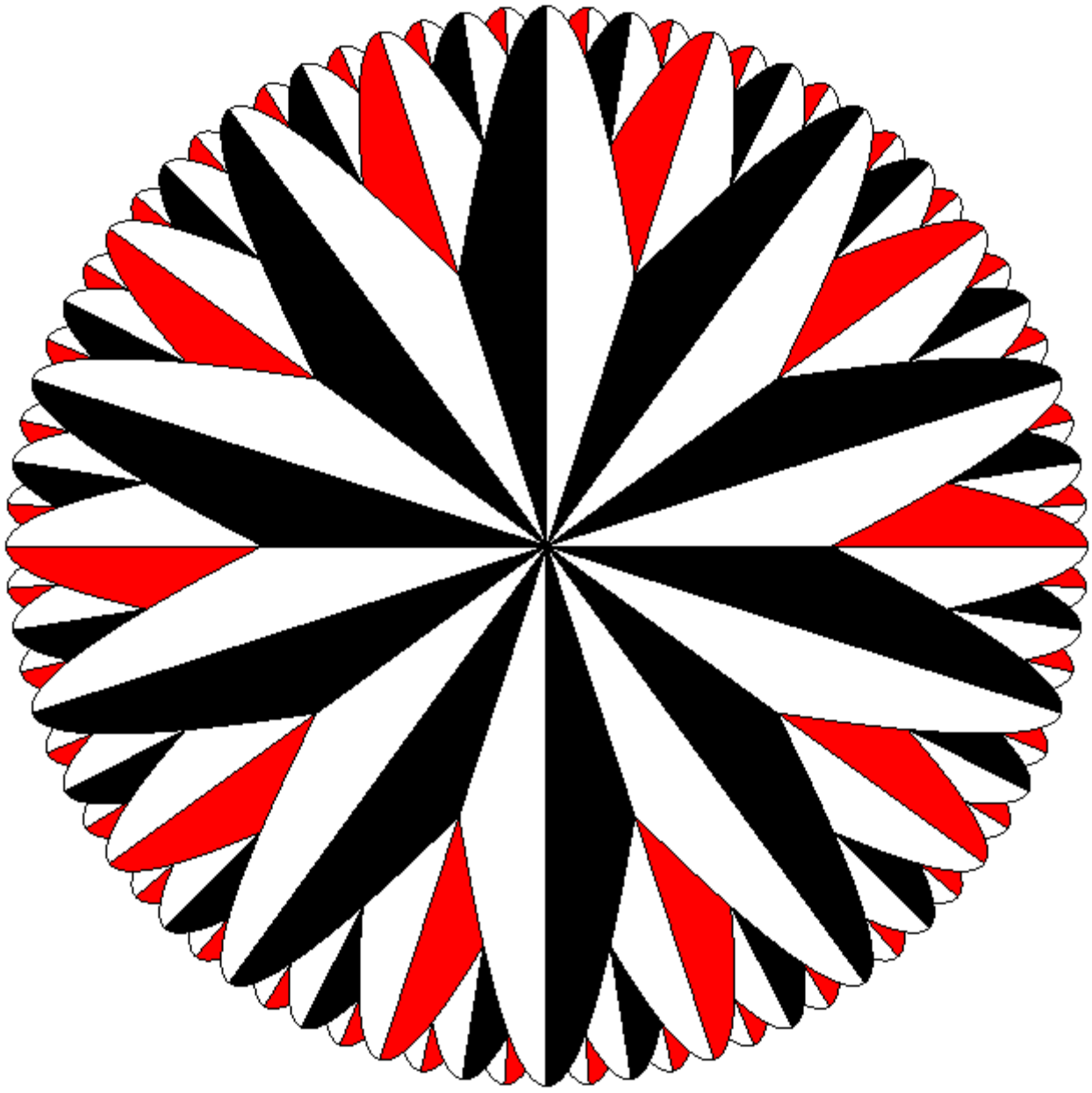
// Branche parabolique
function B = parabole(R,r,b)
    u = r*sin(%pi/b)
    v = R - r*cos(%pi/b)
    X = linspace(0,u,100)'
    Y = -(v/u^2)*X^2 + v
    B1 = %i*r*cos(%pi/b) + (X + %i*Y)
    B2 = %i*r*cos(%pi/b) + (-X + %i*Y)
    B = [0,0;B1,B2]
endfunction

// Dessin de l'étoile
function Star(B,C,b,n)
  for p = n:-1:1
    for q = 1:min(p,2):2^(p-1)*b
      Z = B*exp(%i*q*%pi/2^(p-2)/b)
      xfpolys(real(Z), imag(Z), C(p,:))
    end
  end
endfunction

// Programme principal
n = 4 ; b = 10 ;
R = 1 ; r = R/(2*cos(%pi/b)) ;
B = parabole(R,r,b) ;
C = ones(ceil(n/2),1).*.[8,1;8,5] ;
f2 = scf(2) ;
drawlater()
plot2d(%inf,%inf,frameflag = 3,rect = [-R,-R,R,R],axesflag = 0)
Star(B,C,b,n) ;
drawnow()
```

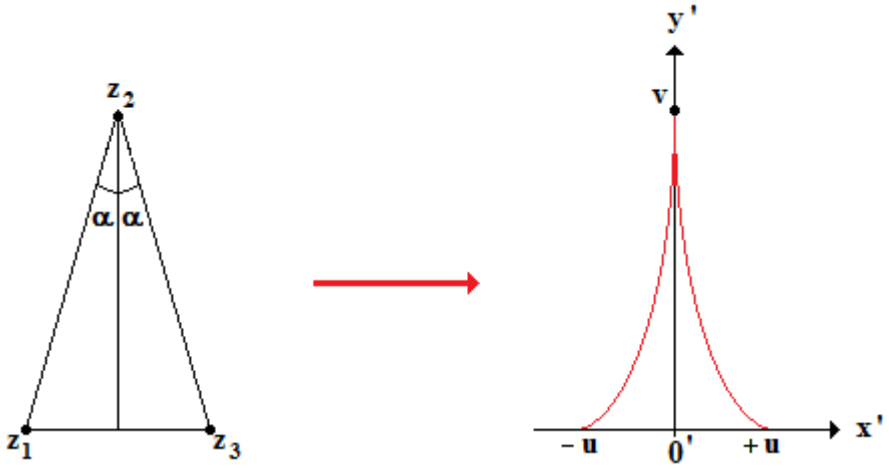
Mathématique & Informatique

Etoile parabolique



Etoile elliptique

D'une manière générale, on peut donner à la pointe de chaque branche la forme symétrique que l'on veut en choisissant une courbe paramétée x(t), y(t) passant par les deux sommets de la base du triangle :



Prenons, par exemple, l'ellipse de demi-axes u et v pour obtenir des branches très pointues.
Les équations paramétriques de la courbe dans le repère relatif (0',x',y') sont :

$$\begin{cases} x'(t) = u(1 - \cos t) \\ y'(t) = v(1 - \sin t) \end{cases}$$

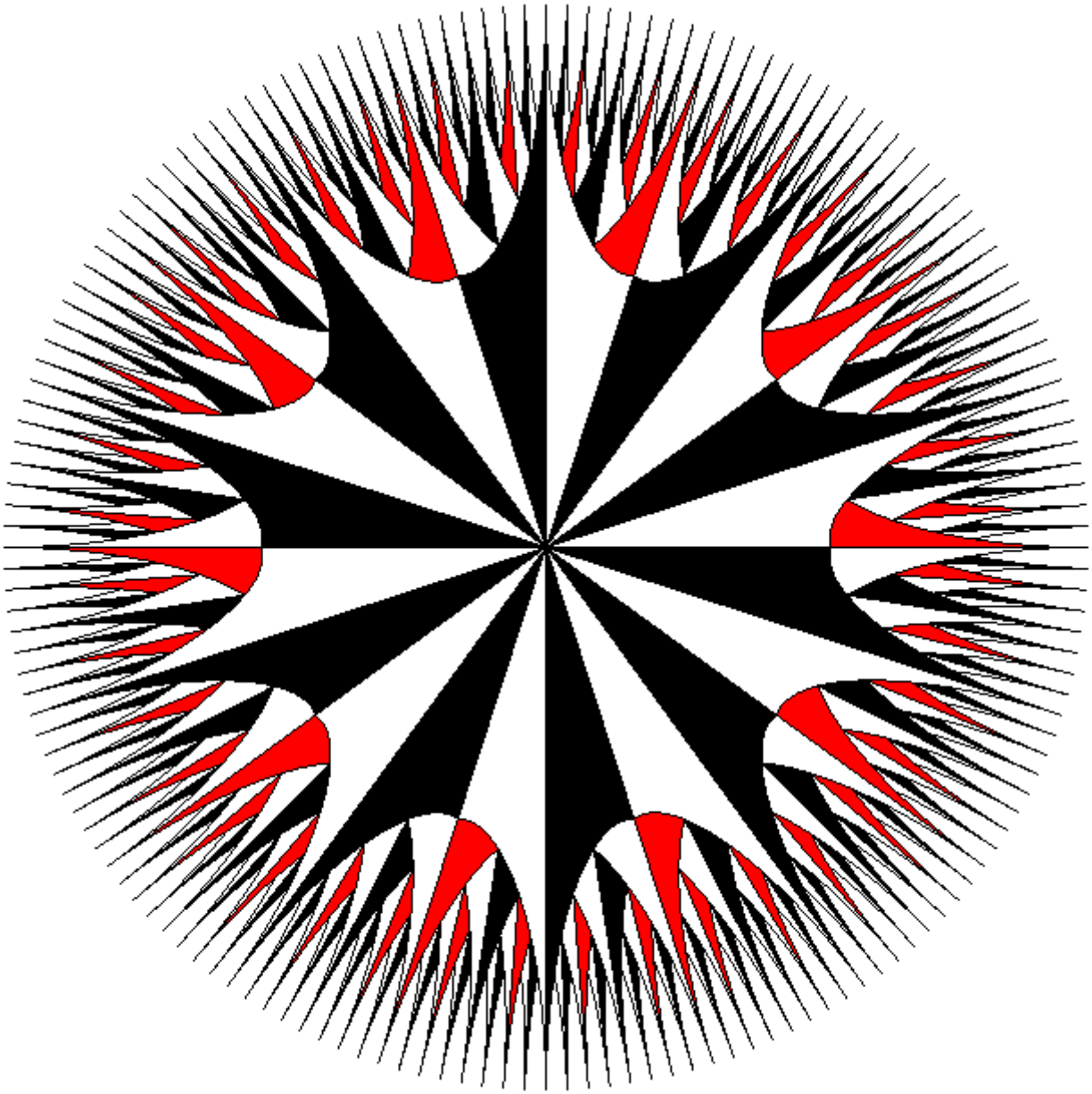
avec $t \in [0,\pi/2]$ (le paramètre t n'est pas l'angle polaire formé par un rayon vecteur avec un axe fixe)
La fonction de calcul du contour curviligne d'une branche est la suivante :

```
function B = ellipse(R,r,b)
    u = r*sin(%pi/b)
    v = R - r*cos(%pi/b)
    t = linspace(0,%pi/2,100)'
    X = u*(1-cos(t))
    Y = v*(1-sin(t))
    B1 = %i*r*cos(%pi/b) + (X + %i*Y)
    B2 = %i*r*cos(%pi/b) + (-X + %i*Y)
    B = [0,0;B1,B2]
endfunction
```

Pour une étoile décagonale à 5 niveaux de branche, le programme appelant est :

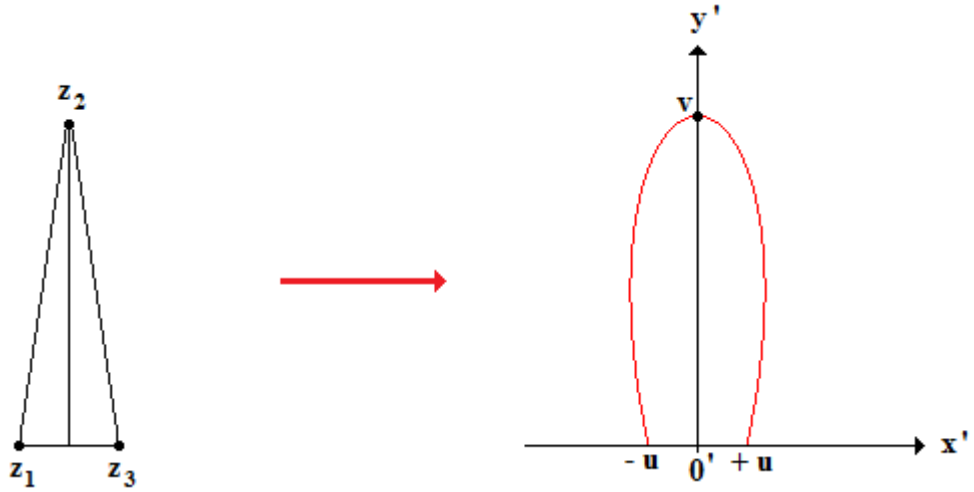
```
n = 5 ; b = 10 ;
R = 1 ; r = R/(2*cos(%pi/b)) ;
B = ellipse(R,r,b) ;
C = ones(ceil(n/2),1).*.[8,1;8,5] ;
f3 = scf(3) ;
drawlater()
plot2d(%inf,%inf,frameflag = 3,rect = [-R,-R,R,R],axesflag = 0)
Star(B,C,b,n) ;
drawnow()
```

Etoile elliptique



Une marguerite

Chaque branche de l'étoile est une rosace d'équation polaire : $\rho = R\ \mathbf{cos}(k\theta)$



Les équations paramétriques du contour curviligne dans le repère absolu (0,x,y) sont :

$$\begin{cases} x(\theta) = R\cos(k\theta)\sin(\theta) \\ y(\theta) = R\cos(k\theta)\cos(\theta) \end{cases}$$

avec $\theta \in [0,\theta_m]$ et θ_m = **Arc cos**(r/R) / k (θ_m est la valeur maximale de l'angle polaire θ sur une demi-branche)
On attribue à la constante k la valeur E(b/2) où b est le nombre de branches de niveau 1.
Pour représenter la partie pollinifère de la fleur, on rajoute sur le dessin un disque jaune de centre 0 et de rayon r .
Fonction de calcul d'un pétale :

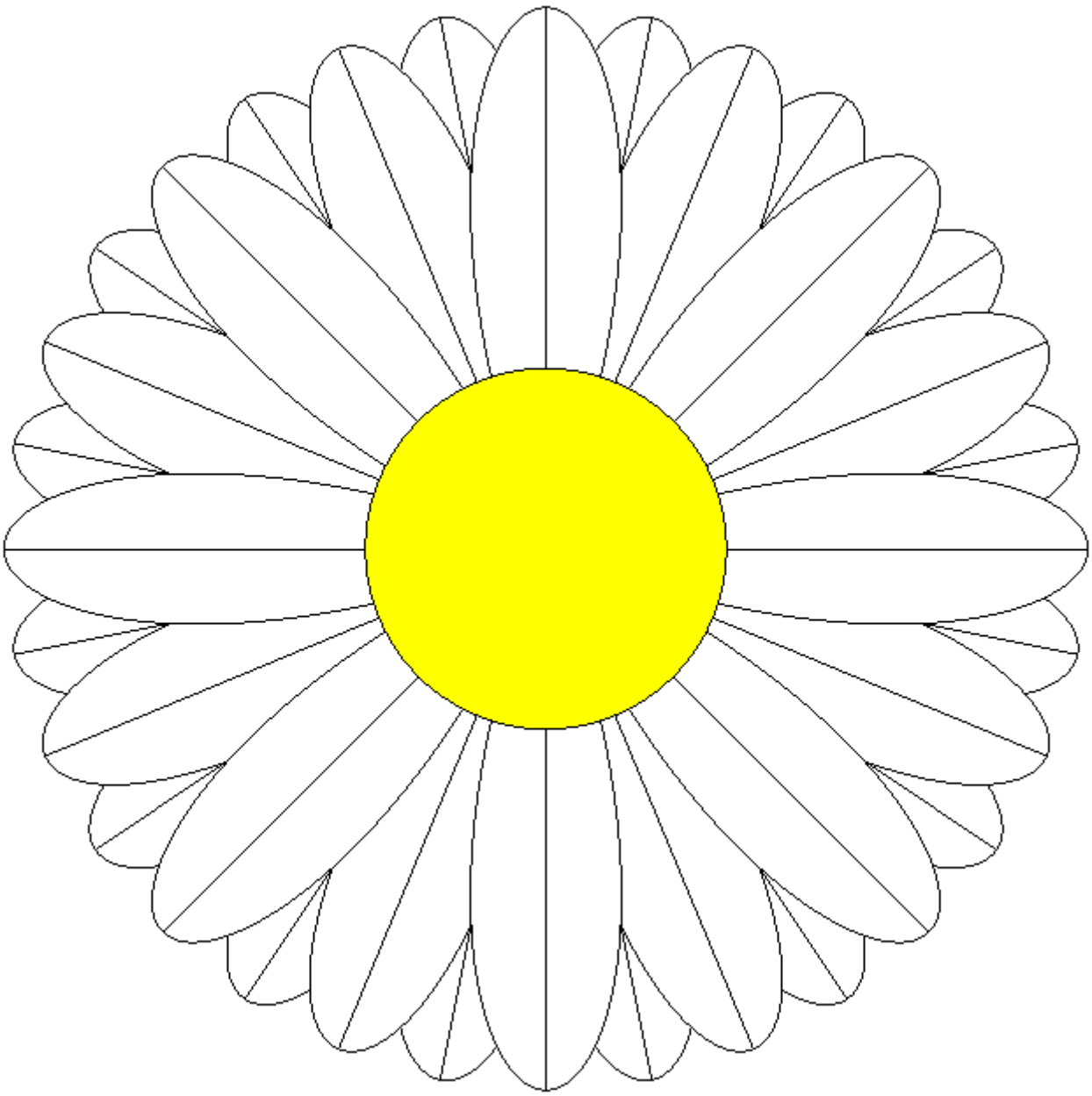
```
function B = marguerite(R,r,b)
    k = floor(b/2)
    thetam = acos(r/R)/k
    theta = linspace(0,thetam,100)'
    X = R*cos(k*theta).*sin(theta)
    Y = R*cos(k*theta).*cos(theta)
    B1 = X + %i*Y
    B2 = -X + %i*Y
    B = [0,0;B1,B2]
endfunction
```

Programme principal pour une marguerite à 32 pétales (étoile à 8 branches sur 3 niveaux) :

```
n = 3 ; b = 8 ;
R = 1 ; r = R/3 ;
B = marguerite(R,r,b) ;
C = ones(n,1)*[8,8] ;
f4 = scf(4) ;
drawlater()
plot2d(%inf,%inf,frameflag = 3,rect = [-R,-R,R,R],axesflag = 0)
Star(B,C,b,n) ;
xset('color',7) ; xfarc(-r,+r,2*r,2*r,0,64*360)
xset('color',1) ; xarc(-r,+r,2*r,2*r,0,64*360)
drawnow()
```

Mathématique & Informatique

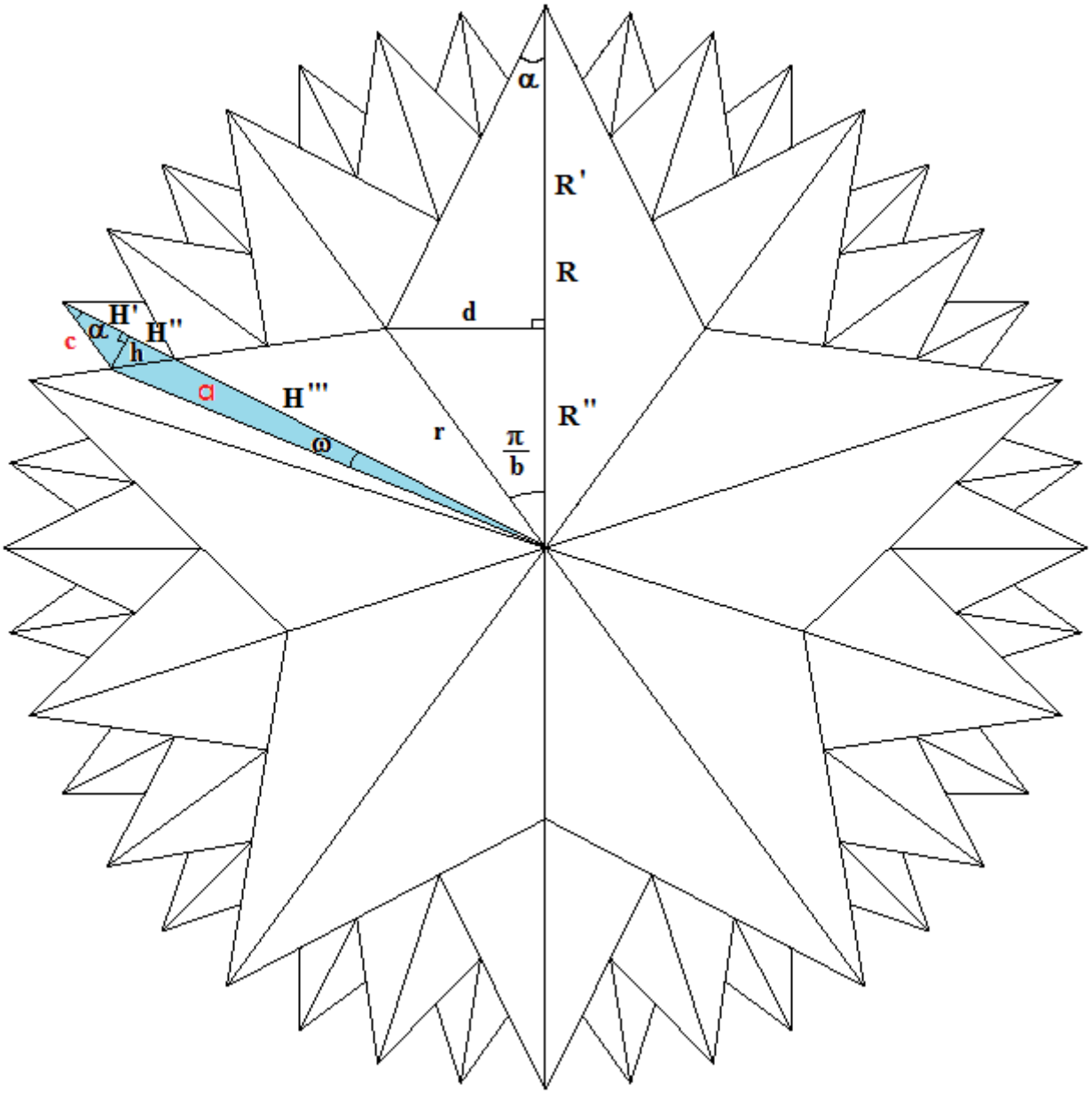
Marguerite



Mathématique & Informatique

Eléments de géométrie de l'étoile

Considérons une étoile à b branches sur n niveaux obtenue par rotation d'un quadrilatère de paramètres R et r.
Par exemple : n = 4, b = 5, R = 1, r = 1/2 correspond à l'étoile suivante :



L'angle α et l'angle ω

On a : d = r**sin**(π/b) et R" = r**cos**(π/b)
On déduit : R' = R - R" = R - r**cos**(π/b) et **cotan**(α) = R'/d

$$\operatorname{cotan}\alpha = \frac{\frac{R}{r} - \cos\frac{\pi}{b}}{\sin\frac{\pi}{b}}$$

Lorsque le niveau p de rotation augmente de 1, l'angle θ de rotation est divisé par 2. Ainsi :

$$\theta = \frac{\pi}{2^{p-2}b} \qquad (1 \le p \le n)$$

Toutes les pointes de l'étoile ont le même angle α au sommet. Par suite, l'angle ω est la moitié de l'angle au centre formé par les 2 rayons R, c'est-à-dire : ω = θ/2. Pour p = n, on obtient : ω = π/(2^{n-1}b)

$$\operatorname{cotan}\omega = \operatorname{cotan}\frac{\pi}{2^{n-1}b}$$

Le côté c et l'aire *a*

On a : H' = h**cotan**(α) et H" + H"' = h**cotan**(ω)
On en déduit : R = H' + H" + H"' = h(**cotan**(α) + **cotan**(ω))
D'une part : c = (H'^{2} + h^2)$^{1/2}$ = (h^2**cotan**^{2}(α) + h^2)$^{1/2}$ = h(1 + **cotan**^{2}(α))$^{1/2}$

$$c = \frac{R\sqrt{1 + \operatorname{cotan}^2\alpha}}{\operatorname{cotan}\alpha + \operatorname{cotan}\omega}$$

D'autre part : $a = \frac{1}{2}$ Rh

$$a = \frac{R^2}{2(\operatorname{cotan}\alpha + \operatorname{cotan}\omega)}$$

Le périmètre P et la surface S

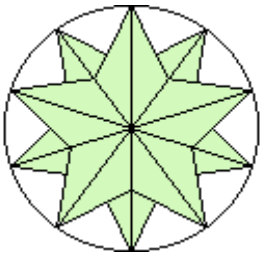
L'étoile d'ordre n possède un périmètre P constitué de 2^nb côtés c et une surface S constituée de 2^nb aires *a*.
Les expressions de P et S en fonction de n, b, R, r sont :

$$P = \frac{2^n b R\sqrt{1 + \operatorname{cotan}^2\alpha}}{\operatorname{cotan}\alpha + \operatorname{cotan}\omega}$$

$$S = \frac{2^{n-1} b R^2}{2(\operatorname{cotan}\alpha + \operatorname{cotan}\omega)}$$

Remarque

Pour certaines valeurs de b, on peut exprimer le périmètre P et la surface S de l'étoile avec des racines carrées.
Prenons l'exemple n = 2, b = 5, R = 1, r = 1/2 correspondant à la figure suivante :



On a :

$$P = \frac{20\sqrt{1 + \cot an^2\alpha}}{\cot an\alpha + \cot an\omega} \quad \text{et} \quad S = \frac{10}{\cot an\alpha + \cot an\omega}$$

avec :

$$\cot an\alpha = \frac{2 - \cos(\pi/5)}{\sin(\pi/5)} \quad \text{et} \quad \cot an\omega = \cot an\frac{\pi}{10}$$

La somme des nombres complexes $e^{i(k-1)\frac{2\pi}{5}}$, $1 \le k \le 5$, est nulle. On déduit :

$$\cos\frac{\pi}{5} = \frac{1}{4}\left(1 + \sqrt{5}\right) = \frac{\varphi}{2} = \lambda \qquad \varphi \text{ nombre d'or}$$

Par suite :

$$\sin\frac{\pi}{5} = \sqrt{1 - \lambda^2} \;\; ; \;\; \cos\frac{\pi}{10} = \sqrt{\frac{1 + \lambda}{2}} \;\; ; \;\; \sin\frac{\pi}{10} = \sqrt{\frac{1 - \lambda}{2}}$$

Posons : $u = \sqrt{1 + \lambda}$ et $v = \sqrt{1 - \lambda}$, on obtient :

$$P = \frac{20\sqrt{1 + \left(\frac{1 + v^2}{uv}\right)^2}}{\frac{1 + v^2}{uv} + \frac{u}{v}} \quad \text{et} \quad S = \frac{10}{\frac{1 + v^2}{uv} + \frac{u}{v}}$$

Après simplifications :

$$P = \frac{20}{3}\sqrt{5 - 4\lambda} \quad \text{et} \quad S = \frac{10}{3}\sqrt{1 - \lambda^2}$$

Soit :

$$P = \frac{20}{3}\sqrt{4 - \sqrt{5}}$$

$$S = \frac{5}{6}\sqrt{10 - 2\sqrt{5}}$$

Approximativement :

$$P \approx 8.854206$$

$$S \approx 1.959284$$

Plus généralement, si l'on peut exprimer le rapport R/r avec des racines carrées et s'il en est de même de cos(π/b), alors il est possible d'exprimer le périmètre P et la surface S de l'étoile avec des racines carrées. En effet :

$$\operatorname{cotan}\alpha = \frac{\frac{R}{r} - \cos\frac{\pi}{b}}{\sin\frac{\pi}{b}} = \frac{\frac{R}{r} - \cos\frac{\pi}{b}}{\sqrt{1 - \cos^2\frac{\pi}{b}}} \quad \text{et} \quad \operatorname{cotan}\omega = \operatorname{cotan}\frac{\pi}{2^{n-1}b} = \frac{\cos\frac{\pi}{2^{n-1}b}}{\sqrt{1 - \cos^2\frac{\pi}{2^{n-1}b}}}$$

De la relation : $\cos\theta = \frac{1}{2}\sqrt{2 + 2\cos 2\theta}$, on déduit :

$$\cos\frac{\pi}{2^{n-1}b} = \frac{1}{2}\sqrt{2 + 2\cos\frac{\pi}{2^{n-2}b}} = \frac{1}{2}\sqrt{2 + \sqrt{2 + 2\cos\frac{\pi}{2^{n-3}b}}} = \cdots$$

$$= \frac{1}{2}\sqrt{2 + \sqrt{2 + \cdots + \sqrt{2 + 2\cos\frac{\pi}{b}}}} \qquad (n-1) \text{ racines carrées}$$

En particulier, si $b = 2^m$ est une puissance de 2 (2, 4, 8, 16, ...), c'est toujours possible puisque :

$$\cos\frac{\pi}{2^m} = \frac{1}{2}\sqrt{2 + \sqrt{2 + \cdots + \sqrt{2}}} \qquad (m-1) \text{ racines carrées}$$

Valeurs exactes des cosinus et des sinus d'angles particuliers :

θ	cos θ	sin θ
$\frac{\pi}{2}$	0	1
$\frac{\pi}{3}$	$\frac{1}{2}$	$\frac{1}{2}\sqrt{3}$
$\frac{\pi}{4}$	$\frac{1}{2}\sqrt{2}$	$\frac{1}{2}\sqrt{2}$
$\frac{\pi}{5}$	$\frac{1}{4}(\sqrt{5} + 1)$	$\frac{1}{4}\sqrt{10 - 2\sqrt{5}}$
$\frac{\pi}{6}$	$\frac{1}{2}\sqrt{3}$	$\frac{1}{2}$
$\frac{\pi}{8}$	$\frac{1}{2}\sqrt{2 + \sqrt{2}}$	$\frac{1}{2}\sqrt{2 - \sqrt{2}}$
$\frac{\pi}{10}$	$\frac{1}{4}\sqrt{10 + 2\sqrt{5}}$	$\frac{1}{4}(\sqrt{5} - 1)$
$\frac{\pi}{12}$	$\frac{1}{4}(\sqrt{6} + \sqrt{2})$	$\frac{1}{4}(\sqrt{6} - \sqrt{2})$

Mathématique & Informatique

A partir de la formule précédente, on obtient une expression du nombre π qui ne fait intervenir que le chiffre 2 et la racine carrée, analogue à la formule de **Viète** :

$$\pi = 2\prod_{n=1}^{\infty}\frac{2}{\sqrt{2_{(1)}+\sqrt{2_{(2)}+\cdots+\sqrt{2_{(n)}}}}}$$

On a :

$$\lim_{x\to 0}\frac{\sin x}{x}=1 \;\Rightarrow\; \sin\frac{\pi}{2^{n+1}}\underset{n\infty}{\sim}\frac{\pi}{2^{n+1}} \;\Rightarrow\; 2^{n+1}\sin\frac{\pi}{2^{n+1}}\underset{n\infty}{\to}\pi$$

Or :

$$\sin\frac{\pi}{2^{n+1}}=\sqrt{1-\cos^2\frac{\pi}{2^{n+1}}}=\sqrt{1-\frac{1}{4}\left[2_{(1)}+\sqrt{2_{(2)}+\cdots+\sqrt{2_{(n)}}}\right]}=\frac{1}{2}\sqrt{2_{(1)}-\sqrt{2_{(2)}+\cdots+\sqrt{2_{(n)}}}}$$

D'où :

$$\pi=\lim_{n\to+\infty}2^n\sqrt{2_{(1)}-\sqrt{2_{(2)}+\cdots+\sqrt{2_{(n)}}}}$$

Vérification numérique

Introduisons la suite récurrente (x_n) définie par :

$$\begin{cases} x_0 = 0 \\ x_n = \sqrt{2+x_{n-1}} \end{cases}$$

Le nombre $x_n=\sqrt{2_{(1)}+\sqrt{2_{(2)}+\cdots+\sqrt{2_{(n)}}}}$ présente n racines carrées, les radicandes sont indicés de 1 à n.
Exprimons π :

$$\pi=\lim_{n\to+\infty}2^n\sqrt{2-x_{n-1}}=\lim_{n\to+\infty}2^{n+1}\sqrt{2-x_n}$$

Limitons la précision à $n = 12$ et écrivons le programme correspondant :

```
n = 12 ;
x = 0 ;
for i = 1:n
    x = sqrt(2 + x) ;
end
pi = 2^(n+1)*sqrt(2-x) ;
disp(pi)
```

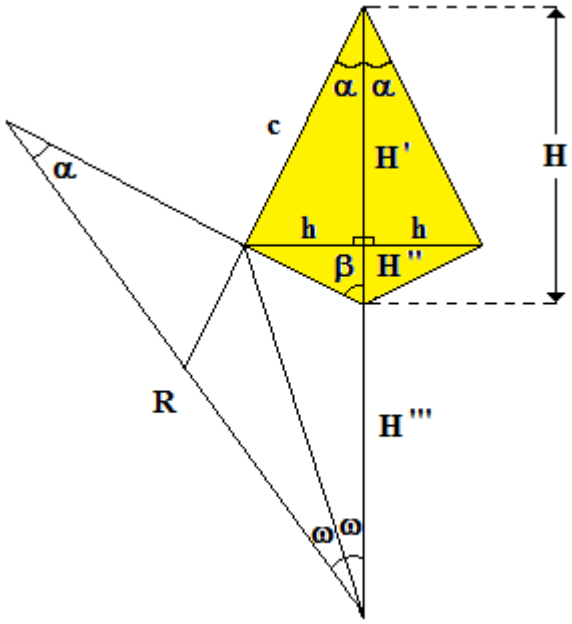
On trouve : $pi = 3.1415926$

Noter qu'il est difficile d'obtenir une grande précision sur π avec cette méthode car le calcul consiste à multiplier un grand nombre $\left(2^{n+1}\to\infty\right)$ avec un petit nombre $\left(\sqrt{2-x_n}\to 0\right)$ ce qui accroît les erreurs d'arrondi.

Mathématique & Informatique

Le quadrilatère Q

L'élément de base de l'étoile d'ordre n est le quadrilatère Q coloré en jaune sur le schéma ci-dessous :



A chaque niveau de rotation p ($1 \le p \le n$) correspond un type de quadrilatère Q. Calculons sa hauteur H.
Sachant que la somme des angles d'un triangle quelconque est égale à π, on trouve : $\beta = \alpha + 2\omega$
D'où : $H'' = h\mathbf{cotan}(\alpha + 2\omega)$
Comme : $H' = h\mathbf{cotan}(\alpha)$ et $h = R/(\mathbf{cotan}(\alpha) + \mathbf{cotan}(\omega))$
On en déduit : $H = H' + H'' = [(\mathbf{cotan}(\alpha) + \mathbf{cotan}(\alpha + 2\omega))/(\mathbf{cotan}(\alpha) + \mathbf{cotan}(\omega))]R$
Finalement, on aboutit aux formules suivantes :

$$h = \frac{1}{\operatorname{cotan}\alpha + \operatorname{cotan}\omega} R$$

$$H = \frac{2(\operatorname{cotan}\alpha \operatorname{cotan}\omega - 1)}{\operatorname{cotan}^2\omega + 2\operatorname{cotan}\alpha \operatorname{cotan}\omega - 1} R$$

$$H' = \frac{\operatorname{cotan}\alpha}{\operatorname{cotan}\alpha + \operatorname{cotan}\omega} R$$

$$H'' = \frac{\operatorname{cotan}\alpha \operatorname{cotan}^2\omega - \operatorname{cotan}\alpha - 2\operatorname{cotan}\omega}{(\operatorname{cotan}^2\omega + 2\operatorname{cotan}\alpha \operatorname{cotan}\omega - 1)(\operatorname{cotan}\alpha + \operatorname{cotan}\omega)} R$$

Ces expressions sont vraies pour $p \ge 2$.
Pour p = 1, les valeurs recherchées se déduisent de la figure représentant l'étoile :

$\beta = \pi/b$, $h = r\mathbf{sin}(\pi/b)$, $H' = R - r\mathbf{cos}(\pi/b)$, $H'' = r\mathbf{cos}(\pi/b)$, $H = R$

Rappelons que l'angle ω est une fonction du niveau de rotation p : $\omega = \frac{\pi}{2^{p-1}b}$

On vérifie aisément que le quadrilatère Q tend vers le losange de côté $\pi R/(2^{p-1}b\sin\alpha)$ lorsque $p \to \infty$

Mathématique & Informatique

L'étoile

La fonction **cotan**(x) est équivalente à 1/x quand $x \to 0$
Par conséquent, lorsque $n \to \infty$, on a : **cotan**(ω) ~ $(2^{n-1}b)/\pi \to \infty$
La valeur **cotan**(α) est indépendante de n, donc constante.
D'où, quand n tend vers l'infini :

$$P \sim 2\pi R \sqrt{1 + \left(\frac{\frac{R}{r} - \cos\frac{\pi}{b}}{\sin\frac{\pi}{b}} \right)^2}$$

$$S \sim \pi R^2$$

Il apparaît que l'étoile couvre le disque de rayon R.
Son périmètre P est constitué d'une infinité de segments de longueur tendant vers 0 dont la somme est finie et toujours supérieure à $2\pi R$. En réalité, le périmètre P peut être arbitrairement grand en augmentant le nombre de branches b de l'étoile d'ordre 1. Remarquons que ce résultat est en accord avec le théorème des isopérimètres qui stipule qu'aucune courbe de périmètre inférieur à celui du cercle de rayon R ne contient la surface πR^2. Ajoutons que les pointes de l'étoile rencontrent le cercle de rayon R en 2^{n-1} b points distants d'un arc de longueur $\pi R/(2^{n-2}b)$, valeur de plus en plus petite mais jamais nulle. L'étoile a donc une infinité de points équidistants sur le cercle de périmètre $2\pi R$.

On peut considérer l'étoile comme une fractale constituée d'une infinité de petits quadrilatères Q.
Recherchons sa dimension fractale.

⇨ http://www.mat.ulaval.ca/fileadmin/Cours/MAT-19519/Notes_de_cours/DimensionFractale.pdf

Si l'on utilise la définition de **Bouligand-Minkowski**, équivalente à la méthode du Box-counting, on a :

$$D = \lim_{\varepsilon \to 0} \left(2 - \frac{\log A}{\log \varepsilon} \right) = \lim_{\varepsilon \to 0} \frac{\log N}{\log\left(\frac{1}{\varepsilon}\right)}$$

où A est une aire approchant la fractale à ε près, et N est le nombre de carrés de côté ε nécessaires pour couvrir tous les éléments de la fractale.
Comme on a montré la convergence de l'étoile vers le disque de rayon R lorsque $n \to \infty$, on prend l'aire A égale à celle du disque de rayon R + ε, soit $A = \pi(R + \varepsilon)^2$. Quand $\varepsilon \to 0$, le numérateur $\log \pi(R + \varepsilon)^2$ tend vers $\log \pi R^2$ et le dénominateur log ε tend vers - ∞, donc $\log A / \log \varepsilon \to 0$. En conséquence, la dimension fractale de l'étoile, selon la définition de **Bouligand-Minkowski**, est D = 2.
En fait, cette définition de la dimension fractale convient bien pour caractériser un type de fractale dont les éléments sont tous identiques à l'ordre n avec un rapport d'homothétie constant pour les construire.
Tel est le cas du carré ou du triangle de **Sierpinski** où l'on peut facilement appliquer la méthode du Box-counting :

Carré de **Sierpinski** : $N = 8^n$ petits carrés identiques dans des carrés de côté $\varepsilon = 1/3^n$. D'où : $D = \log 8/\log 3$
Triangle de **Sierpinski** : $N = 3^n$ petits triangles identiques dans des carrés de côté $\varepsilon = 1/2^n$. D'où : $D = \log 3/\log 2$

Dans le cas de l'étoile, les quadrilatères qui la compose diffèrent en fonction du niveau de rotation et ne sont pas homothétiques puisque leurs angles homologues ne sont pas égaux (l'angle α est constant mais l'angle β dépend du niveau de rotation).

Etoile hérissée

Construisons une nouvelle étoile, similaire à la précédente, mais avec des pointes de plus en plus étroites lorsque le niveau de branche augmente. Pour obtenir ce résultat, l'angle α au sommet d'une branche est divisé par un facteur constant $f > 1$ quand le niveau de branche p croît de 1. Ainsi :

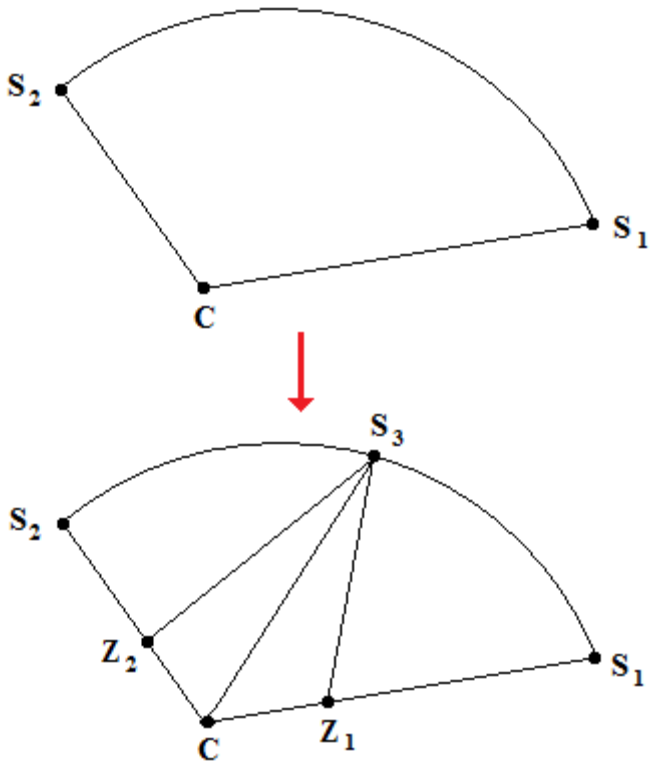
$$\alpha_p = \frac{\alpha}{f^{p-1}} \qquad (1 \le p \le n)$$

avec α défini par : $$\operatorname{cotan}\alpha = \frac{\frac{R}{r} - \cos\frac{\pi}{b}}{\sin\frac{\pi}{b}}$$

L'étoile ordinaire précédente correspond en fait au cas particulier $f = 1$. Lorsqu'il n'y a pas de rétrécissement des pointes, on peut dessiner l'étoile d'ordre n par de simples rotations de l'étoile d'ordre 1. Cette méthode ne s'applique pas si $f > 1$. Une étoile hérissée est caractérisée par 5 paramètres :

- n : le nombre de niveaux de branche
- b : le nombre de branches de niveau 1
- R : le grand rayon d'une branche de niveau 1
- r : le petit rayon d'une branche de niveau 1
- f : le facteur de rétrécissement des pointes

Le principe de construction de l'étoile repose sur l'insertion d'un quadrilatère dans un secteur angulaire :

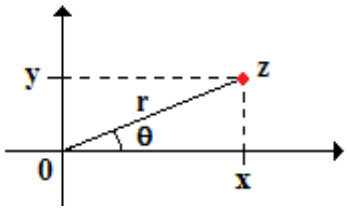


Les sommets S_1, S_2 et S_3 appartiennent au même cercle de centre O et de rayon *R*. Le sommet S_3 est au milieu de l'arc de cercle S_1S_2. L'aspect récurrent du processus apparaît immédiatement puisqu'on peut renouveler l'opération sur les secteurs (Z_1,S_1,S_3) et (Z_2,S_3,S_2).

- Présentons brièvement les fonctions Scilab de base pour utiliser les nombres complexes.

Soit un nombre complexe z défini sous forme algébrique, trigonométrique et exponentielle :

$$z = x + iy = r(\cos\theta + i\sin\theta) = re^{i\theta}$$



La partie réelle de z est : x = real(z) x ∈]-∞ , +∞[

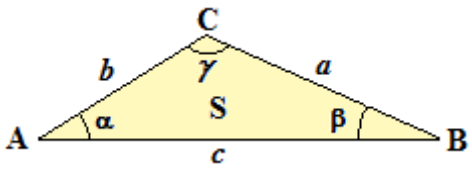
La partie imaginaire de z est : y = imag(z) y ∈]-∞ , +∞[

Le module de z est : r = abs(z) r ∈ [0 , +∞[

L'argument de z est : θ = arg(z) θ ∈]-π , +π]

- Ajoutons quelques résultats sur la géométrie du triangle dans $\mathbb{C}$.

Soit un triangle quelconque de surface S, de sommets (A,B,C), d'affixes (z_A,z_B,z_C) et de côtés (a,b,c).



On a :

$$z_C = z_B + (z_A - z_B)k\,e^{-i\beta} \qquad (\text{Similitude de centre B, de rapport } k \text{ et d'angle } -\beta)$$

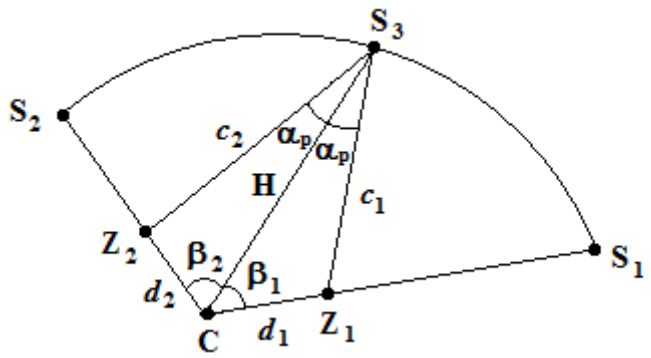
$$\beta = \text{Arg}\left(\frac{z_A - z_B}{z_C - z_B}\right) \qquad (\beta = \text{angle des vecteurs } \overrightarrow{BC} \text{ et } \overrightarrow{BA})$$

$$k = \frac{a}{c} = \frac{\sin\alpha}{\sin(\alpha+\beta)} \qquad \left(\text{Règle des sinus : } \frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma}\right)$$

$$S = \frac{1}{2}\left|\,\text{Im}\left[(z_A - z_B)\overline{(z_A - z_C)}\,\right]\right| \qquad \left(\text{Surface du triangle : } S = \frac{1}{2}\left\|\overrightarrow{AB} \wedge \overrightarrow{AC}\right\|\right)$$

Dans ces expressions **Arg** désigne la fonction Argument et **Im** la fonction Imaginaire d'un nombre complexe.

Calculons maintenant les différentes affixes du secteur angulaire.

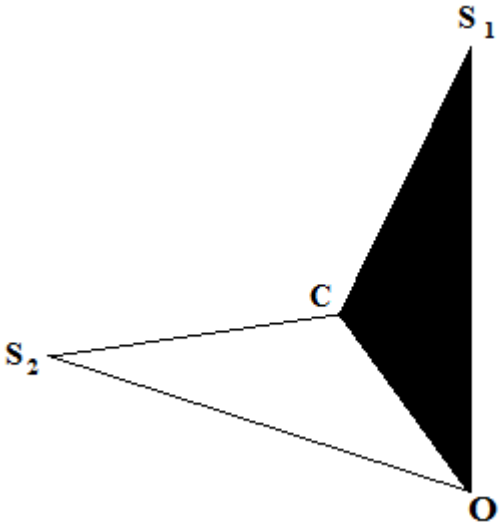


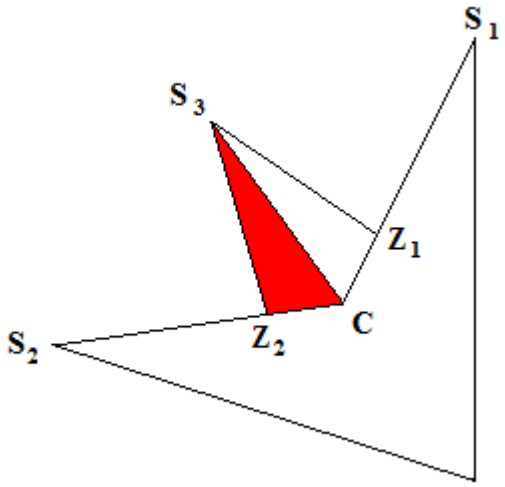
Le sommet S_3, situé sur la bissectrice de l'angle $\widehat{S_1OS_2}$, appartient au cercle de centre O et de rayon R.
Notons β_1 et β_2 les angles $(\overrightarrow{CS_1}, \overrightarrow{CS_3})$ et $(\overrightarrow{CS_3}, \overrightarrow{CS_2})$, k_1 et k_2 les rapports d_1/H et d_2/H.
On obtient :

$$S_3 = Re^{\,i\,\mathrm{Arg}\,(S_1+S_2)}$$

$\beta_1 = \mathrm{Arg}\left(\dfrac{S_3 - C}{S_1 - C}\right)$	$\beta_2 = \mathrm{Arg}\left(\dfrac{S_2 - C}{S_3 - C}\right)$
$k_1 = \dfrac{\sin\alpha_p}{\sin(\alpha_p + \beta_1)}$	$k_2 = \dfrac{\sin\alpha_p}{\sin(\alpha_p + \beta_2)}$
$Z_1 = C + (S_3 - C)k_1\,e^{-i\beta_1}$	$Z_2 = C + (S_3 - C)k_2\,e^{+i\beta_2}$

Dans le cas particulier où $f = 1$, on a : $\alpha_p = \alpha = \mathrm{Cte} \Rightarrow \beta_1 = \beta_2 = \beta$ et $k_1 = k_2 = k$





Le dessin de l'étoile s'effectue de manière récursive avec la fonction *Branche* qui insère le quadrilatère (C,Z_1,S_3,Z_2) dans le secteur angulaire (C,S_1,S_2).
On désigne par *S* la liste des affixes des sommets S_k de l'étoile d'ordre *n* et par *Peri* et *Surf* les variables qui contiennent respectivement la valeur du périmètre et la valeur de la surface de l'étoile. *Surf* se calcule sur les nœuds de l'arbre des appels, autrement dit à chaque fois que l'on crée une nouvelle branche. Par contre, *Peri* se calcule uniquement sur les extrémités de l'arbre des appels (sur les feuilles) lorsqu'on ne crée plus de nouvelle branche.
Concernant la programmation, il est indispensable de déclarer les variables *S*, *Peri*, *Surf* en mode global pour que la fonction *Branche* puisse les modifier.

```
// Paramètres de l'étoile
n = 5 ; b = 10 ; R = 1 ; r = 1/2 ; f = 5/2 ;
alpha = acot((R/r-cos(%pi/b))/sin(%pi/b)) ;

// Argument d'un nombre complexe
function a = arg(z)
    a = atan(imag(z),real(z))
endfunction

// Aire   d'un triangle
function A = aire(za,zb,zc)
    A = abs(imag((za-zb)*conj(za-zc)))/2
endfunction

// Branche en forme de quadrilatère
function Branche(C,S1,S2,p)
    global S Peri Surf
    if p > 0 then
        alphap = alpha/f^(n-p)
        S3 = R*exp(%i*arg(S1+S2))
        beta1 = arg((S3-C)/(S1-C))
        k1 = sin(alphap)/sin(alphap+beta1)
        Z1 = C + (S3-C)*k1*exp(-%i*beta1)
        beta2 = arg((S2-C)/(S3-C))
        k2 = sin(alphap)/sin(alphap+beta2)
        Z2 = C + (S3-C)*k2*exp(+%i*beta2)
        Q = [C,C;Z1,S3;S3,Z2]
        couleur = [8,4*modulo(n-p,2)+1]
        xfpolys(real(Q),imag(Q),couleur)
        Surf = Surf + aire(C,S3,Z1) + aire(C,S3,Z2)
        Branche(Z1,S1,S3,p-1)
        Branche(Z2,S3,S2,p-1)
    else
        S = lstcat(S,list(S1,C))
        Peri = Peri + abs(C-S1) + abs(C-S2)
    end
endfunction
```

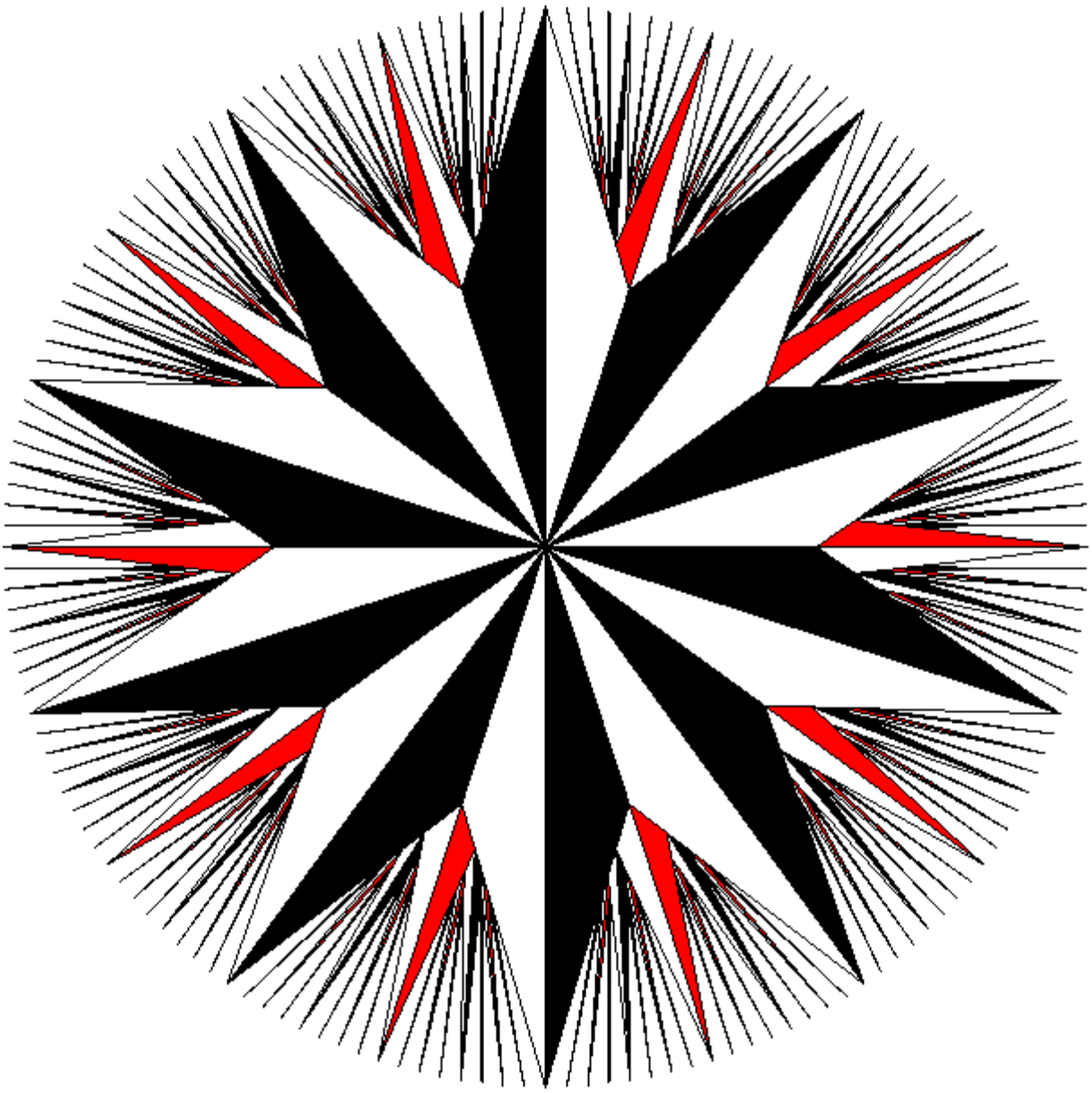
```
// Programme principal
f1 = scf(1) ;
xtitle('Etoile') ;
plot2d(%inf, %inf, frameflag = 3, rect = [-R,-R,R,R], axesflag = 0)
drawlater()
global S Peri Surf
S = list() ;
Peri = 0 ;
Surf = 0 ;
for k = 1:b
    C = r*exp(%i*(%pi/2 + (2*k-1)*%pi/b)) ;
    S1 = R*exp(%i*(%pi/2 + (2*k-2)*%pi/b)) ;
    S2 = R*exp(%i*(%pi/2 + (2*k)*%pi/b)) ;
    Q = [0,0;S1,C;C,S2] ;
    couleur = [1,8] ;
    xfpolys(real(Q),imag(Q),couleur)
    Surf = Surf + aire(0,C,S1) + aire(0,C,S2) ;
    Branche(C,S1,S2,n-1)
end
drawnow()
printf('\n Périmètre = %.4f',Peri)
printf('\n Surface = %.4f',Surf)
```

L'algorithme d'obtention de l'étoile hérissée s'écrit également de façon itérative à l'aide de 2 boucles imbriquées et d'une liste temporaire *St* en plus de la liste *S* (voir le code Scilab page suivante).
A noter que la procédure récursive et la procédure itérative ne dessinent pas l'étoile de la même manière.
La procédure récursive remplit le premier secteur de l'ordre 1 à l'ordre *n*, puis le deuxième secteur de l'ordre 1 à l'ordre *n*, jusqu'au b^e secteur de l'ordre 1 à l'ordre *n*. La procédure itérative remplit tous les secteurs de 1 à *b* en commençant par l'ordre 1, puis tous les secteurs à l'ordre 2, et ainsi de suite jusqu'à l'ordre *n*.

```
// Programme principal
f1 = scf(1) ;
xtitle('Etoile')
plot2d(%inf, %inf, frameflag = 3, rect = [-R,-R,R,R], axesflag = 0)
drawlater()
S = list() ;
Peri = 0 ;
Surf = 0 ;
for k = 1:b
    C = r*exp(%i*(%pi/2 + (2*k-1)*%pi/b)) ;
    S1 = R*exp(%i*(%pi/2 + (2*k-2)*%pi/b)) ;
    S2 = R*exp(%i*(%pi/2 + (2*k)*%pi/b)) ;
    Q = [0,0;S1,C;C,S2] ;
    couleur = [1,8] ;
    xfpolys(real(Q),imag(Q),couleur)
    S = lstcat(S,list(S1,C)) ;
    Peri = Peri + abs(C-S1) + abs(C-S2) ;
    Surf = Surf + aire(0,C,S1) + aire(0,C,S2) ;
end
S($+1) = S(1) ;
for p = 2:n
    St = list() ;
    alphap = alpha/f^(p-1) ;
    for k = 1:(length(S)-1)/2
        C = S(2*k) ;
        S1 = S(2*k-1) ;
        S2 = S(2*k+1) ;
        S3 = R*exp(%i*arg(S1+S2)) ;
        beta1 = arg((S3-C)/(S1-C)) ;
        k1 = sin(alphap)/sin(alphap+beta1) ;
        Z1 = C + (S3-C)*k1*exp(-%i*beta1) ;
        beta2 = arg((S2-C)/(S3-C)) ;
        k2 = sin(alphap)/sin(alphap+beta2) ;
        Z2 = C + (S3-C)*k2*exp(+%i*beta2) ;
        Q = [C,C;Z1,S3;S3,Z2] ;
        couleur = [8,4*modulo(p-1,2)+1] ;
        xfpolys(real(Q),imag(Q),couleur)
        St = lstcat(St,list(S1,Z1,S3,Z2)) ;
        Peri = Peri + abs(S3-Z1) + abs(S3-Z2) - abs(C-Z1) - abs(C-Z2) ;
        Surf = Surf + aire(C,S3,Z1) + aire(C,S3,Z2) ;
    end
    St($+1) = St(1) ;
    S = St ;
end
S($) = null() ;
drawnow()
printf('\n Périmètre = %.4f',Peri)
printf('\n Surface = %.4f',Surf)
```

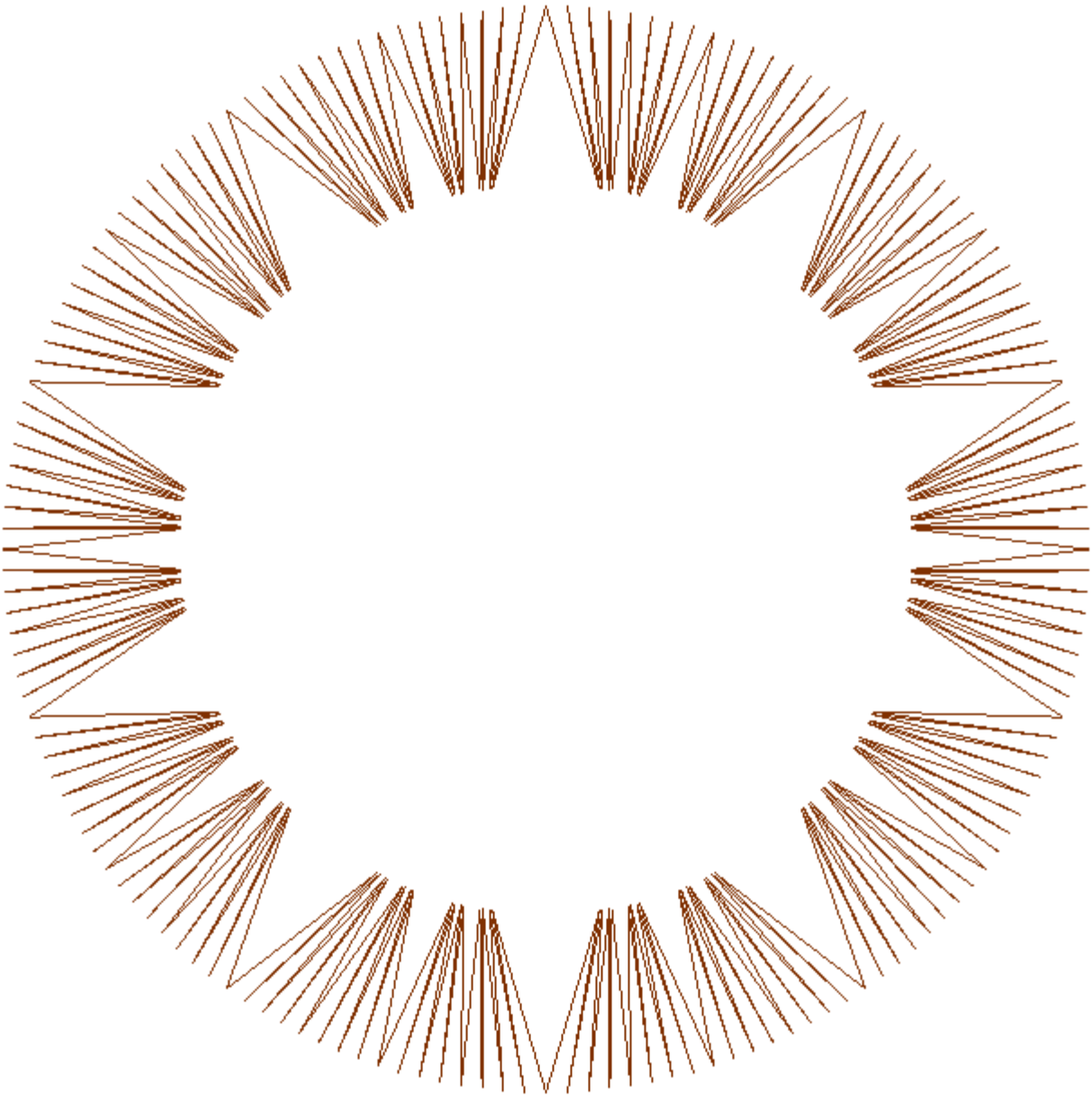
Mathématique & Informatique

L'étoile hérissée n = 5, b =10, R = 1, r = 1/2, f = 5/2



Mathématique & Informatique

L'enveloppe hérissée n = 5, b =10, R = 1, r = 1/2, f = 5/2



Mathématique & Informatique

Géométrie fractale

Vu la complexité des formules de l'étoile hérissée (f > 1), il n'est guère envisageable de donner l'expression analytique de son périmètre *Peri* ou de sa surface *Surf* comme on l'a fait pour l'étoile ordinaire (f = 1).
L'étoile est un polygone concave (non convexe), c'est-à-dire une ligne brisée fermée dont tous les sommets ne sont pas dans le même demi-plan délimité par une droite passant par l'un de ses côtés. Le calcul de son périmètre et de sa surface s'effectue de manière algorithmique. Il repose sur la propriété suivante : ajouter une nouvelle branche à l'étoile revient à modifier son périmètre de la longueur des segments (S_3,Z_1) et (S_3,Z_2), et à augmenter sa surface de l'aire des triangles (C,S_3,Z_1) et (C,S_3,Z_2).
Par exemple, pour n = 5, b = 5, R = 1, r = 1/2, f = 4/3, on trouve : Périmètre = 25.8888 ; Surface = 2.6514
En fin de programme, la liste S contient les affixes des sommets de l'étoile hérissée en commençant par le sommet d'ordonnée maximale (S(1) = Ri) puis en se déplaçant dans le sens trigonométrique.
Notons N le nombre d'éléments de la liste S.
Le nombre de branches de l'étoile de paramètres n et b est : $Nb = b + b + 2b + ... + 2^{n-2}b = 2^{n-1}b$
Le nombre de sommets de l'étoile de paramètres n et b est $2 \times Nb$.
Donc, pour une étoile de paramètres n et b :

$$N = 2^n b$$

Tous les sommets d'indices impairs S(2k-1), $1 \le k \le 2^{n-1}b$, sont sur le cercle de rayon R et tous les sommets d'indices pairs S(2k), $1 \le k \le 2^{n-1}b$, sont à l'intérieur. On connait l'expression analytique des points équidistants S(2k-1) :

$$S(2k-1) = Re^{i\left[\frac{\pi}{2} + \frac{(k-1)\pi}{2^{n-2}b}\right]} \qquad k = 1, 2, \ldots, 2^{n-1}b$$

A partir de la liste S, il est possible de dessiner le contour de l'étoile, de calculer son périmètre et sa surface :

- Soit LE la liste égale à la liste S augmentée de son premier élément en queue de liste (LE ≡ S + S(1)).

L'enveloppe de l'étoile de paramètres n et b s'obtient en parcourant la liste LE et en traçant, les uns après les autres, les segments [LE(k);LE(k+1)], $1 \le k \le 2^n b$.

- Soit LP la liste égale à la liste S augmentée de son premier élément en queue de liste (LP ≡ S + S(1)), on a :

$$Peri = \sum_{k=1}^{2^n b} Segment\left(LP(k), LP(k+1)\right)$$

- Soit LS la liste égale à la liste S augmentée de son dernier élément en tête de liste (LS ≡ S($) + S), on a :

$$Surf = \sum_{k=1}^{2^{n-1}b} \left[Triangle(LS(2k-1), LS(2k), LS(2k+1)) + Triangle(0, LS(2k-1), LS(2k+1))\right]$$

Remarque

Dans ces formules, *Segment* est la longueur d'un segment et *Triangle* est la surface d'un triangle.
Les S(k), $1 \le k \le 2^n b$, sont des nombres complexes, par suite la fonction *Segment* s'identifie à la fonction *abs* de Scilab et la fonction *Triangle* à la fonction *aire* précédemment définie.

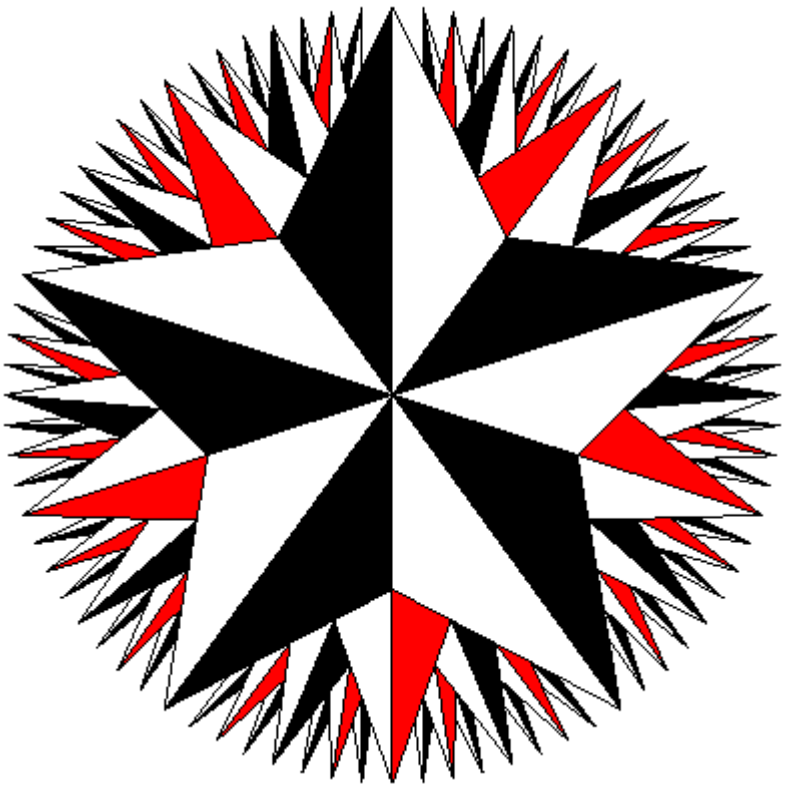
```
// Dessin de l'enveloppe
f2 = scf(2) ;
xtitle('Enveloppe')
plot2d(%inf, %inf, frameflag = 3, rect = [-R,-R,R,R], axesflag = 0)
drawlater()
LE = S ;
LE($+1) = S(1) ;
for k = 1:length(LE)-1
    seg = [LE(k);LE(k+1)] ;
    xsegs(real(seg),imag(seg),25)
end
drawnow()

// Calcul du périmètre
LP = S ;
LP($+1) = S(1) ;
Peri = 0 ;
for k = 1:length(LP)-1
    Peri = Peri + abs(LP(k+1) - LP(k)) ;
end
printf('\n Périmètre = %.4f',Peri)

