Optimization Techniques with SCILAB

By

Gilberto E. Urroz, Ph.D., P.E.

Distributed by

infoClearinghouse.com

A "zip" file containing all of the programs in this document (and other SCILAB documents at InfoClearinghouse.com) can be downloaded at the following site:

http://www.engineering.usu.edu/cee/faculty/gurro/Software_Calculators/Scilab_Docs/ScilabBookFunctions.zip

The author's SCILAB web page can be accessed at:

http://www.engineering.usu.edu/cee/faculty/gurro/Scilab.html

Please report any errors in this document to: gurro@cc.usu.edu

Optimization using SCILAB

The objective of optimization techniques is to determine the values of the independent variables that minimize or maximize a function (known, in general, as the *objective function*) subject to a number of constraints. Techniques presented in this chapter include graphic approach for functions of two variables, linear programming, quadratic programming, and general non-linear programming.

Definitions

The general minimization problem can be stated as

$$\min_{x\in D}\{f_1(x), f_2(x),..., f_m\},$$

where $f_i(x)$, $i=1,2,\ldots,m$, are scalar *objective functions* that map a vector x, known as the *design variable vector*, into an *objective space*. The values of x are restricted to a *feasible domain* region D described by N inequality constraints and M equality constraints. The domain D is described as

$$D = \{x : g_j(x)\le 0,\ h_k(x)=0,\ j=1,2,\ldots,N,\ k=1,2,\ldots,M\}.$$

A linear programming problem results when the objective function and the constraints are linear functions of the design variables. Non-linear programming problems result from having at least one of the objective or constraint functions be non-linear.

Graphical solution

A graphical solution is presented here for an objective function of two variables, $w = f(x,y) = \sin(x)\cos(y)$. The purpose of this solution is to illustrate the idea of optimization with a simple function whose graph provides a good idea of the solution. Methods for pinpointing the solution will be presented later.

The idea behind the graphical solution is to produce a contour plot of the function in the feasible domain as well as a combined three-dimensional and contour plot. These two visual aids would be, in many cases, enough to determine the values of the design variables that produce a minimum or a maximum of the function.

The problem to be solved is specified as

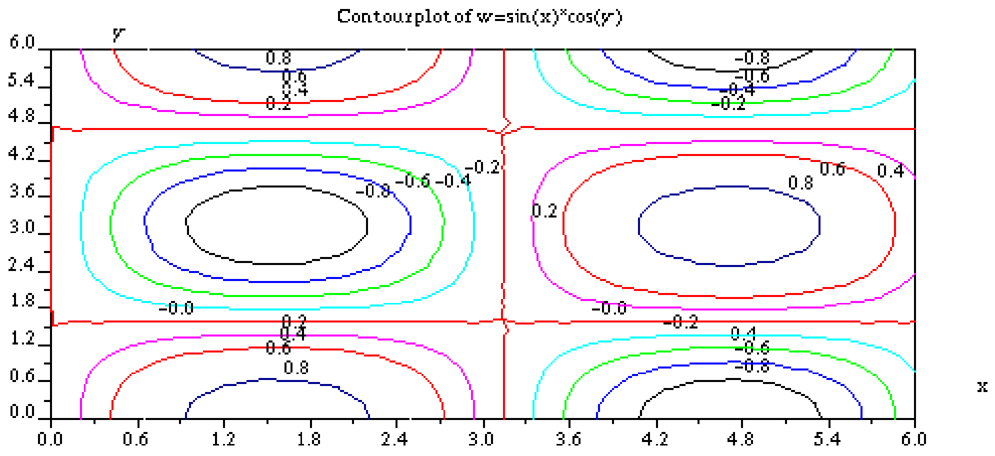
$$\min_{x,y\in D}\{w = f(x,y)\},$$

where D is defined by

$$D = \{(x,y) : 0<x<6,\ 0<y<6\}.$$

Using SCILAB we will define the function *f(x,y)* and produce a contour plot of the function in the feasible domain:

```
-->deff('[w]=f(x,y)','w=sin(x)*cos(y)')

-->x=[0:0.25:6];y=[0:0.25:6];z=feval(x,y,f);

-->contour(x,y,z,10)

-->xtitle('Contour plot of w=sin(x)*cos(y)','x','y')
```

