
NUMERICAL ANALYSIS USING SCILAB: SOLVING NONLINEAR EQUATIONS

In this tutorial we provide a collection of numerical methods for solving nonlinear equations using Scilab.

Level



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.



Step 1: The purpose of this tutorial

The purpose of this Scilab tutorial is to provide a collection of numerical methods for finding the zeros of scalar nonlinear functions. The methods that we present are:

- Bisection;
- Secant;
- Newton-Raphson;
- Fixed point iteration method.

These classical methods are typical topics of a numerical analysis course at university level.

An introduction to

NUMERICAL ANALYSIS USING SCILAB

solving nonlinear equations

Step 2: Roadmap

This tutorial is composed of two main parts: the first one (Steps 3-10) contains an introduction about the problem of solving nonlinear equations, presents some solution strategies and introduces properties and issues of such problems and solutions. The second part (Steps 11-23) is dedicated to the specific methods, equipped with many Scilab examples.

Descriptions	Steps
Introduction and solution strategies	3-6
Conditioning and convergence	7-10
Bisection method	11-12
Secant method	13-14
Newton method	15-18
Fixed point iteration method	19-22
Conclusions and remarks	23-25

Step 3: Introduction

Many problems that arise in different areas of engineering lead to the solution of scalar nonlinear equations of the form

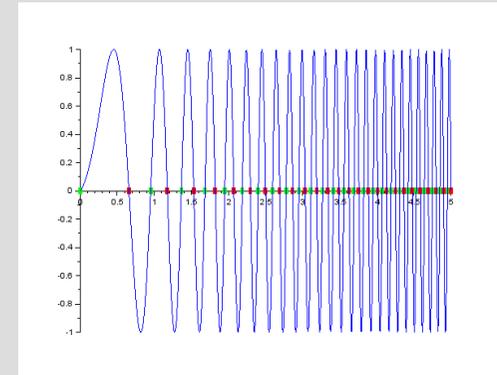
$$f(x) = 0$$

i.e. to find a zero of a nonlinear function.

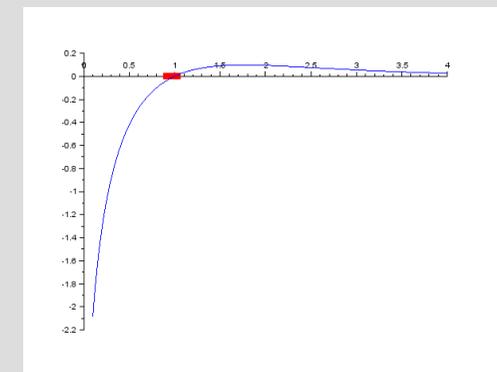
Nonlinear equations can have none, one, two, or an infinite number of solutions. Some examples are presented on the right.

Note: A special class of nonlinear equations is constituted by polynomials of the form

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 = 0.$$



(Linear chirp function $y = \sin(\phi_0 + 2\pi(f_0 t + \frac{k}{2} t^2))$ with infinite zeros)



(Function $y = \log(x) \exp(-x)$ with one zero)

The code of the examples is available in the file [ex1.sce](#)

Step 4: Solution strategies

Many solution methods exist and the correct choice depends on the type of function f . For example, different methods are used whether f is a polynomial or it is a continuous function whose derivatives are not available.

Moreover, the problem can be stated in equivalent formulations. For example, the original formulation $f(x) = 0$ can be converted into a fixed point formulation of the form $x = g(x)$ or into a minimization problem of the form $\min_x f^2(x)$.

It is important to note that even if these formulations are mathematically equivalent (their zeros are the same ones), the numerical methods used to approximate the solution do not have all the same behavior.

Hence, the numerical solution strategy should take into account the kind of problem we try to solve.

Example of equivalent formulations:

Original problem:

$$f(x) = x^2 - a = 0$$

Examples of fixed point formulation:

$$x = \frac{a}{x}$$

or

$$x = \frac{1}{2} \left(x + \frac{a}{x} \right)$$

Example of minimization formulation:

$$\min_x (x^2 - a)^2$$

Step 5: Graphical interpretation and separation of zeros

The first step of many numerical methods for solving nonlinear equations is to identify a starting point or an interval where to search a single zero: this is called “separation of zeros”. If no other information is available, this can be done by evaluating the function f at several values x_i and plotting the results $f(x_i)$.

Solving the problem $f(x) = 0$ is equivalent to find the solutions of the following system

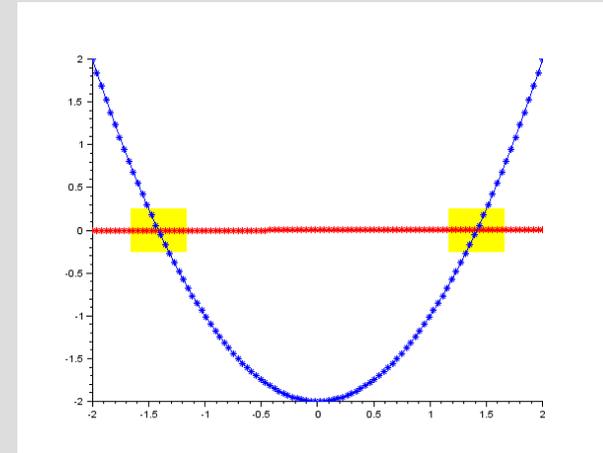
$$\begin{cases} y = f(x) \\ y = 0 \end{cases}$$

i.e., graphically, to determine, in a Cartesian plane, the intersections of the graph of the function $y = f(x)$ with the x -axis.

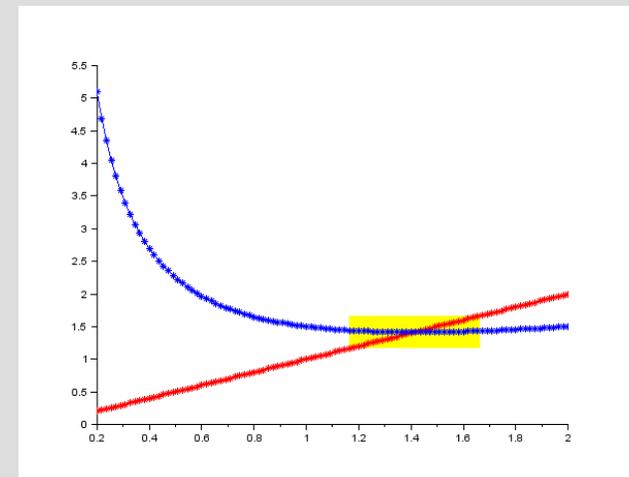
In the case of fixed point formulation $x = g(x)$ its graphical formulation is related to the system

$$\begin{cases} y = g(x) \\ y = x \end{cases}$$

i.e. the solutions are given by the intersections of the function $y = f(x)$ with the bisector $y = x$.



(Separation of zeros of the original problem: $x^2 - 2 = 0$)



(Fixed point equivalent formulation: $x = \frac{1}{2} \left(x + \frac{2}{x} \right)$)

The code of the example is available in the file [ex2.sce](#)

Step 6: Example of a bracketing strategy

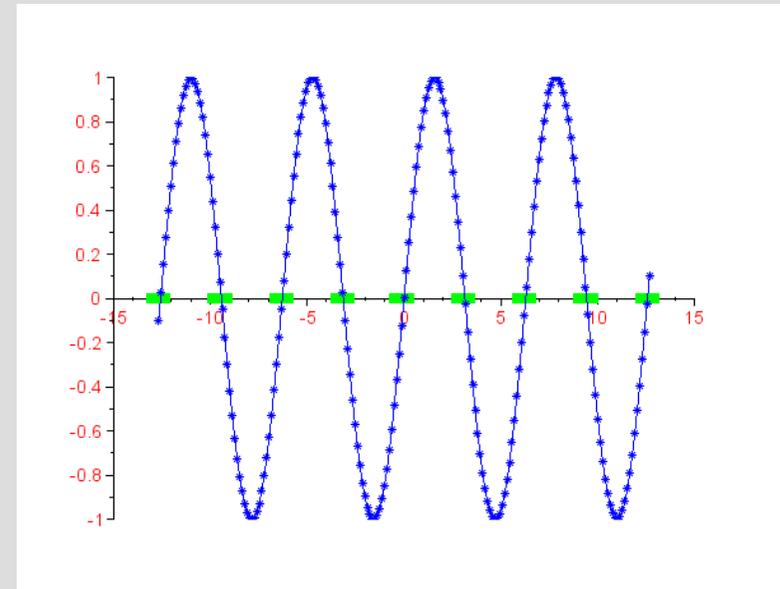
Bracketing is an automatic strategy for finding intervals containing a zero of a given function f . An example of bracketing is given in the following lines of code; the idea is to identify the points in which the function changes sign:

```
function xsol=fintsearch(f, xmin, xmax, neval)
// Generate x vector
x = linspace(xmin, xmax, neval)';
// Evaluate function
y = f(x);

// Check for zeros
indz = find(abs(y)<=1000*%eps);
y(indz) = 0;

// Compute signs
s = sign(y);
// Find where f changes sign
inds = find(diff(s)~=0);
// Compute intervals
xsol = [x(inds),x(inds+1)];
endfunction
```

The code is also available in the file [fintsearch.sci](#), while the example can be found in [fintsearch_test.sce](#).