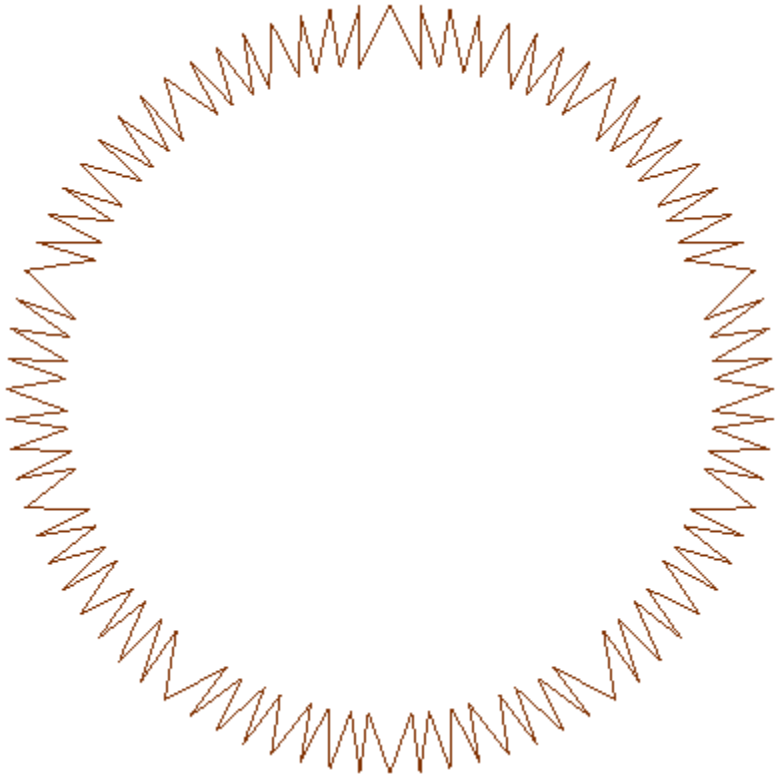
// Calcul de la surface
LS = S ;
LS(0) = S($) ;
Surf = 0 ;
for k = 1:(length(LS)-1)/2
    Surf = Surf + aire(LS(2*k-1),LS(2*k),LS(2*k+1)) + aire(0,LS(2*k-1),LS(2*k+1)) ;
end
printf('\n Surface = %.4f',Surf)
```

L'étoile hérissée de paramètres n = 5, b = 5, R = 1, r = 1/2, f = 4/3 correspond aux figures ci-dessous :

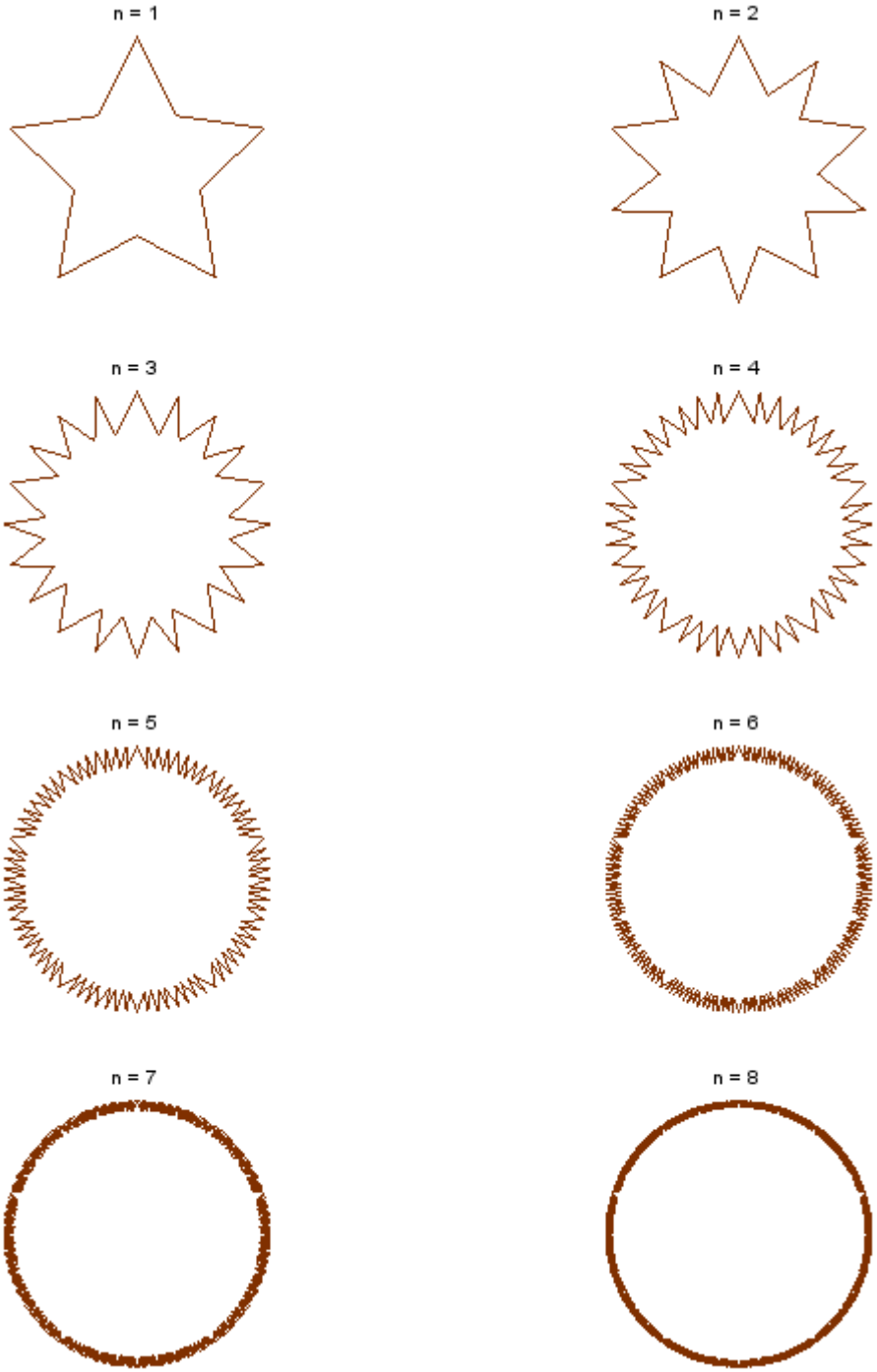
Etoile



Enveloppe



Voici les différentes enveloppes de l'étoile hérissée b = 5, R = 1, r = 1/2, f = 4/3 lorsque n varie de 1 à 8 :



Au fur et à mesure que n augmente, le contour de l'étoile est de plus en plus fin et de plus en plus rugueux.
Quand $n \to +\infty$, l'étoile hérissée devient extrêmement pointue et son périmètre tend vers l'infini.
A grande échelle et pour de grandes valeurs de n, le contour semble lisse ; ce n'est qu'à petite échelle qu'apparaît sa structure irrégulière constituée d'une multitude de petits segments.

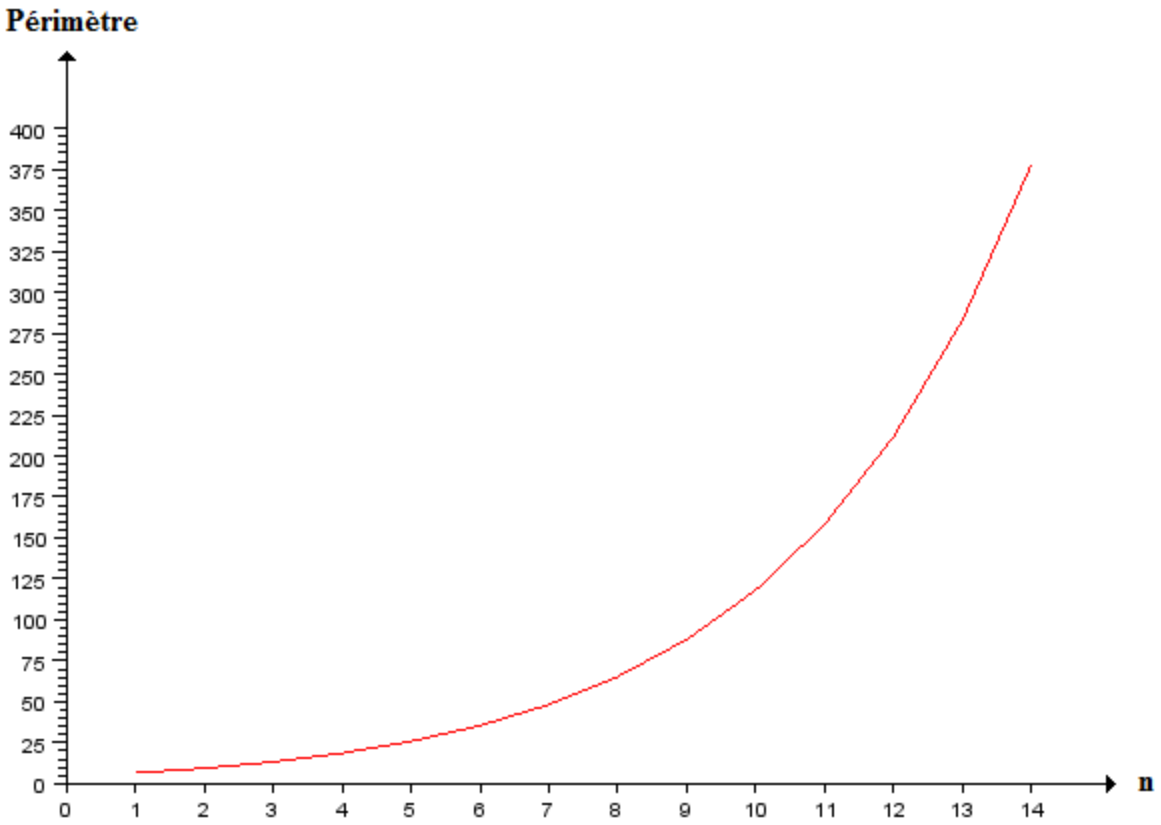
Cliquer sur  dans la barre d'outils pour visualiser les détails d'une figure avec le curseur en forme de loupe.

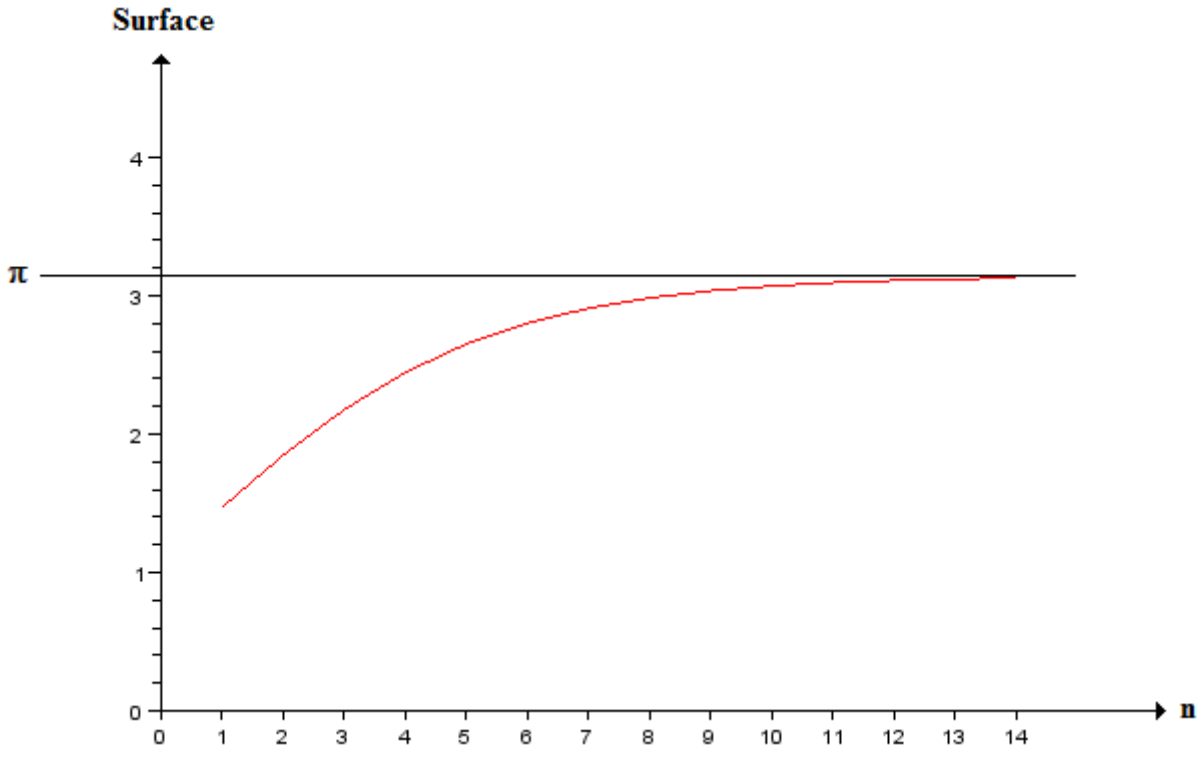
Mathématique & Informatique

Donnons le tableau des valeurs du périmètre et de la surface de l'étoile hérissée de paramètres b = 5, R = 1, r = 1/2, f = 4/3, pour n = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14

n	Périmètre	Surface
1	6.6407	1.4695
2	9.4079	1.8461
3	13.3227	2.1765
4	18.6804	2.4455
5	25.8888	2.6514
6	35.5107	2.8024
7	48.3149	2.9097
8	65.3396	2.9843
9	87.9794	3.0355
10	118.1001	3.0703
11	158.1939	3.0938
12	211.5866	3.1096
13	282.7141	3.1202
14	377.4920	3.1273

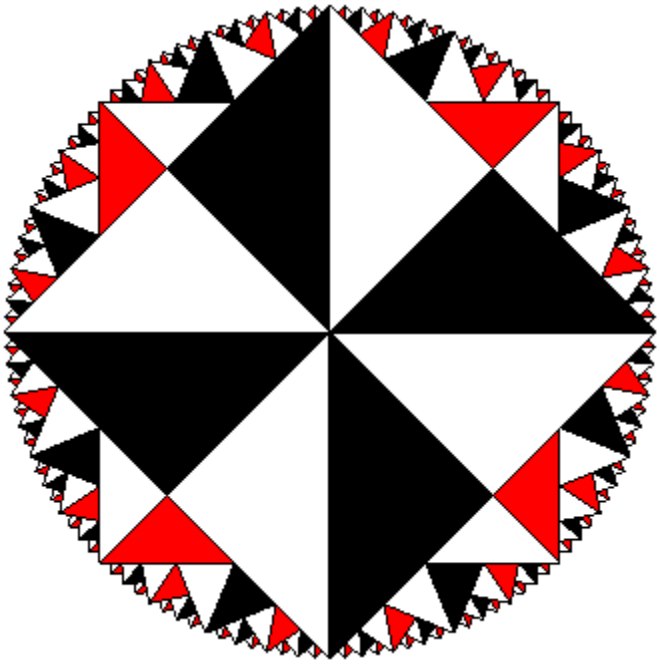
On déduit de ces données les 2 courbes de variation suivantes :



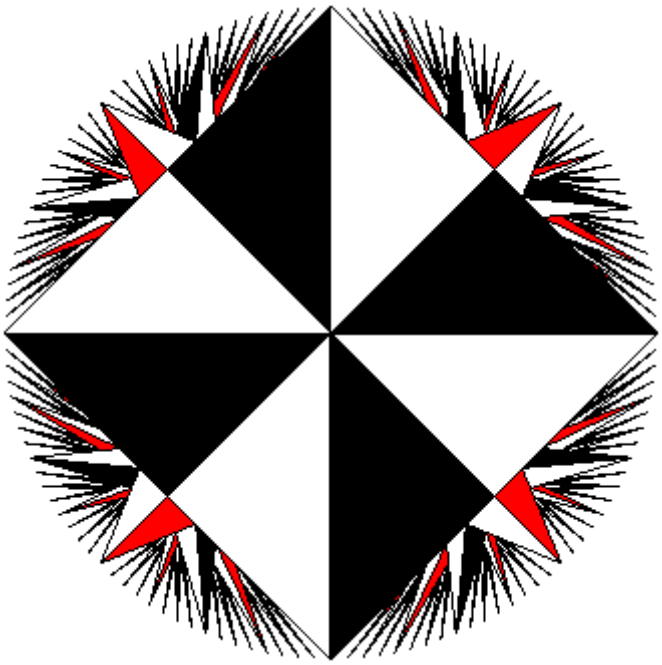


Mathématique & Informatique

L'étoile ordinaire $n = 6, b = 4, R = \sqrt{2}, r = 1, f = 1$

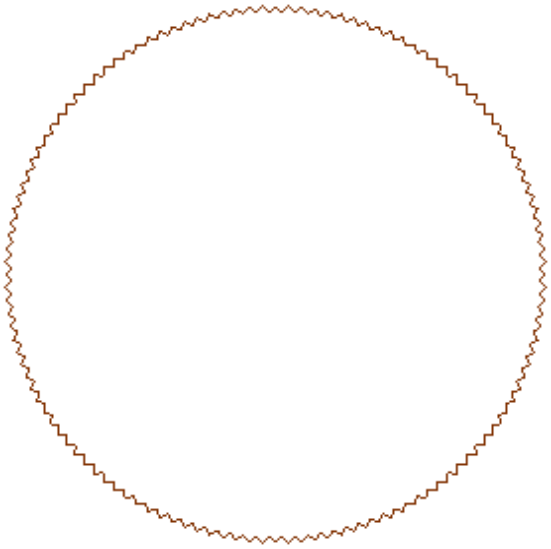


L'étoile hérissée $n = 6, b = 4, R = \sqrt{2}, r = 1, f = 2$

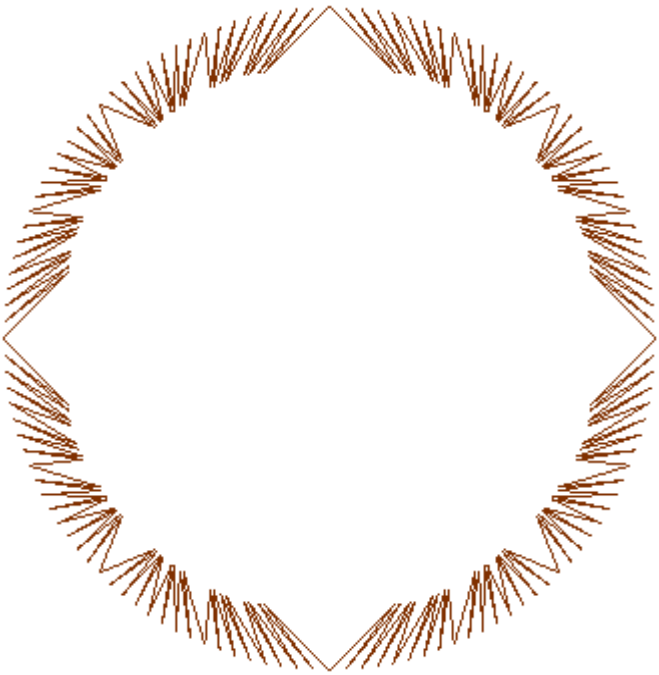


Mathématique & Informatique

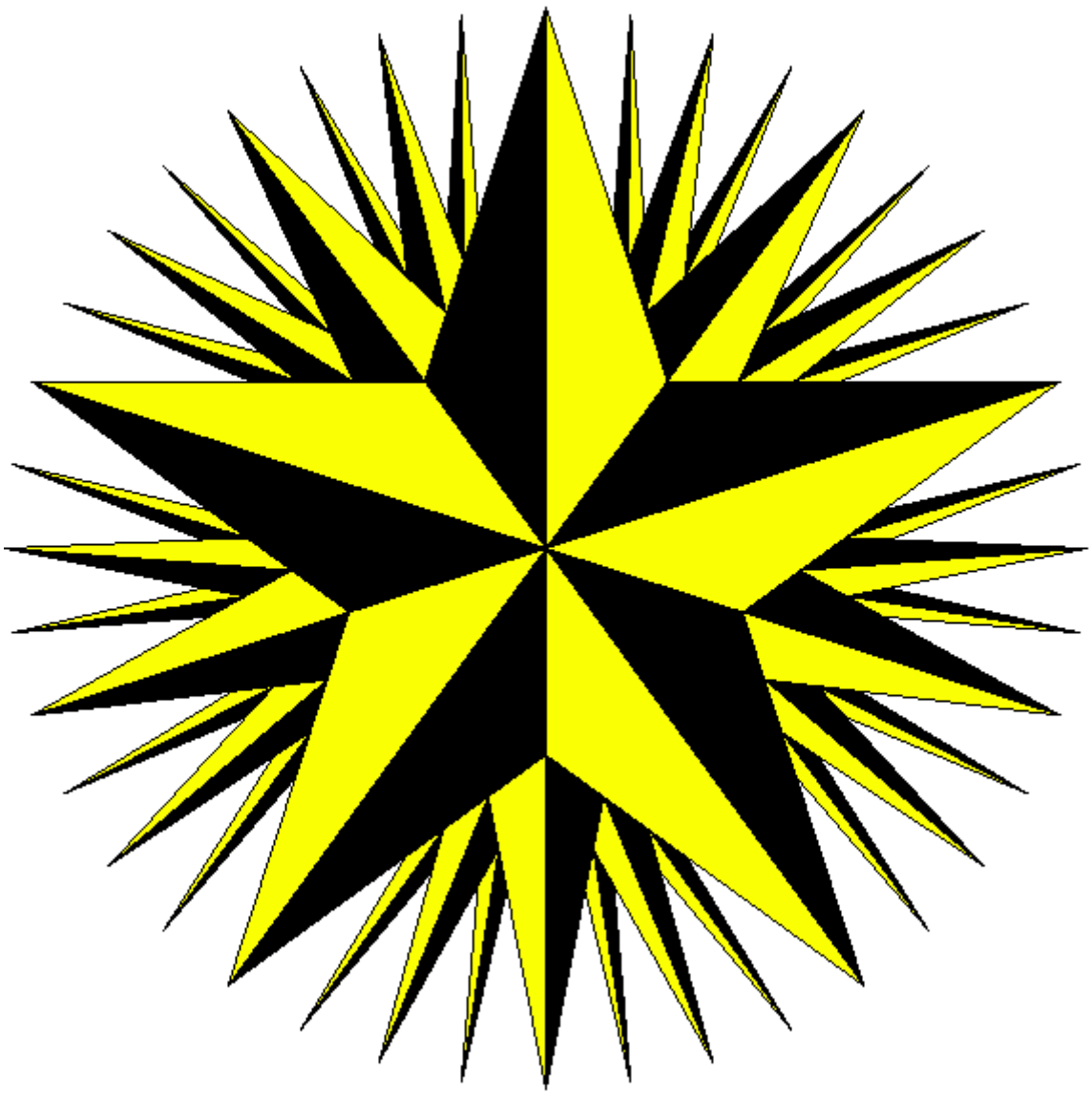
L'enveloppe ordinaire $n = 6, b = 4, R = \sqrt{2}, r = 1, f = 1$



L'enveloppe hérissée $n = 6, b = 4, R = \sqrt{2}, r = 1, f = 2$

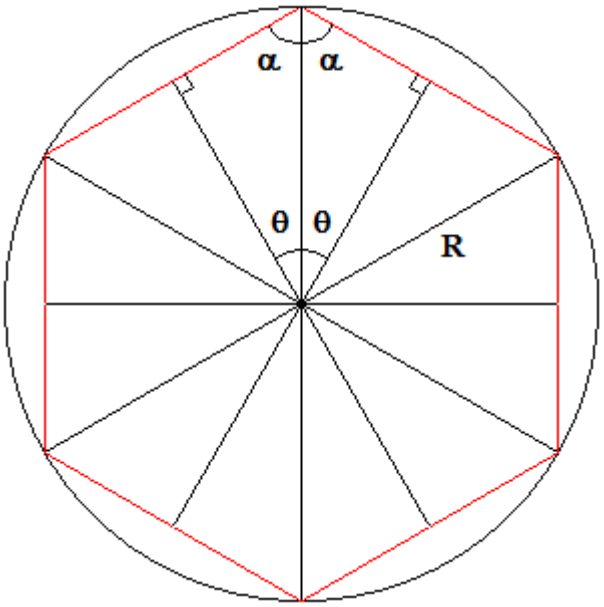


L'étoile hérissée $n = 4, b = 5, R = \varphi^2, r = 1, f = \varphi$



Propriété des étoiles hérissées

Il s'agit de démontrer qu'il existe des courbes fermées de longueur infinie qui contiennent une aire égale à π.
Construisons une étoile hérissée différente des précédentes, mais plus simple, afin de prouver cette proposition.
Partons d'un polygone régulier à b côtés inscrit dans un cercle de rayon R.
Par exemple, pour b = 6 :



Le polygone est constitué de 2b triangles rectangles. La somme des angles dans l'un de ces triangles est :

$$\pi = \theta + \alpha + \frac{\pi}{2}$$

Comme : $\theta = \pi/b$, on déduit :

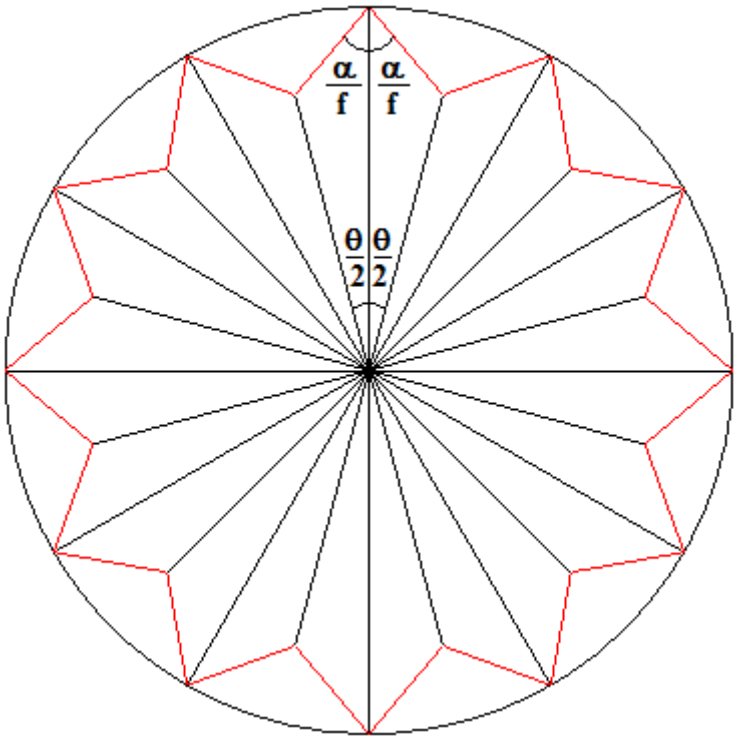
$$\alpha = \left(\frac{b-2}{2b}\right)\pi$$

On considère que l'étoile d'ordre 1 est égale à ce polygone.

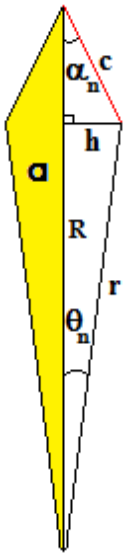
NB

On peut choisir un angle α non nul inférieur à $[(b-2)/(2b)]\pi$; l'étoile d'ordre 1 est alors un polygone concave.
La suite des calculs demeure inchangée.

A l'ordre 2, on divise l'angle au centre θ par 2 et l'angle au sommet α par un facteur de rétrécissement f > 1.
On obtient une étoile à 2b branches :



Répétons l'opération jusqu'à l'ordre n.
On aboutit à une étoile à 2^{n-1}b branches qui sont des quadrilatères de la forme :



$$\alpha_n = \frac{\alpha}{f^{n-1}}$$

$$\theta_n = \frac{\pi}{2^{n-1} b}$$

L'étoile d'ordre n possède un périmètre P constitué de 2^nb côtés c (trait en rouge sur le schéma) et une surface S constituée de 2^nb aires *a* (zone en jaune sur le schéma).

Les relations trigonométriques dans le triangle d'ordre n donnent :

$$c = \frac{\sin\theta_n}{\sin(\theta_n + \alpha_n)} R$$

$$r = \frac{\sin\alpha_n}{\sin(\theta_n + \alpha_n)} R$$

$$h = c \sin\alpha_n$$

$$a = \frac{1}{2} R h$$

Il s'ensuit les expressions du périmètre P et de la surface S de l'étoile d'ordre n :

$$P = \frac{\sin\dfrac{\pi}{2^{n-1} b}}{\sin\left(\dfrac{\pi}{2^{n-1} b} + \dfrac{\alpha}{f^{n-1}}\right)} 2^n b R$$

$$S = \frac{\sin\dfrac{\pi}{2^{n-1} b} \sin\dfrac{\alpha}{f^{n-1}}}{\sin\left(\dfrac{\pi}{2^{n-1} b} + \dfrac{\alpha}{f^{n-1}}\right)} 2^{n-1} b R^2$$

Calculons la limite de P et de S quand n $\to \infty$, sachant que sin x ~ x quand x $\to 0$
D'une part :

$$P \sim \frac{\pi}{2^{n-1} b} \frac{2^{n-1} b f^{n-1}}{\pi f^{n-1} + \alpha b 2^{n-1}} 2^n b R = \frac{2\pi b R}{\dfrac{\pi}{2^{n-1}} + \dfrac{\alpha b}{f^{n-1}}}$$

Si f > 1, le dénominateur tend vers 0 quand n $\to \infty$ alors que le numérateur reste constant.
Donc, P $\to \infty$ quand n $\to \infty$ avec f > 1
D'autre part :

$$S \sim \frac{\pi}{2^{n-1} b} \frac{\alpha}{f^{n-1}} \frac{2^{n-1} b f^{n-1}}{\pi f^{n-1} + \alpha b 2^{n-1}} 2^{n-1} b R^2 = \frac{\pi R^2}{1 + \dfrac{\pi}{\alpha b}\left(\dfrac{f}{2}\right)^{n-1}}$$

Si f < 2, le dénominateur tend vers 1 quand n $\to \infty$ alors que le numérateur reste constant.
Donc, S $\to \pi R^2$ quand n $\to \infty$ avec f < 2

Etudions la variation du périmètre et de la surface de l'étoile hérissée simplifiée en fonction du facteur f de rétrécissement des pointes.

- Cas f = 1

$$P \sim \frac{\pi}{2^{n-1} b} \frac{2^n b R}{\sin\alpha} = \frac{2\pi R}{\sin\alpha}$$

$$S \sim \frac{\pi}{2^{n-1} b} \frac{\sin\alpha}{\sin\alpha} 2^{n-1} b R^2 = \pi R^2$$

- Cas 1 < f < 2

$$P \sim \infty$$

$$S \sim \pi R^2$$

- Cas f = 2

$$P \sim \infty$$

$$S \sim \frac{\pi R^2}{1 + \frac{\pi}{\alpha b}} = \left(\frac{\alpha b}{\alpha b + \pi} \right) \pi R^2$$

- Cas f > 2

$$P \sim \infty$$

$$S \sim 0$$

Regroupons dans un tableau les valeurs du périmètre et de la surface de l'étoile hérissée quand $n \to \infty$

	f = 1	1 < f < 2	f = 2	f > 2
Périmètre	$\frac{2\pi R}{\sin\alpha}$	∞	∞	∞
Surface	πR^2	πR^2	$\left(\frac{\alpha b}{\alpha b + \pi} \right) \pi R^2$	0

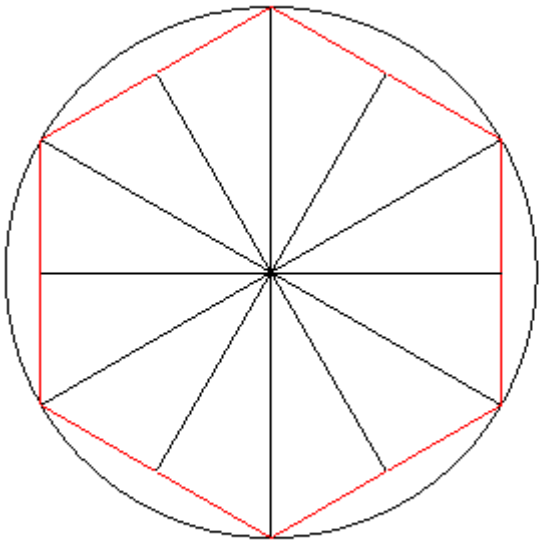
Programme de dessin de l'étoile hérissée simplifiée d'ordre n

Exemple des 8 premières étoiles ($1 \leq n \leq 8$) obtenues à partir d'un hexagone (b = 6) inscrit dans un cercle de rayon unité (R = 1) avec un facteur de rétrécissement égal à 3/2 (f = 3/2)

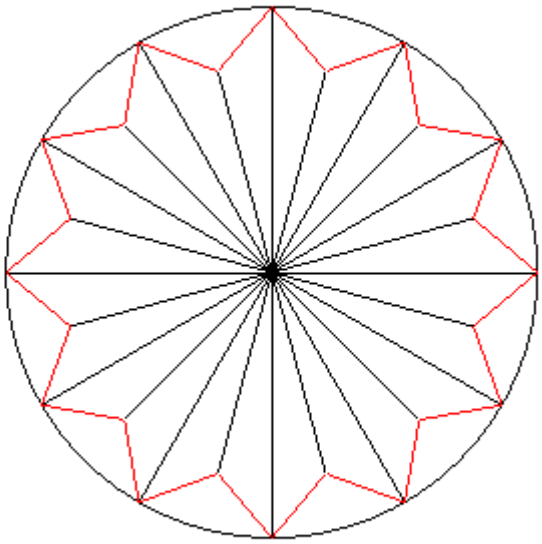
```
nmax = 8 ; b = 6 ; R = 1 ; f = 3/2 ;
alpha = ((b-2)/(2*b))*%pi ;
f1 = scf(1) ;
drawlater()
for n = 1:nmax
    subplot(ceil(nmax/2),2,n)
    xtitle('n = ' + string(n))
    plot2d(%inf,%inf,frameflag=3,rect=[-R,-R,+R,+R],axesflag=0)
    xarc(-R,+R,2*R,2*R,0,64*360)
    alphan = alpha/f^(n-1) ;
    thetan = %pi/(2^(n-1)*b) ;
    r = (sin(alphan)/sin(thetan + alphan))*R ;
    z0 = 0 ;
    z1 = r*exp(%i*(%pi/2 + thetan)) ;
    z2 = R*%i ;
    z3 = r*exp(%i*(%pi/2 - thetan)) ;
    B = [z0;z1;z2;z3] ;
    for q = 1:2^(n-1)*b
        Z = B*exp(%i*2*(q-1)*thetan) ;
        S1 = [Z(1),Z(1);Z(3),Z(4)] ;
        xsegs(real(S1),imag(S1),1) ;
        S2 = [Z(3),Z(3);Z(2),Z(4)] ;
        xsegs(real(S2),imag(S2),5) ;
    end
end
drawnow()
```

Mathématique & Informatique

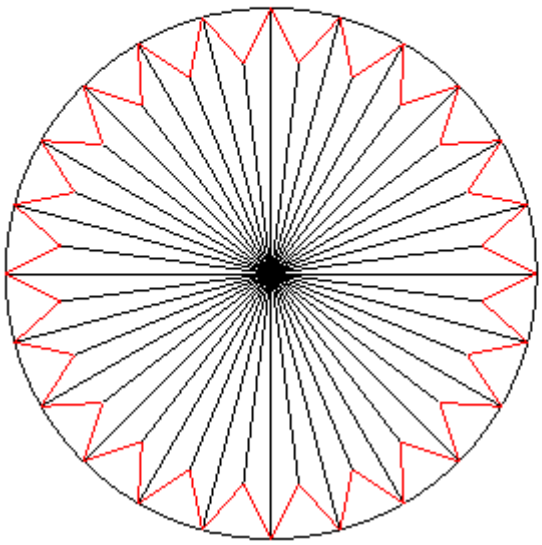
n = 1



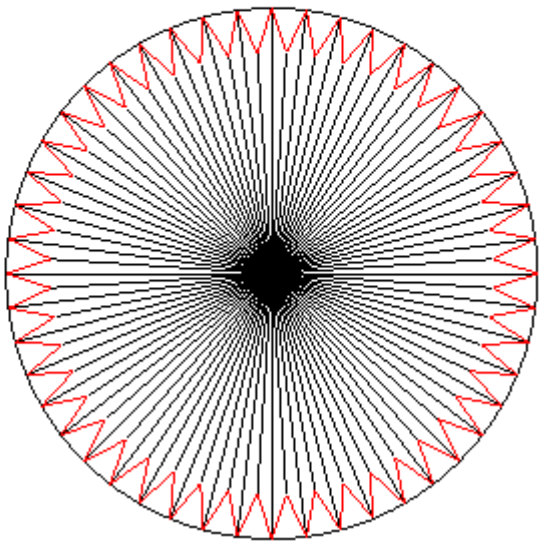
n = 2



n = 3

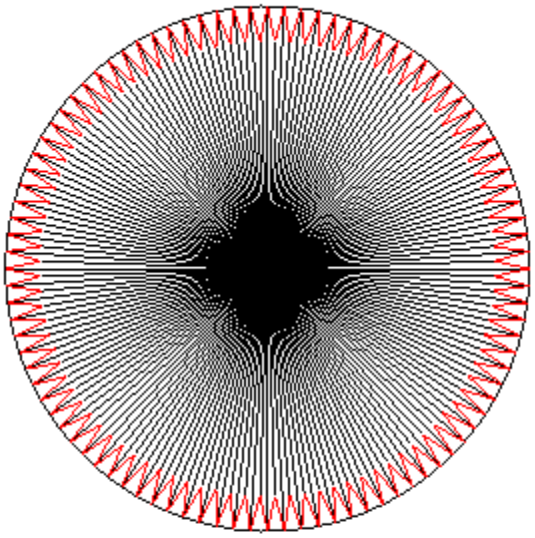


n = 4

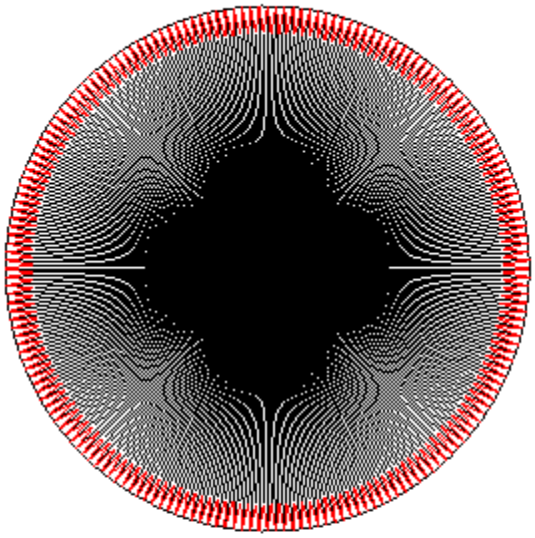


Mathématique & Informatique

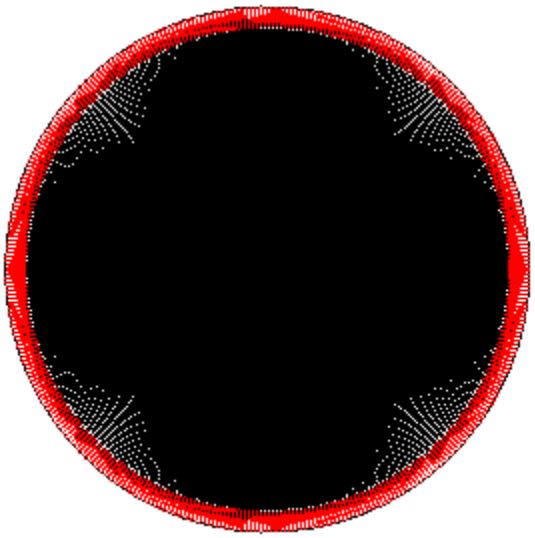
n = 5



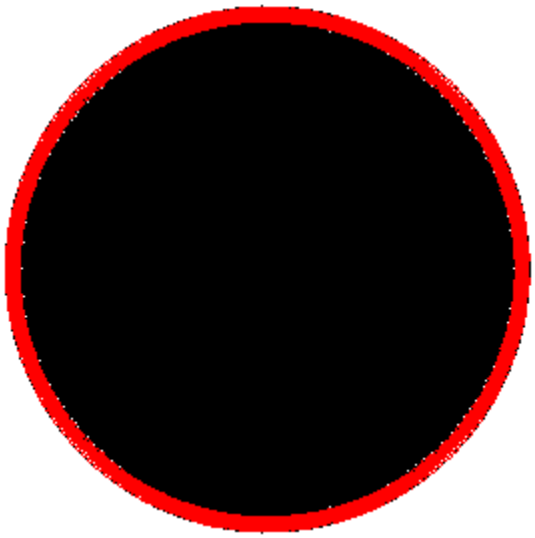
n = 6



n = 7



n = 8

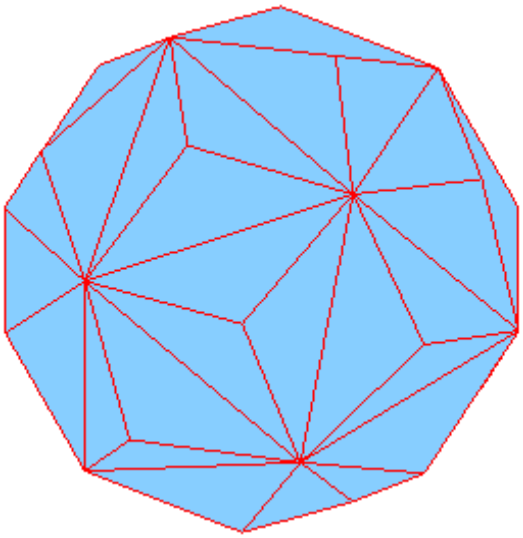


Conclusion



D'après le théorème isopérimétrique, la courbe fermée minimale qui contient une aire égale à π est le cercle de longueur 2π. Le cercle est un concept mathématique, courbe limite parfaite, qui n'existe pas dans la nature.
Un polygone régulier à n côtés inscrit dans un cercle de rayon R a un périmètre $P = 2nR\sin(\pi/n)$ et une surface intérieure $S = \frac{1}{2}nR^2\sin(2\pi/n)$. D'où, quand $n \to \infty$: $P \sim 2\pi R$ et $S \sim \pi R^2$, le polygone s'identifie au cercle de rayon R et sa surface intérieure au disque de rayon R.
L'étoile ordinaire ($f = 1$) d'ordre n a un périmètre fini et toujours supérieur à $2\pi R$ quand $n \to \infty$. Sa surface tend vers celle du disque de rayon R, soit πR^2.
L'étoile hérissée ($f > 1$) d'ordre n a un périmètre qui tend vers l'infini quand $n \to \infty$. Lorsque le facteur de rétrécissement est faible ($1 < f < 2$) sa surface tend également vers celle du disque de rayon R. Plus le facteur f est grand, plus la convergence du périmètre P vers l'infini est rapide et inversement plus la convergence de la surface S vers πR^2 est lente.
Pour R = 1, l'enveloppe de l'étoile hérissée est un exemple de courbe fermée de longueur infinie qui contient une aire finie égale à π.
Ce résultat s'applique à une surface plane quelconque d'aire A entourée d'une ligne brisée dont les pointes sont de plus en plus petites (côté $c \to 0$) et de plus en plus aiguës (angle $\alpha \to 0$) de sorte que le périmètre P tende vers l'infini et que la surface intérieure S tende vers A.
On pourrait étendre la méthode à l'espace à 3 dimensions en considérant un polyèdre convexe possédant ses sommets sur une sphère et dont les faces sont des petits triangles sur lesquels viennent se greffer des pointes en forme de tétraèdres. L'algorithme consiste à construire une suite de polyèdres concaves de volume croissant dont la hauteur des pointes et l'angle diédral à leur sommet diminuent à chaque étape. On obtient ainsi une étoile hérissée dont la surface devient infinie et dont le volume tend vers celui de la sphère circonscrite.

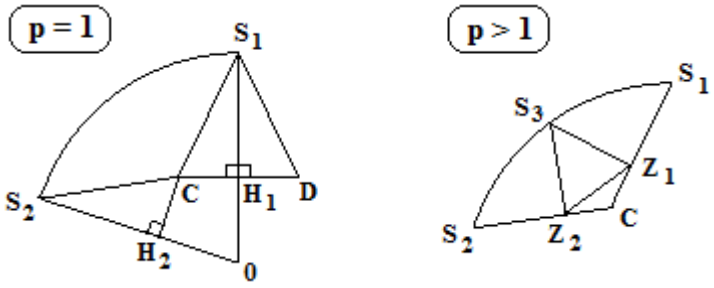
Etoile tétraédrique à 60 faces construite sur un icosaèdre



L'observation, la géométrie et le dessin sont souvent la source des idées. On oppose parfois les disciplines entre elles, la géométrie à l'algèbre ou à l'analyse par exemple, et elles sont étudiées séparément. Il est préférable de les unir et de montrer les liens qui existent entre elles, d'où le terme de mathématique (au singulier) pour présenter l'ensemble comme un tout. De la même façon, il est difficile de dissocier aujourd'hui les mathématiques de l'informatique, de l'algorithmique et du calcul numérique.

Etoile triangulaire

En définissant les éléments de l'étoile d'une manière différente, on obtient une mosaïque circulaire de triangles.



Reprenons l'algorithme de l'étoile hérissée en ajoutant 2 nouveaux points H_1 et H_2 sur les branches de niveau 1 :

$$H_1 = C\left(\cos\frac{\pi}{b}\right) e^{-i\frac{\pi}{b}} \qquad H_2 = C\left(\cos\frac{\pi}{b}\right) e^{+i\frac{\pi}{b}}$$

Ecrivons un programme qui dessine une étoile triangulaire ordinaire (f = 1) à 5 niveaux (n = 5) reposant sur un hexagone (b = 6) et composée de petits triangles équilatéraux (R/r = $\sqrt{3}$) dont seuls les contours seront représentés.
On utilise la fonction Scilab *xpoly(x,y,"lines",close)* pour tracer une ligne brisée ouverte (paramètre *close = 0*) ou fermée (paramètre *close = 1*).

```
// Paramètres de l'étoile
b = 6 ; n = 5 ; R = sqrt(3) ; r = 1 ; f = 1 ;
alpha = acot((R/r-cos(%pi/b))/sin(%pi/b)) ;

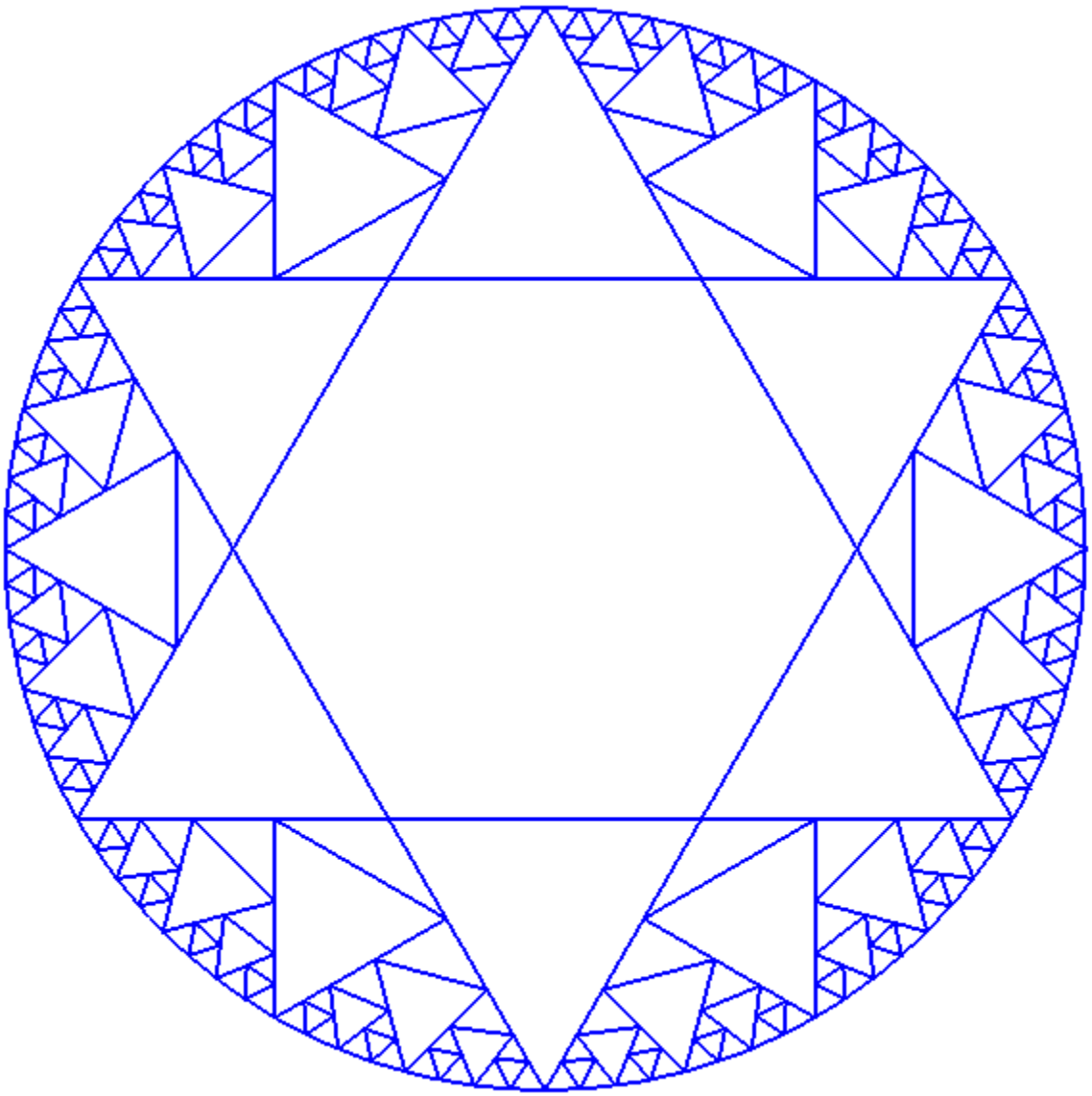
// Programme principal
f1 = scf(1) ;
plot2d(%inf,%inf,frameflag = 3,rect = [-R,-R,R,R],axesflag = 0)
xset("color",2) ; xset("thickness",2)
drawlater()
for k = 1:b
    C = r*exp(%i*(%pi/2 + (2*k-1)*%pi/b)) ;
    S1 = R*exp(%i*(%pi/2 + (2*k-2)*%pi/b)) ;
    S2 = R*exp(%i*(%pi/2 + (2*k)*%pi/b)) ;
    H1 = C*cos(%pi/b)*exp(-%i*%pi/b) ;
    H2 = C*cos(%pi/b)*exp(+%i*%pi/b) ;
    T = [S1;C;H1] ;
    xpoly(real(T),imag(T),"lines",0)
    T = [S2;C;H2] ;
    xpoly(real(T),imag(T),"lines",0)
    Branch(C,S1,S2,n-1)
end
xarc(-R,+R,2*R,2*R,0,64*360)
drawnow()
```

```
// Argument d'un nombre complexe
function a = arg(z)
    a = atan(imag(z),real(z))
endfunction

// Branches de l'étoile
function Branch(C,S1,S2,p)
    if p > 0 then
        alphap = alpha/f^(n-p)
        S3 = R*exp(%i*arg(S1+S2))
        beta1 = arg((S3-C)/(S1-C))
        k1 = sin(alphap)/sin(alphap+beta1)
        Z1 = C + (S3-C)*k1*exp(-%i*beta1)
        beta2 = arg((S2-C)/(S3-C))
        k2 = sin(alphap)/sin(alphap+beta2)
        Z2 = C + (S3-C)*k2*exp(+%i*beta2)
        T = [S3;Z2;Z1]
        xpoly(real(T),imag(T),"lines",1)
        Branch(Z1,S1,S3,p-1)
        Branch(Z2,S3,S2,p-1)
    end
endfunction
```

Mathématique & Informatique

L'étoile triangulaire $b = 6, n = 5, R = \sqrt{3}, r = 1, f = 1$



Un autre exemple, avec un octogone (b = 8) comme figure de départ et 3 niveaux de branche (n = 3).
Chaque triangle de la figure est divisé en quatre petits triangles : les 3 triangles périphériques sont colorés et le triangle central ne l'est pas (voir triangle de **Sierpinski**). On impose en plus que toutes les extrémités des branches soient constituées de triangles équilatéraux, ce qui se traduit par la condition :

$$\cot an\frac{\pi}{6} = \frac{\frac{R}{r} - \cos\frac{\pi}{b}}{\sin\frac{\pi}{b}} \qquad \Rightarrow \qquad r = 1 \quad \text{et} \quad R = \sqrt{3}\sin\frac{\pi}{b} + \cos\frac{\pi}{b}$$

```
// Paramètres de l'étoile
b = 8 ; n = 3 ; r = 1 ; f = 1 ;
R = sqrt(3)*sin(%pi/b) + cos(%pi/b) ;
alpha = acot((R/r-cos(%pi/b))/sin(%pi/b)) ;

// Programme principal
f2 = scf(2) ;
plot2d(%inf, %inf,frameflag = 3,rect = [-R,-R,R,R],axesflag = 0)
drawlater()
for k = 1:b
    C = r*exp(%i*(%pi/2 + (2*k-1)*%pi/b)) ;
    D = r*exp(%i*(%pi/2 + (2*k-3)*%pi/b)) ;
    S1 = R*exp(%i*(%pi/2 + (2*k-2)*%pi/b)) ;
    S2 = R*exp(%i*(%pi/2 + (2*k)*%pi/b)) ;
    H1 = C*cos(%pi/b)*exp(-%i*%pi/b) ;
    H2 = C*cos(%pi/b)*exp(+%i*%pi/b) ;
    T = [0;H1;H2] ;
    Color(T,1)
    T = [S1;C;D] ;
    Color(T,1)
    Branch(C,S1,S2,n-1)
end
xarc(-R,+R,2*R,2*R,0,64*360)
drawnow()
```

```
// Argument d'un nombre complexe
function a = arg(z)
    a = atan(imag(z),real(z))
endfunction

// Coloration d'un triangle
function Color(T,c)
    T1 = [T(1);(T(1)+T(2))/2;(T(1)+T(3))/2]
    xfpolys(real(T1),imag(T1),c)
    T2 = [(T(1)+T(2))/2;T(2);(T(2)+T(3))/2]
    xfpolys(real(T2),imag(T2),c)
    T3 = [(T(1)+T(3))/2;(T(2)+T(3))/2;T(3)]
    xfpolys(real(T3),imag(T3),c)
endfunction

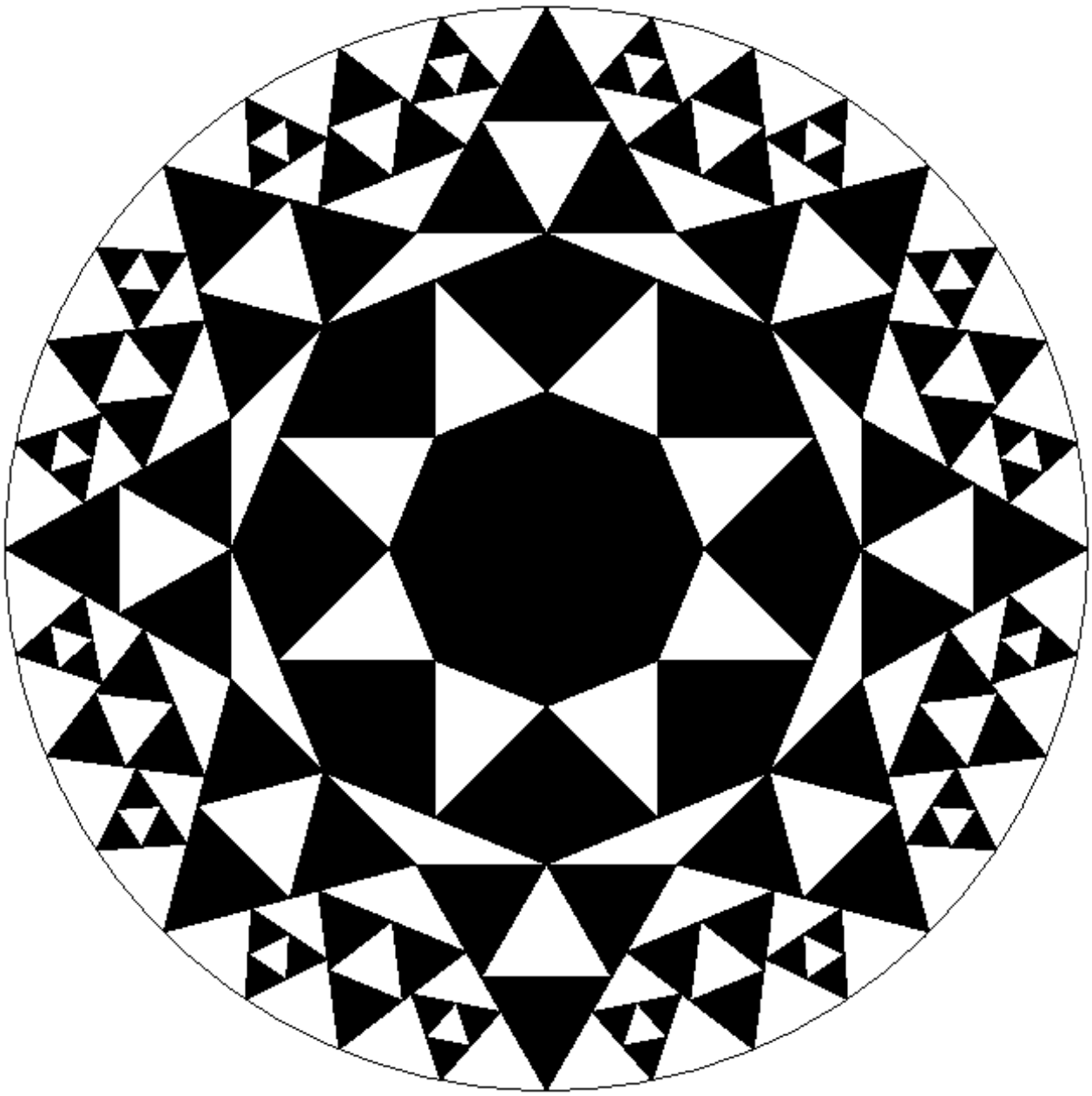
// Branches de l'étoile
function Branch(C,S1,S2,p)
    if p > 0 then
        alphap = alpha/f^(n-p)
        S3 = R*exp(%i*arg(S1+S2))
        beta1 = arg((S3-C)/(S1-C))
        k1 = sin(alphap)/sin(alphap+beta1)
        Z1 = C + (S3-C)*k1*exp(-%i*beta1)
        beta2 = arg((S2-C)/(S3-C))
        k2 = sin(alphap)/sin(alphap+beta2)
        Z2 = C + (S3-C)*k2*exp(+%i*beta2)
        T = [S3;Z2;Z1]
        Color(T,1)
        Branch(Z1,S1,S3,p-1)
        Branch(Z2,S3,S2,p-1)
    end
endfunction
```

Idem pour l'heptagone, l'hexagone, le pentagone, le carré et le triangle avec les paramètres respectifs :

```
b = 7 ; n = 3 ;
b = 6 ; n = 4 ;
b = 5 ; n = 4 ;
b = 4 ; n = 5 ;
b = 3 ; n = 5 ;
```

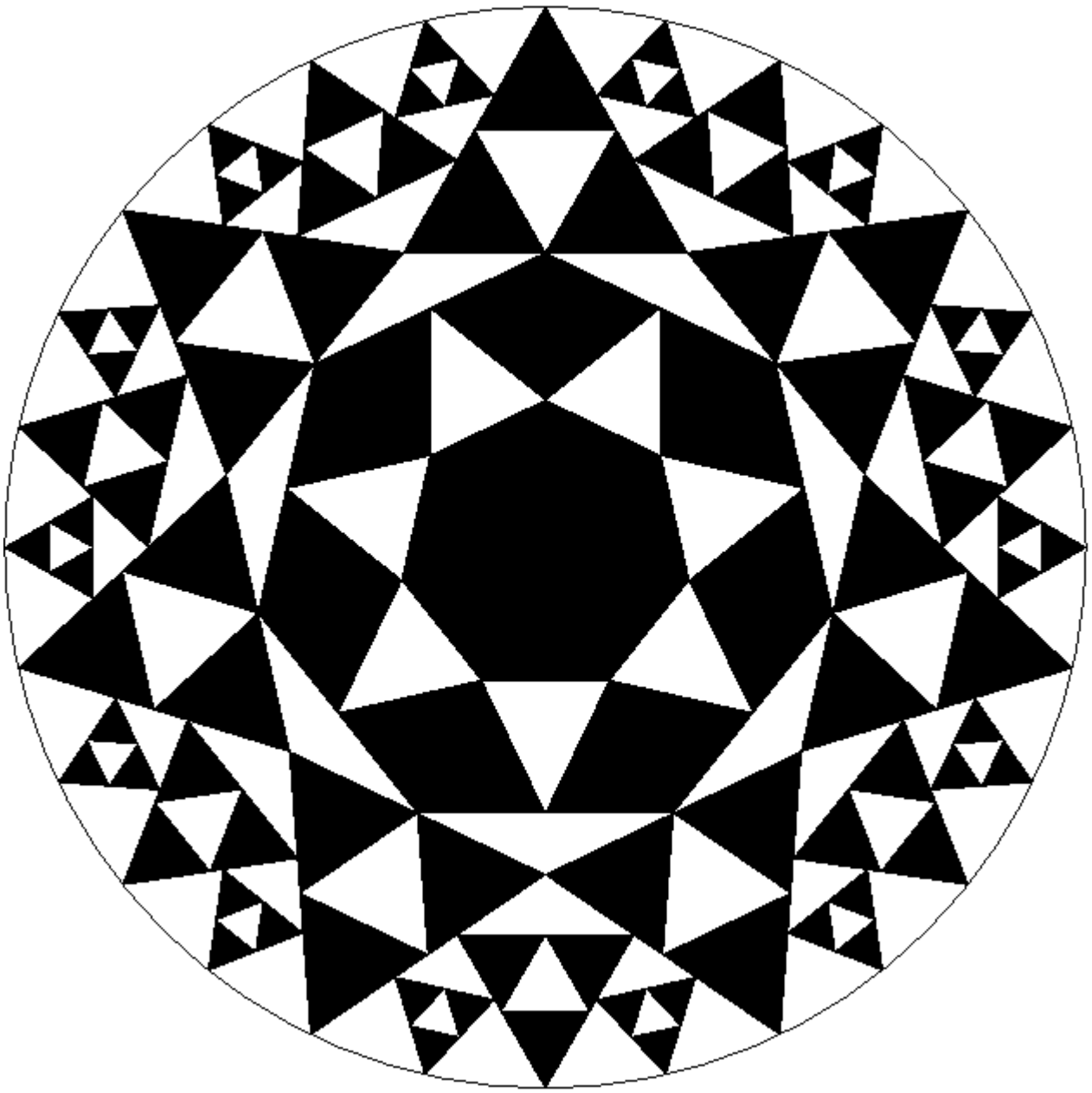
Mathématique & Informatique

L'étoile octogonale $b = 8, n = 3, R = 1.587, r = 1, f = 1$



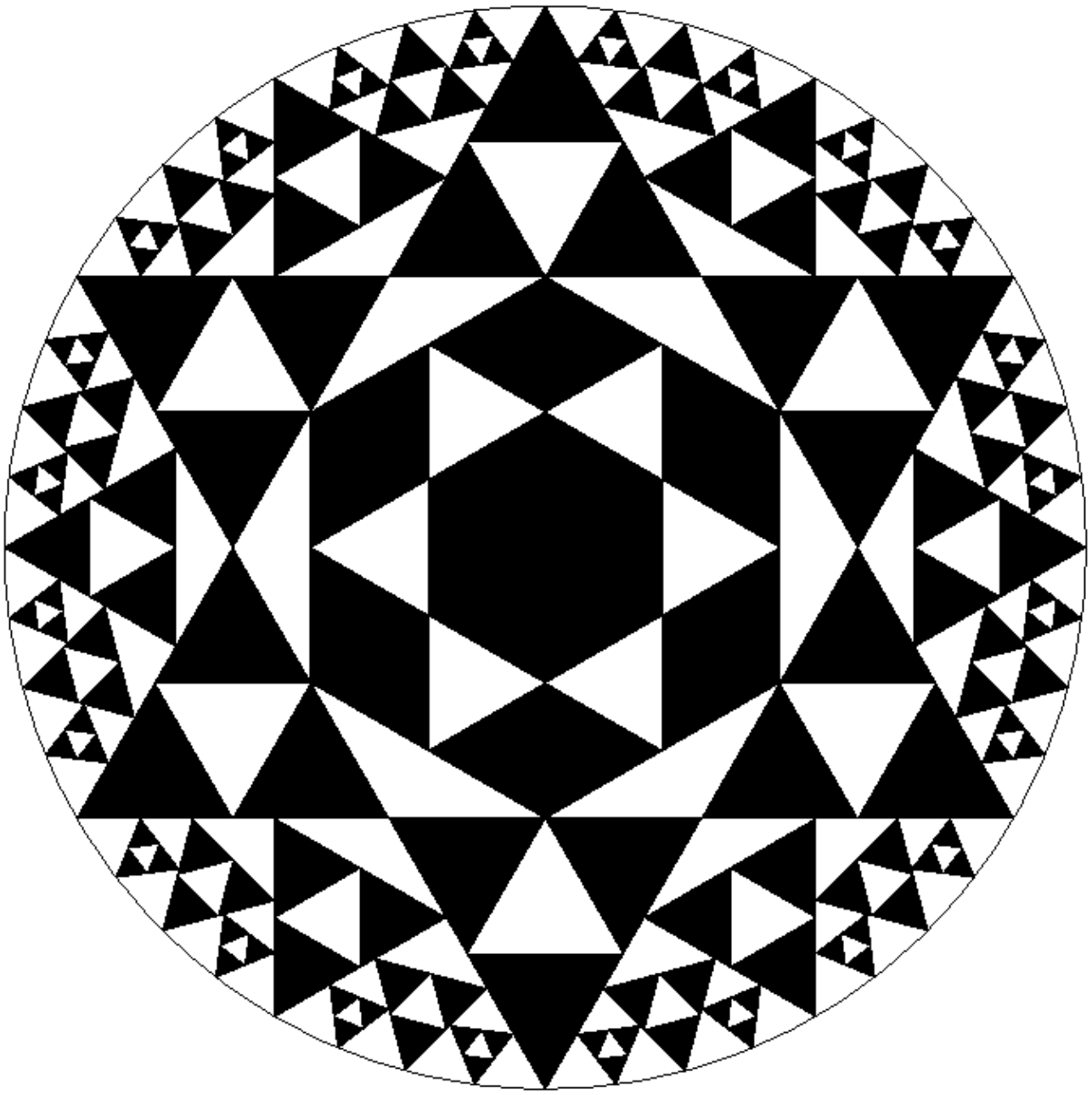
Mathématique & Informatique

L'étoile heptagonale $b = 7, n = 3, R = 1.652, r = 1, f = 1$



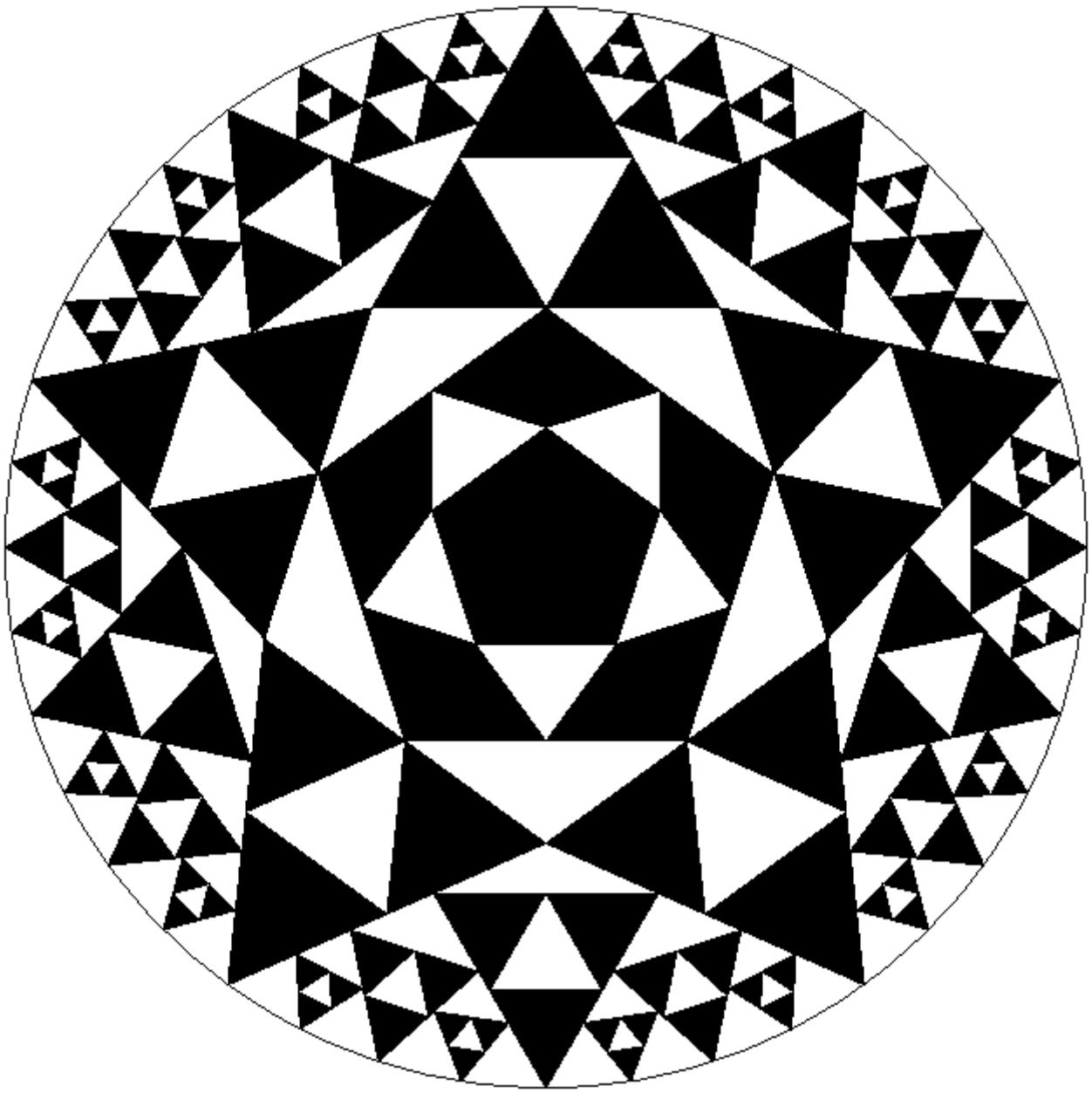
Mathématique & Informatique

L'étoile hexagonale $b = 6, n = 4, R = 1.732, r = 1, f = 1$



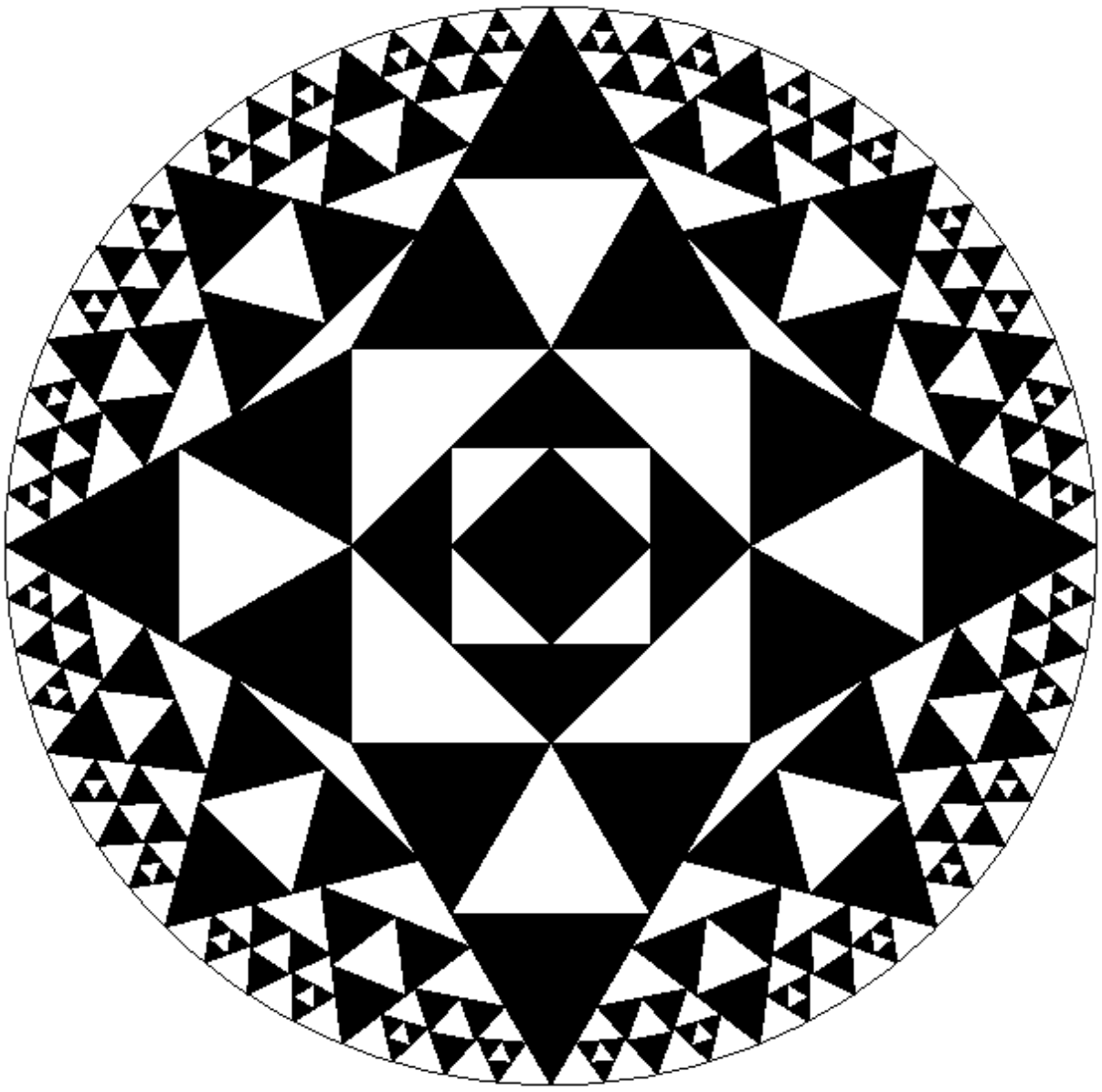
Mathématique & Informatique

L'étoile pentagonale $b = 5, n = 4, R = 1.827, r = 1, f = 1$



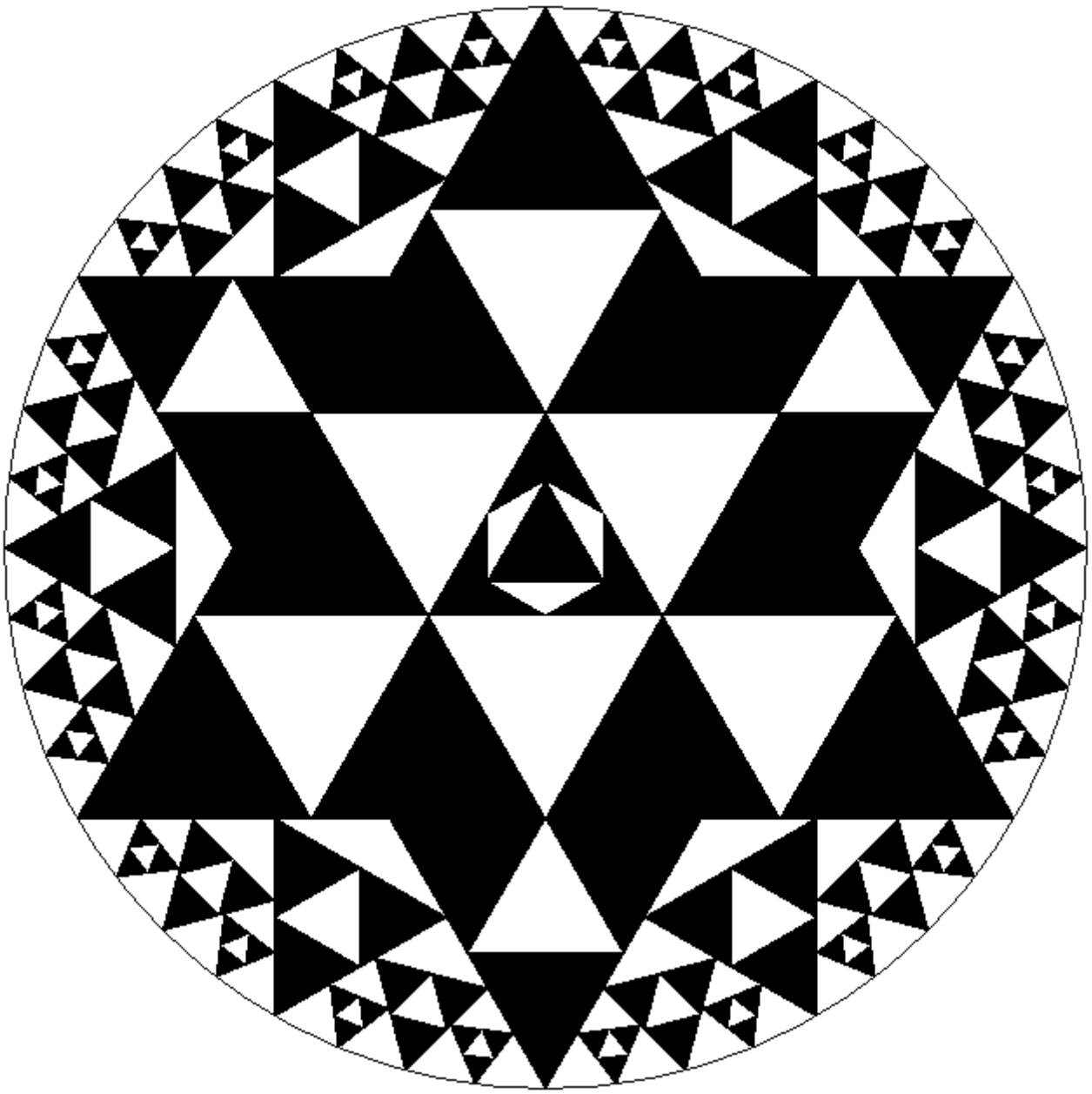
Mathématique & Informatique

L'étoile carrée $b = 4, n = 5, R = 1.932, r = 1, f = 1$



Mathématique & Informatique

L'étoile triangulaire $b = 3, n = 5, R = 2, r = 1, f = 1$



Dans l'exemple ci-dessous, associons l'algorithme d'une étoile dodécagonale (b = 12) à 5 niveaux de branche (n = 5) avec une division récurrente des triangles qui la compose :

```
// Paramètres de l'étoile
b = 12 ; n = 5 ; r = 1 ; f = 1 ;
R = sqrt(3)*sin(%pi/b) + cos(%pi/b) ;
alpha = acot((R/r-cos(%pi/b))/sin(%pi/b)) ;

// Programme principal
f3 = scf(3) ;
plot2d(%inf,%inf,frameflag=3,rect=[-R,-R,R,R],axesflag=0)
black = color(0,0,0) ; blue = color(0,0,208) ;
drawlater()
for k = 1:b
    C = r*exp(%i*(%pi/2 + (2*k-1)*%pi/b)) ;
    D = r*exp(%i*(%pi/2 + (2*k-3)*%pi/b)) ;
    S1 = R*exp(%i*(%pi/2 + (2*k-2)*%pi/b)) ;
    S2 = R*exp(%i*(%pi/2 + (2*k)*%pi/b)) ;
    H1 = C*cos(%pi/b)*exp(-%i*%pi/b) ;
    H2 = C*cos(%pi/b)*exp(+%i*%pi/b) ;
    T = [0;H1;H2] ;
    Triangle(T,n-1,blue)
    T = [S1;C;D] ;
    Triangle(T,n-1,black)
    Branch(C,S1,S2,n-1)
end
drawnow()
```

```
// Argument d'un nombre complexe
function a = arg(z)
    a = atan(imag(z),real(z))
endfunction

// Division d'un triangle
function Triangle(T,q,c)
    if q > 0 then
        Triangle([T(1);(T(1)+T(2))/2;(T(1)+T(3))/2],q-1,c)
        Triangle([(T(1)+T(2))/2;T(2);(T(2)+T(3))/2],q-1,c)
        Triangle([(T(1)+T(3))/2;(T(2)+T(3))/2;T(3)],q-1,c)
    else
        xfpolys(real(T),imag(T),c)
    end
endfunction

// Branches de l'étoile
function Branch(C,S1,S2,p)
    if p > 0 then
        alphap = alpha/f^(n-p)
        S3 = R*exp(%i*arg(S1+S2))
        beta1 = arg((S3-C)/(S1-C))
        k1 = sin(alphap)/sin(alphap+beta1)
        Z1 = C + (S3-C)*k1*exp(-%i*beta1)
        beta2 = arg((S2-C)/(S3-C))
        k2 = sin(alphap)/sin(alphap+beta2)
        Z2 = C + (S3-C)*k2*exp(+%i*beta2)
        T = [S3;Z2;Z1]
        Triangle(T,p,black)
        Branch(Z1,S1,S3,p-1)
        Branch(Z2,S3,S2,p-1)
    end
endfunction
```

Cette fractale est un objet mathématique complexe décrit par un algorithme et des formules analytiques.

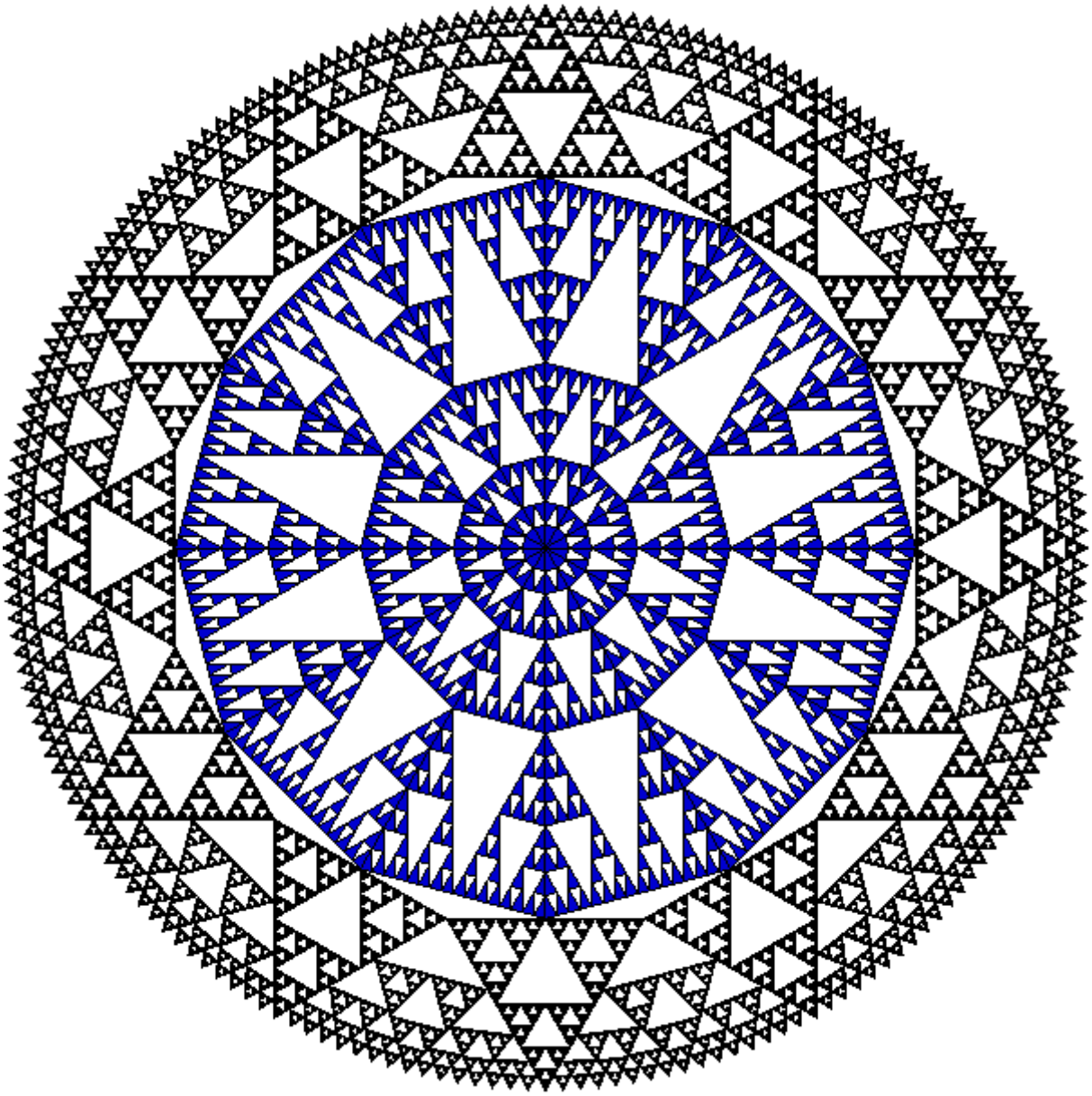
Remarquons que :

$$R = \sqrt{3}\sin\frac{\pi}{12} + \cos\frac{\pi}{12} = \sqrt{3}\left(\frac{\sqrt{6}-\sqrt{2}}{4}\right) + \left(\frac{\sqrt{6}+\sqrt{2}}{4}\right) = \frac{\sqrt{18}-\sqrt{6}+\sqrt{6}+\sqrt{2}}{4} = \frac{3\sqrt{2}+\sqrt{2}}{4} = \sqrt{2}$$

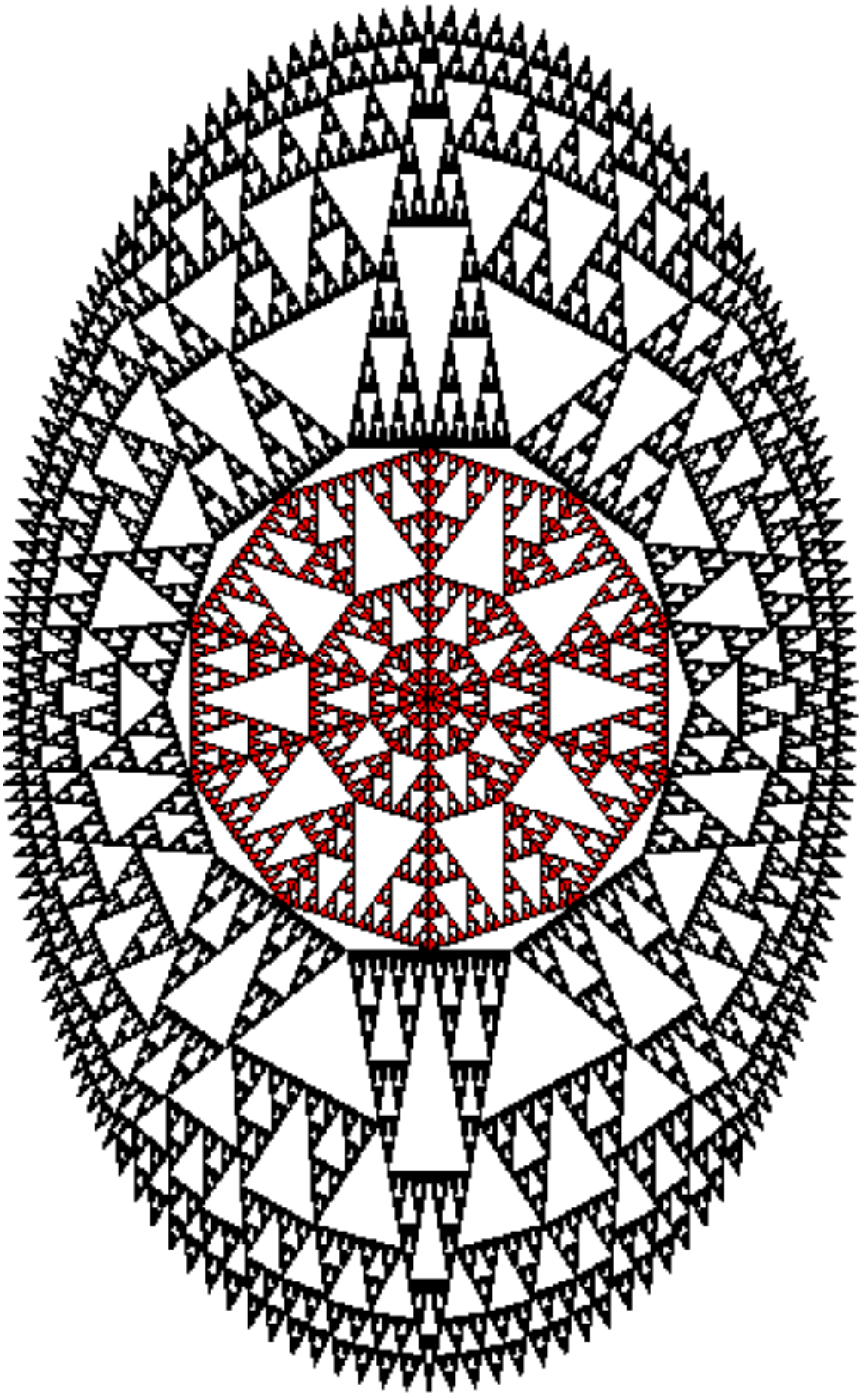
L'étoile triangulaire b = 12, n = 5, R = $\sqrt{2}$, r = 1, f = 1 est représentée page suivante.

Le dernier exemple est celui d'une étoile triangulaire elliptique, de paramètres principaux b = 10 et n = 5, construite selon un procédé analogue.

Mathématique & Informatique



Mathématique & Informatique



```
//-----------------------------------------------
//                  Etoile
//-----------------------------------------------

funcprot(0) ; lines(0)

// Argument d'un nombre complexe
function a = arg(z)
    a = atan(imag(z),real(z))
endfunction

// Division d'un triangle
function Triangle(T, q, k)
    select k
    case 1 then
        if q > 0 then
            z1 = T(1)
            z2 = T(2)
            z3 = T(3)
            z4 = (z1 + z2)/2
            z5 = (z2 + z3)/2
            z6 = (z1 + z3)/2
            z7 = (z1 + z2 + z3)/3
            T1 = [z4;z6;z7]
            T2 = [z2;z3;z7]
            T3 = [z4;z2;z7]
            T4 = [z6;z3;z7]
            // Dessin des triangles
            xfpolys(real(T1),imag(T1),[1;1;nc])
            xfpolys(real(T2),imag(T2),[1;1;nc])
            xfpolys(real(T3),imag(T3),c1)
            xfpolys(real(T4),imag(T4),c1)
            T = [z1;z4;z6]
            Triangle(T,q-1,1)
        else
            // Dessin du triangle
            xfpolys(real(T),imag(T),[1;1;nc])
        end
    case 2 then
        if q > 1 then
            z1 = T(1)
            z2 = T(2)
            z3 = T(3)
            z4 = (z1 + z2)/2
            z5 = (z2 + z3)/2
            z6 = (z1 + z3)/2
            T1 = [z1;z4;z6]
            T2 = [z4;z2;z5]
            T3 = [z6;z5;z3]
            T4 = [z5;z6;z4]
            Triangle(T1,q-1,2)
            Triangle(T2,q-1,2)
            Triangle(T3,q-1,2)
            Triangle(T4,q-1,2)
        else
```

```
            // Affixes des sommets du triangle
            z1 = T(1)
            z2 = T(2)
            z3 = T(3)
            // Côtés du triangle
            a = abs(z2 - z3)
            b = abs(z1 - z3)
            c = abs(z1 - z2)
            // Rayon du cercle circonscrit
            p = (a + b + c)/2
            R = a*b*c/(4*sqrt(p*(p - a)*(p - b)*(p - c)))
            // Affixe du centre du cercle circonscrit
            u = a^2*(b^2 + c^2 - a^2)
            v = b^2*(a^2 + c^2 - b^2)
            w = c^2*(a^2 + b^2 - c^2)
            z0 = (u*z1 + v*z2 + w*z3)/(u + v +w)
            // Dessin du triangle
            xfpolys(real(T),imag(T),c2)
            // Dessin du deltoïde
            m = 50
            t = linspace(0,2*%pi,m)
            zd = (R/3)*(2*exp(%i*t) + exp(-2*%i*t))
            theta = arg(z3 - z2) - %pi/6
            z = z0 + zd*exp(%i*theta)
            for i = 1:m-1
                Z = [z0;z(i);z(i+1)]
                xfpolys(real(Z),imag(Z),[2*nc;nc+1;nc+1])
            end
        end
    end
endfunction

// Branches de l'étoile
function Branch(C, S1, S2, p)
    if p > 0 then
        alphap = alpha/f^(n-p)
        S3 = R*exp(%i*arg(S1 + S2))
        beta1 = arg((S3 - C)/(S1 - C))
        k1 = sin(alphap)/sin(alphap + beta1)
        Z1 = C + (S3 - C)*k1*exp(-%i*beta1)
        beta2 = arg((S2 - C)/(S3 - C))
        k2 = sin(alphap)/sin(alphap + beta2)
        Z2 = C + (S3 - C)*k2*exp(+%i*beta2)
        T = [S3;Z2;Z1]
        Triangle(T,p,2)
        Branch(Z1,S1,S3,p-1)
        Branch(Z2,S3,S2,p-1)
    end
endfunction

// Paramètres de l'étoile
b = 16 ; n = 3 ; r = 1 ; f = 1 ;
R = sqrt(3)*sin(%pi/b) + cos(%pi/b) ;
alpha = acot((R/r - cos(%pi/b))/sin(%pi/b)) ;
```

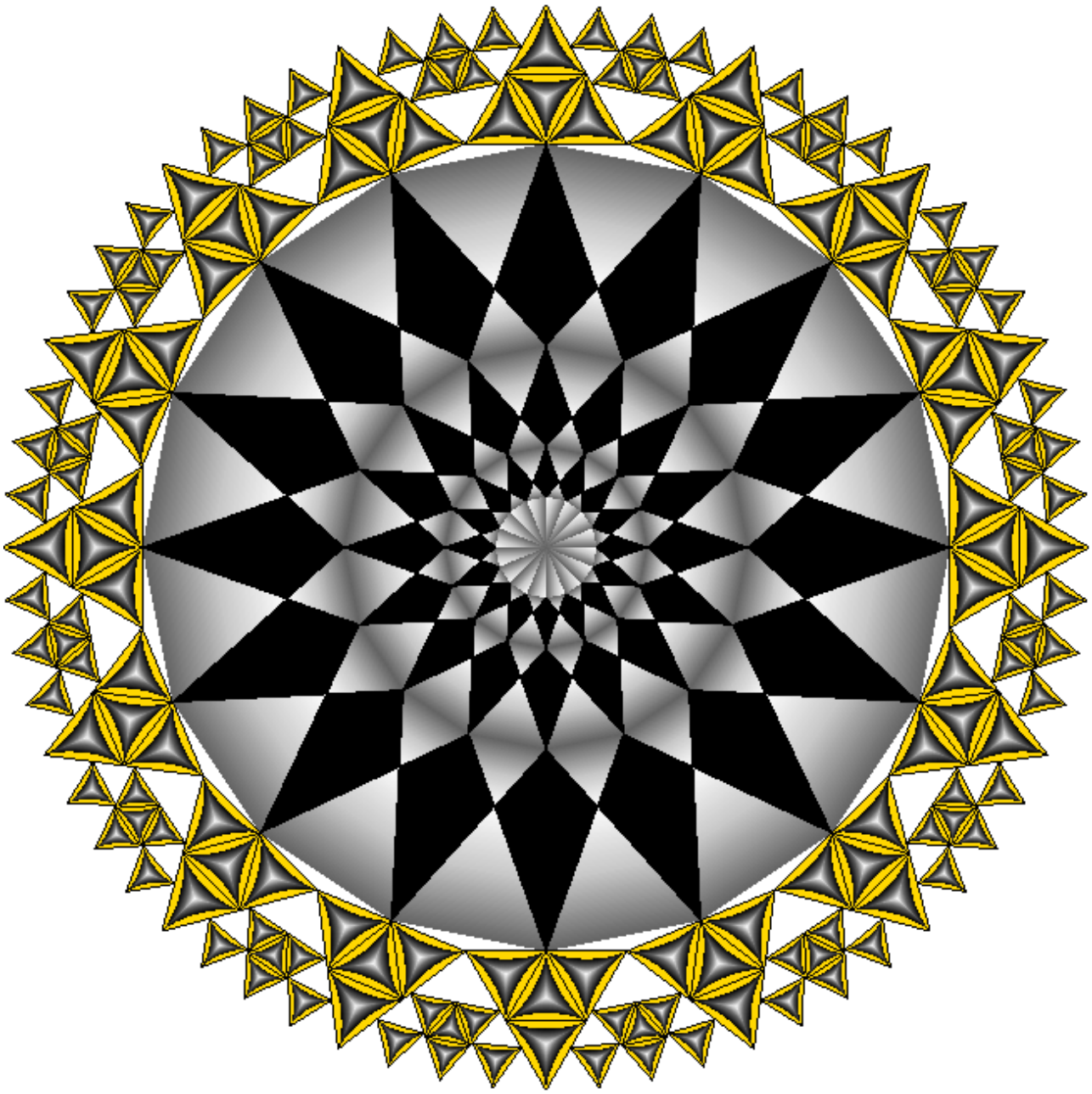
```
// Carte de couleurs
function cmap = colormap(C, n)
    function ccode = colorcode(c1, c2, k)
        R = linspace(c1(1),c2(1),k)'/255
        G = linspace(c1(2),c2(2),k)'/255
        B = linspace(c1(3),c2(3),k)'/255
        ccode = [R(1:k-1),G(1:k-1),B(1:k-1)]
    endfunction
    cmap = []
    m = length(C)
    if m < 2 | n < m then, return, end
    q = floor((n - m)/(m - 1))
    r = (n - m) - q*(m - 1)
    for i = 1:m-1
        if r <> 0 then
            cmap = [cmap ; colorcode(C(i),C(i+1),q+3)]
            r = r - 1
        else
            cmap = [cmap ; colorcode(C(i),C(i+1),q+2)]
        end
    end
    cmap = [cmap ; C(m)/255]
endfunction

// Fenêtre graphique
f27 = scf(27) ;
nc = 64 ;
cmap1 = colormap(list([100,100,100],[200,200,200],[255,255,255]),nc) ;
cmap2 = colormap(list([0,0,0],[100,100,100],[255,255,255]),nc) ;
f27.color_map = [cmap1 ; cmap2] ;

// Couleurs particulières
Noir = color(0,0,0) ;
Blanc = color(255,255,255) ;
Rouge = color(255,0,0) ;
Vert = color(0,144,0) ;
Bleu = color(0,0,255) ;
Cyan = color(0,255,255) ;
Jaune = color(255,255,0) ;
Magenta = color(208,0,208) ;
Violet = color(127,0,255) ;
Bronze = color(97,78,26) ;
Or = color(255,215,0) ;
Gris = color(210,210,210) ;
Orange = color(255,127,0) ;
c1 = Noir ;
c2 = Or ;

// Programme principal
plot2d(%inf,%inf,frameflag=3,rect=[-R,-R,R,R],axesflag=0)
drawlater()
for k = 1:b
    C = r*exp(%i*(%pi/2 + (2*k-1)*%pi/b)) ;
    D = r*exp(%i*(%pi/2 + (2*k-3)*%pi/b)) ;
    S1 = R*exp(%i*(%pi/2 + (2*k-2)*%pi/b)) ;
    S2 = R*exp(%i*(%pi/2 + (2*k)*%pi/b)) ;
    H1 = C*cos(%pi/b)*exp(-%i*%pi/b) ;
    H2 = C*cos(%pi/b)*exp(+%i*%pi/b) ;
    T = [0;H1;H2] ;
    Triangle(T,n,1)
    T = [S1;C;D] ;
    Triangle(T,n-1,2)
    Branch(C,S1,S2,n-1)
end
drawnow()
```

Mathématique & Informatique



Mathématique & Informatique

Polyèdre sphérique

Une sphère peut être approchée d'aussi près que l'on veut par un polyèdre convexe à faces triangulaires.
La première méthode (itérative) consiste à tracer sur la sphère des demi-cercles méridiens séparés d'une variation de longitude $\Delta\theta$ et des cercles parallèles séparés d'une variation de colatitude $\Delta\alpha$. En reliant convenablement entre eux les points d'intersection par des segments de droites, on construit un polyèdre convexe à faces triangulaires.
Etudions une deuxième méthode (récursive) basée sur un icosaèdre à 20 faces et 30 arêtes inscrit dans la sphère.
Prenons les rayons issus du centre de la sphère et passant par les milieux des arêtes de chaque face de l'icosaèdre.
On obtient un ensemble de points qui sont les sommets d'un nouveau polyèdre constitué de $20\times4 = 80$ faces.
En répétant le procédé n fois on aboutit à un polyèdre convexe à 20×4^n faces, 30×4^n arêtes et $10\times4^n + 2$ sommets.
Notons que ce polyèdre n'est pas régulier ($n > 0$) puisque ses arêtes ne sont pas toutes identiques.
La fonction de calcul des facettes du polyèdre d'ordre n, inscrit dans la sphère de rayon R, est la suivante :