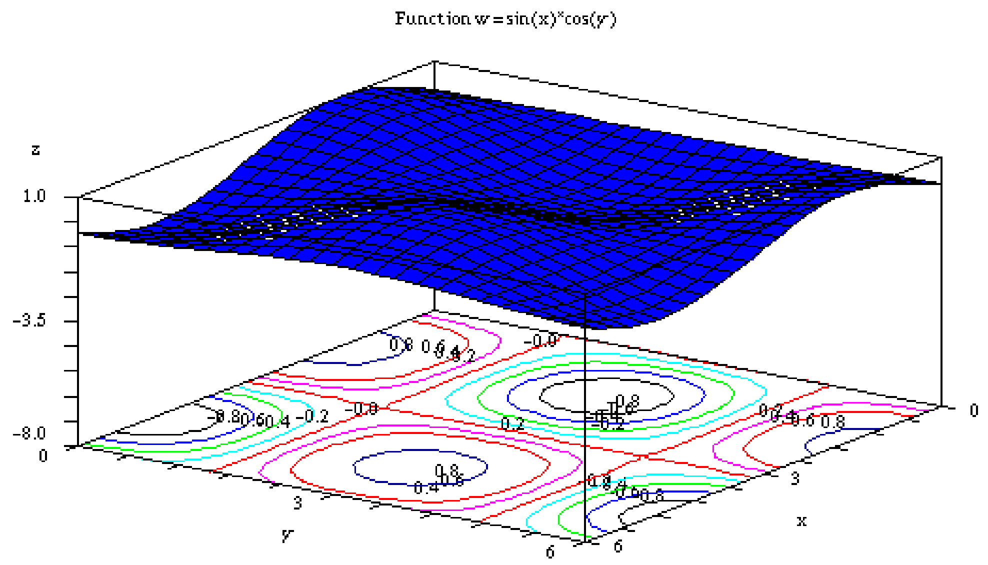


The contour plot is good enough to help us determine the solution. Within the solution domain we can identify three maximum values (points near (1.5,0.0), (1.5,6.0), and (4.7,3.2)) and three minimum values (points near(1.5,3.2), (4.7,0), and (4.7,6)).

A combined three-dimensional plot with contours can be generated using functions *contour* and *plot3d* as shown next. First, we define a parallelepiped (*rect*) large enough to contain the three-dimensional plot and to be able to show the projected contour plots in a plane parallel to the x-y plane. The values in *rect* are [x_{min},x_{max},y_{min},y_{max},z_{min},z_{max}]. The parameters in the call to *contour* are as follows: *x,y* are vectors containing values of x and y coordinates; *z* is the matrix of values of *z = f(x,y)* evaluated earlier; the next three values are *nc = 10*, the number of contours to be shown, $\theta = 35$ and $\phi = 45$ are the values, in degrees, of the angles of the viewpoint used for the three-dimensional graph; the string *'x@y@z'* indicates the axes labels to be used; the vector [1,1,1] is a vector specifying the graph style (see `--> help plot3d`); *rect* is the vector defined above; and, zl = -8 is the value of the plane z = zl where the contour plot is to be projected. We chose a value of -8 to allow enough spacing between the surface, which ranges in -1 < z < 1, and the contour plane. The parameters in the call to function *plot3d* are very similar to those in the call to function *contour*, except that a number of contours is not required.

```
-->rect=[0,6,0,6,-8,1]
 rect  =

!   0.    6.    0.    6.  - 8.    1. !

-->contour(x,y,z,10,35,45,'x@y@z ',[1,1,1],rect,-8)

-->plot3d(x,y,z,35,45,'x@y@z',[2,1,4],rect)
```

```
-->xtitle('Function w = sin(x)*cos(y)')
```



The combined three-dimensional and contour plot serves to verify the observations already made from the contour plot alone about the location of the minimum and maximum values of the function in the domain of interest.

A graphical solution is limited to functions of one or two variables and provides only an approximate location for the solution. The methods presented next for linear, quadratic, and non-linear functions will produce more accurate solutions.

Linear programming

Linear programming is an optimization method applied to the solution of problems in which both the objective function and the constraint functions are linear functions of the design variables. The problem may be expressed as

$$\textit{minimize } f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$$

subject to

$$\mathbf{Ax} = \mathbf{b},$$
$$\mathbf{Gx} \le \mathbf{h},$$

and

$$\mathbf{x}_L \le \mathbf{x} \le \mathbf{x}_U$$

where $\mathbf{x}$ is a vector containing the n design variables, i.e., $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, $\mathbf{c}$ is the constant column vector containing the n coefficients of the objective function, $\mathbf{A}$ is an $m\times n$ matrix, $\mathbf{b}$ is a column vector with m components, $\mathbf{G}$ is an $p\times q$ matrix, $\mathbf{h}$ is a column vector with p components, and $\mathbf{x}_L$ and $\mathbf{x}_U$ are vectors indicating constraints on the values of the design variables.

As an example, consider the problem of minimizing the objective function

$$f(\mathbf{x}) = 3x_1 + 5x_2 - 2x_3,$$

subject to

$$x_1+3x_2 = 5$$
$$x_1+x_2 - x_3 = 2$$

$$2x_1 - x_2 \leq 3$$
$$x_1 + x_2 + x_3 \leq 25$$

$$0\leq x_1\ 5,$$
$$0\leq x_2 \leq 10,$$
$$0\leq x_3 \leq 3$$

The problem can be re-written as *minimize* $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$, subject to $\mathbf{Ax} = \mathbf{b}$, $\mathbf{Gx} \leq \mathbf{h}$, and $\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U$, with

$$\mathbf{c} = \begin{bmatrix} 3 \\ 5 \\ -2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 1 & 3 & 0 \\ 1 & 1 & -1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 5 \\ 2' \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 1 & 3 & 0 \\ 1 & 1 & -1 \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} 5 \\ 2' \end{bmatrix},$$

$$\mathbf{x}_L = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{x}_U = \begin{bmatrix} 5 \\ 10 \\ 3 \end{bmatrix}.$$