(Separation of zeros for the function $f(x) = \sin(x)$)

Step 7: Conditioning of a zero-finding problem

The conditioning of a zero-finding problem is a measure of how it is sensitive to perturbations of the equation.

Here we denote by α a zero of the function $f(x)$, i.e. $f(\alpha) = 0$.

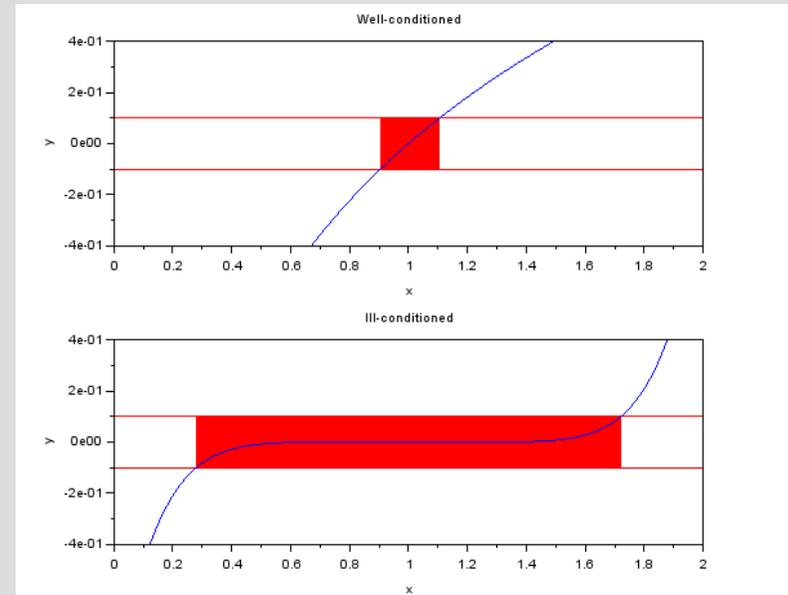
From the first figure on the right we can intuitively see that if the derivative $|f'(\alpha)|$ is “large” the problem is well-conditioned. In this case we can clearly identify the zero of f , even if there are rounding errors. Conversely, if the derivative is “small”, the zero-finding problem is said to be ill-conditioned and there is no clear identification of the zero. In this case, if rounding errors are present, the zero is spread up over a “large” interval of uncertainty.

In summary, we can state the following:

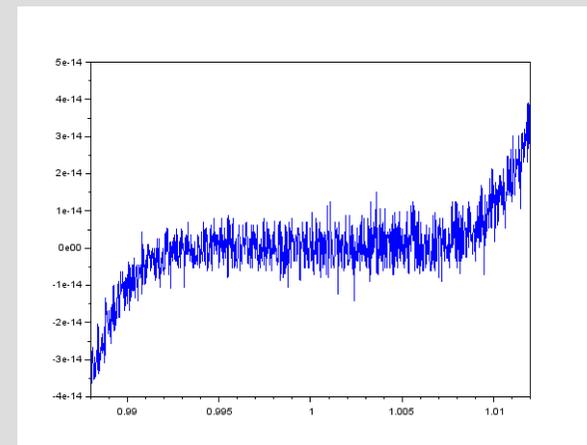
- The conditioning number of the root finding problem is $1/|f'(\alpha)|$;
- The problem is ill-conditioned if $1/|f'(\alpha)|$ is large, i.e. $|f'(\alpha)|$ is small.

In the graphic on the right down we can see that the zero $x^* = 1$ of $p(x) = x^7 - x^6 + x^5 - x^4 + x^3 - x^2 + x - 1$ can not be identified because of ill-conditioning.

The code of the example is available in the file [ex3.sce](#)



(Example of well- and ill-conditioned root finding problem)



(Given a very ill-conditioned problem, the unique zero cannot be identified)

Estimating the conditioning of the problem of finding a (single) zero α of a (continuously differentiable) function $f(x)$ means to provide an estimate of the relative error of a perturbed solution.

Finding the zero α of a function $f(x)$, i.e. $f(\alpha) = 0$, is equivalent (for continuously differentiable functions) to solving the inverse problem $\alpha = f^{-1}(0)$.

If we consider a perturbed solution $\tilde{\alpha}$, i.e. $f(\tilde{\alpha}) = d$ or, equivalently, $\tilde{\alpha} = f^{-1}(d)$, making an error d on its evaluation, we have the following error:

$$\Delta\alpha = \tilde{\alpha} - \alpha = f^{-1}(d) - f^{-1}(0)$$

Using the following Taylor expansion

$$f^{-1}(d) \approx f^{-1}(0) + (f^{-1}(0))' \cdot (d - 0) + \dots$$

and the relation obtained on the right

$$(f^{-1}(0))' = \frac{1}{f'(\alpha)}$$

the error can be written as

$$\Delta\alpha \approx f^{-1}(0) + (f^{-1}(0))' \cdot d - f^{-1}(0) = (f^{-1}(0))' \cdot d = \frac{d}{f'(\alpha)}$$

Hence, the relative error can be stated as

$$\left| \frac{\Delta\alpha}{\alpha} \right| = \left| \frac{1}{f'(\alpha)} \right| \cdot \left| \frac{d}{\alpha} \right|$$

The code of the example on the right is available in the file [ex4.sce](#)

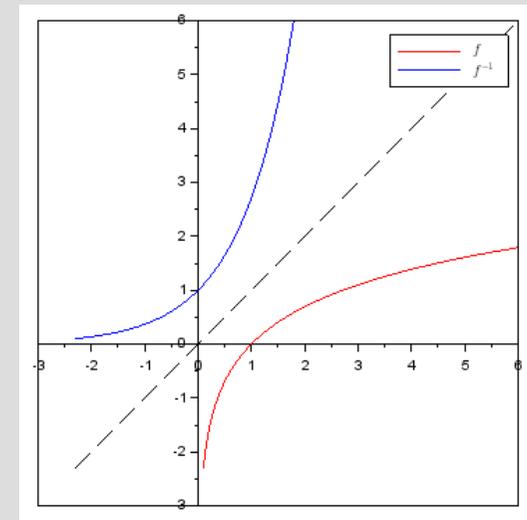
Original problem:

Find α such that $f(\alpha) = 0$ i.e. $\alpha = f^{-1}(0)$

Perturbed problem:

Find $\tilde{\alpha}$ such that $f(\tilde{\alpha}) = d$ i.e. $\tilde{\alpha} = f^{-1}(d)$

Inverse function:



(Example of inverse function)

The function and its inverse are related from the relation

$$f^{-1}(f(x)) = f(f^{-1}(x)) = x$$

and the derivative of the inverse function satisfies the relation

$$(f^{-1}(y))' \cdot f'(x) = 1 \quad \text{i.e.} \quad (f^{-1}(y))' = \frac{1}{f'(x)} \quad (\text{where } y = f(x))$$

Step 8: Convergence rates of iterative methods

Typically, methods for approximating nonlinear equations are based on iterative strategies, *i.e.* starting from an initial guess solution x_0 , and computing a sequence of solutions $\{x_i\}$ that converge to the desired solution x^* (where $f(x^*) = 0$), *i.e.*

$$x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \dots \rightarrow x_i \rightarrow \dots \rightarrow x^*$$

We define the rate of convergence p of the sequence as

$$\lim_{n \rightarrow \infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^p} = \lim_{n \rightarrow \infty} \frac{|\epsilon_{n+1}|}{|\epsilon_n|^p} = C$$

for some constant $C > 0$ and $p \geq 1$. C is called the asymptotic error constant while ϵ_n is the error at the n^{th} iteration.

If $p = 1$ and $C < 1$ the convergence is called **linear**. We require $C < 1$ to ensure the convergence of the method indeed the error must be reduced at each iteration as explained by this relation:

$$|\epsilon_{n+1}| \leq C|\epsilon_n| < C \cdot (C|\epsilon_{n-1}|) = C^2|\epsilon_{n-1}| < \dots < C^{n+1}|\epsilon_0|$$

Here we compare the $n+1$ -th step error with the initial error.