```
function [xf,yf,zf] = Mesh_sphere(R,n)
    function triangle(s1,s2,s3,n)
        global xf yf zf
        if n > 0 then
            p1 = (s1 + s2)/2
            p2 = (s2 + s3)/2
            p3 = (s3 + s1)/2
            sp1 = p1/norm(p1)*R
            sp2 = p2/norm(p2)*R
            sp3 = p3/norm(p3)*R
            triangle(s1,sp1,sp3,n-1)
            triangle(s2,sp2,sp1,n-1)
            triangle(s3,sp3,sp2,n-1)
            triangle(sp1,sp2,sp3,n-1)
        else
            xf = [xf,[s1(1);s2(1);s3(1)]]
            yf = [yf,[s1(2);s2(2);s3(2)]]
            zf = [zf,[s1(3);s2(3);s3(3)]]
        end
    endfunction
    global xf yf zf
    xf = [] ; yf = [] ; zf = []
    phi = (1 + sqrt(5))/2
    S = R/sqrt(phi+2)*[phi,0,1;-phi,0,1;-phi,0,-1;phi,0,-1;0,1,phi;0,1,-phi;
                       0,-1,-phi;0,-1,phi;1,phi,0;1,-phi,0;-1,-phi,0;-1,phi,0]
    F = [8,5,1;8,2,5;8,11,2;8,10,11;8,1,10;10,1,4;1,9,4;1,5,9;5,12,9;5,2,12;
         2,3,12;2,11,3;11,7,3;11,10,7;10,4,7;6,4,9;6,9,12;6,12,3;6,3,7;6,7,4]
    for i = 1:20
        s1 = S(F(i,1),:)
        s2 = S(F(i,2),:)
        s3 = S(F(i,3),:)
        triangle(s1,s2,s3,n)
    end
endfunction
```

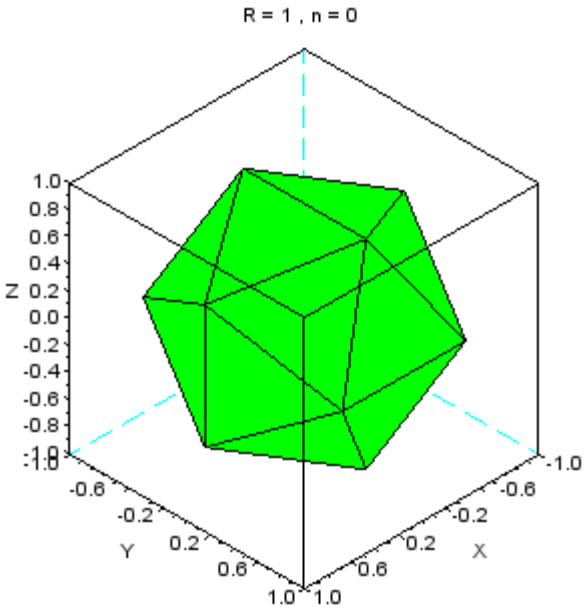
Ce calcul de maillage sphérique n'utilise pas les fonctions cosinus et sinus, mais uniquement la racine carrée (*sqrt*).
Pour positionner différemment l'icosaèdre de départ dans le repère (0,x,y,z) on multiplie la matrice S des sommets par une matrice de rotation. Par exemple, si l'on veut que le premier sommet de l'icosaèdre soit sur l'axe des z :

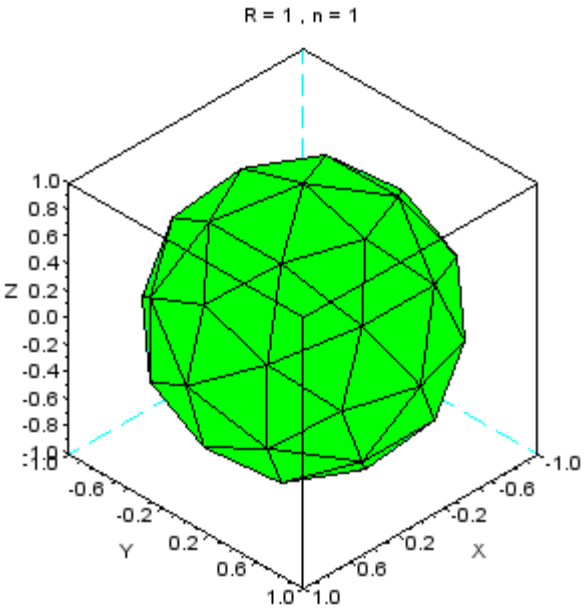
```
S = S*[1,0,phi;0,sqrt(phi+2),0;-phi,0,1]/sqrt(phi+2)
```

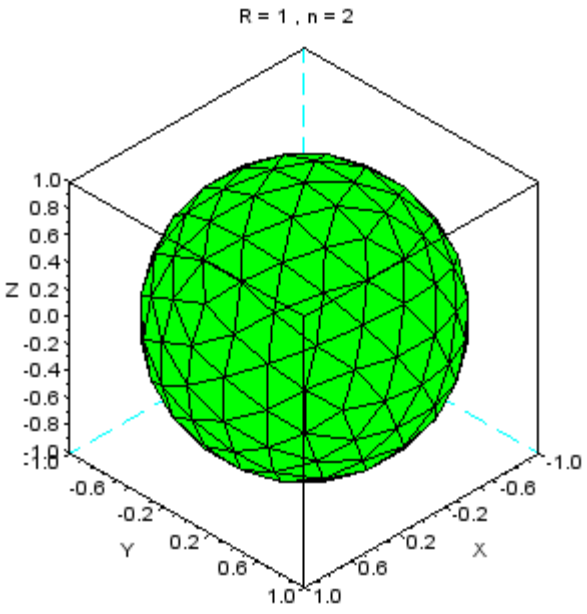
Représentons les polyèdres d'ordres 0, 1, 2, 3, inscrits dans la sphère unité :

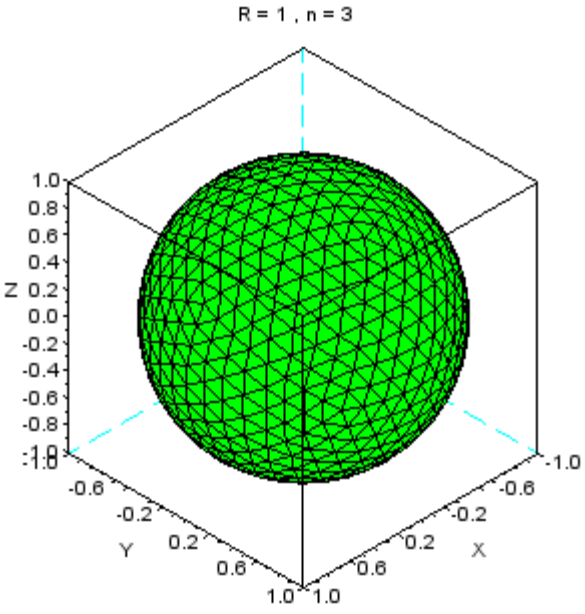
```
R = 1 ;
for   n = 0:3
    [xf,yf,zf] = Mesh_sphere(R,n) ;
    subplot(2,2,n+1) ; xtitle('R = ' + string(R) + ' , n = ' + string(n))
    plot3d(xf, yf, zf, theta = 45, alpha = 55, flag = [3,4,4])
end
```

Mathématique & Informatique









Mathématique & Informatique

Annexe : le calcul du nombre de faces, d'arêtes et de sommets du polyèdre sphérique d'ordre n

A l'ordre 0, le polyèdre est un icosaèdre à $f_0 = 20$ faces, $a_0 = 30$ arêtes et $s_0 = 12$ sommets.
Par construction, lorsqu'on passe de l'ordre n-1 à l'ordre n :

- le nombre de faces est multiplié par 4
- le nombre d'arêtes est doublé et chaque face donne lieu à 3 nouvelles arêtes
- le nombre de sommets augmente de la valeur du nombre d'arêtes

Ce qui se traduit par les équations de récurrence :

$$f_n = 4\, f_{n-1}$$
$$a_n = 2\, a_{n-1} + 3\, f_{n-1}$$
$$s_n = s_{n-1} + a_{n-1}$$

En notation matricielle :

$$\begin{pmatrix} f_n \\ a_n \\ s_n \end{pmatrix} = \begin{pmatrix} 4 & 0 & 0 \\ 3 & 2 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} f_{n-1} \\ a_{n-1} \\ s_{n-1} \end{pmatrix} = \begin{pmatrix} 4 & 0 & 0 \\ 3 & 2 & 0 \\ 0 & 1 & 1 \end{pmatrix}^n \begin{pmatrix} f_0 \\ a_0 \\ s_0 \end{pmatrix} = M^n \begin{pmatrix} 20 \\ 30 \\ 12 \end{pmatrix}$$

La matrice M admet 3 valeurs propres distinctes : 4, 2, 1. Elle est donc diagonalisable sous la forme D = diag(4,2,1) dans une base de vecteurs propres. On peut prendre comme matrice de passage :

$$P = \begin{pmatrix} 2 & 0 & 0 \\ 3 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \quad \Rightarrow \quad P^{-1} = \begin{pmatrix} 1/2 & 0 & 0 \\ -3/2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix}$$

L'égalité $M = P\, D\, P^{-1}$, élevée à la puissance n, conduit à :

$$M^n = PD^nP^{-1} = P\begin{pmatrix} 4^n & 0 & 0 \\ 0 & 2^n & 0 \\ 0 & 0 & 1^n \end{pmatrix} P^{-1} = \begin{pmatrix} 2^{2n} & 0 & 0 \\ 3.2^{2n-1} - 3.2^{n-1} & 2^n & 0 \\ 2^{2n-1} - 3.2^{n-1} + 1 & 2^n - 1 & 1 \end{pmatrix}$$

D'où :

$$f_n = 20 \times 4^n \quad ; \quad a_n = 30 \times 4^n \quad ; \quad s_n = 10 \times 4^n + 2$$

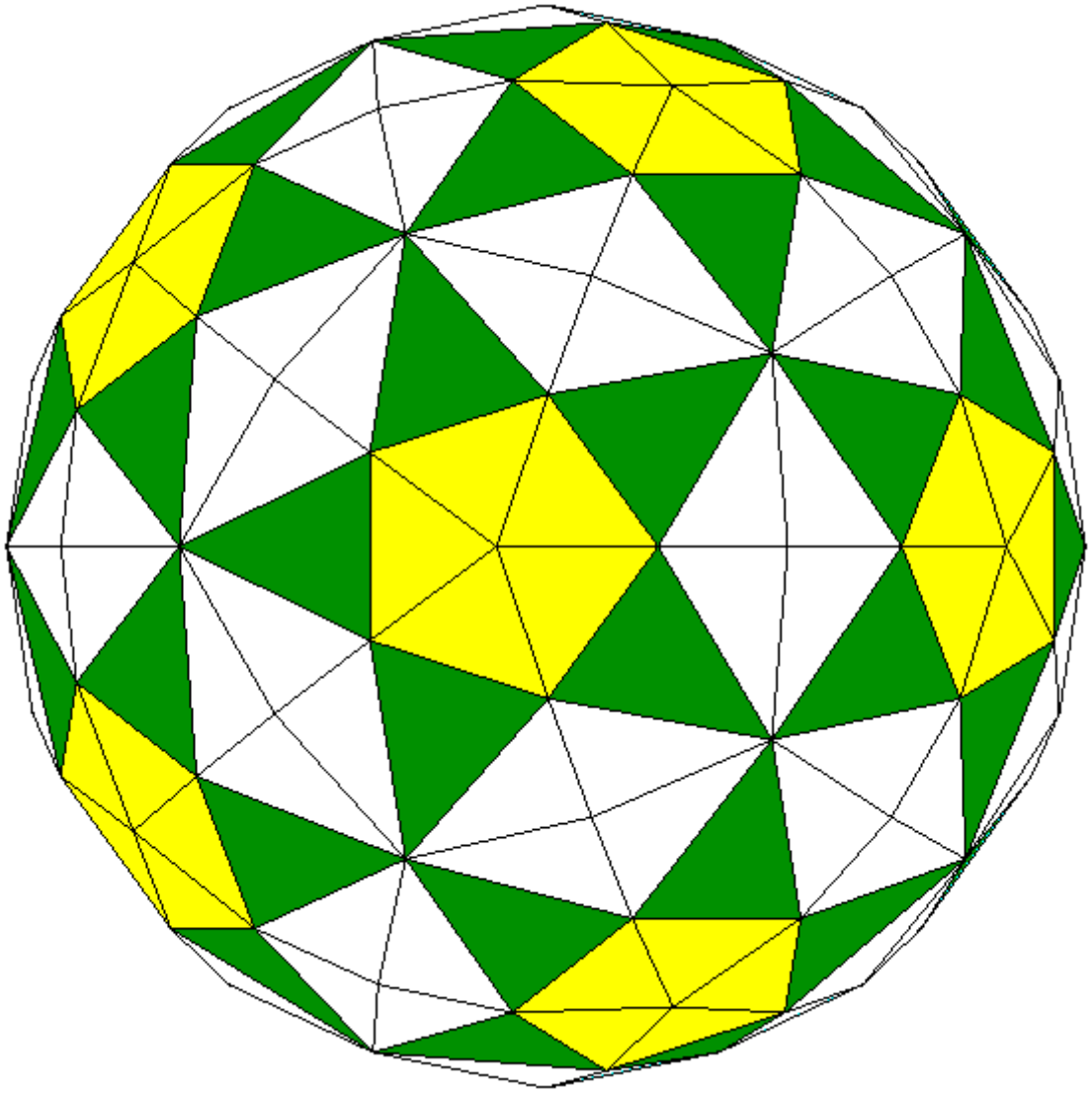
Remarque

Les nombres entiers f_n , a_n , s_n vérifient la relation d'**Euler - Descartes** applicable à tout polyèdre convexe :

$$f - a + s = 2$$

Cette formule, dont découlent plusieurs propriétés des polyèdres et des polygones, peut se démontrer par récurrence sur le nombe d'arêtes en se ramenant à la théorie des graphes. Dans ce cas, un polyèdre convexe est assimilé à un graphe "planaire" où les sommets sont des points distincts, les arêtes sont des courbes simples qui ne se rencontrent pas en dehors de leurs extrémités, et les faces sont des cycles élémentaires du graphe, c'est-à-dire des séquences circulaires alternées de sommets et d'arêtes qui ne comportent pas deux fois le même élément.

D'autres combinaisons sont possibles : en partant d'un dodécaèdre on construit un polyèdre sphérique, autrement dit un polyèdre dont tous les sommets appartiennent à une sphère, de sorte que ses facettes soient disposées en étoiles à cinq branches. Le polyèdre représenté ci-dessous possède 240 faces, 360 arêtes et 122 sommets.



Mathématique & Informatique

Application de la formule d'**Euler - Descartes**

1) Un polyèdre convexe possède au moins un sommet de degré inférieur ou égal à 5
2) Il existe uniquement 5 polyèdres réguliers dans l'espace
3) Le plan ne peut être recouvert qu'avec 3 types de polygones réguliers

Pour démontrer ces propositions, on confondra un polyèdre convexe avec le graphe simple planaire associé, graphe sans arêtes qui se coupent, sans boucles sur les sommets et sans arêtes multiples joignant 2 sommets.
Rappelons que le degré d'un sommet est le nombre d'arêtes adjacentes à ce sommet et convenons que le périmètre d'une face est le nombre d'arêtes bordant cette face.
Soient f le nombre de faces, a le nombre d'arêtes, s le nombre de sommets d'un polyèdre.
Introduisons les grandeurs arithmétiques $d = 2a/s$ (degré moyen) et $p = 2a/f$ (périmètre moyen).
Pour exprimer f, a, s en fonction de d et p, on résout un système linéaire de 3 équations à 3 inconnues :

$$\begin{cases} f - a + s = 2 \\ -2a + ds = 0 \\ pf - 2a = 0 \end{cases} \Rightarrow \begin{cases} f = \dfrac{4d}{2d + 2p - dp} \\ a = \dfrac{2dp}{2d + 2p - dp} \\ s = \dfrac{4p}{2d + 2p - dp} \end{cases}$$

Comme f, a, s > 0 et d, p > 0, on a nécessairement 2d + 2p - dp > 0 pour un graphe fini. D'où l'inégalité :

$$(d - 2)(p - 2) < 4$$

NB Dans le cas général, d et p sont des rationnels alors que f, a, s sont toujours des entiers

Le sommet de degré minimal

Notons f_k le nombre de faces de périmètre k. Chaque face est bordée par au moins 3 arêtes et chaque arête borde 2 faces.
En prenant les arêtes une à une, on compte les faces bordées par chaque arête : on dénombre 2a faces en tout
et chaque face de périmètre k est comptée k fois. Il s'ensuit :

$$2a = \Sigma k f_k \text{ , } k \geq 3 \quad \Rightarrow \quad 2a - 3f = \Sigma k f_k - 3\Sigma f_k = \Sigma(k - 3)f_k \geq 0 \quad \Rightarrow \quad 2a \geq 3f \quad \Rightarrow \quad p \geq 3$$

Notons s_k le nombre de sommets de degré k. En prenant les arêtes une à une, chaque arête augmente de 1 le degré de 2 sommets différents (ou de 2 le degré du même sommet pour une boucle). Ainsi, la somme des degrés des sommets d'un graphe vaut 2a. Supposons que chaque sommet x possède un degré d(x) strictement supérieur à 5. On a alors :

$$2a = \Sigma k s_k \text{ , } k \geq 6 \quad \Rightarrow \quad 2a - 6s = \Sigma k s_k - 6\Sigma s_k = \Sigma(k - 6)s_k \geq 0 \quad \Rightarrow \quad 2a \geq 6s \quad \Rightarrow \quad d \geq 6$$

Sous l'hypothèse précédente : $(d - 2)(p - 2) \geq (6 - 2)(3 - 2) = 4$ CONTRADICTION

Donc : $\exists\, x \,/\, d(x) \leq 5$

Par récurrence sur le nombre de sommets, on déduit de cette propriété que tout graphe planaire est 5-coloriable.
En fait, 4 couleurs suffisent. La preuve, connue sous le nom de "théorème des 4 couleurs", est difficile à établir.
La démonstration consiste à diviser le problème en un nombre fini de 663 configurations possibles et à utiliser un ordinateur pour vérifier qu'une solution existe toujours ; voir le sujet proposé par Georges **Gonthier** à l'adresse
⇨ **http://www.enseignement.polytechnique.fr/profs/informatique/Georges.Gonthier/pi2000/pro/gonthier/**

Mathématique & Informatique

Les solides de **Platon**

Considérons un polyèdre convexe dont tous les sommets ont le même degré, égal au degré moyen d, et toutes les faces ont le même périmètre, égal au périmètre moyen p.
Autrement dit, on impose à d et p d'être des entiers, et compte tenu du résultat précédent, on doit vérifier dans $\mathbb{N}$:

$$(d - 2)(p - 2) < 4 \quad \text{avec } d \in \{3,4,5\} \text{ et } p \in \{3,4,5, ...\}$$

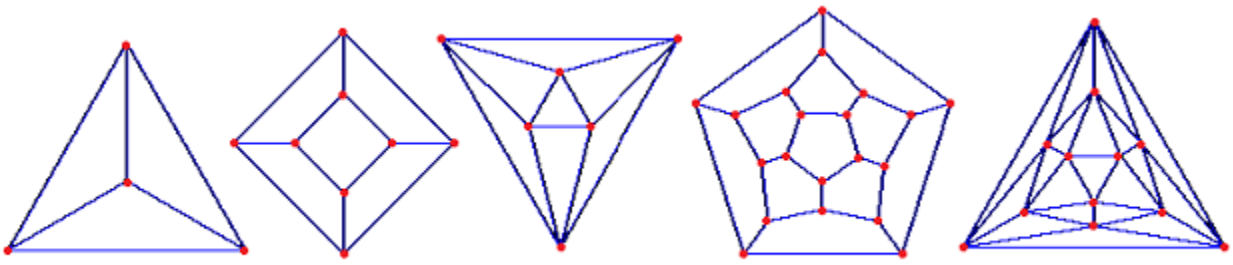
Étudions les différentes possibilités :

- $d = 3 \Rightarrow p < 6 \Rightarrow p = 3, 4, 5$
- $d = 4 \Rightarrow p < 4 \Rightarrow p = 3$
- $d = 5 \Rightarrow p < 10/3 \Rightarrow p = 3$

Regroupons dans un tableau les valeurs de d et p, puis calculons les nombres f, a, s correspondants :

p	d	f	a	s	Nom usuel	Solide de **Platon**
3	3	4	6	4	tétraèdre	Le feu
3	4	8	12	6	octaèdre	L'air
3	5	20	30	12	icosaèdre	L'eau
4	3	6	12	8	hexaèdre	La terre
5	3	12	30	20	dodécaèdre	L'univers

La notation {p,d} est le symbole de **Schläfli** d'un polyèdre régulier.
Par exemple, {5,3} est le symbole de **Schläfli** du dodécaèdre.
Ci-dessous, les diagrammes de **Schlegel** (graphes planaires) des 5 polyèdres réguliers :
⇨ **http://www.ac-noumea.nc/maths/polyhedr/Schlegel.htm**



Le pavage du plan

Tout revient à imaginer l'existence d'un polyèdre régulier ayant une infinité de faces, d'arêtes et de sommets ; ce cas limite s'obtient avec $2d + 2p - dp = 0$, soit $p = 2d/(d - 2)$
On a vu que : $p \geq 3$. D'où : $d \leq 6$
Les seules valeurs entières possibles des 2 paramètres sont :

- $d = 3$ et $p = 6$: hexagone
- $d = 4$ et $p = 4$: carré
- $d = 6$ et $p = 3$: triangle

Polyèdre ellipsoïdal

Soit un ellipsoïde de demi-axes *a*, *b*, *c* et d'équation cartésienne : $x^2/a^2 + y^2/b^2 + z^2/c^2 = 1$
Le rayon issu du centre O et passant par le point P*(u,v,w)* rencontre la surface de l'ellipsoïde au point de coordonnées:

$$x = u / \sqrt{(u/a)^2 + (v/b)^2 + (w/c)^2}$$
$$y = v / \sqrt{(u/a)^2 + (v/b)^2 + (w/c)^2}$$
$$z = w / \sqrt{(u/a)^2 + (v/b)^2 + (w/c)^2}$$

Considérons l'octaèdre à 8 faces triangulaires dont les 6 sommets de coordonnées $(\pm a,0,0)$, $(0,\pm b, 0)$, $(0,0,\pm c)$ sont les extrémités de l'ellipsoïde. Ecrivons la fonction de calcul des facettes du polyèdre d'ordre n, comme on l'a fait pour la sphère, avec en plus une coloration des triangles utilisant 2 couleurs numérotées *d* et *e* dans la palette Scilab :

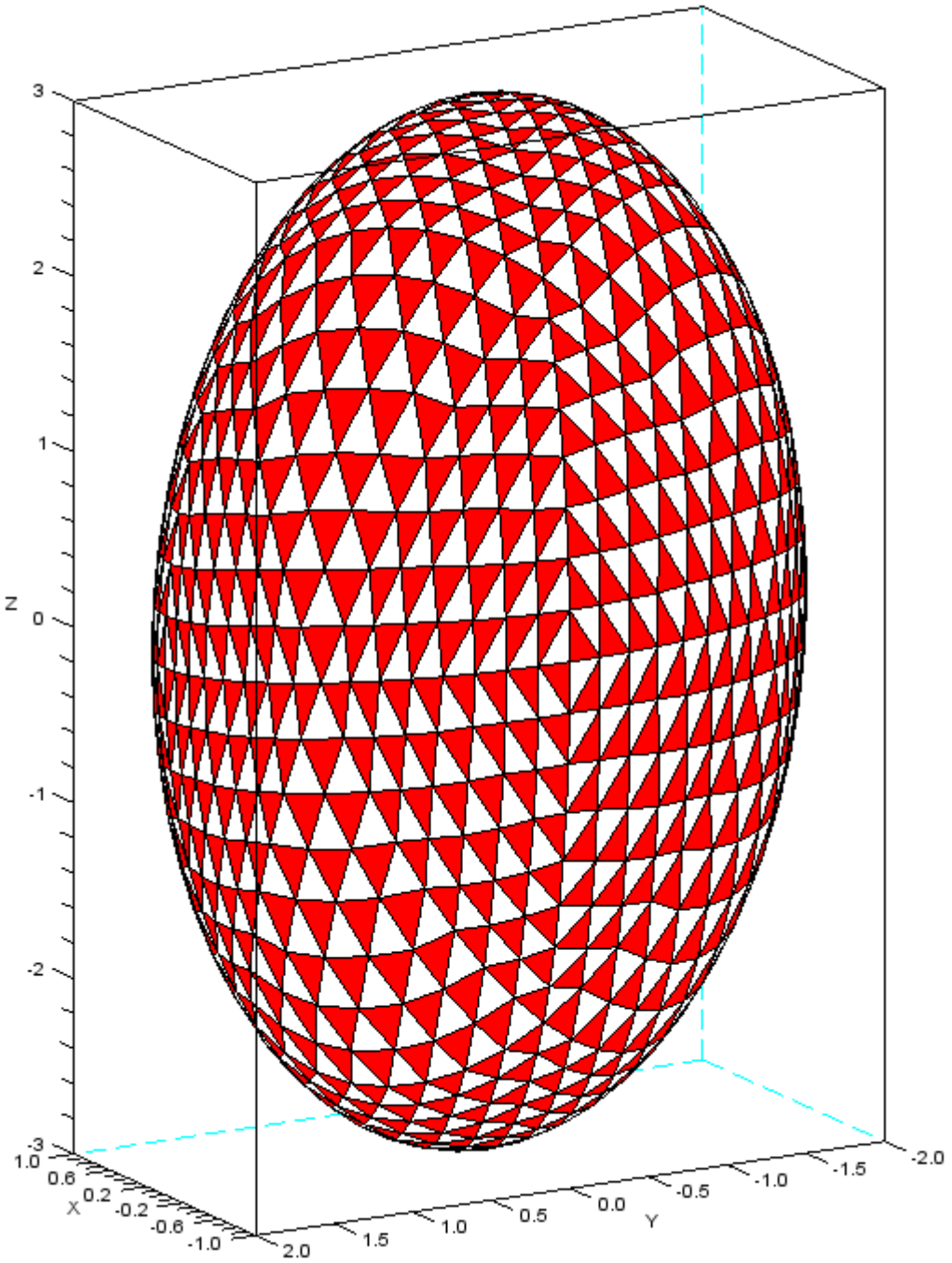
```
function [xf,yf,zf,cf] = Mesh_ellipsoid(a,b,c,d,e,n)
    function triangle(s1,s2,s3,ct,n)
        global xf yf zf cf
        if n > 0 then
            p1 = (s1 + s2)/2
            p2 = (s2 + s3)/2
            p3 = (s3 + s1)/2
            sp1 = p1/norm(p1./[a,b,c])
            sp2 = p2/norm(p2./[a,b,c])
            sp3 = p3/norm(p3./[a,b,c])
            triangle(s1,sp1,sp3,ct,n-1)
            triangle(s2,sp2,sp1,ct,n-1)
            triangle(s3,sp3,sp2,ct,n-1)
            triangle(sp1,sp2,sp3,d+e-ct,n-1)
        else
            xf = [xf,[s1(1);s2(1);s3(1)]]
            yf = [yf,[s1(2);s2(2);s3(2)]]
            zf = [zf,[s1(3);s2(3);s3(3)]]
            cf = [cf,ct]
        end
    endfunction
    global xf yf zf cf
    xf = [] ; yf = [] ; zf = [] ; cf = []
    S = [a,0,0;0,b,0;0,0,c;-a,0,0;0,-b,0;0,0,-c]
    F = [1,5,3;2,1,3;4,2,3;5,4,3;1,2,6;2,4,6;4,5,6;5,1,6]
    C = [d,e,d,e,d,e,d,e]
    for i = 1:8
        s1 = S(F(i,1),:)
        s2 = S(F(i,2),:)
        s3 = S(F(i,3),:)
        ct = C(i)
        triangle(s1,s2,s3,ct,n)
    end
endfunction
```

Par exemple, pour obtenir un polyèdre ellipsoïdal de demi-axes 1, 2, 3, à l'ordre 4 avec les couleurs 8 et 5 :

```
[xf,yf,zf,cf] = Mesh_ellipsoid(1,2,3,8,5,4) ;
plot3d(xf, yf, list(zf,cf), alpha = 75, theta = 150, flag = [4,4,4])
```

Remarque
On peut généraliser la méthode à un volume quelconque contenant un polyèdre convexe à faces triangulaires.

Mathématique & Informatique



Mathématique & Informatique

Ensembles de Mandelbrot et de Julia

Les ensembles de points de Benoît **Mandelbrot** et de Gaston **Julia** se déduisent de l'étude de la convergence des suites de nombres complexes définies par la relation de récurrence $z_{n+1} = z_n^2 + c$ où les valeurs z_0 et c sont données.

Ensemble de **Mandelbrot**

En partant de $z_0 = 0$ et en faisant varier $c = a + ib$ dans une partie du plan complexe, on affiche en noir (couleur par défaut) le point c de coordonnées (a,b) si la suite (z_n) reste bornée, c'est-à-dire si $|z_n| \leq 2$. Dans le cas contraire, le point c est visualisé avec une couleur différente qui dépend de la vitesse de divergence de la suite (z_n), plus exactement une couleur qui correspond au nombre de termes z_n qui ne vérifient pas l'inégalité $|z_n| > 2$.

```
// Ensemble de Mandelbrot
f = scf(1) ;
xtitle('Ensemble de Mandelbrot')
a = gca() ; a.isoview ='on' ;
N = 50 ;
f.color_map = jetcolormap(N) ;
drawlater()
step = 1/100 ;
for a = -2:step:1
    for b = -1.5:step:1.5
        plot2d(a,b,0)
        c = a + %i*b ;
        z = 0 ;
        for n = 1:N
            if abs(z) > 2 then
              e = gce() ; e.children.mark_foreground = n   ;
              break
            end
            z = z*z + c ;
        end
    end
end
drawnow()
```

Remarques

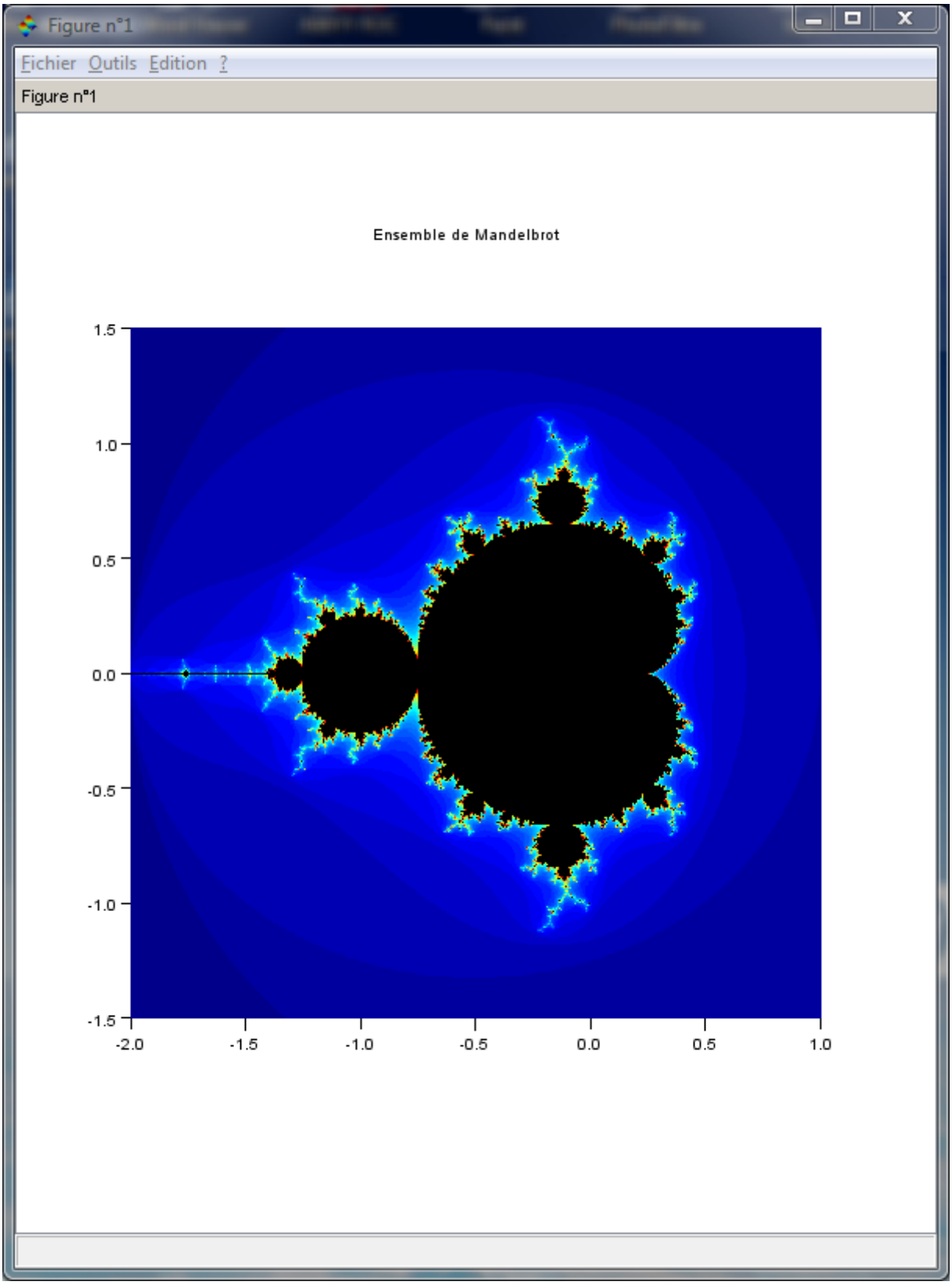
- Montrons la divergence de la suite (z_n) lorsque $|z_n| > 2$ et $|c| \leq 2$

Hypothèse : $\exists\ n, |z_n| > 2 \quad \Rightarrow \quad |z_n| = 2 + \varepsilon$
Il s'ensuit : $|z_{n+1}| = |z_n^2 + c| \geq |z_n|^2 - |c| > |z_n|^2 - |z_n| = |z_n|(1 + \varepsilon)$
On déduit : $\forall\ p > 0, |z_{n+p}| > |z_n|(1 + \varepsilon)^p$
Donc : $|z_n| \rightarrow + \infty$ quand $n \rightarrow + \infty$

- La représentation de l'ensemble de **Mandelbrot** (zone en noir sur le dessin) n'est qu'approximative en raison de la résolution choisie (step = 1/100) et du nombre limité d'itérations (N = 50) pour vérifier si la suite (z_n) est divergente. Schématiquement, la frontière de l'ensemble de **Mandelbrot** est la juxtaposition d'une cardioïde d'équation polaire $r = \frac{1}{2}(1 - \cos\theta)$ centrée en $(\frac{1}{4},0)$ et d'un cercle de rayon $R = \frac{1}{4}$ centré en (-1,0). On montre que ces 2 courbes correspondent aux points fixes des applications $f(z) = z^2 + c$ et $f^2(z) = z^4 + 2cz^2 + c^2 + c$ avec les conditions limites de stabilité $|f'(z)| = 1$ et $|(f^2)'(z)| = 1$. Plus généralement, chaque p-cycle stable de f (p = 1, 2, 3, ...), c'est-à-dire tous les points c tels que $f^p(z) = z$ avec $|(f^p)'(z)| = 1$, ajoute de nouveaux éléments à l'ensemble initial. Concernant la topologie, l'ensemble de **Mandelbrot** est connexe. Pour une étude plus approfondie, consulter le document d'Adrien **Douady** sur la dynamique des polynômes complexes à l'adresse internet :

⇨ **http://www.math.cornell.edu/~hubbard/OrsayFrench.pdf**

Mathématique & Informatique

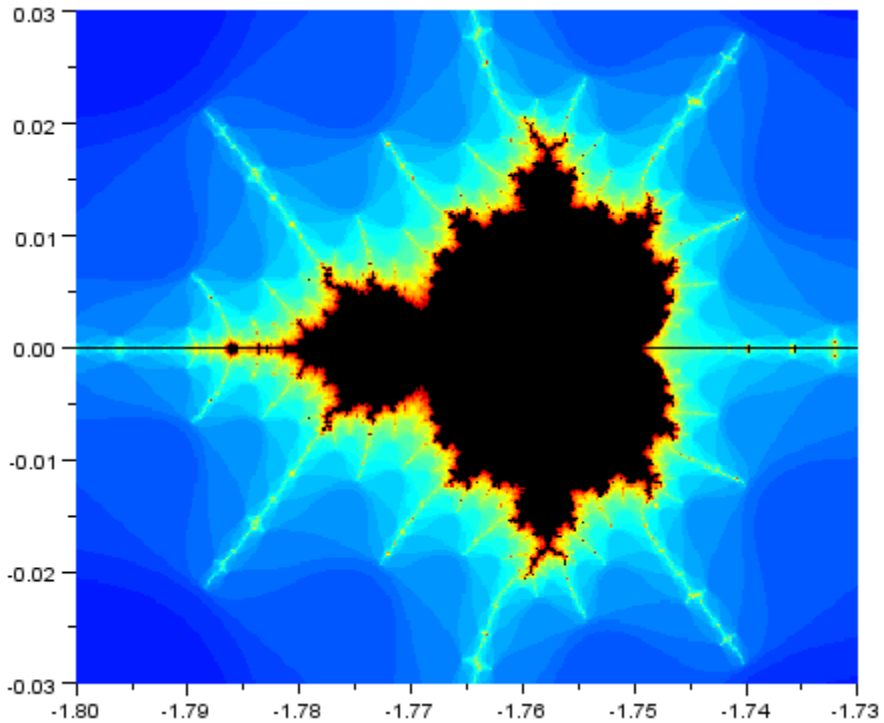


Zoom

Pour effectuer un zoom sur une partie de l'image, on utilise l'algorithme précédent en restreignant le domaine de variation du complexe c et en diminuant la valeur du pas de calcul.
Par exemple, agrandissons la pointe de la fractale de **Mandelbrot** située dans le rectangle de coordonnées cartésiennes [xmin, ymin, xmax, ymax] = [-1.80, -0.03, -1.73, +0.03] en prenant un pas de 1/5000 :

```
step = 1/5000 ;
for a = -1.80:step:-1.73
    for b = -0.03:step:+0.03
        plot2d(a,b,0)
        c = a + %i*b ;
        z = 0 ;
        for n = 1:N
            if abs(z) > 2 then
              e = gce() ; e.children.mark_foreground = n   ;
              break
            end
            z = z*z + c ;
        end
    end
end
```

On obtient la figure suivante :



La partie centrale (en noir) est l'homothétique de la première figure représentant l'ensemble de **Mandelbrot** en entier.
Les petites différences dans la forme sont dues au manque de précision dans les calculs.
Ce résultat atteste de la propriété d'autosimilarité de la fractale : le même motif géométrique apparaît à des échelles de plus en plus petites jusqu'à des dimensions infinitésimales.

Mathématique & Informatique

Ensemble de **Julia**

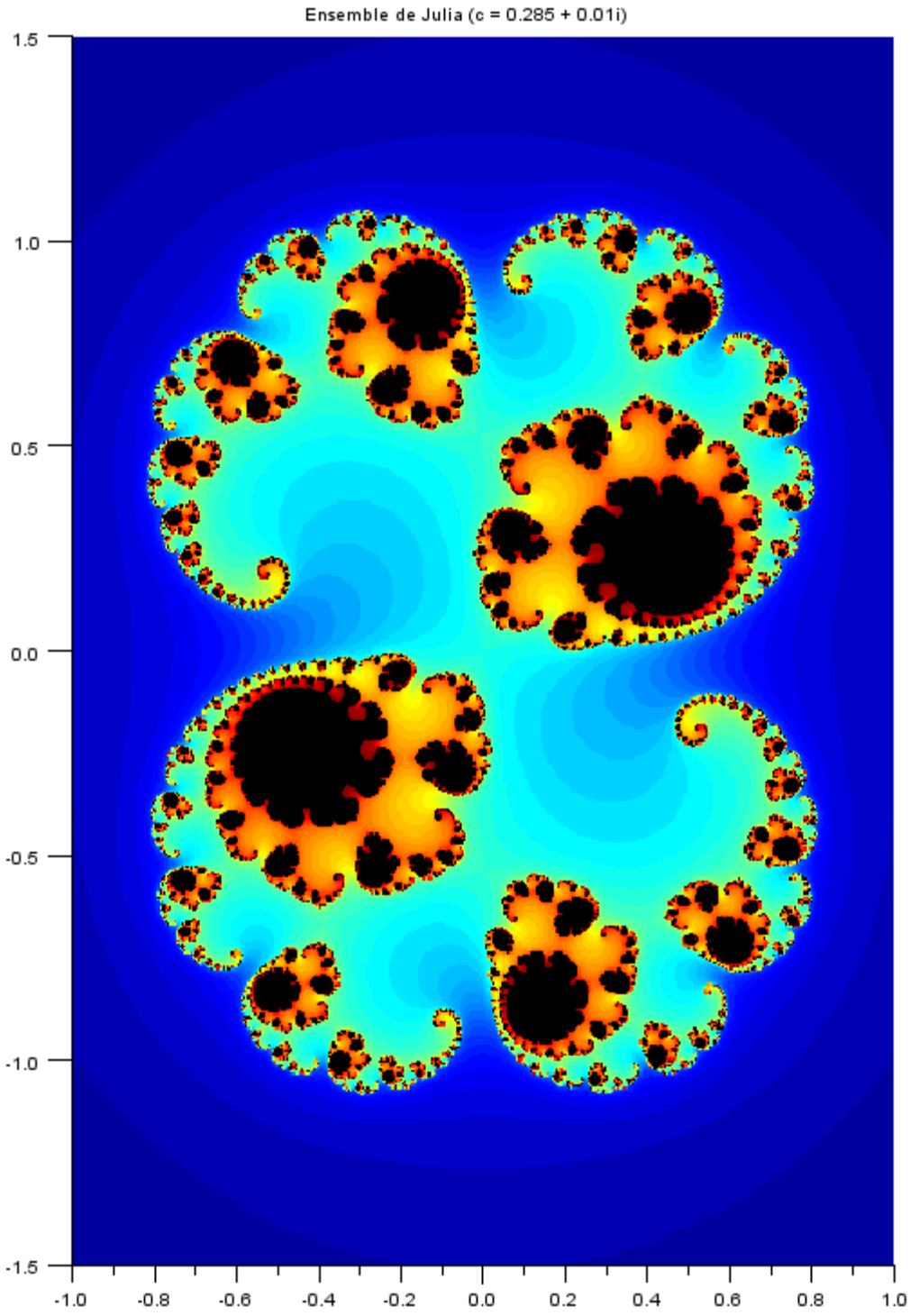
En attribuant à c une valeur particulière et en faisant varier $z_0 = x + iy$ dans une partie du plan complexe, on affiche en noir (couleur par défaut) le point c de coordonnées (x,y) si la suite (z_n) reste bornée, c'est-à-dire si $|z_n| \leq 2$. Dans le cas contraire, le point c est visualisé avec une couleur différente qui dépend de la vitesse de divergence de la suite (z_n), plus exactement une couleur qui correspond au nombre de termes z_n qui ne vérifient pas l'inégalité $|z_n| > 2$.

```
// Ensemble de Julia
f = scf(3) ;
xtitle('Ensemble de Julia (c = 0.285 + 0.01i)')
a = gca() ; a.isoview ='on' ;
N = 50 ;
f.color_map = jetcolormap(N) ;
drawlater()
c = 0.285 + 0.01*%i ;
step = 5/1000 ;
for x = -1:step:1
    for y = -1.5:step:1.5
        plot2d(x,y,0)
        z = x + %i*y ;
        for n = 1:N
            if abs(z) > 2 then
              e = gce() ; e.children.mark_foreground = n   ;
              break
            end
            z = z*z + c ;
        end
    end
end
drawnow()
```

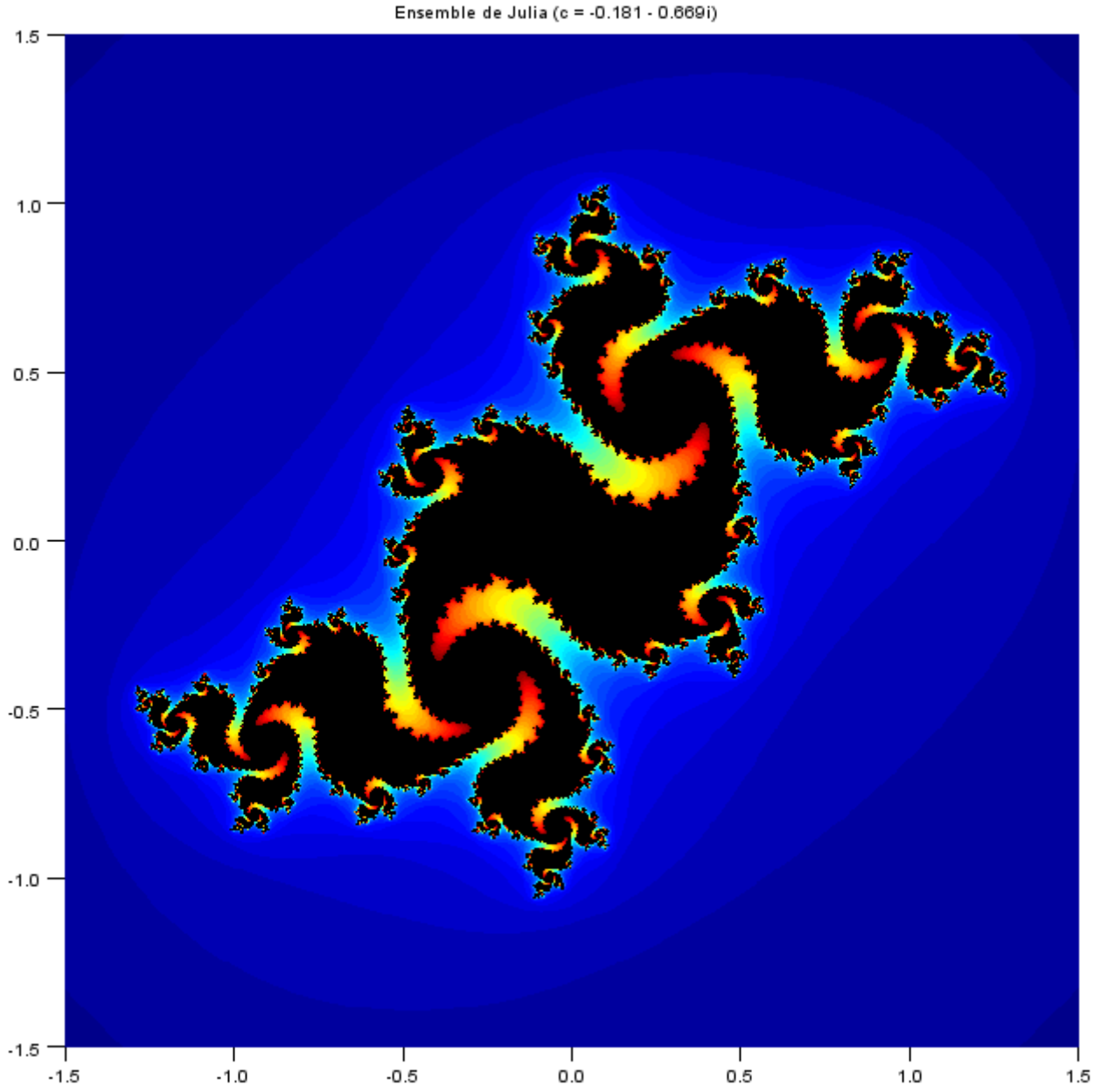
Note Le temps d'exécution de ce programme est d'environ 30 mn avec un micro-ordinateur.

Soit f_c la fonction qui associe au nombre complexe z le nombre complexe $z^2 + c$. L'ensemble de **Julia** proprement dit, noté $J(f_c)$, est la frontière de l'ensemble de **Julia** rempli, noté $J_r(f_c)$. Par abus de langage, l'ensemble $J_r(f_c)$ est aussi appelé simplement ensemble de **Julia**. Il existe une infinité d'ensembles de **Julia** (infinité de valeurs de c). Par contre, l'ensemble de **Mandelbrot** est unique.
On a pris c = 0.285 + 0.01i, mais on aurait pu choisir d'autres valeurs, par exemple : c = - 0.181 - 0.669i
Lorsqu'il contient le point 0, l'ensemble de **Julia** est connexe (exemple c = - 0.181 - 0.669i). A l'inverse, lorsqu'il ne contient pas le point 0, l'ensemble de **Julia** est non connexe (exemple c = 0.285 + 0.01i).

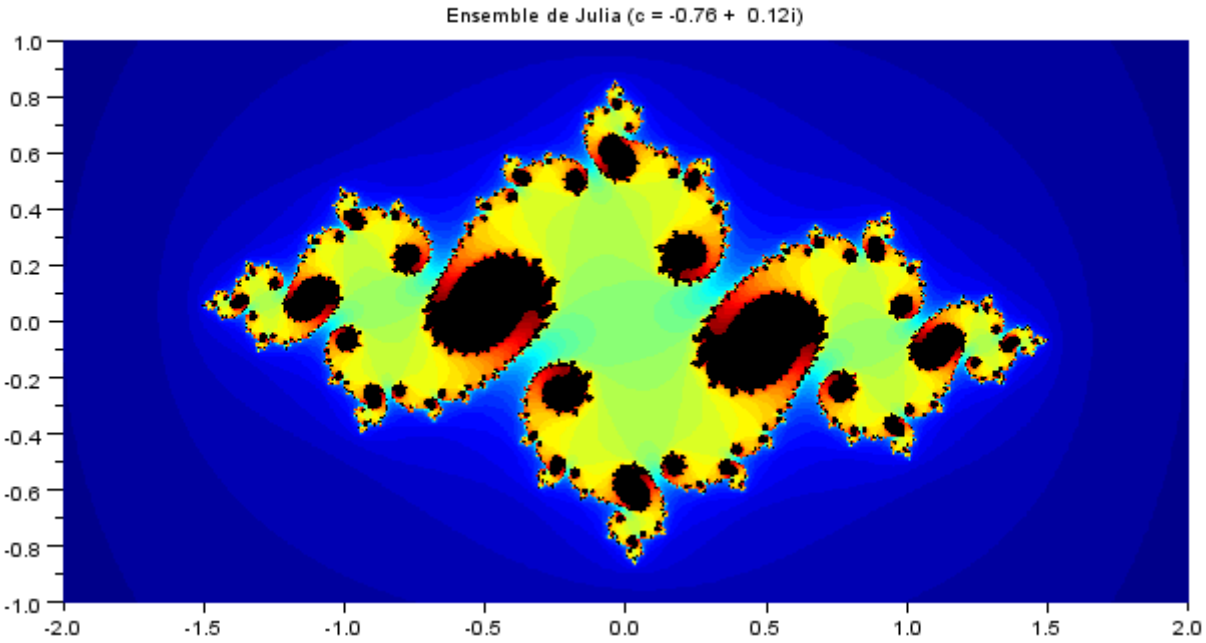
Mathématique & Informatique

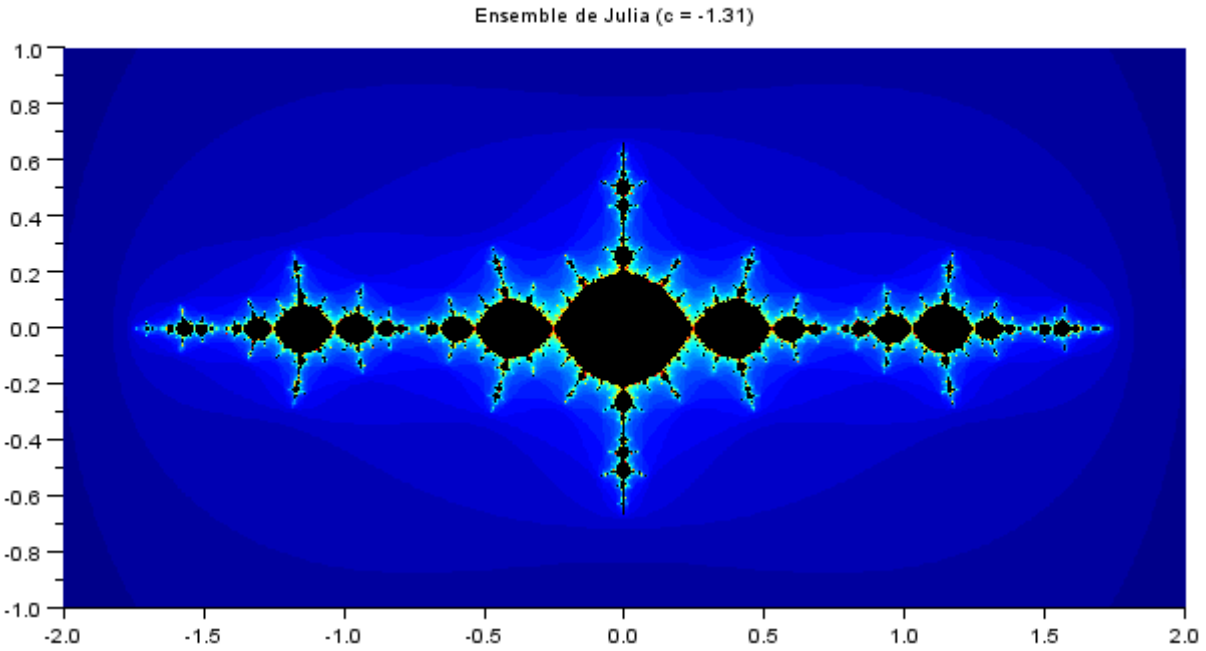


Mathématique & Informatique



Mathématique & Informatique





Mathématique & Informatique

Construction rapide de l'ensemble de **Julia** avec une méthode récursive d'itération inverse

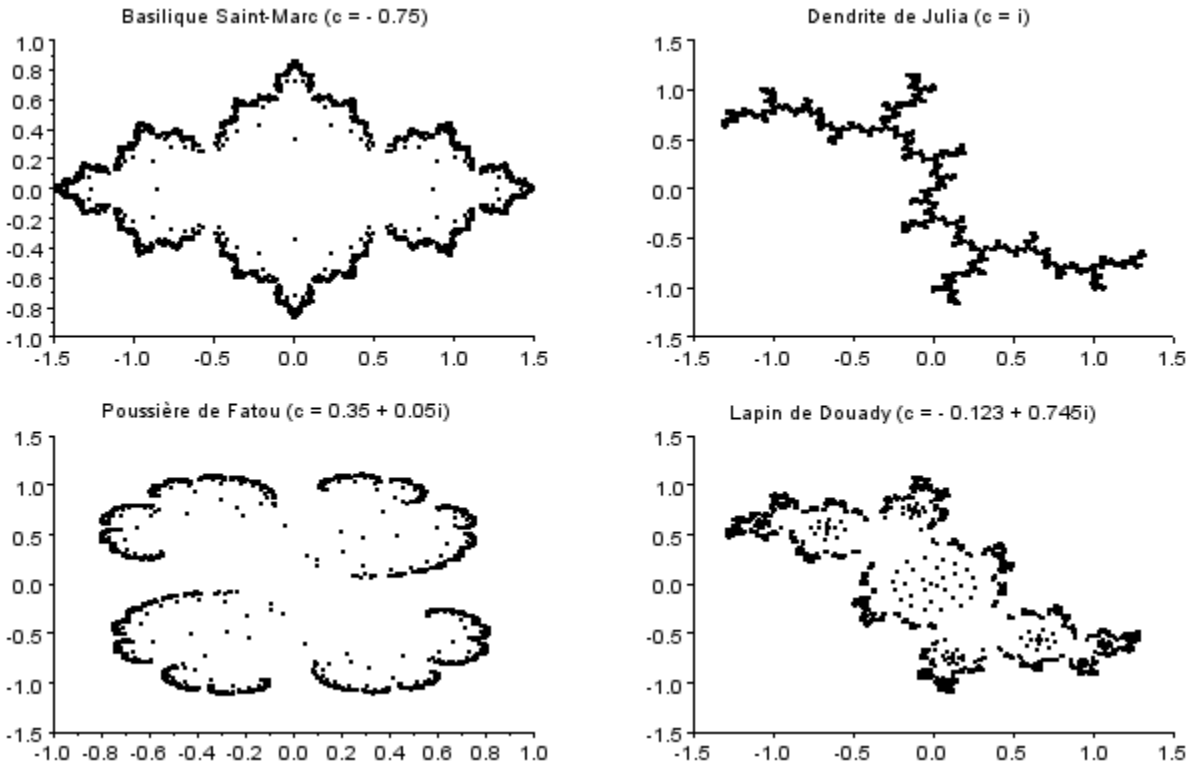
Le temps de calcul d'une fractale de **Julia** en haute résolution avec une palette de couleurs élargie est long.
On peut approximer rapidement la frontière de l'ensemble de **Julia** en utilisant une méthode itérative inverse :
⇨ **http://maths.wikidot.com/iteration-inverse** (John **Bonobo**)
Ecrivons une fonction Scilab récursive *Julia* qui effectue cette construction à partir de la formule $z_n = \pm (z_{n+1} - c)^{1/2}$

```
function Julia(z,n)
    if n > 0 then
        z = sqrt(z-c)
        plot2d(real(z),imag(z),0)
        plot2d(-real(z),-imag(z),0)
        Julia(z,n-1)
        Julia(-z,n-1)
    end
endfunction
```

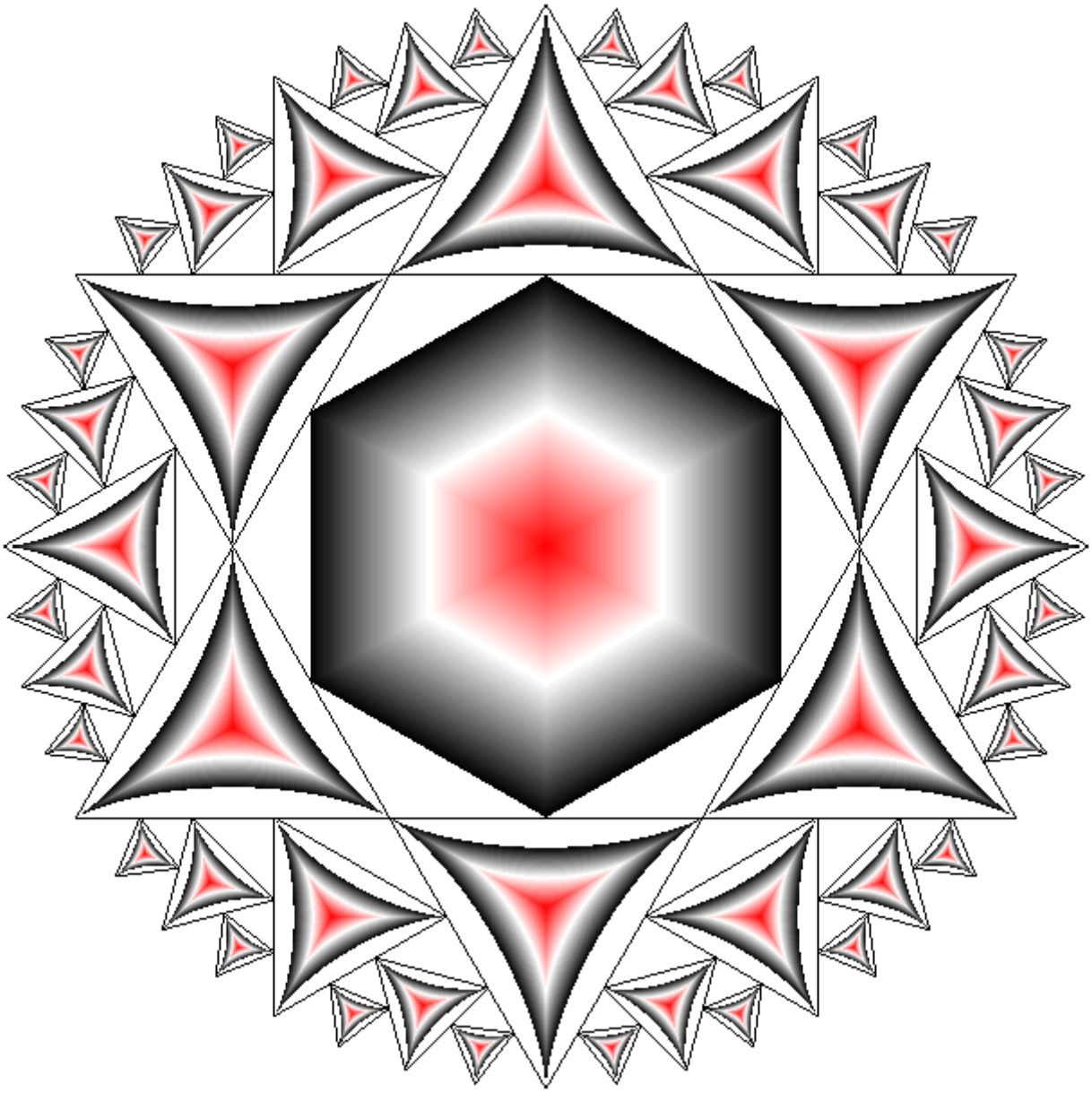
L'appel de cette fonction, avec une valeur n entière, affiche 2^{n+1} - 2 points.
Application à quelques ensembles de **Julia** remarquables :
⇨ **http://www.mathcurve.com/fractals/julia/julia.shtml** (Robert **Ferréol**)

```
f = scf() ;
drawlater()
subplot(2,2,1) ; c = - 0.75 ; xtitle('Basilique Saint-Marc (c = - 0.75)') ; Julia(0,12)
subplot(2,2,2) ; c = %i ; xtitle('Dendrite de Julia (c = i)') ; Julia(0,12)
subplot(2,2,3) ; c = 0.35 + 0.05*%i ; xtitle('Poussière de Fatou (c = 0.35 + 0.05i)') ; Julia(0,12)
subplot(2,2,4) ; c = - 0.123 + 0.745*%i ; xtitle('Lapin de Douady (c = - 0.123 + 0.745i)') ; Julia(0,12)
drawnow()
```

Résultat :



Mathématique & Informatique



Mathématique & Informatique

ALGORITHME EXHAUSTIF

Problème

On dispose de n variables X_i , $1 \le i \le n$, pouvant prendre respectivement N_i ($1 \le i \le n$) valeurs connues :

$X_1 = x_{11} , \ldots , x_{1N1}$
...
$X_n = x_{n1} , \ldots , x_{nNn}$

On souhaite tester tous les vecteurs $X = (X_1 , \ldots , X_n)$ possibles. Trivialement, l'algorithme s'écrit :

```
pour  in = 1  à  Nn
|       ......
|       pour  i1 = 1  à  N1
|       |       X = (x1i1 , ... , xnin)
|       |       < Tester le vecteur X >
|       fin
|       ......
fin
< Tous les vecteurs X ont été vérifiés >
```

Le nombre de vecteurs X est $N_1 \times \ldots \times N_n$. Dans le cas particulier où $N_i = N$, on a N^n possibilités.
Supposons que l'on ne connaisse pas à l'avance le nombre de variables (n est considéré comme un paramètre), alors le nombre de boucles imbriquées n'est pas constant et il faut trouver une autre méthode.

Solution

Posons $N = Max(N_1 , \ldots , N_n) + 1$
On construit le tableau T de dimension (n,N) tel que :

$$T(i,j) = \begin{cases} x_{ij} , 1 \le j \le N_i \\ 0 , N_i < j < N \\ N_i , j = N \end{cases}$$

L'algorithme suivant parcourt le tableau T à l'aide d'un vecteur d'indices I(i), $1 \le i \le n$, et teste tous les vecteurs $X = (X_1 , \ldots , X_n)$ possibles :

```
début :  pour  i = 1  à  n  faire  I(i) = 1
test :   pour  i = 1  à  n  faire  X(i) = T(i,I(i))
         < Tester le vecteur X >
         p = 0
suite :  p = p + 1
         I(p) = I(p) + 1
         si  I(p) ≤ T(p,N)  alors  aller à  test
         I(p) = 1
         si  p < n  alors  aller à  suite
fin :    < Tous les vecteurs X ont été vérifiés >
```

Cet algorithme a déjà été utilisé dans la résolution du SUDOKU où chaque case est assimilée à une variable (n = 81). Pour éviter une explosion combinatoire, toute la difficulté consiste à déterminer un ensemble de règles permettant de réduire autant que faire se peut le nombre de cas possibles, c'est-à-dire à diminuer la taille du tableau T.

Le programme correspondant s'obtient en remplaçant les étiquettes et les branchements conditionnels par une boucle :

```
X = zeros(1,n)
I = ones(1,n)
//   Définir le tableau T
loop = %t
test = %t
while loop
    if test then
        for i = 1:n
            X(i) = T(i,I(i))
        end
        //   Tester le vecteur X
        p = 0
    else
        I(p) = 1
    end
    p = p + 1
    I(p) = I(p) + 1
    test = I(p) <= T(p,N)
    loop = test | p < n
end
//   Tous les vecteurs X ont été vérifiés
```

On peut effectuer une simplification si les n variables X_i prennent le même nombre N de valeurs.
Le tableau T contient uniquement les valeurs possibles des X_i (ligne par ligne) :

$$T(i,j) = x_{ij} \qquad (1 \le i \le n\ ,\ 1 \le j \le N)$$

D'où le programme :

```
X = zeros(1,n)
I = ones(1,n)
//   Définir le tableau T
loop = %t
test = %t
while loop
    if test then
        for i = 1:n
            X(i) = T(i,I(i))
        end
        //   Tester le vecteur X
        p = 0
    else
        I(p) = 1
    end
    p = p + 1
    I(p) = I(p) + 1
    test = I(p) <= N
    loop = test | p < n
end
//   Tous les vecteurs X ont été vérifiés
```

EXEMPLE : Formule du multinôme de Newton

Il s'agit de la formule du binôme de **Newton** généralisée :

$$(x_1 + x_2 + \cdots + x_m)^n = \sum \frac{n!}{k_1!k_2!\cdots k_m!}\, x_1^{k_1} x_2^{k_2} \cdots x_m^{k_m}$$

La somme (Σ) porte sur tous les entiers k_1 , k_2 , ... , k_m tels que : $k_1 + k_2 + ... + k_m = n$

Le nombre de termes du développement de $(x_1 + x_2 + \cdots + x_m)^n$ est égal à : $C_{m+n-1}^{m-1} = \frac{(m+n-1)!}{(m-1)!\ n!}$

Ecrivons une fonction *multinome(m,n)* qui calcule les coefficien ts multinomiaux : $C_n^{\vec{k}} = \binom{n}{\vec{k}} = \frac{n!}{k_1!\,k_2!\cdots k_m!}$

```
function multinome(m,n)
    if m == 0 then, return, end
    printf('\n Développement de (')
    for i = 1:m-1
        printf('x%i + ',i)
    end
    printf('x%i)^%i :',m,n)
    K = zeros(1,m)
    I = ones(1,m)
    T = ones(m,1) .*. [0:n]
    loop = %t
    test = %t
    while loop
        if test then
            for i = 1:m
                K(i) = T(i,I(i))
            end
            if sum(K) == n then
                c = factorial(n)
                for i = 1:m
                    c = c/factorial(K(i))
                end
                printf('\n %i ',c)
                for i = 1:m
                    if K(i) == 1 then
                        printf('x%i',i)
                    elseif K(i) > 1 then
                        printf('x%i^%i',i,K(i))
                    end
                end
            end
            p = 0
        else
            I(p) = 1
        end
        p = p + 1
        I(p) = I(p) + 1
        test = I(p) <= n+1
        loop = test | p < m
    end
endfunction
```

Par exemple, pour afficher le développement de $(x_1 + x_2 + x_3 + x_4)^7$ à la console Scilab, il suffit de saisir l'instruction :

multinome(4,7)

On obtient :

$1\ x_1^7$
$7\ x_1^6x_2$
$21\ x_1^5x_2^2$
$35\ x_1^4x_2^3$
$35\ x_1^3x_2^4$
$21\ x_1^2x_2^5$
$7\ x_1x_2^6$
$1\ x_2^7$
$7\ x_1^6x_3$
$42\ x_1^5x_2x_3$
$105\ x_1^4x_2^2x_3$
$140\ x_1^3x_2^3x_3$
$105\ x_1^2x_2^4x_3$
$42\ x_1x_2^5x_3$
$7\ x_2^6x_3$
$21\ x_1^5x_3^2$
$105\ x_1^4x_2x_3^2$
$210\ x_1^3x_2^2x_3^2$
$210\ x_1^2x_2^3x_3^2$
$105\ x_1x_2^4x_3^2$
$21\ x_2^5x_3^2$
$35\ x_1^4x_3^3$
$140\ x_1^3x_2x_3^3$
$210\ x_1^2x_2^2x_3^3$
$140\ x_1x_2^3x_3^3$
$35\ x_2^4x_3^3$
$35\ x_1^3x_3^4$
$105\ x_1^2x_2x_3^4$
$105\ x_1x_2^2x_3^4$
$35\ x_2^3x_3^4$
$21\ x_1^2x_3^5$
$42\ x_1x_2x_3^5$
$21\ x_2^2x_3^5$
$7\ x_1x_3^6$
$7\ x_2x_3^6$
$1\ x_3^7$
$7\ x_1^6x_4$
$42\ x_1^5x_2x_4$
$105\ x_1^4x_2^2x_4$
$140\ x_1^3x_2^3x_4$
$105\ x_1^2x_2^4x_4$
$42\ x_1x_2^5x_4$
$7\ x_2^6x_4$
$42\ x_1^5x_3x_4$
$210\ x_1^4x_2x_3x_4$
$420\ x_1^3x_2^2x_3x_4$
$420\ x_1^2x_2^3x_3x_4$
$210\ x_1x_2^4x_3x_4$
$42\ x_2^5x_3x_4$
$105\ x_1^4x_3^2x_4$
$420\ x_1^3x_2x_3^2x_4$
$630\ x_1^2x_2^2x_3^2x_4$
$420\ x_1x_2^3x_3^2x_4$
$105\ x_2^4x_3^2x_4$
$140\ x_1^3x_3^3x_4$
$420\ x_1^2x_2x_3^3x_4$
$420\ x_1x_2^2x_3^3x_4$
$140\ x_2^3x_3^3x_4$
$105\ x_1^2x_3^4x_4$
$210\ x_1x_2x_3^4x_4$
$105\ x_2^2x_3^4x_4$
$42\ x_1x_3^5x_4$
$42\ x_2x_3^5x_4$
$7\ x_3^6x_4$
$21\ x_1^5x_4^2$
$105\ x_1^4x_2x_4^2$
$210\ x_1^3x_2^2x_4^2$
$210\ x_1^2x_2^3x_4^2$
$105\ x_1x_2^4x_4^2$
$21\ x_2^5x_4^2$
$105\ x_1^4x_3x_4^2$
$420\ x_1^3x_2x_3x_4^2$
$630\ x_1^2x_2^2x_3x_4^2$
$420\ x_1x_2^3x_3x_4^2$
$105\ x_2^4x_3x_4^2$
$210\ x_1^3x_3^2x_4^2$
$630\ x_1^2x_2x_3^2x_4^2$
$630\ x_1x_2^2x_3^2x_4^2$
$210\ x_2^3x_3^2x_4^2$
$210\ x_1^2x_3^3x_4^2$
$420\ x_1x_2x_3^3x_4^2$
$210\ x_2^2x_3^3x_4^2$
$105\ x_1x_3^4x_4^2$
$105\ x_2x_3^4x_4^2$
$21\ x_3^5x_4^2$
$35\ x_1^4x_4^3$
$140\ x_1^3x_2x_4^3$
$210\ x_1^2x_2^2x_4^3$
$140\ x_1x_2^3x_4^3$
$35\ x_2^4x_4^3$
$140\ x_1^3x_3x_4^3$
$420\ x_1^2x_2x_3x_4^3$
$420\ x_1x_2^2x_3x_4^3$
$140\ x_2^3x_3x_4^3$
$210\ x_1^2x_3^2x_4^3$
$420\ x_1x_2x_3^2x_4^3$
$210\ x_2^2x_3^2x_4^3$
$140\ x_1x_3^3x_4^3$
$140\ x_2x_3^3x_4^3$
$35\ x_3^4x_4^3$
$35\ x_1^3x_4^4$
$105\ x_1^2x_2x_4^4$
$105\ x_1x_2^2x_4^4$
$35\ x_2^3x_4^4$
$105\ x_1^2x_3x_4^4$
$210\ x_1x_2x_3x_4^4$
$105\ x_2^2x_3x_4^4$
$105\ x_1x_3^2x_4^4$
$105\ x_2x_3^2x_4^4$
$35\ x_3^3x_4^4$
$21\ x_1^2x_4^5$
$42\ x_1x_2x_4^5$
$21\ x_2^2x_4^5$
$42\ x_1x_3x_4^5$
$42\ x_2x_3x_4^5$
$21\ x_3^2x_4^5$
$7\ x_1x_4^6$
$7\ x_2x_4^6$
$7\ x_3x_4^6$
$1\ x_4^7$

$$(x_1 + x_2 + x_3 + x_4)^7 = x_1^7 + 7x_1^6x_2 + 21x_1^5x_2^2 + 35x_1^4x_2^3 + 35x_1^3x_2^4 + 21x_1^2x_2^5 + 7x_1x_2^6 + x_2^7 + \ldots + 7x_2x_4^6 + 7x_3x_4^6 + x_4^7$$

Le développement de $(x_1 + x_2 + x_3 + x_4)^7$ contient $C_{10}^{3} = \binom{10}{3} = 120$ termes.

Note
La fonction *multinome(m,n)* adopte la notation xi^k pour désigner la variable x_i^k $(1 \le i \le m\ ,\ 2 \le k \le n)$; pour transformer les xi^k en x_i^k dans un texte, on peut utiliser la commande Remplacer de Word.

Mathématique & Informatique

Sur la représentation des graphes par des matrices

Un graphe simple non orienté est représenté par sa matrice d'adjacence A. Sous Scilab, A peut être définie :

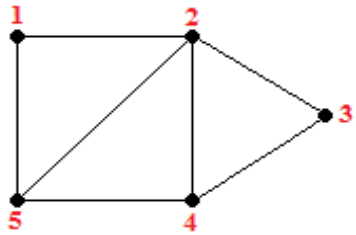
– soit directement par : $A = [a_{11}, \ldots, a_{1n} ; \ldots . ; a_{n1}, \ldots, a_{nn}]$ où les $a_{ij} \in \{0,1\}$

– soit par un ensemble d'instructions :

A = zeros(n,n)	→	A est une matrice n×n initialisée à zéro
$A(i , [v_1 , \ldots , v_p]) = 1$	→	les sommets v_j sont des voisins du sommet i
A = A + A'	→	A est additionnée à sa transposée A'

Notons que dans la deuxième définition on prend uniquement les sommets de numéros v_j strictement supérieurs au sommet de numéro i, c'est-à-dire $v_j > i$, ce qui revient à construire la partie triangulaire supérieure de la matrice d'adjacence. Comme la matrice d'adjacence a une diagonale nulle, il suffit d'additionner la matrice triangulaire supérieure obtenue et sa transposée triangulaire inférieure pour obtenir la matrice symétrique A.

Exemple




```
A = zeros(5,5) ;
A(1,[2,5]) = 1 ;
A(2,[3,4,5]) = 1 ;
A(3,4) = 1 ;
A(4,5) = 1 ;
A = A + A' ;
```

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Sur l'intérêt des matrices dans les graphes

Lorsqu'il est possible d'assimiler un graphe à une matrice, on peut effectuer facilement certaines opérations.
Par exemple, soit A la matrice d'adjacence d'un graphe simple non orienté G :

- Nombre de sommets du graphe G : *n = size(A,2)*
- Nombre d'arêtes du graphe G : *m = sum(A)/2*
- Voisins d'un sommet i : *V = find(A(i,:) == 1)*
- Degré d'un sommet i : *d = sum(A(i,:))*
- Suppression de l'arête (i,j) : *A(i,j) = 0 ; A(j,i) = 0*
- Supression du sommet i : *A = A([1:i-1,i+1:n],[1:i-1,i+1:n])*

Graphe adjoint et matrice adjointe

Le graphe adjoint (aussi appelé graphe-arête) d'un graphe G est le graphe G^* défini de la manière suivante :

- les sommets de G^* sont les arêtes de G
- deux sommets de G^* sont reliés si et seulement si les arêtes correspondantes dans G sont adjacentes

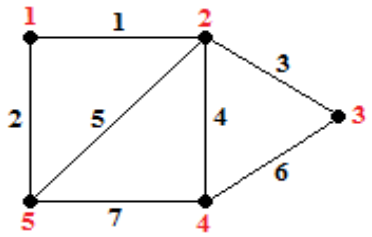
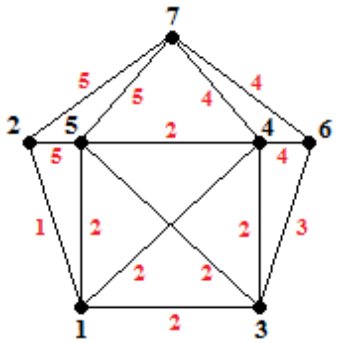
Soit G un graphe simple non orienté à n sommets et m arêtes et soit A sa matrice d'adjacence de dimension n × n.
Recherchons la matrice d'adjacence A^* de dimension m × m du graphe adjoint G^*.
Parcourons la matrice A ligne par ligne et attribuons aux arêtes rencontrées (A(i,j) = 1) un numéro croissant.
Construisons en parallèle la liste permettant d'effectuer la correspondance : arête p ↔ sommets (i_p , j_p)
L'élément $A^*(p,q)$ est égal à 1 si les arêtes n° p et n° q sont adjacentes dans G, c'est-à-dire si elles ont un sommet commun.
Autrement dit, si $p \equiv (i_p , j_p)$ et $q \equiv (i_q , j_q)$ alors : $A^*(p,q) = 1 \Leftrightarrow i_p = i_q$ ou $i_p = j_q$ ou $j_p = i_q$ ou $j_p = j_q$

Mathématique & Informatique

La fonction de calcul de la matrice d'adjacence adjointe B (= A^*) d'une matrice d'adjacence A est :

```
function B = adjointe(A)
    [n,n] = size(A)
    L = list()
    m = 0
    for i = 1:n
        for j = i+1:n
            if A(i,j) == 1 then
                m = m + 1
                L($+1) = [i,j]
            end
        end
    end
    B = zeros(m,m)
    for p = 1:m
        for q = p+1:m
            ip = L(p)(1) ; jp = L(p)(2)
            iq = L(q)(1) ; jq = L(q)(2)
            if ip == iq | ip == jq | jp == iq | jp == jq then
                B(p,q) = 1
            end
        end
    end
    B = B + B'
endfunction
```

Exemple

	*A = zeros(5,5) ;* *A(1,[2,5]) = 1 ;* *A(2,[3,4,5]) = 1 ;* *A(3,4) = 1 ;* *A(4,5) = 1 ;* *A = A + A' ;*	$A = \begin{pmatrix} 0&1&0&0&1 \\ 1&0&1&1&1 \\ 0&1&0&1&0 \\ 0&1&1&0&1 \\ 1&1&0&1&0 \end{pmatrix}$
	*B = adjointe(A) ;*	$B = \begin{pmatrix} 0&1&1&1&1&0&0 \\ 1&0&0&0&1&0&1 \\ 1&0&0&1&1&1&0 \\ 1&0&1&0&1&1&1 \\ 1&1&1&1&0&0&1 \\ 0&0&1&1&0&0&1 \\ 0&1&0&1&1&1&0 \end{pmatrix}$

Remarque
Si G est un polygone, ses n sommets de degré 2 et ses n arêtes peuvent être numérotés dans le sens horaire par un nombre croissant de 1 à n ; on a : $G^* = G$ et $A^* = A$

Coloration des graphes

Par définition, un graphe est k-coloriable si l'on peut attribuer à chaque sommet une couleur parmi k de sorte que 2 sommets reliés par une arête soient de couleur différente.
La plus petite valeur k est appelée nombre chromatique du graphe.

Soit un graphe G non orienté possédant n sommets.
Sans perte de généralité, on suppose que le graphe est sans boucle et sans arêtes multiples entre 2 sommets adjacents (s'il y a des boucles on les enlève et s'il y a des arêtes multiples on en garde une seule).
La matrice d'adjacence A du graphe simple G est une matrice de dimension n×n donnée par :

$$A(i,j) = \begin{cases} 1 \text{ si le sommet } x_i \text{ est adjacent au sommet } x_j \\ 0 \text{ sinon} \end{cases}$$

A est symétrique et sa diagonale est constituée de 0.
Convenons qu'une couleur est un entier compris entre 1 et k.
Considérons la matrice T de dimension n×k dont les lignes sont les couleurs possibles des sommets du graphe :

$$T(i,j) = j \quad (1 \le i \le n\ ,\ 1 \le j \le k)$$

Effectuons une recherche exhaustive à partir de T en testant la matrice de coloration C de dimension n×n définie par :

$C(i,i) = T(i,I(i))$ couleur du sommet x_i
$C(i,j) = A(i,j) * T(j,I(j))$ couleur du sommet x_j adjacent au sommet x_i

NB Dans le cas où x_j n'est pas adjacent à x_i , on a : $C(i,j) = 0$

Une k-coloration du graphe est obtenue si chaque ligne de C vérifie : $C(i,i) \neq C(i,j)\ ,\ \forall\ j > i$
Notons S le vecteur des couleurs des n sommets du graphe. On prend : $S(i) = C(i,i)\ ,\ 1 \le i \le n$

Ecrivons une fonction Scilab qui recherche une k-coloration des sommets d'un graphe dont on fournit la matrice d'adjacence A :

```
function [C,S] = coloration(A,k)
    [n,n] = size(A)
    C = zeros(n,n)
    I = ones(1,n)
    T = zeros(n,k)
    for i = 1:n
        for j = 1:k
            T(i,j) = j
        end
    end
    loop = %t
    test = %t
    while loop
        if test then
            solution = %t
            for i = 1:n
                C(i,i) = T(i,I(i))
                for j = i+1:n
                    C(i,j) = A(i,j)*T(j,I(j))
                    C(j,i) = C(i,j)
                    solution = solution & C(i,i) <> C(i,j)
                    if ~solution then, break, end
                end
                if ~solution then, break, end
            end
            if solution then, break, end
            p = 0
        else
            I(p) = 1
        end
        p = p + 1
        I(p) = I(p) + 1
        test = I(p) <= k
        loop = test | p < n
    end
    if solution then
        S = diag(C)
    else
        C = []
        S = []
    end
endfunction
```

Remarque

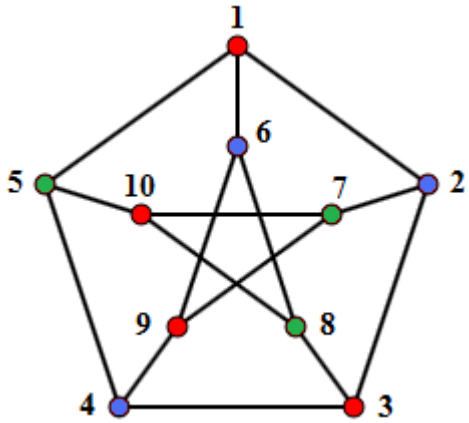
La fonction *coloration* ne recherche pas toutes les solutions possibles. Le programme s'arrête dès qu'une k-coloration du graphe est trouvée.

On peut écrire une autre fonction *coloration*, similaire à la précédente, mais qui a pour but de déterminer le nombre N de façons d'effectuer une k-coloration d'un graphe à partir de sa matrice d'adjacence A :

```
function N = coloration(A,k)
    N = 0
    [n,n] = size(A)
    I = ones(1,n)
    T = zeros(n,k)
    for i = 1:n
        for j = 1:k
            T(i,j) = j
        end
    end
    loop = %t
    test = %t
    while loop
        if test then
            solution = %t
            for i = 1:n
                for j = i+1:n
                    solution = solution & T(i,I(i)) <> A(i,j)*T(j,I(j))
                    if ~solution then, break, end
                end
                if ~solution then, break, end
            end
            if solution then, N = N + 1, end
            p = 0
        else
            I(p) = 1
        end
        p = p + 1
        I(p) = I(p) + 1
        test = I(p) <= k
        loop = test | p < n
    end
endfunction
```

Mathématique & Informatique

Graphe de **Petersen**



Matrice d'adjacence :

$$A=\begin{pmatrix}
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0\\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0\\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1\\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1\\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1\\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0
\end{pmatrix}$$

Programme de coloration :

```
A = [0 1 0 0 1 1 0 0 0 0 ; 1 0 1 0 0 0 1 0 0 0 ; 0 1 0 1 0 0 0 1 0 0 ; 0 0 1 0 1 0 0 0 1 0 ;1 0 0 1 0 0 0 0 0 1 ;
     1 0 0 0 0 0 0 1 1 0 ; 0 1 0 0 0 0 0 0 1 1 ; 0 0 1 0 0 1 0 0 0 1 ; 0 0 0 1 0 1 1 0 0 0 ; 0 0 0 0 1 0 1 1 0 0];
[C,S] = coloration(A,3) ;
disp(S)
```

Résultat : S = (1 3 1 3 2 3 2 2 1 1)

Choix des couleurs : 1 = 🔴 , 2 = 🟢 , 3 = 🔵

Remarque

Un graphe planaire est 4-coloriable (théorème des 4 couleurs).

Si l'on applique la méthode exhaustive avec k = 4 à un graphe quelconque et qu'il n'existe pas de solution, alors le graphe en question n'est pas planaire. Par contre, si l'on trouve une solution, cela ne signifie pas pour autant que le graphe est planaire. C'est le cas en particulier du graphe de **Petersen** qui est 3-coloriable, donc a fortiori 4-coloriable, mais qui n'est pas planaire puisqu'il possède un mineur isomorphe à K_5 (théorème de **Kuratowski**). Précisons : il suffit de contracter les arêtes 1-6, 2-7, 3-8, 4-9, 5-10 reliant l'étoile centrale au pentagone périphérique pour aboutir au graphe complet sur 5 sommets.

Mathématique & Informatique

Carte géographique

Pour colorer les pays d'une carte de géographie sans que 2 pays frontaliers aient la même couleur, il faut au moins 4 couleurs différentes comme le montre le petit schéma suivant symbolisant 1 pays entouré par 3 autres.
Le théorème des 4 couleurs affirme que ce nombre est suffisant.



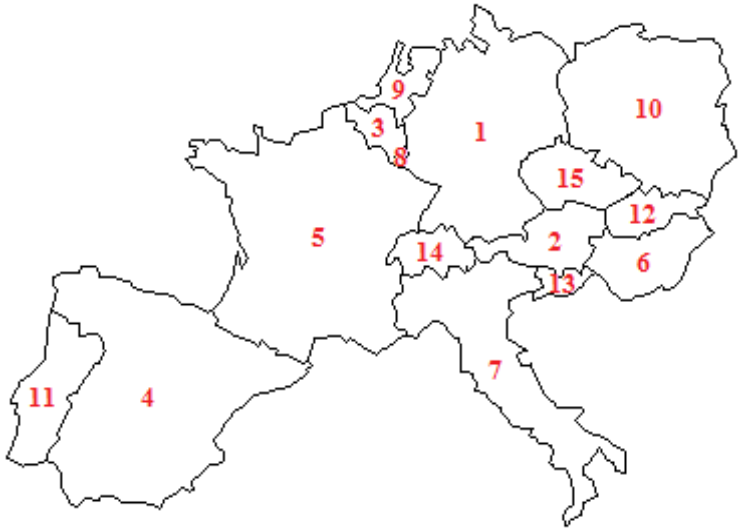
(le logo de Google Chrome



rappelle aussi le théorème des 4 couleurs)

A titre d'exemple, effectuons une 4-coloration d'une carte représentant 15 pays d'Europe.
On adopte une numérotation alphabétique des pays :

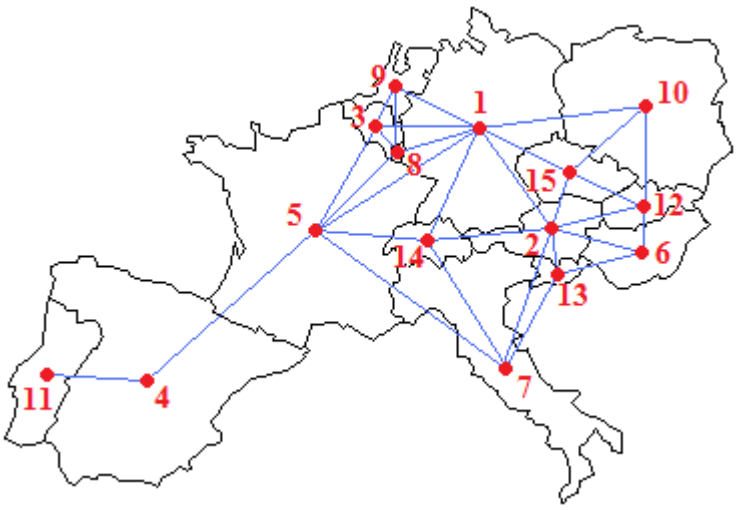
1. Allemagne
2. Autriche
3. Belgique
4. Espagne
5. France
6. Hongrie
7. Italie
8. Luxembourg
9. Pays-Bas
10. Pologne
11. Portugal
12. Slovaquie
13. Slovénie
14. Suisse
15. Tchéquie



Soit G le graphe défini à partir d'une carte de géographie de la manière suivante :

- les sommets sont les pays
- les arêtes sont les frontières entre les pays

Ainsi, une arête relie 2 sommets si les 2 pays correspondants ont une frontière commune.
Précisons que si 2 pays se touchent en 1 seul point sur la carte, on considère qu'ils n'ont pas de frontière commune.
A cette condition les arêtes ne se croisent pas, donc le graphe G est planaire et 4-coloriable.



On obtient le tableau de synthèse :

	N° du pays	N^{os} des pays frontaliers	N° de couleur
Allemagne	1	2 - 3 - 5 - 8 - 9 - 10 - 14 -15	2
Autriche	2	1 - 6 - 7 - 12 - 13 - 14 - 15	4
Belgique	3	1 - 5 - 8 - 9	4
Espagne	4	5 - 11	2
France	5	1 - 3 - 4 - 7 - 8 - 14	3
Hongrie	6	2 - 12 - 13	3
Italie	7	2 - 5 - 13 - 14	2
Luxembourg	8	1 - 3 - 5	1
Pays - Bas	9	1 - 3	1
Pologne	10	1 - 12 - 15	3
Portugal	11	4	1
Slovaquie	12	2 - 6 - 10 - 15	2
Slovénie	13	2 - 6 - 7	1
Suisse	14	1 - 2 - 5 - 7	1
Tchéquie	15	1 - 2 - 10 - 12	1

D'où la matrice d'adjacence :

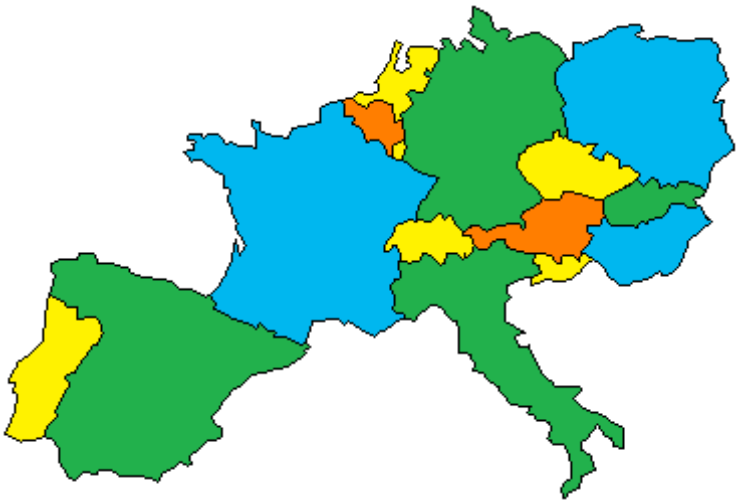
$$A=\begin{pmatrix}
0&1&1&0&1&0&0&1&1&1&0&0&0&1&1\\
1&0&0&0&0&1&1&0&0&0&0&1&1&1&1\\
1&0&0&0&1&0&0&1&1&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
1&0&1&1&0&0&1&1&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&1&1&0&0\\
0&1&0&0&1&0&0&0&0&0&0&0&1&1&0\\
1&0&1&0&1&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&1&0&0&1\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&0&1\\
0&1&0&0&0&1&1&0&0&0&0&0&0&0&0\\
1&1&0&0&1&0&1&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&1&0&1&0&0&0
\end{pmatrix}$$

La matrice d'adjacence A est une matrice 15×15 de 225 éléments.
La dernière colonne (N° de couleur) du tableau est le résultat de l'exécution du programme ci-dessous :

```
A = zeros(15,15) ;
A(1,[2,3,5,8,9,10,14,15]) = 1 ;
A(2,[6,7,12,13,14,15]) = 1 ;
A(3,[5,8,9]) = 1 ;
A(4,[5,11]) = 1 ;
A(5,[7,8,14]) = 1 ;
A(6,[12,13]) = 1 ;
A(7,[13,14]) = 1 ;
A(10,[12,15]) = 1 ;
A(12,[15]) = 1 ;
A = A + A' ;
[C,S] = coloration(A,4) ;
disp(S)
```

Choix des couleurs :

1
2
3
4



Notes

- Si l'on remplace l'instruction *[C,S] = coloration(A,4)* par *[C,S] = coloration(A,3)* on trouve S = [], ce qui signifie que le graphe G n'est pas 3-coloriable. Donc, le nombre chromatique du graphe G est égal à 4.
- On aurait pu simplifier les calculs en retirant les sommets 4 et 11 (Espagne et Portugal) et choisir leurs couleurs après avoir obtenu celle du sommet 5 (France). Les arêtes 4-5 et 4-11 sont des isthmes : elles ont la propriété de n'appartenir à aucun cycle et leur suppression augmente le nombre de composantes connexes du graphe.
- Pour représenter l'océan/la mer avec une couleur différente de tous les pays, on prendra une 5^{e} couleur.

Mathématique & Informatique

Recherche du nombre chromatique d'un graphe

La méthode exhaustive de k-coloration s'exécute en un temps exponentiel, ce qui limite son utilisation à des graphes de quelques dizaines de sommets avec les ordinateurs actuels. Dans le cas général, le problème du calcul du nombre chromatique χ d'un graphe est NP-complet ; il ne peut se résoudre en un temps polynomial. Toutefois, il existe plusieurs algorithmes qui permettent d'obtenir une k-coloration ($k \geq \chi$) en un temps quadratique, donc rapide, et très souvent on a même $k = \chi$. Le plus simple d'entre eux est l'algorithme "glouton" qui consiste à colorer les sommets les uns après les autres, dans un ordre quelconque, et sans jamais remettre en cause les choix déjà faits.

Soient :

- A la matrice d'adjacence n×n du graphe
- S le vecteur des couleurs des n sommets du graphe
- g le nombre chromatique glouton du graphe

On construit pas à pas la matrice n×n de coloration C définie par :

C(i,i) = k	couleur du sommet x_i ($1 \leq k \leq g$)
C(i,j) = A(i,j)*C(j,j)	couleur du sommet x_j adjacent au sommet x_i

L'algorithme glouton s'écrit :

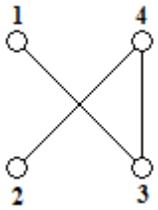
```
C = A
g = 0
pour i = 1 à n
|    ① Attribuer au sommet x_i une couleur différente de celles déjà connues de ses voisins x_j d'indices inférieurs :
|          C(i,i) = la plus petite valeur k comprise entre 1 et g qui n'apparaît pas dans les termes C(i,j) > 0 avec j < i
|          Si cette valeur k n'existe pas alors augmenter le nombre chromatique de 1 et prendre C(i,i) = g
|    ② La couleur du sommet x_i étant choisie, mettre à jour les termes non nuls de la colonne i de la matrice C :
|          si C(p,i) ≠ 0 alors C(p,i) = C(i,i) , 1 ≤ p ≤ n
fin
S = diagonale(C)
```

On déduit la fonction Scilab correspondante :