SCILAB function for solving linear programming problems

SCILAB provides function *linpro* for the solution of linear programs. The function can be called using any of the following forms:

[xopt,lagr,fopt]=linpro(c,C,d [,x0])
[xopt,lagr,fopt]=linpro(c,C,d,xL,xU,[,x0])
[xopt,lagr,fopt]=linpro(c,C,d,xL,xU, m, [,x0])
[xopt,lagr,fopt]=linpro(c,C,d,xL,xU, m,x0 [,imp])

The function returns values *xopt*, *lagr*, and *fopt*, where *xopt* is the value of the design variables vector that minimizes the objective function, *f(x) = c'*x*, *lagr* is a vector of Lagrange multipliers (see below), and *fopt* is the optimal value of the function, i.e., *fopt = f(xopt).* The arguments of the function call are:

- the vector of *n* coefficients of the objective function, *c*,
- a matrix *C* of (*m*+*p*) rows and *n* columns containing the rows of matrices **A** and **G** in that order,
- a column vector *d* with (*m*+*p*) rows containing the elements of vectors **b** and **h** in that order,
- column vectors *xL* and *xU* representing the vectors of lower and upper bounds of the design variables, i.e., $\mathbf{x}_L$ and $\mathbf{x}_U$, respectively,
- *m* is the number of equality constraints (i.e., the number of rows in matrix **A**),

- *x0*, an optional argument for the first three forms of the function call, is the vector containing the *n* initial guesses for the solution,
- *imp,* also an optional argument is an integer value that determines the amount of information provided by function *linpro* (try values *imp = 7,8, …*)

The first form of the call to *linpro*, namely, *[xopt,lagr,fopt]=linpro(c,C,d,xL,xU,[,x0])*, is used when there are only inequality constraints. The second form, *[xopt,lagr,fopt] = linpro(c, C, d, xL, xU,[,x0]),* is used when inequality constraints as well as lower and upper boundaries for the design variables. The second form of the function call, *[xopt,lagr,fopt]=linpro(c,C,d,xL,xU, m, [,x0])*, is used when there are inequality and equality constraints (*m* equality constraints), as well as lower and upper boundaries of the design variables. Finally, the last form of the function call, *[xopt,lagr,fopt]=linpro(c,C,d,xL,xU, m,x0 [,imp])*, requires an initial guess *x0* besides equality and inequality constraints and upper and lower bounds of the design variables.

To illustrate the application of function *linpro* we will use the problem stated earlier, namely, *minimize* $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$, subject to $\mathbf{A}\cdot\mathbf{x} = \mathbf{b}$, $\mathbf{G}\cdot\mathbf{x} \le \mathbf{h}$, and $\mathbf{x}_L \le \mathbf{x} \le \mathbf{x}_U$, with

$$\mathbf{c} = \begin{bmatrix} 3 \\ 5 \\ -2 \end{bmatrix},\quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},\quad \mathbf{A} = \begin{bmatrix} 1 & 3 & 0 \\ 1 & 1 & -1 \end{bmatrix},\quad \mathbf{b} = \begin{bmatrix} 5 \\ 2, \end{bmatrix},\quad \mathbf{G} = \begin{bmatrix} 1 & 3 & 0 \\ 1 & 1 & -1 \end{bmatrix},\quad \mathbf{h} = \begin{bmatrix} 5 \\ 2, \end{bmatrix},$$

$$\mathbf{x}_L = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},\quad \mathbf{x}_U = \begin{bmatrix} 5 \\ 10 \\ 3 \end{bmatrix}.$$

We will illustrate the use of function *linpro* by changing the number of constraints of the problem. First, we load the constant vectors and matrices shown using SCILAB:

```
-->c=[3;5;-2];A=[1,3,0;1,1,-1];b=[5;2];G=[1,3,0;1,1,-1];h=[5;2];

-->xL = zeros(3,1); xU=[5;10;3];
```

Applications of function *linpro* - case 1: inequality constraints and bounds present

The following SCILAB commands set up and solve the linear programming problem using only inequality constraints and bounds for the design variables. For this case we take C = **G**, d = **h**, and obtain the solution to the linear programming problem:

```
-->C=G,d=h
 C  =

!   1.    3.    0. !
!   1.    1.  - 1. !



 d  =

!   5. !
!   2. !

-->[xopt,lagr,fopt]=linpro(c,C,d,xL,xU)
 fopt  = - 6.
 lagr  =

! - 3. !
! - 5. !
!   2. !
!   0. !
!   0. !

 xopt  =

!   0. !
!   0. !
!   3. !
```

The optimal solution is $x_1 = 0$, $x_2 = 0$, and $x_3 = 3$, corresponding to a value of -6 for the function.