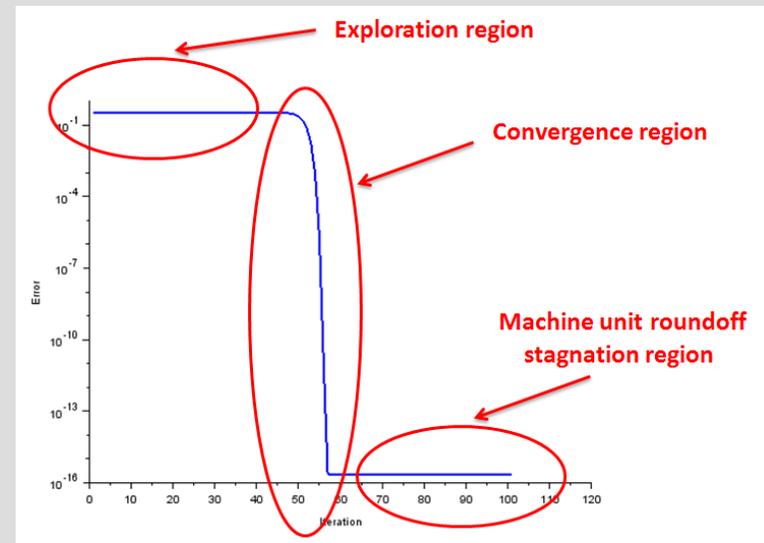
If $1 < p < 2$ the convergence is called **superlinear**.

If $p = 2$ the convergence is called **quadratic**.

The following figure shows a typical convergence rate profile where we can identify three different regions:

- an exploration region: in this region there is an exploration of the solution space trying to find an initial guess solution (starting point) where convergence properties are guaranteed, and, moreover, there is no significant reduction of the error;
- a convergence region: the basin of attraction of the solution;
- a stagnation region: this last region is due to round-off errors of the floating point system that are unavoidable.

This figure stresses the fact that the definition of the convergence rate is valid only “in the convergence region”, hence it is a local definition.



(Typical behavior of a convergence rate profile)

Step 9: Examples of convergence rates

Let us suppose we are looking for the zero $x^* = 1$.

Linear rate:

Consider the following error model:

$$\epsilon_0 = 1 \text{ and } \epsilon_{n+1} = \frac{1}{10} \epsilon_n + eps$$

In this case we get the following errors:

$\epsilon_0 = 1$	zero significant figures
$\epsilon_1 = 0.1$	one significant figures
$\epsilon_2 = 0.01$	two significant figures
$\epsilon_3 = 0.001$	three significant figures

With a linear rate of convergence, the number of significant figures the method gains is constant at each step (a multiple of the iteration number).

Quadratic rate:

Consider the following error model:

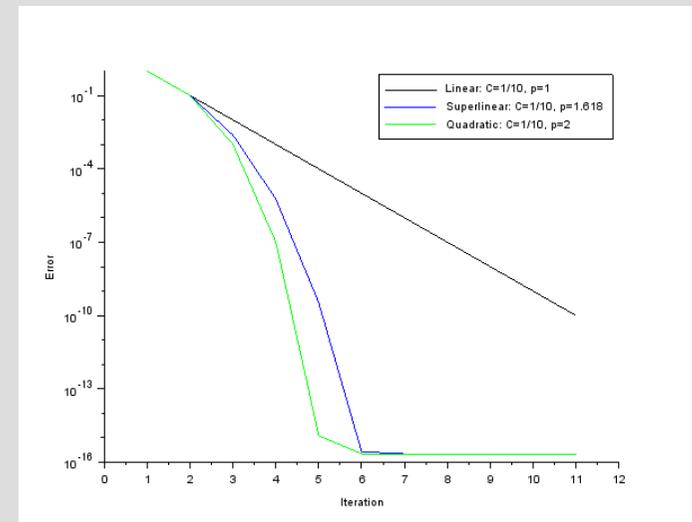
$$\epsilon_0 = 1 \text{ and } \epsilon_{n+1} = \frac{1}{10} \epsilon_n^2 + eps$$

In this case we get the following errors:

$\epsilon_0 = 1$	zero significant figures
$\epsilon_1 = 0.1$	one significant figures (one figure gained)
$\epsilon_2 = 0.001$	three significant figures (two figures gained)
$\epsilon_3 = 0.0000001$	seven significant figures (four figures gained)

With a quadratic rate of convergence, the number of significant figures the method gains at each iteration is twice the previous iteration.

A comparison of the typical rate of convergence (when rounding errors are present) is shown in the following figure:



(Comparison between linear, superlinear and quadratic rate of convergence)

The number of figures gained per iteration can be summarized in the following table:

Convergence rate	Figures gained per iteration
Linear	Constant
Superlinear	Increasing
Quadratic	Double

The code of the example is available in the file [ex5.sce](#)

Step 10: Convergence criteria

When we approximate a solution with an iterative method it is necessary to choose how to properly stop the algorithm and hence provide the solution. As each evaluation of the function can be computationally expensive, it is important to avoid unnecessary evaluations (for instance, avoiding evaluations in the stagnation region).

The convergence criteria reported on the right refer to the following problem:

- Find a solution x^* such that $f(x^*) = 0$ starting from an initial guess $(x_0, f(x_0))$, with x_0 in $\Delta = b - a$.

The design criteria are based on the absolute or relative error for the variable x or for the value of the function f . The difference between a criterion based on x or f depends on the conditioning of the nonlinear problem, while a choice based on the absolute or relative error depends on the scaling of the nonlinear equations.

In our example, we consider the relative errors for f and x they are adimensional, *i.e.* they allow to avoid multiplicative constants.

Example of convergence criteria:

- Absolute error between two iterates on x :
 $|x_k - x_{k-1}| \leq \delta_x$
- Relative error between two iterates on x :
 $|x_k - x_{k-1}| \leq \delta_x \cdot \Delta$
- Absolute residual on f :
 $|f(x_k)| \leq \delta_f$
- Relative residual on f :
 $|f(x_k)| \leq \delta_f \cdot f(x_0)$

Example of implementation of a stopping criterion:

```
// Check for convergence
if (abs(fxnew)/fref < ftol) | (abs(dx)/xref < xtol)
    // The root is found
    x = xnew;
    fx = fxnew;
end
```

The code checks the convergence both on x and f . If we are dealing with an ill-conditioned problem it is likely that x will not converge, so the check on f will stop the iterative method.

Step 11: Bisection method

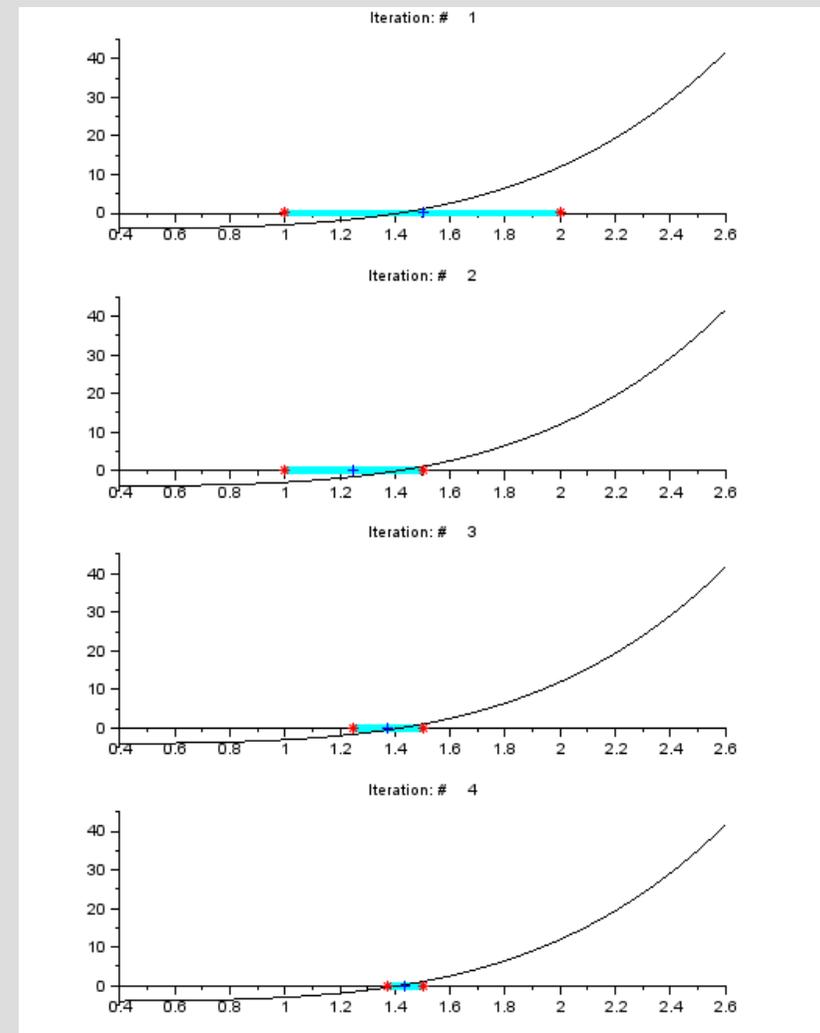
Supposing we are looking for a zero of a continuous function, this method starts from an interval $[a,b]$ containing the solution and then evaluates the function at the midpoint $m=(a+b)/2$. Then, according to the sign of the function, it moves to the subinterval $[a,m]$ or $[m,b]$ containing the solution and it repeats the procedure until convergence.