```
function [C,S,g] = glouton(A)
    C = A
    g = 0
    [n,n] = size(A)
    for i = 1:n
        couleur = [1:g]
        for j = 1:i-1
            if C(i,j) <> 0 then, couleur(C(i,j)) = 0, end
        end
       choix = min(find(couleur > 0))
       if choix <> [] then
            C(i,i) = choix
       else
            g = g + 1
            C(i,i) = g
       end
       for p = 1:n
            if C(p,i) <> 0 then, C(p,i) = C(i,i), end
       end
    end
    S = diag(C)
endfunction
```

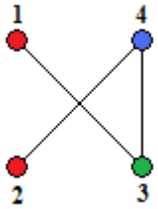
Remarque importante

L'algorithme glouton ne donne pas toujours le nombre chromatique d'un graphe comme le montre l'exemple qui suit.
Soit le graphe G représenté par :

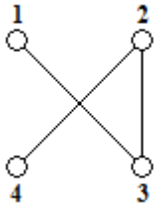


Sa matrice d'adjacence est : $A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$

L'instruction : *[C,S,g] = glouton([0,0,1,0;0,0,0,1;1,0,0,1;0,1,1,0])* donne : g = 3 et S = [1, 1, 2, 3]
Avec les couleurs : 1 = 🔴 , 2 = 🟢 , 3 = 🔵 , on obtient :

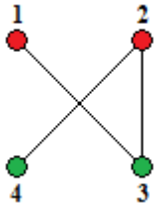


Numérotons le graphe différemment (dans le sens horaire) :



On a cette fois : $A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

L'instruction : *[C,S,g] = glouton([0,0,1,0;0,0,1,1;1,1,0,0,;0,1,0,0])* donne : g = 2 et S = [1, 1, 2, 2]
Toujours avec les mêmes couleurs : 1 = 🔴 , 2 = 🟢 , 3 = 🔵 , on obtient :

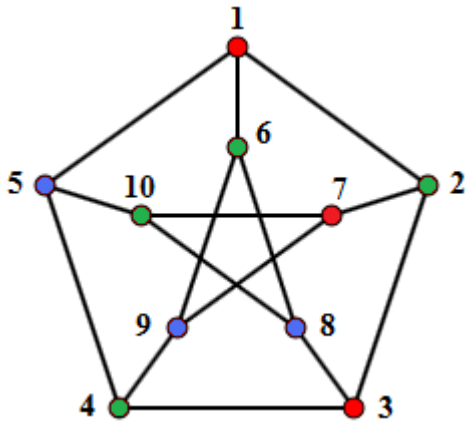


Conclusion :
Le graphe G est 2-coloriable, son nombre chromatique est $\chi = 2$. Avec la première numérotation on a trouvé $g > \chi$ et avec la seconde $g = \chi$. Dans le cas général : $g \geq \chi$, le nombre chromatique glouton est supérieur ou égal au nombre chromatique. La valeur de g dépend de la numérotation des sommets du graphe, alors que χ est une valeur intrinsèque au graphe, donc indépendante de toute numérotation.
⇨ **http://www-sop.inria.fr/maestro/MASTER-RSD/html/2006-07/marie-aste.pdf**

Application

Reprenons l'exemple du graphe de **Petersen** et celui de la carte de l'Europe en utilsant la fonction *glouton* au lieu de la fonction *coloration*.
Dans le premier cas, on obtient : g = 3 et S = [1 , 2 , 1 , 2 , 3 , 2 , 1 , 3 , 3 , 2]

Choix des couleurs : 1 = ● , 2 = ● , 3 = ●



Dans le deuxième cas, on obtient : g = 4 et S = [1 , 2 , 2 , 1 , 3 , 1 , 1 , 4 , 3 , 2 , 2 , 3 , 3 , 4 , 4]

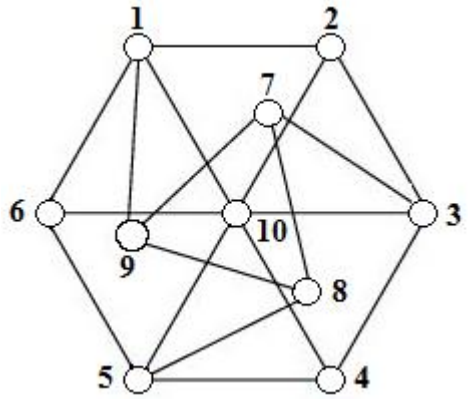
Choix des couleurs : 1 ● , 2 ● , 3 ● , 4 ●



Conclusion : dans les 2 cas l'algorithme glouton fournit la bonne valeur du nombre chromatique χ du graphe et une χ-coloration de ses sommets.

Mathématique & Informatique

Graphe de **Golomb**



Matrice d'adjacence :

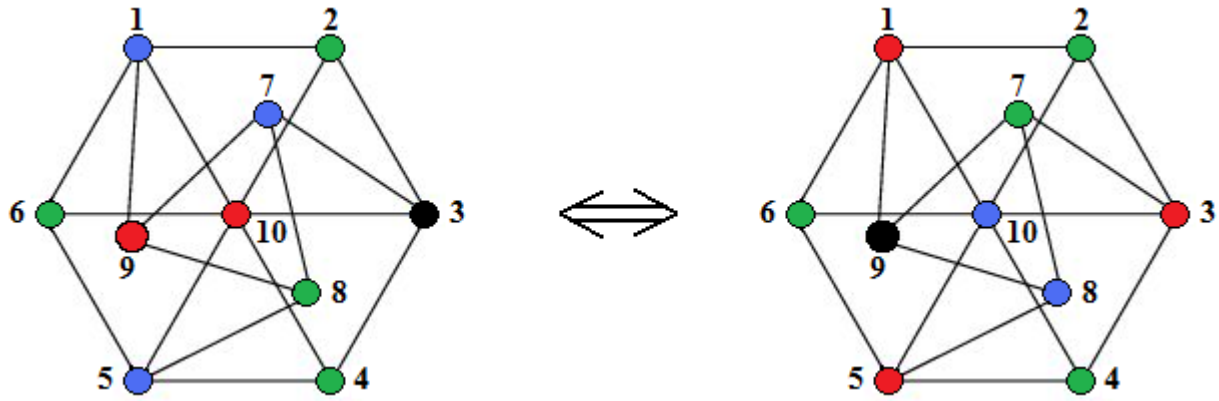
A = zeros(10,10) ;
A(1,[2,6,9,10]) = 1 ;
A(2,[3,10]) = 1 ;
A(3,[4,7,10]) = 1 ;
A(4,[5,10]) = 1 ;
A(5,[6,8,10]) = 1 ;
A(6,[10]) = 1 ;
A(7,[8,9]) = 1 ;
A(8,[9]) = 1 ;
A = A + A' ;

Nombre chromatique :

- *[C,S] = coloration(A,3)* donne : S = []
- *[C,S] = coloration(A,4)* donne : S = [3 , 2 , 4 , 2 , 3 , 2 , 3 , 2 , 1 , 1]
- *[C,S,g] = glouton(A)* donne : S = [1 , 2 , 1 , 2 , 1 , 2 , 2 , 3 , 4 , 3] et g = 4

Donc : $\chi = g = 4$

Choix des couleurs : 1 = ● , 2 = ● , 3 = ● , 4 = ●

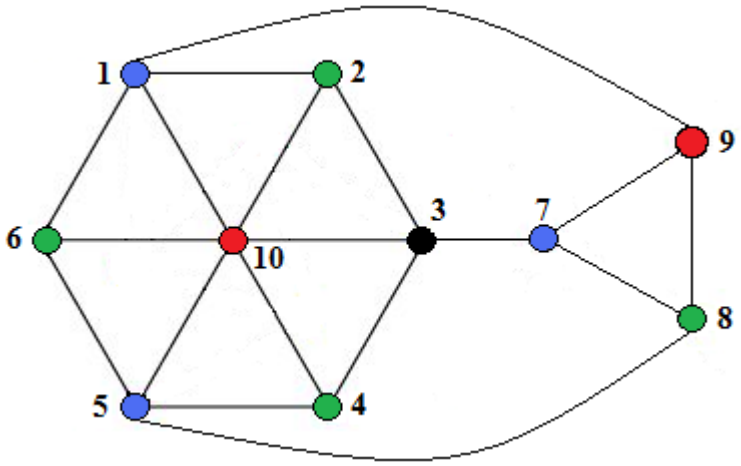


Ces 4-colorations ne sont que 2 des multiples possibilités de colorer ce graphe. Si l'on exécute l'instruction *N = coloration(A,4)* on trouve N = 2280 façons d'attribuer 4 couleurs aux sommets du graphe de **Golomb**.

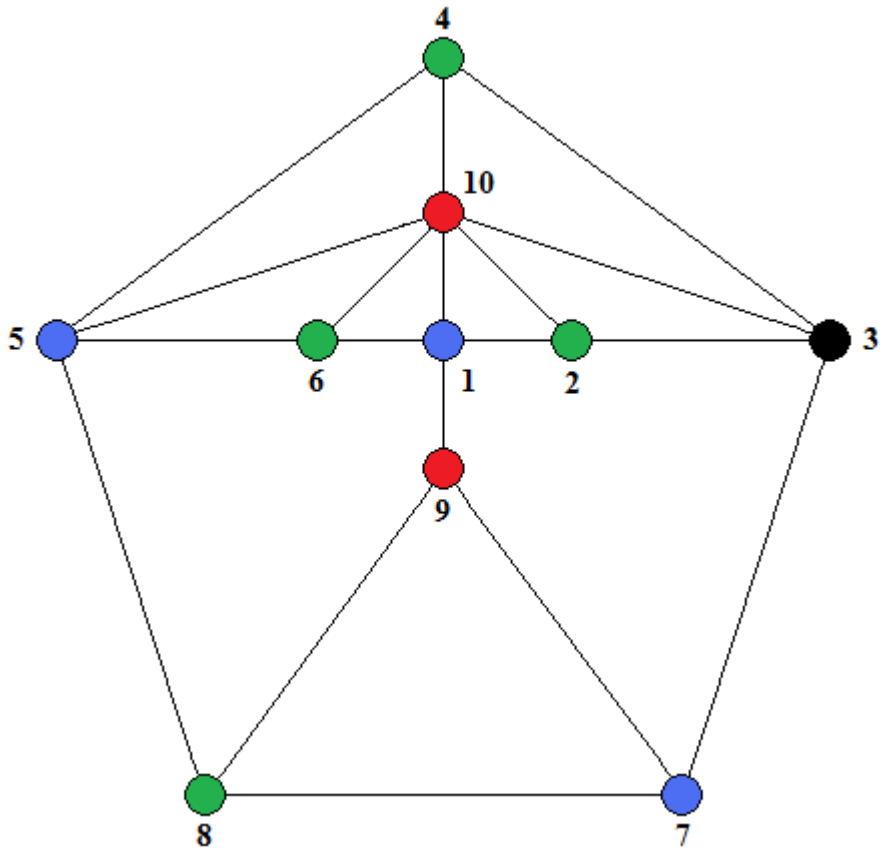
Note : Le graphe de **Golomb** peut être dessiné de telle sorte que toutes ses arêtes aient la même longueur.

Attention : les petits dessins bien utiles sont parfois trompeurs !

Par définition, un graphe planaire est un graphe qui peut être représenté sans que ses arêtes se rencontrent en dehors de leurs extrémités.
Le graphe de **Golomb** précédent semble non planaire puisque plusieurs arêtes se croisent sur le schéma.
Mais il n'en n'est rien car l'on peut extraire le triangle central pour obtenir une représentation planaire :



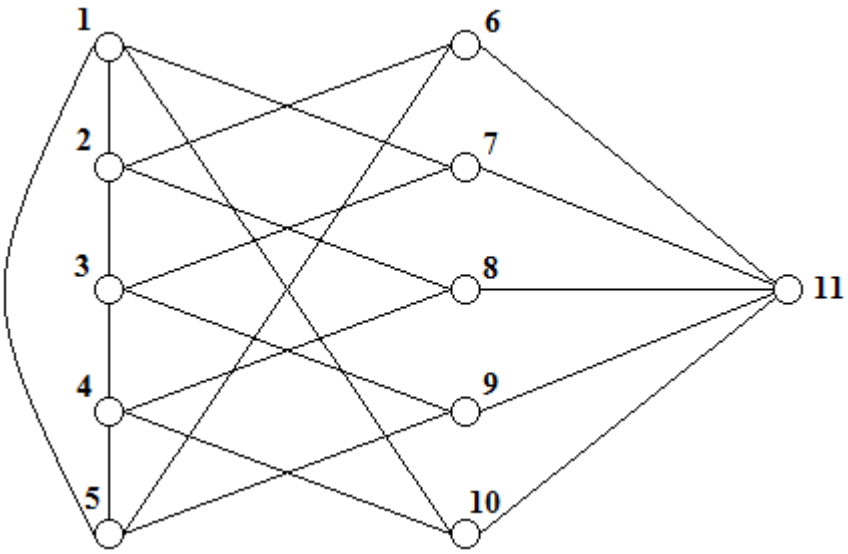
On peut redessiner l'ensemble sous la forme d'un pentagone par exemple :



En conclusion, la représentation visuelle d'un graphe n'est pas le plus important, ce sont les relations liant les sommets entre eux qui comptent.

Mathématique & Informatique

Graphe de **Mycielski**



Matrice d'adjacence :

```
A = zeros(11,11) ;
A(1,[2,5,7,10]) = 1 ;
A(2,[3,6,8]) = 1 ;
A(3,[4,7,9]) = 1 ;
A(4,[5,8,10]) = 1 ;
A(5,[6,9]) = 1 ;
A(6,[11]) = 1 ;
A(7,[11]) = 1 ;
A(8,[11]) = 1 ;
A(9,[11]) = 1 ;
A(10,[11]) = 1 ;
A = A + A' ;
```

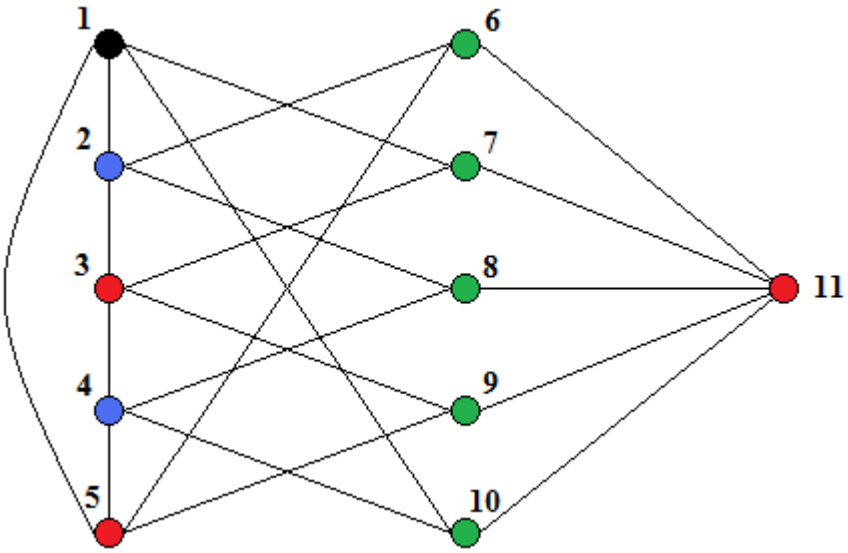
Programme :

```
disp('---   Graphe de Mycielski   ---')
[C,S,g] = glouton(A) ;
disp(g,'Nombre chromatique glouton')
disp(S,'Couleurs des sommets')
disp('---   3 coloration     ---')
[C,S] = coloration(A,3) ;
disp(S)
disp('---   4 coloration     ---')
[C,S] = coloration(A,4) ;
disp(S)
```

Mathématique & Informatique

Nombre chromatique χ :

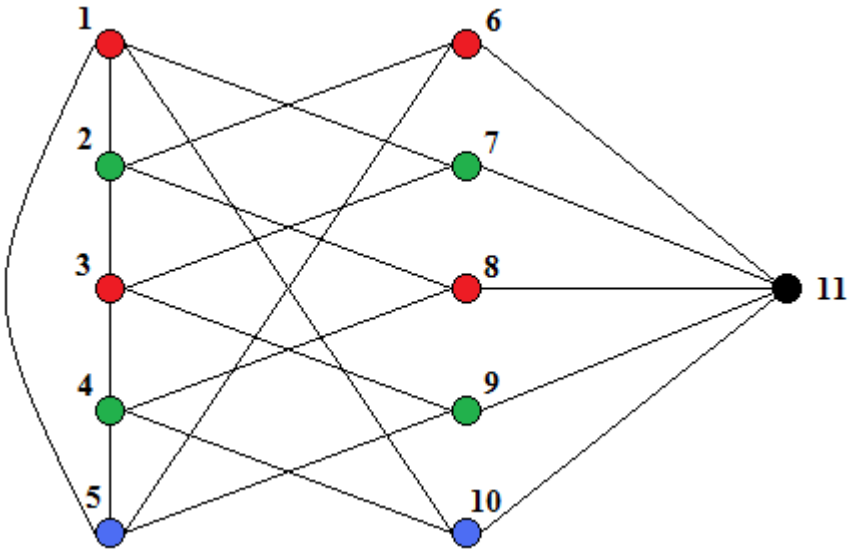
- *coloration(A,3)* donne : S = []
- *coloration(A,4)* donne : S = [4 , 3 , 1 , 3 , 1 , 2 , 2 , 2 , 2 , 2 , 1]
  Choix des couleurs : 1 = ● , 2 = ● , 3 = ● , 4 = ●



Donc : $\chi = 4$

Nombre chromatique glouton g :

- *glouton(A)* donne : g = 4 et S = [1 , 2 , 1 , 2 , 3 , 1 , 2 , 1 , 2 , 3 , 4]
  Choix des couleurs : 1 = ● , 2 = ● , 3 = ● , 4 = ●



Donc, ici encore g = χ, l'algorithme glouton trouve une solution optimale (différente de la précédente)

Note : Les graphes de **Mycielski** sont notés M_k (dans l'exemple ci-dessus on a représenté M_4).

Matrice de **Mycielski**

Le graphe de **Mycielski** M_k , $k \in \{2, 3, 4, ...\}$, est construit de manière à ne contenir aucun triangle (c'est-à-dire aucun sous-graphe de 3 sommets reliés 2 à 2) et afin que son nombre chromatique soit égal à k, donc arbitrairement grand.
Ces graphes sont souvent utilisés pour tester les algorithmes de coloration.
On appellera matrice de **Mycielski** la matrice d'adjacence A_k du graphe M_k.
A_k est une matrice carrée qui se calcule par récurrence.

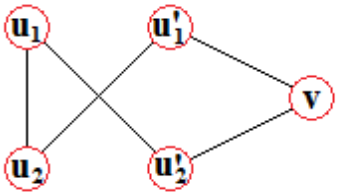
Supposons que le graphe M_k soit connu et notons $u_1 \dots u_n$ ses n sommets. Par définition, le graphe M_{k+1} possède $2n + 1$ sommets notés dans l'ordre $u_1 \dots u_n\ u'_1 \dots u'_n\ v$ et contient les arêtes suivantes :

- les arêtes du graphe M_k entre les sommets u_i
- s'il existe une arête entre u_i et u_j , on ajoute une arête entre u'_i et u_j
- une arête entre chaque sommet u'_i et le sommet v

Ainsi, les 2 premiers graphes de **Mycielski** et leurs matrices associées sont :

$M_2 =$ 

$\Longrightarrow \quad A_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

$M_3 =$ 

$\Longrightarrow \quad A_3 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$

On montre par récurrence sur k que si M_k n'a pas de triangle alors M_{k+1} n'en a pas non plus et que si M_k est k-coloriable alors M_{k+1} est $(k+1)$-coloriable.
Le nombre n de sommets du graphe M_k est le terme général d'une suite arithmético-géométrique $n_k = 2n_{k-1} + 1$, $k \geq 3$, avec $n_2 = 2$. D'où : $n = 2^{k-2}.2 + [(2^{k-2} - 1)/(2 - 1)].1 = 3.2^{k-2} - 1$

Si l'on connait les éléments non nuls $a_k(i,j)$ de la matrice A_k de dimension (n,n), alors les éléments non nuls $a_{k+1}(i,j)$ de la matrice A_{k+1} de dimension $(2n+1,2n+1)$ sont donnés par :

$\forall\ (i,j) \in [1,n]^2$
si $a_k(i,j) = 1$ alors
| $a_{k+1}(i,j) = 1$
| $a_{k+1}(n+i,j) = 1$
| $a_{k+1}(j,n+i) = 1$
fin
$a_{k+1}(n+i,2n+1) = 1$
$a_{k+1}(2n+1,n+i) = 1$

On déduit la fonction de calcul de la matrice d'adjacence *A* du graphe de **Mycielski** d'ordre *k* :

```
function A = Mycielski(k)
    if k < 2 then, A = [], return, end
    A = [0,1;1,0]
    n = 2
    while k > 2
        B = zeros(2*n+1,2*n+1)
        for i = 1:n
            for j = 1:n
                if A(i,j) == 1 then
                    B(i,j) = 1
                    B(n+i,j) = 1
                    B(j,n+i) = 1
                end
            end
            B(n+i,2*n+1) = 1
            B(2*n+1,n+i) = 1
        end
        A = B
        n = 2*n + 1
        k = k - 1
    end
endfunction
```

En appliquant l'algorithme glouton précédemment décrit aux graphes M_2 à M_{13} on obtient à chaque fois g = *k*. L'algorithme glouton donne donc rapidement la bonne valeur du nombre chromatique et une *k*-coloration possible des sommets du graphe M_k.

Exemple : une 10-coloration du graphe M_{10}

Les numéros de couleur des 767 sommets du graphe sont dans l'ordre :
1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 1, 2,
1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 7, 1, 2, 1,
2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 1, 2, 1, 2, 3,
1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 1, 2, 3,
1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 1, 2, 1, 2, 3, 1, 2,
1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 7, 1, 2, 1, 2, 3, 1, 2, 1,
2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3,
4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 1, 2, 3, 1, 2, 1, 2,
3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4,
1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 7, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1,
2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1,
2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1,
2, 3, 1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3,
1, 2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 7, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1,
2, 1, 2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1,
2, 3, 4, 5, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Remarque

La matrice d'adjacence d'un graphe simple non orienté est une matrice carrée symétrique constituée de 0 et de 1 avec une diagonale composée uniquement de 0.
Le nombre *n* de sommets d'un graphe est égal au nombre de lignes, ou de colonnes, de sa matrice d'adjacence.
Le nombre *m* d'arêtes d'un graphe est égal à la somme divisée par 2 de tous les éléments de sa matrice d'adjacence.
Par suite, si l'on note *A* la matrice d'adjacence, les nombres *n* et *m* se calculent par les instructions :

```
n = size(A,2)
m = sum(A)/2
```

Dans le cas des matrices A_k de **Mycielski**, on trouve :

k	*n*	*m*
2	2	1
3	5	5
4	11	20
5	23	71
6	47	236
7	95	755
8	191	2360
9	383	7271
10	767	22196

Notons n_k le nombre de sommets du graphe M_k et m_k son nombre d'arêtes.
Par construction du graphe de **Mycielski** d'ordre *k*, on a les relations de récurrence :

$$\begin{cases} n_k = 2n_{k-1} + 1 & \text{pour } k \geq 3 \text{ avec } n_2 = 2 \\ m_k = 3m_{k-1} + n_{k-1} & \text{pour } k \geq 3 \text{ avec } m_2 = 1 \end{cases}$$

Déduisons de ces formules les expressions de n_k et m_k en fonction de *k*.
Pour n_k , il s'agit d'une progression arithmético-géométrique et on a vu que :

$$n_k = 3.2^{k-2} - 1$$

Pour m_k , on établit facilement que :

$$m_k = 3^{k-2} + \sum_{i=2}^{k-1} 3^{k-i-1} n_i$$

En exprimant la valeur des n_i , on aboutit à :

$$m_k = \frac{7.3^{k-2} - 3.2^{k-1} + 1}{2}$$

Représentation des graphes de **Mycielski**

Ici, on dessine le graphe M_k pour $k = 2, 3, 4, 5$ (au-delà de $k = 5$ le nombre d'arêtes est important et le dessin est trop chargé à petite échelle).
Adoptons une représentation circulaire du graphe M_k où ses n_k sommets forment un polygone régulier à n_k cotés inscrit dans le cercle unité. Ainsi, l'affixe du sommet s_p ($p = 1, 2, \dots, n_k$) est :

$$s_p = e^{i\left[\frac{\pi}{2} + (p-1)\frac{2\pi}{n_k}\right]} \qquad p = 1, 2, \cdots, n_k$$

Soit A_k la matrice de **Mycielski** d'ordre k. Deux sommets s_p et s_q sont reliés par une arête si et seulement si :

$A_k(p,q) = 1$ $\qquad p = 1, 2, \dots, n_k$; $q = p+1, \dots, n_k$

Les sommets seront numérotés dans le sens trigonométrique en partant du sommet d'affixe $s_1 = i$.
Une k-coloration du graphe sera effectuée avec les couleurs : rouge = 1, vert = 2, bleu = 3, noir = 4, jaune = 5

La fonction graphique est la suivante :

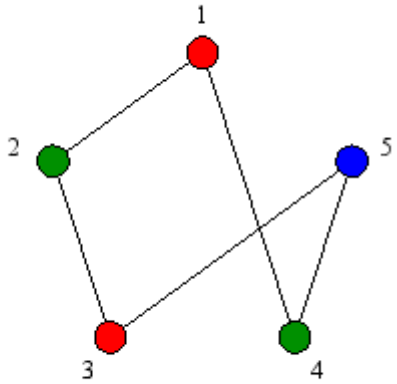
```
function Mycielski_graphic(k)
    if k < 2 | k > 5 then, return, end
    Color = [5,13,2,1,7]
    r = 2^(5-k)*0.025
    d  = (6-k)*0.05
    R = 1 + r + 2*d
    plot2d(%inf,%inf,frameflag = 1,rect = [-1,-1,1,1]*R,axesflag = 0)
    a = gca() ; a.isoview = 'on' ;
    A = Mycielski(k)
    [C,S,g] = glouton(A)
    n = size(A,2)
    for p = 1:n
        for q = p+1:n
            if A(p,q) == 1 then
                sp = exp(%i*(%pi/2 + (p-1)*2*%pi/n))
                sq = exp(%i*(%pi/2 + (q-1)*2*%pi/n))
                edge = [sp;sq]
                xsegs(real(edge),imag(edge),1)
            end
        end
    end
    for p = 1:n
        sp = exp(%i*(%pi/2 + (p-1)*2*%pi/n))
        xfarc(real(sp)-r,imag(sp)+r,2*r,2*r,0,64*360)
        e = gce() ; e.background = Color(S(p))
        xarc(real(sp)-r,imag(sp)+r,2*r,2*r,0,64*360)
        e = gce() ; e.foreground = 1
        z = (1+r+d)*sp
        xstringb(real(z)-d,imag(z)-d,string(p),2*d,2*d)
        e = gce() ; e.font_style = 2 ; e.font_size = 2
    end
endfunction
```

Mathématique & Informatique

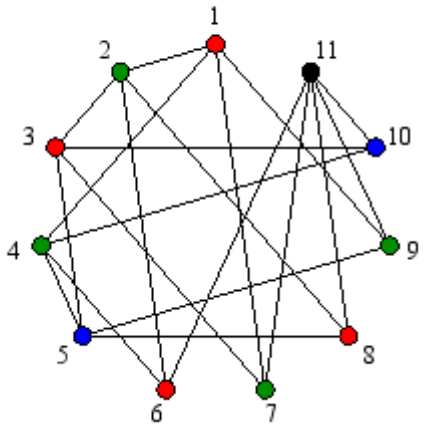
M_2



M_3

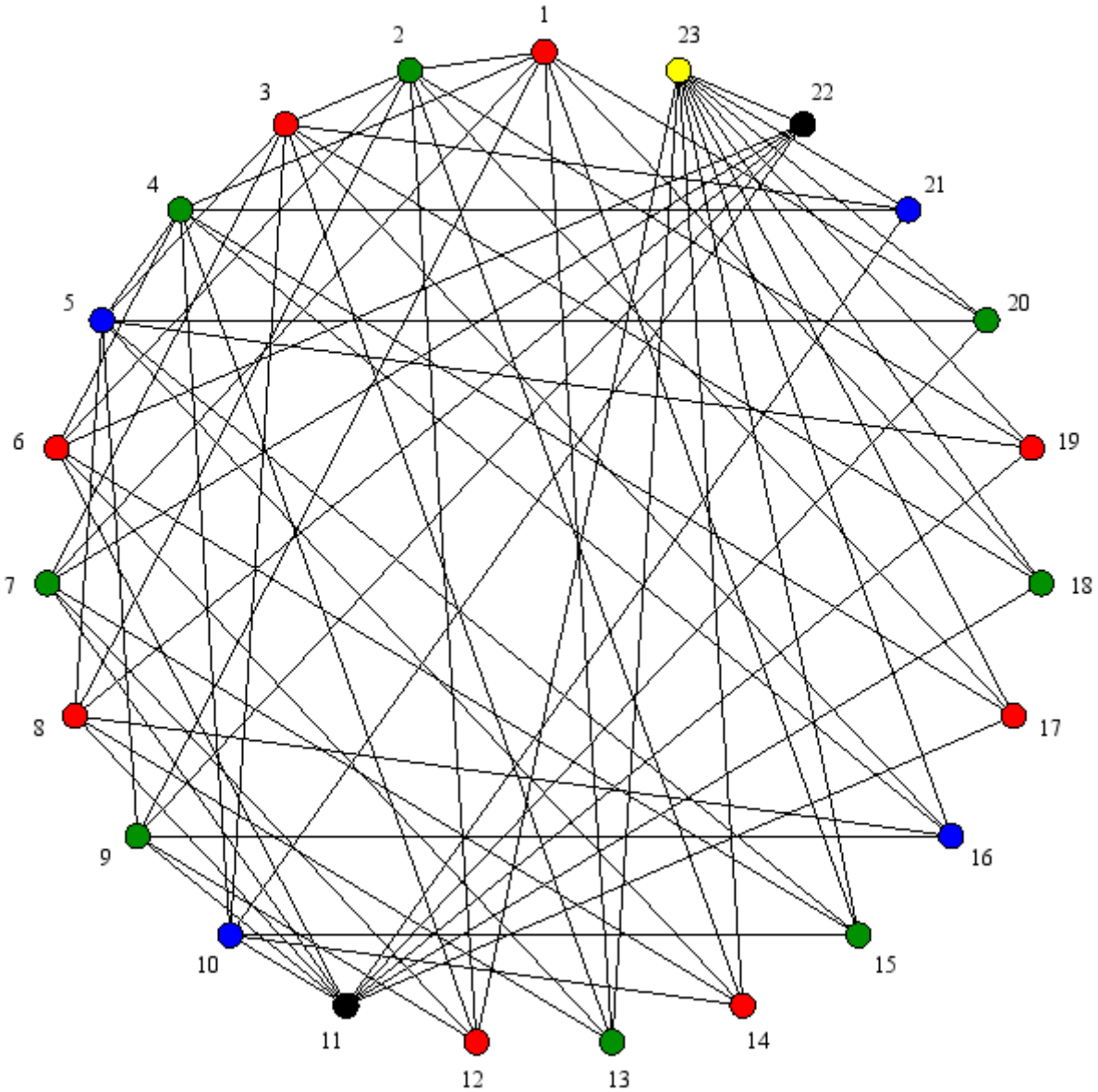


M_4



Mathématique & Informatique

M_5



Parcours exhaustif

Cette fonction recherche le plus court chemin (sans cycle) d'un sommet numéro i vers un sommet numéro j dans un graphe G dont la matrice d'adjacence est A :

```
function M = chemin(A,i,j)
    function parcours(S,i)
        global M
        for k = 1:length(S)
            if S(k) == i then, return, end
        end
        S($+1) = i
        if i == j then
            if length(S) < length(M) | M == list() then
                M = S
            end
        else
            for k = 1:n
                if A(i,k) == 1 then
                    parcours(S,k)
                end
            end
        end
    endfunction
    global M
    M = list()
    [n,n] = size(A)
    parcours(list(),i)
endfunction
```

Après l'exécution, la variable M contient la liste des numéros des sommets du plus court chemin de i vers j.
Il existe des procédés plus performants pour trouver le plus court chemin dans un graphe – algorithme de **Dijkstra**, algorithme de **Ford** (**Bell-End-Ford**) – mais le but ici est d'écrire un algorithme général recherchant tous les chemins d'un sommet à un autre afin de retenir celui, ou ceux, qui vérifient des conditions particulières (pas uniquement la plus courte distance globale).

Remarque

La fonction *chemin* s'applique aussi bien aux graphes simples non orientés qu'aux graphes simples orientés.
Dans le premier cas la matrice d'adjacence est symétrique, dans le second elle ne l'est pas (en général).
Pour éviter le passage inutile de données en paramètres lors des appels récursifs et le risque de saturer la pile, on utilise l'instruction *global* de Scilab : *global M*
Il faut écrire cette déclaration à la fois dans le corps du programme appelant (fonction *chemin*) et dans le corps du programme appelé (fonction *parcours*), sans oublier d'initialiser la variable globale : *M = list()*

Cycle hamiltonien

Un cycle hamiltonien est un circuit qui passe une seule fois par tous les sommets d'un graphe.
Partons du sommet $x_0 = 1$ et prenons un de ses voisins x_j . On teste s'il existe un chemin de x_0 vers x_j qui passe par tous les sommets du graphe, ce qui revient à vérifier si la liste S des sommets visités a une longueur égale au nombre de sommets du graphe. Lorsque x_j aura décrit l'ensemble des voisins du sommet x_0 , tous les cycles hamiltoniens possibles auront été examinés. Notons A la matrice d'adjacence de dimension $n \times n$ d'un graphe G à n sommets.
Un cycle hamiltonien $H = \{h_1 , ... , h_n\}$ est une permutation de l'ensemble $\{1 , ... , n\}$ des numéros des sommets.
Les arêtes du cycle H sont les couples (h_k , h_{k+1}), $1 \le k \le n-1$, plus le couple (h_n , h_1) pour fermer le circuit de n arêtes.
La fonction de recherche d'un cycle hamiltonien H dans un graphe G de matrice d'adjacence A est :

```
function H = Hamilton(A)
    function parcours(S,i)
        global H
        if H <> list() then, return, end
        for k = 1:length(S)
            if S(k) == i then, return, end
        end
        S($+1) = i
        if i == j then
            if length(S) == n then
                H = S
                return
            end
        else
            for k = 1:n
                if A(i,k) == 1 then
                    parcours(S,k)
                    if H <> list() then, return, end
                end
            end
        end
    endfunction
    global H
    H = list()
    [n,n] = size(A)
    i = 1
    for j = 1:n
        if A(i,j) == 1 then
            parcours(list(),i)
            if H <> list() then, return, end
        end
    end
endfunction
```

Bien que cet algorithme donne toujours le résultat, il est inadapté aux grands graphes en raison du temps de calcul excessif. Utilisons le, par exemple, pour le graphe de **Golomb** :

```
A = zeros(10,10) ;
A(1,[2,6,9,10]) = 1 ; A(2,[3,10]) = 1 ; A(3,[4,7,10]) = 1 ; A(4,[5,10]) = 1 ;
A(5,[6,8,10]) = 1 ; A(6,[10]) = 1 ; A(7,[8,9]) = 1 ; A(8,[9]) = 1 ;
A = A + A' ;
H = Hamilton(A) ;
disp(H,'Cycle hamiltonien =')
```

Le résultat est : Cycle hamiltonien = 1 6 5 8 9 7 3 4 10 2

Ecrivons également une fonction *Hamilton* qui compte le nombre N de cycles hamiltoniens dans un graphe G de matrice d'adjacence A :

```
function N = Hamilton(A)
    function parcours(S,i)
        global N
        for k = 1:length(S)
            if S(k) == i then, return, end
        end
        S($+1) = i
        if i == j then
            if length(S) == n then
                N = N + 1
            end
        else
            for k = 1:n
                if A(i,k) == 1 then
                    parcours(S,k)
                end
            end
        end
    endfunction
    global N
    N = 0
    [n,n] = size(A)
    i = 1
    for j = 1:n
        if A(i,j) == 1 then
            parcours(list(),i)
        end
    end
endfunction
```

Remarques

- Dans le corps de la fonction *N = Hamilton(A)*, si l'on prend un autre sommet de départ que le n°1, c'est-à-dire si l'on remplace l'instruction i = 1 par i = k ($1 < k \leq n$), le résultat est le même.

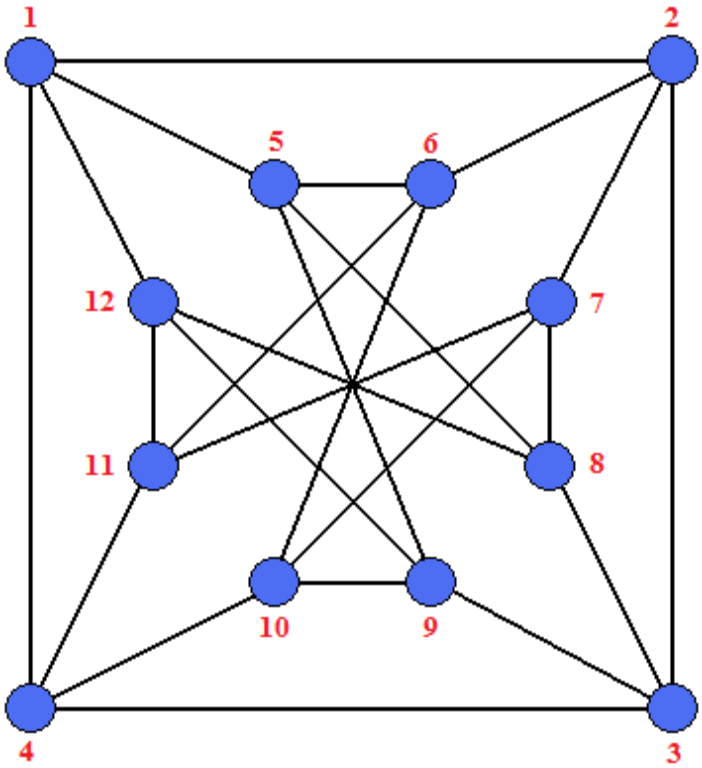
- La fonction *H = Hamilton(A)* se contente de trouver un cycle hamiltonien quelconque, c'est pourquoi on quitte la procédure dès que $H \neq \emptyset$. Dans le cas où l'on souhaite obtenir un cycle hamiltonien particulier, au sens d'un autre critère, le test $H \neq \emptyset$ ne sera pas effectué et tous les voisins du sommet i seront examinés.

- Les 2 boucles suivantes sont équivalentes :

```
for j = 1:n                    V = find(A(i,:) == 1)
|    if A(i,j) == 1 then      for j = V
|    |                        |
|    |                        |
|    end                      |
end                            end
```

Mathématique & Informatique

On peut trouver plusieurs cycles hamiltoniens dans le même graphe.
Prenons le graphe de **Chvátal** :



A = zeros(12,12) ;
A(1,[2,4,5,12]) = 1 ; A(2,[3,6,7]) = 1 ; A(3,[4,8,9]) = 1 ; A(4,[10,11]) = 1 ;
A(5,[6,8,9]) = 1 ; A(6,[10,11]) = 1 ; A(7,[8,10,11]) = 1 ; A(8,[12]) = 1 ; A(9,[10,12]) = 1 ; A(11,[12]) = 1 ;
A = A + A' ;

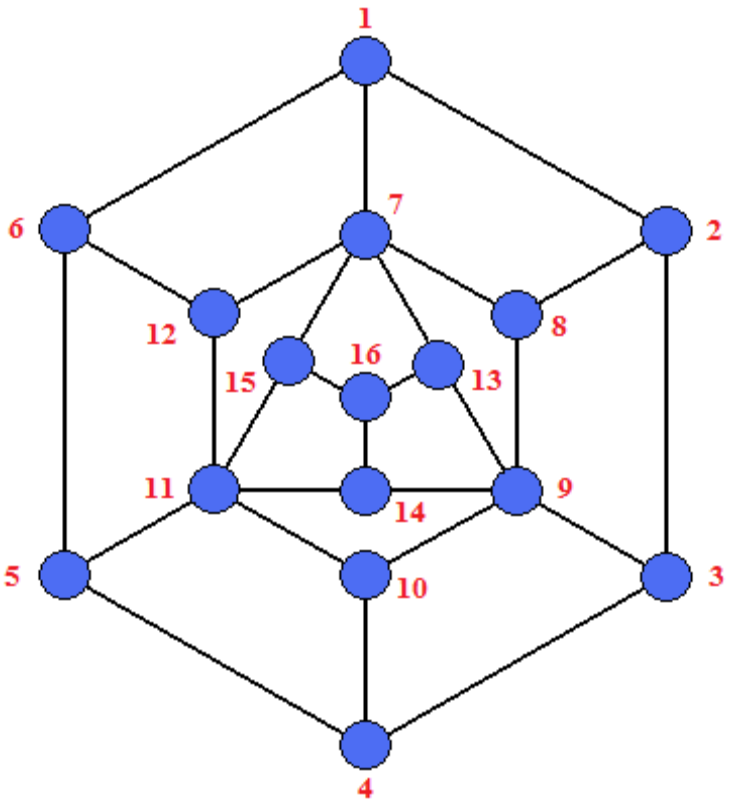
Calculons le nombre de cycles hamiltoniens :

```
N = Hamilton(A) ;
disp(N,'Nombre de cycles hamiltoniens =')
```

On obtient : Nombre de cycles hamiltoniens = 370

Donc, le graphe de **Chvátal** possède 370 cycles hamiltoniens.

Il n'existe pas toujours de cycle hamiltonien dans un graphe. Soit, par exemple, le graphe planaire :




```
A = zeros(16,16) ;
A(1,[2,6,7]) = 1 ; A(2,[3,8]) = 1 ; A(3,[4,9]) = 1 ; A(4,[5,10]) = 1 ; A(5,[6,11]) = 1 ;
A(7,[8,12]) = 1 ; A(8,[9]) = 1 ; A(9,[10,13,14]) = 1 ; A(10,[11]) = 1 ; A(11,[12,14,15]) = 1 ;
A(13,[16]) = 1 ; A(14,[16]) = 1 ; A(15,[16]) = 1 ;
A = A + A' ;
```

Appelons la fonction *Hamilton* modifiée :

```
N = Hamilton(A) ;
disp(N,'Nombre de cycles hamiltoniens =')
```

On obtient : Nombre de cycles hamiltoniens = 0

Donc, ce graphe à 16 sommets ne possède pas de cycle hamiltonien.

Problème du voyageur de commerce

Définissons la matrice P des poids des arêtes d'un graphe de la manière suivante :

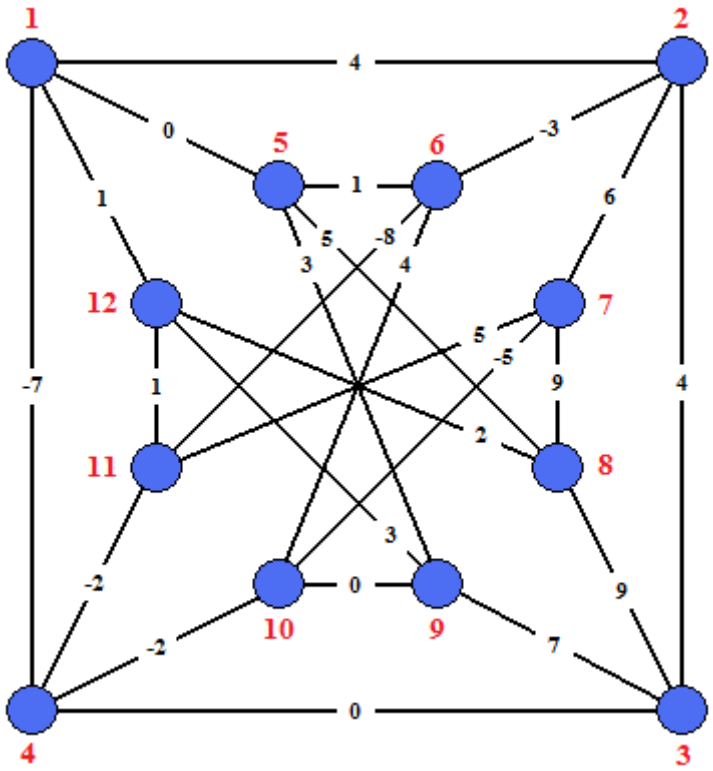
$$P(i,j) = \begin{cases} p & \text{si l'arête joignant le sommet n° i au sommet n° j a le poids p} \\ \infty & \text{sinon} \end{cases}$$

Si G est un graphe simple non orienté à n sommets alors P est une matrice carrée n × n symétrique.
Cette matrice est similaire à la matrice d'adjacence A du graphe, le booléen "1" est remplacé par un nombre quelconque et le booléen "0" par une valeur infinie (∞)
On recherche un cycle hamiltonien de poids minimal. Ce problème est connu sous le nom de "problème du voyageur de commerce". Il est de type NP-complet, donc difficile à résoudre pour de grands graphes.
Ecrivons une fonction *PVC* (Problème du Voyageur de Commerce) qui recherche un cycle hamiltonien H de poids minimal p dans un graphe G défini par sa matrice des poids P :

```
function [H,p] = PVC(P)
    function parcours(S,i)
        global H p
        for k = 1:length(S)
            if S(k) == i then, return, end
        end
        S($+1) = i
        if i == j then
            if length(S) == n then
                q = 0
                for k = 1:n-1
                    q = q + P(S(k),S(k+1))
                end
                q = q + P(S(n),S(1))
                if q < p | p == [] then
                    H = S
                    p = q
                end
            end
        else
            for k = 1:n
                if P(i,k) < %inf then
                    parcours(S,k)
                end
            end
        end
    endfunction
    global H p
    H = list()
    p = []
    [n,n] = size(P)
    i = 1
    for j = 1:n
        if P(i,j) < %inf then
            parcours(list(),i)
        end
    end
endfunction
```

Mathématique & Informatique

Reprenons le graphe de **Chvátal** et attribuons des poids à ses arêtes.
Sur le schéma, les poids des 24 arêtes sont en noir et les numéros des 12 sommets sont en rouge :



La matrice des poids est :

$$P = \begin{pmatrix}
\infty & 4 & \infty & -7 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & 1 \\
4 & \infty & 4 & \infty & \infty & -3 & 6 & \infty & \infty & \infty & \infty & \infty \\
\infty & 4 & \infty & 0 & \infty & \infty & \infty & 9 & 7 & \infty & \infty & \infty \\
-7 & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & -2 & -2 & \infty \\
0 & \infty & \infty & \infty & \infty & 1 & \infty & 5 & 3 & \infty & \infty & \infty \\
\infty & -3 & \infty & \infty & 1 & \infty & \infty & \infty & \infty & 4 & -8 & \infty \\
\infty & 6 & \infty & \infty & \infty & \infty & \infty & 9 & \infty & -5 & 5 & \infty \\
\infty & \infty & 9 & \infty & 5 & \infty & 9 & \infty & \infty & \infty & \infty & 2 \\
\infty & \infty & 7 & \infty & 3 & \infty & \infty & \infty & \infty & 0 & \infty & 3 \\
\infty & \infty & \infty & -2 & \infty & 4 & -5 & \infty & 0 & \infty & \infty & \infty \\
\infty & \infty & \infty & -2 & \infty & -8 & 5 & \infty & \infty & \infty & \infty & 1 \\
1 & \infty & \infty & \infty & \infty & \infty & \infty & 2 & 3 & \infty & 1 & \infty
\end{pmatrix}$$

Mathématique & Informatique

Soit en Scilab :

```
P = ones(12,12)*%inf ;
P(1,2) = 4 ; P(1,4) = - 7 ; P(1,5) = 0 ; P(1,12) = 1 ;
P(2,3) = 4 ; P(2,6) = - 3 ; P(2,7) = 6 ;
P(3,4) = 0 ; P(3,8) = 9 ; P(3,9) = 7 ;
P(4,10) = - 2 ; P(4,11) = - 2 ;
P(5,6) = 1 ; P(5,8) = 5 ; P(5,9) = 3 ;
P(6,10) = 4 ; P(6,11) = - 8 ;
P(7,8) = 9 ; P(7,10) = - 5 ; P(7,11) = 5 ;
P(8,12) = 2 ;
P(9,10) = 0 ; P(9,12) = 3 ;
P(11,12) = 1 ;
P = min(P + P',P,P') ;
```

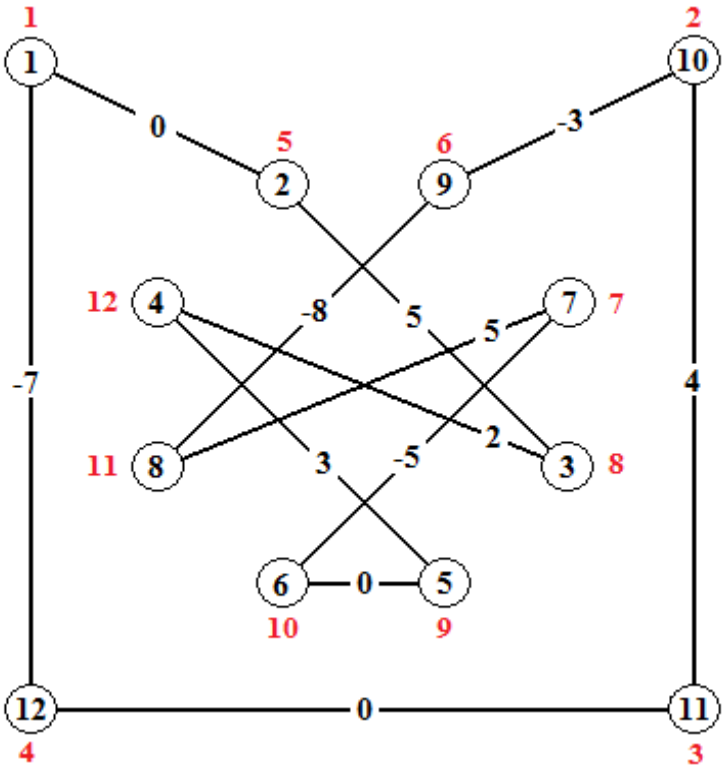
Recherchons le cycle hamiltonien de poids minimal :

```
[H,p] = PVC(P) ;
disp(H,'Cycle hamiltonien =')
disp(p,'Poids minimal =')
```

Résultat :

```
Cycle hamiltonien =   1   5   8   12   9   10   7   11   6   2   3   4
Poids minimal =   - 4
```

Graphique :



Mathématique & Informatique

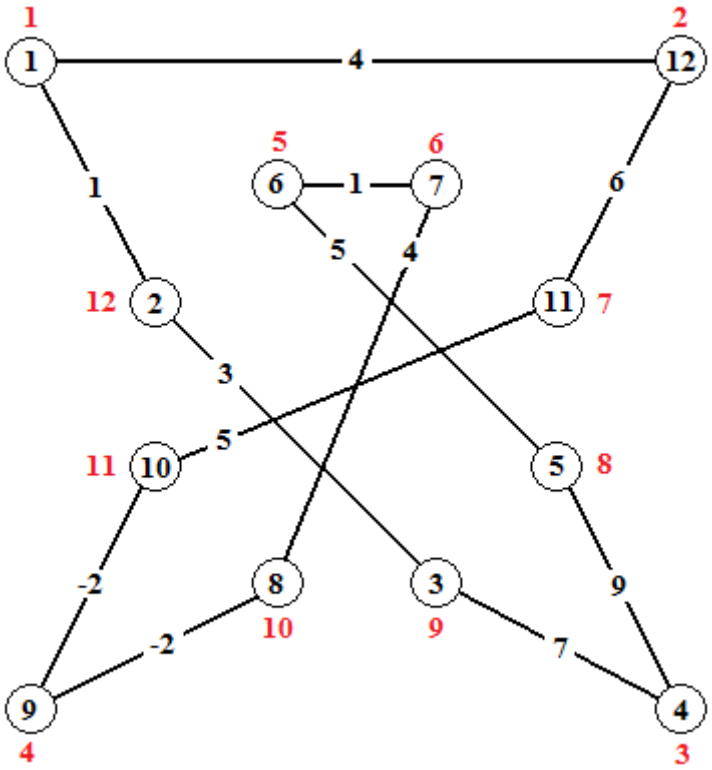
Pour trouver le cycle hamiltonien de poids maximal, on inverse le signe des arêtes et on inverse aussi le signe du poids p du cycle optimal obtenu (Max[f(x)] = - Min[- f(x)]) :

```
P = ones(12,12)*%inf ;
P(1,2) = - 4 ; P(1,4) = 7 ; P(1,5) = 0 ; P(1,12) = - 1 ;
P(2,3) = - 4 ; P(2,6) = 3 ; P(2,7) = - 6 ;
P(3,4) = 0 ; P(3,8) = - 9 ; P(3,9) = - 7 ;
P(4,10) = 2 ; P(4,11) = 2 ;
P(5,6) = - 1 ; P(5,8) = - 5 ; P(5,9) = - 3 ;
P(6,10) = - 4 ; P(6,11) = 8 ;
P(7,8) = - 9 ; P(7,10) = 5 ; P(7,11) = - 5 ;
P(8,12) = - 2 ;
P(9,10) = 0 ; P(9,12) = - 3 ;
P(11,12) = - 1 ;
P = min(P + P',P,P') ;
[H,p] = PVC(P) ;
disp(H,'Cycle hamiltonien =')
disp(-p,'Poids maximal =')
```

Résultat :

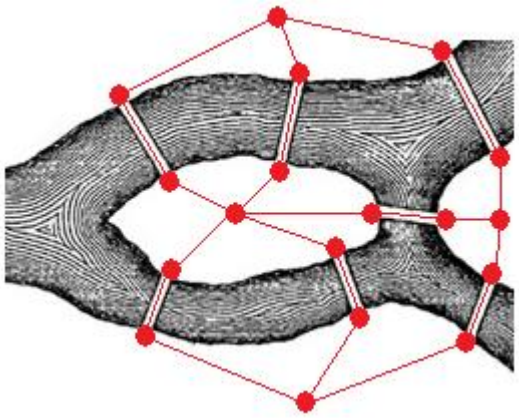
```
Cycle hamiltonien =   1   12   9   3   8   5   6   10   4   11   7   2
Poids maximal =   41
```

Graphique :



Cycle eulérien

Un cycle eulérien est un circuit qui passe une seule fois par toutes les arêtes d'un graphe.
Prenons l'exemple célèbre des 7 ponts de Köenigsberg (aujourd'hui Kaliningrad en Russie) :
⇨ **http://home.nordnet.fr/~ajuhel/Koenigsberg/koenigsberg.html**



Existe-t-il un itinéraire circulaire où l'on passe une seule fois sur tous les ponts ?
Euler prouva, en 1736, que la réponse à cette question est non.
Historiquement, la formulation mathématique de ce genre de problème est à l'origine de la théorie des graphes.

On montre qu'un graphe connexe G admet un circuit eulérien si et seulement si ses sommets sont de degré pair.
Pour un sommet x quelconque :

- $d(x) \equiv 0 \quad (2)$ (graphe non orienté)
- $d^-(x) = d^+(x)$ (graphe orienté)

où d(x) est le nombre d'arêtes adjacentes, $d^-(x)$ est le nombre d'arcs entrants adjacents, $d^+(x)$ est le nombre d'arcs sortants adjacents. Cette propriété est un moyen très simple de reconnaître les graphes eulériens.
Donnons une fonction booléenne *eulerien* qui vérifie si un graphe non orienté de matrice d'adjacence A est eulérien :

```
function e = eulerien(A)
    [n,n] = size(A)
    for x = 1:n
        e = modulo(sum(A(x,:)),2) == 0
        if ~e then, return, end
    end
endfunction
```

Il reste à construire le cycle eulérien en question avec l'algorithme de **Fleury** (1883) :

Initialiser le cycle eulérien : $E = \varnothing$
Prendre un premier sommet : $x = x_0$
tantque il existe une arête
| ① Ajouter le sommet x au cycle E : $E = E + \{x\}$
| ② Choisir si possible une arête (x,y) qui ne soit pas un isthme et la retirer du graphe G : $G = G - \{(x,y)\}$
| ③ Faire : $x = y$
fin

En ce qui concerne le graphisme, le tracé d'un graphe eulérien peut s'effectuer sans lever le crayon.

Ecrivons une fonction *Euler* qui recherche un cycle eulérien E dans un graphe G de matrice d'adjacence A :

```
function E = Euler(A)
    function y = cycle(A,x)
        function parcours(S,y)
            global trouve
            if trouve then, return, end
            for k = 1:length(S)
                if S(k) == y then, return, end
            end
            S($+1) = y
            if y == x then
                trouve = length(S) >= 3
                if trouve then, return, end
            else
                V = find(A(y,:) == 1)
                for z = V
                    parcours(S,z)
                    if trouve then, return, end
                end
            end
        endfunction
        global trouve
        trouve = %f
        V = find(A(x,:) == 1)
        if length(V) == 1 then
            y = V
            return
        end
        for y = V
            parcours(list(),y)
            if trouve then, return, end
        end
        y = []
    endfunction
    [n,n] = size(A)
    m = sum(A)/2
    E = list()
    if n < 3 then, return, end
    x = 1
    while m > 0
        E($+1) = x
        y = cycle(A,x)
        if y == [] then
            E = list()
            return
        end
        A(x,y) = 0
        A(y,x) = 0
        m = m - 1
        x = y
    end
endfunction
```

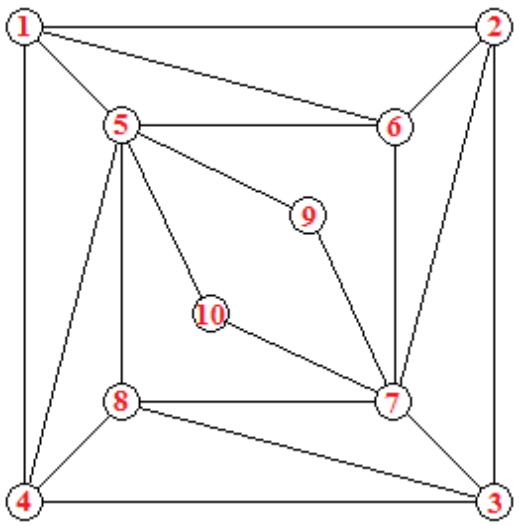
La fonction *cycle* renvoie le premier voisin y du sommet x tel que l'arête (x,y) ne déconnecte pas le graphe, ou bien le seul voisin y du sommet x si ce dernier est un sommet pendant (i.e. de degré 1). Dans les autres cas, elle retourne ∅.

Mathématique & Informatique

Remarques

- La liste E est une suite de sommets à parcourir dans l'ordre (après le dernier élément on revient au premier). En prenant $x_0 = 1$ (sommet numéroté 1 du graphe), la liste E commence toujours par 1.
- Lorsqu'on supprime l'arête (x,y) d'un sommet pendant x, on déconnecte effectivement le graphe mais c'est sans importance ici puisque la composante connexe créée est réduite au sommet isolé x et ne contient pas d'arête.
- L'algorithme de **Fleury** est un "algorithme glouton" ("greedy algorithm" dans la littérature anglo-saxonne) : au cours du processus on est amené à faire des choix qui ne seront jamais infirmés par la suite.
- Pour tester l'algorithme sur des graphes ayant de nombreuses arêtes, on peut prendre les graphes complets K_n qui sont eulériens lorsque le nombre n de sommets est impair. On écrira : *E = Euler(ones(n,n) - eye(n,n))*
- Il est possible de construire un chemin eulérien dans un graphe possédant 2 sommets de degré impair. Soit a et b ces 2 sommets, on créé l'arête (a,b) pour constituer un graphe eulérien et trouver un cycle eulérien empruntant nécessairement l'arête (a,b), puis on la retire du cycle pour obtenir un chemin eulérien de a vers b.

EXEMPLE 1



Programme :

```
A = zeros(10,10) ;
A(1,[2,4,5,6]) = 1 ;
A(2,[3,6,7]) = 1 ;
A(3,[4,7,8]) = 1 ;
A(4,[5,8]) = 1 ;
A(5,[6,8,9,10]) = 1 ;
A(6,[7]) = 1 ;
A(7,[8,9,10]) = 1 ;
A = A + A' ;
E = Euler(A) ;
disp(E)
```

Résultat :

1 2 3 4 1 5 4 8 3 7 2 6 5 8 7 9 5 10 7 6

EXEMPLE 2

Le graphe de **Chvátal** est un graphe eulérien puisque tous ses sommets sont de degré pair (d(x) = 4, ∀ x).
Recherchons un cycle eulérien avec la fonction *Euler* :

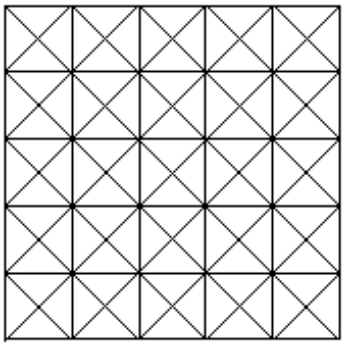
```
A = zeros(12,12) ;
A(1,[2,4,5,12]) = 1 ; A(2,[3,6,7]) = 1 ; A(3,[4,8,9]) = 1 ; A(4,[10,11]) = 1 ;
A(5,[6,8,9]) = 1 ; A(6,[10,11]) = 1 ; A(7,[8,10,11]) = 1 ; A(8,[12]) = 1 ; A(9,[10,12]) = 1 ; A(11,[12]) = 1 ;
A = A + A' ;
E = Euler(A) ;
disp(E,'Cycle eulérien = ')
```

On obtient immédiatement :

Cycle eulérien = 1 2 3 4 1 5 6 2 7 8 3 9 5 8 12 9 10 4 11 6 10 7 11 12

Mathématique & Informatique

Un problème de dessin



Dans un grand carré composé de 25 petits carrés et leurs diagonales, le but est de tracer le maximum de petits carrés (avec leurs diagonales) sans lever le crayon et sans repasser 2 fois sur le même trait.

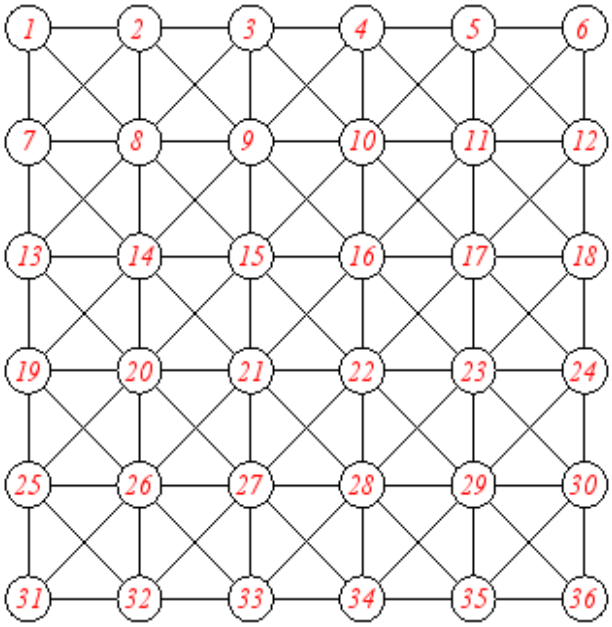
Méthode n° 1
On prend son crayon et une gomme, puis on se lance à l'aventure dans les dédales du schéma. Après de nombreuses tentatives infructueuses et avec un peu (beaucoup) de chance on arrive enfin à la solution ...

Méthode n° 2
Raisonnons en termes de graphe.
La figure ci-dessus correspond à un graphe G dont les 36 sommets sont les points d'intersection de 6 lignes horizontales et de 6 lignes verticales. Les arêtes sont les traits (horizontaux, verticaux, obliques) qui relient
ces points entre eux.
Voici un programme Scilab qui dessine et numérote ce graphe (de gauche à droite et de haut en bas) :

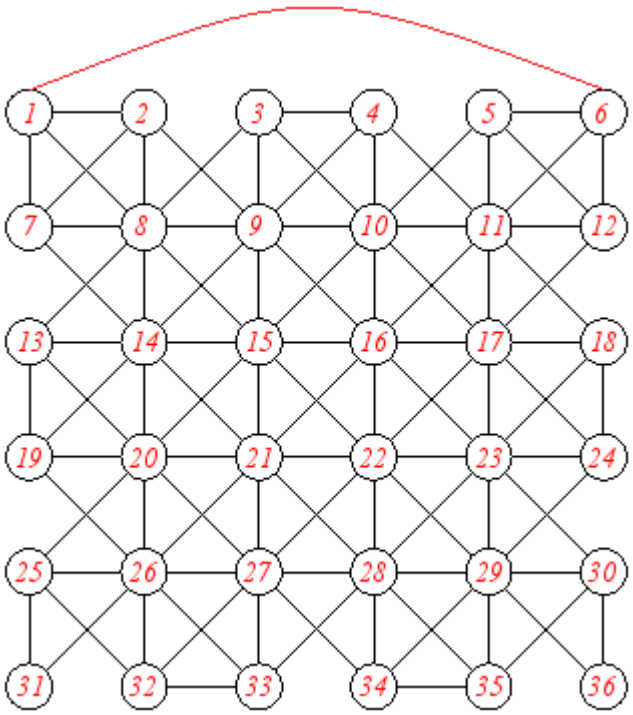
```
function cercle(r,x0,y0,c1,c2)
    xfarc(x0-r,y0+r,2*r,2*r,0,64*360)
    e = gce() ; e.background = c1
    xarc(x0-r,y0+r,2*r,2*r,0,64*360)
    e = gce() ; e.foreground = c2
endfunction

f = scf(1) ;
drawlater()
c = 10 ; r = 2 ;
plot2d(%inf,%inf,frameflag=1,rect=[-r,-r,5*c+r,5*c+r],axesflag=0)
a = gca() ; a.isoview = 'on' ;
for i = 0:4
    for j = 1:5
        xrect(i*c,j*c,c,c)
        xsegs([i*c,i*c;i*c+c,i*c+c],[j*c,j*c-c;j*c-c,j*c],[1,1])
    end
end
for i = 0:5
    for j = 0:5
        cercle(r,i*c,j*c,8,1)
        n = 31 + i - 6*j ;
        xstringb(i*c-r,j*c-r,string(n),2*r,2*r)
        e = gce() ; e.font_style = 3 ; e.font_size = 3 ; e.font_foreground = 5 ;
    end
end
drawnow()
```

Mathématique & Informatique



Le graphe G n'est pas eulérien puisque les sommets qui se trouvent à la périphérie du grand carré sont tous de degré impair : degré 3 pour les 4 coins et degré 5 pour les 16 autres sommets du bord.
Un petit carré élémentaire et ses diagonales constitue un sous-graphe complet K_4 sur 4 sommets.
Le problème consiste à extraire un graphe partiel qui soit eulérien et qui contienne le maximum de sous-graphes K_4.
En enlevant 10 arêtes périphériques au graphe G, à savoir les arêtes (1,2), (3,4), (5,6), (7,13), (12,18), (19,25), (24,30), (31,32), (33,34), (35,36), on obtient un graphe partiel dont tous les sommets sont de degré pair (2, 4, 8) et qui contient 15 sous-graphes K_4. Mais on peut mieux faire. En fait, il suffit de construire un chemin eulérien.
On peut donc conserver 2 sommets de degré impair et les relier par une arête. Par exemple, en reliant les sommets 1 et 6 on aboutit à un graphe eulérien contenant 16 sous-graphes K_4 :



L'arête (1,6) en rouge est une arête provisoire qui sert uniquement à la recherche algorithmique du cycle eulérien.
Elle sera supprimée par la suite pour créer le chemin eulérien allant du sommet n° 1 au sommet n° 6.
Ce chemin pourra être tracé sans lever le crayon.

Programmation

Enregistrons la matrice d'adjacence A du graphe eulérien :

```
A = zeros(36,36) ;
A(1,[2,6,7,8]) = 1 ;
A(2,[7,8,9]) = 1 ;
A(3,[4,8,9,10]) = 1 ;
A(4,[9,10,11]) = 1 ;
A(5,[6,10,11,12]) = 1 ;
A(6,[11,12]) = 1 ;
A(7,[8,14]) = 1 ;
A(8,[9,13,14,15]) = 1 ;
A(9,[10,14,15,16]) = 1 ;
A(10,[11,15,16,17]) = 1 ;
A(11,[12,16,17,18]) = 1 ;
A(12,[17]) = 1 ;
A(13,[14,19,20]) = 1 ;
A(14,[15,19,20,21]) = 1 ;
A(15,[16,20,21,22]) = 1 ;
A(16,[17,21,22,23]) = 1 ;
A(17,[18,22,23,24]) = 1 ;
A(18,[23,24]) = 1 ;
A(19,[20,26]) = 1 ;
A(20,[21,25,26,27]) = 1 ;
A(21,[22,26,27,28]) = 1 ;
A(22,[23,27,28,29]) = 1 ;
A(23,[24,28,29,30]) = 1 ;
A(24,[29]) = 1 ;
A(25,[26,31,32]) = 1 ;
A(26,[27,31,32,33]) = 1 ;
A(27,[28,32,33,34]) = 1 ;
A(28,[29,33,34,35]) = 1 ;
A(29,[30,34,35,36]) = 1 ;
A(30,[35,36]) = 1 ;
A(32,[33]) = 1 ;
A(34,[35]) = 1 ;
A = A + A' ;
```

Cette matrice correspond à un graphe de m = sum(A)/2 = 102 arêtes.

L'appel à la fonction *Euler(A)* donne un cycle eulérien commençant par le sommet n° 1 :

1 2 7 1 6 5 10 3 4 9 2 8 3 9 8 7 14 8 13 14 9 10 4 11 5 12 6 11
10 15 9 16 10 17 11 12 17 16 11 18 17 22 15 14 19 13 20 14 21 15 16
21 20 19 26 20 25 26 21 22 16 23 17 24 18 23 22 27 21 28 22 29 23
24 29 28 23 30 29 34 27 26 31 25 32 26 33 27 28 34 35 29 36 30 35
28 33 32 27 20 15 8

Ecrivons une fonction *C = chemin_eulerien(E,i,j)* qui fournit le chemin eulérien C du sommet i au sommet j à partir d'un cycle eulérien E dans un graphe G non orienté :

```
function C = chemin_eulerien(E,i,j)
    function L2 = lstinv(L1)
        L2 = list()
        for k = length(L1):-1:1
            L2($+1) = L1(k)
        end
    endfunction
    C = list() ;
    if E == list() then, return, end
    C1 = list() ; C2 = list()
    flag1 = %f ; flag2 = %f
    E($+1) = E(1)
    for k = 1:length(E)-1
        if ~flag1 & ~flag2 then
            flag1 = (E(k) == i & E(k+1) == j)
            flag2 = (E(k) == j & E(k+1) == i)
            C1($+1) = E(k)
        else
            C2($+1) = E(k)
        end
    end
    if flag1 then, C = lstcat(lstinv(C1),lstinv(C2)), end
    if flag2 then, C = lstcat(C2,C1), end
endfunction
```

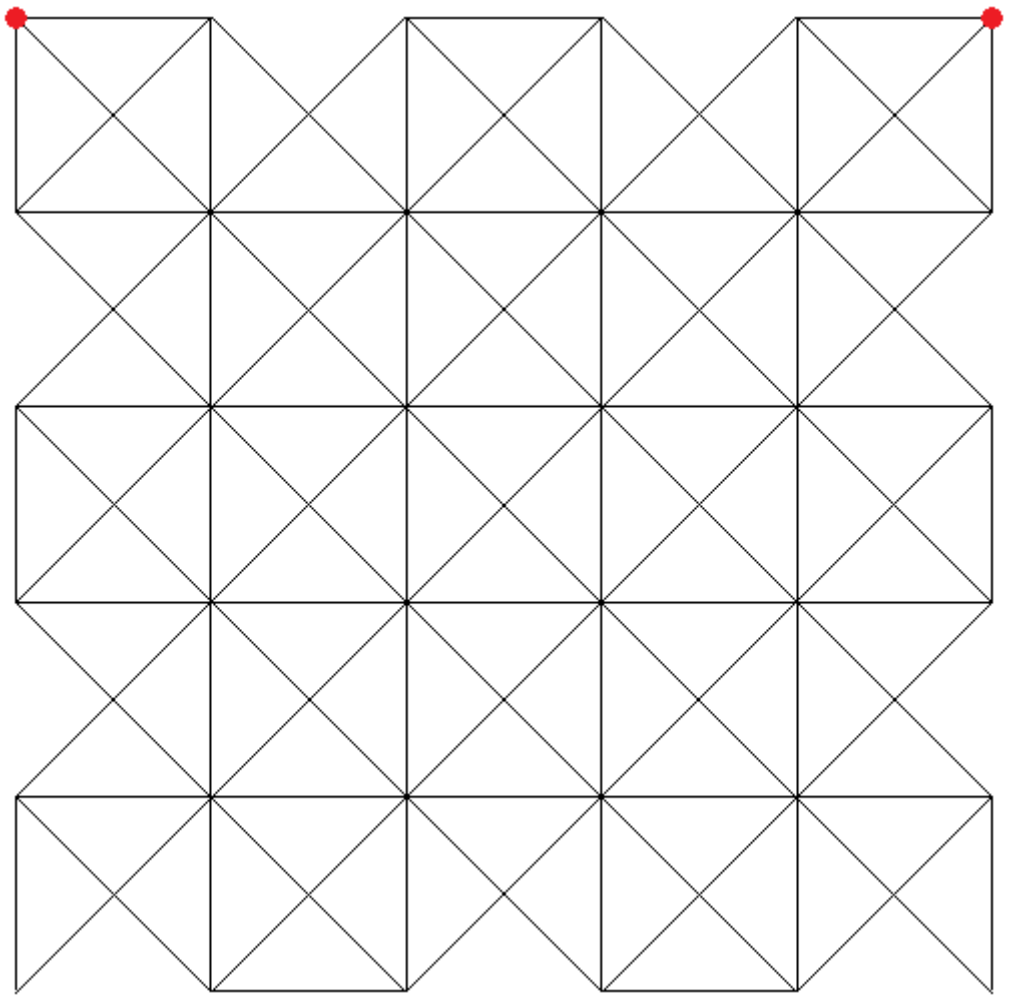
L'appel à la fonction *chemin_eulerien(E,1,6)* donne un chemin eulérien du sommet n° 1 au sommet n° 6 :

1 7 2 1 8 15 20 27 32 33 28 35 30 36 29 35 34 28 27 33 26 32 25 31
26 27 34 29 30 23 28 29 24 23 29 22 28 21 27 22 23 18 24 17 23 16
22 21 26 25 20 26 19 20 21 16 15 21 14 20 13 19 14 15 22 17 18 11
16 17 12 11 17 10 16 9 15 10 11 6 12 5 11 4 10 9 14 13 8 14 7 8 9
3 8 2 9 4 3 10 5 6

Il ne reste plus qu'à tracer à la main les 16 petits carrés en suivant dans l'ordre cette liste de numéros.

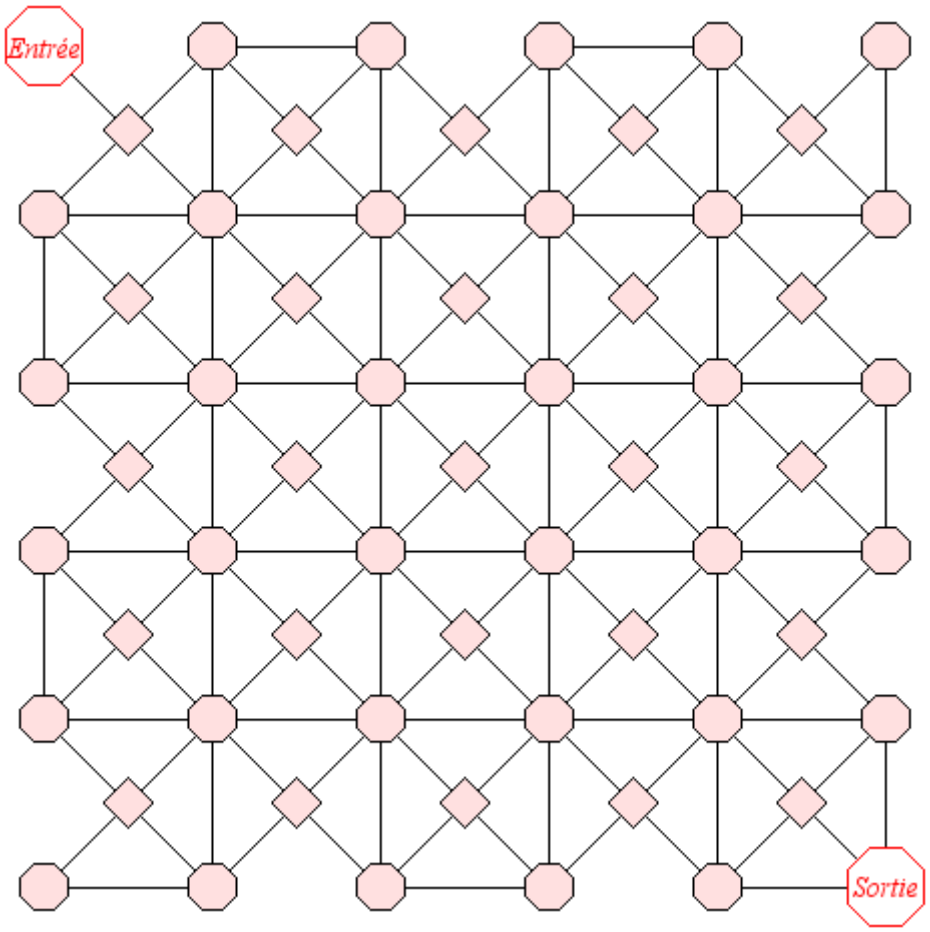
On peut aussi générer le tracé automatiquement dans une fenêtre graphique Scilab :

```
E = Euler(A) ;
C = chemin_eulerien(E,1,6) ;
f = scf(2) ;
plot2d(%inf,%inf,frameflag=3,rect=[0,0,5,5],axesflag=0)
for k = 1:length(C)-1
    i1 = modulo(C(k)-1,6) ; j1 = (31 - C(k) + i1)/6 ;
    i2 = modulo(C(k+1)-1,6) ; j2 = (31 - C(k+1) + i2)/6 ;
    xsegs([i1;i2],[j1;j2],1)
    xpause(0.3e6)
end
```



Labyrinthe eulérien

On dispose du plan d'un labyrinthe où chaque pièce est représentée par un polygone et chaque couloir par un segment de droite. A l'entrée, figure l'inscription sibylline : << Ouvre toutes les portes >>



L'enigme

Chaque couloir possède deux portes à ses extrémités. Une porte s'ouvre en appuyant sur un poussoir de commande situé à proximité, mais uniquement si toutes les autres portes du labyrinthe sont en position fermée. Un mécanisme referme automatiquement la porte qui vient d'être franchie. Chaque porte ne peut être ouverte qu'une seule fois, elle demeure verrouillée par la suite. Un détecteur de passage, situé au milieu de chaque couloir, signale que celui-ci a été traversé. La porte de sortie se déverrouille lorsque tous les couloirs ont été empruntés, donc si toutes les portes ont été ouvertes. Quel trajet doit on effectuer pour quitter ce labyrinthe ?

La solution

Il s'agit de construire un chemin eulérien entre l'entrée et la sortie.
Pour simplifier les calculs les pièces situées à l'intersection de 2 petites diagonales ne seront pas prises en compte, leur utilité est de permettre le croisement des couloirs obliques lors de la construction du labyrinthe. Sur le graphe, ces 4 petits couloirs et la pièce centrale carrée qui les relie seront représentés par 2 arêtes qui se croisent.
Ainsi, le labyrinthe se réduit à un graphe eulérien à 36 sommets (symbolisant les pièces octogonales) où l'entrée et la sortie sont reliées par une arête fictive (couloir imaginaire).
Après avoir défini la matrice d'adjacence A du graphe, on applique la fonction *Euler* pour trouver un cycle eulérien.
La fonction *chemin_eulerien* donne le résultat recherché en retirant l'arête joignant l'entrée à la sortie.

Remarque

Plusieurs chemins sont possibles.