Applications of function *linpro* - case 2: inequality constraints and bounds present

The following SCILAB commands set up and solve the linear programming problem using both equality and inequality constraints and bounds for the design variables. For this case we take C = **[A;G]**, d = **[b;h]**, as follows:

```
-->C=[A;G], d = [b;h]
 C  =

!   1.    3.    0. !
!   1.    1.  - 1. !
!   1.    3.    0. !
!   1.    1.  - 1. !
 d  =

!   5. !
!   2. !
!   5. !
!   2. !

-->[xopt,lagr,fopt]=linpro(c,C,d,xL,xU,2)
 fopt  =

    9.
 lagr  =

!   0. !
!   0. !
!   0. !
! - 1. !
! - 2. !
!   0. !
!   0. !

 xopt  =

!    .5 !
!   1.5 !
!   0.  !
```

Alternative calls to function *linpro* may include an initial guess for the solution as the last argument in the function call. In the first case, a specific value of x is provided. In the second case, the option ‘v’ indicates that a vertex of the feasible region is to be used as an initial guess. In the third case, the option ‘g’ indicates that an arbitrary initial value -- generated by the function -- is to be used. You can check, by using SCILAB, that the solution is the same in any of the following function calls:

```
-->[xopt,lagr,fopt]=linpro(c,C,d,xL,xU,2,[0;0;0]);

-->[xopt,lagr,fopt]=linpro(c,C,d,xL,xU,2,'v');

-->[xopt,lagr,fopt]=linpro(c,C,d,xL,xU,2,'g');
```

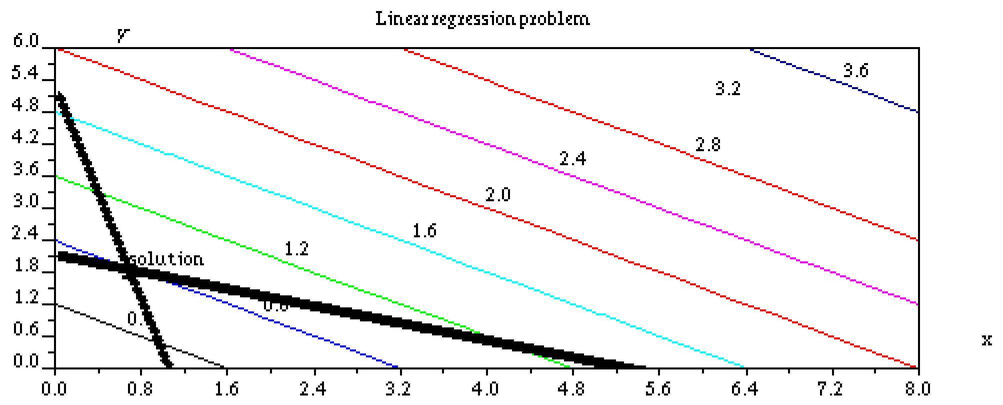
Graphical illustration of linear programming solution

A linear programming involving two design variables *(x,y)* can be used to illustrate the solution of the problem through graphics. The problem to be solved is to minimize $f(x,y) = x_1/4 + x_2/3$, subject to the constraints $-5x_1-x_2 \le -5$, $-2x_1+5x_2 \le -10$, $x_1 \ge 0$, $x_2 \ge 0$. The problem can be set up and solved using SCILAB as follows:

```
-->c=[1/4;1/3];xL=[0;0];xU=[1e10;1e10]; C = [-5,-1;-2,-5]; d = [-5;-10];

-->[xopt,lagr,fopt]=linpro(c,C,d,xL,xU,0);

-->xopt
 xopt  =

!    .6521739 !
!   1.7391304 !
```

To illustrate the problem graphically we first define the objective function, *w=f(x,y)*, and plot a set of contours for it. We also define functions representing the constraints, *y1=f1(x)* and *y2=f2(x)*, and plot the lines corresponding to these constraints. The solution is the vertex where the two constraint lines intercept as labeled in the plot.

```
-->deff('[w]=f(x,y)','w=c(1)*x+c(2)*y')
-->xx=[0:1:8];yy=[0:1:6];zz=feval(xx,yy,f);
-->contour(xx,yy,zz,10)
-->deff('[y1]=f1(x)','y1=-5*x+5')
-->deff('[y2]=f2(x)','y2=(10-2*x)/5')
-->xxx=[0:0.01:8];yy1=f1(xxx);yy2=f2(xxx);
-->contour(xx,yy,zz,10)
-->plot2d([xxx' xxx'],[yy1' yy2'],[-1,-1],'011',' ',[0 0 8 6])
-->xstring(xopt(1)+0.05, xopt(2)+0.05,'solution')
-->xtitle('Linear regression problem','x','y')
```



Notice that the point of intersection of the lines corresponds with the minimum value of the function illustrated by the contours. The contours decrease towards the origin. The feasible domain is the region above and to the right of the two thick lines, above the x-axis and to the right of the y-axis.

Lagrange multipliers

The elements in the vector of Lagrange multipliers *lagr* provide information about the effect of upper and lower boundaries as well as on any constraints. . If vectors of lower and upper bounds, i.e., xL and xU, are provided, the vector of Lagrange multipliers, *lagr*, will have (*n*+*m*+*p*) components. Components *1* to *n* are associated with the *n* upper and lower bounds of the constraint variables, components *n+1* through (*n+m)* are associated with the *m* equality constraints, while components *(n+m+1)* through (*n+m+q)* are associated with the *q* inequality constraints.