The main pseudo-code of the algorithm is the following:

```
Algorithm pseudo-code  
while ((b - a) > tol) do  
    m = (a + b)/2  
    if sign(f(a)) = sign(f(m)) then  
        a = m  
    else  
        b = m  
    end  
end1
```

The figure on the right refers to the first 4 iterations of the bisection method applied to the function $f(x) = x^4 - 4$ in the interval $[1,2]$. The method starts from the initial interval $[a,b]=[1,2]$ and evaluates the function at the midpoint $m=1.5$. Since the sign of the function in $m=1.5$ is equal to the sign of the function in $b=2$, the method moves to the interval $[a,m]=[1,1.5]$, which contains the zero. At the second step, it starts from the interval $[a,b]=[1,1.5]$, it evaluates the function at the midpoint $m=1.25$ and it moves to the interval $[1.25, 1.5]$. And so on.

The function is available in the file [bisection.sci](#), while the example can be found in [bisection_test.sce](#).



(Example of the first four iterations of the bisection method)

Step 12: Convergence of the bisection method

At each iteration of the method the searching interval is halved (and contains the zero), i.e.

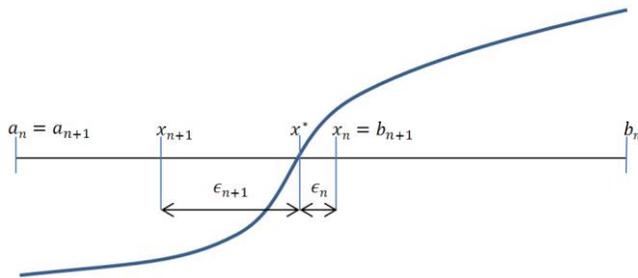
$$(b_n - a_n) = \frac{b_{n-1} - a_{n-1}}{2} = \frac{b_{n-2} - a_{n-2}}{4} = \dots = \frac{1}{2^n} (b_0 - a_0)$$

Hence, the absolute error at the nth iteration is

$$|\epsilon_n| = |x_n - x^*| \leq \frac{(b_n - a_n)}{2} = \frac{1}{2^{n+1}} (b_0 - a_0)$$

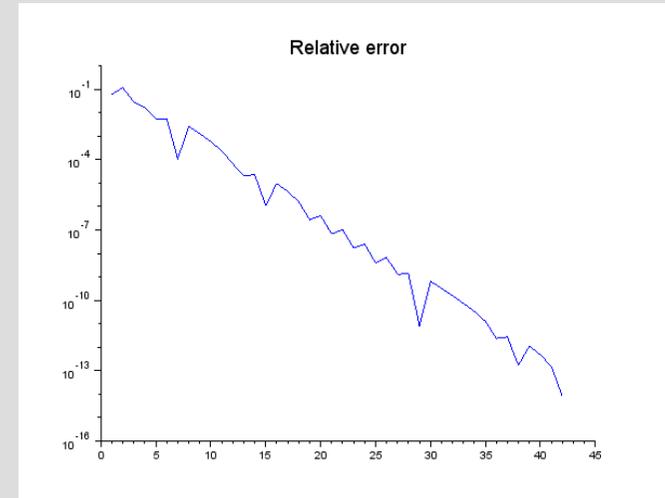
and the converge $|\epsilon_n| \rightarrow 0$ is guaranteed for $n \rightarrow \infty$.

Observe that at each iteration the interval is halved, i.e. $(b_n - a_n) = \frac{(b_{n-1} - a_{n-1})}{2}$, but this relation does not guarantee that $|\epsilon_{n+1}| < |\epsilon_n|$ (i.e. monotone convergence to x^*) as explained in the figure below.



However, we define the rate of the convergence for this method linear.

The figure on the right shows the relative error related to the iterations (reported in the table below) of the method applied to the function $f(x) = x^4 - 4$ in the interval $[1,2]$ where the analytic solution is $\sqrt{2}$. As expected, the method gains 1 significant figure every 3/4 iterations.



(Relative error of the bisection method)

iter	xm	f(xm)
1	1.5000000000000000e+000	1.0625000000000000e+000
2	1.2500000000000000e+000	-1.5585937500000000e+000
3	1.3750000000000000e+000	-4.2553710937500000e-001
4	1.4375000000000000e+000	2.700347900390625e-001
5	1.4062500000000000e+000	-8.933925628662109e-002
6	1.4218750000000000e+000	8.738619089126587e-002
7	1.4140625000000000e+000	-1.708801835775375e-003
8	1.4179687500000000e+000	4.265461512841284e-002
9	1.4160156250000000e+000	2.042701352911536e-002
10	1.4150390625000000e+000	9.347648389848473e-003
11	1.4145507812500000e+000	3.816560889447374e-003
12	1.4143066406250000e+000	1.053164176941124e-003
13	1.4141845703125000e+000	-3.279976360204273e-004
14	1.4142456054687500e+000	3.625385649508317e-004
15	1.4142150878906250e+000	1.725928857077008e-005
16	1.414199829101563e+000	-1.553719676383736e-004
17	1.414207458496094e+000	-6.905703801995955e-005
18	1.414211273193359e+000	-2.589904934735543e-005
19	1.414213180541992e+000	-4.319924044260404e-006

(Iterations of the bisection method)

Step 13: Secant method

Supposing we are looking for a zero of a continuous function, this method starts from two approximations $(x_0, f(x_0))$ and $(x_1, f(x_1))$ of the unknown zero $(x^*, f(x^*) = 0)$ and computes the new approximation x_2 as the zero of the straight line passing through the two given points. Hence, x_2 can be obtained solving the following system:

$$\begin{cases} \frac{y - f(x_0)}{f(x_1) - f(x_0)} = \frac{x - x_0}{x_1 - x_0} \\ y = 0 \end{cases}$$

giving

$$(x_2 =) x = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)}$$

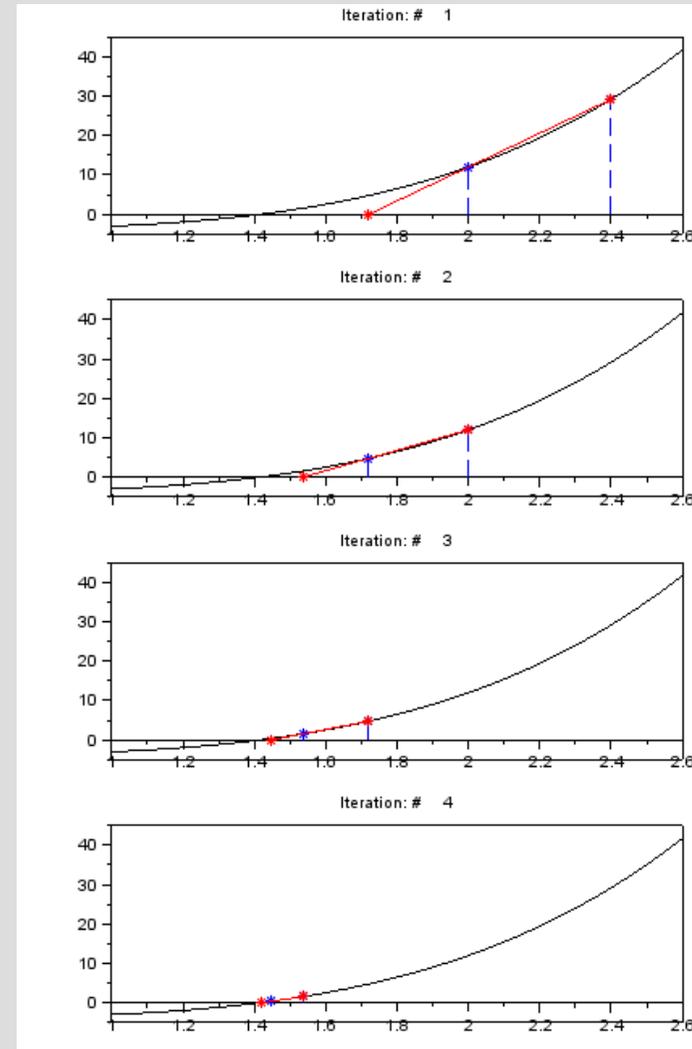
Once the new approximation is computed, we repeat the same procedure with the new initial points $(x_1, f(x_1))$ and $(x_2, f(x_2))$.