```
// ---------------------------------------------------------------------
//                        Labyrinthe eulérien
// ---------------------------------------------------------------------
//
//                  Loïc Vrillon - Février 2011
//

funcprot(0) ; lines(0)

// Fenêtre graphique

f = scf(1) ;
f.figure_name = 'Labyrinthe' ;
X = getsystemmetrics('SM_CXSCREEN') ; Y = getsystemmetrics('SM_CYSCREEN') ;
f.figure_size = [800,800] ; f.figure_position = [(X - 800)/2,(Y - 800)/2] ;

// Dessin du labyrinthe

function polygone(n,r,x0,y0,theta0,c1,c2)
    X = zeros(n,1) ; Y = zeros(n,1)
    for p = 1:n
        z = x0 + %i*y0 + r*exp(%i*(theta0 + (p-1)*2*%pi/n))
        X(p) = real(z) ; Y(p) = imag(z)
    end
    xfpoly(X,Y)
    e = gce() ; e.closed = 'on' ; e.background = c1
    xpoly(X,Y)
    e = gce() ; e.closed = 'on' ; e.foreground = c2
endfunction

c = 10 ; r = 1.5 ; R = 2.5 ;
a = gca() ;
a.data_bounds = [-R,-R;5*c+R,5*c+R] ;
a.axes_visible = 'off' ;
a.isoview = 'on' ;
drawlater()
for i = 0:4
    for j = 1:5
        xrect(i*c,j*c,c,c)
        xsegs([i*c,i*c;i*c+c,i*c+c],[j*c,j*c-c;j*c-c,j*c],[1,1])
    end
end
S = [0,5,1,5 ; 2,5,3,5 ; 4,5,5,5 ; 5,4,5,3 ; 5,2,5,1 ; 4,0,3,0 ; 2,0,1,0 ; 0,0,0,1 ; 0,2,0,3 ; 0,4,0,5] ;
xsegs([S(:,1)'*c;S(:,3)'*c],[S(:,2)'*c;S(:,4)'*c],ones(1,size(S,1))*8)
for i = 0:5
    for j = 0:5
        polygone(8,r,i*c,j*c,%pi/8,31,1) ;
        if i < 5 & j > 0 then
            polygone(4,r,(i+1/2)*c,(j-1/2)*c,0,31,1) ;
        end
        if (i == 0 & j == 5) | (i == 5 & j == 0) then
            polygone(8,R,i*c,j*c,%pi/8,8,5) ;
            if i == 0 then, texte = 'Entrée' ; else, texte = 'Sortie' ; end
            xstringb(i*c-R,j*c-R,texte,2*R,2*R)
            e = gce() ; e.font_style = 3 ; e.font_size = 3 ; e.font_foreground = 5 ;
        end
    end
end
drawnow()
```

```
// Recherche d'un cycle eulérien

function E = Euler(A)
    function y = cycle(A,x)
        function parcours(S,y)
            global trouve
            if trouve then, return, end
            for k = 1:length(S)
                if S(k) == y then, return, end
            end
            S($+1) = y
            if y == x then
                trouve = length(S) >= 3
                if trouve then, return, end
            else
                V = find(A(y,:) == 1)
                for z = V
                    parcours(S,z)
                    if trouve then, return, end
                end
            end
        endfunction
        global trouve
        trouve = %f
        V = find(A(x,:) == 1)
        if length(V) == 1 then
            y = V
            return
        end
        for y = V
            parcours(list(),y)
            if trouve then, return, end
        end
        y = []
    endfunction
    [n,n] = size(A)
    m = sum(A)/2
    E = list()
    if n < 3 then, return, end
    x = 1
    while m > 0
        E($+1) = x
        y = cycle(A,x)
        if y == [] then
            E = list()
            return
        end
        A(x,y) = 0
        A(y,x) = 0
        m = m - 1
        x = y
    end
endfunction
```

```
// Recherche d'un chemin eulérien

function C = chemin_eulerien(E,i,j)
    function L2 = lstinv(L1)
        L2 = list()
        for k = length(L1):-1:1
            L2($+1) = L1(k)
        end
    endfunction
    C = list() ;
    if E == list() then, return, end
    C1 = list() ; C2 = list()
    flag1 = %f ; flag2 = %f
    E($+1) = E(1)
    for k = 1:length(E)-1
        if ~flag1 & ~flag2 then
            flag1 = (E(k) == i & E(k+1) == j)
            flag2 = (E(k) == j & E(k+1) == i)
            C1($+1) = E(k)
        else
            C2($+1) = E(k)
        end
    end
    if flag1 then, C = lstcat(lstinv(C1),lstinv(C2)), end
    if flag2 then, C = lstcat(C2,C1), end
endfunction
```

```
// Matrice d'adjacence

A = zeros(36,36) ;
A(1,[8,36]) = 1 ;
A(2,[3,7,8,9]) = 1 ;
A(3,[8,9,10]) = 1 ;
A(4,[5,9,10,11]) = 1 ;
A(5,[10,11,12]) = 1 ;
A(6,[11,12]) = 1 ;
A(7,[8,13,14]) = 1 ;
A(8,[9,13,14,15]) = 1 ;
A(9,[10,14,15,16]) = 1 ;
A(10,[11,15,16,17]) = 1 ;
A(11,[12,16,17,18]) = 1 ;
A(12,[17]) = 1 ;
A(13,[14,20]) = 1 ;
A(14,[15,19,20,21]) = 1 ;
A(15,[16,20,21,22]) = 1 ;
A(16,[17,21,22,23]) = 1 ;
A(17,[18,22,23,24]) = 1 ;
A(18,[23,24]) = 1 ;
A(19,[20,25,26]) = 1 ;
A(20,[21,25,26,27]) = 1 ;
A(21,[22,26,27,28]) = 1 ;
A(22,[23,27,28,29]) = 1 ;
A(23,[24,28,29,30]) = 1 ;
A(24,[29]) = 1 ;
A(25,[26,32]) = 1 ;
A(26,[27,31,32,33]) = 1 ;
A(27,[28,32,33,34]) = 1 ;
A(28,[29,33,34,35]) = 1 ;
A(29,[30,34,35,36]) = 1 ;
A(30,[35,36]) = 1 ;
A(31,[32]) = 1 ;
A(33,[34]) = 1 ;
A(35,[36]) = 1 ;
A = A + A' ;

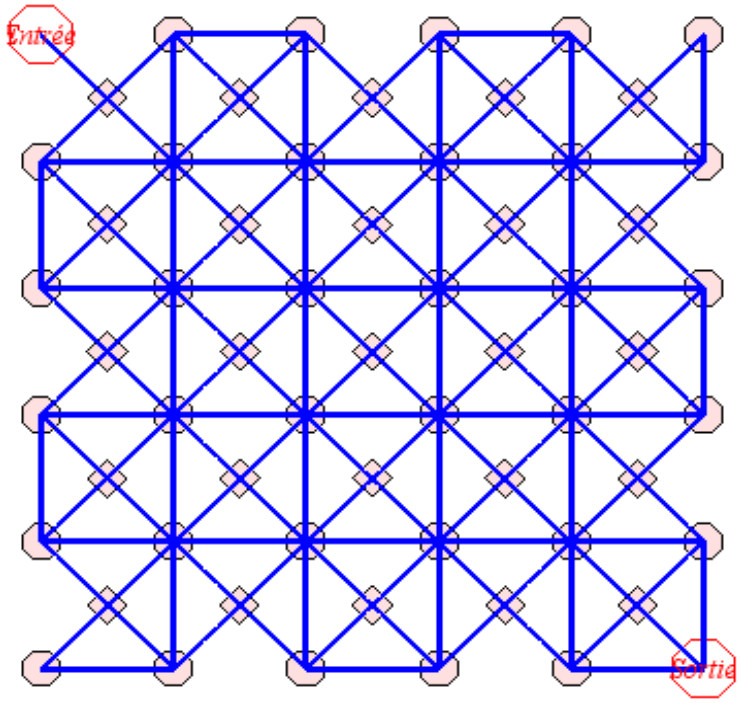
// Calcul

E = Euler(A) ;
C = chemin_eulerien(E,1,36) ;
printf('\n Chemin eulérien =   ')
for i = 1:length(C), printf('   %i',C(i)), end
```

```
// Tracé du chemin eulérien

function segment(x1,y1,x2,y2)
    v = 10 ; t = 2e4
    if x1 == x2 then
        X = ones(v,1)*x1
        Y = linspace(y1,y2,v)
    end
    if y1 == y2 then
        X = linspace(x1,x2,v)
        Y = ones(v,1)*y1
    end
    if (x1 <> x2) & (y1 <> y2) then
        a = (y2-y1)/(x2-x1)
        b = y1 - a*x1
        X = linspace(x1,x2,v)
        Y = a*X + b
    end
    for i = 1:v-1
        xsegs([X(i);X(i+1)],[Y(i);Y(i+1)],2)
        e = gce() ; e.thickness = 3
        xpause(t)
    end
endfunction

for k = 1:length(C)-1
    i1 = modulo(C(k)-1,6) ; j1 = (31 - C(k) + i1)/6 ;
    i2 = modulo(C(k+1)-1,6) ; j2 = (31 - C(k+1) + i2)/6 ;
    segment(i1*c,j1*c,i2*c,j2*c)
end
```

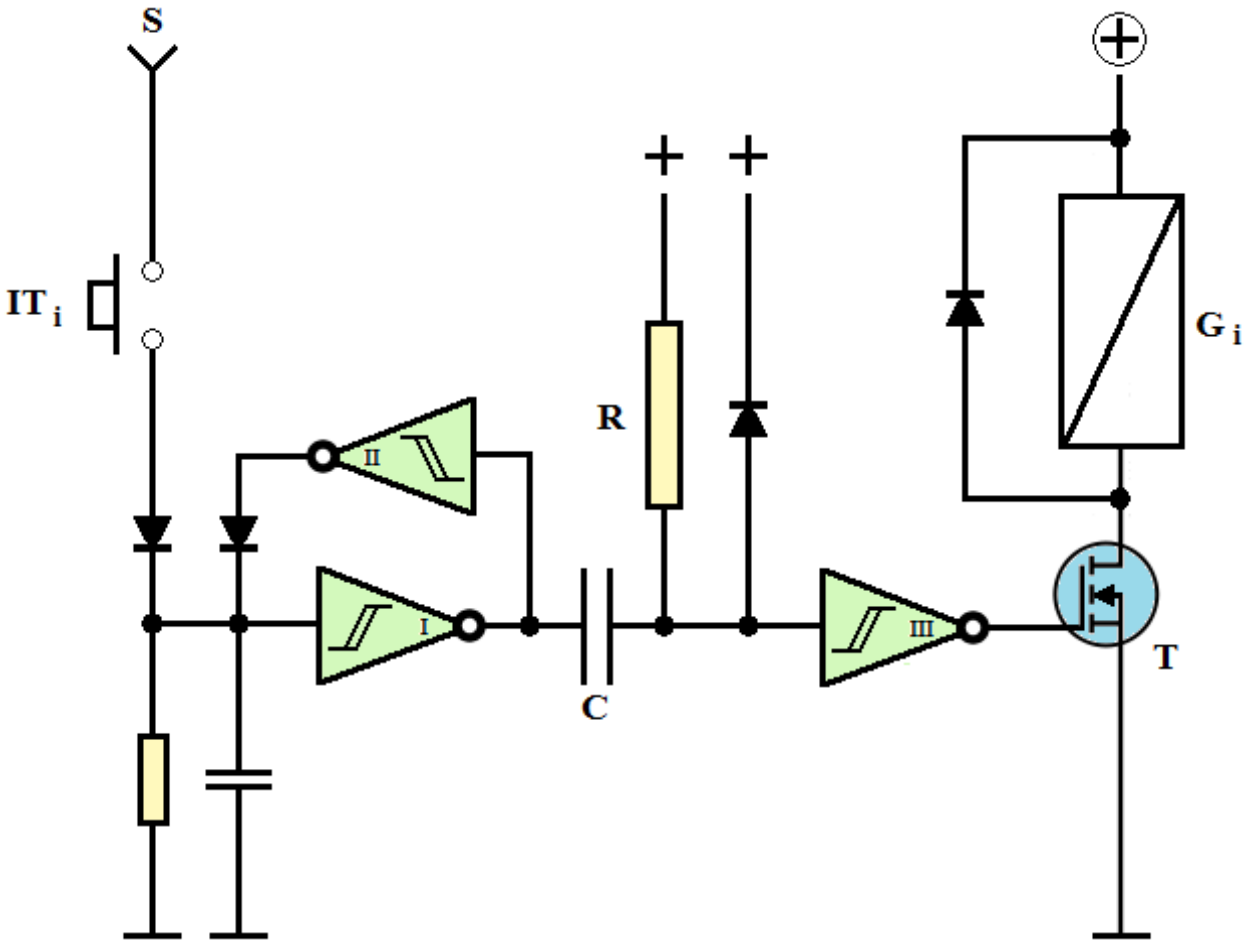


Supplément : réalisation pratique

Décrivons un dispositif électronique contrôlant l'ouverture des portes du labyrinthe.
On suppose que chaque porte est équipée d'un micro-contact normalement fermé et d'une gâche électrique commandée par un interrupteur manuel.
Notons NF_i le micro-contact de la porte n° i et IT_i l'interrupteur qui actionne la gâche G_i
Pour qu'une porte puisse s'ouvrir il faut au préalable que toutes les portes du labyrinthe soient en position fermée.
Un signal S de fermeture s'obtient en câblant en série tous les contacts NF_i avec la tension d'alimentation :



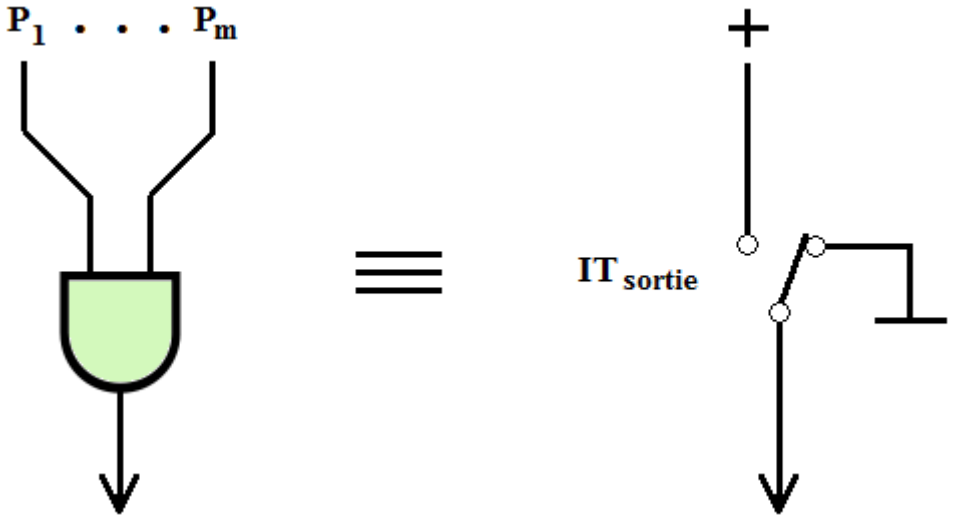
Bien sûr, on peut utiliser un microcontrôleur pour gérer l'ensemble du système, mais de simples portes logiques suffisent.
Prenons, par exemple, des inverseurs CMOS contenus dans le circuit intégré de référence CD 4584.



Note
Pour représenter des fonctions logiques élémentaires, on a le choix entre les symboles ANSI et les symboles CEI.

Lorsque le signal S est à "1", l'appui sur IT_i est mémorisé à l'aide des inverseurs I et II : la sortie de l'inverseur I chute à "0" et un "1" apparaît en sortie de l'inverseur II. Par suite, la sortie de l'inverseur III passe également à "1" entraînant la conduction du transistor MOSFET de commutation T et l'alimentation de la gâche G_i pendant le temps de la charge du condensateur C à travers la résistance R (Δt de l'ordre de RC). Notons que les inverseurs sont des triggers de Schmitt dont les 2 seuils de basculement ("haut" et "bas") permettent d'obtenir une transition franche du signal de déclenchement en palliant les irrégularités de la charge de C. Les diodes court-circuitant la résistance R et la gâche G_i sont des protections contre les surtensions. En résumé, un appui sur IT_i actionne la gâche G_i et tout nouvel appui sera sans effet. Donc, une porte ne pourra être ouverte qu'une seule fois.

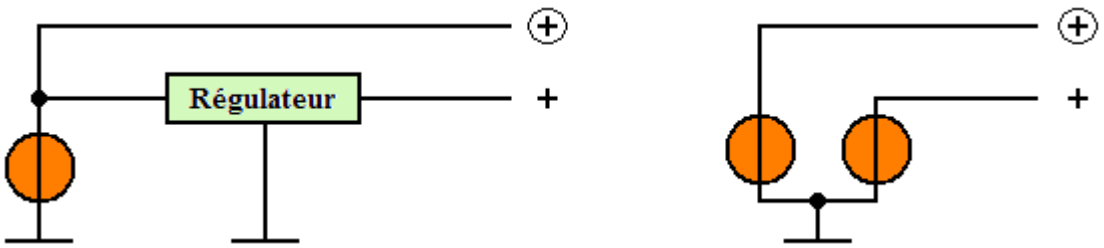
Dans chaque couloir, un détecteur de passage – par exemple un tapis de contact de type NO dont le fonctionnement est similaire à celui d'un interrupteur – entraine le basculement d'un signal P_i de "0" à "1". Les signaux P_i sont mémorisés et la porte de sortie se libère lorsqu'ils sont tous à "1". Cette porte ne disposant pas d'interrupteur d'ouverture, celui-ci est remplacé par un ET logique des P_i dans un schéma identique au précédent :



Pour réaliser le ET logique de nombreux signaux, une solution consiste à utiliser des circuits CD 4068 (NAND/AND CMOS à 8 entrées) les uns à la suite des autres : utiliser n circuits pour obtenir 7n+1 entrées et une sortie.

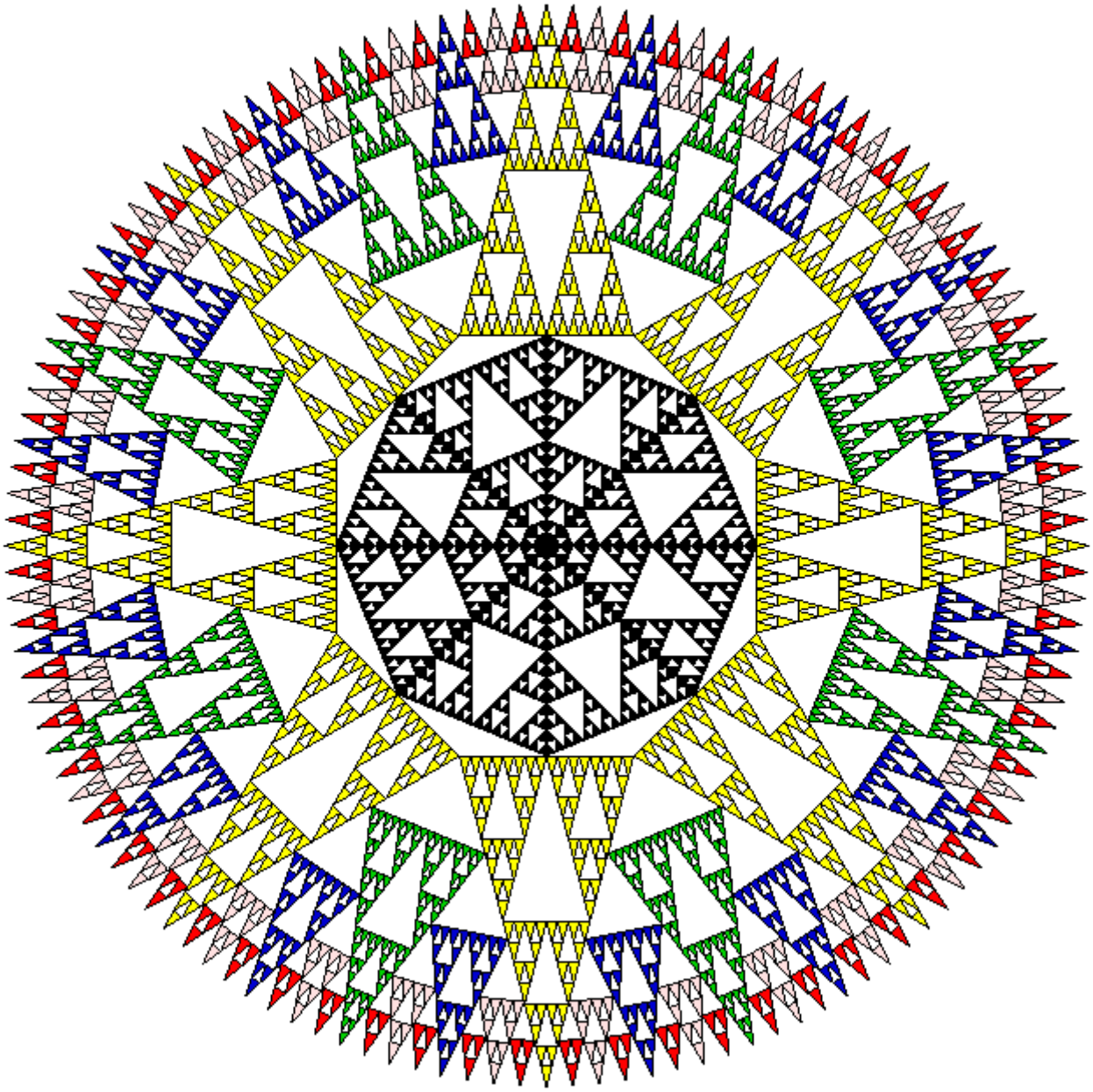
Remarques

- La consommation en veille de l'ensemble du montage est extrêmement faible.
- Une brève coupure de la tension d'alimentation réinitialise le système.
- La logique de commande sera alimentée par une tension régulée (+), alors qu'une tension simplement filtrée convient pour les gâches (⊕) . Autre possibilité : utiliser 2 générateurs différents avec une masse commune.



- Dans le premier schéma électronique, les portes logiques peuvent être remplacées par un circuit spécialisé comme le CD 4538 monté en monostable non redéclenchable.

Mathématique & Informatique



Mathématique & Informatique

ALGÈBRE DE BOOLE

L'idée maîtresse de George **Boole** – mathématicien anglais (Lincoln 1815 - Cork 1864) – est de ramener un raisonnement logique à un calcul mathématique similaire à celui effectué en algèbre des nombres entiers. On parle de logique booléenne.
Les variables booléennes ne prennent que 2 valeurs : 0 ou 1
Les opérations booléennes fondamentales sont :

1) l'addition symbolisée par le signe + : $x + y$
2) la multiplication symbolisée par le signe • ou par l'abscence de signe : $x \cdot y$ ou xy
3) la complémentation symbolisée par le signe ¯ ou par le signe ' : $\bar{x}$ ou x'

L'ensemble B = {0,1} muni des lois +, •, ¯ est une algèbre de **Boole** notée (B, +, •, ¯)

Propriétés élémentaires

$\bar{0} = 1$
$\bar{1} = 0$
$\bar{\bar{x}} = x$
$x + 1 = 1$
$x + 0 = x$
$x\,1 = x$
$x\,0 = 0$
$x + x = x$
$x\,x = x$
$x + \bar{x} = 1$
$x\,\bar{x} = 0$

Règles d'absorption

$x + xy = x$

$x + \bar{x}y = x + y$

Règle de redondance

$xy + \bar{x}z = xy + \bar{x}z + yz$

(le terme yz est appelé consensus)

Lois de **De Morgan**

$\overline{x + y} = \bar{x}\,\bar{y}$

$\overline{xy} = \bar{x} + \bar{y}$

x	y	$\bar{x}$	$\bar{y}$	$\overline{x+y}$	$\bar{x}\,\bar{y}$	$\overline{xy}$	$\bar{x} + \bar{y}$
0	0	1	1	1	1	1	1
0	1	1	0	0	0	1	1
1	0	0	1	0	0	1	1
1	1	0	0	0	0	0	0

Une fonction booléenne de n variables est une application de $\{0,1\}^n$ dans {0,1}, il y en a donc 2^{2^n} différentes.
Donnons l'expression de quelques fonctions usuelles de 2 et 3 variables :

$x \downarrow y = \bar{x}\,\bar{y}$ (fonction de **Peirce** = 1 ssi x = 0 et y = 0)

$x \uparrow y = \bar{x} + \bar{y}$ (fonction de **Sheffer** = 1 ssi x = 0 ou y = 0)

$x \oplus y = x\bar{y} + \bar{x}y$ (fonction disjonction = 1 ssi $x \neq y$)

$x \odot y = xy + \bar{x}\bar{y}$ (fonction conjonction = 1 ssi x = y)

Propriété : $x \odot y = \overline{x \oplus y}$

$Maj(x,y,z) = xy + yz + xz$ (fonction majorité = 1 ssi 2 variables au moins valent 1)

$Min(x,y,z) = \bar{x}\bar{y} + \bar{y}\bar{z} + \bar{x}\bar{z}$ (fonction minorité = 1 ssi 1 variable au plus vaut 1)

Propriété : $Min(x,y,z) = \overline{Maj(x,y,z)}$

Mathématique & Informatique

Lien avec la logique des propositions

Une proposition est un énoncé susceptible d'être vrai ou faux.
Soient P, Q, R, … des propositions.

Proposition	Booléen
faux	0
vrai	1
P	p
$\neg$ P (non)	$\overline{p}$
P $\vee$ Q (ou)	$p + q$
P $\wedge$ Q (et)	pq
P $\Rightarrow$ Q (implication)	$\overline{p} + q$
P $\Leftrightarrow$ Q (équivalence)	$p \odot q$

Note : $P \Leftrightarrow Q = (P \Rightarrow Q) \wedge (Q \Rightarrow P) \equiv (\overline{p} + q)(\overline{q} + p) = \overline{p}\,\overline{q} + \overline{p}\,p + q\overline{q} + qp = pq + \overline{p}\,\overline{q} = p \odot q$

EXEMPLE

Montrer que :

P $\Rightarrow$ Q est équivalent à non Q $\Rightarrow$ non P (Contraposition)
Si P $\Rightarrow$ Q et Q $\Rightarrow$ R alors P $\Rightarrow$ R (Syllogisme)

Preuve

Remplaçons les propositions P, Q, R par les booléens p, q, r.
La proposition $(P \Rightarrow Q) \Leftrightarrow (\neg Q \Rightarrow \neg P)$ s'écrit :

$$(\overline{p} + q) \odot (\overline{\overline{q}} + \overline{p}) = (\overline{p} + q) \odot (\overline{p} + q) = (\overline{p} + q)(\overline{p} + q) + \overline{(\overline{p} + q)}\ \overline{(\overline{p} + q)} = (\overline{p} + q) + \overline{(\overline{p} + q)} = 1$$

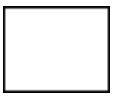
La proposition $\big((P \Rightarrow Q) \wedge (Q \Rightarrow R)\big) \Rightarrow (P \Rightarrow R)$ s'écrit :

$$\overline{(\overline{p} + q)(\overline{q} + r)} + (\overline{p} + r) = \overline{(\overline{p} + q)} + \overline{(\overline{q} + r)} + (\overline{p} + r) = (\overline{p} + p\overline{q}) + (r + \overline{r}\,q) = \overline{p} + \overline{q} + r + q = 1$$

Comme le booléen 1 est analogue à la proposition toujours vraie (Tautologie), le résultat est établi.

Lien avec la théorie des ensembles

Soit un ensemble E et A, B, C, … des sous-ensembles (parties) de E.

Ensemble	Booléen
 ∅ (vide)	0
 E (plein)	1
 A (partie)	a
 $\complement$A (complémentaire)	$\bar{a}$
 A ∪ B (union)	$a+b$
 A ∩ B (intersection)	ab
 A – B (différence)	$a\bar{b}$
 A Δ B (différence symétrique)	$a \oplus b$

Note : A Δ B = A ∪ B – A ∩ B ≡ $(a+b)\overline{ab} = (a+b)(\bar{a}+\bar{b}) = a\bar{a} + a\bar{b} + b\bar{a} + b\bar{b} = a\bar{b} + \bar{a}b = a \oplus b$

EXEMPLE

Montrer que : (A Δ B) Δ C = A Δ (B Δ C)

Preuve

Remplaçons les ensembles A, B, C par les booléens a, b, c.
Pour vérifier l'associativité de la loi Δ, il revient au même de vérifier l'associativité de la loi ⊕ :

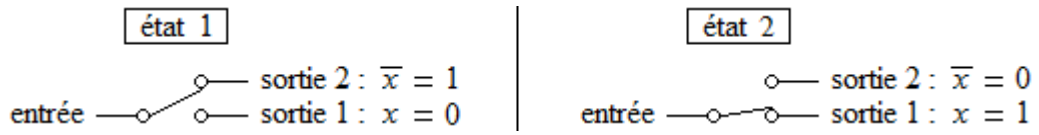
$$(a \oplus b) \oplus c = (a\bar{b} + \bar{a}b)\bar{c} + \overline{(a\bar{b} + \bar{a}b)}c = a\bar{b}\bar{c} + \bar{a}b\bar{c} + (\bar{a} + b)(a + \bar{b})c = abc + a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c$$

$$a \oplus (b \oplus c) = a\overline{(b\bar{c} + \bar{b}c)} + \bar{a}(b\bar{c} + \bar{b}c) = a(\bar{b} + c)(b + \bar{c}) + \bar{a}b\bar{c} + \bar{a}\bar{b}c = abc + a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c$$

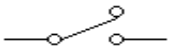
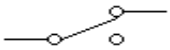
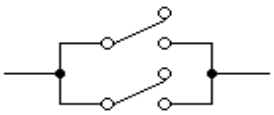
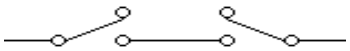
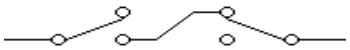
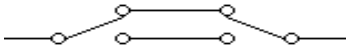
La relation booléenne $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ est équivalente à la relation ensembliste (A Δ B) Δ C = A Δ (B Δ C)

Lien avec les commandes électriques

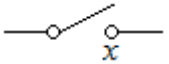
Un interrupteur inverseur (3 bornes) est un mécanisme à 1 entrée et 2 sorties. Il présente 2 états stables.
On associe la variable booléenne x à la première sortie et la variable booléenne complémentée $\bar{x}$ à la deuxième sortie.
Si le courant peut passer entre l'entrée et une sortie, la variable booléenne correspondante vaut 1, sinon elle vaut 0.

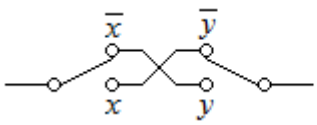


Sur les schémas, la sortie x est alignée avec l'entrée alors que la sortie $\bar{x}$ est décalée, ce qui permet de les différencier.
Une fonction booléenne $f(x_1, \ldots, x_n)$ de n variables est associée à un montage à n inverseurs : $f = 1 \Leftrightarrow$ le courant passe

Commande	Booléen
(ouvert)	0
(fermé)	1
 (sortie 1)	x
 (sortie 2)	$\bar{x}$
 (parallèle)	$x + y$
 (série)	xy
 (croisement)	$x\bar{y}$
 (va-et-vient)	$x \odot y$

Note

Un interrupteur simple (2 bornes) est un mécanisme à 1 entrée et 1 sortie (pas de sortie $\bar{x}$) : 

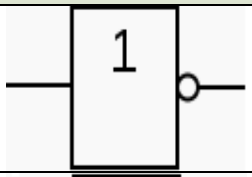
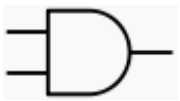
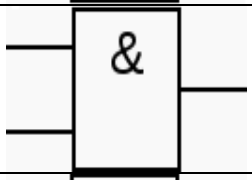
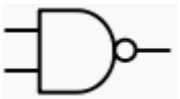
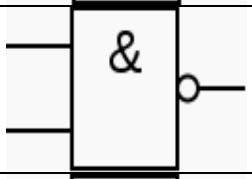
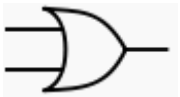
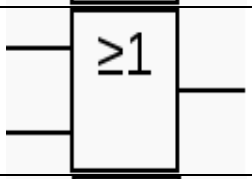
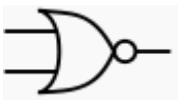
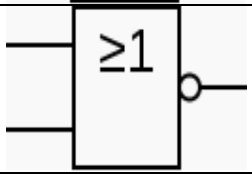
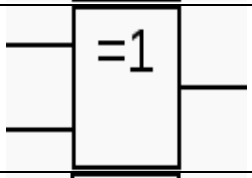
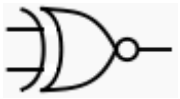
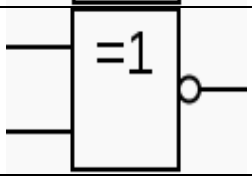
Autre montage va-et-vient équivalent (fils de navette croisés) : 

$x\overline{y} + \overline{x}\,y = x \oplus y$

Lien avec les circuits électroniques

Communément, le "0" de l'algèbre de **Boole** est représenté par une tension nulle et le "1" par une tension positive [(*)], en général + 5 volts. Les opérations booléennes de base sont réalisées avec des portes logiques (associations de transistors) possédant une entrée x et une sortie f(x) ou deux entrées x, y et une sortie f(x,y).

Les principales portes logiques

ANSI (américain)	CEI (européen)	fonction
		$\overline{x}$ (NOT)
		$x\,y$ (AND)
		$\overline{x\,y}$ $(NAND)$
		$x + y$ (OR)
		$\overline{x + y}$ (NOR)
		$x \oplus y$ (XOR)
		$\overline{x \oplus y}$ $(XNOR)$

Ces portes existent dans différentes technologies (familles TTL, ECL, CMOS et leurs diverses sous-familles).
On choisit une technologie plutôt qu'une autre en fonction de ses caractéristiques en matière de rapidité, de consommation, d'immunité aux bruits, de sortance, …
La famille TTL est la plus ancienne. La famille ECL est adaptée aux grandes vitesses (voir fibre optique).
La famille la plus utilisée est la famille CMOS en raison de sa faible consommation et de sa simplicité.

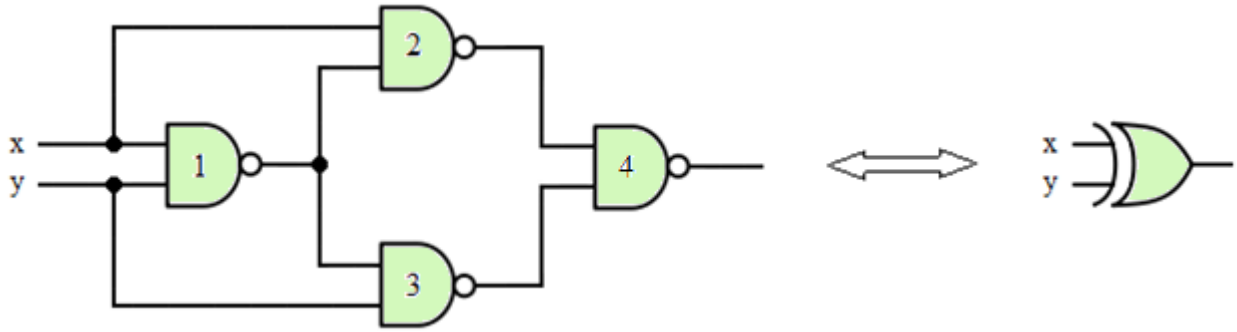
[(*)] Cette convention dépend de la technologie utilisée. Par exemple, en logique ECL, le "0" est à -1.6V et le "1" à -0.8V

Mathématique & Informatique

EXEMPLE 1

Montrer qu'il est possible d'obtenir l'équivalent logique d'une porte XOR avec 4 portes NAND.

Solution



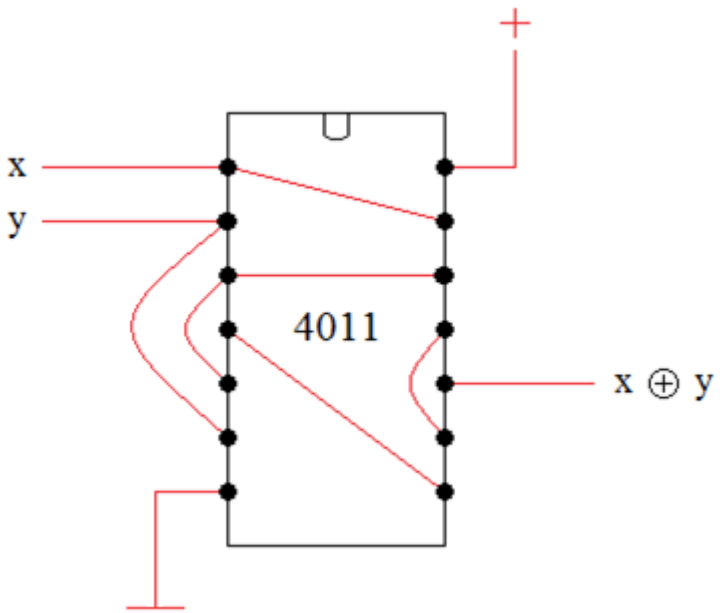
Soient z_1, z_2, z_3, z_4 les sorties des portes $NAND_1$, $NAND_2$, $NAND_3$, $NAND_4$:

$$z_1 = \overline{x\,y}$$

$$z_2 = \overline{x\,\overline{x\,y}}$$

$$z_3 = \overline{y\,\overline{x\,y}}$$

$$z_4 = \overline{\overline{x\,\overline{x\,y}}\;\overline{y\,\overline{x\,y}}} = x\overline{x\,y} + y\overline{x\,y} = x(\bar{x} + \bar{y}) + y(\bar{x} + \bar{y}) = x\bar{y} + \bar{x}y = x \oplus y$$

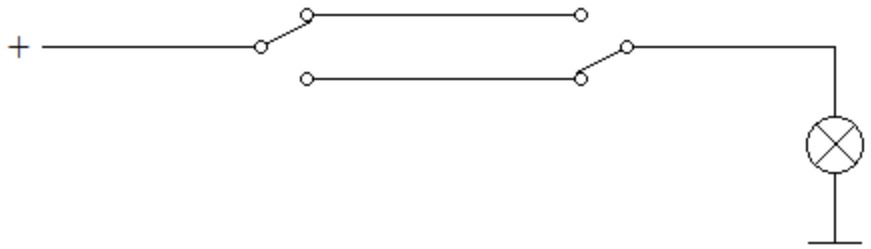


Remarque : en remplaçant les NAND par des NOR, on obtient une porte XNOR.

Attention
Les 2 montages sont équivalents sur le plan de la logique des circuits, mais ils diffèrent sur le plan électrique car la somme des temps de propagation des signaux à travers les portes NAND est supérieure au temps de propagation des signaux à travers la "vraie" porte XOR. En conséquence, la porte XOR "équivalente" fonctionnera à une fréquence plus faible.

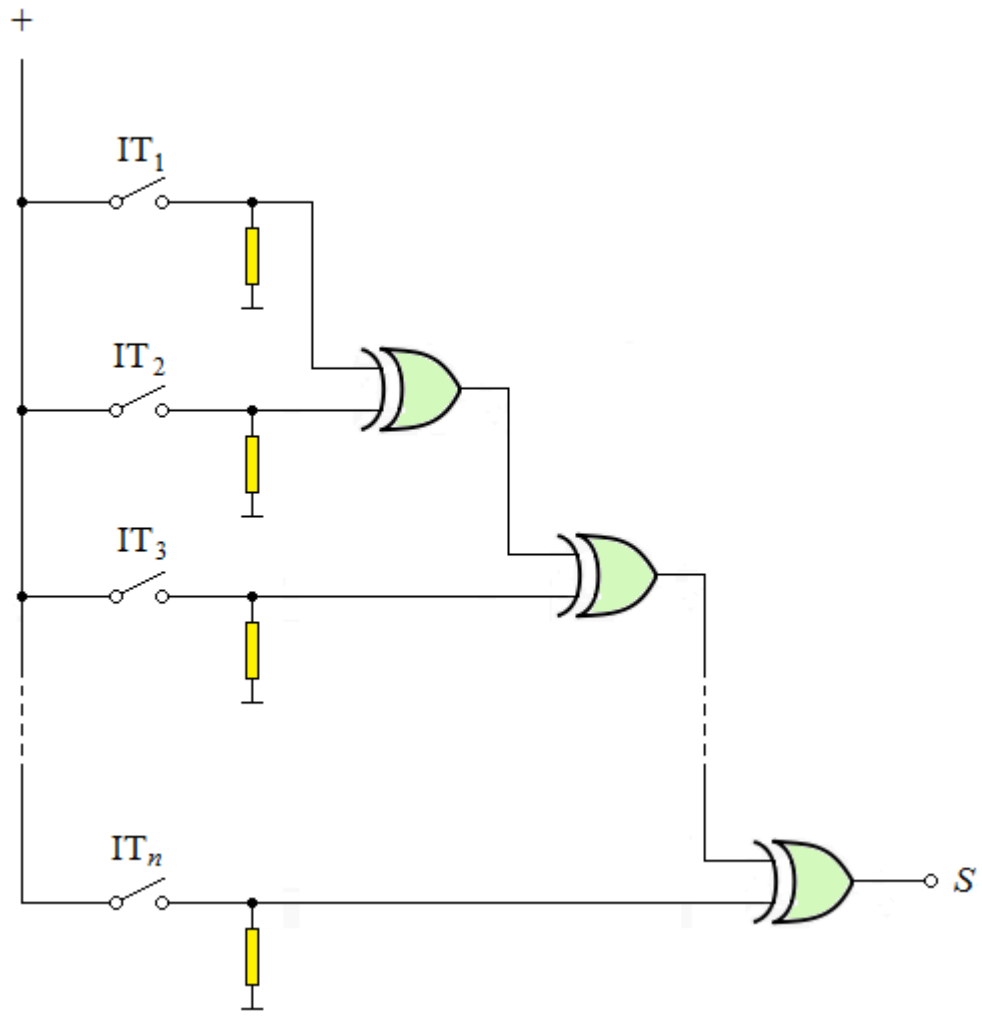
EXEMPLE 2

Un va-et-vient à 2 inverseurs est un montage électrique qui permet d'allumer ou d'éteindre une lampe à partir de 2 endroits. Le schéma classique est le suivant :



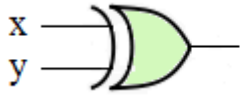
Réaliser un va-et-vient à *n* interrupteurs, uniquement avec des portes logiques.

Solution



En actionnant un interrupteur IT quelconque, la sortie *S* change d'état : elle bascule du plus (lampe allumée) à la masse (lampe éteinte) ou de la masse (lampe éteinte) au plus (lampe allumée). Noter que chaque interrupteur est soit ouvert, soit fermé, donc qu'il y a 2^n configurations possibles de l'ensemble des *n* interrupteurs.

Montrons que le montage précédent convient.
Rappel sur la porte XOR (OU exclusif) :



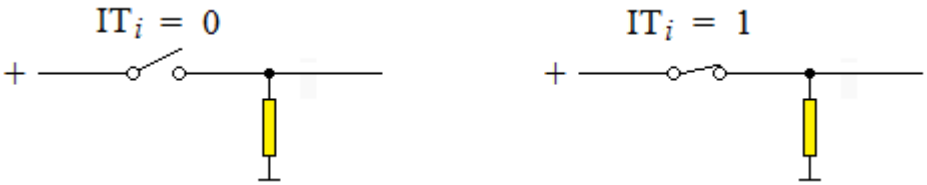
$$x \oplus y = x\overline{y} + \overline{x}\,y$$

La loi $\oplus$ est commutative et associative, c'est-à-dire :

$$x \oplus y = y \oplus x$$

$$(x \oplus y) \oplus z = x \oplus (y \oplus z)$$

Notons IT_i le booléen qui vaut 0 lorsque l'interrupteur n° i est ouvert et 1 lorsqu'il est fermé.



On a :

$$S = IT_1 \oplus IT_2 \oplus IT_3 \oplus \cdots \oplus IT_i \oplus \cdots \oplus IT_n = \bigoplus_{i=1}^{i=n} IT_i$$

Si l'interrupteur n° i est ouvert :

$$S = IT_1 \oplus IT_2 \oplus IT_3 \oplus \cdots \oplus 0 \oplus \cdots \oplus IT_n = \left[\bigoplus_{j=1,\, j\neq i}^{j=n} IT_j\right] \oplus 0 = \left[\bigoplus_{j=1,\, j\neq i}^{j=n} IT_j\right]\overline{0} + \overline{\left[\bigoplus_{j=1,\, j\neq i}^{j=n} IT_j\right]}\,0$$

$$= \bigoplus_{j=1,\, j\neq i}^{j=n} IT_j$$

Si l'interrupteur n° i est fermé :

$$S = IT_1 \oplus IT_2 \oplus IT_3 \oplus \cdots \oplus 1 \oplus \cdots \oplus IT_n = \left[\bigoplus_{j=1,\, j\neq i}^{j=n} IT_j\right] \oplus 1 = \left[\bigoplus_{j=1,\, j\neq i}^{j=n} IT_j\right]\overline{1} + \overline{\left[\bigoplus_{j=1,\, j\neq i}^{j=n} IT_j\right]}\,1$$

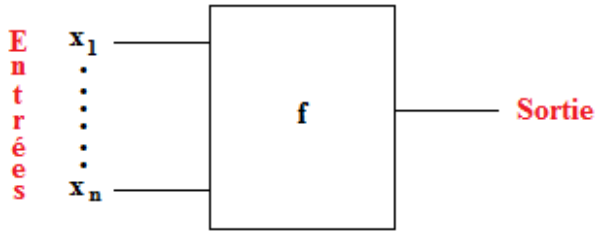
$$= \overline{\bigoplus_{j=1,\, j\neq i}^{j=n} IT_j}$$

Lorsque l'interrupteur n° i est ouvert puis fermé et inversement, le booléen IT_i passe de 0 à 1 et inversement.
Au cours de cette transition, la sortie S est complémentée (S est changée en $\overline{S}$).

C.Q.F.D.

Algorithme de Quine - McCluskey

Introduction



Soit f une fonction booléenne de n variables x_1 , ... , x_n définie par sa table de vérité :

x_1 ... x_n	f
0 ... 0	0/1
0 ... 1	0/1
: :	:
: :	:
1 ... 1	0/1

On écrit en abrégé :

$$f = \Sigma\ (b_1\ ,\ \dots\ ,\ b_k\ ,\ \dots\ ,\ b_N) \qquad 0 < N < 2^n$$

où les b_k sont les n-uplets binaires pour lesquels f vaut 1.

$$b_k = [x_{k,1}\ ,\ \dots\ ,\ x_{k,i}\ ,\ \dots\ ,\ x_{k,n}] \qquad x_{k,i} \in \{0,1\},\ 1 \le i \le n,\ 1 \le k \le N$$

Noter que l'on écarte le cas N = 0 correspondant à la fonction nulle (f = 0) et le cas $N = 2^n$ correspondant à la fonction unité (f = 1).
A chaque b_k on associe un monôme $m_k = m_{k,1} \dots m_{k,i} \dots m_{k,n}$ où $m_{k,i} = x_i$ si $x_{k,i} = 1$ et $m_{k,i} = \overline{x_i}$ si $x_{k,i} = 0$
La fonction f est égale à la somme des monômes m_k, elle s'exprime sous la forme d'un polynôme booléen P :

$$f(x_1\ ,\dots,x_n) = \sum_{k=1}^{N} m_k \ = \ \sum_{k=1}^{N}\prod_{i=1}^{n} m_{k,i} \ = \ P(x_1\ ,\dots,x_n\ ,\ \overline{x_1}\ ,\dots,\ \overline{x_n})$$

On souhaite simplifier cette expression algébrique appelée première forme canonique de f ou forme $\sum\prod$ de f.
L'ensemble des monômes m_k constitue la base canonique de f.
Un monôme est dit premier s'il ne possède pas de diviseur qui couvre partiellement f.
L'ensemble de tous les monômes premiers de f est la base première de f.
On appelle base irrédondante de f un ensemble minimal de monômes premiers dont la somme couvre entièrement f.
En général, il existe plusieurs bases irrédondantes qui n'ont pas nécessairement le même nombre de monômes premiers.
Parmi les bases irrédondantes, celle qui contient le moins de monômes premiers est une base minimale.
La méthode de **Quine - McCluskey** consiste, premièrement, à trouver tous les monômes premiers de f et, deuxièmement, à construire une base irrédondante de f.
Le temps d'exécution de l'algorithme croît de manière exponentielle avec le nombre de variables (problème NP-complet).

Remarque
D'autres méthodes existent : les tableaux de **Karnaugh** (pratique jusqu'à 4 variables), les consensus de **Tison**, ...
La méthode de **Quine - McCluskey** n'est pas la plus rapide lorsqu'on fait les calculs manuellement, mais elle se prête bien à la programmation sur ordinateur qui est quasi indispensable lorsque f possède plus de 5 variables. Pour un grand nombre de variables (plusieurs dizaines), le temps de calcul est souvent rédhibitoire et on utilise d'autres procédés basés sur des heuristiques pour approcher au mieux la solution optimale : minimiseur EXPRESSO (Robert **Brayton** à Berkeley).
⇨ **http://www.lirmm.fr/~pravo/cours/Logique/Polycops-pdf/Chap5.pdf**

Mathématique & Informatique

Terminologie, conventions et notations

Une base désigne un ensemble $\{m_k\}$ de monômes dont la somme couvre une fonction booléenne f :

$$f = \sum_{k=1}^{N} m_k$$

Le terme de "base" se retrouve dans des domaines variés des mathématiques et de l'informatique avec des significations différentes. Par exemple : base d'un cône ou base d'un trapèze en géométrie, base orthonormale en algèbre linéaire, base 10 en arithmétique ; et aussi : base de données d'un SGBD, base de faits ou base de règles d'un système expert, …

Appelons *McCluskey* la fonction Scilab qui fournit l'expression simplifiée de f en fonction des variables x_i ($1 \le i \le n$)

```
function [fc,fp,fi] = McCluskey(f)
/
/
endfunction
```

Le paramètre d'entrée de la fonction Scilab *McCluskey* est noté *f*, les paramètres de sortie sont notés *fc*, *fp*, *fi* :

- *f* est la liste des vecteurs de $\{-1,0,1\}^n$ représentant les monômes d'une base quelconque de $f(x_1, \ldots, x_n)$
- *fc* est la liste des vecteurs de $\{-1,0,1\}^n$ représentant les monômes de la base canonique de $f(x_1, \ldots, x_n)$
- *fp* est la liste des vecteurs de $\{-1,0,1\}^n$ représentant les monômes premiers de la base première de $f(x_1, \ldots, x_n)$
- *fi* est la liste des vecteurs de $\{-1,0,1\}^n$ représentant les monômes premiers d'une base irrédondante de $f(x_1, \ldots, x_n)$

Le paramètre *f* de *McCluskey* est passé sous la forme d'une liste de N vecteurs de même dimension n :

$f = list([x_{1,1}, \ldots, x_{1,i}, \ldots, x_{1,n}], \ldots, [x_{k,1}, \ldots, x_{k,i}, \ldots, x_{k,n}], \ldots, [x_{N,1}, \ldots, x_{N,i}, \ldots, x_{N,n}])$ avec $x_{k,i} \in \{-1,0,1\}$

Trois cas peuvent se présenter :

Cas 1
La fonction booléenne f est donnée par sa table de vérité.
Le paramètre *f* de *McCluskey* est la liste des vecteurs binaires de même dimension représentant les nombres binaires pour lesquels la fonction f vaut 1.
Exemple : le nombre binaire 01101 correspond au vecteur [0,1,1,0,1]

Cas 2
La fonction booléenne f est donnée par sa forme canonique.
Le paramètre *f* de *McCluskey* est la liste des vecteurs binaires de même dimension représentant les monômes de l'expression algébrique de f en notant 1 une variable non complémentée x_i et 0 une variable complémentée $\overline{x_i}$.
Exemple : le monôme $x_1 \overline{x_2} x_3 \overline{x_4} \overline{x_5}$ correspond au vecteur [1,0,1,0,0]

Cas 3
La fonction booléenne f est donnée par une expression algébrique quelconque.
Le paramètre *f* de *McCluskey* est la liste des vecteurs de même dimension représentant les monômes de l'expression algébrique de f avec les conventions suivantes : une variable non complémentée x_i est notée 1, une variable complémentée $\overline{x_i}$ est notée 0, une variable abscente est notée -1.
Exemple : le monôme $x_2 \overline{x_3} x_5$ correspond au vecteur [-1,1,0,-1,1]

Par convention, la fonction nulle est représentée par le paramètre *f = list()*, c'est-à-dire la liste vide puisqu'il n'existe aucun point où f vaut 1, la fonction unité par le paramètre *f = list([-1, ... , -1])*, c'est-à-dire la liste constituée du vecteur ligne de dimension n dont toutes les composantes sont égales à -1 puisqu'aucune variable x_i ou $\overline{x_i}$ ne figure dans l'expression de f. Pour simplifier et avoir une unique représentation de la fonction unité, on décide de lui attribuer le paramètre *f = list(-1)*.

Précisons que la fonction *McCluskey* ne retourne qu'une seule base irrédondante *fi*, celle qui minimise l'expression logique de f. Pour obtenir toutes les bases irrédondantes, on peut effectuer un examen exhaustif de toutes les sommes possibles de monômes premiers.

Mathématique & Informatique

Les 4 fonctions booléennes d'une variable :

$f = 0$	*f = list()*
$f = 1$	*f = list(-1)*
$f = \overline{x_1}$	*f = list(0)*
$f = x_1$	*f = list(1)*

Les 16 fonctions booléennes de deux variables :

$f = 0$	*f = list()*
$f = 1$	*f = list(-1)*
$f = \overline{x_1}$	*f = list([0,-1])*
$f = x_1$	*f = list([1,-1])*
$f = \overline{x_2}$	*f = list([-1,0])*
$f = x_2$	*f = list([-1,1])*
$f = \overline{x_1}\,\overline{x_2}$	*f = list([0,0])*
$f = \overline{x_1}\,x_2$	*f = list([0,1])*
$f = x_1\,\overline{x_2}$	*f = list([1,0])*
$f = x_1\,x_2$	*f = list([1,1])*
$f = \overline{x_1} + \overline{x_2}$	*f = list([0,-1],[-1,0])*
$f = \overline{x_1} + x_2$	*f = list([0,-1],[-1,1])*
$f = x_1 + \overline{x_2}$	*f = list([1,-1],[-1,0])*
$f = x_1 + x_2$	*f = list([1,-1],[-1,1])*
$f = x_1\,x_2 + \overline{x_1}\,\overline{x_2}$	*f = list([1,1],[0,0])*
$f = x_1\,\overline{x_2} + \overline{x_1}\,x_2$	*f = list([1,0],[0,1])*

Une expression booléenne faisant intervenir 2 variables x_1 et x_2 se ramène toujours à l'une des 16 formes du tableau ci-dessus.

Par exemple : $f = x_1 + x_1\,x_2 + \overline{x_1}\,\overline{x_2}$ ⇔ *f = list([1,-1],[1,1],[0,0])*

se ramène à : $f = x_1 + \overline{x_2}$ ⇔ *f = list([1,-1],[-1,0])*

Une fonction booléenne f s'écrit sous la forme d'une somme de monômes constitués chacun de la juxtaposition de variables x_i ou $\overline{x_i}$, un indice i > 0 n'apparaissant en principe qu'une seule fois dans un monôme. Un monôme peut également être représenté par la constante 0 ou par la constante 1.
Par exemple, les fonctions $f_1 = x_3x_2x_1x_2 + \overline{x_1}x_1x_4$, $f_2 = 0 + x_5$, $f_3 = \overline{x_2} + 1$ sont identiques à $f_1 = x_1x_2x_3$, $f_2 = x_5$, $f_3 = 1$

Ajoutons que par commodité d'écriture dans les programmes, la variable x_i sera notée xi et la variable $\overline{x_i}$ (x_i barre) sera notée xi' (xi prime), cette dernière écriture étant d'ailleurs la notation habituelle dans un treillis complémenté.

Par exemple : $x_1\,\overline{x_2} + \overline{x_1}\,x_2\,\overline{x_3}$ (tapuscrit) ⇔ x1x2′ + x1′ x2 x3′ (programme)

On contrôlera la validité d'une formule booléenne avec la fonction Scilab *regexp* qui utilise les expressions régulières du langage Perl dont on peut lire un résumé de la syntaxe à l'adresse Web **http://perl.enstimac.fr/DocFr/perlrequick.html**
Avec nos conventions, l'expression régulière '/^\s*(0|1|(x[1-9]\d*"?)+)(\s*\+\s*(0|1|(x[1-9]\d*"?)+))*\s*$/' appliquée à une chaîne de caractères, permet de vérifier s'il s'agit d'une expression booléenne correcte.

Remarque

Soient mc_i , mp_j , mi_k les monômes représentés par les éléments *fc*(i), *fp*(j), *fi*(k) respectivement, autrement dit les monômes correspondant aux i^e, j^e, k^e vecteurs des listes *fc*, *fp*, *fi*. On a :

$$f(x_1,\ldots,x_n) = \sum_i mc_i = \sum_j mp_j = \sum_k mi_k$$

La dernière somme est une expression simplifiée de f.

Lorsque *fp = fi*, tous les monômes premiers de f sont obligatoires et la base irrédondante obtenue est unique (condition suffisante mais pas nécessaire). Pour vérifier cette hypothèse, après l'appel *[fc,fp,fi] = McCluskey(f)*, on exécutera l'instruction suivante : *if isequal(fp,fi) then, printf('\n Base irrédondante unique'), end*

Premier exemple, soit : $f(x_1,\ldots,x_n) = x_1 + x_2 + \cdots + x_n = \sum_{i=1}^{n} x_i$

Si l'on applique l'algorithme de **Quine – McCluskey** à f, on trouvera *fp = fi* pour tout n. Les x_i ($1 \le i \le n$) sont les n monômes premiers obligatoires de f, ils forment à la fois la base première et la seule base irrédondante de f.

Deuxième exemple, celui de la fonction parité : parite(x_1 , ... , x_n) = 1 si et seulement si le nombre de variables x_i à 1 est pair. Cette fonction possède 2^{n-1} monômes premiers et satisfait à la relation *fc = fp = fi*, donc tous les monômes de la base canonique sont premiers et ils constituent à la fois la base première et la seule base irrédondante de la fonction.

Principe de l'algorithme de **Quine - McCluskey**

Pour obtenir les monômes premiers, itérer : réduire 2 monômes m_1 et m_2 qui diffèrent uniquement d'une variable biforme x_k ($m_1 + m_2 = \alpha\, x_k\, \beta + \alpha\, \overline{x_k}\, \beta = \alpha\beta = m$)

Pour obtenir une base irrédondante, itérer : sélectionner un monôme premier *mp* et supprimer les valeurs binaires $x_1 \ldots x_n$ couvertes ($mp = 1 \Rightarrow f(x_1 , \ldots , x_n) = 1$)

Mise en oeuvre de l'algorithme de **Quine - McCluskey**

Pour trouver les monômes premiers de f, on part de la base canonique de la fonction et on effectue toutes les réductions possibles de deux monômes du type $\alpha\, x_k\, \beta$ et $\alpha\, \overline{x_k}\, \beta$ en un monôme $\alpha\beta$ (x_k est une variable et α, β sont des fonctions des variables $x_i \ne x_k$). Les monômes qui n'ont pas été réduits sont premiers et on les retire de la liste. On renouvelle l'opération avec les nouveaux monômes obtenus (en supprimant les doublons) et ainsi de suite jusqu'à ce que l'on ne puisse plus effectuer de réduction. On dispose alors de tous les monômes premiers de f.

Pour trouver une base irrédondante de f, on introduit la matrice booléenne M de p+1 lignes et q+1 colonnes, où p est le nombre de monômes premiers de la fonction f et q est le nombre de valeurs binaires $x_1 \ldots x_n$ pour lesquelles f vaut 1.

$$M(i,j) = \begin{cases} 1 \text{ si le } i^e \text{ monôme premier vaut 1 pour la } j^e \text{ valeur binaire} \\ 0 \text{ sinon} \end{cases} \qquad (1 \le i \le p\,, 1 \le j \le q)$$

La $(p+1)^e$ ligne et la $(q+1)^e$ colonne de M servent à indiquer les choix possibles : M(i,q+1) = 1 (inversement 0) ou M(p+1,j) = 1 (inversement 0) signifie que la ligne i ou que la colonne j est (inversement n'est pas) sélectionnable.
La construction de la base irrédondante s'effectue pas à pas. Chaque étape se déroule en 2 phases : on sélectionne d'abord la valeur binaire $b_1 \ldots b_n$ qui est la moins couverte par des monômes premiers (i.e. la colonne de M qui possède le moins de "1" : phase MIN) puis on prend le monôme premier qui couvre le plus de valeurs binaires $x_1 \ldots x_n$ en incluant $b_1 \ldots b_n$ (i.e. la ligne de M qui possède le plus de "1" : phase MAX). Au fur et à mesure que l'on choisit des monômes premiers pour couvrir f, on marque les lignes et les colonnes de M devenues inutiles dans la poursuite du calcul.
Lorsque toutes les colonnes sont marquées, on dispose d'une base irrédondante de monômes premiers de f.

Programmation de l'algorithme de **Quine – McCluskey**

```
function [fc,fp,fi] = McCluskey(f)

    // Base canonique
    function fc = canonique(f)
        fc = list()
        if f == list() then, return, end
        N = length(f)
        n = length(f(1))
        X = zeros(1,n)
        I = ones(1,n)
        T = [zeros(n,1),ones(n,1)]
        loop = %t
        test = %t
        while loop
            if test then
                for i = 1:n
                    X(i) = T(i,I(i))
                end
                for i = 1:N
                    m = f(i)
                    val = 1
                    for j = 1:n
                        if m(j) + X(j) == 1 then
                            val = 0
                            break
                        end
                    end
                    if val == 1 then
                        fc($+1) = X
                        break
                    end
                end
                p = n + 1
            else
                I(p) = 1
            end
            p = p - 1
            I(p) = I(p) + 1
            test = I(p) <= 2
            loop = test | p > 1
        end
    endfunction
```

```
// Base première
function fp = premiere(f)
    fp = list()
    if f == list() then, return, end
    fr = fc
    while length(fr) <> 0
        ft = fr
        It = ones(1,length(fr))
        fr = list()
        for i = 1:length(ft)-1
            for j = i+1:length(ft)
                m1 = ft(i)
                m2 = ft(j)
                m = []
                p = 0
                for q = 1:length(m1)
                    if m1(q) <> m2(q) then
                        if p <> 0 then
                            p = 0
                            break
                        end
                        p = q
                    end
                end
                if p <> 0 then
                    if m1(p) + m2(p) == 1 then
                        m = [m1(1:p-1),-1,m1(p+1:$)]
                    end
                end
                if m <> [] then
                    It(i) = 0
                    It(j) = 0
                    existe = %f
                    for k = 1:length(fr)
                        existe = isequal(fr(k),m)
                        if existe then, break, end
                    end
                    if ~existe then
                        fr($+1) = m
                    end
                end
            end
        end
        for k = find(It == 1)
            fp($+1) = ft(k)
        end
    end
    if length(fp) == 1 then
        if find(fp(1) <> -1) == [] then
            fp = list(-1)
        end
    end
endfunction
```

```
// Base irrédondante
function fi = irredondante(f)
    fi = list()
    if f == list() then, return, end
    p = length(fp)
    q = length(fc)
    M = ones(p+1,q+1)
    for i = 1:p
        m1 = fp(i)
        for j = 1:q
            m2 = fc(j)
            for k = 1:length(m1)
                if m1(k) + m2(k) == 1 then
                    M(i,j) = 0
                    break
                end
            end
        end
    end
    ajout = %t
    while ajout
        ajout = %f
        Min = %inf
        Imin = []
        for j = 1:q
            if M(p+1,j) == 0 then, continue, end
            I = find(M(1:p,j) == 1)
            if I <> [] & length(I) < Min then
                Min = length(I)
                Imin = I
            end
        end
        Max = 0
        imax = 0
        if Imin <> [] then
            for k = 1:length(Imin)
                if M(Imin(k),q+1) == 0 then, continue, end
                J = find(M(Imin(k),1:q) == 1)
                if length(J) > Max then
                    Max = length(J)
                    imax = Imin(k)
                end
            end
        end
        if imax <> 0 then
            M(imax,q+1) = 0
            fi($+1) = fp(imax)
            ajout = %t
            for j = 1:q
                if M(imax,j) == 1 then
                    M(p+1,j) = 0
                    for i = 1:p
                        M(i,j) = 0
                    end
                end
            end
        end
    end
endfunction
```

fc = canonique(f)
fp = premiere(f)
fi = irredondante(f)

endfunction

Sur le caractère NP-difficile de l'algorithme de **Quine - McCluskey**

On peut construire une fonction booléenne avec une base canonique ne contenant aucun monôme premier, une base première et une base irrédondante possédant chacune un nombre exponentiel de monômes premiers.
Considérons la fonction $f_n(x_1 , ... , x_n)$ de n variables dont la forme canonique est la somme de tous les monômes s'écrivant avec $E\left(\frac{n}{2}\right)$ variables complémentées $\overline{x_i}$ et n - $E\left(\frac{n}{2}\right)$ variables non complémentées $x_j \neq x_i$ $(1 \leq i,j \leq n)$, ou avec $E\left(\frac{n}{2}\right) + 1$ variables complémentées $\overline{x_i}$ et n - $E\left(\frac{n}{2}\right)$ - 1 variables non complémentées $x_j \neq x_i$ $(1 \leq i,j \leq n)$.

La base canonique de la fonction f_n est constituée de $C_n^{E\left(\frac{n}{2}\right)} + C_n^{E\left(\frac{n}{2}\right)+1} = C_{n+1}^{E\left(\frac{n}{2}\right)+1}$ monômes différents.

L'application de la formule de réduction $\alpha\, x_k\, \beta + \alpha\, \overline{x_k}\, \beta = \alpha\beta$ à cet ensemble de monômes donne en une seule passe la base première de f_n , c'est-à-dire les $n\, C_{n-1}^{E\left(\frac{n}{2}\right)}$ monômes premiers de la fonction f_n.

Une sélection de $C_n^{E\left(\frac{n+1}{2}\right)}$ d'entre eux permet d'obtenir une base irrédondante de f_n.

On établit que : $n\, C_{n-1}^{E\left(\frac{n}{2}\right)} \underset{n\infty}{\sim} \sqrt{\frac{n}{2\pi}}\, 2^n \xrightarrow[n\infty]{} \infty$ et que : $C_n^{E\left(\frac{n+1}{2}\right)} \underset{n\infty}{\sim} \sqrt{\frac{2}{n\pi}}\, 2^n \xrightarrow[n\infty]{} \infty$

Par exemple, prenons la fonction $f_4(x_1 , x_2 , x_3 , x_4)$.
La base canonique de f_4 est :

$$\overline{x_1}\,\overline{x_2}\,x_3\,x_4 , x_1\,\overline{x_2}\,x_3\,x_4 , \overline{x_1}\,x_2\,x_3\,x_4 , x_1\,x_2\,x_3\,x_4 , \overline{x_1}\,\overline{x_2}\,\overline{x_3}\,x_4 , x_1\,\overline{x_2}\,\overline{x_3}\,x_4 , x_1\,x_2\,\overline{x_3}\,\overline{x_4} ,$$
$$\overline{x_1}\,\overline{x_2}\,\overline{x_3}\,x_4 , x_1\,\overline{x_2}\,\overline{x_3}\,\overline{x_4} , \overline{x_1}\,x_2\,\overline{x_3}\,\overline{x_4} , x_1\,x_2\,\overline{x_3}\,\overline{x_4} , \overline{x_1}\,\overline{x_2}\,x_3\,\overline{x_4} , x_1\,\overline{x_2}\,x_3\,\overline{x_4} , \overline{x_1}\,x_2\,x_3\,\overline{x_4} , x_1\,x_2\,x_3\,\overline{x_4}$$

La base première de f_4 est :

$$\overline{x_1}\,\overline{x_2}\,x_4 , x_1\,\overline{x_2}\,x_3 , \overline{x_1}\,x_2\,x_3 , x_1\,x_2\,x_4 , \overline{x_1}\,\overline{x_3}\,x_4 , x_1\,\overline{x_3}\,x_4 , \overline{x_2}\,\overline{x_3}\,x_4 , x_1\,\overline{x_2}\,\overline{x_3} , x_2\,\overline{x_3}\,x_4 , x_1\,\overline{x_2}\,x_4 , \overline{x_2}\,x_3\,x_4 , x_1\,x_3\,x_4$$

Une base irrédondante de f_4 est :

$$\overline{x_1}\,\overline{x_2}\,x_3 , x_1\,\overline{x_2}\,x_3 , x_1\,x_2\,x_4 , \overline{x_1}\,x_3\,x_4 , x_2\,\overline{x_3}\,\overline{x_4} , \overline{x_2}\,x_3\,x_4 , x_1\,\overline{x_3}\,\overline{x_4}$$

Pour n = 100, la fonction f_{100} de 100 variables possède $100\, C_{99}^{50} = 50445672272782096667406248628 00$ monômes premiers dont $C_{100}^{50} = 100891344545564193334812497256$ d'entre eux forment une base irrédondante.
On conçoit dès lors qu'il n'est guère envisageable d'utiliser l'algorithme de **Quine – McCluskey** pour la fonction f_{100} .
D'une manière générale, tout algorithme basé sur la recherche exhaustive des monômes premiers d'une fonction booléenne ayant un grand nombre de variables se révélera inefficace en raison du temps de calcul nécessaire.

En pratique, l'algorithme de **Quine – McCluskey** convient pour trouver une base irrédondante d'une fonction booléenne de 1 à 10 variables, donc pour fournir l'expression simplifiée des $\sum_{k=1}^{10} 2^{2^k} \approx 1.8 \times 10^{309}$ premières fonctions booléennes.

Remarque

Soit N_n la borne supérieure du nombre de monômes premiers d'une fonction booléenne $f(x_1 , ... , x_n)$ de n variables.
Ecrivons : $f(x_1, ... ,x_n) = x_1\, g(x_2 , ... ,x_n) + \overline{x}_1\, h(x_2, ... ,x_n) = x_1\, g(x_2, ... ,x_n) + \overline{x}_1\, h(x_2, ... ,x_n) + g(x_2, ... ,x_n) \times h(x_2, ... ,x_n)$
Notons n_f , n_g , n_h , n_{gh} le nombre de monômes premiers des fonctions booléennes f, g, h, gh. On a :

$$n_f \le n_g + n_h + n_{gh} \quad \Rightarrow \quad N_n \le 3N_{n-1} \le \ldots \le 3^{n-1}N_1$$

Comme une fonction booléenne d'une variable n'a pas plus d'un monôme premier ($N_1 = 1$), il vient :

$$N_n \le 3^{n-1}$$

En particulier, une fonction booléenne de 10 variables a toujours moins de $3^9 = 19683$ monômes premiers.
Le nombre de valeurs binaires $x_1 \ldots x_n$ pour lesquelles $f(x_1, ... ,x_n)$ vaut 1 est inférieur ou égal à 2^n ; pour n =10, une fonction booléenne de 10 variables vaut 1 en au plus $2^{10} = 1024$ points.
Par conséquent, la matrice M précédemment définie dans la mise en œuvre de l'algorithme de **Quine – McCluskey** a une dimension inférieure à 19683×1024 pour n = 10 (il faudra augmenter si besoin la taille de la pile Scilab).

On peut rechercher le nombre maximal exact de monômes premiers d'une fonction booléenne de n variables (il suffit d'examiner les 2^{2^n} fonctions possibles, n petit !). On trouve :

Nombre de variables de la fonction booléenne	Nombre maximal exact de monômes premiers
1	1
2	2
3	6
4	13
5	32

Ainsi, une fonction booléenne de 5 variables n'a pas plus de 32 monômes premiers.
Ce résultat permet d'affiner le majorant du nombre de monômes premiers d'une fonction booléenne de 10 variables.

$$N_{10} \le 3^5\, N_5 = 243 \times 32 = 7776 < 7777$$

D'autre part, on a vu qu'une base irrédondante d'une fonction booléenne de n variables peut être constituée d'au moins 2^{n-1} monômes premiers (prendre la fonction parité par exemple). En particulier, une base irrédondante d'une fonction booléenne de 10 variables peut avoir au moins $2^9 = 512$ monômes premiers (ce nombre est-il maximal, c'est-à-dire existe-t-il une fonction booléenne de 10 variables dont la base irrédondante minimale a plus de 512 monômes premiers ?).

Ecriture de l'expression algébrique d'une fonction booléenne

Soit f une fonction booléenne de n variables x_i ($1 \le i \le n$) mise sous la forme d'une somme de monômes m_k.
La fonction *chaine(f)* crée la chaîne de caractères *s* correspondant à l'expression algébrique de f à partir du paramètre *f* représentant la liste des monômes m_k.
Les monômes apparaîtront triés en premier lieu dans l'ordre croissant du nombre de variables x_i et $\overline{x}_i$, en second lieu dans l'ordre croissant des indices i, en troisième lieu dans l'ordre croissant du nombre de variables complémentées $\overline{x}_i$.

Par exemple :

s = chaine(list([-1,-1,-1,1,-1,0,0,-1,-1,-1],[-1,-1,0,-1,1,-1,-1,-1,-1,1],[1,-1,-1,-1,-1,-1,-1,-1,-1,-1])) ;

crée la chaîne de caractères *s = 'x1 + x3''x5x10 + x4x6''x7'''* correspondant à la fonction $f = x_1 + \overline{x_3}\, x_5\, x_{10} + x_4\, \overline{x_6}\, \overline{x_7}$

```
function s = chaine(f)
    // Cas particulier f = 0 ou f = 1
    select f
        case list() then, s = '0', return
        case list(-1) then, s = '1', return
    end
    // Tri des monômes
    j = 1
    while j < length(f)
        i = j
        while i >= 1
            a = find(f(i) <> -1)
            b = find(f(i+1) <> -1)
            sup = length(a) > length(b)
            if length(a) == length(b) then
                if isequal(a,b) then
                    c = find(f(i) == 0)
                    d = find(f(i+1) == 0)
                    sup = length(c) > length(d)
                else
                    for k = 1:length(a)
                        sup = a(k) > b(k)
                        if a(k) <> b(k) then, break, end
                    end
                end
            end
            if sup then
                t = f(i+1)
                f(i+1) = f(i)
                f(i) = t
                i = i - 1
            else
                i = 0
            end
        end
        j = j + 1
    end
    // Expression algébrique
    s = ''
    for i = 1:length(f)
        if i <> 1 then, s = s + ' + ', end
        m = f(i)
        for j = 1:length(m)
            select m(j)
                case 0 then, s = s + 'x' + string(j) + ''''
                case 1 then, s = s + 'x' + string(j)
            end
        end
    end
endfunction
```

Mathématique & Informatique

Lecture de l'expression algébrique d'une fonction booléenne

Il est inhabituel d'écrire une fonction booléenne f sous la forme d'une liste *f* de vecteurs d'entiers.
C'est pourquoi, on peut choisir de calculer automatiquement le paramètre *f* à partir d'une chaîne de caractères *s* dans laquelle la variable x_i est notée xi et la variable $\overline{x_i}$ est notée xi'.

NB Le signe ' (apostrophe), utilisé pour désigner une variable complémentée, doit être doublé dans la chaîne *s*.

Par exemple, pour obtenir le paramètre *f* associé à $f = x_1 + \overline{x_3}\, x_5\, x_{10} + x_4\, \overline{x_6}\, \overline{x_7}$, on écrit l'instruction :

```
f = liste('x1 + x3''x5x10 + x4x6''x7''') ;
```

Ce qui est équivalent à :

```
f = list([1,-1,-1,-1,-1,-1,-1,-1,-1,-1],[-1,-1,0,-1,1,-1,-1,-1,-1,-1,1],[-1,-1,-1,1,-1,0,0,-1,-1,-1]) ;
```

La fonction de conversion est la suivante :

```
function f = liste(s)
    s = strsubst(s,' ','')
    S = strsplit(s,'+')
    n = 0
    F = list()
    for k = 1:size(S,1)
        m = S(k)
        if isequal(m,'0') then
            continue
        end
        if isequal(m,'1') then
            f = list(-1)
            return
        end
        m = strsubst(m,'''','*-1')
        m = strsplit(m,'x')
        m = evstr(m)
        m = unique(m)
        uam = unique(abs(m))
        if size(uam,1) == size(m,1) then
            n = max([n;uam])
            F($+1) = m
        end
    end
    f = list()
    for k = 1:length(F)
        v = - ones(1,n)
        m = F(k)
        for i = 1:size(m,1)
            if m(i) < 0 then
                v(-m(i)) = 0
            else
                v(m(i)) = 1
            end
        end
        f($+1) = v
    end
endfunction
```

Mathématique & Informatique

Table de vérité d'une fonction booléenne

Soit f une fonction booléenne de n variables dont le paramètre *f* a été défini. Programmons :

1) une fonction Scilab qui retourne la valeur *y* prise par f en un point $X \in \{0,1\}^n$
2) une fonction Scilab qui fournit la table de vérité T de f

```
function y = valeur(f,X)
    y = 0
    for i = 1:length(f)
        m = f(i)
        y = 1
        for j = 1:length(m)
            if m(j) + X(j) == 1 then
                y = 0
                break
            end
        end
        if y == 1 then, break, end
    end
endfunction
```

```
function T = verite(f,n)
    T = emptystr(2^n,1)
    for N = 0:2^n-1
        x = dec2bin(N,n)
        X = evstr(strsplit(x))'
        y = valeur(f,X)
        T(N+1) = x + ' / ' + string(y)
    end
endfunction
```

Par exemple, pour obtenir la table de vérité T de la fonction $f = \overline{x_1}\, x_3 + x_1 x_2 x_4$ de 4 variables, on écrit :

```
f = liste('x1''x3 + x1x2x4') ;
T = verite(f,4) ;
printf('\n %s',T)
```

On obtient :

```
0000 | 0
0001 | 0
0010 | 1
0011 | 1
0100 | 0
0101 | 0
0110 | 1
0111 | 1
1000 | 0
1001 | 0
1010 | 0
1011 | 0
1100 | 0
1101 | 1
1110 | 0
1111 | 1
```

Remarquer que l'expression algébrique de f peut contenir moins de variables que sa table de vérité n'en contient, d'où la nécessité de préciser le nombre n. Par exemple, pour la fonction f de n = 2 variables, d'expression $f = x_1$, on a :

```
00 | 0
01 | 0
10 | 1
11 | 1
```

EXEMPLE 1

Soit f la fonction booléenne de 4 variables définie par la table de vérité suivante :

$x_1x_2x_3x_4$	f
0000	1
0001	1
0010	1
0011	0
0100	0
0101	1
0110	0
0111	1
1000	1
1001	1
1010	1
1011	0
1100	0
1101	1
1110	0
1111	1

Réaliser une couverture minimale de f avec des monômes premiers.

Solution

On a :

$$f = \Sigma(0000,0001,0010,0101,0111,1000,1001,1010,1101,1111)$$

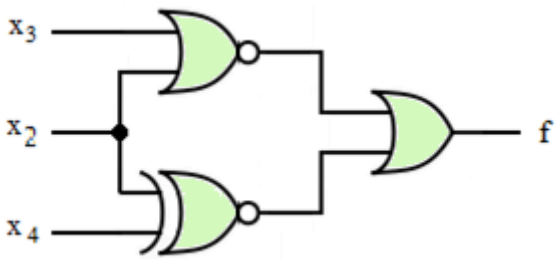
Pour obtenir l'expression algébrique recherchée, on écrit :

```
f = list([0,0,0,0],[0,0,0,1],[0,0,1,0],[0,1,0,1],[0,1,1,1],[1,0,0,0],[1,0,0,1],[1,0,1,0],[1,1,0,1],[1,1,1,1]) ;
[fc,fp,fi] = McCluskey(f) ;
printf('\n f = '), printf(chaine(fi))
```

Ce qui donne :

$$f = \bar{x}_2\,\bar{x}_3 + x_2\,x_4 + \bar{x}_2\,\bar{x}_4$$

Avec des portes logiques :



Note

L'expression de f n'est pas unique, on vérifie que $x_2\,x_4 + \bar{x}_2\,\bar{x}_4 + \bar{x}_3x_4$ est également solution.

EXEMPLE 2

Soit f la fonction booléenne de 5 variables définie par la forme canonique suivante :

$$f = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5 + \bar{x}_1\bar{x}_2x_3\bar{x}_4\bar{x}_5 + \bar{x}_1x_2\bar{x}_3\bar{x}_4\bar{x}_5 + x_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5 + x_1x_2x_3x_4\bar{x}_5 +$$
$$x_1x_2x_3\bar{x}_4\bar{x}_5 + x_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5 + x_1\bar{x}_2x_3\bar{x}_4\bar{x}_5 + x_1\bar{x}_2\bar{x}_3\bar{x}_4x_5 + x_1\bar{x}_2x_3\bar{x}_4x_5 +$$
$$x_1x_2x_3x_4x_5 + x_1x_2x_3\bar{x}_4x_5 + \bar{x}_1x_2\bar{x}_3\bar{x}_4x_5 + \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4x_5 + \bar{x}_1\bar{x}_2x_3\bar{x}_4x_5$$

Simplifier f avec l'algorithme de **McCluskey**.

Solution

Programme :

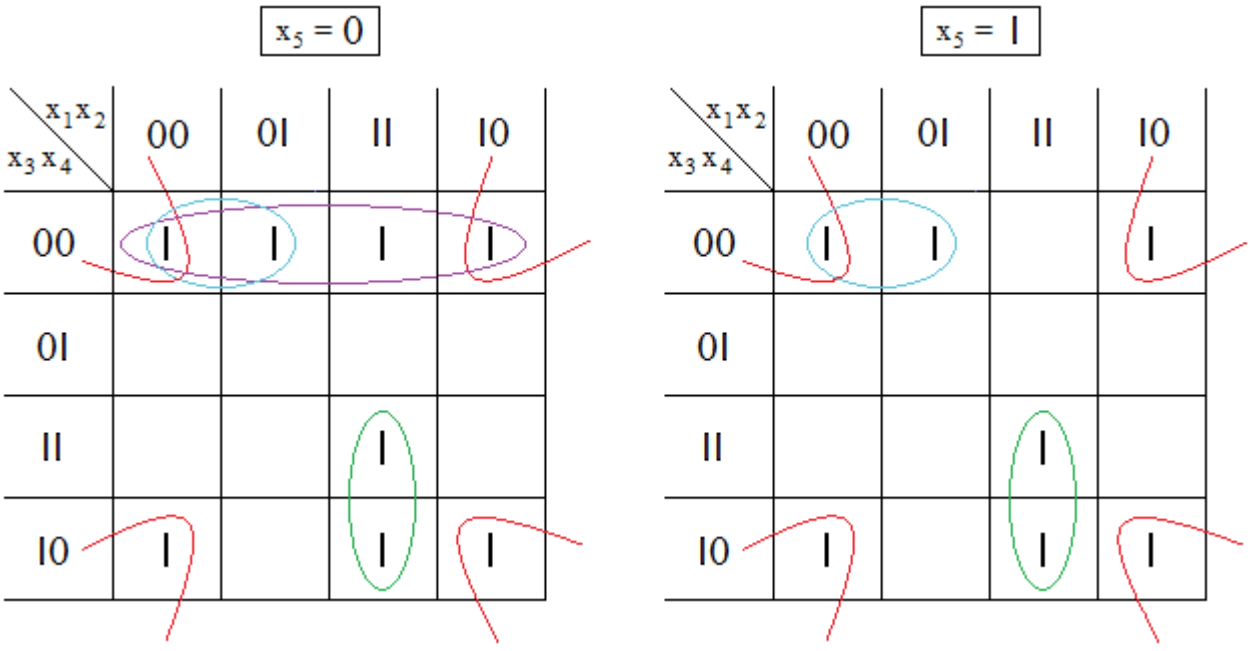
```
f = list([0,0,0,0,0],[0,0,1,0,0],[0,1,0,0,0],[1,1,0,0,0],[1,1,1,1,0], ...
         [1,1,1,0,0],[1,0,0,0,0],[1,0,1,0,0],[1,0,0,0,1],[1,0,1,0,1], ...
         [1,1,1,1,1],[1,1,1,0,1],[0,1,0,0,1],[0,0,0,0,1],[0,0,1,0,1]) ;
[fc,fp,fi] = McCluskey(f) ;
printf('\n f = '), printf(chaine(fi))
```

Résultat :

$$f = \bar{x}_2\,\bar{x}_4 + x_1\,x_2\,x_3 + \bar{x}_1\,\bar{x}_3\,\bar{x}_4 + \bar{x}_3\,\bar{x}_4\,\bar{x}_5$$

Note

On retrouve ce résultat avec la méthode graphique de **Karnaugh** en utilisant 2 tableaux de 16 cases superposés.



En —— le monôme premier $\bar{x}_2\,\bar{x}_4$
En —— le monôme premier $x_1\,x_2\,x_3$
En —— le monôme premier $\bar{x}_1\,\bar{x}_3\,\bar{x}_4$
En —— le monôme premier $\bar{x}_3\,\bar{x}_4\,\bar{x}_5$

EXEMPLE 3

Soient f_1, f_2, f_3 trois fonctions booléennes définies par les formules algébriques suivantes :

$$f_1 = x_1 x_2 x_3 + x_1 \overline{x_2} + x_1 \overline{x_3} + \overline{x_1}$$

$$f_2 = x_1 \overline{x_2}\ \overline{x_3} + \overline{x_1}\ \overline{x_3} x_4 + x_1 x_2 x_4 + x_2 x_3 x_4 + x_2 \overline{x_4}$$

$$f_3 = \overline{x_1}\ \overline{x_5} + \overline{x_1}\ \overline{x_6} + \overline{x_1}\ \overline{x_7} + \overline{x_1}\ \overline{x_8} + x_1 x_5 x_6 + \overline{x_1} x_7 x_8 + x_2 x_3 x_4 x_5 x_6 x_7 x_8$$

Minimiser l'expression logique de chacune de ces fonctions.

Solution

Les fonctions f_1, f_2, f_3 ne sont pas sous forme canonique (toutes les variables ne sont pas présentes dans les monômes). L'appel à la fonction Scilab *McCluskey* s'effectue, soit à partir d'une liste de vecteurs d'entiers où les variables manquantes sont remplacées par -1, soit directement à partir de l'expression algébrique assimilée à une chaîne de caractères.

```
f1 = liste('x1x2x3 + x1x2'' + x1x3'' + x1''') ;
[f1c,f1p,f1i] = McCluskey(f1) ;
printf('\n f1 = '), printf(chaine(f1i))

f2 = liste('x1x2''x3'' + x1''x3''x4 + x1x2x4 + x2x3x4 + x2x4''') ;
[f2c,f2p,f2i] = McCluskey(f2) ;
printf('\n f2 = '), printf(chaine(f2i))

f3 = liste('x1''x5'' + x1''x6'' + x1''x7'' + x1''x8'' + x1x5x6 + x1''x7x8 + x2x3x4x5x6x7x8') ;
[f3c,f3p,f3i] = McCluskey(f3) ;
printf('\n f3 = '), printf(chaine(f3i))
```

On trouve :

$$f_1 = 1$$

$$f_2 = x_2 + x_1 \overline{x_3} + \overline{x_3} x_4$$

$$f_3 = \overline{x_1} + x_5 x_6$$

Note

L'expression simplifiée de f_3 peut se retrouver avec les règles du calcul booléen (voir méthode des consensus de **Tison**) :

$f_3 = x_1x_5x_6 + \overline{x_1}x_7x_8 + x_2x_3x_4x_5x_6x_7x_8 + \overline{x_1}\,\overline{x_5} + \overline{x_1}\,\overline{x_6} + \overline{x_1}\,\overline{x_7} + \overline{x_1}\,\overline{x_8}$

Redondance par rapport à x_1 :

$f_3 = x_1x_5x_6 + \overline{x_1}x_7x_8 + x_5x_6x_7x_8 + x_5x_6x_7x_8(x_2x_3x_4) + \overline{x_5x_6x_7x_8}\,(\overline{x_1})$

Absorption par rapport à $x_5x_6x_7x_8$:

$f_3 = x_1x_5x_6 + \overline{x_1}x_7x_8 + x_5x_6x_7x_8 + \overline{x_1}$

Redondance par rapport à x_1 :

$f_3 = x_1x_5x_6 + \overline{x_1}x_7x_8 + \overline{x_1}$

Absorption par rapport à $\overline{x_1}$:

$f_3 = \overline{x_1} + x_5x_6$

EXEMPLE 4

Soit f = Diff(x_1 , ... , x_n) la fonction booléenne de n variables x_i définie ainsi :

f *vaut 1 lorsque le nombre de variables* x_i *à 0 est différent du nombre de variables* x_i *à 1*

Soit $x_1 \dots x_n$ un entier binaire. Par définition f(x_1 , ... , x_n) vaut 0 lorsque $x_1 \dots x_n$ s'écrit avec le même nombre p de 0 et de 1. Dans ce cas, on a nécessairement n = 2p, c'est-à-dire n pair. On en déduit que f vaut toujours 1 si n est impair. Rechercher une expression algébrique simplifiée de f pour n = 6.

Solution

Soit X = [x_1 , … , x_n] un n-uplet binaire ($x_i \in \{0,1\}$) et $N = x_1 \times 2^{n-1} + \dots + x_n \times 2^0$ le nombre entier décimal correspondant. La condition Card $\{x_i \in X / x_i = 0\} \neq$ Card $\{x_i \in X / x_i = 1\}$ s'écrit *length(find(X == 0)) <> length(find(X == 1))* en Scilab. Le paramètre *f* de la fonction *McCluskey* est la liste des vecteurs X qui remplissent la condition imposée parmi l'ensemble des 2^n vecteurs X possibles. Pour obtenir un vecteur X à partir d'un entier N, on utilise les fonctions Scilab suivantes :

- S = *dec2bin*(N,n) : convertit l'entier N en une chaîne S de n caractères 0 ou 1
- V = *strsplit*(S) : convertit la chaîne S en un vecteur colonne V de caractères
- Y = *evstr*(V) : convertit le vecteur V de caractères en un vecteur Y de nombres
- X = Y' : convertit le vecteur colonne Y en un vecteur ligne X (transposition)

D'où : X = *evstr*(*strsplit*(*dec2bin*(N,n)))'
Par exemple, N = 105 est équivalent à l'octuplet binaire X = *evstr*(*strsplit*(*dec2bin*(105,8)))' = [0,1,1,0,1,0,0,1]

Pour exprimer la fonction Diff(x_1,x_2,x_3,x_4,x_5,x_6), calculons d'abord le paramètre *f* :

```
f = list() ;
n = 6 ;
for N = 0:2^n-1
    X = evstr(strsplit(dec2bin(N,n)))' ;
    if length(find(X == 0)) <> length(find(X == 1)) then
        f($+1) = X ;
    end
end
```

Ensuite, les instructions *[fc,fp,fi] = McCluskey(f) ; printf('\n f = '), printf(chaine(fi))* permettent d'obtenir la fonction f sous la forme d'une somme de monônes premiers :

```
f = x1x2x3x4 + x1'x2'x3'x4' + x1x2x3x5 + x1'x2'x3'x5' + x1x2x3x6 + x1'x2'x3'x6' + x1x2x4x5 + x1'x2'x4'x5' +
    x1x2x4x6 + x1'x2'x4'x6' + x1x2x5x6 + x1'x2'x5'x6' + x1x3x4x5 + x1'x3'x4'x5' + x1x3x4x6 + x1'x3'x4'x6' +
    x1x3x5x6 + x1'x3'x5'x6' + x1x4x5x6 + x1'x4'x5'x6' + x2x3x4x5 + x2'x3'x4'x5' + x2x3x4x6 + x2'x3'x4'x6' +
    x2x3x5x6 + x2'x3'x5'x6' + x2x4x5x6 + x2'x4'x5'x6' + x3x4x5x6 + x3'x4'x5'x6'
```

Note

Dans l'expression de f, les notations xi et xi' ($1 \le i \le 6$) remplacent les notations x_i et $\bar{x}_i$ ($1 \le i \le 6$).

EXEMPLE 5

Soit f = Compose(x_1 , ... , x_n) la fonction booléenne qui retourne 1 si le nombre binaire x_1 … x_n est composé et 0 sinon.
Donner une expression algébrique simplifiée de la fonction f dans le cas n = 8.

Solution

Soit X = [x_1 , … , x_n] un n-uplet binaire et N le nombre entier décimal correspondant.
Parmi toutes les valeurs possibles de X, on retient celles pour lesquelles N est composé, c'est-à-dire N non premier.
Pour n = 8, on a 54 nombres premiers inférieurs à $2^8 - 1$:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139
149 151 157 163 167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251

D'où :

f = Σ(0, 1, 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25, 26, 27, 28, 30, 32, 33, 34, 35, 36, 38, 39, 40, 42, 44, 45, 46, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 60, 62, 63, 64, 65, 66, 68, 69, 70, 72, 74, 75, 76, 77, 78, 80, 81, 82, 84, 85, 86, 87, 88, 90, 91, 92, 93, 94, 95, 96, 98, 99, 100, 102, 104, 105, 106, 108, 110, 111, 112, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 128, 129, 130, 132, 133, 134, 135, 136, 138, 140, 141, 142, 143, 144, 145, 146, 147, 148, 150, 152, 153, 154, 155, 156, 158, 159, 160, 161, 162, 164, 165, 166, 168, 169, 170, 171, 172, 174, 175, 176, 177, 178, 180, 182, 183, 184, 185, 186, 187, 188, 189, 190, 192, 194, 195, 196, 198, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 224, 225, 226, 228, 230, 231, 232, 234, 235, 236, 237, 238, 240, 242, 243, 244, 245, 246, 247, 248, 249, 250, 252, 253, 254, 255)

Créons au préalable le tableau P des nombres premiers inférieurs à 2^8 - 1 :

```
fd = mopen("PREMIERS",'rb') ;
P = mget(54) ;
mclose(fd) ;
```

Le programme suivant calcule le paramètre *f* de la fonction *McCluskey* :

```
f = list() ;
n = 8 ;
for N = 0:2^n-1
    if find(P == N) == [] then
        X = evstr(strsplit(dec2bin(N,n)))' ;
        f($+1) = X ;
    end
end
```

Les instructions *[fc,fp,fi] = McCluskey(f) ; printf('\n f = '), printf(chaine(fi))* donnent :

f = x1x8' + x2x8' + x4x8' + x5x8' + x6x8' + x2x4x5'x6 + x1x2x3'x4'x5 + x1x2x3x4x6 + x1x2'x3'x4'x6 + x1x2'x3'x4x6' + x1x2'x3x5x6' + x1'x2'x3x5'x7 + x1'x2x3'x5'x7' + x1'x2x3'x6x7' + x1x2'x4'x5'x7' + x1x2x5x6x7' + x1x3x4x5x7' + x1x3'x4x6'x7' + x1'x3x5'x6'x7 + x2'x3'x4x5x6' + x2x3x4x5'x7 + x2x3x4x5x7' + x2x3'x5x6'x7 + x2'x3x5'x6'x7' + x2'x4x5x6'x7' + x3x4x5'x6x7 + x1'x2x3'x4x5x7 + x1'x2'x3x4x6x7 + x1x2x3'x4'x6'x7 + x1'x2'x3'x4'x6'x7' + x1x2x3x5'x6x7 + x1x2'x3'x5x6x7 + x1'x2x3x5x6'x7' + x1x2'x4'x5x6x7 + x1'x2x4x5x6'x7 + x1x3x4'x5x6'x7 + x1'x3'x4x5'x6x7' + x1x3x4'x5'x6'x7' + x2'x3'x4'x5x6x7 + x1'x2x3x4'x5x6x7 + x1'x2'x3x4'x5x6x7'

Note

Soit g = Premier(x_1 , ... , x_n) la fonction booléenne qui retourne 1 si le nombre binaire x_1 … x_n est premier et 0 sinon.
Les fonctions f et g sont complémentaires : $g = \overline{f} \Leftrightarrow \text{Premier}(x_1, \ldots, x_n) = \overline{\text{Compose}(x_1, \ldots, x_n)}$

EXEMPLE 6

On appelle fonction booléenne symétrique *S*(n,p,q), la fonction booléenne de n variables égale à la somme de tous les monômes produits de p variables directes x_i $(1 \le i \le n)$ par q variables complémentées $\overline{x_j}$ i.e. x'_j $(1 \le j \le n)$.
Ces monômes sont tous premiers et ce sont les seuls monômes premiers de la fonction *S*(n,p,q). Ainsi, la base première de *S*(n,p,q) est constituée de $C_n^p C_{n-p}^q = \frac{n!}{p!q!(n-p-q)!}$ monômes premiers de p + q variables.

Par exemple, *S*(4,2,1) possède $\frac{4!}{2!\,1!\,1!} = 12$ monômes premiers de 2 + 1 = 3 variables et on a :

S(4,2,1) = x1x2x3' + x1x2'x3 + x1'x2x3 + x1x2x4' + x1x2'x4 + x1'x2x4 + x1x3x4' + x1x3'x4 + x1'x3x4 + x2x3x4' + x2x3'x4 + x2'x3x4

Ecrire une fonction Scilab qui calcule le paramètre *f* correspondant à la définition de *S*(n,p,q) dans le cas général.

Pour n = 7, p = 2, q = 3, exprimer *S*(7,2,3) et simplifier la formule obtenue avec l'algorithme de **McCluskey**.

Solution

On effectue un examen exhaustif de tous les n-uplets X = [x_1 , … , x_i , … , x_n], où $x_i \in \{-1, 0, 1\}$, afin de constituer la liste *f* de ceux qui possèdent p variables à 1 et q variables à 0 (les n – p – q autres variables sont nécessairement à -1).