If any component of the Lagrange multiplier vector is zero, it means that that particular bound or constraint is not active (i.e., the solution was found without using that particular condition). If an lower bound constraint is active, then the corresponding Lagrange multiplier is negative. On the other hand, if an upper bound constraint is active, the corresponding Lagrange multiplier is positive.

For example, in the first application of function *linpro* used above, we found the following results for the vector of Lagrange multipliers:

```
lagr  =

! - 3. !
! - 5. !
!   2. !
!   0. !
!   0. !
```

The first three elements are related to the upper and lower bounds of the variables x_1, x_2, and x_3, respectively. The fact that *lagr(1)* = *-3* and *lagr(2)* = *-5* indicates that the lower bound constraint was used for variables x_1 and x_2. What this means, in this case, is that $x_1 = 0$ and $x_2 = 0$, i.e., variables x_1 and x_2 took their lower bounds in the optimal solution. The fact that *lagr(3)* = *2* indicates that the upper bound constraint was used for x_3, i.e., $x_3 = 3.0$. Finally, the fact that *lagr(4)* = *lagr(5)* = *0* indicates that none of the two inequality constrains in the problem was used. The optimal solution found was $x_1 = 0$, $x_2 = 0$, and $x_3 = 3$.

In the second example of *linpro* presented above the Lagrange multipliers vector was found to be

```
lagr  =

!   0. !
!   0. !
!   0. !
! - 1. !
! - 2. !
!   0. !
!   0. !
```

This is interpreted as indicating that none of the three upper or lower bound constraints for the design variables were used (*lagr(1) = lagr(2) = lagr(3) = 0*), neither were any of the two inequality constraints (*lagr(6) = lagr(7) = 0*). The equality constraints, represented by *lagr(4) = -1* and *lagr(5) = -2*, were both used in the solution.

What are Lagrange multipliers?

Lagrange multipliers are variables introduced in an optimization problem to incorporate the problem constraints into an expanded objective function. For example, given the objective function $f(x_1,x_2,\dots,x_n)$ subject to the constraints represented by $m(<n)$ equations

$$\phi_1(x_1,x_2,\dots,x_n) = 0,\ \phi_2(x_1,x_2,\dots,x_n) = 0,\ \dots,\ \phi_n(x_1,x_2,\dots,x_n) = 0,$$

we can form the expanded objective function

$$\Phi(x_1,x_2,\dots,x_n) = f(x_1,x_2,\dots,x_n) + \sum_{j=1}^{m} \lambda_j \cdot \phi(x_1,x_2,\dots,x_n)..$$

The variables λ_j introduced in function Φ are known as Lagrange multipliers. The solution to the problem is obtained by solving simultaneously the m constraint equations and the n equations resulting from the conditions

$$\partial\Phi/\partial x_1 = 0,\ \partial\Phi/\partial x_2 = 0,\dots,\ \partial\Phi/\partial x_n = 0.$$

The resulting problem has $(n+m)$ unknowns, i.e., $x_1,\ x_2,\ \dots,\ x_n$, and $\lambda_1,\ \lambda_2,\ \dots,\ \lambda_m$, and $(n+m)$ equations. Thus, the problem has a unique solution that provides the minimum (or maximum) value of the objective function $f(x_1,x_2,\dots,x_n)$.

Quadratic programming

Quadratic programming is an optimization method applied to the solution of problems in which both the objective function is defined, in general, by an objective function consisting of a quadratic form plus a linear combination of the design variables. The constraint functions of the problem are still linear functions of the design variables. The problem may be expressed as

$$\text{minimize } f(\mathbf{x}) = ½\, \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x}$$

subject to

$$\mathbf{Ax} = \mathbf{b},$$
$$\mathbf{Gx} \le \mathbf{h},$$

and

$$\mathbf{x}_L \le \mathbf{x} \le \mathbf{x}_U$$

where $\mathbf{x}$ is a vector containing the n design variables, i.e., $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$, $\mathbf{Q}$ is a $n\times n$ symmetric square matrix, $\mathbf{c}$ is a constant column vector containing n coefficients, $\mathbf{A}$ is an $m\times n$ matrix, $\mathbf{b}$ is a column vector with m components, $\mathbf{G}$ is an $p\times q$ matrix, $\mathbf{h}$ is a column vector with p components, and $\mathbf{x}_L$ and $\mathbf{x}_U$ are vectors indicating constraints on the values of the design variables.