The iterative formula is

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$

The figures on the right refer to the first four iterations of the method for the function $f(x) = x^4 - 4$ with initial guess values $x_0 = 2.4$ and $x_1 = 2$.

The function is available in the file [secant.sci](#), while the example can be found in [secant_test.sce](#).



(First four iterations of the secant method)

Step 14: Convergence of the secant method

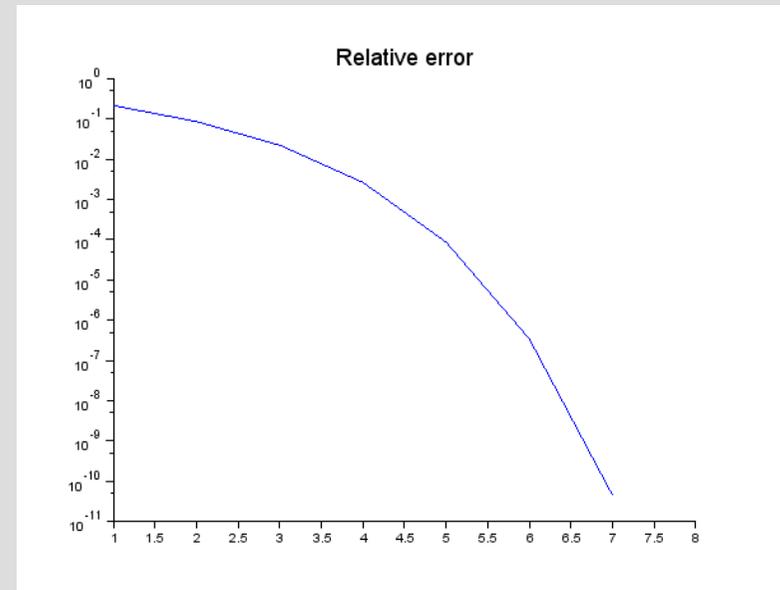
The main pseudo-code of the algorithm is the following:

```
Algorithm
xkm1 = x0; fkm1 = f(x0)      // Step: k-1
xk = x1; fk = f(x1)         // Step: k
xkp1 = xk                    // Initialization
iter = 1                     // Current iteration
while iter <= itermax do
    iter = iter+1
    xkp1 = xk - (fk*(xk-xkm1))/(fk-fkm1)
    if abs(xkp1-xk) < tol break // Converg. test
    xkm1 = xk; fkm1 = fk
    xk = xkp1; fk = f(xkp1)
end
```

The algorithm iterates until convergence or until the maximum number of iterations is reached. At each iteration only one function evaluation is required. The “break” statement terminates the execution of the while loop.

For the secant method it is possible to prove the following result: if the function f is continuous with continuous derivatives until order 2 near the zero, the zero is simple (has multiplicity 1) and the initial guesses x_0 and x_1 are picked in a neighborhood of the zero, then the method converges and the convergence rate is equal to $p = \frac{1+\sqrt{5}}{2}$ (superlinear).

The figure on the right shows the relative error related to the iterations (reported in the table below) of the method applied to the function $f(x) = x^4 - 4$ in the interval $[1,2]$, where the analytic solution is $\sqrt{2}$.



(Relative error of the secant method)

iter	x	f(x)
1	1.720566318926975e+000	4.763662991752302e+000
2	1.536615808653681e+000	1.575209448553538e+000
3	1.445737714328982e+000	3.687585363051467e-001
4	1.417960311448227e+000	4.255838213105800e-002
5	1.414336274760076e+000	1.388512885929671e-003
6	1.414214048925191e+000	5.504711427128939e-006
7	1.414213562436418e+000	7.164207005416756e-010
8	1.414213562373095e+000	8.881784197001252e-016

(Iterations of the secant method)

Step 15: Newton method

Supposing we are looking for a zero of a continuous function with continuous derivatives, this method starts from an approximation x_0 of the unknown zero x^* and computes the new approximation x_1 as the zero of the straight line passing through the initial point and tangent to the function. Hence, x_1 can be obtained solving the following system:

$$\begin{cases} y - f(x_0) = m(x - x_0) & \text{with } m = f'(x_0) \\ y = 0 \end{cases}$$

giving

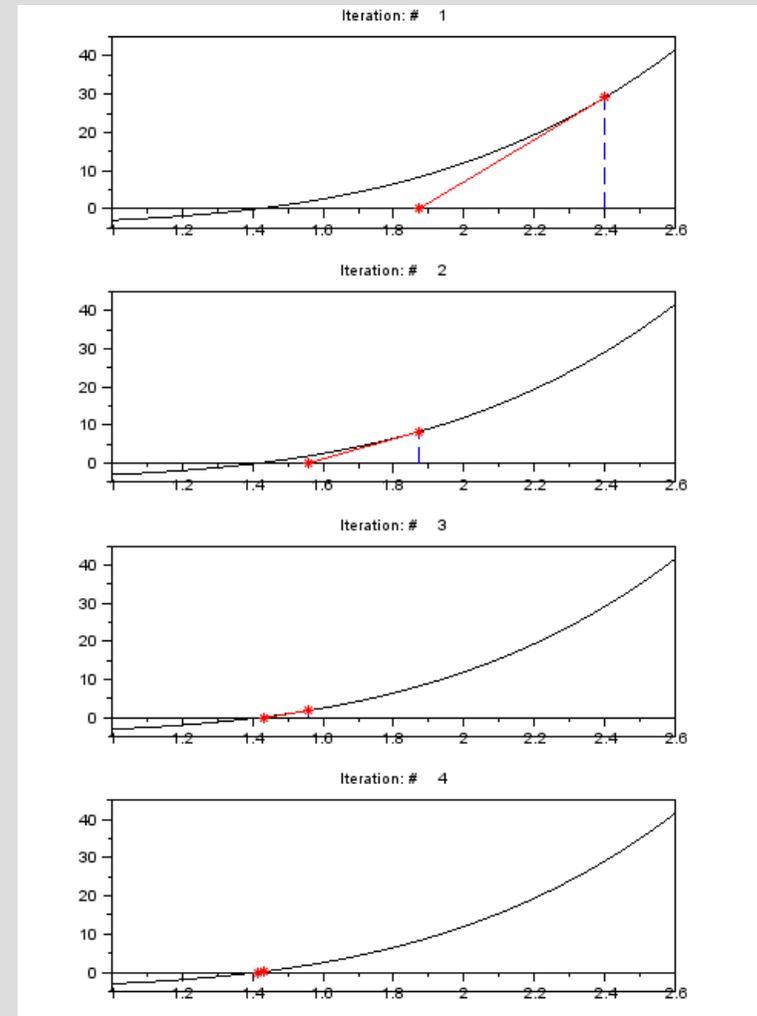
$$(x_1 =) x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Once the new approximation is computed, we repeat the same procedure with the new initial point x_1 . The iterative formula is then

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

The figures on the right refer to the first four iterations of the method for the function $f(x) = x^4 - 4$ with initial guess value $x_0 = 2.4$.

The function is available in the file [newton.sci](#), while all the examples related to this method can be found in [newton_test.sce](#).