```
function f = S(n,p,q)
    f = list()
    X = zeros(1,n)
    I = ones(1,n)
    T = ones(n,1)*[-1,0,1]
    loop = %t
    test = %t
    while loop
        if test then
            for i = 1:n
                X(i) = T(i,I(i))
            end
            test1 = length(find(X == 1)) == p
            test2 = length(find(X == 0)) == q
            if test1 & test2 then
                f($+1) = X
            end
            k = 0
        else
            I(k) = 1
        end
        k = k + 1
        I(k) = I(k) + 1
        test = I(k) <= 3
        loop = test | k < n
    end
endfunction
```

Pour exprimer et simplifier *S*(7,2,3), on écrit :

```
n = 7 ; p = 2 ; q = 3 ;
f = S(n,p,q) ;
[fc,fp,fi] = McCluskey(f) ;
printf('\nForme initiale de S(%i,%i,%i) ===> %i monômes premiers\n',n,p,q,length(f)), printf(chaine(f))
printf('\nForme simplifiée de S(%i,%i,%i) ===> %i monômes premiers\n',n,p,q,length(fi)), printf(chaine(fi))
```

Remarque : les listes de monômes *fp* et *f* sont identiques à une permutation près de leurs éléments.

On obtient :

Forme initiale de S(7,2,3) ===> 210 monômes premiers

x1x2x3'x4'x5' + x1x2'x3x4'x5' + x1'x2x3x4'x5' + x1x2'x3'x4x5' + x1'x2x3'x4x5' + x1'x2'x3x4x5' + x1x2'x3'x4'x5 +
x1'x2x3'x4'x5 + x1'x2'x3x4'x5 + x1'x2'x3'x4x5 + x1x2x3'x4'x6' + x1x2'x3x4'x6' + x1'x2x3x4'x6' + x1x2'x3'x4x6' +
x1'x2x3'x4x6' + x1'x2'x3x4x6' + x1x2'x3'x4'x6 + x1'x2x3'x4'x6 + x1'x2'x3x4'x6 + x1'x2'x3'x4x6 + x1x2x3'x4'x7' +
x1x2'x3x4'x7' + x1'x2x3x4'x7' + x1x2'x3'x4x7' + x1'x2x3'x4x7' + x1'x2'x3x4x7' + x1x2'x3'x4'x7 + x1'x2x3'x4'x7 +
x1'x2'x3x4'x7 + x1'x2'x3'x4x7 + x1x2x3'x5'x6' + x1x2'x3x5'x6' + x1'x2x3x5'x6' + x1x2'x3'x5x6' + x1'x2x3'x5x6' +
x1'x2'x3x5x6' + x1x2'x3'x5'x6 + x1'x2x3'x5'x6 + x1'x2'x3x5'x6 + x1'x2'x3'x5x6 + x1x2x3'x5'x7' + x1x2'x3x5'x7' +
x1'x2x3x5'x7' + x1x2'x3'x5x7' + x1'x2x3'x5x7' + x1'x2'x3x5x7' + x1x2'x3'x5'x7 + x1'x2x3'x5'x7 + x1'x2'x3x5'x7 +
x1'x2'x3'x5x7 + x1x2x3'x6'x7' + x1x2'x3x6'x7' + x1'x2x3x6'x7' + x1x2'x3'x6x7' + x1'x2x3'x6x7' + x1'x2'x3x6x7' +
x1x2'x3'x6'x7 + x1'x2x3'x6'x7 + x1'x2'x3x6'x7 + x1'x2'x3'x6x7 + x1x2x4'x5'x6' + x1x2'x4x5'x6' + x1'x2x4x5'x6' +
x1x2'x4'x5x6' + x1'x2x4'x5x6' + x1'x2'x4x5x6' + x1x2'x4'x5'x6 + x1'x2x4'x5'x6 + x1'x2'x4x5'x6 + x1'x2'x4'x5x6 +
x1x2x4'x5'x7' + x1x2'x4x5'x7' + x1'x2x4x5'x7' + x1x2'x4'x5x7' + x1'x2x4'x5x7' + x1'x2'x4x5x7' + x1x2'x4'x5'x7 +
x1'x2x4'x5'x7 + x1'x2'x4x5'x7 + x1'x2'x4'x5x7 + x1x2x4'x6'x7' + x1x2'x4x6'x7' + x1'x2x4x6'x7' + x1x2'x4'x6x7' +
x1'x2x4'x6x7' + x1'x2'x4x6x7' + x1x2'x4'x6'x7 + x1'x2x4'x6'x7 + x1'x2'x4x6'x7 + x1'x2'x4'x6x7 + x1x2x5'x6'x7' +
x1x2'x5x6'x7' + x1'x2x5x6'x7' + x1x2'x5'x6x7' + x1'x2x5'x6x7' + x1'x2'x5x6x7' + x1x2'x5'x6'x7 + x1'x2x5'x6'x7 +
x1'x2'x5x6'x7 + x1'x2'x5'x6x7 + x1x3x4'x5'x6' + x1x3'x4x5'x6' + x1'x3x4x5'x6' + x1x3'x4'x5x6' + x1'x3x4'x5x6' +
x1'x3'x4x5x6' + x1x3'x4'x5'x6 + x1'x3x4'x5'x6 + x1'x3'x4x5'x6 + x1'x3'x4'x5x6 + x1x3x4'x5'x7' + x1x3'x4x5'x7' +
x1'x3x4x5'x7' + x1x3'x4'x5x7' + x1'x3x4'x5x7' + x1'x3'x4x5x7' + x1x3'x4'x5'x7 + x1'x3x4'x5'x7 + x1'x3'x4x5'x7 +
x1'x3'x4'x5x7 + x1x3x4'x6'x7' + x1x3'x4x6'x7' + x1'x3x4x6'x7' + x1x3'x4'x6x7' + x1'x3x4'x6x7' + x1'x3'x4x6x7' +
x1x3'x4'x6'x7 + x1'x3x4'x6'x7 + x1'x3'x4x6'x7 + x1'x3'x4'x6x7 + x1x3x5'x6'x7' + x1x3'x5x6'x7' + x1'x3x5x6'x7' +
x1x3'x5'x6x7' + x1'x3x5'x6x7' + x1'x3'x5x6x7' + x1x3'x5'x6'x7 + x1'x3x5'x6'x7 + x1'x3'x5x6'x7 + x1'x3'x5'x6x7 +
x1x4x5'x6'x7' + x1x4'x5x6'x7' + x1'x4x5x6'x7' + x1x4'x5'x6x7' + x1'x4x5'x6x7' + x1'x4'x5x6x7' + x1x4'x5'x6'x7 +
x1'x4x5'x6'x7 + x1'x4'x5x6'x7 + x1'x4'x5'x6x7 + x2x3x4'x5'x6' + x2x3'x4x5'x6' + x2'x3x4x5'x6' + x2x3'x4'x5x6' +
x2'x3x4'x5x6' + x2'x3'x4x5x6' + x2x3'x4'x5'x6 + x2'x3x4'x5'x6 + x2'x3'x4x5'x6 + x2'x3'x4'x5x6 + x2x3x4'x5'x7' +
x2x3'x4x5'x7' + x2'x3x4x5'x7' + x2x3'x4'x5x7' + x2'x3x4'x5x7' + x2'x3'x4x5x7' + x2x3'x4'x5'x7 + x2'x3x4'x5'x7 +
x2'x3'x4x5'x7 + x2'x3'x4'x5x7 + x2x3x4'x6'x7' + x2x3'x4x6'x7' + x2'x3x4x6'x7' + x2x3'x4'x6x7' + x2'x3x4'x6x7' +
x2'x3'x4x6x7' + x2x3'x4'x6'x7 + x2'x3x4'x6'x7 + x2'x3'x4x6'x7 + x2'x3'x4'x6x7 + x2x3x5'x6'x7' + x2x3'x5x6'x7' +
x2'x3x5x6'x7' + x2x3'x5'x6x7' + x2'x3x5'x6x7' + x2'x3'x5x6x7' + x2x3'x5'x6'x7 + x2'x3x5'x6'x7 + x2'x3'x5x6'x7 +
x2'x3'x5'x6x7 + x2x4x5'x6'x7' + x2x4'x5x6'x7' + x2'x4x5x6'x7' + x2x4'x5'x6x7' + x2'x4x5'x6x7' + x2'x4'x5x6x7' +
x2x4'x5'x6'x7 + x2'x4x5'x6'x7 + x2'x4'x5x6'x7 + x2'x4'x5'x6x7 + x3x4x5'x6'x7' + x3x4'x5x6'x7' + x3'x4x5x6'x7' +
x3x4'x5'x6x7' + x3'x4x5'x6x7' + x3'x4'x5x6x7' + x3x4'x5'x6'x7 + x3'x4x5'x6'x7 + x3'x4'x5x6'x7 + x3'x4'x5'x6x7

Forme simplifiée de S(7,2,3) ===> 35 monômes premiers

x1'x2'x3x4x5' + x1'x2x3'x4x5' + x1'x2x3x4'x5' + x1x2'x3'x4x5' + x1x2'x3x4'x5' + x1x2x3'x4'x5' + x1'x2'x3x4'x7 +
x1'x2x3'x4'x7 + x1x2'x3'x4'x7 + x1'x2x3'x5x6' + x1x2'x3'x5x6' + x1'x2'x3'x6x7 + x1'x2x3'x6x7' + x1x2'x3'x6x7' +
x1'x2'x4x5x6' + x1'x2'x5x6x7' + x1'x3x4'x5x6' + x1'x3x4'x6x7' + x1'x3x4x6'x7' + x1'x4x5'x6'x7 + x1'x4x5'x6x7' +
x2'x3x4x5'x6' + x2x3'x4'x5x6' + x2x3'x4x5'x6' + x2x3x4'x5'x6' + x2'x3x4'x5x7' + x2'x4'x5x6'x7 + x2'x4x5'x6x7' +
x2'x4x5x6'x7' + x3'x4'x5x6x7' + x3'x4x5'x6x7' + x3'x4x5x6'x7' + x3x4'x5'x6'x7 + x3x4x5'x6'x7'

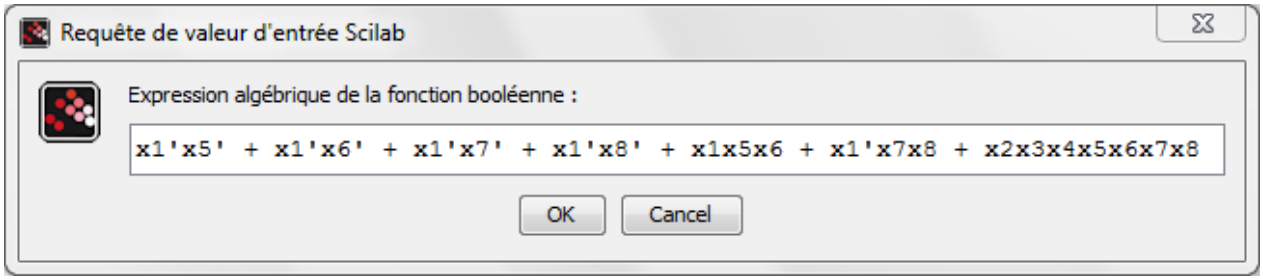
EXEMPLE 7

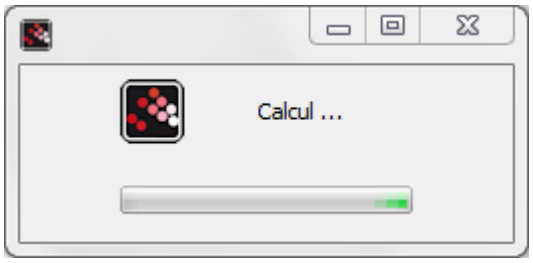
Ecrire une procédure interactive qui ouvre une première fenêtre de dialogue pour saisir l'expression algébrique d'une fonction booléenne et une seconde fenêtre de dialogue pour afficher une expression algébrique simplifiée de cette fonction. Pour limiter le temps de calcul, on supposera que la fonction booléenne possède au plus 10 variables.

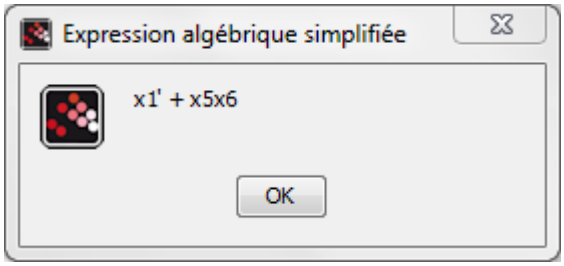
Solution

```
in = x_dialog('Expression algébrique de la fonction booléenne :','') ;
if in == [] then, abort, else, in = strcat(in,' ') ; end
check = regexp(in,'/^\s*(0|1|(x([1-9]|10)''?)+)(\s*\+\s*(0|1|(x([1-9]|10)''?)+))*\s*$/') ;
if check == [] then, messagebox('Donnée incorrecte','Erreur','error') ; abort, end
f = liste(in) ;
winId = progressionbar('Calcul ...') ;
[fc,fp,fi] = McCluskey(f) ;
close(winId) ;
out = chaine(fi) ;
messagebox(out,'Expression algébrique simplifiée') ;
```

Prenons la fonction $f = \bar{x}_1\,\bar{x}_5 + \bar{x}_1\,\bar{x}_6 + \bar{x}_1\,\bar{x}_7 + \bar{x}_1\,\bar{x}_8 + x_1\,x_5\,x_6 + \bar{x}_1\,x_7\,x_8 + x_2\,x_3\,x_4\,x_5\,x_6\,x_7\,x_8$





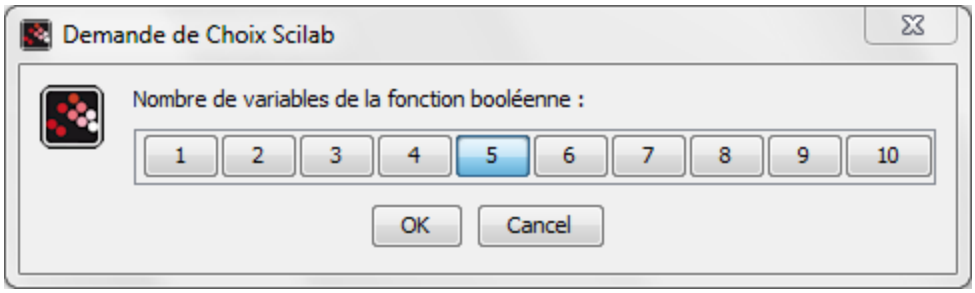


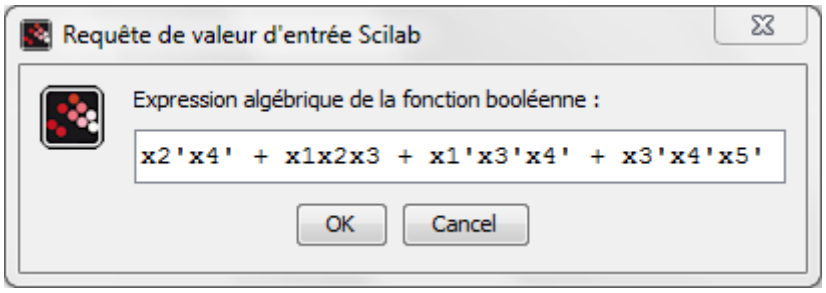
Mathématique & Informatique

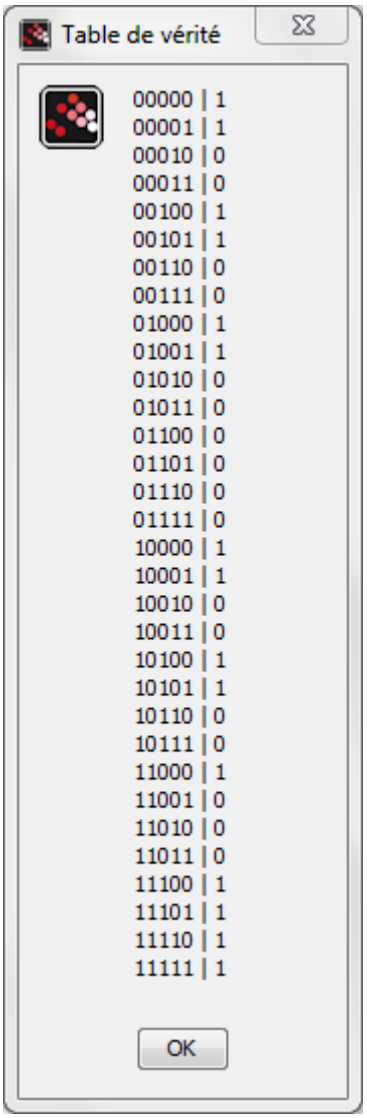
La procédure suivante affiche la table de vérité d'une fonction booléenne après la saisie de son expression algébrique :

```
n = x_choices('Nombre de variables de la fonction booléenne :',list(list('',1,string([1:10])))) ;
if n == [] then, abort, end
s = x_dialog('Expression algébrique de la fonction booléenne :','') ;
if s == [] then, abort, else, s = strcat(s,' ') ; end
I = strcat(string([1:n]),'|') ;
check = regexp(s,'/^\s*(0|1|(x(' + I + ')''?)+)(\s*\+\s*(0|1|(x(' + I + ')''?)+))*\s*$/') ;
if check == [] then, messagebox('Donnée incorrecte','Erreur','error') ; abort, end
f = liste(s) ;
T = verite(f,n) ;
messagebox(T,'Table de vérité') ;
```

Par exemple, pour la fonction $f = \bar{x}_2\,\bar{x}_4 + x_1\,x_2\,x_3 + \bar{x}_1\,\bar{x}_3\,\bar{x}_4 + \bar{x}_3\,\bar{x}_4\,\bar{x}_5$ de 5 variables :







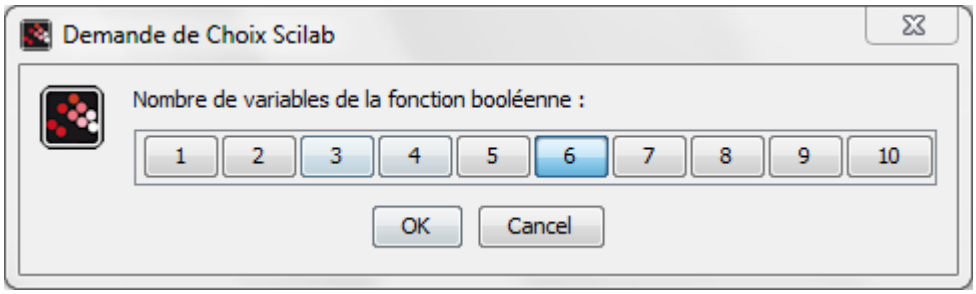
Mathématique & Informatique

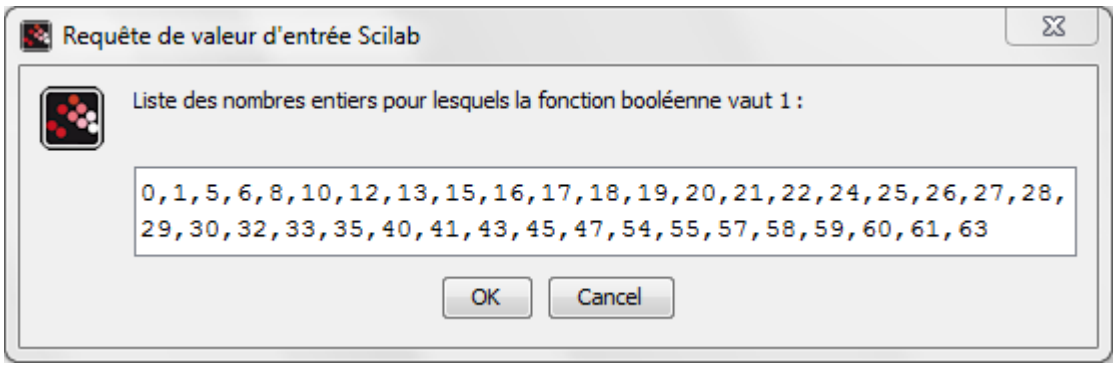
Procédure analogue à la première, mais la fonction booléenne est définie par l'ensemble des points où elle vaut 1 :

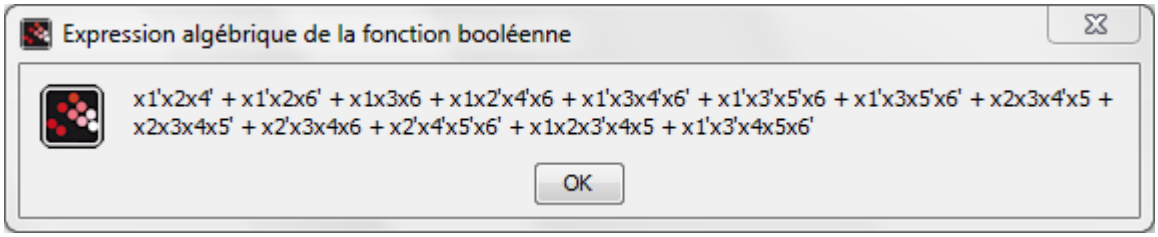
```
n = x_choices('Nombre de variables de la fonction booléenne :',list(list('',1,string([1:10])))) ;
if n == [] then, abort, end
in = x_dialog('Liste des nombres entiers pour lesquels la fonction booléenne vaut 1 :','') ;
if in == [] then, abort, else, in = strcat(in,' ') ; end
check = regexp(in,'/^\s*\d+(\s*(,|\s)\s*\d+)*\s*$/') ;
if check == [] & length(in) <> 0 then, messagebox('Donnée incorrecte','Erreur','error') ; abort, end
L = unique(evstr(in)) ;
if find(L > 2^n-1) <> [] then, messagebox('Donnée incorrecte','Erreur','error') ; abort, end
f = list() ;
for k = 1:length(L)
    f($+1) = evstr(strsplit(dec2bin(L(k),n))') ;
end
winId = progressionbar('Calcul ...') ;
[fc,fp,fi] = McCluskey(f) ;
close(winId) ;
out = chaine(fi) ;
messagebox(out,'Expression algébrique de la fonction booléenne') ;
```

Par exemple, pour la fonction f de 6 variables donnée par :

f = Σ(0,1,5,6,8,10,12,13,15,16,17,18,19,20,21,22,24,25,26,27,28,29,30,32,33,35,40,41,43,45,47,54,55,57,58,59,60,61,63)

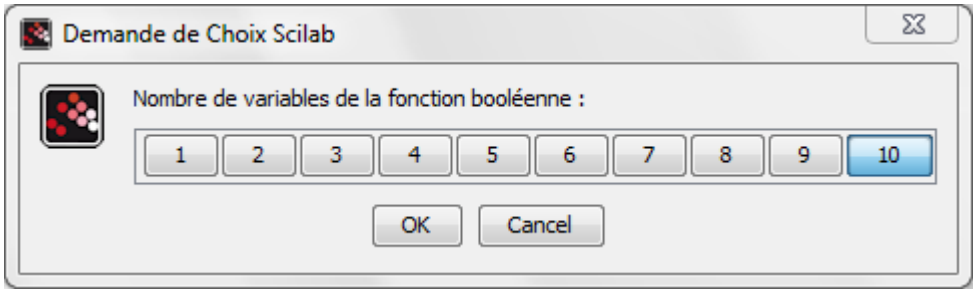


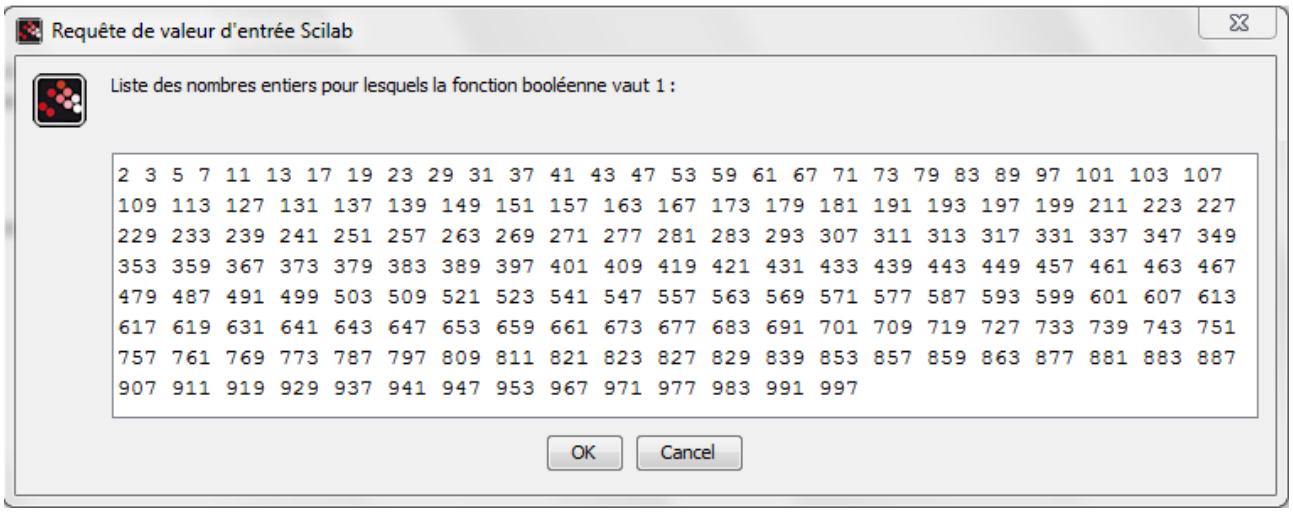


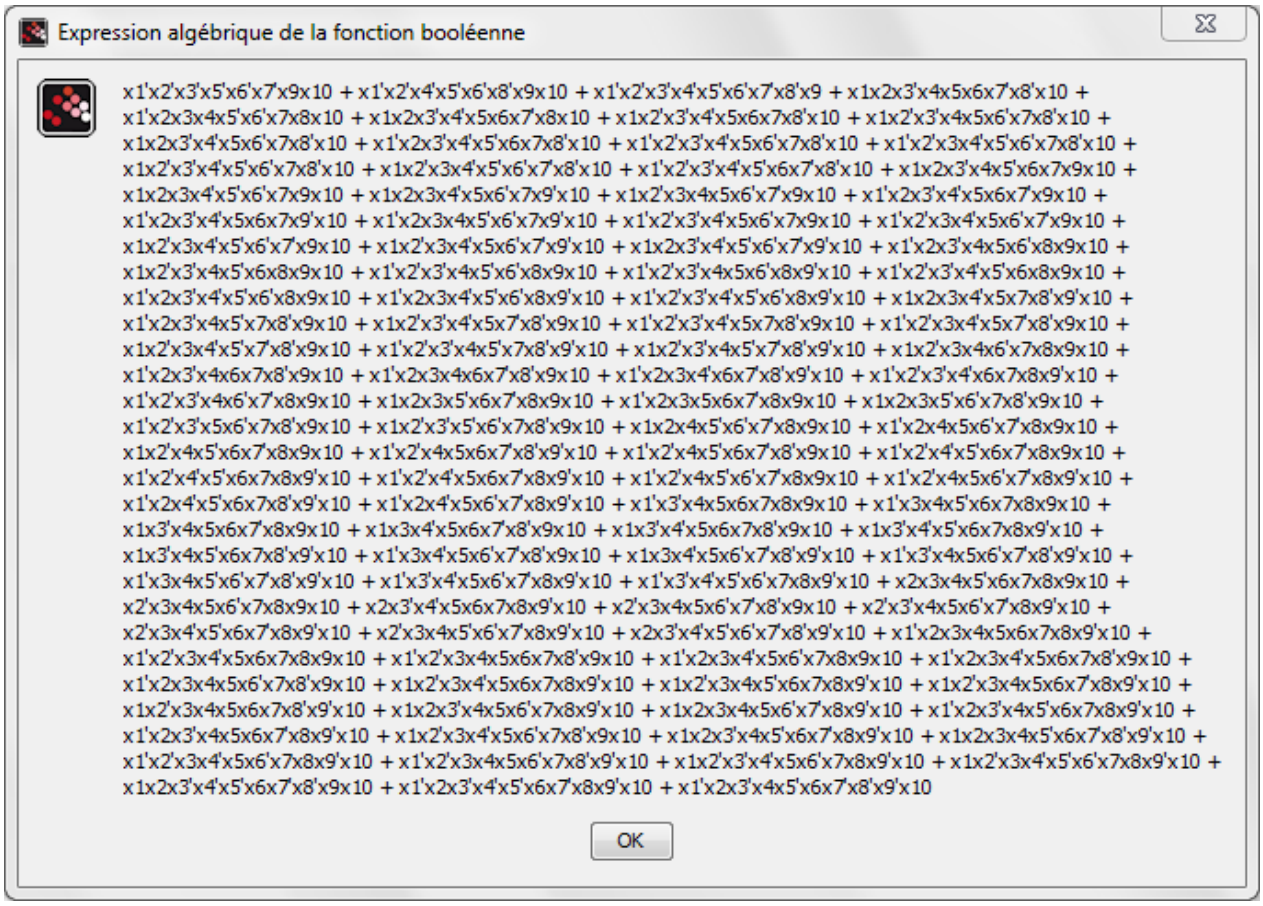


Mathématique & Informatique

Quelle est l'expression algébrique de la fonction booléenne de 10 variables, égale à 1 pour tous les nombres premiers inférieurs à 1000 et égale à 0 pour les autres nombres entiers ?

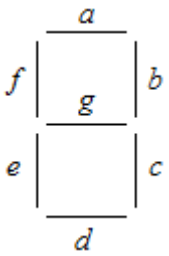




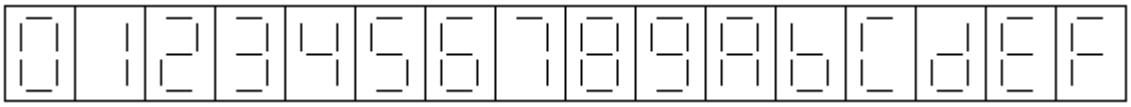


Algorithme de Quine - McCluskey pour les fonctions ϕ-booléennes

Introduction sur un exemple



Un afficheur 7 segments *a, b, c, d, e, f, g* permet de représenter les 16 chiffres hexadécimaux 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F de la manière suivante :



En codant les chiffres hexadécimaux sur 4 bits x1x2x3x4, recherchons les expressions booléennes simplifiées f(x1,x2,x3,x4) des segments *a, b, c, d, e, f, g* (on suppose qu'un segment apparaît si et seulement si la fonction booléenne associée vaut 1et on note x1', x2', x3', x4' les compléments de x1, x2, x3, x4 dans les expressions) :

```
a = [0,2,3,5,6,7,8,9,10,12,14,15] ;
b = [0,1,2,3,4,7,8,9,10,13] ;
c = [0,1,3,4,5,6,7,8,9,10,11,13] ;
d = [0,2,3,5,6,8,9,11,12,13,14] ;
e = [0,2,6,8,10,11,12,13,14,15] ;
f = [0,4,5,6,8,9,10,11,12,14,15] ;
g = [2,3,4,5,6,8,9,10,11,13,14,15] ;
Afficheur = list(a,b,c,d,e,f,g) ;
for i = 1:length(Afficheur)
    F = Afficheur(i) ;
    f = list() ;
    for j = 1:length(F)
        f($+1) = evstr(strsplit(dec2bin(F(j),4)))' ;
    end
    [fc,fp,fi] = McCluskey(f) ;
    printf('\n '+ ascii(ascii('a') + i - 1) + ' = ')
    printf(chaine(fi))
end
```

On obtient :

```
a = x1'x3 + x1x4' + x2x3 + x2'x4' + x1x2'x3' + x1'x2x4
b = x1'x2' + x2'x4' + x1'x3x4 + x1x3'x4 + x1'x3'x4'
c = x1'x2 + x1x2' + x1'x3' + x1'x4 + x3'x4
d = x1x3' + x1'x2'x4' + x2x3'x4 + x2'x3x4 + x2x3x4'
e = x1x2 + x1x3 + x2'x4' + x3x4'
f = x1x2' + x1x3 + x2x4' + x3'x4' + x1'x2x3'
g = x1x2' + x1x4 + x2'x3 + x3x4' + x1'x2x3'
```

Si l'on souhaite afficher uniquement les chiffres décimaux 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 alors les valeurs de *a, b, c, d, e, f, g* peuvent être quelconques (0 ou 1 au choix) pour 10, 11, 12, 13, 14, 15, c'est-à-dire pour les entrées x1x2x3x4 > 1001. Dans ce cas, les sorties *a, b, c, d, e, f, g* sont des fonctions ϕ-booléennes.

Mathématique & Informatique

Programmation de l'algorithme

Soit f une fonction ϕ-booléenne de n variables $x_1, \dots, x_n$

x_1 ... x_n	f
0 ... 0	0/1/ϕ
0 ... 1	0/1/ϕ
: :	:
: :	:
1 ... 1	0/1/ϕ

Introduisons une nouvelle fonction *McCluskey(f,phi)* où :

- *f* est identique au paramètre de la fonction *McCluskey(f)* décrite précédemment
- *phi* est la liste des n-uplets binaires pour lesquels f vaut ϕ (valeur indéterminée 0 ou 1)

La fonction *McCluskey(f,phi)* retourne l'expression booléenne de f qui contient le moins de variables x_i et $\bar{x}_i$

Remarque 1 : La notation ϕ (lettre grecque phi) représente un 0 et un 1 superposés.
Ne pas confondre avec ∅ (o barré symbole de l'ensemble vide).
Remarque 2 : Il y a 3^{2^n} fonctions ϕ-booléennes de n variables.
Remarque 3 : *McCluskey(f)* est équivalent à *McCluskey(f,list())*, où *list()* désigne la liste vide.

Lemme

Soit un ensemble E à n éléments : $E = \{e_1, \dots, e_n\}$
Le nombre des parties de E est égal à la somme des combinaisons de p éléments parmi n ($0 \le p \le n$), soit $\sum_{p=0}^{p=n} C_n^p = 2^n$
C'est aussi le nombre de valeurs que peut prendre le n-uplet binaire $[x_1, \dots, x_n]$ de $[0, \dots, 0]$ à $[1, \dots, 1]$.
En faisant correspondre à un nombre entier N, le n-uplet binaire $[x_1, \dots, x_n]$ et le sous-ensemble $A = \{e_1 \times x_1, \dots, e_n \times x_n\}$ où $e_i \times x_i = \emptyset$ si $x_i = 0$ et $e_i \times x_i = e_i$ si $x_i = 1$, on obtient toutes les parties A de E lorsque N varie de 0 à $2^n - 1$.
Par exemple, la boucle suivante affiche toutes les parties de $E = \{e_1, e_2, e_3, e_4\}$:

```
E = list('e1','e2','e3','e4') ;
n = length(E) ;
for N = 0:2^n-1
    X = evstr(strsplit(dec2bin(N,n)))' ;
    A = list() ;
    for i = find(X == 1)
        A($+1) = E(i) ;
    end
    printf('\n{')
    for i = 1:length(A)
        if i <> 1 then, printf(','), end
        printf(A(i))
    end
    printf('}')
end
```

```
{}
{e4}
{e3}
{e3,e4}
{e2}
{e2,e4}
{e2,e3}
{e2,e3,e4}
{e1}
{e1,e4}
{e1,e3}
{e1,e3,e4}
{e1,e2}
{e1,e2,e4}
{e1,e2,e3}
{e1,e2,e3,e4}
```

Ce procédé sera utilisé pour obtenir toutes les formes canoniques possibles de f lorsque ϕ est remplacé par 0 ou par 1.

Mathématique & Informatique

```
function [fc,fp,fi] = McCluskey(f,phi)

    function fc = canonique(f)
        fc = list()
        if f == list() then, return, end
        N = length(f)
        n = length(f(1))
        X = zeros(1,n)
        I = ones(1,n)
        T = [zeros(n,1),ones(n,1)]
        loop = %t
        test = %t
        while loop
            if test then
                for i = 1:n
                    X(i) = T(i,I(i))
                end
                for i = 1:N
                    m = f(i)
                    val = 1
                    for j = 1:n
                        if m(j) + X(j) == 1 then
                            val = 0
                            break
                        end
                    end
                    if val == 1 then
                        fc($+1) = X
                        break
                    end
                end
                p = n + 1
            else
                I(p) = 1
            end
            p = p - 1
            I(p) = I(p) + 1
            test = I(p) <= 2
            loop = test | p > 1
        end
    endfunction
```

```
// Base première
function fp = premiere(f)
    fp = list()
    if f == list() then, return, end
    fr = fc
    while length(fr) <> 0
        ft = fr
        It = ones(1,length(fr))
        fr = list()
        for i = 1:length(ft)-1
            for j = i+1:length(ft)
                m1 = ft(i)
                m2 = ft(j)
                m = []
                p = 0
                for q = 1:length(m1)
                    if m1(q) <> m2(q) then
                        if p <> 0 then
                            p = 0
                            break
                        end
                        p = q
                    end
                end
                if p <> 0 then
                    if m1(p) + m2(p) == 1 then
                        m = [m1(1:p-1),-1,m1(p+1:$)]
                    end
                end
                if m <> [] then
                    It(i) = 0
                    It(j) = 0
                    existe = %f
                    for k = 1:length(fr)
                        existe = isequal(fr(k),m)
                        if existe then, break, end
                    end
                    if ~existe then
                        fr($+1) = m
                    end
                end
            end
        end
        for k = find(It == 1)
            fp($+1) = ft(k)
        end
    end
    if length(fp) == 1 then
        if find(fp(1) <> -1) == [] then
            fp = list(-1)
        end
    end
endfunction
```

```
// Base irrédondante
function fi = irredondante(f)
    fi = list()
    if f == list() then, return, end
    p = length(fp)
    q = length(fc)
    M = ones(p+1,q+1)
    for i = 1:p
        m1 = fp(i)
        for j = 1:q
            m2 = fc(j)
            for k = 1:length(m1)
                if m1(k) + m2(k) == 1 then
                    M(i,j) = 0
                    break
                end
            end
        end
    end
    ajout = %t
    while ajout
        ajout = %f
        Min = %inf
        Imin = []
        for j = 1:q
            if M(p+1,j) == 0 then, continue, end
            I = find(M(1:p,j) == 1)
            if I <> [] & length(I) < Min then
                Min = length(I)
                Imin = I
            end
        end
        Max = 0
        imax = 0
        if Imin <> [] then
            for k = 1:length(Imin)
                if M(Imin(k),q+1) == 0 then, continue, end
                J = find(M(Imin(k),1:q) == 1)
                if length(J) > Max then
                    Max = length(J)
                    imax = Imin(k)
                end
            end
        end
        if imax <> 0 then
            M(imax,q+1) = 0
            fi($+1) = fp(imax)
            ajout = %t
            for j = 1:q
                if M(imax,j) == 1 then
                    M(p+1,j) = 0
                    for i = 1:p
                        M(i,j) = 0
                    end
                end
            end
        end
    end
endfunction
```

```
    // Valeurs phi-booléennes
    fparam = f
    lfimin = %inf
    n = length(phi)
    for N = 0:2^n-1
        X = evstr(strsplit(dec2bin(N,n)))'
        f = fparam
        for i = find(X == 1)
            f($+1) = phi(i)
        end
        fc = canonique(f)
        fp = premiere(f)
        fi = irredondante(f)
        lfi = 0
        for i = 1:length(fi)
            lfi = lfi + length(find(fi(i) <> -1))
        end
        if lfi < lfimin then
            fcmin = fc
            fpmin = fp
            fimin = fi
            lfimin = lfi
        end
    end
    fc = fcmin
    fp = fpmin
    fi = fimin

endfunction
```

Mathématique & Informatique

EXEMPLE 1

Soit la fonction ϕ-booléenne f : $\{0,1\}^5 \rightarrow \{0,1,\phi\}$, définie par la table de vérité suivante :

$x_1x_2x_3x_4x_5$	f
00000	0
00001	1
00010	ϕ
00011	1
00100	0
00101	1
00110	ϕ
00111	ϕ
01000	1
01001	ϕ
01010	1
01011	1
01100	0
01101	0
01110	1
01111	0
10000	1
10001	1
10010	1
10011	ϕ
10100	0
10101	ϕ
10110	ϕ
10111	0
11000	ϕ
11001	ϕ
11010	0
11011	1
11100	0
11101	0
11110	ϕ
11111	1

Trouver l'expression logique minimale de f.

Solution

Programme :

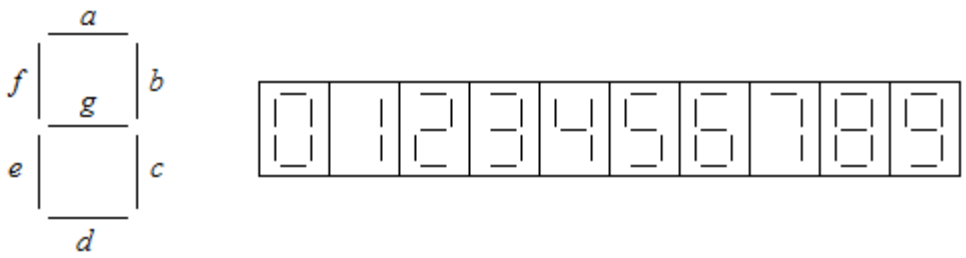
```
f = list([0,0,0,0,1],[0,0,0,1,1],[0,0,1,0,1],[0,1,0,0,0],[0,1,0,1,0],...
         [0,1,0,1,1],[0,1,1,1,0],[1,0,0,0,0],[1,0,0,0,1],[1,0,0,1,0],...
         [1,1,0,1,1],[1,1,1,1,1]) ;
phi = list([0,0,0,1,0],[0,0,1,1,0],[0,0,1,1,1],[0,1,0,0,1],[1,0,0,1,1],...
           [1,0,1,0,1],[1,0,1,1,0],[1,1,0,0,0],[1,1,1,0,1],[1,1,1,1,0]) ;
[fc,fp,fi] = McCluskey(f,phi) ;
printf('\n f = '), printf(chaine(fi))
```

Résultat :

f = x1'x2x3' + x1x2'x3' + x1'x2'x5 + x3x4x5' + x1x2x4x5 = $\bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_5 + x_3x_4\bar{x}_5 + x_1x_2x_4x_5$

EXEMPLE 2

Appliquer l'algorithme de *McCluskey* à l'afficheur 7 segments utilisé pour représenter les chiffres décimaux de 0 à 9.



Solution

On écrit :

```
a = [0,2,3,5,6,7,8,9] ;
b = [0,1,2,3,4,7,8,9] ;
c = [0,1,3,4,5,6,7,8,9] ;
d = [0,2,3,5,6,8,9] ;
e = [0,2,6,8] ;
f = [0,4,5,6,8,9] ;
g = [2,3,4,5,6,8,9] ;
PHI = [10,11,12,13,14,15] ;
Afficheur = list(a,b,c,d,e,f,g) ;
for i = 1:length(Afficheur)
    F = Afficheur(i) ;
    f = list() ;
    for j = 1:length(F)
        f($+1) = evstr(strsplit(dec2bin(F(j),4)))' ;
    end
    phi = list() ;
    for j = 1:length(PHI)
        phi($+1) = evstr(strsplit(dec2bin(PHI(j),4)))' ;
    end
    [fc,fp,fi] = McCluskey(f,phi) ;
    printf('\n '+ ascii(ascii('a') + i - 1) + ' = ')
    printf(chaine(fi))
end
```

On obtient :

```
a = x1 + x3 + x2x4 + x2'x4'
b = x2' + x3x4 + x3'x4'
c = x2 + x3' + x4
d = x1 + x2'x3 + x2'x4' + x3x4' + x2x3'x4
e = x2'x4' + x3x4'
f = x1 + x2x3' + x2x4' + x3'x4'
g = x1 + x2'x3 + x2x3' + x3x4'
```

Note

Une solution pour réaliser les fonctions *a, b, c, d, e, f, g* avec des portes logiques, consiste à utiliser des AND précédés au besoin d'inverseurs pour obtenir les produits de variables et des OR pour effectuer les sommes de termes.

Mathématique & Informatique

EXEMPLE 3

Soit une fonction ϕ-booléenne f. Ecrire une procédure interactive qui effectue les opérations suivantes :

1) Saisie de la liste des points décimaux où f vaut 1 et saisie de la liste des points décimaux où f vaut 0

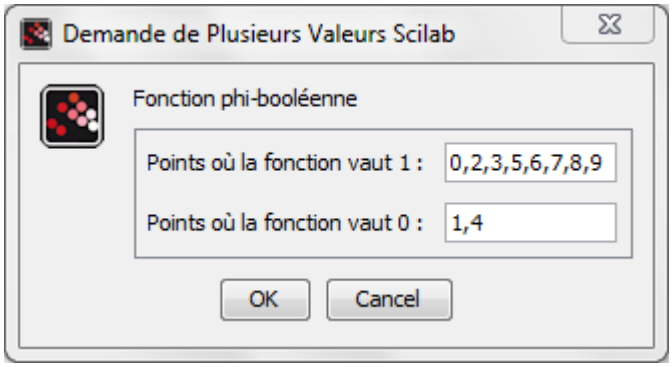
2) Calcul de l'expression algébrique de f et affichage du résultat dans une fenêtre

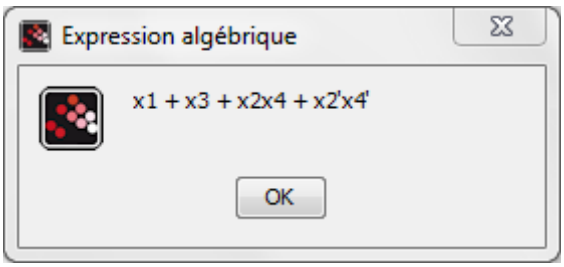
On considèrera que f vaut ϕ pour les points n'appartenant pas aux deux listes qui auront été définies.

Solution

```
in = x_mdialog('Fonction phi-booléenne',['Points où la fonction vaut 1 :';'Points où la fonction vaut 0 :'],['';'']) ;
if length(in(1)) == 0 & length(in(2)) == 0 then, abort, end
check1 = regexp(in(1),'/^\s*\d+(\s*(,|\s)\s*\d+)*\s*$/') ;
if check1 == [] & length(in(1)) <> 0 then, messagebox('Donnée incorrecte','Erreur','error') ; abort, end
check2 = regexp(in(2),'/^\s*\d+(\s*(,|\s)\s*\d+)*\s*$/') ;
if check2 == [] & length(in(2)) <> 0 then, messagebox('Donnée incorrecte','Erreur','error') ; abort, end
L1 = unique(evstr(in(1))) ;
L2 = unique(evstr(in(2))) ;
if intersect(L1,L2) <> [] then, messagebox('Donnée incorrecte','Erreur','error') ; abort, end
if isequal(L1,0) then, n1 = 1 ; else, n1 = ceil(log(max(L1) + 1)/log(2)) ; end
if isequal(L2,0) then, n2 = 1 ; else, n2 = ceil(log(max(L2) + 1)/log(2)) ; end
n = max(n1,n2) ;
f = list() ;
for k = 1:length(L1)
    f($+1) = evstr(strsplit(dec2bin(L1(k),n)))' ;
end
phi = list() ;
for N = 0:2^n-1
    if (find(L1 == N) == []) & (find(L2 == N) == []) then
        phi($+1) = evstr(strsplit(dec2bin(N,n)))' ;
    end
end
winId = progressionbar('Calcul ...') ;
[fc,fp,fi] = McCluskey(f,phi) ;
close(winId) ;
out = chaine(fi) ;
messagebox(out,'Expression algébrique') ;
```

Prenons le cas du segment *a* de l'afficheur précédent :

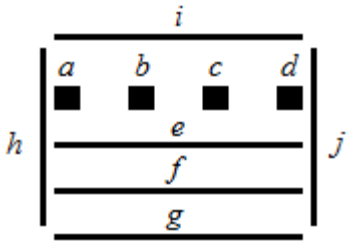




Mathématique & Informatique

EXEMPLE 4

On souhaite réaliser un afficheur vicésimal (base 20) permettant de représenter les chiffres mayas.



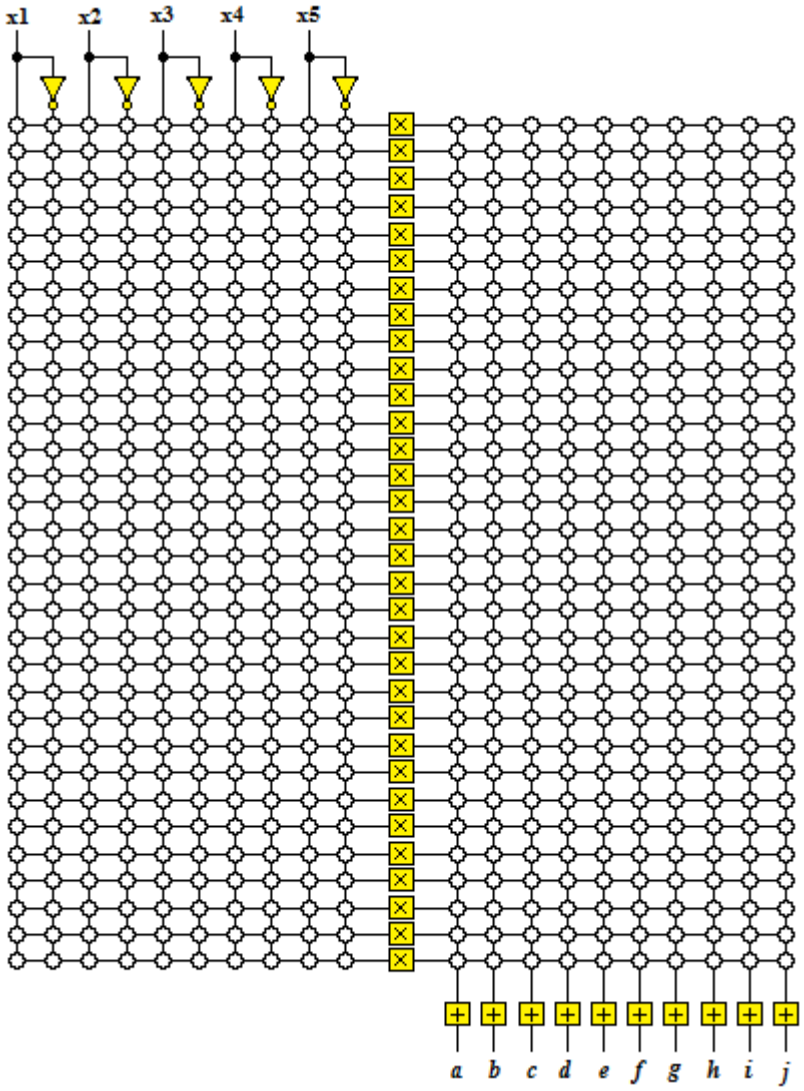
Glyphe	Valeur
	0
	1
	2
	3
	4
	5
	6
	7
	8
	9
	10
	11
	12
	13
	14
	15
	16
	17
	18
	19

Les nombres 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 sont codés sur 5 bits x1x2x3x4x5.
Quelle est l'expression booléenne minimale de chaque segment *a, b, c, d, e, f, g, h, i, j* en fonction de x1, x2, x3, x4, x5 ?

Mathématique & Informatique

Un PLA (Programmable Logic Array) est un circuit intégré spécialement étudié pour réaliser des fonctions logiques : il contient un certain nombre de portes et des liaisons internes librement modifiables par programmation.
Pour établir les connexions entre les portes, une première méthode consiste à introduire des fusibles/anti-fusibles qui seront ouverts/fermés par une surtension (circuit intégré programmable une fois), une deuxième méthode consiste à utiliser des transistors de commutation qui seront commandés par une mémoire (circuit intégré reprogrammable).

Le schéma simplifié du PLA se présente sous la forme d'une matrice de points de connexion :



Les ⊠ sont des portes AND à entrées multiples.
Les ⊞ sont des portes OR à entrées multiples.
Les ○ sont des points de connexion programmables.

Quelles sont les liaisons à effectuer pour obtenir les sorties $a, b, c, d, e, f, g, h, i, j$?

Mathématique & Informatique

Solution

On écrit :

```
a = [4,9,14,19] ;
b = [3,4,8,9,13,14,18,19] ;
c = [2,3,4,7,8,9,12,13,14,17,18,19] ;
d = [1,2,3,4,6,7,8,9,11,12,13,14,16,17,18,19] ;
e = [5,6,7,8,9,10,11,12,13,14,15,16,17,18,19] ;
f = [10,11,12,13,14,15,16,17,18,19] ;
g = [0,15,16,17,18,19] ;
h = [0] ;
i = [0] ;
j = [0] ;
PHI = [20,21,22,23,24,25,26,27,28,29,30,31] ;
Afficheur = list(a,b,c,d,e,f,g,h,i,j) ;
for i = 1:length(Afficheur)
    F = Afficheur(i) ;
    f = list() ;
    for j = 1:length(F)
        f($+1) = evstr(strsplit(dec2bin(F(j),5)))' ;
    end
    phi = list() ;
    for j = 1:length(PHI)
        phi($+1) = evstr(strsplit(dec2bin(PHI(j),5)))' ;
    end
    [fc,fp,fi] = McCluskey(f,phi) ;
    printf('\n '+ ascii(ascii('a') + i - 1) + ' = ')
    printf(chaine(fi))
end
```

On obtient :

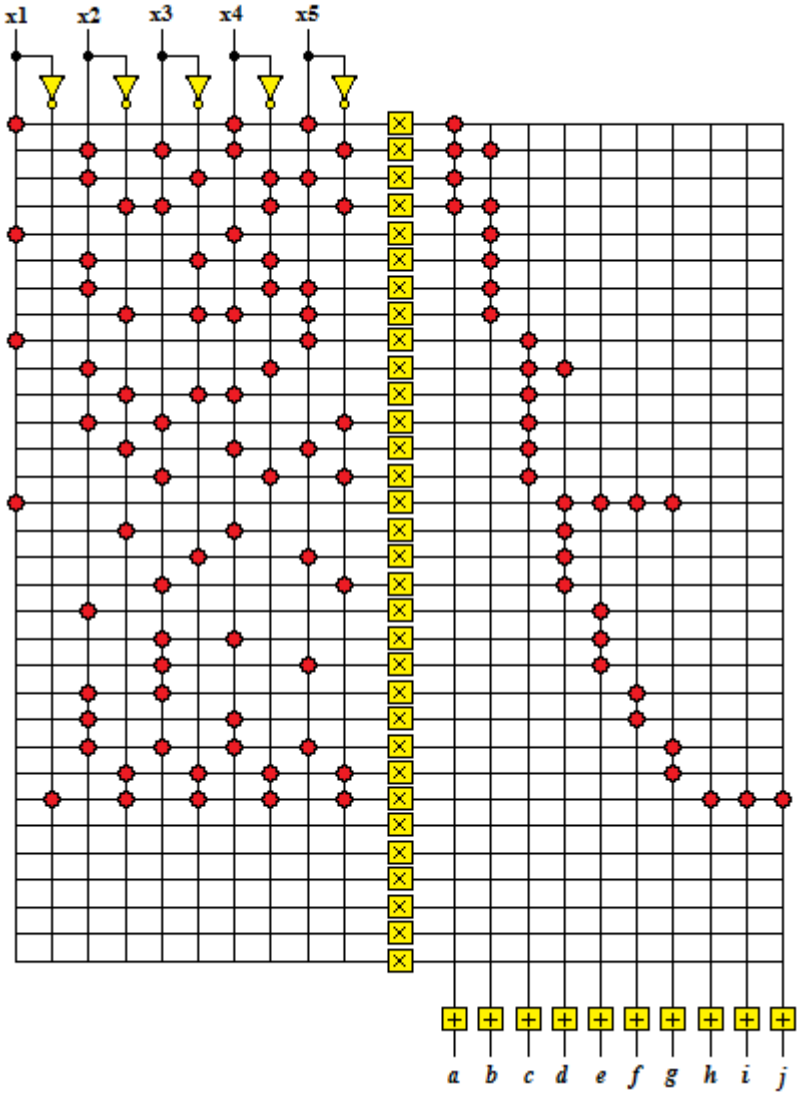
```
a = x1x4x5 + x2x3x4x5' + x2x3'x4'x5 + x2'x3x4'x5'
b = x1x4 + x2x3'x4' + x2x4'x5 + x2x3x4x5' + x2'x3'x4x5 + x2'x3x4'x5'
c = x1x5 + x2x4' + x2'x3'x4 + x2x3x5' + x2'x4x5 + x3x4'x5'
d = x1 + x2'x4 + x2x4' + x3'x5 + x3x5'
e = x1 + x2 + x3x4 + x3x5
f = x1 + x2x3 + x2x4
g = x1 + x2x3x4x5 + x2'x3'x4'x5'
h = x1'x2'x3'x4'x5'
i = x1'x2'x3'x4'x5'
j = x1'x2'x3'x4'x5'
```

Remarque :

Les monômes x1, x2x4', x2x3x4x5', x2'x3x4'x5', x1'x2'x3'x4'x5' sont communs à plusieurs sorties.

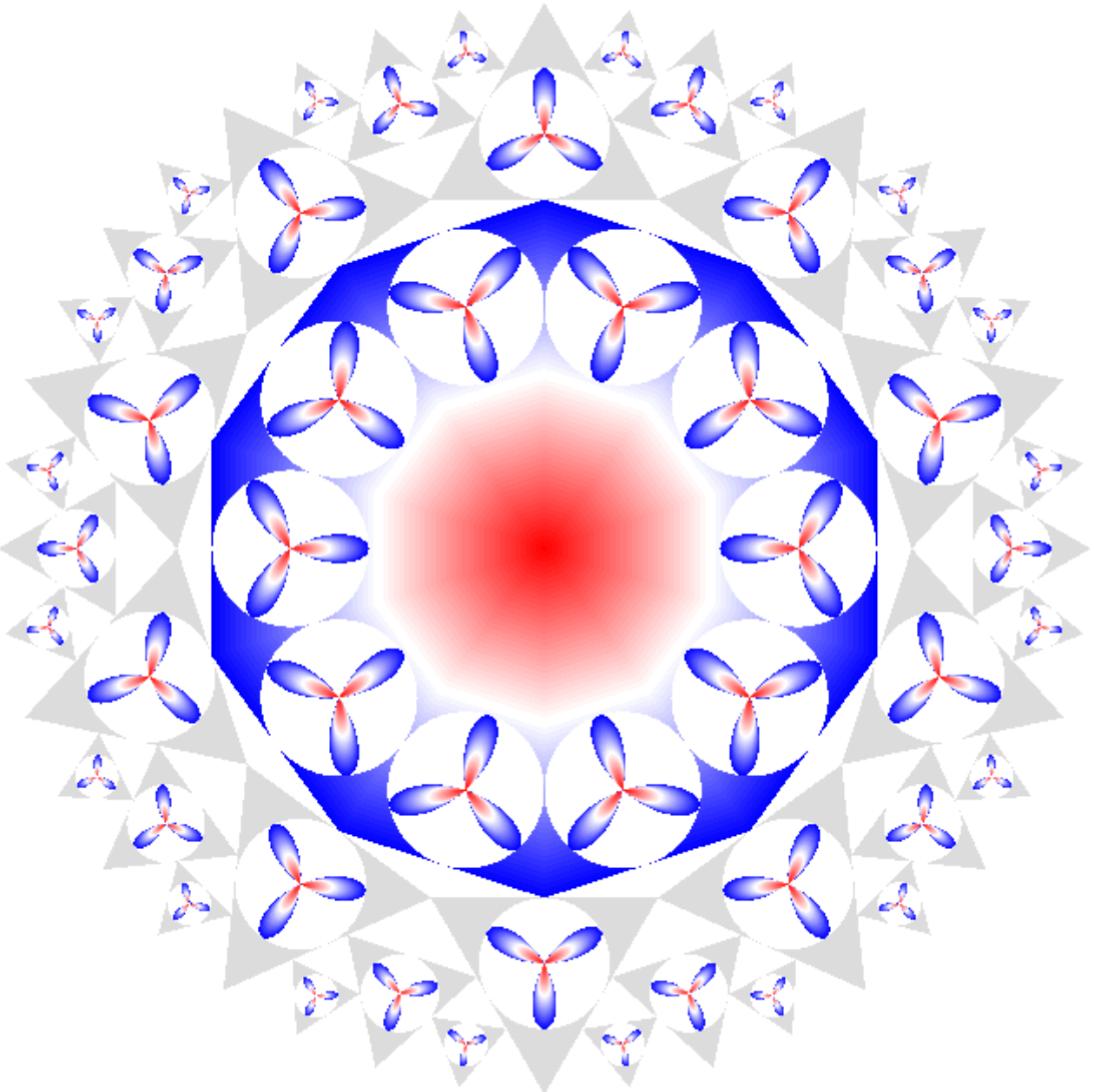
Mathématique & Informatique

Réalisation avec un PLA



Pour établir les connexions du PLA, on a choisi les expressions algébriques simplifiées des sorties. Il est possible de prendre tout simplement les formes canoniques de *a, b, c, d, e, f, g, h, i, j* pour obtenir un PLA à 20 lignes utilisées et 174 points de connexion à activer contre 26 lignes utilisées mais seulement 106 points de connexion à activer pour les formes simplifiées. On peut aussi rechercher le PLA dont la matrice des points de connexion possède le minimum de lignes (les colonnes de la matrice sont prédéterminées, leur nombre est constant). Le problème consiste alors à minimiser globalement les expressions algébriques d'un ensemble $F = \{f_1 , \dots , f_m\}$ de m fonctions ϕ-booléennes de n variables $x_1 , \dots , x_n$. Pour cela, on recherchera la base B de monômes qui contient le minimum d'éléments pour couvrir toutes les fonctions de F.

Mathématique & Informatique



Mathématique & Informatique

STŒCHIOMÉTRIE

Introduction

Rappelons que la loi de conservation des masses et des éléments (**Lavoisier** [1]) s'applique aux réactions chimiques courantes mais pas aux réactions nucléaires où interviennent des transmutations des éléments mettant en jeu des énergies considérables et donc des variations de masses significatives (**Einstein** [3]).

Le problème

On suppose qu'une équation chimique est donnée sous la forme d'une chaîne de caractères.
Nous souhaitons disposer d'un programme informatique qui effectue les 3 opérations suivantes :

1) Vérification de la syntaxe de l'équation chimique
2) Application de la loi de conservation des éléments
3) Application de la loi de conservation des masses

Lors de la première opération on s'assurera que les symboles utilisés font bien partie du tableau de **Mendeleïev** [2], à moins qu'il ne s'agisse de symboles spéciaux encadrés par deux caractères | qui servent de délimiteurs. La deuxième opération revient à vérifier que les deux membres de l'équation sont constitués des mêmes éléments. La troisième opération consiste à trouver la valeur exacte des coefficients stœchiométriques permettant d'équilibrer l'équation chimique. Il est souvent possible de procéder par tâtonnements, mais il est préférable d'appliquer une méthode algébrique pour automatiser le calcul. On aboutit à un système linéaire de m équations à n inconnues : m est le nombre d'éléments différents qui interviennent dans la réaction chimique et n est le nombre de coefficients stœchiométriques à déterminer. D'une manière générale, la résolution d'un tel système nécessite de connaître son rang r et de rechercher les équations et les inconnues principales afin de se ramener à un système de **Cramer** (théorème de **Rouché - Fontené**). Parmi l'infinité de solutions, on retiendra uniquement celle qui donne les plus petits coefficients stœchiométriques entiers strictement positifs.

Un premier exemple

Soit l'équation chimique à équilibrer : $a\ Br_2 + b\ NaOH \rightarrow c\ NaBrO_3 + d\ NaBr + e\ H_2O$ (a, b, c, d, e sont des inconnues)
On convient de représenter cette équation par la chaîne de caractères : "Br2 + NaOH = NaBrO3 + NaBr + H2O"
Ainsi, dans cette expression symbolique :

- les coefficients stœchiométriques ne sont pas mentionnés
- les atomes sont suivis de 0 ou 1 nombre strictement positif
- les formules chimiques sont séparées par des signes +
- les 2 membres de l'équation sont séparés par le signe =

La loi de conservation des masses et des éléments peut se résumer à un tableau (tableau de **Lavoisier**) :

	a	*b*	*c*	*d*	*e*
Br	2	0	1	1	0
H	0	1	0	0	2
Na	0	1	1	1	0
O	0	1	3	0	1

La conservation des éléments est vérifiée puisqu'aucune ligne ne comporte une partie nulle (les parties sont séparées par ||).
La conservation des masses se traduit par un système linéaire homogène (réactifs comptés -, produits comptés +) :

$$\begin{pmatrix} -2 & 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & 0 & 2 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & -1 & 3 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{avec} \quad a > 0,\ b > 0,\ c > 0,\ d > 0,\ e > 0$$

La solution entière minimale donne l'équation chimique équilibrée : $3\ Br_2 + 6\ NaOH \rightarrow NaBrO_3 + 5\ NaBr + 3\ H_2O$

(1) **Antoine Laurent de Lavoisier (1743 – 1794), chimiste français, précurseur de la chimie moderne.**
(2) **Dimitri Ivanovitch Mendeleïev (1834 – 1907), chimiste russe, auteur de la classification périodique des éléments.**
(3) **Albert Einstein (1879 – 1955), physicien américain d'origine allemande, créateur de la théorie de la relativité.**

Mathématique & Informatique

Equation électrochimique

Il s'agit d'appliquer les 3 lois fondamentales qui régissent une réaction chimique :

- Loi de conservation des éléments
- Loi de conservation des masses
- Loi de conservation des charges

On ajoute à la formule chimique d'un ion, positif ou négatif, la valeur de sa charge sous la forme (*n*+) ou (*n*–) dans laquelle le nombre entier $n > 0$ désigne le facteur multiplicatif de l'unité de charge élémentaire égale à $1.602\ 10^{-19}$ coulombs.
Par exemple : le cation aluminium Al^{3+} s'écrira Al(3+), l'anion ferrocyanure $[Fe(CN)_6]^{4-}$ s'écrira [Fe(CN)6](4–).
On adoptera, en outre, les notations équivalentes suivantes :

(1+) ⇔ (+)
(1–) ⇔ (–)
(2+) ⇔ (++)
(2–) ⇔ (––)
(3+) ⇔ (+++)
(3–) ⇔ (–––)

Ainsi, les ions Ca^{++} et PO_4^{---} constituant le phosphate tricalcique $Ca_3(PO_4)_2$ en solution s'écriront Ca(++) et PO4(–––).
L'électron e^-, le proton H^+, l'ion hydroxyde OH^-, l'ion hydronium H_3O^+ seront notés respectivement e(–), H(+), OH(–), H3O(+).

Prenons, par exemple, la réaction de réduction de l'ion dichromate en ion chrome :

$$\alpha\ Cr_2O_7^{2-} + \beta\ H^+ + \gamma\ e^- \rightarrow \delta\ Cr^{3+} + \varepsilon\ H_2O$$

Cette équation à équilibrer sera représentée par la chaîne de caractères :

"Cr2O7(2–) + H(+) + e(–) = Cr(3+) + H2O"

La loi de conservation des éléments est vérifiée car les 2 membres de l'équation sont constitués des mêmes éléments : Cr, H, O.
La loi de conservation des masses fournit 3 équations algébriques :

1) Pour Cr : $2\alpha = \delta$
2) Pour H : $\beta = 2\varepsilon$
3) Pour O : $7\alpha = \varepsilon$

Ajoutons une 4^e équation exprimant la loi de conservation des charges :

4) Electroneutralité : $-2\alpha + \beta - \gamma = 3\delta$

Nous obtenons un système linéaire homogène de 4 équations à 5 inconnues :

$$\begin{pmatrix} -2 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 2 \\ -7 & 0 & 0 & 0 & 1 \\ 2 & -1 & 1 & 3 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ \varepsilon \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

La plus petite solution entière non nulle est : $\alpha = 1, \beta = 14, \gamma = 6, \delta = 2, \varepsilon = 7$
Donc, l'équation équilibrée est :

$$Cr_2O_7^{2-} + 14\ H^+ + 6\ e^- \rightarrow 2\ Cr^{3+} + 7\ H_2O$$

La fonction ***STOECHIOMETRIE*** permet d'équilibrer une équation chimique quelconque. Précisons que cette fonction utilise le module "Scilab-Arithmetique", donc que les coefficients stœchiométriques peuvent éventuellement être de très grands nombres entiers.

Mathématique & Informatique

La résolution des systèmes d'équations linéaires (cas général)

Considérons un système linéaire quelconque AX = B de m équations à n inconnues :

$$\begin{pmatrix} a_{11} & \cdots & \cdots & a_{1n} \\ \vdots & & & \vdots \\ a_{m1} & \cdots & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

A l'aide de permutations de lignes et de colonnes, suivies de divisions par un facteur constant associées à des combinaisons linéaires sur les lignes, on met progressivement le système sous la forme échelonnée suivante :

$$\begin{pmatrix} 1 & \cdots & a'_{1,r} & a'_{1,r+1} & \cdots & \cdots & a'_{1,n} \\ 0 & \ddots & \vdots & \vdots & & & \vdots \\ \vdots & \ddots & 1 & a'_{r,r+1} & \cdots & \cdots & a'_{r,n} \\ 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 \\ \vdots & & \vdots & \vdots & & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 \end{pmatrix} \begin{pmatrix} x'_1 \\ \vdots \\ x'_r \\ x'_{r+1} \\ \vdots \\ \vdots \\ x'_n \end{pmatrix} = \begin{pmatrix} b'_1 \\ \vdots \\ b'_r \\ b'_{r+1} \\ \vdots \\ b'_m \end{pmatrix}$$

Le vecteur $X' = (x'_1, \ldots, x'_n)$ est une permutation du vecteur $X = (x_1, \ldots, x_n)$.
Le rang r du système linéaire, c'est-à-dire le nombre de lignes linéairement indépendantes, est égal au nombre de 1 sur la diagonale de la matrice triangulaire supérieure. Les inconnues $x'_1, \ldots, x'_r$ sont les inconnues principales, les inconnues $x'_{r+1}, \ldots, x'_n$ sont les inconnues non principales.
Deux cas peuvent se présenter :

1) Si $(b'_{r+1}, \ldots, b'_m) \neq (0, \ldots, 0)$, le système est impossible, il n'admet pas de solution.

2) Si $(b'_{r+1}, \ldots, b'_m) = (0, \ldots, 0)$, il y a indétermination d'ordre $n - r$.
On attribue des valeurs au choix à $x'_{r+1}, \ldots, x'_n$ et on peut facilement résoudre le système triangulaire restant en calculant dans l'ordre $x'_r, \ldots, x'_1$. On a :

$$x'_i = b'_i - \sum_{j=i+1}^{n} a'_{i,j}\, x'_j \qquad (i = r, \ldots, 1)$$

Si $r = n$, la solution est unique. Si $r < n$, il y a une infinité de solutions.

La structure générale de la fonction Scilab de résolution du système linéaire AX = B est la suivante :

```
function X = LINEAIRE(A,B,X0)
|
|
|
endfunction
```

A = matrice rectangulaire de dimension (m,n)
B = vecteur colonne de dimension (m,1)
X = vecteur colonne de dimension (n,1)
X0 = vecteur colonne de dimension (n,1)

X0 est le vecteur des valeurs x_i^0 à attribuer aux variables x_i de X en cas d'indétermination : $x_i = x_i^0 = x'_{r+k}$.
La fonction ***LINEAIRE*** ne renvoie qu'une seule solution X, dépendante de X0 si $r < n$.

Remarques

- Lorsque $m = n = r$, la matrice A est carrée et régulière, la solution X est unique : $X = A^{-1}B$. Dans ce cas, l'algorithme se simplifie : il n'est pas nécessaire de permuter les colonnes de A et le vecteur X0 devient inutile. Pour un système cramérien, autrement dit pour un système linéaire de n équations à n inconnues de rang n, l'instruction X = ***GAUSS***(A,B) décrite antérieurement, donne le même résultat que l'instruction X = ***LINEAIRE***(A,B,X0) quelle que soit la valeur de X0.
- Si B = 0, le système est homogène. Il admet toujours au moins la solution banale X = 0.
- Dans le cas où le système est impossible, la fonction ***LINEAIRE*** retourne l'ensemble vide (X est la matrice non définie).
- Pour résoudre le système linéaire associé à une équation chimique à équilibrer, on prend $X0 = (1, 1, \ldots, 1) \in \mathbb{R}^n$.

Résolution dans $\mathbb{R}$ (solution approximative)

```
function X = lineaire(A,B,X0)
    m = size(A,1)
    n = size(A,2)
    I = [1:n]
    r = 0
    for k = 1:min(m,n)
        zero = %t
        for i = k:m
            for j = k:n
                zero = A(i,j) == 0
                if ~zero then, break, end
            end
            if ~zero then, break, end
        end
        if zero then, break, end
        r = r + 1
        if i <> k then
            for q = k:n
                t = A(k,q)
                A(k,q) = A(i,q)
                A(i,q) = t
            end
            t = B(k)
            B(k) = B(i)
            B(i) = t
        end
        if j <> k then
            for p = 1:m
                t = A(p,k)
                A(p,k) = A(p,j)
                A(p,j) = t
            end
            t = I(k)
            I(k) = I(j)
            I(j) = t
        end
        a = A(k,k)
        if a <> 1 then
            for q = k+1:n
                A(k,q) = A(k,q)/a
            end
            B(k) = B(k)/a
        end
        for i = k+1:m
            a = A(i,k)
            if a <> 0 then
                for q = k+1:n
                    A(i,q) = A(i,q) - a*A(k,q)
                end
                B(i) = B(i) - a*B(k)
            end
        end
    end
```

```
    for i = r+1:m
        if B(i) <> 0 then
            X = []
            return
        end
    end
    X = X0
    Y = X
    for i = r+1:n
        Y(i) = X(I(i))
    end
    for i = r:-1:1
        s = 0
        for j = i+1:n
            s = s + A(i,j)*Y(j)
        end
        Y(i) = B(i) - s
        X(I(i)) = Y(i)
    end
endfunction
```

Résolution dans ℚ (solution exacte)

Il s'agit de la traduction du programme précédent en utilisant les fonctions de manipulation des grands nombres rationnels.

```
function X = LINEAIRE(A,B,X0)
    m = A($-1)
    n = A($)
    I = [1:n]
    r = 0
    for k = 1:min(m,n)
        zero = %t
        for i = k:m
            for j = k:n
                zero = F_NUL(A((i-1)*n+j))
                if ~zero then, break, end
            end
            if ~zero then, break, end
        end
        if zero then, break, end
        r = r + 1
        if i <> k then
            for q = k:n
                t = A((k-1)*n+q)
                A((k-1)*n+q) = A((i-1)*n+q)
                A((i-1)*n+q) = t
            end
            t = B(k)
            B(k) = B(i)
            B(i) = t
        end
        if j <> k then
            for p = 1:m
                t = A((p-1)*n+k)
                A((p-1)*n+k) = A((p-1)*n+j)
                A((p-1)*n+j) = t
            end
            t = I(k)
            I(k) = I(j)
            I(j) = t
        end
        a = A((k-1)*n+k)
        if F_COMPARAISON(a,FRACTION(1,1)) <> 0 then
            for q = k+1:n
                A((k-1)*n+q) = F_DIVISION(A((k-1)*n+q),a)
            end
            B(k) = F_DIVISION(B(k),a)
        end
        for i = k+1:m
            a = A((i-1)*n+k)
            if ~F_NUL(a) then
                for q = k+1:n
                    t = F_MULTIPLICATION(a,A((k-1)*n+q))
                    A((i-1)*n+q) = F_SOUSTRACTION(A((i-1)*n+q),t)
                end
                t = F_MULTIPLICATION(a,B(k))
                B(i) = F_SOUSTRACTION(B(i),t)
            end
        end
    end
```

```
    for i = r+1:m
        if ~F_NUL(B(i)) then
            X = list(0,0)
            return
        end
    end
    X = X0
    Y = X
    for i = r+1:n
        Y(i) = X(I(i))
    end
    for i = r:-1:1
        s = FRACTION(0,1)
        for j = i+1:n
            t = F_MULTIPLICATION(A((i-1)*n+j),Y(j))
            s = F_ADDITION(s,t)
        end
        Y(i) = F_SOUSTRACTION(B(i),s)
        X(I(i)) = Y(i)
    end
endfunction
```

Mathématique & Informatique

Exemple

Trouver une solution entière non nulle au système linéaire de 9 équations à 10 inconnues suivant :

$$\begin{pmatrix} 1/4 & 1/9 & 1/16 & 1/25 & 1/36 & 1/49 & 1/64 & 1/81 & 1/100 & 1/121 \\ 1/9 & 1/16 & 1/25 & 1/36 & 1/49 & 1/64 & 1/81 & 1/100 & 1/121 & 1/144 \\ 1/16 & 1/25 & 1/36 & 1/49 & 1/64 & 1/81 & 1/100 & 1/121 & 1/144 & 1/169 \\ 1/25 & 1/36 & 1/49 & 1/64 & 1/81 & 1/100 & 1/121 & 1/144 & 1/169 & 1/196 \\ 1/36 & 1/49 & 1/64 & 1/81 & 1/100 & 1/121 & 1/144 & 1/169 & 1/196 & 1/225 \\ 1/49 & 1/64 & 1/81 & 1/100 & 1/121 & 1/144 & 1/169 & 1/196 & 1/225 & 1/256 \\ 1/64 & 1/81 & 1/100 & 1/121 & 1/144 & 1/169 & 1/196 & 1/225 & 1/256 & 1/289 \\ 1/81 & 1/100 & 1/121 & 1/144 & 1/169 & 1/196 & 1/225 & 1/256 & 1/289 & 1/324 \\ 1/100 & 1/121 & 1/144 & 1/169 & 1/196 & 1/225 & 1/256 & 1/289 & 1/324 & 1/361 \end{pmatrix} \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \\ N_7 \\ N_8 \\ N_9 \\ N_{10} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

On écrit :

```
exec("Scilab-Equations.sce",-1)
m = 9 ; n = 10 ;
A = list() ;
for i = 1:m
    for j = 1:n
        A($+1) = FRACTION(1,(i+j)^2) ;
    end
end
A($+1) = m ; A($+1) = n ;
B = M_ZERO(m,1) ;
X0 = M_UN(n,1) ;
X = LINEAIRE(A,B,X0) ;
L1 = list() ; L2 = list() ;
for i = 1:n
    L1($+1) = NUMERATEUR(X(i)) ;
    L2($+1) = DENOMINATEUR(X(i)) ;
end
[pgcdl1,ppcml1] = PGCDPPCM(L1) ;
[pgcdl2,ppcml2] = PGCDPPCM(L2) ;
N = list() ;
for i = 1:n
    [q1,r1] = DIVISION(NUMERATEUR(X(i)),pgcdl1) ;
    [q2,r2] = DIVISION(ppcml2,DENOMINATEUR(X(i))) ;
    N($+1) = MULTIPLICATION(q1,q2) ;
end
for i = 1:n
    printf('\n N%i = ',i), ECRIRE(N(i)) ;
end
```

On obtient :

```
N1  = - 131193077785449858951723631885481596533579213025 00
N2  = 80263392960552251620246059235527618197535454523062 5
N3  = - 15088979833135340406822753868950952383330743197440000
N4  = 13068991055197699400993293362206079793574168518287 0000
N5  = - 61529652552889708669459890749882735500597755549891 7104
N6  = 17011264625048424019807487410813434499989401250556155 00
N7  = - 28371090141346659443067817225679350649036644093248 00000
N8  = 28046674132663581169992881330867503642646016360496560 00
N9  = - 15119502279786847024202933384577154931051883142898940 00
N10 = 34219691450804968446127918021341646404480437183826348 4
```

L'ensemble des solutions du système linéaire est un espace vectoriel de dimension 1 engendré par cette solution particulière.

Mathématique & Informatique

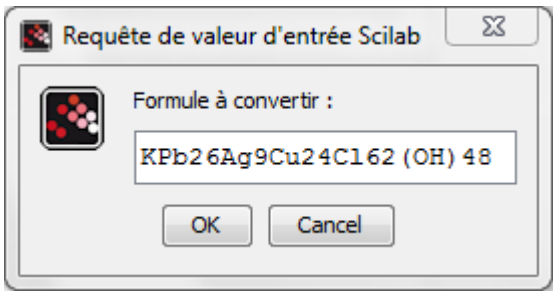
La conversion des formules chimiques

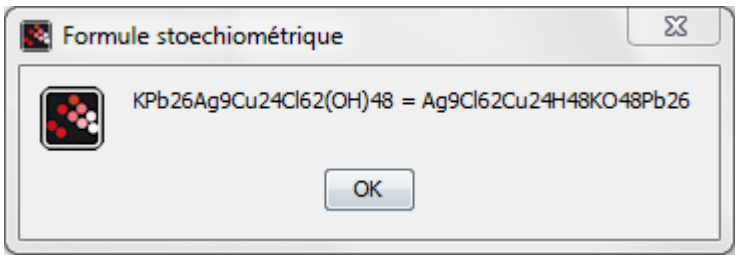
A partir d'une formule chimique composée de sous-formules encadrées par des parenthèses indicées, on souhaite obtenir une expression élémentaire sans parenthèses où chaque atome n'apparaît qu'une seule fois.
La fonction *conversion* transforme une formule chimique complexe ***F***, écrite avec des lettres (majuscules et minuscules), des chiffres, des parenthèses, en une formule chimique abrégée ***f***.
La formule ***F*** est le plus souvent une formule semi-développée qui met en évidence la structure et/ou les fonctions d'une molécule. La formule ***f*** est appelée formule brute ou formule stœchiométrique, elle indique le nombre d'atomes de chaque sorte (classés par ordre alphabétique) présents dans une molécule. Plusieurs molécules différentes peuvent avoir la même formule stœchiométrique : F1 et F2 isomères ⇔ *conversion*(F1) = *conversion*(F2)
NB Pour plus de lisibilité dans les formules, on utilisera aussi les crochets et les accolades en remplacement des parenthèses.