For the case in which two design variables are present, i.e., x = [x_1 x_2], the objective function $f(\mathbf{x}) = ½\, \mathbf{x}^T\mathbf{Qx} + \mathbf{c}^T\mathbf{x}$ is a quadratic equation in x_1 and x_2 given by

$$f(x_1,x_2) = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \cdot \begin{bmatrix} q_{1,1} & q_{1,2} \\ q_{2,1} & q_{2,2} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} c_1 & c_2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

$$f(x_1,x_2) = \frac{1}{2} x_1^{\ 2}\, q_{1,1} + \frac{1}{2} x_1\, x_2\, q_{2,1} + \frac{1}{2} x_2\, x_1\, q_{1,2} + \frac{1}{2} x_2^{\ 2}\, q_{2,2} + c_1\, x_1 + c_2\, x_2$$

Consider, for example, the case in which $q_{11} = 2$, $q_{12} = q_{21} = -1$, $q_{22} = 3$, $c_1 = 2$, and $c_2 = 4$. The objective function is,

$$f(x_1,x_2) = x_1^2 - x_1x_2 + (3/2)x_2^2 + 2x_1+4x_2.$$

Subject to the constraints:

$$-2x_1-3x_2 \le -5,$$
$$-5x_1-x_2 \le -1,$$
$$x_1 \ge 0,\ x_2 \ge 0.$$

SCILAB function for solving quadratic programming problems

SCILAB provides function *quapro* for the solution of a problem composed of a quadratic objective function subjected to linear constraints. The function can be called using any of the following forms:

```
[xopt,lagr,fopt] =quapro(Q,c,C,d [,x0])
[xopt,lagr,fopt]= quapro(Q,c,C,d,xL,xU,[,x0])
[xopt,lagr,fopt]= quapro(Q,c,C,d,xL,xU, m, [,x0])
[xopt,lagr,fopt]= quapro(Q,c,C,d,xL,xU, m,x0 [,imp])
```

where the parameters *c, C, d, xL, xU, m, x0,* and *imp*, and the variables *xopt, lagr,* and *fopt* are exactly the same as in function *linpro*. The only difference is the first argument *Q* which must be a square symmetric matrix.

An application of function *quapro*

We will solve the quadratic programming presented earlier which we re-write as *minimize* $f(\mathbf{x}) = ½\, \mathbf{x}^T\mathbf{Qx} + \mathbf{c}^T\mathbf{x}$, subject to $\mathbf{Gx} \le \mathbf{h}$, and $\mathbf{x}_L \le \mathbf{x} \le \mathbf{x}_U$, with

$$\mathbf{Q}=\begin{bmatrix}2 & -1\\ -1 & 3\end{bmatrix},\quad \mathbf{c}=\begin{bmatrix}2\\ 4\end{bmatrix},\quad \mathbf{G}=\begin{bmatrix}-2 & -3\\ -5 & -1\end{bmatrix},\quad \mathbf{h}=\begin{bmatrix}-5\\ -1\end{bmatrix},\quad \mathbf{x}_L=\begin{bmatrix}0\\ 0\end{bmatrix},\quad \mathbf{x}_U=\begin{bmatrix}1\times 10^{10}\\ 1\times 10^{10}\end{bmatrix}.$$

To use function *quapro* we use the values of *Q*, *c*, *xL*, and *xU*, as described above, and replace *C* with ***G*** and *d* with ***h***. The following SCILAB script produces the solution using *quapro* and shows a graphic illustrating the solution:

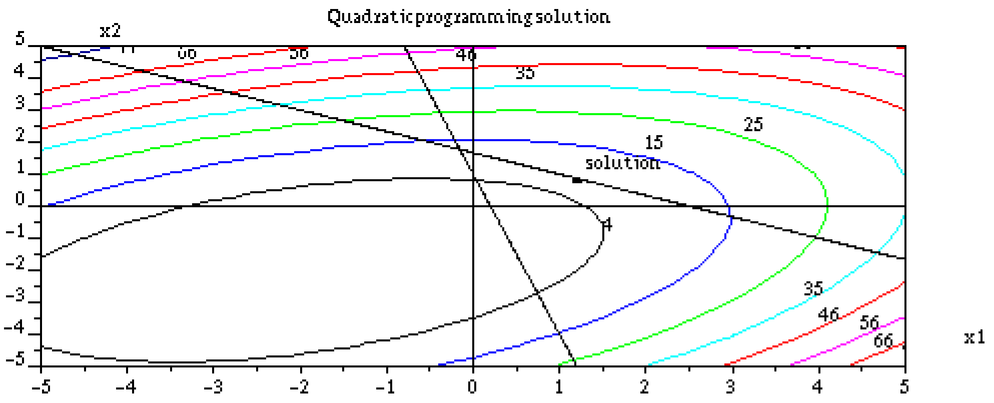
```
//Quadratic programming solution

Q = [2,-1; -1,3]; c=[2;4]; C=[-2,-3;-5,-1]; d=[-5;-1];

xL = [0;0]; xU = [1e10;1e10];

[xopt,lagr,fopt] = quapro(Q,c,C,d,xL,xU);

xopt

//Plot illustrating quadratic programming solution

deff('[w]=f(x1,x2)','w=x1^2-x1*x2+(3/2)*x2^2+2*x1+4*x2')

xx1=[-5:0.5:5]; xx2=[-5:0.5:5]; zz = feval(xx1,xx2,f);

contour(xx1,xx2,zz,10)

deff('[y1]=f1(x)','y1=(5-2*x)/3')

deff('[y2]=f2(x)','y2=1-5*x')

xxx=[-5:0.01:5];yy1=f1(xxx);yy2=f2(xxx);

plot2d([xxx' xxx'],[yy1' yy2'],[1 1],'011',' ',[-5 -5 5 5 ])

xpoly([-5,5],[0,0],'lines'); xpoly([0,0],[-5,5],'lines');

xfrect(xopt(1)-0.05,xopt(2)-0.05,0.1,0.05)

xstring(xopt(1)+0.1, xopt(2)+0.1, 'solution')

xtitle('Quadratic programming solution','x1','x2')
```