(First four iterations of the Newton method)

Step 16: Convergence of the Newton method

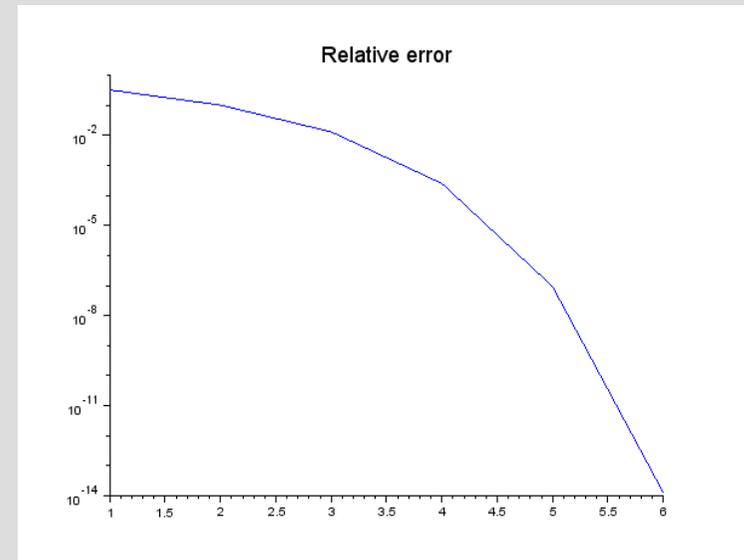
The main pseudo-code of the algorithm is the following:

```
Algorithm
xk = x0;
iter = 1 // Current iteration
while iter <= itermax do
    iter = iter+1
    xkpl = xk-f(xk)/f'(xk)
    if abs(xkpl-xk)<tol break // Converg. test
    xk = xkpl;
end1
```

The algorithm iterates until convergence or the maximum number of iterations is reached. At each iterations a function evaluation with its derivative is required. The “break” statement terminates the execution of the while loop.

For the newton method it is possible to prove the following results: if the function f is continuous with continuous derivatives until order 2 near the zero, the zero is simple (has multiplicity 1) and the initial guess x_0 is picked in a neighborhood of the zero, then the method converges and the convergence rate is equal to $p = 2$ (quadratic).

The figure on the right shows the relative error related to the iterations (reported in the table below) of the method applied to the function $f(x) = x^4 - 4$ in the interval $[1,2]$, where the analytic solution is $\sqrt{2}$. As expected, the number significant figures doubles at each iteration.



(Relative error of the Newton method)

iter	x	f(x)
1	1.872337962962963	8.289578049123726
2	1.556605160415167	1.871024105224409
3	1.432587368746513	0.211962253538407
4	1.414564032544065	0.003966591547560
5	1.414213692599495	0.000001473343727
6	1.414213562373113	0.000000000000203

(Iterations of the Newton method)

Step 17: Newton method (loss of quadratic convergence)

A zero x^* is said to be multiple with multiplicity r if

$$f(x^*) = f'(x^*) = \dots = f^{r-1}(x^*) = 0, \text{ and } f^r(x^*) \neq 0.$$

In the Newton method, if the zero is multiple, the convergence rate decreases from quadratic to linear. As an example, consider the function $f(x) = x^2$ with zero $x^* = 0$. Then the Newton method can be stated as

$$x_{k+1} = x_k - \frac{x_k^2}{2x_k} = \frac{x_k}{2}$$

giving the error

$$\varepsilon_{k+1} = x_{k+1} - 0 = \frac{\varepsilon_k}{2}$$

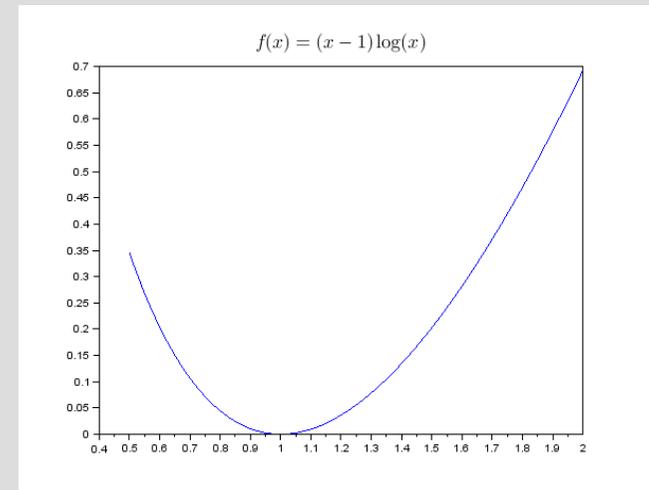
which is clearly linear.

On the right we report an example of loss of quadratic convergence applied to the function

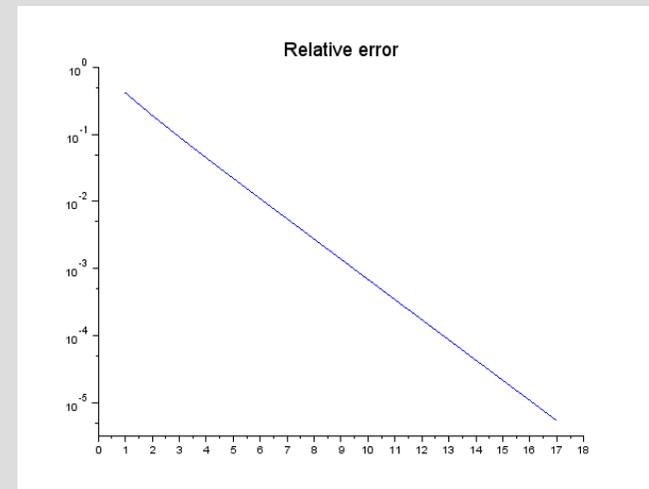
$$f(x) = (x - 1) \log x$$

which has a zero of multiplicity 2 in $x^* = 1$.

The relative error shows a linear behavior, indeed the method gains a constant number of significant figures every 3 or 4 iterations.



(Function with a zero of multiplicity 2)



(Loss of quadratic convergence)

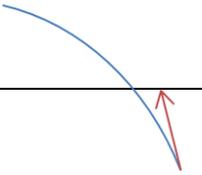
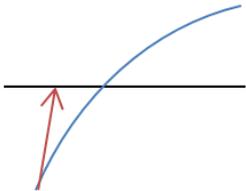
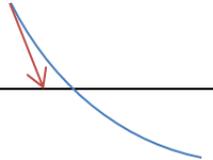
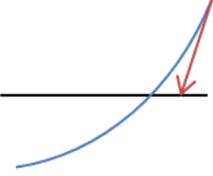
Step 18: Newton method (Global convergence)

If the function of which we are looking for the zeros and its derivatives until order 2 are continuous, it is possible to ensure the global convergence of the Newton method by choosing a proper initial guess point.

As reported in the table on the right, under the above mentioned hypothesis it is possible to identify a neighborhood U of the zero such that, for each initial guess x_0 in U , the sequence $\{x_k\}$ is monotonically decreasing (or increasing) to the zero.

For instance, if the function is convex and increasing (second row and second column case in the table), the Newton method with an initial guess picked on the right of the zero converges monotonically decreasing to the unknown zero.

The Newton method with the above choosing criterion also ensures that all $\{x_k\}$ are well defined, *i.e.* the method does not generate any point in which the function is not defined (an example in which the monotonicity of the convergence is important is the logarithm function, which is not defined for negative values).

Function properties	$f'(x) < 0$ (decrease)	$f'(x) > 0$ (increase)
$f''(x) < 0$ (concave)		
	(Right domain)	(Left domain)
$f''(x) > 0$ (convex)		
	(Left domain)	(Right domain)

(How to choose the initial guess for global convergence)

Step 19: Fixed point iteration method

The fixed point iteration method transforms the original problem $f(x) = 0$ into the problem $x = g(x)$ and solves it using an iterative scheme of the form:

$$x_{k+1} = g(x_k).$$

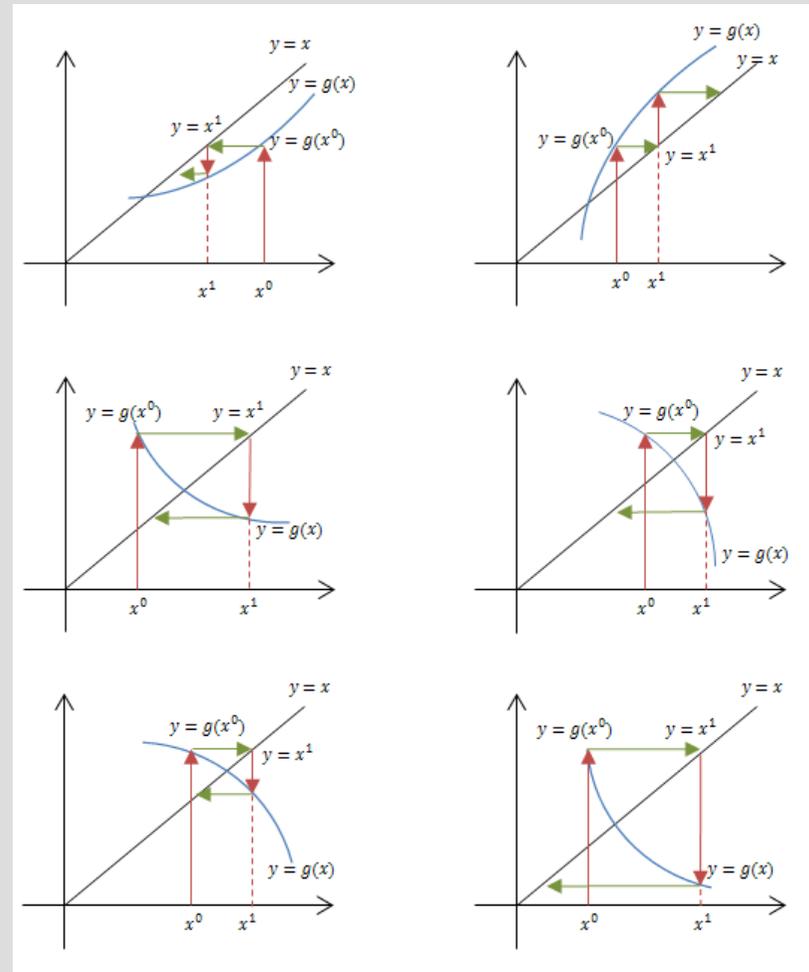
If the iterative scheme converges to the value x_* , i.e. $x_* = g(x_*)$, then x_* is also a zero of $f(x)$, since $f(x_*) = x_* - g(x_*) = 0$.