```
init = '' ;
while %t
    F = x_dialog('Formule à convertir :',init) ;
    if F == [] then, break, end
    f = conversion(F) ;
    if f == [] then
        messagebox('Conversion impossible.','Erreur','error','OK','modal') ; init = F ;
    else
        messagebox(F + ' = ' + f,'Formule stoechiométrique','scilab','OK','modal') ; init = '' ;
    end
end
```

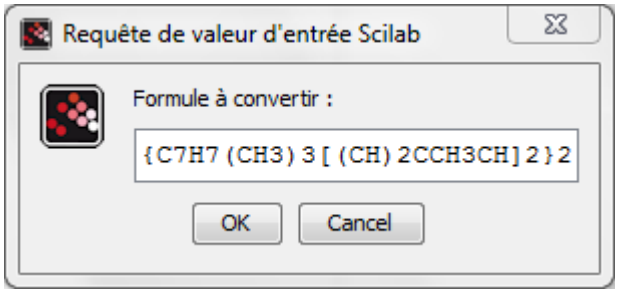
Trois exemples :

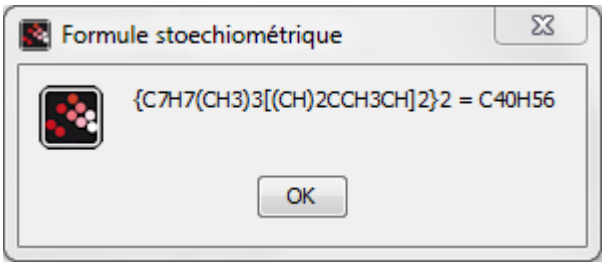
- $KPb_{26}Ag_9Cu_{24}Cl_{62}(OH)_{48} = Ag_9Cl_{62}Cu_{24}H_{48}KO_{48}Pb_{26}$ (boléite)



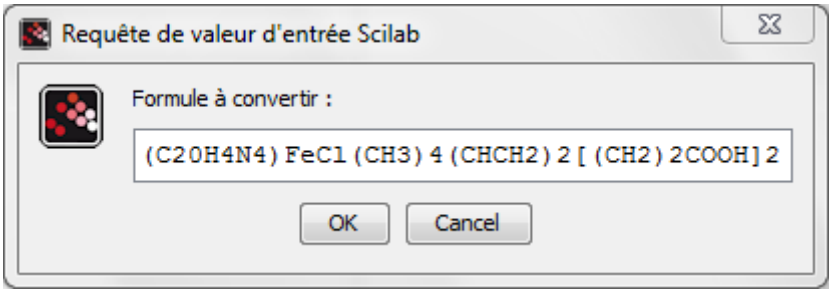


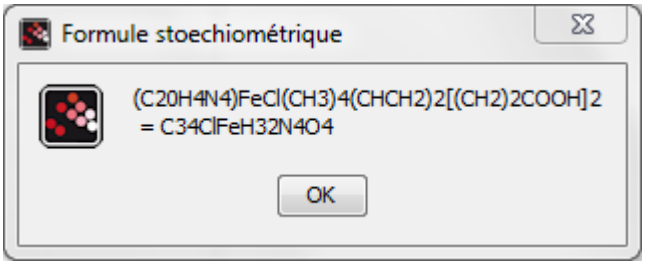
- $\{C_7H_7(CH_3)_3[(CH)_2CCH_3CH]_2\}_2 = C_{40}H_{56}$ (β-carotène)





- $(C_{20}H_4N_4)FeCl(CH_3)_4(CHCH_2)_2[(CH_2)_2COOH]_2 = C_{34}ClFeH_{32}N_4O_4$ (hémine)





```
function f= conversion(F)
    // Vérifier la formule F
    F = strcat(F)
    F = strsubst(F,' ','')
    check1 = regexp(F,'/[^A-Za-z0-9()[]{}]/') <> []
    check2 = regexp(F,'/[^A-Za-z0-9)]}]\d/') <> []
    check3 = regexp(F,'/(^|[0-9()[]{}])[a-z]/') <> []
    if check1 | check2 | check3 then
       f = []
       return
    end
    // Introduire les opérateurs + et *
    S = ''
    s = ''
    for i = 1:length(F)
        c = part(F,i)
        if regexp(c,'/[A-Z([{]/') <> [] then
            S = S + '+'
        end
        if isdigit(c) & ~isdigit(s) then
            S = S + '*'
        end
        S = S + c
        s = c
    end
    // Rechercher les atomes différents
    A = ''
    for i = 1:length(S)
        c = part(S,i)
        if isletter(c) then
            A = A + c
        else
            A = A + ' '
        end
    end
    A = tokens(A)
    A = unique(A)
    A = gsort(A)
    // Construire la formule f
    f = ''
    for i = 1:size(A,1)
        N = S
        for j = 1:size(A,1)
            if i == j then
                N = strsubst(N,A(j),'1')
            else
                N = strsubst(N,A(j),'0')
            end
        end
        err = execstr('N = evstr(N)','errcatch')
        if err <> 0 then
            f = []
            return
        end
        if N == 1 then
            f = A(i) + f
        end
        if N > 1 then
            f = A(i) + string(N) + f
        end
    end
endfunction
```

Mathématique & Informatique

Préambule

Les symboles des éléments sont généralement la première lettre de leur nom (par exemple : C = Carbone, H = Hydrogène, O = Oxygène, …), ou la première lettre suivie d'une autre lettre de leur nom (par exemple : Cu = Cuivre, Mg = Magnésium, Pb = Plomb, …). Il y a huit exceptions :
Au, du latin Aurum = Or
Hg, du latin Hydrargyrum = Mercure
K, du latin Kalium = Potassium
N, du latin Nitrogenium = Azote
Na, du latin Natrium = Sodium
Sb, du latin Stibium = Antimoine
Sn, du latin Stannum = Etain
W, de l'allemand Wolfram = Tungstène
A l'état naturel, il existe 92 éléments : le plus léger est l'hydrogène $^{1}_{1}H$, le plus lourd est l'uranium $^{238}_{92}U$.
Au-delà du numéro atomique Z = 92 de l'uranium, on peut fabriquer artificiellement quelques atomes (les transuraniens).

En chimie organique, on utilise fréquemment les symboles spéciaux suivants :
R, R', R" : radical hydrocarboné (reste de formule C_pH_q)
X : halogène (Cl, Br, I, F)
Me : métal (Na, K, Li, Mg, Cd, Zn, Al, Fe, Cr …)
Dans une équation, on écrira |R|, |R'|, |R"|, |X|, |Me| pour ne pas entraîner une erreur de syntaxe.
Autre exemple, pour simplifier la formule d'une molécule, on désigne parfois par P le radical propanoïque. Afin de lever toute ambiguïté avec l'atome de phosphore (symbole P), le radical $(CH_2)_2COOH$ sera représenté par |P| dans les deux membres d'une équation. De même, le reste aromatique C_6H_5 (phényle) que l'on rencontre sous la forme abrégée Ar, s'écrira |Ar| dans une équation pour ne pas le confondre avec l'Argon. Même méthode avec un polymère $(C_pH_q)_n$ constitutif d'une macromolécule : on le notera |(CpHq)n|, sans définir l'entier n.
En fait, lors de l'exécution du programme de stœchiométrie, tout se passe comme si une chaîne de caractères quelconques (hormis |, /, \, +, =), encadrée par deux barres verticales, était le symbole d'un nouvel élément du tableau de **Mendeleïev**.

Certains corps ont une formule chimique présentée sous forme hydratée avec une terminaison en $.H_2O$ ou $.nH_2O$ (n = 2, 3, …). Par exemple, la formule de l'hydrate d'hydrazine est $N_2H_4.H_2O$ que l'on écrira N2H4.H2O dans une équation. C'est aussi le cas de très nombreux minéraux qui se trouvent à l'état naturel dans l'écorce terrestre. Citons la turquoise de formule chimique $CuAl_6(PO_4)_4(OH)_8.4H_2O$, soit CuAl6(PO4)4(OH)8.4H2O sans mettre les nombres en indice.
Du point de vue stœchiométrique, l'entité .nH2O sera traduite en (H2O)n dans les calculs.

Faire attention à bien différencier le chiffre 0, la lettre majuscule O et la lettre minuscule o lors de la saisie des équations : l'eau s'écrit H2O (≠ H20), le cyclodécane s'écrit C10H20 (≠ C10H2O), le monoxyde de carbone CO, le cobalt Co …

Lors de la saisie de l'équation, certaines syntaxes ne sont pas autorisées. Par exemple, la représentation du pentanol par la formule semi-développée CH3(4CH2)OH n'est pas admise, il faut écrire CH3(CH2)4OH ou l'expression équivalente CH3(C4H8)OH. Précisons qu'aucune vérification sémantique des formules chimiques n'est effectuée. Ainsi, une formule brute telle que C4H12 est considérée comme correcte bien qu'elle ne corresponde à aucun composé organique réel (dans la formule générale C_nH_m d'un hydrocarbure, le nombre m est toujours pair et inférieur ou égal à $2n + 2$).

Si l'on souhaite indiquer que la réaction chimique a lieu dans les deux sens, on séparera les deux membres de l'équation par un double signe = (==). Par exemple, Ag == Ag(+) + e(–) est équivalent à $Ag \rightleftarrows Ag^+ + e^-$ (équation du couple redox Ag/Ag^+).

Pour spécifier une caractéristique physique d'un composé chimique, on rajoutera à la fin de sa formule un commentaire placé entre deux barres obliques (/commentaire/). Par exemple, H2O/l/ désigne l'eau à l'état liquide et H2O/g/ l'eau à l'état gazeux.

Une équation chimique équilibrée doit obligatoirement vérifier la loi de conservation des charges, en plus de la loi de conservation des masses et des éléments. C'est pourquoi la réaction d'oxydation de l'ion ferreux en ion ferrique par le chlore s'écrit $2\ Fe^{2+} + Cl_2 \rightarrow 2\ Fe^{3+} + 2\ Cl^-$ et non pas $Fe^{2+} + Cl_2 \rightarrow Fe^{3+} + 2\ Cl^-$ puisque, dans cette dernière équation, la charge globale du membre de gauche est égale à +2 alors que la charge globale du membre de droite est égale à +1.

Dans une équation bilan du type $a_1A_1 + a_2A_2 + \ldots + a_iA_i + \ldots + a_rA_r \rightarrow a'_1A'_1 + a'_2A'_2 + \ldots + a'_iA'_i + \ldots + a'_pA'_p$, si l'on désigne par m_i la masse d'une mole de réactif A_i et par m'_i la masse d'une mole de produit A'_i (1 mole de A_i ou A_i' est égale à N_A entités élémentaires A_i ou A'_i ; la constante $N_A \approx 6.022\ 10^{23}$ est le nombre d'**Avogadro**), alors les coefficients stœchiométriques a_i et a'_i sont les plus petites valeurs entières strictement positives telles que : $\sum_{i=1}^{i=r} a_i m_i = \sum_{i=1}^{i=p} a'_i m'_i$.
Noter que l'on a pas nécessairement $\sum_{i=1}^{i=r} a_i = \sum_{i=1}^{i=p} a'_i$, c'est-à-dire que le nombre de moles de réactifs peut être différent du nombre de moles de produits.

Mathématique & Informatique

Le programme de stœchiométrie

```
//-------------------------------------------------------------------------------------
//   Calcul des coefficients stoechiométriques d'une équation chimique
//-------------------------------------------------------------------------------------

//----------------------------------
//   Loïc Vrillon (Avril 2013)
//----------------------------------

funcprot(0) ; lines(0)
exec("Scilab-Equations.sce",-1)

function STOECHIOMETRIE()
    text = ['Ce programme permet d''équilbrer une équation chimique.' ; ' ' ; ...
            '1) Equation chimique sans les charges, par exemple :' ; ...
            'H2 + O2 = H2O   ==>   2 H2 + O2 = 2 H2O' ; ' ' ; ...
            '2) Equation chimique avec les charges, par exemple :' ; ...
            'O2 + H(+) + e(-) = H2O2   ==>   O2 + 2 H(+) + 2 e(-) = H2O2']
    messagebox(text,'Stoechiométrie','info','OK','modal')
    // Lecture des équations chimiques à équilibrer
    init = ''
    while %t
        EQ = x_dialog('Equation chimique à équilibrer :',init)
        if EQ == [] then, break, end
        init = EQ
        EQ = strcat(EQ,' ')
        next = stoechiometrie(EQ)
        if next then, init = '', end
    end
endfunction

function next=stoechiometrie(EQ)
    // Syntaxe d'une équation chimique
    electron = 'e'
    atome = ...
    '(Ac|Ag|Al|Am|Ar|As|At|Au|B|Ba|Be|Bh|Bi|Bk|Br|C|Ca|...
    Cd|Ce|Cf|Cl|Cm|Co|Cr|Cs|Cu|Db|Ds|Dy|Er|Es|Eu|F|Fe|...
    Fm|Fr|Ga|Gd|Ge|H|He|Hf|Hg|Ho|Hs|I|In|Ir|K|Kr|La|Li|...
    Lr|Lu|Md|Mg|Mn|Mo|Mt|N|Na|Nb|Nd|Ne|Ni|No|Np|O|Os|P|...
    Pa|Pb|Pd|Pm|Po|Pr|Pt|Pu|Ra|Rb|Re|Rf|Rh|Rn|Ru|S|Sb|Sc|...
    Se|Sg|Si|Sm|Sn|Sr|Ta|Tb|Tc|Te|Th|Ti|Tl|Tm|U|Uub|Uuh|...
    Uuo|Uup|Uuq|Uus|Uut|Uuu|V|W|Xe|Y|Yb|Zn|Zr)'
    symbole = '\|[^\|\/\\\+\=]+\|'
    separateur = '[\(\)\[\]\{\}]'
    nombre = '[1-9]\d*'
    hydrate = '\.([1-9]\d*)?H2O'
    commentaire = '\/[^\|\/\\\+\=]+\/'
    charge = '\(([1-9]\d*(\+|\-)|\+{1,3}|\-{1,3})\)'
    moins = '\((-|1-)\)'
    corps = '((' + atome + '|' + symbole + '|' + separateur + ')(' + nombre + ')?)+'
    formule = corps + '(' + hydrate + ')?(' + charge + ')?(' + commentaire + ')?'
    terme = '(' + formule + '|' + electron + '(' + moins + ')?)'
    membre = terme + '(\s*\+\s*' + terme + ')*'
    equation = '/^\s*' + membre + '\s*(=|==)\s*' + membre + '\s*$/'
```

Mathématique & Informatique

```
// Extraction des lettres
function v=alphabetical(s)
    v = ''
    for i = 1:length(s)
        c = part(s,i)
        if isletter(c) then
            v = v + c
        else
            v = v + ' '
        end
    end
    v = tokens(v,' ')
endfunction
// Extraction des chiffres
function v=numerical(s)
    v = ''
    for i = 1:length(s)
        c = part(s,i)
        if isdigit(c) then
            v = v + c
        else
            v = v + ' '
        end
    end
    v = evstr(tokens(v,' '))
endfunction
// Remplacement des crochets et des accolades par des parenthèses
function f=bracket(F)
    f = ''
    stack = list()
    for i = 1:length(F)
        c = part(F,i)
        if c == '(' | c == '[' | c == '{' then
            stack($+1) = c
        end
        if c == ')' | c == ']' | c == '}' then
            if length(stack) == 0 then
                return
            end
            s = stack($)
            stack($) = null()
            if (c == ')' & s <> '(') | (c == ']' & s <> '[') | (c == '}' & s <> '{') then
                return
            end
        end
    end
    if length(stack) == 0 then
        f = strsubst(F,'[','(')
        f = strsubst(f,']',')')
        f = strsubst(f,'{','(')
        f = strsubst(f,'}',')')
    end
endfunction
```

```
//Formule brute
function f=conversion(F)
    // Vérifier la formule F
    check1 = regexp(F,'/[^A-Za-z0-9)]\d/') <> []
    check2 = regexp(F,'/\(\)/') <> []
    if check1 | check2 then
        f = ''
        return
    end
    // Introduire les opérateurs + et *
    S = ''
    s = ''
    for i = 1:length(F)
        c = part(F,i)
        if regexp(c,'/[A-Z(]/') <> [] then
            S = S + '+'
        end
        if isdigit(c) & ~isdigit(s) then
            S = S + '*'
        end
        S = S + c
        s = c
    end
    // Rechercher les atomes différents
    A = ''
    for i = 1:length(S)
        c = part(S,i)
        if isletter(c) then
            A = A + c
        else
            A = A + ' '
        end
    end
    A = tokens(A)
    A = unique(A)
    A = gsort(A)
    // Construire la formule f
    f = ''
    for i = 1:size(A,1)
        N = S
        for j = 1:size(A,1)
            if i == j then
                N = strsubst(N,A(j),'1')
            else
                N = strsubst(N,A(j),'0')
            end
        end
        err = execstr('N = evstr(N)','errcatch')
        if err <> 0 then
            f = ''
            return
        end
        f = A(i) + string(N) + f
    end
endfunction
```

```
// Vérification de l'équation
check = regexp(EQ,equation)
if check == [] then
    messagebox('Syntaxe incorrecte.','Erreur','error','Correction','modal')
    next = %f
    return
end
// Mise en forme de l'équation
EQ = strsubst(EQ,' ','')
sep = ' '
s = 0
while s <> []
    [s,e,S] = regexp(EQ,'/(\w|\)|\]|\}|\||\/)\+[^)]/','o')
    if s <> [] then
        R = strsubst(S,'+',sep)
        EQ = strsubst(EQ,S,R)
    end
end
dbl = strstr(EQ,'==') <> ''
EQ = strsubst(EQ,'==','=')
E = strsubst(EQ,'=',sep)
F = tokens(E,sep)
n = size(F,1)
X = 'X'
s = 0
while s <> []
    [s,e,S] = regexp(EQ,'/' + symbole + '/','o')
    if s <> [] then
        X = X + 'x'
        EQ = strsubst(EQ,S,X)
    end
end
s = 0
while s <> []
    [s,e,S] = regexp(EQ,'/' + hydrate + '/','o')
    if s <> [] then
        k = strsubst(S,'.','')
        k = strsubst(k,'H2O','')
        R = '(H2O)' + k
        EQ = strsubst(EQ,S,R)
    end
end
s = 0
while s <> []
    [s,e,S] = regexp(EQ,'/' + commentaire + '/','o')
    if s <> [] then
        EQ = strsubst(EQ,S,'')
    end
end
EQ = strsubst(EQ,'(+)','(1+)')
EQ = strsubst(EQ,'(-)','(1-)')
EQ = strsubst(EQ,'(++)','(2+)')
EQ = strsubst(EQ,'(--)','(2-)')
EQ = strsubst(EQ,'(+++)','(3+)')
EQ = strsubst(EQ,'(---)','(3-)')
E = tokens(EQ,'=')
F1 = tokens(E(1),sep)
F2 = tokens(E(2),sep)
n1 = size(F1,1)
n2 = size(F2,1)
```

```
// Réactifs
C1 = []
for i = 1:n1
    if F1(i) == electron then
        F1(i) = electron + '(1-)'
    end
    [s,e,S] = regexp(F1(i),'/' + charge + '$/')
    if s <> [] then
        F1(i) = strsubst(F1(i),'/' + charge + '$/','','r')
        c = part(S,length(S)-1)
        S = c + strsubst(S,c,'')
        C1 = [C1,evstr(S)]
    else
        C1 = [C1,0]
    end
    if F1(i) <> electron then
        F1(i) = bracket(F1(i))
        F1(i) = conversion(F1(i))
    else
        F1(i) = electron + '1'
    end
    if F1(i) == '' then
        messagebox('Syntaxe incorrecte.','Erreur','error','Correction','modal')
        next = %f
        return
    end
end
// Produits
C2 = []
for i = 1:n2
    if F2(i) == electron then
        F2(i) = electron + '(1-)'
    end
    [s,e,S] = regexp(F2(i),'/' + charge + '$/')
    if s <> [] then
        F2(i) = strsubst(F2(i),'/' + charge + '$/','','r')
        c = part(S,length(S)-1)
        S = c + strsubst(S,c,'')
        C2 = [C2,evstr(S)]
    else
        C2 = [C2,0]
    end
    if F2(i) <> electron then
        F2(i) = bracket(F2(i))
        F2(i) = conversion(F2(i))
    else
        F2(i) = electron + '1'
    end
    if F2(i) == '' then
        messagebox('Syntaxe incorrecte.','Erreur','error','Correction','modal')
        next = %f
        return
    end
end
```

Mathématique & Informatique

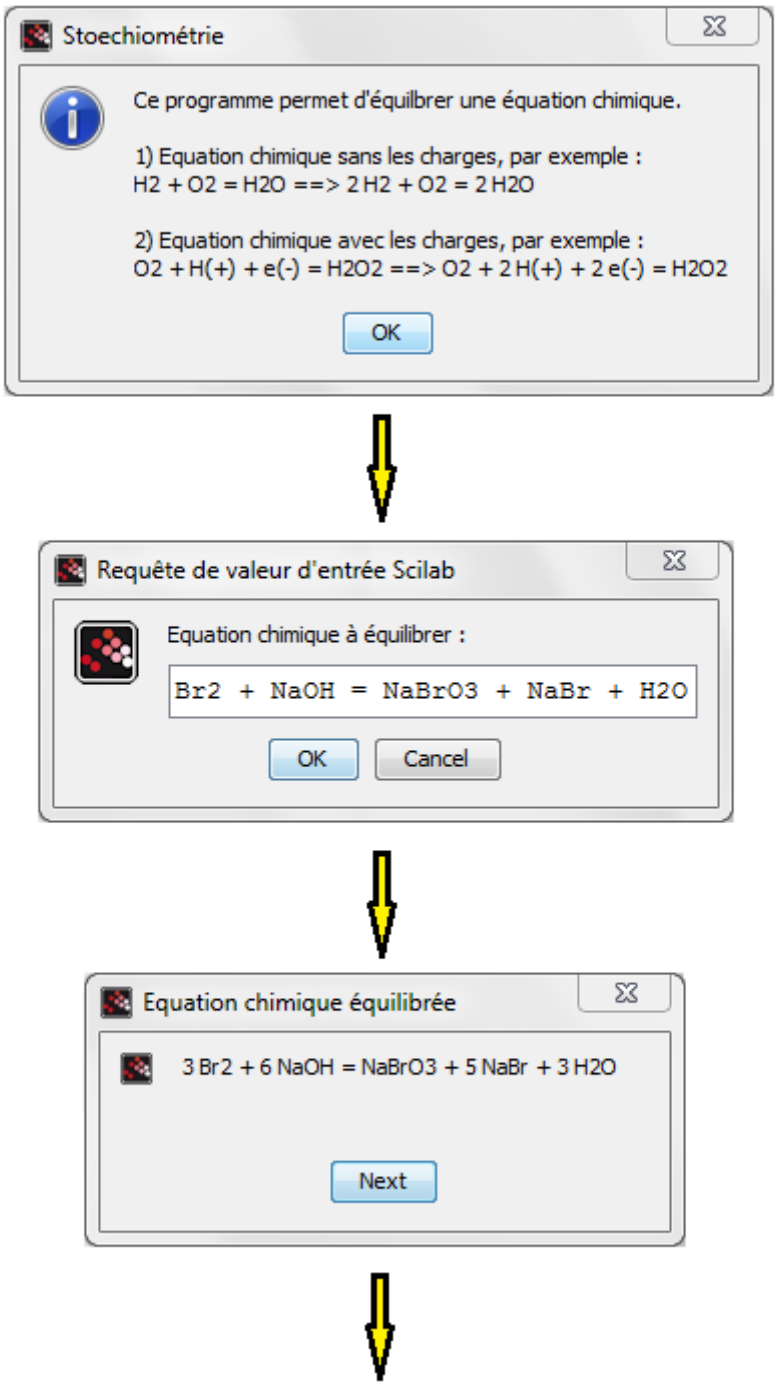
```
// Atomes et Charges
A1 = setdiff(unique(alphabetical(strcat(F1))),electron)
A2 = setdiff(unique(alphabetical(strcat(F2))),electron)
if ~isequal(A1,A2) then
    messagebox('Non conservation des éléments.','Erreur','error','Correction','modal')
    next = %f
    return
end
ATOMS = A1
m = size(ATOMS,1)
C = [C1,C2]
// Création du tableau de Lavoisier
T = zeros(m,n)
for j = 1:n
    if j <= n1 then
        formula = F1(j)
    else
        formula = F2(j-n1)
    end
    if formula <> electron then
        atoms = alphabetical(formula)
        numbers = numerical(formula)
        for k = 1:size(atoms,1)
            i = find(ATOMS == atoms(k))
            T(i,j) = T(i,j) + numbers(k)
        end
    end
end
// Résolution du système linéaire homogène
A = list()
for i = 1:m
    for j = 1:n
        if j <= n1 then
            A($+1) = FRACTION(-T(i,j),1)
        else
            A($+1) = FRACTION(T(i,j),1)
        end
    end
end
if find(C <> 0) <> [] then
    for j = 1:n
        if j <= n1 then
            A($+1) = FRACTION(-C(j),1)
        else
            A($+1) = FRACTION(C(j),1)
        end
    end
    A($+1) = m + 1
    A($+1) = n
    B = M_ZERO(m+1,1)
else
    A($+1) = m
    A($+1) = n
    B = M_ZERO(m,1)
end
X0 = M_UN(n,1)
X = LINEAIRE(A,B,X0)
```

```
// Calcul des coefficients stoechiométriques entiers
L = list()
if ~M_VIDE(X) then
    L1 = list()
    L2 = list()
    for i = 1:n
        L1($+1) = NUMERATEUR(X(i))
        L2($+1) = DENOMINATEUR(X(i))
    end
    [pgcdl1,ppcml1] = PGCDPPCM(L1)
    [pgcdl2,ppcml2] = PGCDPPCM(L2)
    for i = 1:n
        [q1,r1] = DIVISION(NUMERATEUR(X(i)),pgcdl1)
        [q2,r2] = DIVISION(ppcml2,DENOMINATEUR(X(i)))
        L($+1) = MULTIPLICATION(q1,q2)
    end
end
if length(L) == 0 then
    messagebox('Solution non trouvée !','Erreur','error','Next','modal')
    next = %t
    return
end
// Mise en forme du résultat
S = list()
SGN = []
for i = 1:length(L)
    sgn = SIGNE(L(i))
    val = VALEUR(L(i))
    if sgn < 0 then
        s = '-'
    else
        s = ''
    end
    S($+1) = s + strcat(string(val))
    SGN = [SGN,sgn]
end
```

```
    // Ecriture de l'équation chimique équilibrée
    E1 = ''
    for i = 1:n1
        if i <> 1 then
            E1 = E1 + ' + '
        end
        if S(i) == '1' then
            E1 = E1 + F(i)
        else
            E1 = E1 + S(i) + ' ' + F(i)
        end
    end
    E2 = ''
    for i = 1:n2
        if i <> 1 then
            E2 = E2 + ' + '
        end
        if S(n1+i) == '1' then
            E2 = E2 + F(n1+i)
        else
            E2 = E2 + S(n1+i) + ' ' + F(n1+i)
        end
    end
    if dbl then
        EQ = E1 + ' == ' + E2
    else
        EQ = E1 + ' = ' + E2
    end
    if find(SGN <= 0) <> [] then
        messagebox(EQ,'Equation chimique équilibrée avec coefficient négatif ou nul !','warning','Next','modal')
        next = %t
    else
        messagebox(EQ,'Equation chimique équilibrée','scilab','Next','modal')
        next = %t
    end
endfunction

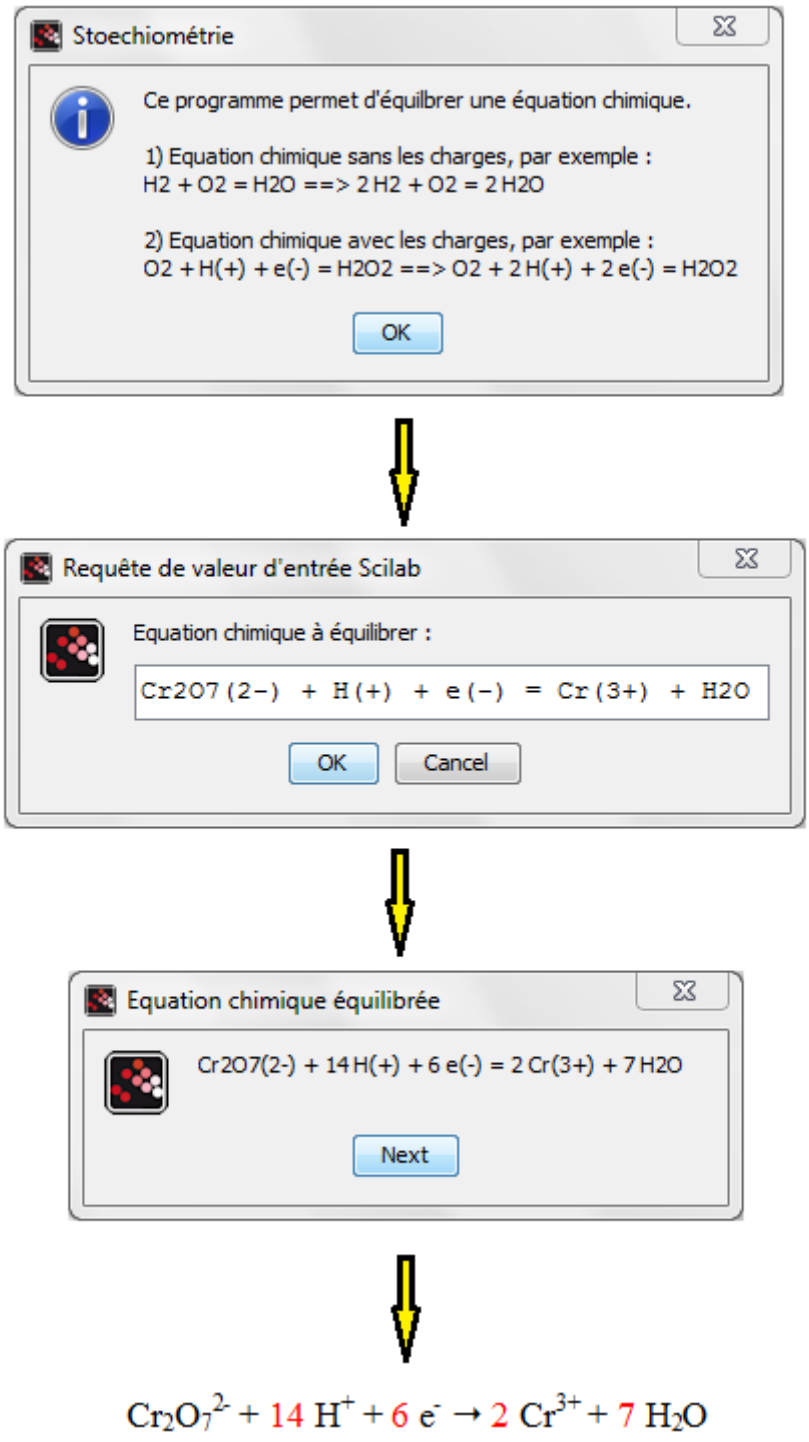
STOECHIOMETRIE()
```

Mathématique & Informatique



$$3\ Br_2 + 6\ NaOH \rightarrow NaBrO_3 + 5\ NaBr + 3\ H_2O$$

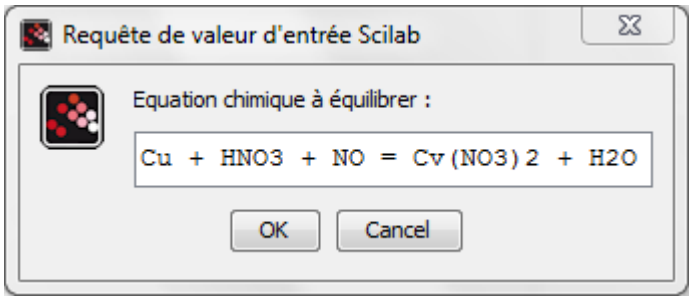
Mathématique & Informatique

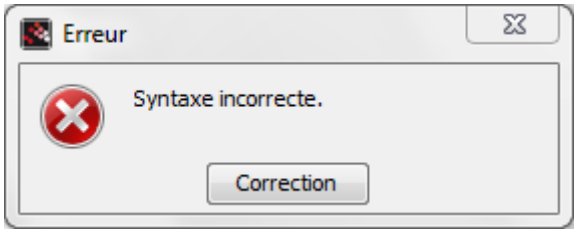


$$Cr_2O_7^{2-} + 14\ H^+ + 6\ e^- \rightarrow 2\ Cr^{3+} + 7\ H_2O$$

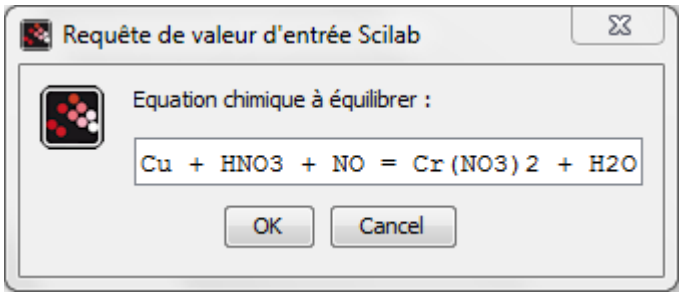
Mathématique & Informatique

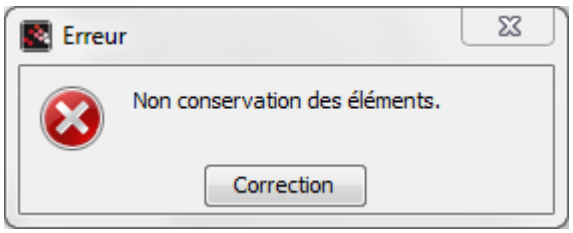
Les messages d'erreur



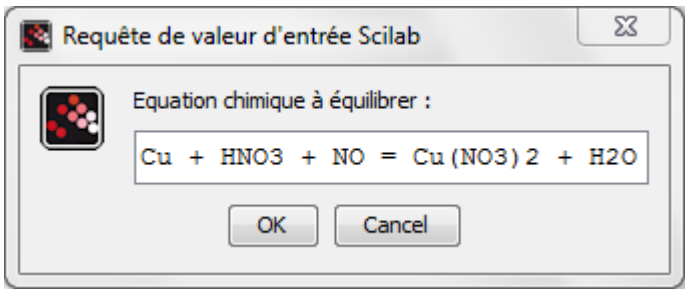


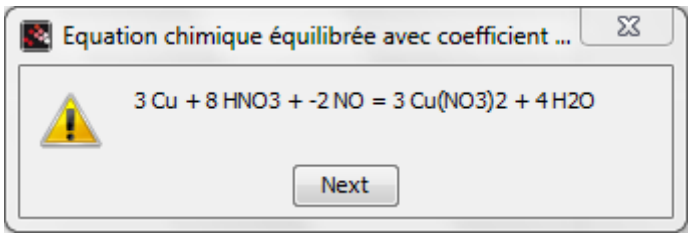
⇨ Le symbole Cv ne fait pas partie du tableau de classification périodique des éléments



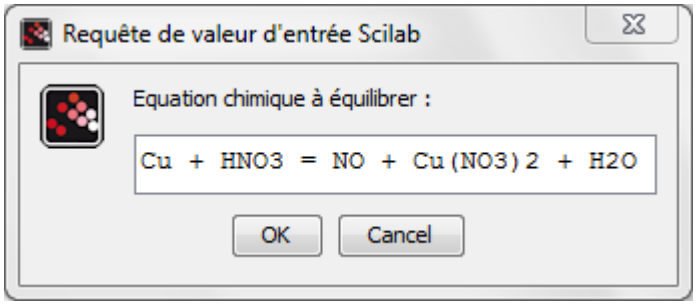


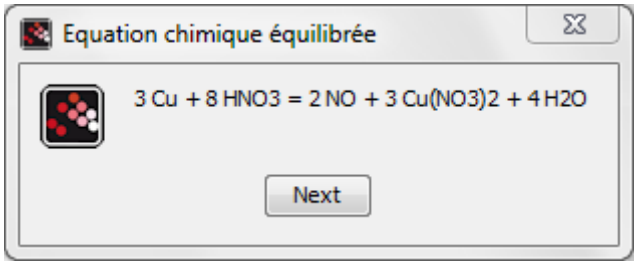
⇨ Les 2 membres de l'équation ne sont pas constitués des mêmes éléments : {Cu,H,N,O} ≠ {Cr,H,N,O}





⇨ Les coefficients stœchiométriques ne sont pas tous strictement positifs : -2 < 0





⇨ En passant NO dans le second membre, on obtient l'équation sous sa forme habituelle, c'est-à-dire avec tous ses coefficients strictement positifs : $3\ Cu + 8\ HNO_3 \rightarrow 2\ NO + 3\ Cu(NO_3)_2 + 4\ H_2O$

Mathématique & Informatique

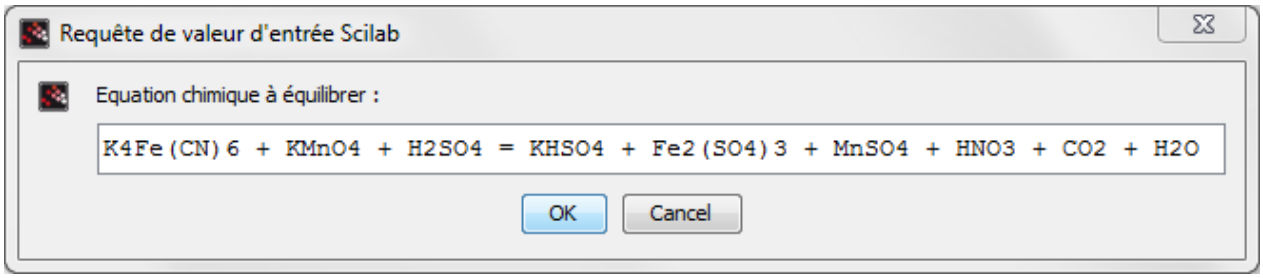
Application

Le nombre de réactions chimiques actuellement connues est estimé à quelques millions.
On trouve de nombreux exemples d'équations chimiques sur le site internet : **http://www.webqc.org/**

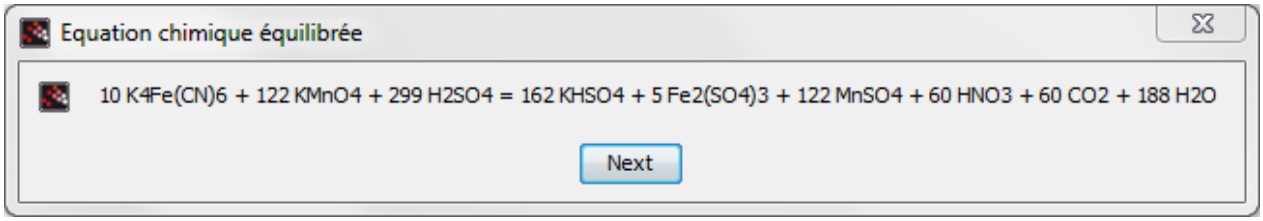
Cherchons à équilibrer :

$a\ K_4Fe(CN)_6 + b\ KMnO_4 + c\ H_2SO_4 \rightarrow d\ KHSO_4 + e\ Fe_2(SO_4)_3 + f\ MnSO_4 + g\ HNO_3 + h\ CO_2 + i\ H_2O$

Pour cela, exécutons la fonction *STOECHIOMETRIE* :



Nous obtenons :

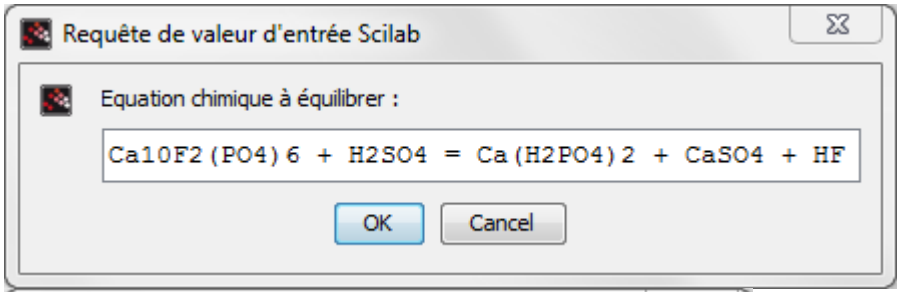


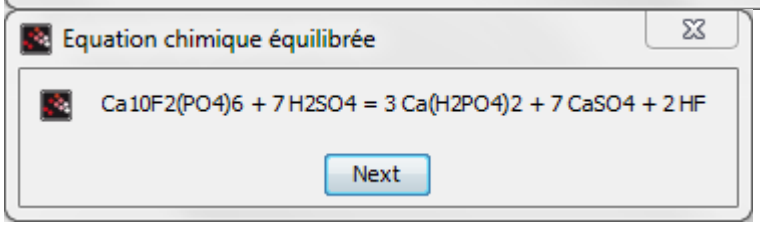
Donc :

$10\ K_4Fe(CN)_6 + 122\ KMnO_4 + 299\ H_2SO_4 \rightarrow 162\ KHSO_4 + 5\ Fe_2(SO_4)_3 + 122\ MnSO_4 + 60\ HNO_3 + 60\ CO_2 + 188\ H_2O$

Autre exemple :

$a\ Ca_{10}F_2(PO_4)_6 + b\ H_2SO_4 \rightarrow c\ Ca(H_2PO_4)_2 + d\ CaSO_4 + e\ HF$



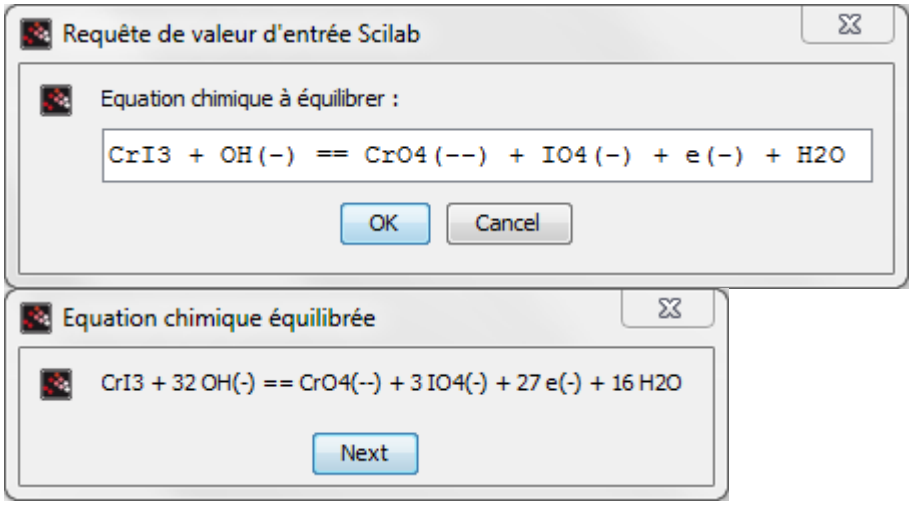


Donc :

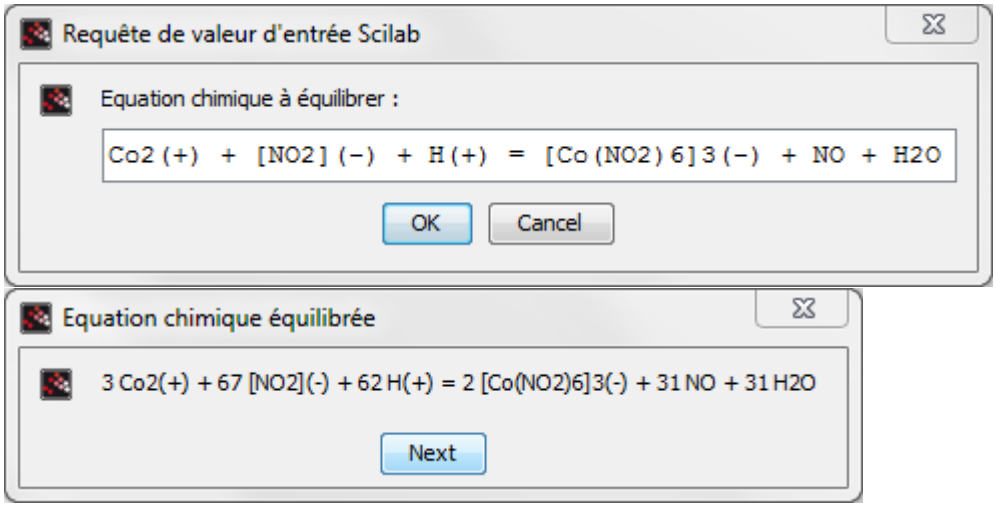
$Ca_{10}F_2(PO_4)_6 + 7\ H_2SO_4 \rightarrow 3\ Ca(H_2PO_4)_2 + 7\ CaSO_4 + 2\ HF$

Mathématique & Informatique

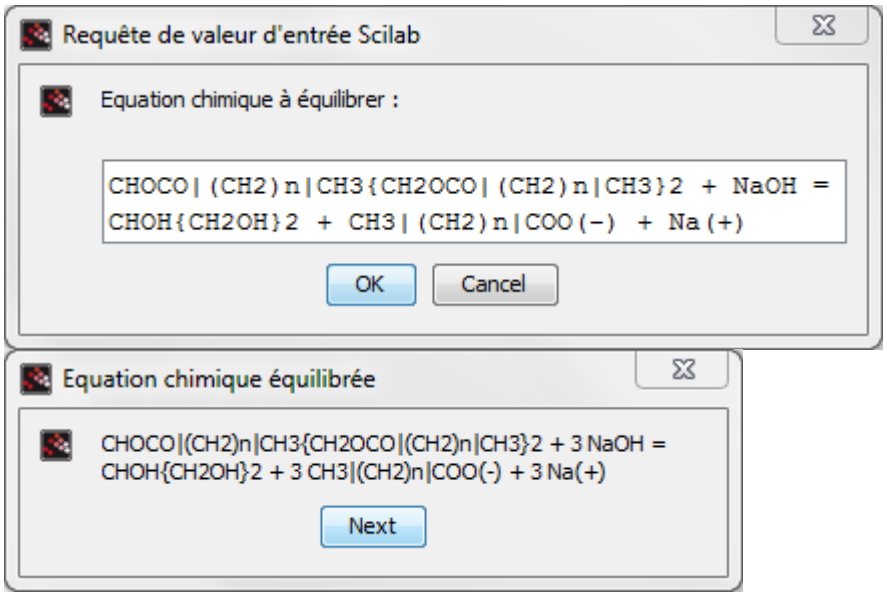
- $CrI_3 + 32\ OH^- \rightleftarrows CrO_4^{--} + 3\ IO_4^- + 27\ e^- + 16\ H_2O$



- $3\ Co_2^+ + 67\ [NO_2]^- + 62\ H^+ \rightarrow 2\ [Co(NO_2)_6]_3^- + 31\ NO + 31\ H_2O$



- $CHOCO(CH_2)_nCH_3\{CH_2OCO(CH_2)_nCH_3\}_2 + 3\ NaOH \rightarrow CHOH\{CH_2OH\}_2 + 3\ CH_3(CH_2)_nCOO^- + 3\ Na^+$



Interface LaTeX

Le logiciel LaTeX permet d'écrire les équations chimiques sous leur forme conventionnelle :

- chiffres de taille normale pour les coefficients stœchiométriques
- chiffres en indice pour définir l'atomicité des molécules
- chiffres en exposant pour indiquer les charges ioniques
- flèche pour séparer les deux membres de l'équation chimique

Par exemple, la chaîne de caractères "3SO4(2-)" sera convertie en "${3}SO_{4}^{2-}$" de façon à obtenir à l'écran $3SO_4^{2-}$.
La flèche droite est codée "\rightarrow". Ainsi, l'instruction *xstring(0,0,"${2}H_{2} + O_{2} \rightarrow {2}H_{2}O$")* ouvre une fenêtre Scilab et affiche l'équation $2H_2 + O_2 \rightarrow 2H_2O$.
Programme de stœchiométrie avec conversion au format LaTeX :

```
//-------------------------------------------------------------------------------------------
//           Calcul des coefficients stoechiométriques d'une équation chimique
//-------------------------------------------------------------------------------------------

//---------------------------------------
//      Loïc Vrillon (Août 2014)
//---------------------------------------

funcprot(0) ; lines(0) ;
exec("Scilab-Equations.sce",-1)

function STOECHIOMETRIE()
    // Ouverture de la fenêtre graphique
    global f a xstr ystr hstr fontId fontSz
    xstr = 0
    ystr = 0
    hstr = 0.5
    fontId = 2
    fontSz = 5
    X = getsystemmetrics("SM_CXSCREEN")
    Y = getsystemmetrics("SM_CYSCREEN")
    f = scf(1)
    f.figure_name = 'STOECHIOMETRIE'
    f.figure_position = [0,0]
    f.figure_size = [0.9*X,0.9*Y]
    toolbar(1,'off')
    a = gca()
    a.data_bounds = [0,-1;1,0]
    a.x_location = 'top'
    a.y_location = 'left'
    a.axes_visible = 'off'
    // Lecture des équations chimiques à équilibrer
    init = ''
    while %t
        resp = x_dialog('Equation(s) chimique(s) à équilibrer :',init)
        if resp == [] then, break, end
        init = resp
        RESP = tokens(strcat(resp,';'),';')
        for i = 1:size(RESP,1)
            EQ = RESP(i)
            next = stoechiometrie(EQ)
            if ~next then, break, end
        end
        if next then, init = '', end
    end
endfunction
```

```
// Conversion d'une équation chimique au format LaTeX
function strltx=LaTeX(EQ)
    strltx = ''
    current = ''
    lower = %f
    upper = %f
    symbol = %f
    comment = %f
    coefficient = %f
    colorname = 'red'
    L = list()
    EQ = strsplit(EQ)
    for i = 1:size(EQ,1)
        previous = current
        current = EQ(i)
        if coefficient then
            if isdigit(current) then
                L($+1) = current
            else
                coefficient = %f
                L($+1) = '}'
                select current
                case '|' then
                    symbol = %t
                    L($+1) = '{'
                case '{' then
                    L($+1) = '\{'
                case '}' then
                    L($+1) = '\}'
                else
                    L($+1) = current
                end
            end
            continue
        end
        if lower then
            if isdigit(current) then
                L($+1) = current
            else
                lower = %f
                L($+1) = '}'
                select current
                case '|' then
                    symbol = %t
                    L($+1) = '{'
                case '/' then
                    comment = %t
                    L($+1) = '_{('
                case '{' then
                    L($+1) = '\{'
                case '}' then
                    L($+1) = '\}'
                else
                    L($+1) = current
                end
            end
            continue
        end
```

```
if upper then
    if current == ')' then
        upper = %f
        L($+1) = '}'
    else
        L($+1) = current
    end
    continue
end
if symbol then
    if current == '|' then
        symbol = %f
        L($+1) = '}'
    else
        select current
        case '{' then
            L($+1) = '\{'
        case '}' then
            L($+1) = '\}'
        else
            L($+1) = current
        end
    end
    continue
end
if comment then
    if current == '/' then
        comment = %f
        L($+1) = ')}'
    else
        select current
        case '{' then
            L($+1) = '\{'
        case '}' then
            L($+1) = '\}'
        else
            L($+1) = current
        end
    end
    continue
end
if isdigit(current) then
    if regexp(previous,'/[A-Za-z\)\]\}\|]/') <> [] then
        lower = %t
        L($+1) = '_{'
        L($+1) = current
    elseif previous == '(' then
        upper = %t
        L($) = '^{'
        L($+1) = current
    else
        coefficient = %t
        if previous == '.' then
            L($+1) = '{'
        else
            L($+1) = '\textcolor{' + colorname + '}{'
        end
        L($+1) = current
    end
    continue
end
```

```
        if isletter(current) then
            L($+1) = current
            continue
        end
        if current == '|' then
            symbol = %t
            L($+1) = '{'
            continue
        end
        if current == '/' then
            comment = %t
            L($+1) = '_{('
            continue
        end
        if current == '+' then
            if previous == '(' then
                upper = %t
                L($) = '^{'
                L($+1) = current
            else
                L($+1) = '+'
            end
            continue
        end
        if current == '-' then
            if previous == '(' then
                upper = %t
                L($) = '^{'
                L($+1) = current
            else
                L($+1) = '(-)'
            end
            continue
        end
        if current == '=' then
            if previous == '=' then
                L($) = '\rightleftarrows'
            else
                L($+1) = '\rightarrow'
            end
            continue
        end
        select current
        case '{' then
            L($+1) = '\{'
        case '}' then
            L($+1) = '\}'
        else
            L($+1) = current
        end
    end
    if lower then
        lower = %f
        L($+1) = '}'
    end
    for i = 1:length(L)
        strltx = strltx + L(i)
    end
    strltx = "$" + strltx + "$"
endfunction
```

Mathématique & Informatique

```
function next=stoechiometrie(EQ)
    global f a xstr ystr hstr fontId fontSz
    // Syntaxe d'une équation chimique
    electron = 'e'
    atome = ...
    '(Ac|Ag|Al|Am|Ar|As|At|Au|B|Ba|Be|Bh|Bi|Bk|Br|C|Ca|...
    Cd|Ce|Cf|Cl|Cm|Co|Cr|Cs|Cu|Db|Ds|Dy|Er|Es|Eu|F|Fe|...
    Fm|Fr|Ga|Gd|Ge|H|He|Hf|Hg|Ho|Hs|I|In|Ir|K|Kr|La|Li|...
    Lr|Lu|Md|Mg|Mn|Mo|Mt|N|Na|Nb|Nd|Ne|Ni|No|Np|O|Os|P|...
    Pa|Pb|Pd|Pm|Po|Pr|Pt|Pu|Ra|Rb|Re|Rf|Rh|Rn|Ru|S|Sb|Sc|...
    Se|Sg|Si|Sm|Sn|Sr|Ta|Tb|Tc|Te|Th|Ti|Tl|Tm|U|Uub|Uuh|...
    Uuo|Uup|Uuq|Uus|Uut|Uuu|V|W|Xe|Y|Yb|Zn|Zr)'
    symbole = '\|[^\|\/\\\+\=]+\|'
    separateur = '[\(\)\[\]\{\}]'
    nombre = '[1-9]\d*'
    hydrate = '\.([1-9]\d*)?H2O'
    commentaire = '\/[^\|\/\\\+\=]+\/'
    charge = '\(([1-9]\d*(\+|\-)|\+{1,3}|\-{1,3})\)'
    moins = '\((-|1-)\)'
    corps = '((' + atome + '|' + symbole + '|' + separateur + ')(' + nombre + ')?)+'
    formule = corps + '(' + hydrate + ')?(' + charge + ')?(' + commentaire + ')?'
    terme = '(' + formule + '|' + electron + '(' + moins + ')?)'
    membre = terme + '(\s*\+\s*' + terme + ')*'
    equation = '/^\s*' + membre + '\s*(=|==)\s*' + membre + '\s*$/'
    // Extraction des lettres
    function v=alphabetical(s)
        v = ''
        for i = 1:length(s)
            c = part(s,i)
            if isletter(c) then
                v = v + c
            else
                v = v + ' '
            end
        end
        v = tokens(v,' ')
    endfunction
    // Extraction des chiffres
    function v=numerical(s)
        v = ''
        for i = 1:length(s)
            c = part(s,i)
            if isdigit(c) then
                v = v + c
            else
                v = v + ' '
            end
        end
        v = evstr(tokens(v,' '))
    endfunction
```

```
// Remplacement des crochets et des accolades par des parenthèses
function f=bracket(F)
    f = ''
    stack = list()
    for i = 1:length(F)
        c = part(F,i)
        if c == '(' | c == '[' | c == '{' then
            stack($+1) = c
        end
        if c == ')' | c == ']' | c == '}' then
            if length(stack) == 0 then
                return
            end
            s = stack($)
            stack($) = null()
            if (c == ')' & s <> '(') | (c == ']' & s <> '[') | (c == '}' & s <> '{') then
                return
            end
        end
    end
    if length(stack) == 0 then
        f = strsubst(F,'[','(')
        f = strsubst(f,']',')')
        f = strsubst(f,'{','(')
        f = strsubst(f,'}',')')
    end
endfunction
```

```
//Formule brute
function f=conversion(F)
    // Vérifier la formule F
    check1 = regexp(F,'/[^A-Za-z0-9)]\d/') <> []
    check2 = regexp(F,'/\(\)/') <> []
    if check1 | check2 then
        f = ''
        return
    end
    // Introduire les opérateurs + et *
    S = ''
    s = ''
    for i = 1:length(F)
        c = part(F,i)
        if regexp(c,'/[A-Z(]/') <> [] then
            S = S + '+'
        end
        if isdigit(c) & ~isdigit(s) then
            S = S + '*'
        end
        S = S + c
        s = c
    end
    // Rechercher les atomes différents
    A = ''
    for i = 1:length(S)
        c = part(S,i)
        if isletter(c) then
            A = A + c
        else
            A = A + ' '
        end
    end
    A = tokens(A)
    A = unique(A)
    A = gsort(A)
    // Construire la formule f
    f = ''
    for i = 1:size(A,1)
        N = S
        for j = 1:size(A,1)
            if i == j then
                N = strsubst(N,A(j),'1')
            else
                N = strsubst(N,A(j),'0')
            end
        end
        err = execstr('N = evstr(N)','errcatch')
        if err <> 0 then
            f = ''
            return
        end
        f = A(i) + string(N) + f
    end
endfunction
```

```
// Vérification de l'équation
check = regexp(EQ,equation)
if check == [] then
    messagebox('Syntaxe incorrecte.','Erreur','error','Correction','modal')
    next = %f
    return
end
// Mise en forme de l'équation
EQ = strsubst(EQ,' ','')
sep = ' '
s = 0
while s <> []
    [s,e,S] = regexp(EQ,'/(\w|\)|\]|\}|\||\/)\+[^)]/','o')
    if s <> [] then
        R = strsubst(S,'+',sep)
        EQ = strsubst(EQ,S,R)
    end
end
dbl = strstr(EQ,'==') <> ''
EQ = strsubst(EQ,'==','=')
E = strsubst(EQ,'=',sep)
F = tokens(E,sep)
n = size(F,1)
X = 'X'
s = 0
while s <> []
    [s,e,S] = regexp(EQ,'/' + symbole + '/','o')
    if s <> [] then
        X = X + 'x'
        EQ = strsubst(EQ,S,X)
    end
end
s = 0
while s <> []
    [s,e,S] = regexp(EQ,'/' + hydrate + '/','o')
    if s <> [] then
        k = strsubst(S,'.','')
        k = strsubst(k,'H2O','')
        R = '(H2O)' + k
        EQ = strsubst(EQ,S,R)
    end
end
s = 0
while s <> []
    [s,e,S] = regexp(EQ,'/' + commentaire + '/','o')
    if s <> [] then
        EQ = strsubst(EQ,S,'')
    end
end
EQ = strsubst(EQ,'(+)','(1+)')
EQ = strsubst(EQ,'(-)','(1-)')
EQ = strsubst(EQ,'(++)','(2+)')
EQ = strsubst(EQ,'(--)','(2-)')
EQ = strsubst(EQ,'(+++)','(3+)')
EQ = strsubst(EQ,'(---)','(3-)')
E = tokens(EQ,'=')
F1 = tokens(E(1),sep)
F2 = tokens(E(2),sep)
n1 = size(F1,1)
n2 = size(F2,1)
```

```
// Réactifs
C1 = []
for i = 1:n1
    if F1(i) == electron then
        F1(i) = electron + '(1-)'
    end
    [s,e,S] = regexp(F1(i),'/' + charge + '$/')
    if s <> [] then
        F1(i) = strsubst(F1(i),'/' + charge + '$/','','r')
        c = part(S,length(S)-1)
        S = c + strsubst(S,c,'')
        C1 = [C1,evstr(S)]
    else
        C1 = [C1,0]
    end
    if F1(i) <> electron then
        F1(i) = bracket(F1(i))
        F1(i) = conversion(F1(i))
    else
        F1(i) = electron + '1'
    end
    if F1(i) == '' then
        messagebox('Syntaxe incorrecte.','Erreur','error','Correction','modal')
        next = %f
        return
    end
end
// Produits
C2 = []
for i = 1:n2
    if F2(i) == electron then
        F2(i) = electron + '(1-)'
    end
    [s,e,S] = regexp(F2(i),'/' + charge + '$/')
    if s <> [] then
        F2(i) = strsubst(F2(i),'/' + charge + '$/','','r')
        c = part(S,length(S)-1)
        S = c + strsubst(S,c,'')
        C2 = [C2,evstr(S)]
    else
        C2 = [C2,0]
    end
    if F2(i) <> electron then
        F2(i) = bracket(F2(i))
        F2(i) = conversion(F2(i))
    else
        F2(i) = electron + '1'
    end
    if F2(i) == '' then
        messagebox('Syntaxe incorrecte.','Erreur','error','Correction','modal')
        next = %f
        return
    end
end
```

```
// Atomes et Charges
A1 = setdiff(unique(alphabetical(strcat(F1))),electron)
A2 = setdiff(unique(alphabetical(strcat(F2))),electron)
if ~isequal(A1,A2) then
    messagebox('Non conservation des éléments.','Erreur','error','Correction','modal')
    next = %f
    return
end
ATOMS = A1
m = size(ATOMS,1)
C = [C1,C2]
// Création du tableau de Lavoisier
T = zeros(m,n)
for j = 1:n
    if j <= n1 then
        formula = F1(j)
    else
        formula = F2(j-n1)
    end
    if formula <> electron then
        atoms = alphabetical(formula)
        numbers = numerical(formula)
        for k = 1:size(atoms,1)
            i = find(ATOMS == atoms(k))
            T(i,j) = T(i,j) + numbers(k)
        end
    end
end
// Résolution du système linéaire homogène
A = list()
for i = 1:m
    for j = 1:n
        if j <= n1 then
            A($+1) = FRACTION(-T(i,j),1)
        else
            A($+1) = FRACTION(T(i,j),1)
        end
    end
end
if find(C <> 0) <> [] then
    for j = 1:n
        if j <= n1 then
            A($+1) = FRACTION(-C(j),1)
        else
            A($+1) = FRACTION(C(j),1)
        end
    end
    A($+1) = m + 1
    A($+1) = n
    B = M_ZERO(m+1,1)
else
    A($+1) = m
    A($+1) = n
    B = M_ZERO(m,1)
end
X0 = M_UN(n,1)
X = LINEAIRE(A,B,X0)
```

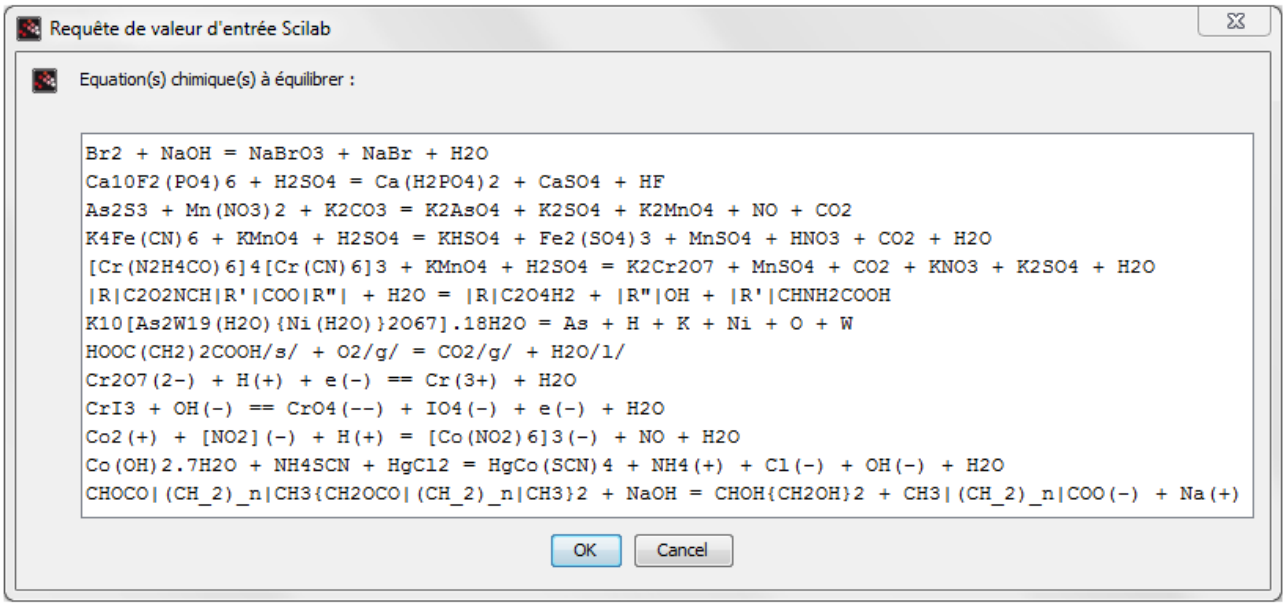
```
// Calcul des coefficients stoechiométriques entiers
L = list()
if ~M_VIDE(X) then
    L1 = list()
    L2 = list()
    for i = 1:n
        L1($+1) = NUMERATEUR(X(i))
        L2($+1) = DENOMINATEUR(X(i))
    end
    [pgcdl1,ppcml1] = PGCDPPCM(L1)
    [pgcdl2,ppcml2] = PGCDPPCM(L2)
    for i = 1:n
        [q1,r1] = DIVISION(NUMERATEUR(X(i)),pgcdl1)
        [q2,r2] = DIVISION(ppcml2,DENOMINATEUR(X(i)))
        L($+1) = MULTIPLICATION(q1,q2)
    end
end
if length(L) == 0 then
    messagebox('Solution non trouvée !','Erreur','error','Next','modal')
    next = %t
    return
end
// Mise en forme du résultat
S = list()
for i = 1:length(L)
    sgn = SIGNE(L(i))
    val = VALEUR(L(i))
    if sgn < 0 then
        s = '-'
    else
        s = ''
    end
    S($+1) = s + strcat(string(val))
end
```

```
    // Ecriture de l'équation chimique équilibrée
    E1 = ''
    for i = 1:n1
        if i <> 1 then
            E1 = E1 + ' + '
        end
        if S(i) == '1' then
            E1 = E1 + F(i)
        else
            E1 = E1 + S(i) + F(i)
        end
    end
    E2 = ''
    for i = 1:n2
        if i <> 1 then
            E2 = E2 + ' + '
        end
        if S(n1+i) == '1' then
            E2 = E2 + F(n1+i)
        else
            E2 = E2 + S(n1+i) + F(n1+i)
        end
    end
    if dbl then
        EQ = E1 + ' == ' + E2
    else
        EQ = E1 + ' = ' + E2
    end
    strltx = LaTeX(EQ)
    rect = xstringl(0,0,strltx,fontId,fontSz)
    h = rect(4)
    ystr = ystr - h
    xstring(xstr,ystr,strltx)
    e = gce()
    e.font_style = fontId
    e.font_size = fontSz
    ystr = ystr - h*hstr
    next = %t
endfunction

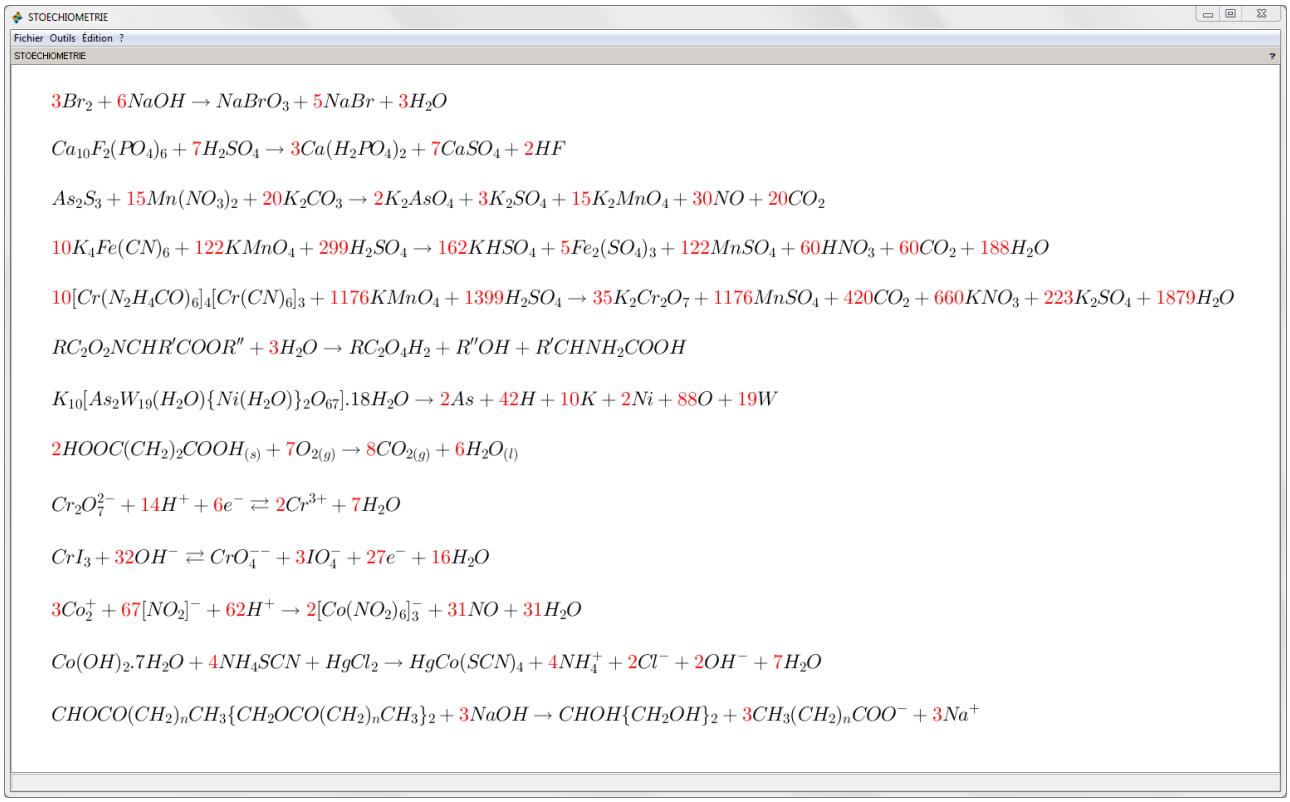
STOECHIOMETRIE()
```

Mathématique & Informatique

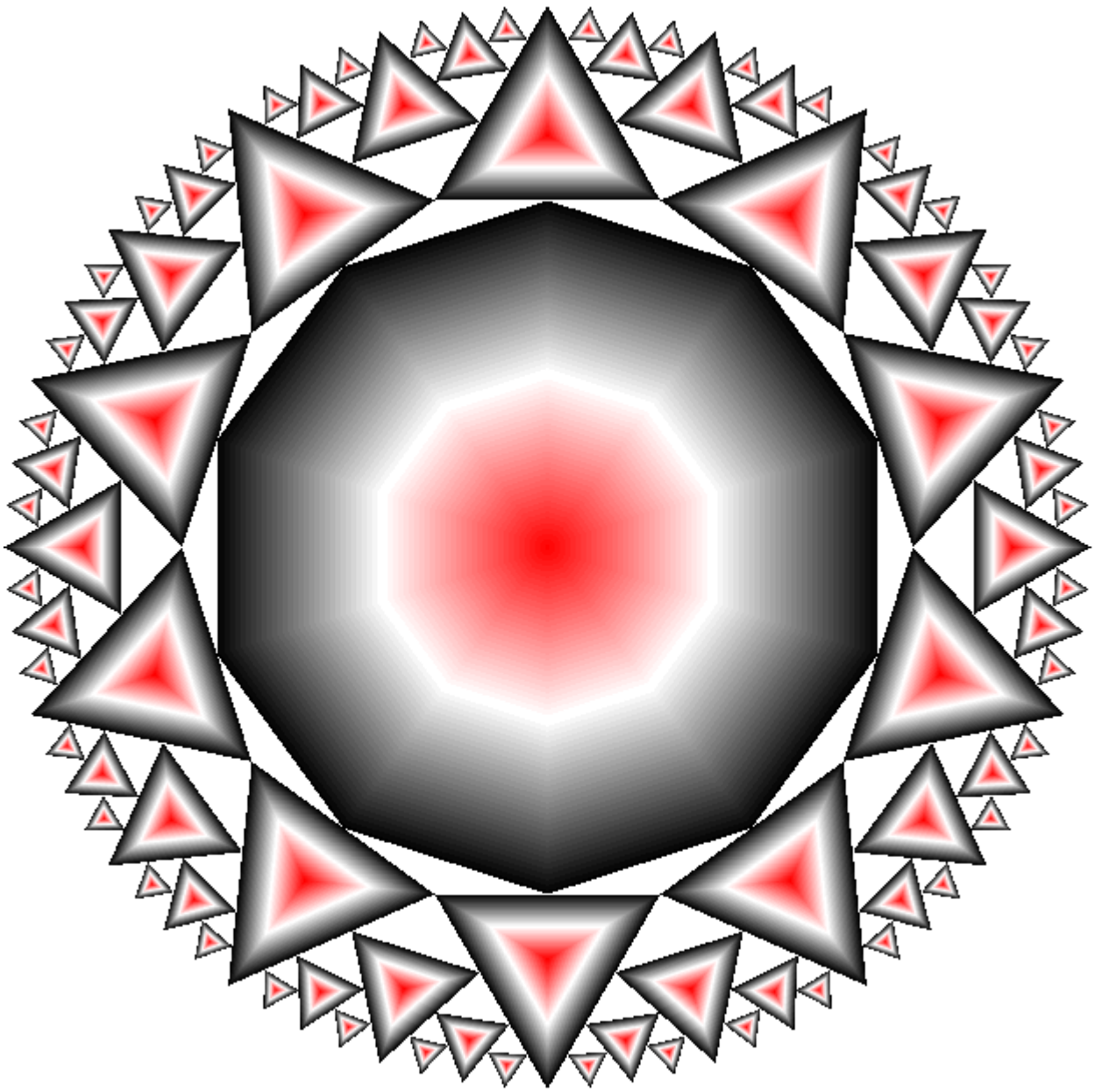
En entrée :



En sortie :



Mathématique & Informatique



Mathématique & Informatique

ATOMISTIQUE

L'équation différentielle de **Schrödinger** ($H\psi = E\psi$) conduit à introduire 4 nombres quantiques n, l, m, s pour caractériser un électron d'un atome :

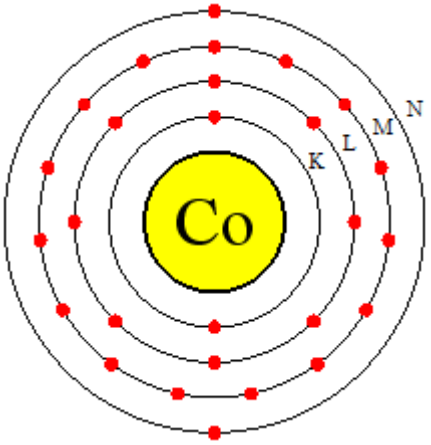
1) Le nombre quantique principal **n** définit le numéro de la couche électronique (niveau d'énergie croissant)
n = 1, 2, 3, 4, 5, 6, 7 correspond aux couches K, L, M, N, O, P, Q

2) Le nombre quantique secondaire ou azimutal **l** définit la géométrie de l'orbitale (sphérique, piriforme, torique)
l = 0, 1, 2, 3 correspond aux orbitales s, p, d, f

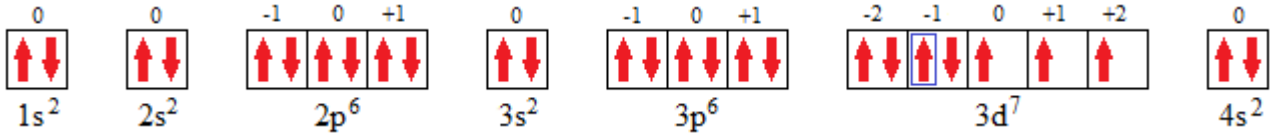
3) Le nombre quantique tertiaire ou magnétique **m** définit l'orientation dans l'espace de l'orbitale (Ox, Oy, Oz)
$-l \leq m \leq +l$ soit m = 0, ±1, ±2, … , ±l

4) Le nombre quantique de spin **s** propre à l'électron (moment cinétique intrinsèque évoquant une rotation)
$s = \pm \frac{1}{2}$

Une orbitale est définie par un triplet (n, l, m) et un électron est défini par un quadruplet (n, l, m, s). Comme s ne peut prendre que 2 valeurs, une orbitale ne contient jamais plus de 2 électrons. Dans un atome, la couche électronique n possède au plus $2n^2$ électrons et la dernière couche au plus 8 (sauf pour le Palladium Pd). L'ordre de remplissage des sous-couches électroniques l se fait selon les énergies croissantes (règle de **Klechkowski** avec des exceptions : Cr, Cu, Ag, Au, …). Les électrons occupent un maximum d'orbitales avant de les compléter par un deuxième électron de spin opposé (règle de **Hund**).
Prenons, par exemple, l'atome de Cobalt de symbole Co et de numéro atomique 27. Les couches K, L, M, N contiennent respectivement 2, 8, 15, 2 électrons. Dessinons de manière schématique le nuage des 27 électrons autour du noyau :



La configuration théorique des orbitales s'écrit : $1s^2\ 2s^2\ 2p^6\ 3s^2\ 3p^6\ 3d^7\ 4s^2$
La structure électronique du Cobalt, sous la forme de cases quantiques ☐ représentant les orbitales, est :



De la configuration partielle $3s^2$, $3p^6$, $3d^7$ on déduit que la couche n = 3 (M) est composée de 2 + 6 + 7 = 15 électrons.
L'électron encadré en bleu sur le schéma est caractérisé par le quadruplet quantique $(n, l, m, s) = (3, 2, -1, +\frac{1}{2})$

On dispose d'un fichier *ATOMES* contenant les caractéristiques des atomes connus, classés par numéro atomique croissant de l'Hydrogène (numéro atomique 1) au Copernicium (numéro atomique 112).
Chaque ligne du fichier est organisée de la manière suivante :

Numéro_atomique ; Nom ; Symbole ; Couches_électroniques ; Orbitales_ électroniques

Ainsi, à la 27e ligne du fichier *ATOMES* on trouve les caractéristiques de l'atome de Cobalt :

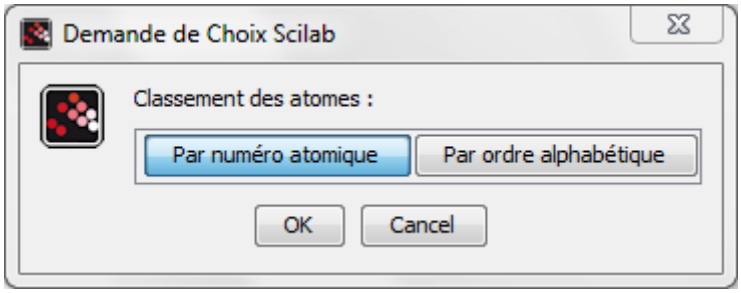
27 ; Cobalt ; Co ; K2 L8 M15 N2 ; 1s2 2s2 2p6 3s2 3p6 3d7 4s2

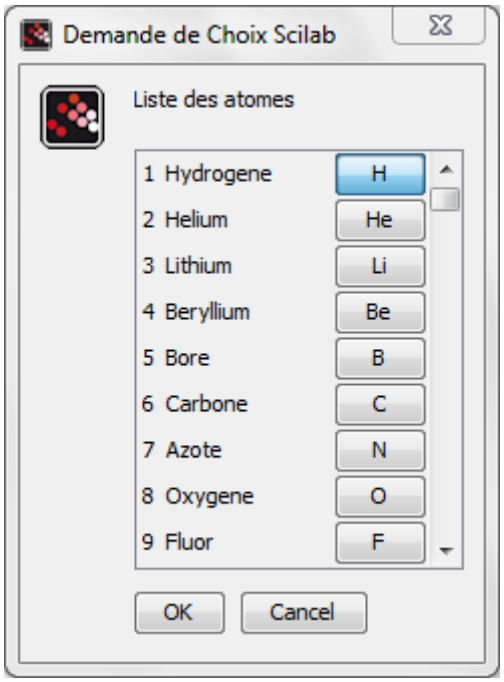
Ecrivons un programme qui permet de sélectionner une liste d'atomes, puis qui indique, pour chacun d'eux, le nombre d'électrons présents sur ses couches électroniques et la configuration de ses différentes orbitales.

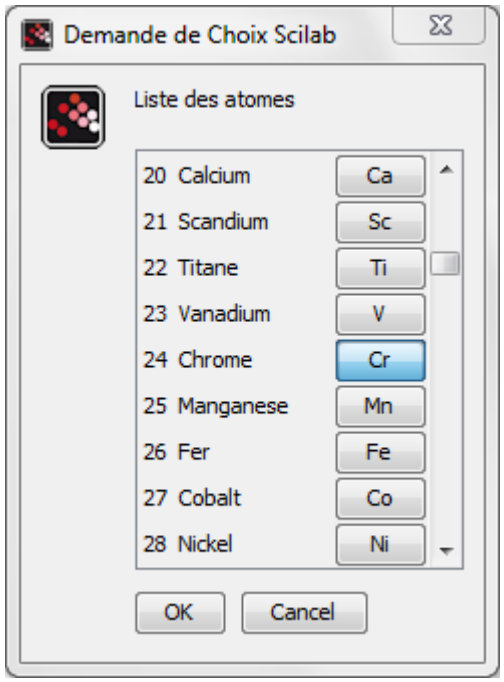
```
A = mgetl('ATOMES') ;
n = size(A,1) ;
A1 = emptystr(n,1) ;
for i = 1:n
    s = strsubst(A(i),' ; ',';') ;
    S = tokens(s,';') ;
    A1(i) = strcat([S(2),S(3),S(1),S(4),S(5)],';') ;
end
A2 = gsort(A1,'g','i') ;
name1 = list() ;
name2 = list() ;
for i = 1:n
    S1 = tokens(A1(i),';') ;
    S2 = tokens(A2(i),';') ;
    name1($+1) = list(' ' + S1(3) + '   ' + S1(1),0,S1(2)) ;
    name2($+1) = list('   ' + S2(1),0,S2(2)) ;
end
name1($+1) = list('   TOUS LES ATOMES',0,'*') ;
name2($+1) = list('   TOUS LES ATOMES',0,'*') ;
while %t
    choice = list('',1,['Par numéro atomique','Par ordre alphabétique']) ;
    choice1 = x_choices('   Classement des atomes :',list(choice)) ;
    if choice1 == [] then, break, end
    select choice1
        case 1 then, choice2 = x_choices('Liste des atomes',name1) ;
        case 2 then, choice2 = x_choices('Liste des atomes',name2) ;
    end
    if choice2 == [] then, break, end
    I = find(choice2 == 1) ;
    if I == [] then, continue, end
    if find(I == n+1) <> [] then, I = [1:n] ; end
    mess = '' ;
    for i = I
        select choice1
            case 1 then, S = tokens(A1(i),';') ;
            case 2 then, S = tokens(A2(i),';') ;
        end
        text1 = S(1) + ' (' + S(2) + ')' ;
        text2 = '----------------------------' ;
        text3 = 'Numéro atomique : ' + S(3) ;
        text4 = 'Couches électroniques : ' + S(4) ;
        text5 = 'Orbitales électroniques : ' + S(5) ;
        mess = [mess ; '' ; text1 ; text2 ; text3 ; text4 ; text5] ;
    end
    messagebox(mess,'Atomes','','modal') ;
end
```

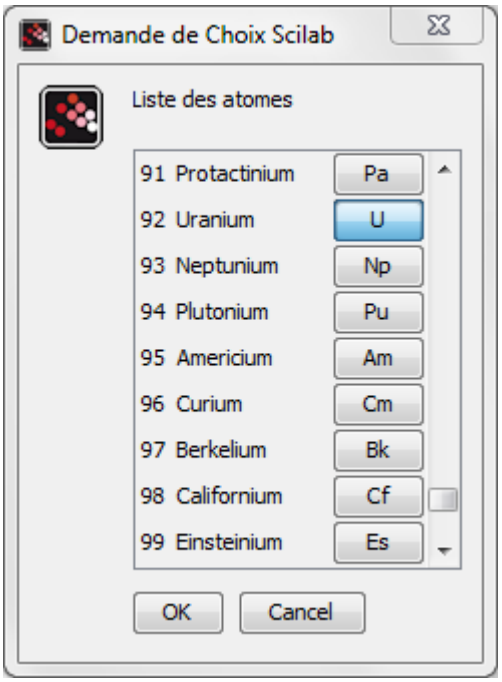
Mathématique & Informatique

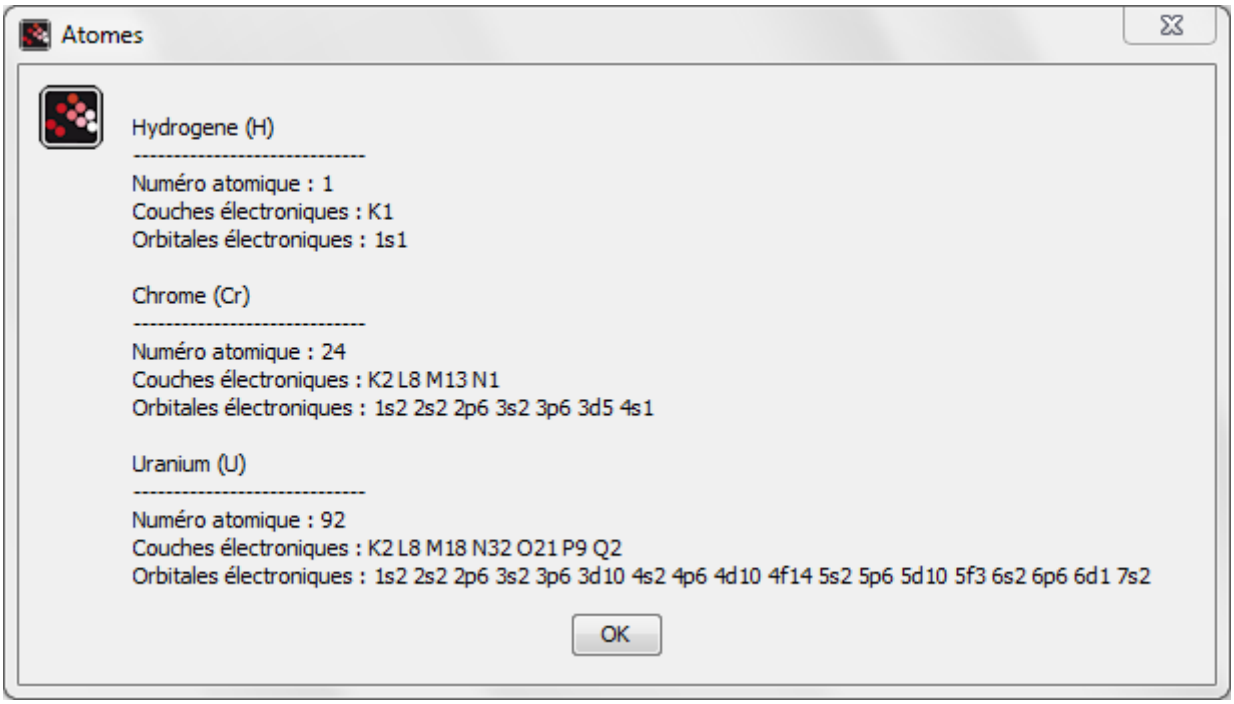
Exemple











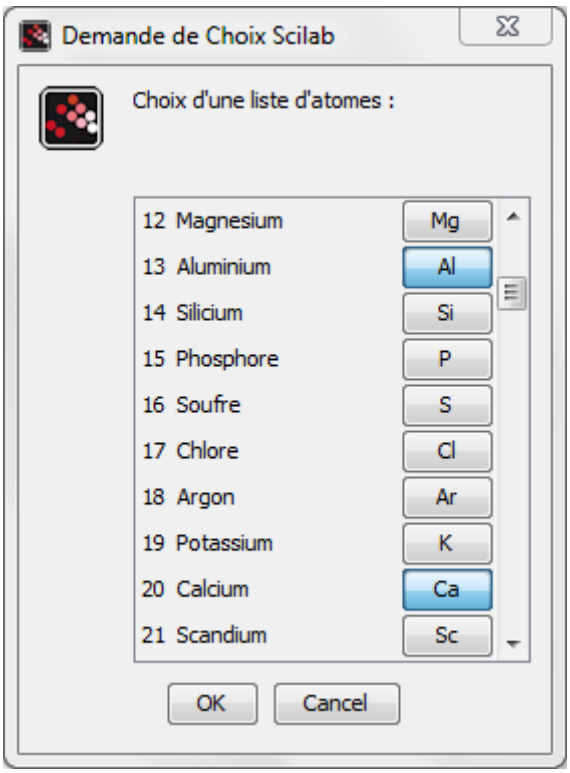
Mathématique & Informatique

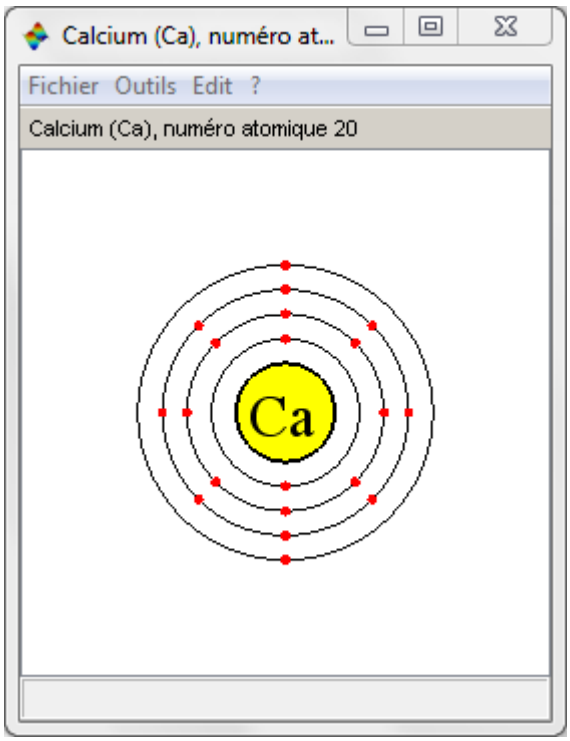
Le programme suivant donne une représentation plane schématique d'un atome avec ses différentes couches électroniques (voir le modèle planétaire de **Rutherford** et le modèle quantifié de **Bohr** ; ces deux modèles déterministes sont antérieurs au modèle probabiliste actuel de la mécanique quantique dont on peut déduire le principe d'incertitude d'**Heisenberg** : il est impossible de connaître à la fois avec précision la position et la vitesse de l'électron à l'instant t. Ce <<flou>> sur la localisation de l'électron conduit à abandonner la notion d'orbite de la mécanique classique et à la remplacer par la notion d'orbitale de la mécanique quantique).

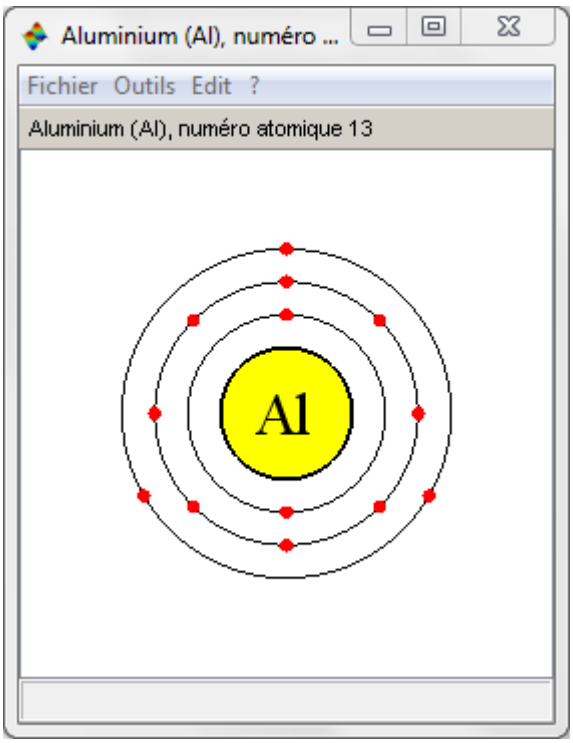
```
// Lecture du fichier des atomes
A = mgetl('ATOMES') ;
n = size(A,1) ;
name = list() ;
for i = 1:n
    s = strsubst(A(i),' ; ',';') ;
    S = tokens(s,';') ;
    name($+1) = list(' ' + S(1) + '   ' + S(2),0,S(3)) ;
end
// Choix d'une liste d'atomes
choice = x_choices('   Choix d"une liste d"atomes :',name) ;
if choice == [] then, abort, end
I = find(choice == 1) ;
if I == [] then, abort, end
// Dessin des atomes sélectionnés
R0 = 10 ; r0 = 1 ; r = 5 ;
nf = 0 ;
for i = I
    s = strsubst(A(i),' ; ',';') ;
    S = tokens(s,';') ;
    C = tokens(S(4)) ;
    nc = size(C,1) ;
    nf = nf + 1 ;
    f = scf(nf) ; toolbar(nf,'off') ;
    f.figure_position = [0,0] ; f.figure_size = [500,500] ;
    f.figure_name = S(2) + ' (' + S(3) + '), numéro atomique ' + S(1) ;
    R = R0 + nc*r + r0 ;
    plot2d(%inf, %inf, frameflag = 3, rect = [-R,-R,+R,+R], axesflag = 0) ;
    drawlater() ;
    // Noyau atomique
    xfarc(-R0,+R0,2*R0,2*R0,0,64*360) ;
    e = gce() ; e.background = 7 ;
    xarc(-R0,+R0,2*R0,2*R0,0,64*360) ;
    e = gce() ; e.foreground = 1 ; e.thickness = 2 ;
    xstringb(-R0,-R0,S(3),2*R0,2*R0) ;
    e = gce() ; e.font_size = 6 ; e.font_style = 2 ;
    // Couches électroniques
    for j = 1:nc
        R = R0 + j*r ;
        xarc(-R,+R,2*R,2*R,0,64*360) ;
        e = gce() ; e.foreground = 1 ;
        ne = part(C(j),2:length(C(j))) ;
        ne = evstr(ne) ;
        for k = 1:ne
            theta = %pi/2 + (k-1)*(2*%pi/ne) ;
            x = R*cos(theta) ; y = R*sin(theta) ;
            xfarc(x-r0,y+r0,2*r0,2*r0,0,64*360) ;
            e = gce() ; e.background = 5 ;
        end
    end
    drawnow() ;
end
```

Mathématique & Informatique

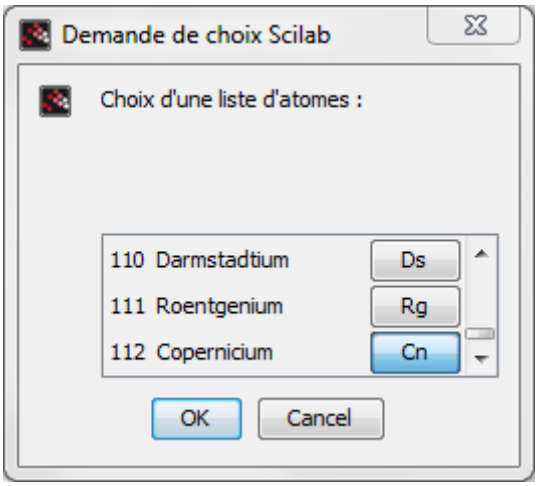
Exemple

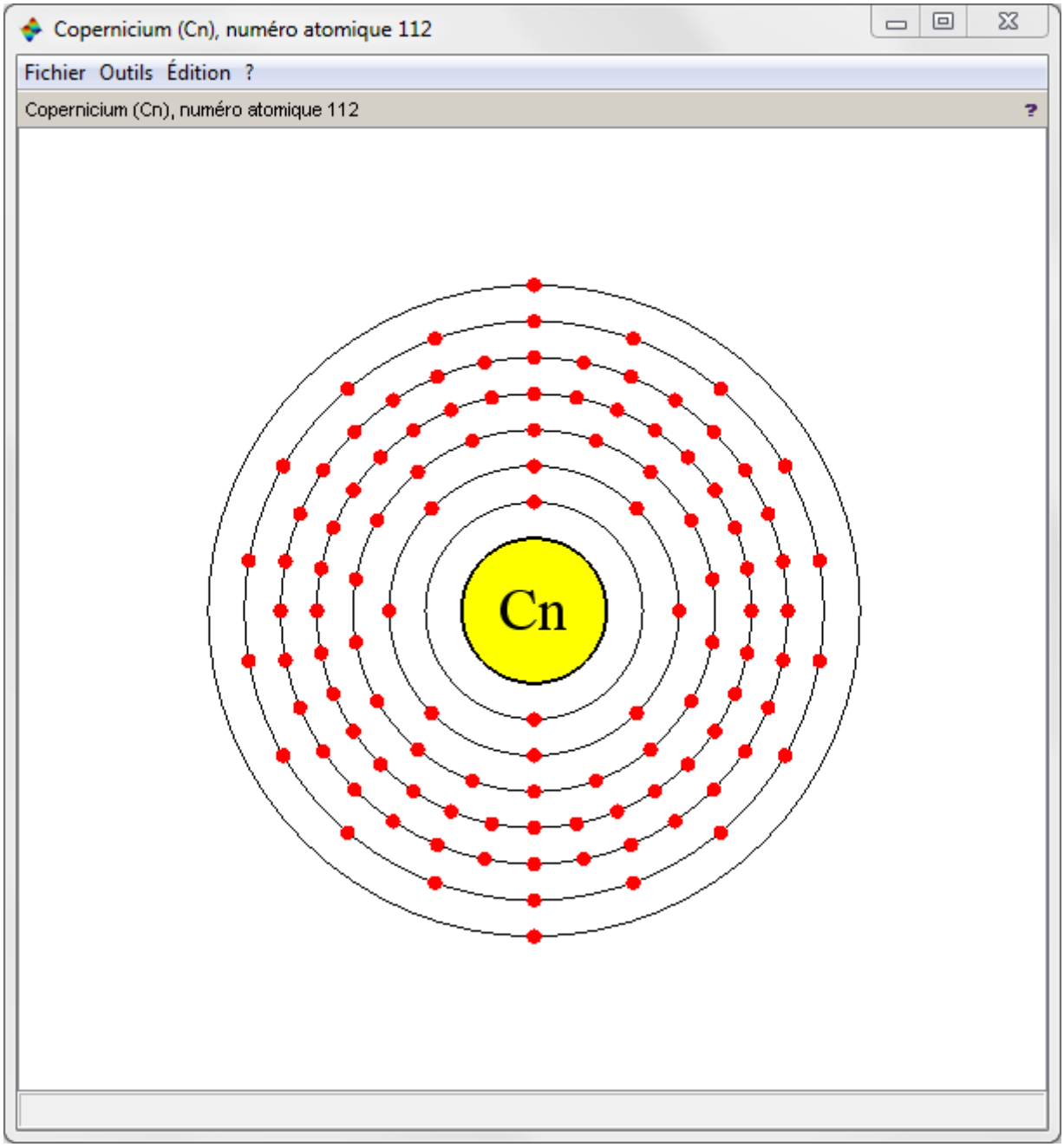






Mathématique & Informatique





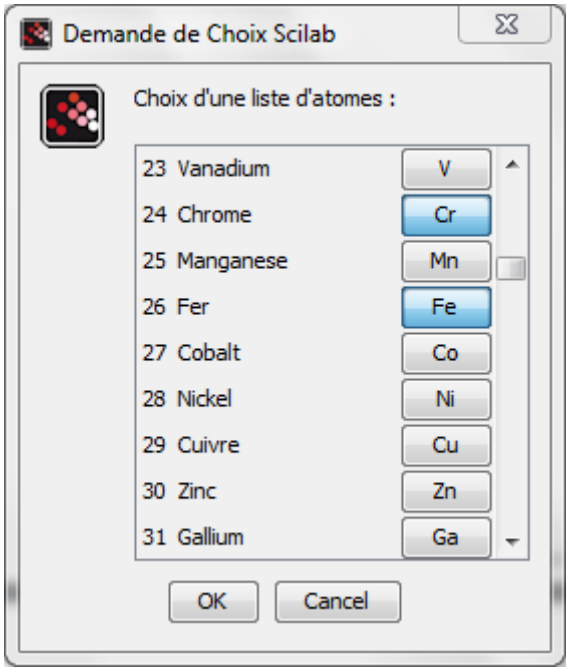
Mathématique & Informatique

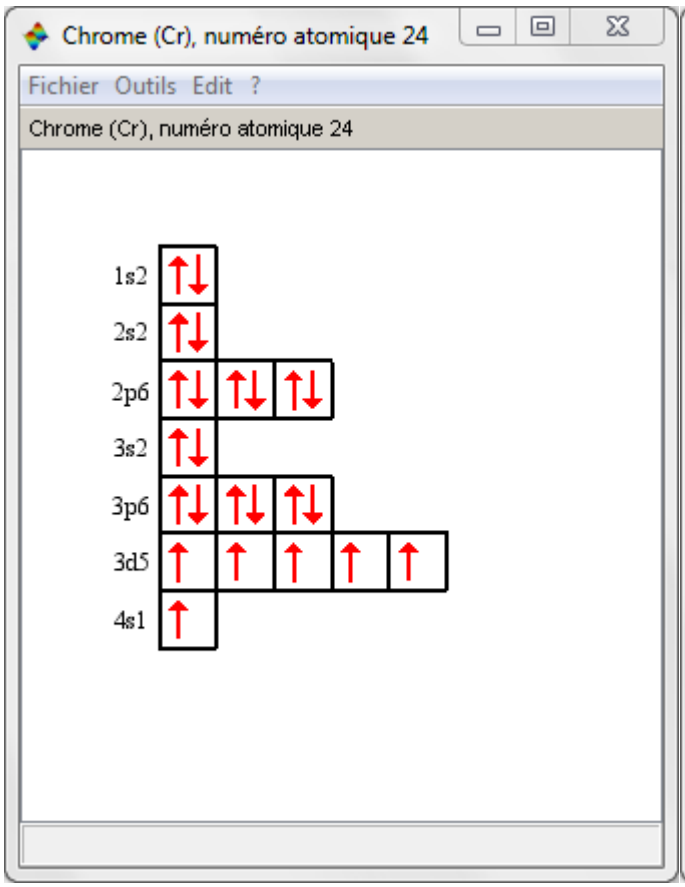
Ecrivons un programme donnant la structure électronique d'un d'atome, sous la forme de cases quantiques représentant les orbitales. Une flèche ↑ symbolisera un électron de spin + $\frac{1}{2}$ et une flèche ↓ symbolisera un électron de spin - $\frac{1}{2}$.
Une liste déroulante permettra de choisir un ou plusieurs atomes. Dans cette liste, les atomes seront ordonnés par numéro atomique croissant (le numéro atomique Z d'un atome est le nombre de protons contenus dans son noyau, c'est également le nombre d'électrons de l'atome s'il n'est pas ionisé).

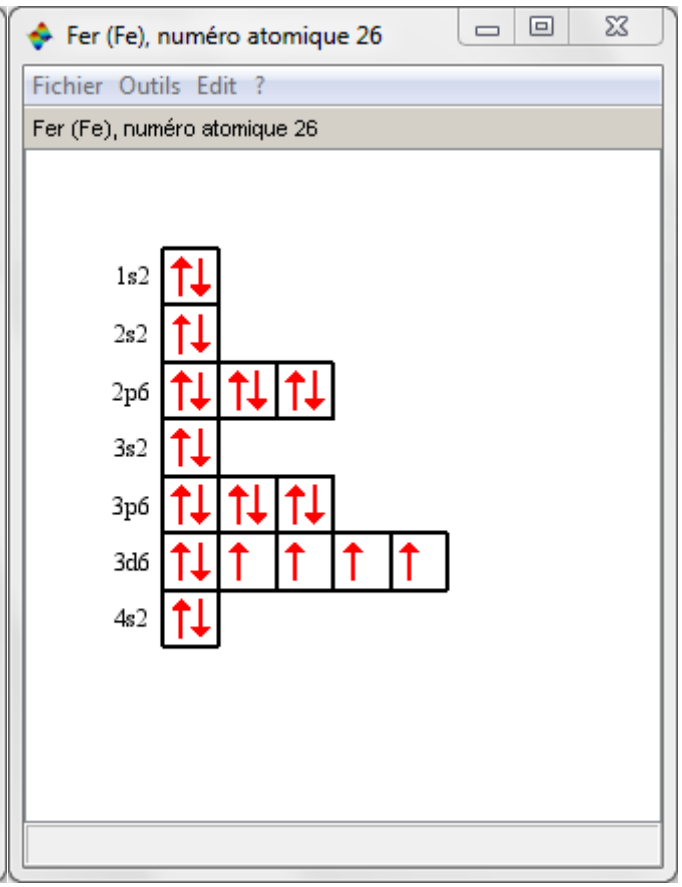
```
// Lecture du fichier des atomes
A = mgetl('ATOMES') ;
n = size(A,1) ;
name = list() ;
for i = 1:n
    s = strsubst(A(i),' ; ',';') ;
    S = tokens(s,';') ;
    name($+1) = list(' ' + S(1) + '   ' + S(2),0,S(3)) ;
end
// Choix d'une liste d'atomes
choice = x_choices('   Choix d"une liste d"atomes :',name) ;
if choice == [] then, abort, end
I = find(choice == 1) ;
if I == [] then, abort, end
// Dessin des électrons
function spin1(x, y)
    xsegs([x+w/3,x+w/3],[y+w/6,y+2*w/3])
    e = gce() ; e.segs_color = 5 ; e.thickness = 2
    xfpoly([x+w/3,x+w/6,x+w/2],[y+5*w/6,y+2*w/3,y+2*w/3])
    e = gce() ; e.background = 5 ; e.foreground = 5
endfunction
function spin2(x, y)
    xsegs([x+2*w/3,x+2*w/3],[y+5*w/6,y+w/3])
    e = gce() ; e.segs_color = 5 ; e.thickness = 2
    xfpoly([x+2*w/3,x+5*w/6,x+w/2],[y+w/6,y+w/3,y+w/3])
    e = gce() ; e.background = 5 ; e.foreground = 5
endfunction
// Dessin des cases quantiques
function orbitale(x, y, l)
    for i = 1:2*l+1
        x = x + w
        xrect(x,y+w,w,w)
        e = gce() ; e.foreground = 1 ; e.thickness = 2
    end
endfunction
```