To run the script (called *Quad1*), assumed stored in the current working directory, use:

```
-->exec('Quad1')
```

The results are:

```
-->xopt
xopt  =

!    1.2142857 !
!      .8571429 !
```



The contour plot suggest that the absolute minimum of the function occurs somewhere in the third quadrant (x_1<0 and x_2<0). The feasible domain is located in the first quadrant (x_1>0, x_2>0) and to the right and above the two lines shown. The optimal solution is the point labeled 'solution' in the graph.

SCILAB non-linear programming functions

SCILAB provides functions *optim* and *leastsq* to obtain the value of ***x*** that minimizes a non-linear function *f(z)* allowing only constrains on the lower and upper limits of the design variables, if any.

Function *optim*

The simplest call to function *optim* is

[fopt,xopt]=optim(costf,x0)

where *xopt* is the value of the design variables vector ***x*** that minimizes function *costf*. The value of the function at *x=xopt* is given by *fopt*. An initial guess to the solution, *x0*, is provided as argument to the function.

The function *costf* must be defined so that the general call to this function is

[f,g,ind]=costf(x,ind)

The function *costf* is a function which returns *f = f(x),* i.e., the value of cost (or objective) function at *x*, and *g* = gradient vector of cost function at x. The variable *ind* is used by function *optim*. If *ind=1*, function *costf* produces no return. If *ind=2*, function *costf* must provide *f* as a result. If *ind=3*, function *costf* must return the gradient *g*, and if *ind=4,* function *costf* must return both *f* and *g*.

<u>Application of function *optim*</u>

We will try to minimize the function

$$f(x_1,x_2,x_3) = \{(x_1-a_1)^2 + (x_2-a_2)^2 + (x_3-a_3)^2\}^{1/2},$$

whose gradient is given by

$$g(x_1,x_2,x_3) = [(x_1-a_1),(x_2-a_2),(x_3-a_3)]/\{(x_1-a_1)^2 + (x_2-a_2)^2 + (x_3-a_3)^2\}^{1/2}.$$

We use the expressions in functions *f* and *g* to define function *costf*, as follows:

```
-->deff('[f,g,ind]=costf(x,ind)', ['f=norm(x-a)','g=(x-a)/f'])
```

The simplest call possible for function *optim* is:

```
-->[fopt,xopt] = optim(costf,x0)
 xopt  =

!   2. !
!   3. !
! - 2. !
 fopt  =

    1.271E-12
```

Specifying the method of solution

Function *optim* can use a third argument specifying the algorithm used for the solution. Possible values for this argument are *'qn'* for *quasi-Newton method*, *'gc'* for *conjugate gradient*, and *'nd'* for *non-differentiable method.* The following call to *optim* uses the quasi-Newton method:

```
-->[fopt,xopt] = optim(costf,x0,'qn')
 xopt  =

!   2. !
!   3. !
! - 2. !
 fopt  =

    1.271E-12
```

Obtaining the function gradient at the optimum point

The call to function *optim* can be modified to include a third value *gopt* representing the gradient of the function at the optimal value *xopt*:

```
-->[fopt,xopt,gopt]=optim(costf,x0)
 gopt  =

! -  .2181594 !
! -  .8261763 !
!    .5194605 !
 xopt  =

!   2. !
!   3. !
! - 2. !
 fopt  =
    1.271E-12
```

Obtaining a work vector for quasi-Newton method

You can also get back from function *optim* a work array (*work*) that can be used for re-starting calculations in a quasi-Newton method:

```
-->[fopt,xopt,gopt,work]=optim(costf,x0)
 work  =

         column 1 to 5

!   2.838E+11     .1519917     .3348070    6.786E+11  -  .0552042 !

         column  6 to 12

!   3.772E+11    0.    0.    0.     .2181594     .7930178  -  .5487239 !

         column 13 to 19

!   2.    3.  - 2.  -  .2181594  -  .8261763     .5194605     .0407068 !

         column 20 to 24

!    .5886515    1.0947067  -  .2181594  -  .8261763     .5194605 !


 gopt  =

! -  .2181594 !
! -  .8261763 !
!    .5194605 !
 xopt  =

!   2. !
!   3. !
! - 2. !
 fopt  =

    1.271E-12
```

Placing simple constraints on the design variables

Function *optim* allow for constraints to be placed on the design variables by adding as third argument the string *'b'* followed by two arrays representing the lower bound and upper bound values of the variables. For example, if we restrict the variables to $x_1 \geq 0$, $x_2 \geq 0$,, and $x_3 \geq 0$, we can use:

```
-->[fopt,xopt,gopt]=optim(costf,'b',[0;0;0],[1e10;1e10;1e10],x0)
 gopt  =

!    .1960336 !
!    .9804776 !
!    .0153146 !
```

```
 xopt  =

!   27.60096  !
!   131.04521 !
!   0.        !
 fopt  =

    2.5495098
```