Solving the equation $x = g(x)$ is equivalent to solve the following system

$$\begin{cases} y = g(x) \\ y = x \end{cases}$$

On the right we reported some graphical examples of iterations applied to 6 different functions g . The three examples on the left show cases in which the method converges to the unknown zero, while among the examples on the right there is no convergence of the method, even if the functions seem to be quite similar to the ones on the left.

Note: The Newton method is a particular case of fixed point iteration method where $g(x_k) = x_k - \frac{f(x_k)}{f'(x_k)}$.



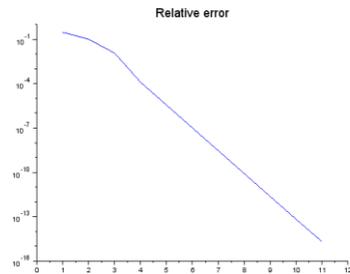
(Graphical examples of iterations applied to 6 different functions g)

Step 20: Fixed point iteration method - example #1

The example on the right refers to the function $f(x) = x^4 - 4$, where the considered fixed point iteration is

$$x_{k+1} = g(x_k) = \frac{4 + 11x_k - x_k^4}{11}$$

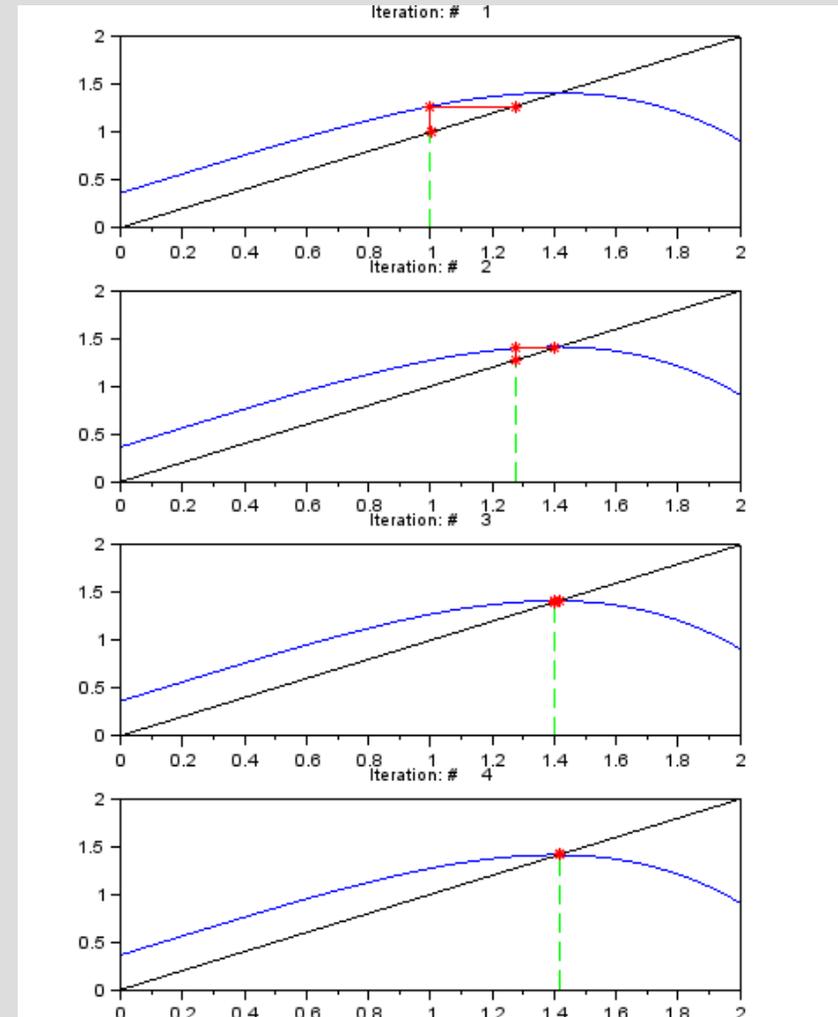
Here the relative error shows a linear behavior of the method, while on the right we can see the first four iterations of the method.



iter	xold	xnew
1	1.0000000000000000	1.2727272727272727
2	1.272727272727273	1.3978305008972314
3	1.397830500897231	1.4143902398254873
4	1.414390239825487	1.4142084896613987
5	1.414208489661399	1.4142137070134573
6	1.414213707013457	1.4142135582480804
7	1.414213558248080	1.4142135624907362
8	1.414213562490736	1.4142135623697401
9	1.414213562369740	1.4142135623731908
10	1.414213562373191	1.4142135623730923
11	1.414213562373092	1.4142135623730951

(Relative error and iterations of the fixed point iteration method)

The function is available in the file [fixedpoint.sci](#), while the example can be found in [fixedpoint_test.sce](#).



(First four iterations of the fixed-point iteration method)

Step 21: Convergence of the fixed point iteration method

It is possible to prove that the sequence $\{x_k\}$ converges to x_* if $|g'(x)| \leq m < 1$ (i.e. it is a *contraction*) in the interval I containing the initial guess and x_k for all k (i.e. $g(x) \in I$). Moreover, in this case, it is possible to prove that the solution x_* is the unique solution in the interval I of the equation $f(x) = 0$. While, if $|g'(x)| > 1$ in the whole interval I , the sequence does not converge to the solution (even if we start very close to the zero x_*).

The convergence rate of the fixed point iteration method is:

- If $g'(x_*) \neq 0$, the method has a linear convergence rate;
- If $g'(x_*) = 0$ and $g''(x_*) \neq 0$, the method has a quadratic convergence rate;

Example: The fixed point iteration method applied to the Newton method with fixed point function

$$g(x) = x - \frac{f(x)}{f'(x)}$$

with $f(x_*) = 0$ and $f'(x_*) \neq 0$ shows a quadratic rate of convergence, indeed we have $g'(x_*) = 0$ and $g''(x_*) \neq 0$ ($g'(x) = 1 - \frac{f'(x)f'(x) - f(x)f''(x)}{(f'(x))^2}$).

Proof of convergence (idea)

Using the Taylor formula we have

$$\begin{aligned} x_1 = g(x_0) &\rightarrow \varepsilon_1 = x_* - x_1 = g(x_*) - g(x_0) \approx (x_* - x_0)g'(\xi_0) \\ x_2 = g(x_1) &\rightarrow \varepsilon_2 = x_* - x_2 = g(x_*) - g(x_1) \approx (x_* - x_1)g'(\xi_1) \\ &\vdots \\ x_k = g(x_{k-1}) &\rightarrow \varepsilon_k = x_* - x_k = g(x_*) - g(x_{k-1}) \approx (x_* - x_{k-1})g'(\xi_{k-1}) \end{aligned}$$

where ξ_i are unknown values in the intervals (x_*, x_i) . If the derivatives are bounded by the relation

$$|g'(\xi_i)| \leq m$$

the error can be written as

$$|\varepsilon_k| \leq (x_* - x_k) \cdot m = |\varepsilon_{k-1}| \cdot m$$

i.e.

$$|\varepsilon_k| \leq m^k |\varepsilon_0|$$

where $\varepsilon_0 = x_* - x_0$ is the initial error.