```
// Structure électronique des atomes sélectionnés
nf = 0 ;
for i = I
    x0 = 0 ; y0 = 0 ;
    w = 3 ;
    s = strsubst(A(i),' ; ',';') ;
    S = tokens(s,';') ;
    ORB = tokens(S(5)) ;
    norb = size(ORB,1) ;
    nf = nf + 1 ;
    f = scf(nf) ; toolbar(nf,'off') ;
    f.figure_position = [0,0] ; f.figure_size = [500,500] ;
    f.figure_name = S(2) + ' (' + S(3) + '), numéro atomique ' + S(1) ;
    plot2d(%inf, %inf, frameflag = 3, rect = [x0,y0-norb*w,x0+8*w,y0], axesflag = 0)
    drawlater() ;
    for j = 1:norb
        y = y0 - j*w ;
        orb = ORB(j) ;
        xstringb(x0,y,orb,w,w)
        e = gce() ; e.font_size = 2 ; e.font_style = 2 ;
        l = part(orb,2) ;
        select l
            case 's' then, l = 0 ;
            case 'p' then, l = 1 ;
            case 'd' then, l = 2 ;
            case 'f' then, l = 3 ;
        end
        orbitale(x0,y,l)
        ne = part(orb,3:length(orb)) ;
        ne = evstr(ne) ;
        for k = 1:ne
            if k == 1 | k == 2*l+2 then
                x = x0 ;
            end
            x = x + w ;
            if k < 2*l+2 then
                spin1(x,y)
            else
                spin2(x,y)
            end
        end
    end
    drawnow() ;
end
```

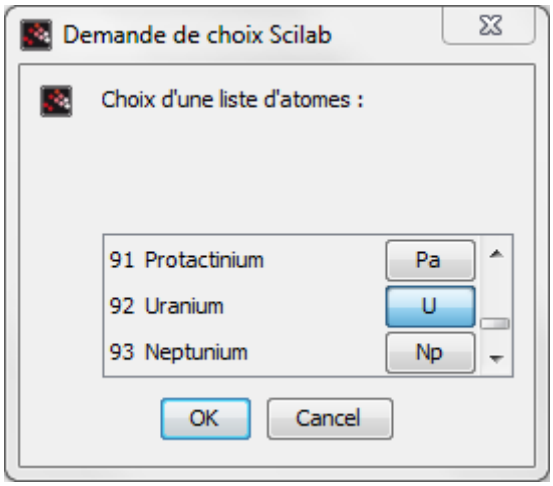
Mathématique & Informatique

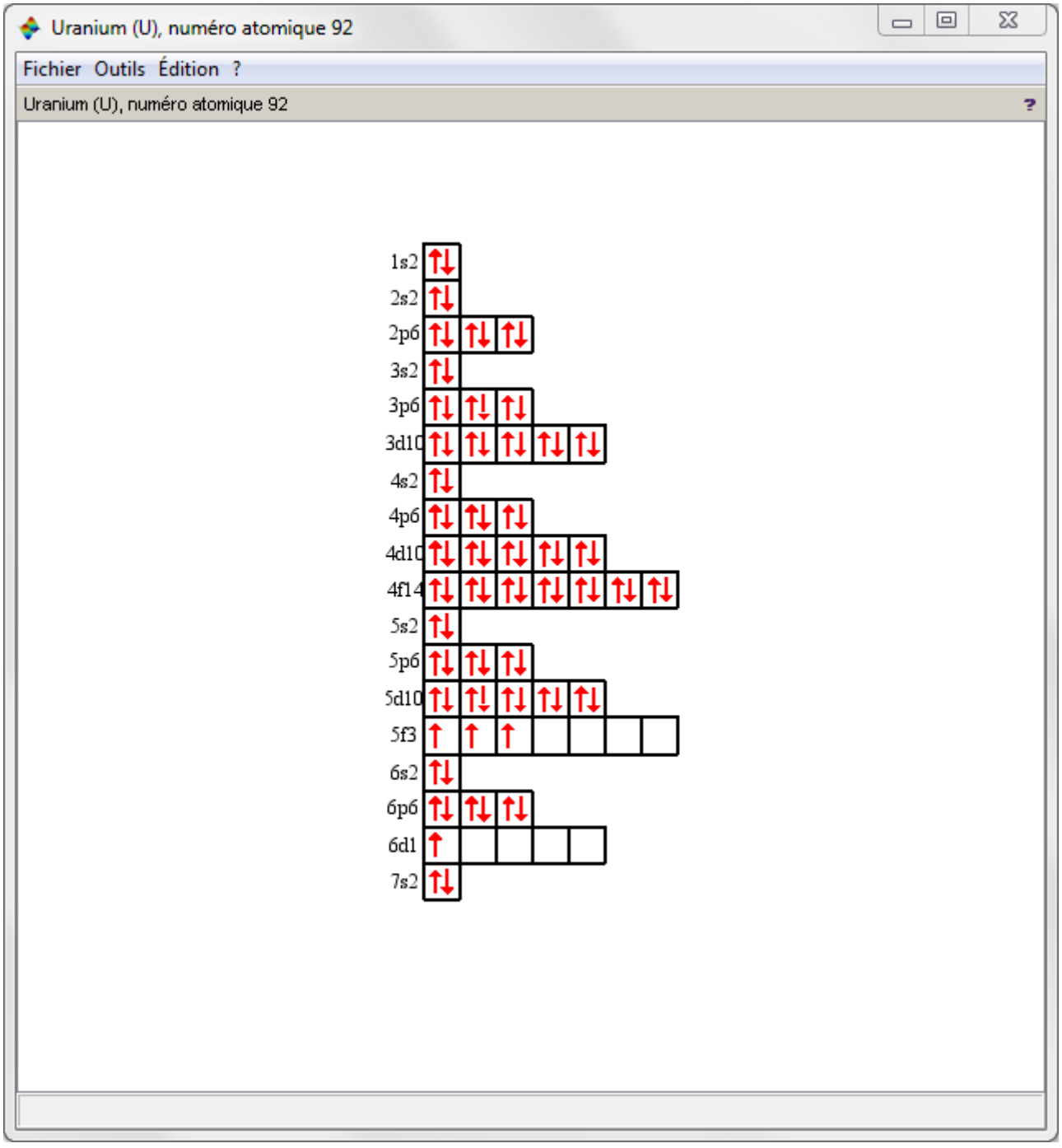




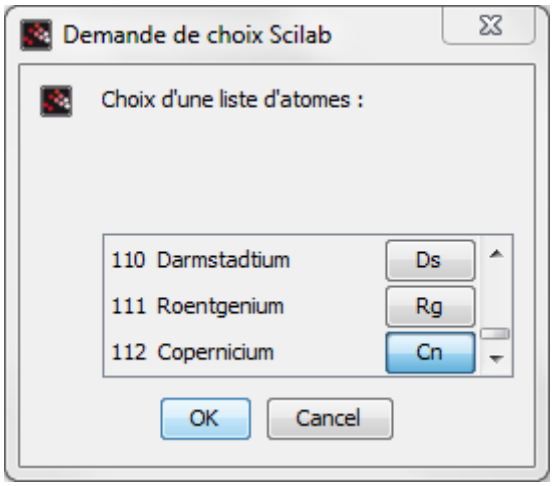


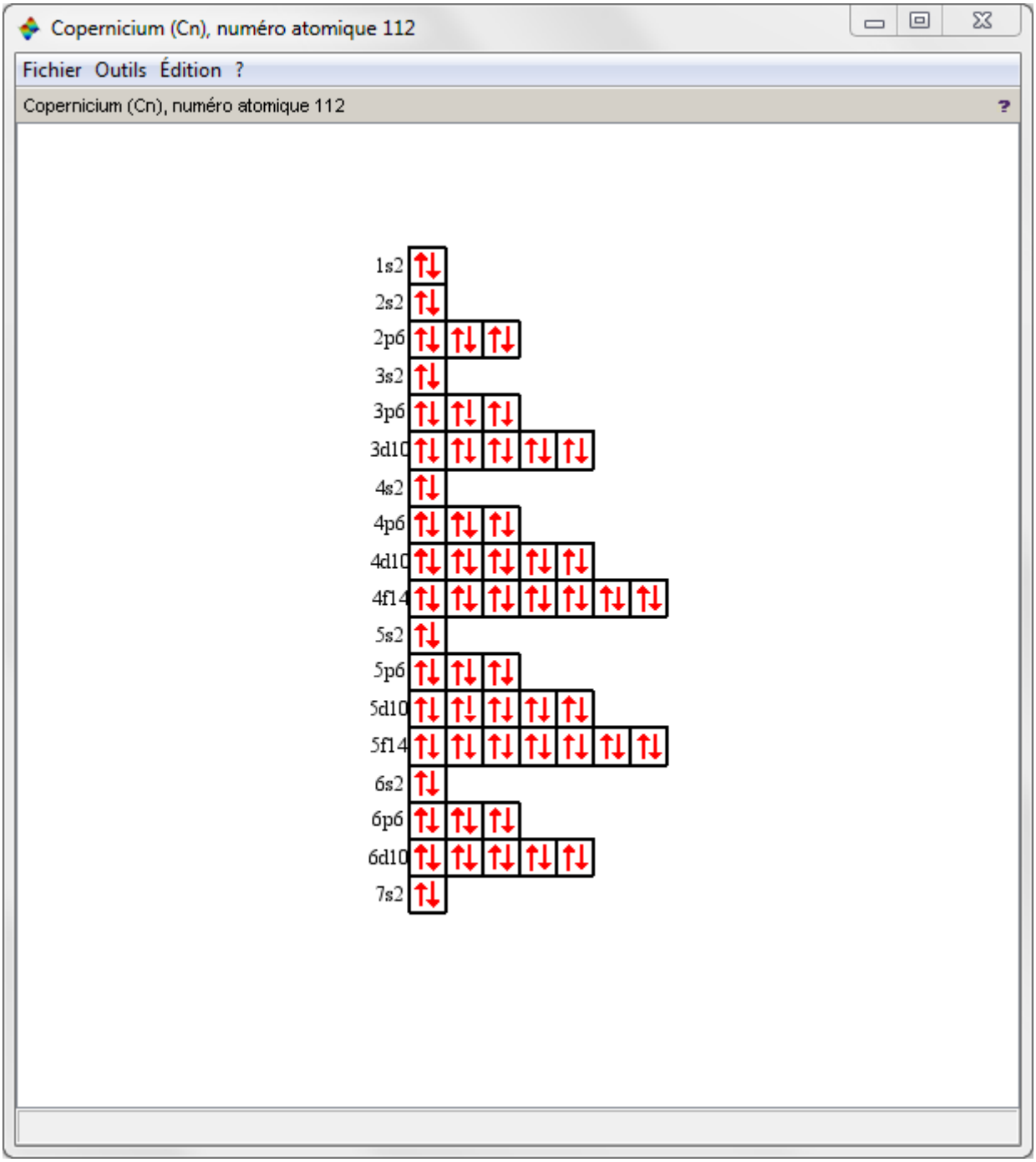
Mathématique & Informatique





Mathématique & Informatique





Mathématique & Informatique

ORBITALES

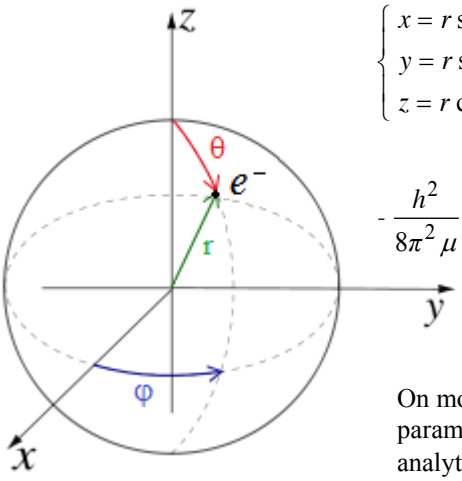
La mécanique quantique est basée sur la notion d'orbitale définie comme étant une région de l'espace où la probabilité de présence de l'électron est élevée (95% à 99%). La forme de l'orbitale est obtenue en résolvant l'équation de **Schrödinger**. Par exemple, pour un atome stable et un électron qui est soumis uniquement au champ électrostatique créé par le noyau, on recherche la fonction d'onde $\psi(x,y,z)$ indépendante du temps qui vérifie l'équation aux dérivées partielles :

$$H\psi = E\psi \Leftrightarrow -\frac{h^2}{8\pi^2\mu}\left(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{\partial^2\psi}{\partial z^2}\right)-\frac{1}{4\pi\varepsilon_0}\frac{Ze^2}{r}\psi = E\psi$$

$h = 6.626\ 10^{-34}$ J.s est la constante de **Planck**.
$\varepsilon_0 = 8.854\ 10^{-12}$ F.m^{-1} est la permittivité du vide.
$e = 1.602\ 10^{-19}$ C est la charge élémentaire.
$\mu = \dfrac{m_e M}{m_e + M} \approx m_e$ où $m_e = 9.109\ 10^{-31}$ kg est la masse de l'électron, $M >> m_e$ est la masse du noyau.

L'électron est situé à la distance $r = \sqrt{x^2+y^2+z^2}$ du centre du noyau de l'atome de numéro atomique Z.
L'énergie totale E de l'électron (inconnue de l'équation) est une valeur propre de l'opérateur hamiltonien H.

Le passage en coordonnées sphériques (r,θ,φ) transforme le laplacien :



$$\begin{cases} x = r\sin\theta\cos\varphi \\ y = r\sin\theta\sin\varphi \\ z = r\cos\theta \end{cases} \qquad r\in[0,+\infty[\,,\ \theta\in[0,\pi]\,,\ \varphi\in[0,2\pi]$$

$$-\frac{h^2}{8\pi^2\mu}\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial\psi}{\partial r}\right)+\frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial\psi}{\partial\theta}\right)+\frac{1}{r^2\sin^2\theta}\frac{\partial^2\psi}{\partial\varphi^2}\right]-\frac{1}{4\pi\varepsilon_0}\frac{Ze^2}{r}\psi = E\psi$$

On montre qu'il est possible de séparer les variables r, θ, φ et que l'introduction de 3 paramètres entiers n, l, m appelés nombres quantiques permet de trouver l'expression analytique de la fonction d'onde $\psi(r,\theta,\varphi)$.

Dans le cas de l'atome d'hydrogène (Z = 1), on aboutit à la formule suivante :

$$\psi_{n,l,m}(r,\theta,\varphi) = R_{n,l}(r).Y_{l,m}(\theta,\varphi) \quad \text{avec} \quad n = 1, 2, 3, \ldots;\ 0 \le l \le n-1\ ;\ -l \le m \le l$$

où $R_{n,l}(r)$ et $Y_{l,m}(\theta,\varphi)$ sont respectivement la partie radiale et la partie angulaire de la fonction d'onde.

$$R_{n,l}(r) = \sqrt{\left(\frac{2}{na_0}\right)^3\frac{(n-l-1)!}{2n(n+l)!}}\ \rho^l\ L_{n-l-1}^{2l+1}(\rho)\,e^{-\frac{\rho}{2}} \quad \text{avec} \quad \rho = \frac{2r}{na_0} \quad \text{et} \quad a_0 = \frac{\varepsilon_0 h^2}{\pi\mu e^2} \approx 0.53\,\text{Å}$$ le rayon de **Bohr**

$$Y_{l,m}(\theta,\varphi) = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}}\ P_l^{|m|}(\cos\theta)\,e^{im\varphi}$$ sont les fonctions harmoniques sphériques normalisées

$$L_p^q(x) = \frac{x^{-q}e^x}{p!}\frac{d^p}{dx^p}\left(x^{p+q}e^{-x}\right)$$ sont les polynômes de **La guerre** généralisés $(p \ge 0$ et $q \ge 0)$

$$P_p^q(x) = \frac{1}{2^p\,p!}\left(1-x^2\right)^{\frac{q}{2}}\frac{d^{p+q}}{dx^{p+q}}(x^2-1)^p$$ sont les fonctions de **Legendre** généralisées $(p \ge 0$ et $0 \le q \le p)$

Les polynômes de **Laguerre** généralisés $L_p^q(x)$ vérifient l'équation différentielle :

$$xy'' + (q + 1 - x)y' + py = 0$$

Ils sont donnés par :

$$L_p^q(x) = \sum_{k=0}^{p} (-1)^k \frac{(p+q)!}{(p-k)!\,(q+k)!\,k!}\, x^k$$

Ainsi, pour $p = 0, 1, 2, 3$:

$$\begin{aligned}
L_0^q(x) &= 1 \\
L_1^q(x) &= -x + q + 1 \\
L_2^q(x) &= \frac{1}{2}x^2 - (q+2)x + \frac{1}{2}(q+1)(q+2) \\
L_3^q(x) &= -\frac{1}{6}x^3 + \frac{1}{2}(q+3)x^2 - \frac{1}{2}(q+2)(q+3)x + \frac{1}{6}(q+1)(q+2)(q+3)
\end{aligned}$$

Propriété d'orthogonalité : $$\int_0^{+\infty} x^q e^{-x} L_i^q(x) L_j^q(x)\,dx = \frac{(i+q)!}{i!}\,\delta_{ij}$$

Les fonctions de **Legendre** généralisées $P_p^q(x)$ vérifient l'équation différentielle :

$$(1 - x^2)y'' - 2xy' + \left[p(p+1) - \frac{q^2}{1-x^2} \right] y = 0$$

Elles se calculent par récurrence :

$$\begin{aligned}
&\text{Sur } p: \quad (p - q + 1)P_{p+1}^q(x) - (2p+1)xP_p^q(x) + (p+q)P_{p-1}^q(x) = 0 \\
&\text{Sur } q: \quad (1-x^2)^{1/2} P_p^{q+1}(x) - 2qxP_p^q(x) + (p+q)(p-q+1)(1-x^2)^{1/2} P_p^{q-1}(x) = 0
\end{aligned}$$

Ainsi, pour $p = 0, 1, 2, 3$:

$$\begin{aligned}
P_0^0(x) &= 1 \\
P_1^0(x) &= x \\
P_1^1(x) &= (1-x^2)^{1/2} \\
P_2^0(x) &= \frac{1}{2}(3x^2 - 1) \\
P_2^1(x) &= 3x(1-x^2)^{1/2} \\
P_2^2(x) &= 3(1-x^2) \\
P_3^0(x) &= \frac{1}{2}x(5x^2 - 3) \\
P_3^1(x) &= \frac{3}{2}(5x^2 - 1)(1-x^2)^{1/2} \\
P_3^2(x) &= 15x(1-x^2) \\
P_3^3(x) &= 15(1-x^2)^{3/2}
\end{aligned}$$

Propriété d'orthogonalité : $$\int_{-1}^{+1} P_i^q(x) P_j^q(x)\,dx = \frac{2}{2i+1}\frac{(i+q)!}{(i-q)!}\,\delta_{ij}$$

On en déduit les premières valeurs de la fonction d'onde $\psi_{n,l,m}(r,\theta,\varphi)$ de l'atome d'hydrogène :

n	l	m	$\psi_{n,l,m}(r,\theta,\varphi)$
1	0	0	$\frac{1}{\sqrt{\pi}\ a_0^{3/2}} e^{-\frac{r}{a_0}}$
2	0	0	$\frac{1}{4\sqrt{2\pi}\ a_0^{5/2}} (-r+2a_0) e^{-\frac{r}{2a_0}}$
2	1	0	$\frac{1}{4\sqrt{2\pi}\ a_0^{5/2}} r\cos\theta\, e^{-\frac{r}{2a_0}}$
2	1	1	$\frac{1}{8\sqrt{\pi}\ a_0^{5/2}} r\sin\theta\, e^{-\frac{r}{2a_0}} e^{i\varphi}$
2	1	-1	$\frac{1}{8\sqrt{\pi}\ a_0^{5/2}} r\sin\theta\, e^{-\frac{r}{2a_0}} e^{-i\varphi}$
3	0	0	$\frac{1}{81\sqrt{3\pi}\ a_0^{7/2}} \left(2r^2-18a_0r+27a_0^2\right) e^{-\frac{r}{3a_0}}$
3	1	0	$\frac{2}{81\sqrt{2\pi}\ a_0^{7/2}} \left(-r^2+6a_0r\right)\cos\theta\, e^{-\frac{r}{3a_0}}$
3	1	1	$\frac{1}{81\sqrt{\pi}\ a_0^{7/2}} \left(-r^2+6a_0r\right)\sin\theta\, e^{-\frac{r}{3a_0}} e^{i\varphi}$
3	1	-1	$\frac{1}{81\sqrt{\pi}\ a_0^{7/2}} \left(-r^2+6a_0r\right)\sin\theta\, e^{-\frac{r}{3a_0}} e^{-i\varphi}$
3	2	0	$\frac{1}{81\sqrt{6\pi}\ a_0^{7/2}} r^2\left(3\cos^2\theta - 1\right) e^{-\frac{r}{3a_0}}$
3	2	1	$\frac{1}{81\sqrt{\pi}\ a_0^{7/2}} r^2\sin\theta\cos\theta\, e^{-\frac{r}{3a_0}} e^{i\varphi}$
3	2	-1	$\frac{1}{81\sqrt{\pi}\ a_0^{7/2}} r^2\sin\theta\cos\theta\, e^{-\frac{r}{3a_0}} e^{-i\varphi}$
3	2	2	$\frac{1}{162\sqrt{\pi}\ a_0^{7/2}} r^2\sin^2\theta\, e^{-\frac{r}{3a_0}} e^{2i\varphi}$
3	2	-2	$\frac{1}{162\sqrt{\pi}\ a_0^{7/2}} r^2\sin^2\theta\, e^{-\frac{r}{3a_0}} e^{-2i\varphi}$

L'énergie de l'électron est quantifiée, pour l'atome d'hydrogène elle ne dépend que du nombre quantique n :

$$E_n = -\frac{E_0}{n^2} \text{ avec } E_0 = \frac{\mu e^4}{8\varepsilon_0{}^2 h^2}$$

E_0 représente l'énergie d'ionisation de l'atome d'hydrogène ($H \rightarrow H^+ + e^-$), on l'exprime habituellement en électron-volt ($E_0 \approx 13.6$ eV) ou en rydberg ($E_0 = 1$ Ryd) si l'on se réfère à la formule de **Rydberg** $\frac{1}{\lambda} = \frac{E_0}{hc}\left(\frac{1}{p^2} - \frac{1}{n^2}\right)$ donnant la longueur d'onde du rayonnement lumineux émis lorsqu'un électron effectue une transition du niveau n au niveau $p < n$.

L'électron possède un quantum d'énergie déterminé à l'intérieur de son orbitale, il présente en outre un caractère ondulatoire stationnaire auquel est associé une longueur d'onde de **De Broglie** : $\lambda = \frac{h}{m_e v_e}$, v_e désignant la vitesse de l'électron.

Le postulat de **Born** stipule que la probabilité de présence dP de l'électron dans un élément de volume $d\tau$ est liée à la fonction d'onde ψ par la relation **$dP = \psi^2 d\tau$**. Autrement dit, le module de ψ au carré est une densité volumique de probabilité.

L'élément de volume en coordonnées cartésiennes est $d\tau = dx\,dy\,dz$, on a :

$$\iiint_{R^3} \psi^2 d\tau = 1$$

A noter que ψ est un nombre complexe et que ψ^2 est en fait le nombre réel $|\psi|^2 = \psi\psi^*$, ψ^* conjugué de ψ.
L'électron a d'autant plus de chances de se trouver au voisinage immédiat du point (x,y,z) que $\psi^2(x,y,z)$ est grande en ce point (interprétation infinitésimale de la densité de probabilité), ce qui ne signifie pas que l'électron se trouve en moyenne aux endroits où ψ^2 est la plus élevée. On définit un rayon moyen (espérance mathématique) pour une densité de probabilité ψ^2 :

$$\langle r \rangle = \iiint_{R^3} r\psi^2 d\tau = \frac{3n^2 - l(l+1)}{2} a_0$$

Pour l'atome d'hydrogène dans son état fondamental :

$$\psi^2 = |\psi_{1,0,0}|^2 = \frac{1}{\pi a_0{}^3} e^{-\frac{2r}{a_0}} \quad \text{et} \quad \langle r \rangle = \langle r_{1,0,0} \rangle = \frac{3}{2} a_0 \approx 0.79 \text{ Å}$$

Si l'on souhaite représenter graphiquement une orbitale où la probabilité de présence de l'électron est P (P = 98% par exemple), on recherchera la surface d'isodensité d'équation $\psi^2(x,y,z) = \kappa$ = constante, enveloppant le volume $V(\kappa)$ tel que :

$$\iiint_{V(\kappa)} \psi^2 d\tau = P$$

L'équation différentielle de départ convient pour modéliser un ion hydrogénoïde (He^+, Li^{2+}, Be^{3+}, …), il suffit de remplacer e par $\sqrt{Z}e$ dans les calculs. Pour des atomes polyélectroniques, on ne peut pas négliger les interactions électrostatiques entre les électrons et l'équation de **Schrödinger** devient beaucoup plus difficile à résoudre : l'opérateur hamiltonien H est une somme de laplaciens et d'énergies potentielles, d'où l'impossibilité de trouver l'expression analytique exacte de la fonction d'onde $\psi(x_1,y_1,z_1, \ldots, x_i,y_i,z_i, \ldots, x_N,y_N,z_N)$ attachée à l'ensemble des N électrons.
Seules des approximations mathématiques et des méthodes numériques (modèle de **Slater**, algorithme de **Hartree - Fock**) permettent d'approcher la solution.
Des logiciels spécifiques ont été développés pour dessiner les orbitales électroniques d'atomes ou de molécules.
Ajoutons que l'on tente actuellement de visualiser le mouvement réel des électrons dans la matière à partir d'impulsions laser, ce qui nécessite à la fois une précision dans l'espace de l'ordre de l'angström et dans le temps de l'ordre de l'attoseconde.

Liens internet pour accéder à quelques documents concernant la mécanique quantique :

- ⇨ **http://www.ac-nancy-metz.fr/enseign/physique/chim/jumber/atomistique/atomistique_fichiers/atomisti.html**
- ⇨ **http://www.ensta-paristech.fr/~grimaud/CB101/poly_cours/chap1.pdf**
- ⇨ **http://dlst.ujf-grenoble.fr/data/cours/documents/CHI110-20101008123946.pdf**
- ⇨ **http://www.phys.ens.fr/~dalibard/transparentsX/2010/cours4.pdf**
- ⇨ **http://blake.ism.u-bordeaux1.fr/~crespos/documents/KHI501_CHAP2-1011.pdf**

Mathématique & Informatique

Représentation graphique de la densité de probabilité

Réalisons deux graphiques où les couleurs, du bleu clair au violet, indiquent la valeur croissante de la densité de probabilité. Dans le cas de l'atome d'hydrogène, ψ^2 dépendant uniquement de r et de θ, on effectue une représentation dans l'espace avec la fonction Scilab *fplot3d1* et une représentation dans le plan avec la fonction Scilab *Sfgrayplot*.

Exemple : $n = 3$, $l = 2$, $m = 0$

La densité volumique de probabilté est :

$$\psi^2 = \left[\frac{1}{81\sqrt{6\pi}\ {a_0}^{7/2}} r^2 \left(3\cos^2\theta - 1\right) e^{-\frac{r}{3a_0}} \right]^2$$

La valeur minimale de ψ^2 est 0, elle est atteinte pour $r = 0$ ou $r = +\infty$ et également pour $\theta = \pm \arccos\left(\pm \frac{1}{\sqrt{3}}\right)$

La valeur maximale de ψ^2 est atteinte pour $r = 6a_0 \approx 3.18$ Å avec $\theta = 0$ ou $\theta = \pi$. On a : $\psi^2_{max} = \frac{864}{6561\pi a_0^3 e^4}$

On obtient le programme suivant :

```
// Densité de probabilité
function d=psi2(r, theta)
    d = 1000*(1/(81*sqrt(6*%pi)*a0^(7/2))*r^2*(3*cos(theta)^2-1)*exp(-r/(3*a0)))^2
endfunction

// Paramètres
a0 = 0.53 ;
rmax = 6*a0 ;
psimax = 864/(6561*%pi*a0^3*%e^4) ;

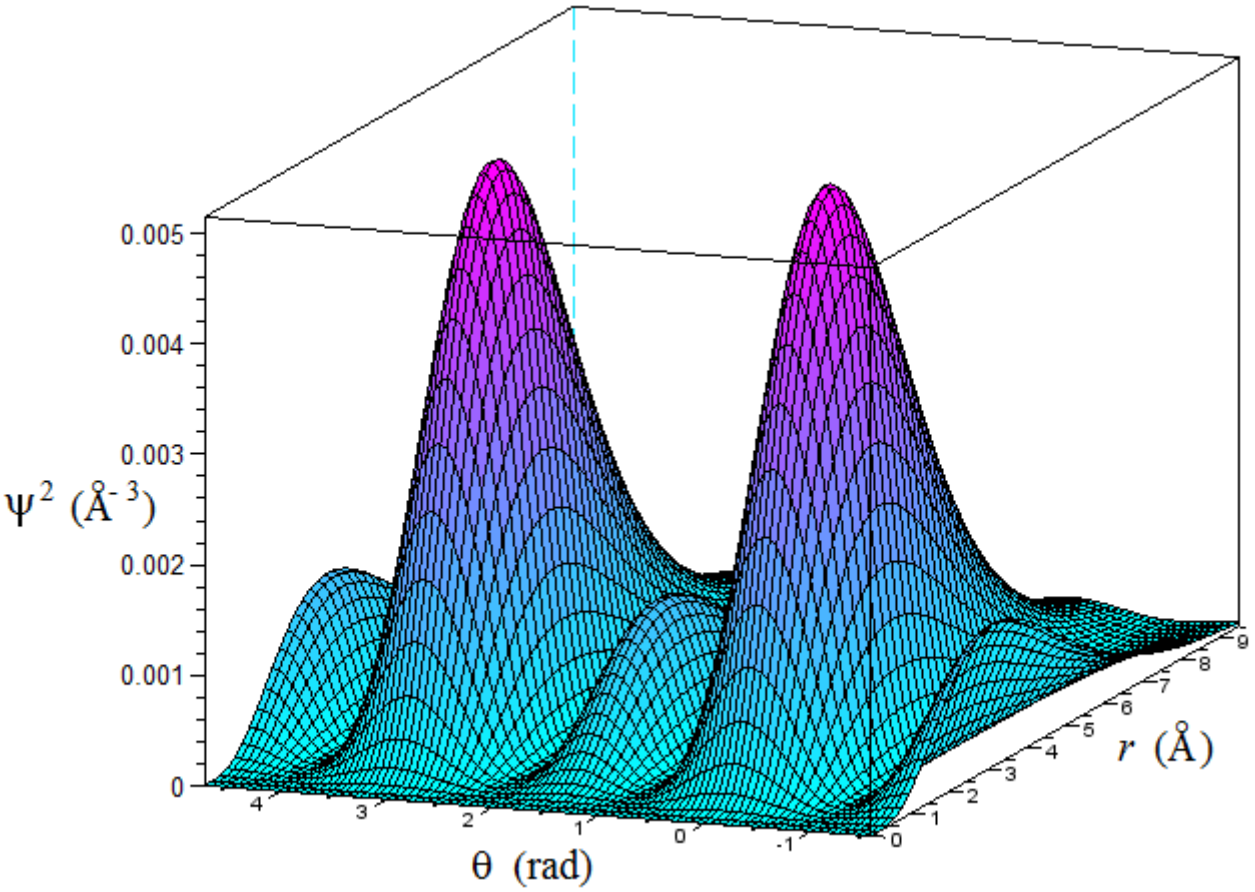
// Graphiques
f1 = scf(1) ;
f1.color_map = coolcolormap(32) ;
r = linspace(0,3*rmax,50) ;
theta = linspace(-%pi/2,3*%pi/2,100) ;
a = gca() ; a.tight_limits = 'on' ; a.data_bounds = [0,-%pi/2,0;3*rmax,3*%pi/2,psimax] ;
fplot3d1(r,theta,psi2,alpha = 78,theta = 200, flag = [1,4,4]) ;

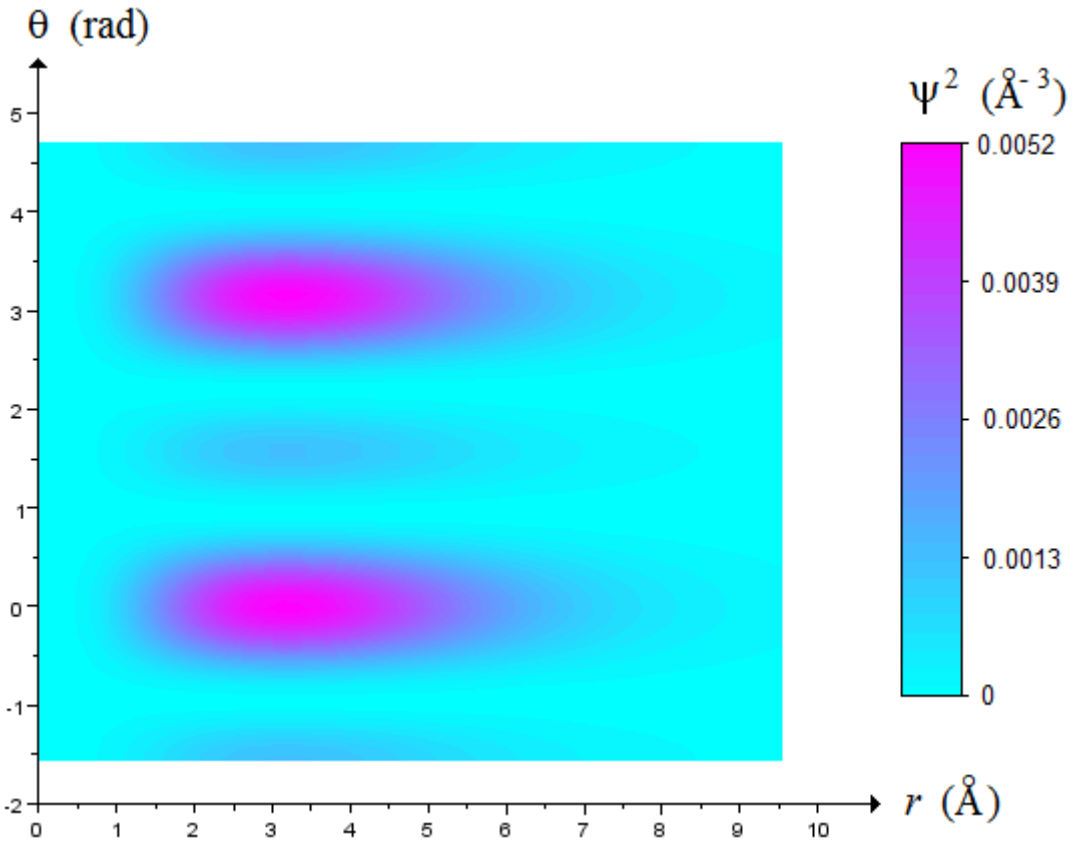
f2 = scf(2) ;
f2.color_map = coolcolormap(32) ;
r = linspace(0,3*rmax,50) ;
theta = linspace(-%pi/2,3*%pi/2,50) ;
colorbar(0,1000*psimax) ;
Sfgrayplot(r,theta,psi2,frameflag = 2) ;
```

Remarque importante

Dans le système international d'unités, la densité de probabilité ψ^2 prend de très grandes valeurs (dans notre exemple : $\psi^2_{max} = 5.16\ 10^{27}$ m^{-3}), peu pratiques dans les calculs et les graphiques. On se place donc à l'échelle atomique avec des longueurs de l'ordre de l'angström (1 Å = 10^{-10} m), ce qui nous donne de petites valeurs pour ψ^2, de l'ordre de quelques millièmes par angström au cube (dans notre exemple : $\psi^2_{max} = 5.16\ 10^{-3}$ Å^{-3}).

Mathématique & Informatique





Mathématique & Informatique

Calcul du rayon moyen d'une orbitale de l'atome d'hydrogène

Par définition :

$$\langle r \rangle = \iiint_{R^3} r\,\psi^2 d\tau$$

L'élément de volume en coordonnées sphériques est $d\tau = r^2 \sin\theta dr d\theta d\varphi$, par suite :

$$\langle r_{n,l,m} \rangle = \int_0^{2\pi}\int_0^{\pi}\int_0^{\infty} r^3 \psi_{n,l,m}(r,\theta,\varphi)^2 \sin\theta\, dr d\theta d\varphi = \int_0^{\infty} r^3 R_{n,l}(r)^2\, dr \cdot \int_0^{2\pi}\int_0^{\pi} Y_{l,m}(\theta,\varphi)^2 \sin\theta\, d\theta d\varphi$$

L'intégrale double en θ et φ vaut 1 car les fonctions harmoniques $Y_{l,m}$ sont orthonormales sur la sphère unité (utiliser la propriété d'orthogonalité des fonctions de **Legendre** généralisées). Exprimons la partie radiale $R_{n,l}$ de la fonction d'onde ψ :

$$\langle r_{n,l,m} \rangle = \int_0^{\infty} r^3 R_{n,l}(r)^2\, dr = \left(\frac{(n-l-1)!}{4(n+l)!} \int_0^{\infty} \rho^{2l+3} e^{-\rho} \left[L_{n-l-1}^{2l+1}(\rho) \right]^2 d\rho \right) a_0 = c_{n,l}\, a_0$$

Le coefficient $c_{n,l}$ est un nombre rationnel qui ne dépend que des nombres quantiques n et l. Calculons sa valeur. Rappelons tout d'abord la propriété d'orthogonalité des polynômes de **Laguerre** généralisés :

$$\int_0^{\infty} x^q e^{-x} L_i^q(x) L_j^q(x)\, dx = \frac{(i+q)!}{i!}\delta_{ij} \qquad (i, j, q \text{ sont des entiers et } \delta_{ij} \text{ est le symbole de } \textbf{Kronecker})$$

Ajoutons la relation de récurrence qui permet d'incrémenter l'indice q : $L_p^q(x) = L_p^{q+1}(x) - L_{p-1}^{q+1}(x)$

On recherche une intégrale de la forme $\int_0^{\infty} x^{q+2} e^{-x} \left[L_p^q(x) \right]^2 dx$. Ecrivons, avec la convention $L_{p-i}^{q+k}(x) = 0$ si $p - i < 0$:

$$\left[L_p^q(x) \right]^2 = \left[L_p^{q+1}(x) - L_{p-1}^{q+1}(x) \right]^2 = \left[L_p^{q+2}(x) - 2L_{p-1}^{q+2}(x) + L_{p-2}^{q+2}(x) \right]^2$$

Dans le développement du membre de droite, il suffit de retenir les termes au carré puisque tous les autres termes multipliés par la fonction poids $x^{q+2}e^{-x}$ et intégrés de 0 à l'infini correspondent à un produit scalaire nul. Il vient :

$$\int_0^{\infty} x^{q+2} e^{-x} \left[L_p^q(x) \right]^2 dx = \int_0^{\infty} x^{q+2} e^{-x} \left[L_p^{q+2}(x) \right]^2 dx + 4\int_0^{\infty} x^{q+2} e^{-x} \left[L_{p-1}^{q+2}(x) \right]^2 dx + \int_0^{\infty} x^{q+2} e^{-x} \left[L_{p-2}^{q+2}(x) \right]^2 dx$$

Soit, en admettant que $\frac{1}{(p-i)!} = 0$ pour $0 \le p < i$:

$$\int_0^{\infty} x^{q+2} e^{-x} \left[L_p^q(x) \right]^2 dx = \frac{(p+q+2)!}{p!} + 4\frac{(p+q+1)!}{(p-1)!} + \frac{(p+q)!}{(p-2)!} = \frac{(p+q)!}{p!}\left[6p(p+q+1) + (q+1)(q+2) \right]$$

Pour obtenir la valeur de $c_{n,l}$, on prend $p = n - l - 1$ et $q = 2l + 1$, puis on multiplie par le facteur $\frac{(n-l-1)!}{4(n+l)!}$:

$$c_{n,l} = \frac{(n-l-1)!}{4(n+l)!}\frac{(n+l)!}{(n-l-1)!}\left[6(n-l-1)(n+l+1) + (2l+2)(2l+3) \right] = \frac{3n^2 - l(l+1)}{2}$$

Donc :

$$\langle r_{n,l,m} \rangle = \frac{3n^2 - l(l+1)}{2} a_0$$

Par exemple, pour l'atome d'hydrogène dans son état fondamental : $\langle r_{1,0,0} \rangle = \frac{3}{2} a_0 \approx \frac{3}{2} \times 0.53 \approx 0.79$ Å.

Le rayon moyen de l'orbitale $n = 3$, $l = 2$, $m = 0$ de l'atome d'hydrogène est : $\langle r_{3,2,0} \rangle = \frac{21}{2} a_0 \approx 5.56$ Å.

Mathématique & Informatique

Calcul de l'écart-type du rayon d'une orbitale de l'atome d'hydrogène

Par définition :

$$\sigma = \sqrt{\langle r^2\rangle - \langle r\rangle^2} = \sqrt{\iiint_{R^3} r^2\,\psi^2 d\tau - \left(\iiint_{R^3} r\,\psi^2 d\tau\right)^2}$$

Connaissant $\langle r\rangle$, il reste à calculer le moment d'ordre 2 de la variable aléatoire, c'est-à-dire :

$$\langle r^2\rangle = \iiint_{R^3} r^2\,\psi^2 d\tau = \int_0^\infty r^4 R_{n,l}(r)^2\,dr = \left(\frac{n(n-l-1)!}{8(n+l)!}\int_0^\infty \rho^{2l+4} e^{-\rho}\left[L_{n-l-1}^{2l+1}(\rho)\right]^2 d\rho\right) a_0^2$$

La relation $L_p^q(x) = L_p^{q+1}(x) - L_{p-1}^{q+1}(x)$ se généralise à un entier k quelconque : $L_p^q(x) = \sum_{i=0}^{\mathrm{Min}(k,p)} (-1)^i \binom{k}{i} L_{p-i}^{q+k}(x)$

Compte tenu de l'orthogonalité des polynômes de **Laguerre**, on déduit quels que soient les entiers k, p, q :

$$\int_0^\infty x^{q+k} e^{-x}\left[L_p^q(x)\right]^2 dx = \sum_{i=0}^{\mathrm{Min}(k,p)} \left[\frac{k!}{i!(k-i)!}\right]^2 \frac{(p+q+k-i)!}{(p-i)!}$$

En particulier, en prenant $k = 3$ et en convenant que $\frac{1}{(p-i)!} = 0$ si $0 \le p < i$, on a :

$$\int_0^\infty x^{q+3} e^{-x}\left[L_p^q(x)\right]^2 dx = \frac{(p+q+3)!}{p!} + 9\frac{(p+q+2)!}{(p-1)!} + 9\frac{(p+q+1)!}{(p-2)!} + \frac{(p+q)!}{(p-3)!}$$

$$= \frac{(p+q)!}{p!}\left[(p+q+1)(p+q+2)(p+q+3) + 9p(p+q+1)(p+q+2) + 9p(p-1)(p+q+2) + p(p-1)(p-2)\right]$$

Pour $p = n - l - 1$, $q = 2l + 1$, $x = \rho$, on obtient :

$$\int_0^\infty \rho^{2l+4} e^{-\rho}\left[L_{n-l-1}^{2l+1}(\rho)\right]^2 d\rho = \frac{(n+l)!}{(n-l-1)!}\left\{[n+(l+1)][n+(l+2)][10n-(8l+6)] + [n-(l+1)][n-(l+2)][10n+(8l+6)]\right\}$$

D'où, après simplification de l'expression entre accolades :

$$\iiint_{R^3} r^2\,\psi^2 d\tau = \frac{n(n-l-1)!}{8(n+l)!}\frac{(n+l)!}{(n-l-1)!}\left\{20n^3 - 4(3l^2+3l-1)n\right\}a_0^2 = \frac{5n^4 - (3l^2+3l-1)n^2}{2}a_0^2$$

Reportons cette valeur dans la formule de l'écart-type : $\sigma_{n,l,m} = \sqrt{\frac{5n^4 - (3l^2+3l-1)n^2}{2}a_0^2 - \left(\frac{3n^2 - l(l+1)}{2}a_0\right)^2}$

Donc :

$$\sigma_{n,l,m} = \frac{\sqrt{(n^2+1)^2 - l^2(l+1)^2 - 1}}{2}a_0$$

Par exemple, pour l'atome d'hydrogène dans son état fondamental : $\sigma_{1,0,0} = \frac{\sqrt{3}}{2}a_0 \approx \frac{\sqrt{3}}{2}\times 0.53 \approx 0.46$ Å.

L'écart-type du rayon de l'orbitale $n = 3$, $l = 2$, $m = 0$ de l'atome d'hydrogène est : $\sigma_{3,2,0} = \frac{\sqrt{63}}{2}a_0 \approx 2.10$ Å.

Mathématique & Informatique

Représentation graphique d'une orbitale

On recherche une surface d'isodensité $\psi^2 = \kappa$ = constante enveloppant un volume qui contient "presque sûrement" l'électron. Pour déterminer les points (r,θ,φ) de cette surface, on fixe $\theta = \theta_0$ et $\varphi = \varphi_0$, puis on calcule les valeurs de r vérifiant l'égalité $\psi^2(r,\theta_0,\varphi_0) = \kappa$, ce qui revient à résoudre un équation du type $f(r) = 0$. Le point (r_0,θ_0,φ_0), où r_0 est la plus grande des racines, appartient à la surface. En discrétisant θ sur $[0,\pi]$ et φ sur $[0,2\pi]$, on obtient l'ensemble des points (r_0,θ_0,φ_0) qui permettent de construire les facettes de l'orbitale.
En toute rigueur l'expression de la densité de probabilité ψ^2 ne dépend pas de φ puisque $|e^{im\varphi}| = 1$. Néanmoins, afin de différencier une orbitale de nombres quantiques n, l, $-|m|$ et une orbitale de nombres quantiques n, l, $+|m|$, on représente la fonction d'onde $\psi_{n,l,m}$ par sa partie réelle si $m < 0$ (ce qui revient à multiplier la fonction densité ψ^2 par le facteur $\cos^2(m\varphi)$) ou par sa partie imaginaire si $m > 0$ (ce qui revient à multiplier la fonction densité ψ^2 par le facteur $\sin^2(m\varphi)$).
Ajoutons que l'on choisit généralement 2 couleurs différentes numérotées c_1 et c_2 pour indiquer le signe de la fonction d'onde sur la surface d'isodensité : le point (x,y,z) est de couleur c_1 si $\psi(x,y,z) > 0$ ou de couleur c_2 si $\psi(x,y,z) < 0$.

Le paramètre R_0

Pour obtenir un point (x,y,z) de la surface d'isodensité, on calcule au préalable ses coordonnées sphériques (r,θ,φ), donc le rayon r si θ et φ sont fixés : $\theta = \theta_0$ et $\varphi = \varphi_0 \Rightarrow r = r_0$.
Trouver r_0 revient à résoudre une équation $f(r) = 0$ avec la fonction Scilab *fsolve*.
L'instruction $r_0 = fsolve(R_0,f)$ nécessite de fournir une approximation grossière R_0 de la racine recherchée r_0.
On peut prendre R_0 égal au rayon de la n^e orbite de **Bohr**, c'est-à-dire $R_0 = a_n = n^2 a_0$ où n est le nombre quantique principal et $a_0 \approx 0.53$ Å. Par exemple, pour $n = 3$: $R_0 = a_3 = 3^2 a_0 = 9a_0 \approx 4.77$ Å.
Une autre possibilité, plus précise, consiste à attribuer à R_0 le rayon moyen de l'orbitale (n, l, m). Dans ce cas :

$$R_0 = \langle r_{n,l,m} \rangle = \frac{3n^2 - l(l+1)}{2} a_0$$ où n est le nombre quantique principal et l est le nombre quantique secondaire.

Par exemple, pour $n = 3$ et $l = 2$: $R_0 = \langle r_{3,2,0} \rangle = \frac{3 \cdot 3^2 - 2(2+1)}{2} a_0 = \frac{21}{2} a_0 \approx 5.56$ Å.

Le paramètre κ

On prend une valeur de κ suffisamment petite pour que la surface d'isodensité $\psi^2 = \kappa$ enveloppe un volume V(κ) où la probabilité de présence P de l'électron est significative (au moins 50%).

$$P(\kappa) = \iiint_{V(\kappa)} \psi^2 d\tau$$

Pour obtenir une valeur approchée de cette intégrale, on utilise la fonction Scilab *int3d*.
Dans un premier temps, le volume V(κ) est scindé en tétraèdres élémentaires V_i construits à partir des facettes de la surface d'isodensité : $V(\kappa) = \Sigma V_i$. Chaque facette possède 4 sommets coplanaires A, B, C, D. Les 2 triangles (A,B,C) et (A,C,D) auxquels on ajoute l'origine O intérieure à V(κ) forment 2 tétraèdres (A,B,C,O) et (A,C,D,O) de volumes V_1 et V_2.
La fonction *int3d* impose que tous les volumes V_i soient non nuls (en pratique on prendra $V_i > \varepsilon$, avec $\varepsilon = 10^{-6}$ par exemple).
Rappelons qu'un tétraèdre de sommets S_1, S_2, S_3, S_4 a pour volume : $V = \frac{1}{6}\left|\det\left(\overrightarrow{S_1S_2}, \overrightarrow{S_1S_3}, \overrightarrow{S_1S_4}\right)\right|$.
Dans un deuxième temps, on définit la fonction densité $\psi^2(x,y,z)$ en coordonnées cartésiennes à partir de son expression $\psi^2(r,\theta,\varphi)$ en coordonnées sphériques ; il suffit pour cela de convertir les variables (x,y,z) en variables (r,θ,φ).
Finalement, les calculs sont regroupés dans une fonction *integrale* qui fournit la probabilté de présence P(κ) de l'électron.
Par exemple, pour l'orbitale $(n = 3, l = 2, m = 0)$, on prend $\kappa = 10^{-5}$ et l'instruction *P = integrale(κ)* donne P = 98%.
Il est possible d'écrire un algorithme de dichotomie qui recherche κ pour $P = P_0$ fixée. Le principe est le suivant :

$\kappa_{min} = 0$; $\kappa_{max} = \psi^2{}_{max}$
répéter
| $\kappa = (\kappa_{min} + \kappa_{max}) / 2$
| si $P(\kappa) < P_0$ alors $\kappa_{max} = \kappa$ sinon $\kappa_{min} = \kappa$
jusqu'à $|P(\kappa) - P_0| < \varepsilon$

```
function P=integrale(K)
    function [x, y, z]=orbitale(phi, theta)
        function d=f(r)
            d = (1/(81*sqrt(6*%pi)*a0^(7/2))*r^2*(3*cos(theta0)^2-1)*exp(-r/(3*a0)))^2 - K
        endfunction
        x = [] ; y = [] ; z = []
        for i = 1:length(phi)
            phi0 = phi(i)
            theta0 = theta(i)
            r0 = fsolve(R0,f)
            x = [x,r0*sin(theta0)*cos(phi0)]
            y = [y,r0*sin(theta0)*sin(phi0)]
            z = [z,r0*cos(theta0)]
        end
    endfunction
    function [X, Y, Z]=tetraedre(xf, yf, zf)
        X = [] ; Y = [] ; Z = []
        for j = 1:size(xf,2)
            A = [xf(1,j);yf(1,j);zf(1,j)]
            B = [xf(2,j);yf(2,j);zf(2,j)]
            C = [xf(3,j);yf(3,j);zf(3,j)]
            D = [xf(4,j);yf(4,j);zf(4,j)]
            V1 = abs(det([A,A-B,A-C]))/6
            V2 = abs(det([A,A-C,A-D]))/6
            if V1 > eps then
                X = [X,[A(1);B(1);C(1);0]]
                Y = [Y,[A(2);B(2);C(2);0]]
                Z = [Z,[A(3);B(3);C(3);0]]
            end
            if V2 > eps then
                X = [X,[A(1);C(1);D(1);0]]
                Y = [Y,[A(2);C(2);D(2);0]]
                Z = [Z,[A(3);C(3);D(3);0]]
            end
        end
    endfunction
    function d=densite(xyz, nf)
        x = xyz(1) ; y = xyz(2) ; z = xyz(3)
        r = sqrt(x^2 + y^2 + z^2)
        if r == 0 then, theta = 0, else, theta = acos(z/r), end
        if x == 0 & y == 0 then, phi = 0, else,
            if y >= 0 then, phi = acos(x/sqrt(x^2+y^2)), else, phi = 2*%pi - acos(x/sqrt(x^2+y^2)), end
        end
        d = (1/(81*sqrt(6*%pi)*a0^(7/2))*r^2*(3*cos(theta)^2-1)*exp(-r/(3*a0)))^2
    endfunction
    phi = linspace(0,2*%pi,Np)
    theta = linspace(0,%pi,Nt)
    [xf,yf,zf] = eval3dp(orbitale,phi,theta)
    [X,Y,Z] = tetraedre(xf,yf,zf)
    [P,err] = int3d(X,Y,Z,densite,1,[0,1e6,1e-2,1e-2])
endfunction

// Paramètres
a0 = 0.53 ; R0 = 5.56 ;
Np = 50 ; Nt = 25 ;
eps = 1e-6 ; K = 1e-5 ;

// Probabilité de présence
P = integrale(K) ;
printf('\n P = %.0f%c',100*P,'%')
```

Exemple : l'orbitale de nombres quantiques $n = 3$, $l = 2$, $m = 0$

$$\psi_{3,2,0} = \frac{1}{81\sqrt{6\pi}\ a_0^{7/2}} r^2 \left(3\cos^2\theta - 1\right) e^{-\frac{r}{3a_0}}$$

$$\langle r_{3,2,0} \rangle = \frac{21}{2} a_0$$

$$\sigma_{3,2,0} = \frac{3\sqrt{7}}{2} a_0$$

```
// Equations
function [x, y, z]=orbitale(phi, theta)
    function d=f(r)
        d = (1/(81*sqrt(6*%pi)*a0^(7/2))*r^2*(3*cos(theta0)^2-1)*exp(-r/(3*a0)))^2 - K
    endfunction
    x = [] ; y = [] ; z = []
    for i = 1:length(phi)
        phi0 = phi(i)
        theta0 = theta(i)
        r0 = fsolve(R0,f)
        x = [x,r0*sin(theta0)*cos(phi0)]
        y = [y,r0*sin(theta0)*sin(phi0)]
        z = [z,r0*cos(theta0)]
    end
endfunction

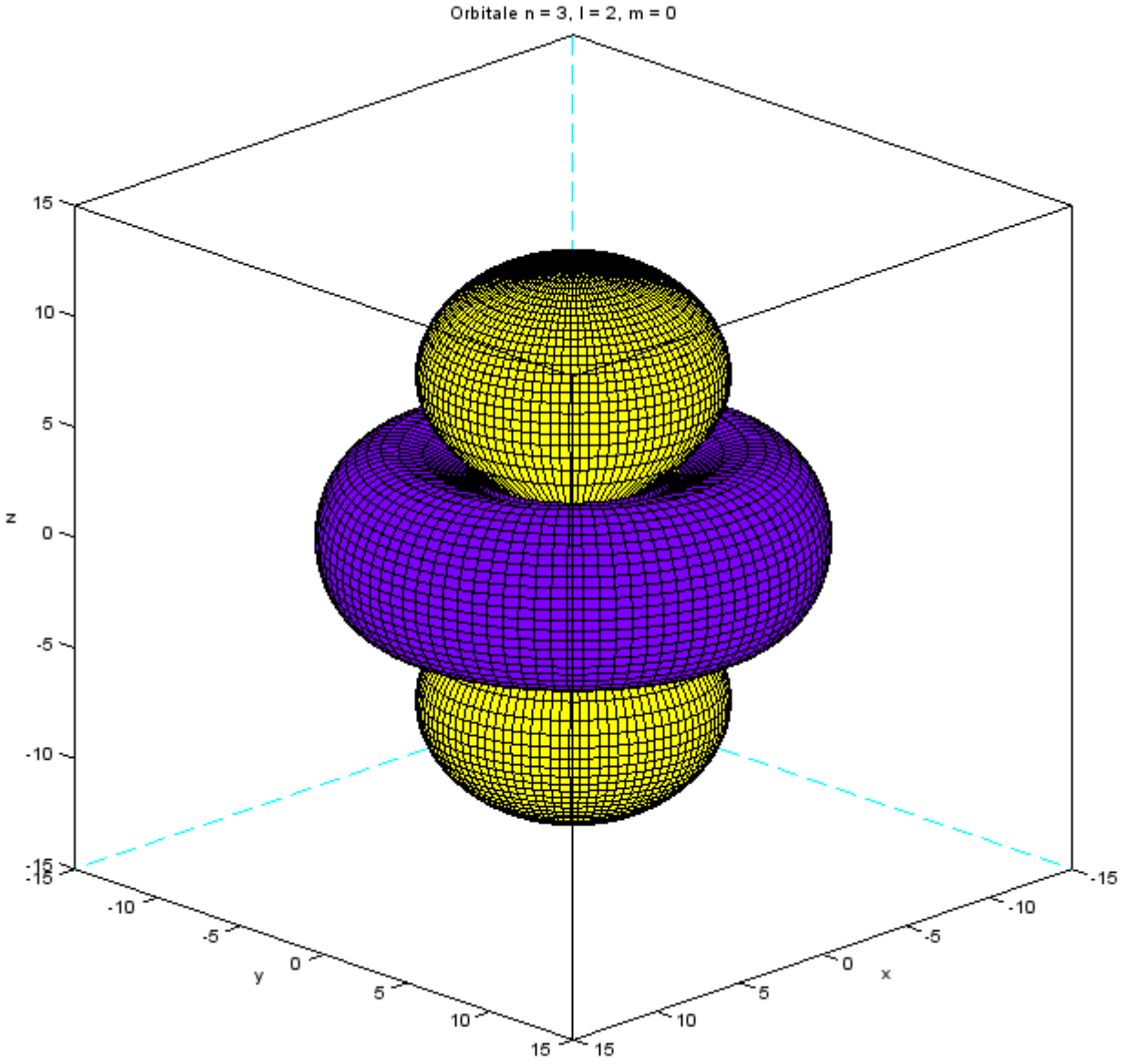
// Paramètres
a0 = 0.53 ;
R0 = 5.56 ;
K = 1e-5 ;

// Graphique
scf(1) ;
c1 = color(255,255,0) ;
c2 = color(127,0,255) ;
Np = 100 ;
Nt = 30 ;
t0 = acos(1/sqrt(3)) ;
drawlater() ;
phi = linspace(0,2*%pi,Np) ;
theta = linspace(0,t0,Nt) ;
[xf,yf,zf] = eval3dp(orbitale,phi,theta) ;
plot3d(xf, yf, zf, theta = 45, alpha = 70, flag = [c1,6,4]) ;
theta = linspace(%pi-t0,%pi,Nt) ;
[xf,yf,zf] = eval3dp(orbitale,phi,theta) ;
plot3d(xf, yf, zf, theta = 45, alpha = 70, flag = [c1,6,4]) ;
theta = linspace(t0,%pi-t0,Nt) ;
[xf,yf,zf] = eval3dp(orbitale,phi,theta) ;
plot3d(xf, yf, zf, theta = 45, alpha = 70, flag = [c2,6,4]) ;
xtitle("Orbitale n = 3, l = 2, m = 0", "x", "y", "z") ;
drawnow() ;
```

Mathématique & Informatique

Atome d'hydrogène

$$\psi_{3,2,0} = \frac{1}{81\sqrt{6\pi}\,a_0^{7/2}} r^2 \left(3\cos^2\theta - 1\right) e^{-\frac{r}{3a_0}}$$



Mathématique & Informatique

Exemple : l'orbitale de nombres quantiques $n = 4$, $l = 3$, $m = -2$

$$\psi_{4,3,-2} = \frac{1}{1024\sqrt{6\pi}\ a_0^{9/2}} r^3 \sin^2\theta \cos\theta\, e^{-\frac{r}{4a_0}}\, e^{-2i\varphi}$$

$$\langle r_{4,3,-2} \rangle = 18a_0$$

$$\sigma_{4,3,-2} = 6a_0$$

```
// Equations
function [x, y, z]=orbitale(phi, theta)
    function d=f(r)
        d = (1/(1024*sqrt(6*%pi)*a0^(9/2))*r^3*sin(theta0)^2*cos(theta0)*exp(-r/(4*a0))*cos(-2*phi0))^2 - K
    endfunction
    x = [] ; y = [] ; z = []
    for i = 1:length(phi)
        phi0 = phi(i)
        theta0 = theta(i)
        r0 = fsolve(R0,f)
        x = [x,r0*sin(theta0)*cos(phi0)]
        y = [y,r0*sin(theta0)*sin(phi0)]
        z = [z,r0*cos(theta0)]
    end
endfunction

// Paramètres
a0 = 0.53 ;
R0 = 9.54 ;
K = 1e-8 ;

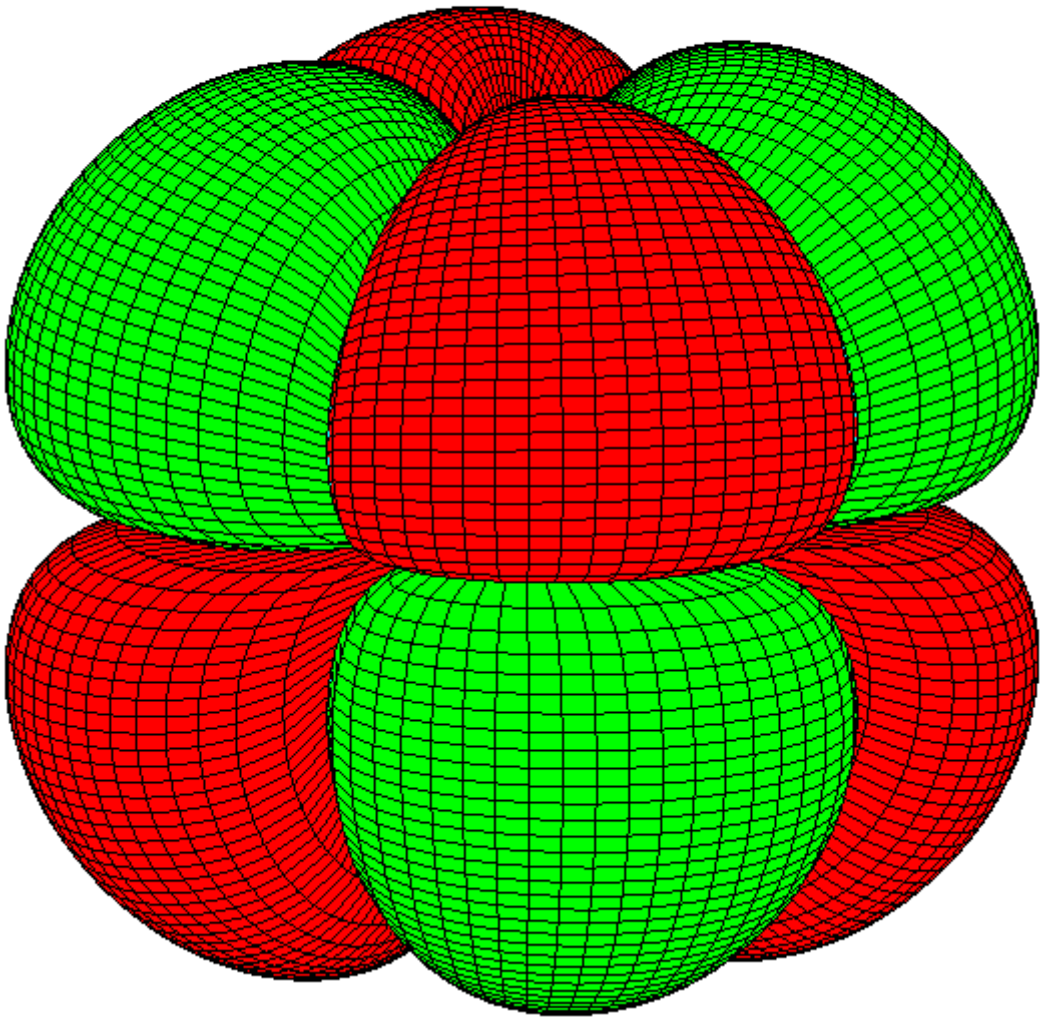
// Graphique
scf(2) ;
c1 = color(0,255,0) ;
c2 = color(255,0,0) ;
Np = 25 ;
Nt = 50 ;
drawlater() ;
for i = 1:4
    phi = linspace(%pi/4+(i-1)*%pi/2,%pi/4+i*%pi/2,Np) ;
    for j = 1:2
        theta = linspace((j-1)*%pi/2,j*%pi/2,Nt) ;
        [xf,yf,zf] = eval3dp(orbitale,phi,theta) ;
        if modulo(i+j,2) == 1 then, c = c1 ; else, c = c2 ; end
        plot3d(xf, yf, zf, theta = 78, alpha = 78, flag = [c,6,0]) ;
    end
end
xtitle("Orbitale n = 4, l = 3, m = -2", "x", "y", "z") ;
drawnow() ;
```

Mathématique & Informatique

Atome d'hydrogène

$$\psi_{4,3,-2} = \frac{1}{1024\sqrt{6\pi}\, a_0^{9/2}} r^3 \sin^2\theta \cos\theta\, e^{-\frac{r}{4a_0}}\, e^{-2i\varphi}$$

Orbitale n = 4, l = 3, m = -2



ANNEXES

Mathématique & Informatique

abs	valeur absolue ou module	abs(-1) = 1 ; abs(3 + 4*%i) = 5
max	maximum	max(1, -1, 2, -3) = 2
min	minimum	min(1, -1, 2, -3) = -3
floor	partie entière inférieure	floor(4.5) = 4 ; floor(-4.5) = -5
ceil	partie entière supérieure	ceil(4.5) = 5 ; ceil(-4.5) = -4
int	arrondi entier vers zéro	int(4.5) = 4 ; int(-4.5) = -4
round	arrondi entier le plus proche	round(4.5) = 5 ; round(-4.5) = -5
modulo	reste de la division entière	modulo(23, 7) = 2
factorial	factorielle	factorial(5) = 120
sqrt	racine carrée	sqrt(2) = 1.4142136
exp	exponentielle	exp(-1) = 0.3678794
log	logarithme népérien	log(%e) = 1
log10	logarithme décimal	log10(10) = 1
sin	sinus	sin(%pi/2) = 1
cos	cosinus	cos(0) = 1
tan	tangente	tan(%pi/4) = 1
cotg	cotangente	cotg(%pi/4) = 1
asin	arc sinus	asin(1) = 1.5707963
acos	arc cosinus	acos(0) = 1.5707963
atan	arc tangente	atan(%inf) = 1.5707963
acot	arc cotangente	acot(0) = 1.5707963
sinh	sinus hyperbolique	$\sinh(x) = (e^x - e^{-x})/2$
cosh	cosinus hyperbolique	$\cosh(x) = (e^x + e^{-x})/2$
tanh	tangente hyperbolique	$\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$
coth	cotangente hyperbolique	$\coth(x) = (e^x + e^{-x})/(e^x - e^{-x})$
asinh	argument sinus hyperbolique	$\operatorname{asinh}(x) = \log(x + \sqrt{x^2 + 1})$
acosh	argument cosinus hyperbolique	$\operatorname{acosh}(x) = \log(x + \sqrt{x^2 - 1})$
atanh	argument tangente hyperbolique	$\operatorname{atanh}(x) = \log(\sqrt{(1+x)/(1-x)})$
acoth	argument cotangente hyperbolique	$\operatorname{acoth}(x) = \log(\sqrt{(x+1)/(x-1)})$
gamma	fonction gamma	$\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t}\, dt$
erf	fonction d'erreur	$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$

Les constantes prédéfinies : %pi = pi ($\pi = 3.14159$) ; %e = exponentielle ($e = 2.71828$) ; %inf = infini (∞) %i = imaginaire ($i = \sqrt{-1}$) ; %t, %f = true, false (vrai ≡ 1 ; faux ≡ 0) ; %eps = précision ($\varepsilon = 2.220\ 10^{-16}$)

Mathématique & Informatique

Graphics : bibliothèque graphique

DESSINS 2D

plot2d — représentation de courbes dans le plan
plot2d2 — représentation d'une fonction constante par morceaux
plot2d3 — représentation de courbes dans le plan sous forme de barres verticales
plot2d4 — représentation de courbes dans le plan sous forme de flèches
fplot2d — représentation d'une courbe définie par une fonction
champ — champ de vecteur 2D
champ1 — champ de vecteur 2D Flèches colorées
fchamp — champ défini par une équation différentielle du premier ordre
contour2d — courbes de niveau d'une surface
fcontour2d — courbes de niveau d'une surface (définie par une fonction)
grayplot — représentation d'une surface en 2D sous forme de couleurs
fgrayplot — représentation d'une surface (définie par une fonction) en 2D sous forme de couleurs
Sgrayplot — représentation d'une surface en 2D sous forme de couleurs interpolées
Sfgrayplot — représentation d'une surface (définie par une fonction) en 2D sous forme de couleurs interpolées
xgrid — ajoute une grille sur un dessin 2D
errbar — ajoute des barres d'erreur sur un dessin 2D
histplot — dessine un histogramme
pie — dessine des secteurs circulaires
Matplot — dessin en 2D d'une matrice en pseudo-couleurs

DESSINS 3D

plot3d — représentation en couleurs d'une surface
plot3d1 — représentation en couleurs d'une surface
fplot3d — représente une surface non paramétrique définie par une fonction
fplot3d1 — représente une surface non paramétrique définie par une fonction
param3d — représente une courbe paramétrique en 3D
param3d1 — représente des courbes paramétriques en 3D
contour — courbes de niveau sur une surface en 3D
fcontour — courbes de niveau sur une surface en 3D définie par une fonction
hist3d — représentation d'un histogramme en 3D
genfac3d — calcule les facettes d'une surface non paramétrique
eval3dp — calcule les facettes d'une surface paramétrique
geom3d — projection 3D vers 2D après un dessin 3D

DESSINS DE POLYGONES ET DE LIGNES

xpoly — dessine une ligne brisée ou un polygone
xpolys — dessine un ensemble de lignes brisées ou de polygones
xrpoly — dessine un polygone régulier
xsegs — dessine des segments déconnectés
xfpoly — remplit un polygone
xfpolys — remplit un ensemble de polygones

DESSINS DE RECTANGLES

xrect — dessine un rectangle
xfrect — remplit un rectangle
xrects — dessine ou remplit un ensemble de rectangles

Mathématique & Informatique

DESSINS DES ARCS

xarc — dessine un arc d'ellipse
xarcs — dessine des arcs d'ellipses
xfarc — remplit un secteur angulaire d'ellipse
xfarcs — remplit des secteurs angulaires d'ellipses

DESSINS DE FLÈCHES

xarrows — dessine un ensemble de flèches

DESSINS DE TEXTES

xstring — dessine des chaînes de caractères
xstringl — calcule une boîte contenant des chaînes de caractères
xstringb — dessine des chaînes de caractères dans une boîte
xtitle — ajoute des titres et légendes sur une fenêtre graphique
titlepage — ajoute un titre au milieu d'une fenêtre graphique
xinfo — affiche une chaîne de caractères dans la sous-fenêtre des messages

CADRES ET AXES

xaxis — dessine un axe
graduate — graduations simplifiées
plotframe — dessine un cadre avec axes et mise à l'échelle

TRANSFORMATIONS DE COORDONNÉES

isoview — échelle isométrique (fonction obsolète)
square — échelle isométrique (par changement de la taille de la fenêtre)
scaling — transformation affine d'un ensemble de points
rotate — rotation d'un ensemble de points
xsetech — choix d'une sous-fenêtre graphique
subplot — divise la fenêtre graphique en sous-fenêtres
xgetech — donne l'échelle graphique courante
xchange — transformation de coordonnées réelles vers pixels

COULEURS

colormap — utilisation des tables de couleurs
getcolor — boîte de dialogue pour sélectionner des couleurs dans la table des couleurs
addcolor — ajout de nouvelles couleurs à la table des couleurs
graycolormap — table de couleurs du noir au blanc
hotcolormap — table de couleurs "chaudes" du rouge au jaune

CONTEXTE GRAPHIQUE

xset — change des valeurs du contexte graphique
xget — récupère des valeurs du contexte graphique
xlfont — charge une police dans le contexte graphique ou donne la liste des polices
getsymbol — boite de dialogue pour choisir un marqueur (ou symbole)

SAUVEGARDE ET CHARGEMENT

xsave — sauve les graphiques dans un fichier
xload — charge des graphiques sauvés
xbasimp — envoie des graphiques à une imprimante Postscript ou un fichier
xs2fig — envoie des graphiques dans un fichier au format XFig

PRIMITIVES GRAPHIQUES

xbasc — efface une fenêtre graphique et les graphiques enregistrés
xclear — efface une fenêtre graphique
driver — choisit un pilote graphique
xinit — initialisation d'un pilote graphique
xend — termine une session graphique
xbasr — redessine une fenêtre graphique
replot — redessine une fenêtre graphique avec de nouvelles bornes
xpause — suspend l'exécution de Scilab
xselect — met une fenêtre graphique au premier plan
xclip — définit une zone de clipping
xdel — supprime une fenêtre graphique
winsid — renvoie la liste des fenêtres graphiques
xname — change le nom d'une fenêtre graphique

POSITION DE LA SOURIS

xclick — attend un clic de souris
locate — sélection d'un ensemble de points
xgetmouse — renvoie la position courante du pointeur de la souris

ÉDITEUR INTERACTIF

edit_curv — éditeur graphique interactif
gr_menu — éditeur graphique élémentaire
sd2sci — conversion d'objets gr_menu en instructions Scilab
ged — éditeur interactif des propriétés des entités graphiques

FONCTIONS GRAPHIQUES EN AUTOMATIQUE

bocle — diagramme de Bode
gainplot — magnitude plot
nyquist — Nyquist plot
m_circle — M-circle plot
chart — Nichols chart
black — Black's diagram
evans — Evans root locus
sgrid — s-plane grid lignes
plzr — pôle-zéro plot
zgrid — zgrid plot

Mathématique & Informatique

Metanet : graphes et réseaux

add_edge — ajoute une arête ou un arc entre deux sommets
add_edge_data — associates new data fields to the edges data structure of a graph
add_node — ajoute un sommet déconnecté à un graphe
add_node_data — associates new data fields to the nodes data structure of a graph
adj_lists — calcule des listes d'adjacence
arc_graph — graphe avec sommets correspondant aux arcs
arc_number — nombre d'arcs d'un graphe
articul — trouve un ou plusieurs points d'articulation
bandwr — réduction de largeur de bande pour une matrice creuse
best_match — meilleur appariement d'un graphe
chain_struct — structure chaînée à partir d'une liste d'adjacence d'un graphe
check_graph — vérifie un graphe Scilab
circuit — trouve un circuit ou la fonction de rang dans un graphe orienté
con_nodes — ensemble de sommets d'une composante connexe
connex — composante connexe
contract_edge — contracte des arêtes entre deux sommets
convex_hull — enveloppe convexe d'un ensemble de points dans le plan
cycle_basis — base des cycles d'un graphe simple non-orienté
delete_arcs — détruit tous les arcs ou arêtes reliant un ensemble de sommets
delete_edges — deletes all the arcs or edges between a set of nodes
delete_nodes — détruit des sommets
edges_data_structure — description of the data structure representing the edges of a graph
edge_number — nombre d'arêtes d'un graphe
edgedatafields — returns the vector of edge data fields names
edit_graph — graph and network graphical editor
edit_graph_menus — edit_graph menus description
egraphic_data_structure — data structure representing the graphic properties used for edges display
find_path — trouve un chemin entre deux sommets
gen_net — génération aléatoire ou interactive d'un réseau
girth — circonférence d'un graphe orienté
glist — création d'un graphe
graph-list — description d'un graphe (obsolete)
graph_2_mat — matrice d'incidence sommets-arcs ou sommets-sommets d'un graphe
graph_center — centre d'un graphe
graph_complement — complément d'un graphe
graph_data_structure — description of the main graph data structure
graph_diameter — diamètre d'un graphe
graph_power — puissance k-ième d'un 1-graphe orienté
graph_simp — convertit un graphe en un graphe simple non-orienté
graph_sum — somme de deux graphes
graph_union — union de deux graphes
hamilton — circuit hamiltonien d'un graphe
hilite_edges — highlights a set of edges — unhighlights a set of edges
hilite_nodes — highlights a set of nodes — unhighlights a set of nodes
index_from_tail_head — Computes the index of edges given by (tail,head) pairs
is_connex — test de connexité
knapsack — résout un problème du sac à dos 0-1 multiple

Mathématique & Informatique

line_graph — graphe avec sommets correspondant aux arêtes
load_graph — charge un graphe
make_graph — création d'un graphe
mat_2_graph — graphe à partir de matrice d'incidence sommets-arcs ou sommets-sommets
max_cap_path — chemin de capacité maximum
max_clique — clique maximum d'un graphe
max_flow — flot maximum entre deux sommets
mesh2d — triangulation of n points in the plane
metanet_module_path — Returns the path of the metanet module
min_lcost_cflow — flot contraint de coût linéaire minimum
min_lcost_flow1 — flot de coût linéaire minimum
min_lcost_flow2 — flot de coût linéaire minimum
min_qcost_flow — flot de coût quadratique minimum
min_weight_tree — arbre couvrant de poids minimum
neighbors — sommets voisins connectés à un sommet
netclose — closes an edit_graph window
netwindow — selects the current edit_graph window
netwindows — gets the numbers of edit_graph windows
ngraphic_data_structure — data structure representing the graphic properties used for nodes display
nodedatafields — returns the vector of node data fields names
nodes_data_structure — description of the data structure representing the nodes of a graph
node_number — nombre de sommets d'un graphe
nodes_2_path — chemin reliant un ensemble de sommets
nodes_degrees — degrés des sommets d'un graphe
path_2_nodes — ensemble de sommets d'un chemin
perfect_match — appariement parfait de coût minimum
pipe_network — résout le problème du réseau de tuyaux
plot_graph — affichage général d'un graphe
predecessors — sommets origines des arcs entrants d'un sommet donné
qassign — résout un problème d'affectation quadratique
salesman — résout le problème du voyageur de commerce
save_graph — sauve un graphe
set_nodes_id — displays labels near selected nodes in a graph display.
shortest_path — chemin le plus court
show_arcs — highlights a set of arcs
show_edges — highlights a set of edges
show_graph — displays a graph
show_nodes — highlights a set of nodes
split_edge — divise une arête en insérant un sommet
strong_con_nodes — ensemble de sommets d'une composante fortement connexe
strong_connex — composantes fortement connexes
subgraph — sous-graphe d'un graphe
successors — sommets extrémités des arcs sortant d'un sommet donné
supernode — remplace un groupe de sommets par un super-sommet
trans_closure — fermeture transitive
update_graph — converts an old graph data structure to the current one

Mathématique & Informatique

Statistiques

cdfbet — fonction de répartition de la distribution Beta
cdfbin — fonction de répartition de la distribution binomiale
cdfchi — fonction de répartition de la distribution du chi-deux
cdfchn — fonction de répartition de la distribution du chi-deux non centrée
cdff — fonction de répartition de la distribution de Fisher
cdffnc — fonction de répartition de la distribution de Fisher non centrée
cdfgam — fonction de répartition de la distribution gamma
cdfnbn — fonction de répartition de la distribution binomiale négative
cdfnor — fonction de répartition de la distribution normale
cdfpoi — fonction de répartition de la distribution de Poisson
cdft — fonction de répartition de la distribution de Student
center — center
wcenter — center and weight
cmoment — central moments of all orders
correl — correlation of two variables
covar — covariance of two variables
ftest — Fischer ratio
ftuneq — Fischer ratio for samples of unequal size.
geomean — geometric mean
harmean — harmonic mean
iqr — interquartile range
labostat — Statistical toolbox for Scilab
mad — mean absolute deviation
mean — mean (row mean, column mean) of vector/matrix entries
meanf — weighted mean of a vector or a matrix
median — médiane des termes d'une matrice
moment — non central moments of all orders
msd — mean squared deviation
mvvacov — computes variance-covariance matrix
nancumsum — Thos function returns the cumulative sum of the values of a matrix
nand2mean — difference of the means of two independent samples
nanmax — max (ignoring Nan's)
nanmean — mean (ignoring Nan's)
nanmeanf — mean (ignoring Nan's) with a given frequency.
nanmedian — median of the values of a numerical vector or matrix
nanmin — min (ignoring Nan's)
nanstdev — standard deviation (ignoring the NANs).
nansum — Sum of values ignoring NAN's
nfreq — frequence of the values in a vector or matrix
pca — Computes principal components analysis with standardized variables
perctl — computation of percentils
princomp — Principal components analysis
quart — computation of quartiles
regress — regression coefficients of two variables
sample — Sampling with replacement
samplef — sample with replacement from a population and frequences of his values.
samwr — Sampling without replacement
show_pca — Visualization of principal components analysis results
st_deviation — écart-type des termes d'une matrice — écart-type des termes d'une matrice
stdevf — standard deviation
strange — range
tabul — frequency of values of a matrix or vector
thrownan — eliminates nan values
trimmean — trimmed mean of a vector or a matrix
variance — variance of the values of a vector or matrix
variancef — standard deviation of the values of a vector or matrix

Mathématique & Informatique

Traitement du signal

analpf — create analog low-pass filter
bilt — bilinear or biquadratic transform SISO system given by a zero/poles representation
buttmag — response of Butterworth filter
casc — cascade realization of filter from coefficients
cepstrum — cepstrum calculation
cheb1mag — response of Chebyshev type 1 filter
cheb2mag — response of type 2 Chebyshev filter
chepol — Chebychev polynomial
convol — convolution
corr — correlation, covariance
cspect — two sided cross-spectral estimate between 2 discrete time signals using the correlation method
czt — chirp z-transform algorithm
detrend — remove constant, linear or piecewise linear trend from a vector
dft — discrete Fourier transform
ell1mag — magnitude of elliptic filter
eqfir — minimax approximation of FIR filter
eqiir — design of iir filters
faurre — filter computation by simple Faurre algorithm
ffilt — coefficients of FIR low-pass
fft — fast Fourier transform
fft2 — two-dimension fast Fourier transform
fftshift — rearranges the fft output, moving the zero frequency to the center of the spectrum
filt_sinc — samples of sinc function
filter — filters a data sequence using a digital filter
find_freq — parameter compatibility for elliptic filter design
findm — for elliptic filter design
frfit — frequency response fit
frmag — magnitude of FIR and IIR filters
fsfirlin — design of FIR, linear phase filters, frequency sampling technique
group — group delay for digital filter
hank — covariance to hankel matrix
hilb — FIR approximation to a Hilbert transform filter
hilbert — discrete-time analytic signal computation of a real signal using Hilbert transform
iir — iir digital filter
iirgroup — group delay Lp IIR filter optimization
iirlp — Lp IIR filter optimization
intdec — changes sampling rate of a signal
jmat — row or column block permutation
kalm — Kalman update
lattn — recursive solution of normal equations
lattp — lattp
lev — Yule-Walker equations (Levinson's algorithm)
levin — Toeplitz system solver by Levinson algorithm (multidimensional)
lindquist — Lindquist's algorithm
mese — maximum entropy spectral estimation
mfft — multi-dimensional fft
mrfit — frequency response fit

Mathématique & Informatique

%asn — elliptic integral
%k — Jacobi's complete elliptic integral
%sn — Jacobi 's elliptic function
phc — Markovian representation
pspect — two sided cross-spectral estimate between 2 discrete time signals
remez — algorithm for the weighted chebyshev approximation of a continuous function with a sum of cosines
remezb — minimax approximation of magnitude response
rpem — recursive prediction error minimization estimation
sincd — digital sinc function or Dirichlet kernel
srfaur — square-root algorithm
srkf — square root Kalman filter
sskf — steady-state Kalman filter
syredi — design of iir filters, syredi code interface
system — observation update
trans — low-pass to other filter transform
wfir — linear-phase FIR filters
wiener — Wiener estimate
wigner — 'time-frequency' wigner spectrum
window — compute symmetric window of various type
yulewalk — least-square filter design
zpbutt — Butterworth analog filter
zpch1 — Chebyshev analog filter
zpch2 — Chebyshev analog filter
zpell — lowpass elliptic filter

Mathématique & Informatique

DOCUMENTATION SUR INTERNET

- Le manuel de référence Scilab en français (1132 pages au format PDF) :
http://www.scilab.org/download/5.1.1/manual_scilab-5.1.1_fr_FR.pdf

- Un cours "Une introduction à Scilab" (Bruno Pinçon) :
http://www.iecn.u-nancy.fr/~pincon/scilab/doc.pdf

- Un cours "Une introduction à Matlab" (Serge Bedwani) :
http://www.bedwani.ch/download/doc/pdf/matlab.pdf

- L'encyclopédie collective WIKIPEDIA via son portail Mathématiques :
http://fr.wikipedia.org/wiki/Portail:Math%C3%A9matiques

- Un site (Serge Melh) traitant des mathématiques en général (histoire, concepts, exemples illustrés) :
http://www.chronomath.com/

- Un site (Robert Ferréol) recensant les courbes et surfaces usuelles en 2D et 3D (équations, animations) :
http://www.mathcurve.com/

- Une liste de récréations informatiques & mathématiques :
http://web.me.com/rouxjeanbernard/Site/AM/html/am.html#amch25.html

- Un ensemble de programmes en ligne, pour décoder des textes, trouver la solution de jeux :
http://www.dcode.fr/

- Un document (Bruno Garcia) sur les définitions et les principaux algorithmes de la théorie des graphes :
http://www.bruno-garcia.net/www/polyro/polyro.html

- Un site regroupant des notions essentielles sur de nombreux sujets des sciences de l'ingénieur :
http://www.sciences.ch/htmlfr/accueil.php

- Un document (Serge Dégerine) sur les Calculs Financiers et les Statistiques, issu d'un cours de MIAGE :
http://ljk.imag.fr/membres/Serge.Degerine/Enseignement/CFS.pdf

- Un site expert des Ecoles Normales Supérieures contenant différents articles sur les Mathématiques :
http://www.dma.ens.fr/culturemath/index.html

- Un site (Eric Weisstein) constituant une documentation régulièrement mise à jour sur les Mathématiques :
http://mathworld.wolfram.com/

- Des simulations sous Java et un cours (Jean-Jacques Rousseau) sur différents sujets de la Physique :
http://subaru2.univ-lemans.fr/enseignements/physique/02/index.html

- Une bibliothèque des mathématiques (définitions, biographies, actualités, ...) :
http://www.bibmath.net/

- Des pages Web (Geoffrey Bruno, Loïc Devilliers) sur les fractales dans la nature et les technologies humaines :
http://geoffreyhistoire.pagesperso-orange.fr/fractales/biographies.html :

FIN