Proof of convergence rate (idea)

Assuming $g(x) \in C^{l+1}$ and using the Taylor formula we have

$$\begin{aligned} \varepsilon_{k+1} &= g(x_*) - g(x_k) \\ &= g'(x_*)\varepsilon_k + \frac{1}{2}g''(x_*)\varepsilon_k^2 + \dots + \frac{1}{l!}g^{(l)}(x_*)\varepsilon_k^l + \frac{1}{(l+1)!}g^{(l+1)}(\xi)\varepsilon_k^{l+1} \end{aligned}$$

where $\varepsilon_k = (x_* - x_k)$ and $\xi \in (x_*, x_k)$.

If $g'(x_*) \neq 0$ we have a **linear** rate of convergence of the sequence, i.e.

$$\lim_{k \rightarrow \infty} \frac{|\varepsilon_{k+1}|}{|\varepsilon_k|} = g'(x_*).$$

If $g'(x_*) = 0$ and $g''(x_*) \neq 0$ we have a **quadratic** rate of convergence of the sequence, i.e.

$$\lim_{k \rightarrow \infty} \frac{|\varepsilon_{k+1}|}{|\varepsilon_k|^2} = \frac{1}{2}g''(x_*).$$

Step 22: Fixed point iteration method - example #2

This example refers to the zero of the function

$$f(x) = x - x^{1/3} - 2$$

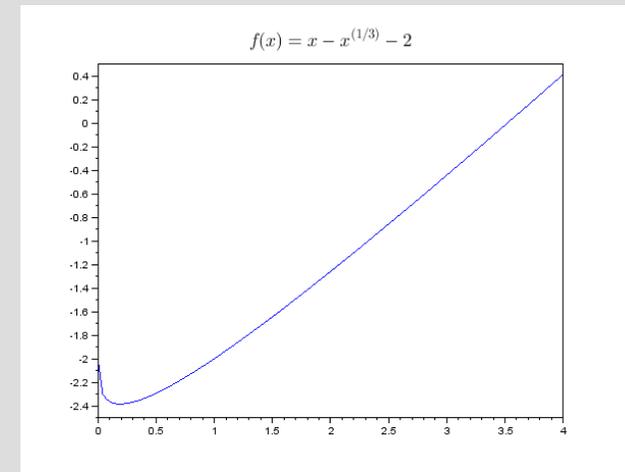
which is $x_* \cong 3.5213797$.

Starting from this function it is possible to write different kinds of fixed point iterations:

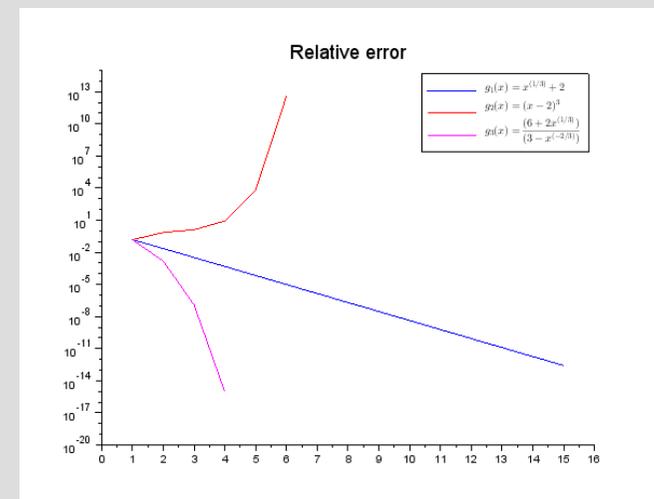
1. $x = x^{1/3} + 2$ with $g(x) = x^{1/3} + 2$: This function shows a linear behavior of convergence (blue line), indeed the derivative $1 > g'(x_*) \cong 0.1440136 \neq 0$;
2. $x = (x - 2)^3$ with $g(x) = (x - 2)^3$: This function does not converge (red line), indeed the derivative $g'(x_*) \cong 6.9437886 > 1$;
3. $x = \frac{6+2x^{1/3}}{3-x^{-2/3}}$ with $g(x) = \frac{6+2x^{1/3}}{3-x^{-2/3}}$: This function shows a quadratic rate of convergence (purple line), indeed the derivative $g'(x_*) = 0$.

The initial guess is equal to $x_0 = 3$.

Also this second example can be found in the file [fixedpoint_test.sce](#).



(Original function)



(Relative error for different fixed point iterations)

Step 23: Comparison of the methods – an example

This example refers to the zero of the function

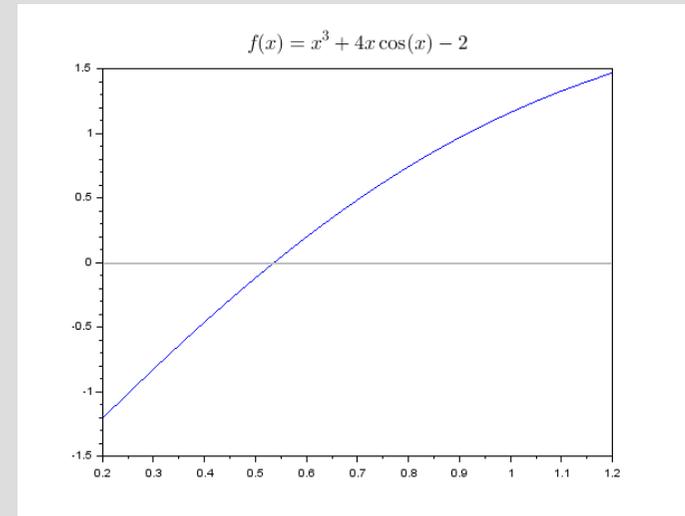
$$f(x) = x^3 + 4x \cos(x) - 2$$

which is $x_* \cong 0.5368385515668$.

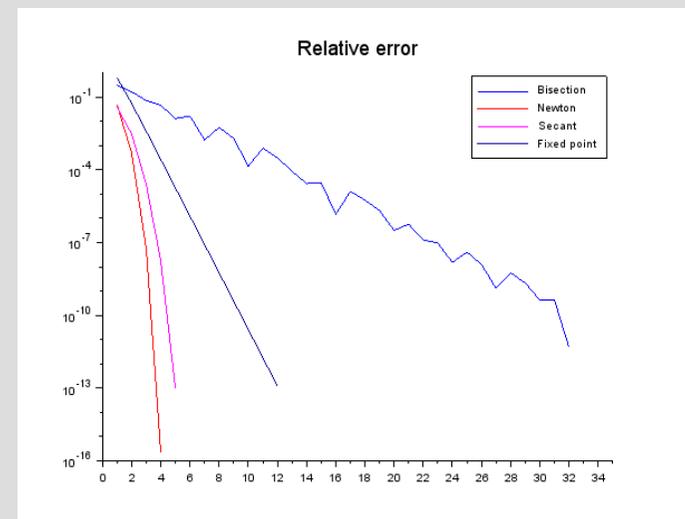
In the figure on the right we can see a typical result on relative errors, in step with the rates of convergence discussed earlier for the different methods.

At a first glance, one might think that the Newton method should be always chosen, since its convergence is the best one, but it has to be considered that this method needs the computation of the derivative, which could require the evaluation of the original function, which can be numerically very expensive.

The source code of this example can be found in the file [main_fcn.sce](#).



(Nonlinear test function)



(Relative error of the 4 methods)

Step 24: Concluding remarks and References

In this tutorial we have collected a series of numerical examples written in Scilab to tackle the problem of finding the zeros of scalar nonlinear equations with the main methods on which are based the state-of-the-art algorithms.

On the right-hand column you may find a list of references for further studies.

1. Scilab Web Page: www.scilab.org.
2. Openeering: www.openeering.com.
3. K. Atkinson, *An Introduction to Numerical Analysis*, John Wiley, New York, 1989.

Step 25: Software content

To report a bug or suggest some improvement please contact the Openeering team at the web site www.openeering.com.

Thank you for your attention,

Anna Bassi and Manolo Venturin

```
-----  
Main directory  
-----  
ex1.sce           : Examples of zero-finding problems  
ex2.sce           : Examples of separation and fixed point  
ex3.sce           : Example on conditioning  
ex4.sce           : Example of inverse function  
ex5.sce           : Examples of convergence rates  
fintsearch.sci    : Search intervals  
fintsearch_test.sce: Test for search intervals  
bisection.sci     : Bisection method  
bisection_test.sce: Test for the bisection method  
secant.sci        : Secant method  
secant_test.sce   : Test for the secant method  
newton.sci        : Newton-Raphson method  
newton_test.sce   : Test for the Newton-Raphson method  
fixedpoint.sci    : Fixed point iteration method  
fixedpoint_test.sce: Tests for the fixed point iteration method  
main_fcn.sce      : Comparison of methods  
license.txt       : License file
```