Function *leastsq*

The simplest call to function *leastsq* is

[fopt,xopt]=leastsq([imp,] f [,Df],x0)

where *imp* is an optional argument may take the values imp=0 to allow only error reporting, imp=1 for initial and final reports, imp=2 adds a report per iteration, and imp>2 add reports on linear search. Function *f* is the function to be minimized, *Df* (optional argument) is the gradient of the function to be minimized. The argument *x0*, and the returned values *fopt* and *xopt* are the same as in function *optim.*

Application of function *leastsq*

As an example, we minimize the function

$$f(x_1,x_2) = x_1^3 - 2x_1x_2x_3 + x_2^2 + 3x_3,$$

whose gradient is

$$Df = [3x_1^2 - 2x_2x_3,\ -2x_1x_3 + 2x_2,\ -2x_1x_2 + 3].$$

First, we define the functions *f* and *Df* and an initial value of x:

```
-->deff('[w]=f(x)','w=x(1)^3-2*x(1)*x(2)*x(3)+x(2)^2+3*x(3)')

-->deff('[w]=Df(x)','w=[3*x(1)^2-2*x(2)*x(3),-2*x(1)*x(3)+2*x(2),-
2*x(1)*x(2)+3]')

-->x0=[1;1;1];
```

The simplest call to function *leastsq* is:

```
-->[fopt,xopt]=leastsq(f,x0)
 xopt  =

! -  .0370534 !
!    .8658123 !
! -  .2446280 !


 fopt  =

    0.
```

Function *leastsq* using the function's gradient

The following call to *leastsq* includes the gradient function:

```
-->[fopt,xopt]=leastsq(f,Df,x0)
 xopt  =

! -  .0370533 !
!    .8658122 !
! -  .244628  !
 fopt  =

    0.
```

Function *leastsq* with different output options

The following calls to the function introduce the argument *imp* with different values. The descriptions following the function call will not show up, in general, in the SCILAB screen, but they will be sent to a text file if the function *diary* has activated one such file for output.

```
-->[fopt,xopt]=leastsq(1,f,x0)
1entree dans n1qn1. dimension du probleme   3,   de zm    24
 mode 1   eps=   .22E-15   niter= 100 nsim=  100 imp=  1
 n1qn1   9 iters    11 simuls   f=   .0000000E+00
 sortie de n1qn1. norme gradient carre =   .0000000E+00
 norm of projected gradient lower than   0.0000000E+00

 xopt  =

! -  .0370534 !
!    .8658123 !
! -  .2446280 !
 fopt  =

    0.
```

With *imp=2* information is provided for each iteration in the solution:

```
-->[fopt,xopt]=leastsq(2,f,x0)
1entree dans n1qn1. dimension du probleme   3,   de zm    24
 mode 1   eps=   .22E-15   niter= 100 nsim=  100 imp=  2
 n1qn1   1 iters     1 simuls   f=   .9000000E+01
 n1qn1   2 iters     3 simuls   f=   .2559197E+01
 n1qn1   3 iters     5 simuls   f=   .1346007E-02
 n1qn1   4 iters     6 simuls   f=   .4660886E-03
 n1qn1   5 iters     7 simuls   f=   .5002594E-07
 n1qn1   6 iters     8 simuls   f=   .1879045E-11
 n1qn1   7 iters     9 simuls   f=   .7546736E-20
 n1qn1   8 iters    10 simuls   f=   .1232595E-31
 n1qn1   9 iters    11 simuls   f=   .0000000E+00
 sortie de n1qn1. norme gradient carre =   .0000000E+00
 norm of projected gradient lower than   0.0000000E+00

 xopt  =

! -  .0370534 !
!    .8658123 !
! -  .2446280 !
 fopt  =

    0.
```

Function *leastsq* returning function gradient at optimum point

The following call to function *leastsq* provides not only the optimal solution and the value of the function at that point, but also the gradient of the function at the optimal point:

```
-->[fopt,xopt,gopt] = leastsq(f,x0)
gopt  =

!   0. !
!   0. !
!   0. !
 xopt  =

! -  .0370534 !
!    .8658123 !
! -  .2446280 !
 fopt  =

    0.
```

Specifying the method of solution for function *leastsq*

You can specify the method to be used, for example, for a quasi-Newton method use:

```
-->[fopt,xopt] = leastsq(f,x0,'qn')
 xopt  =

! -  .0370534 !
!    .8658123 !
! -  .2446280 !
 fopt  =

    0.
```

Function *leastsq* with upper and lower boundary constraints for the design variables

Restrictions on the upper and lower limits of the design variables can be specified as in the following call to the function:

```
-->[fopt,xopt] = leastsq(1,f,Df,'b',[0;0;0],[10;10;10],x0)
 *********** qnbd ****************
 qnbd : pb dans ajour. mode= -1
 qnbd : indqn=       8
 stop during calculation of estimated hessian

 xopt  =

!   0. !
!   1. !
!   0. !
```

```
 fopt  =

    1.
```

In the following call to function *leastsq* we specify the quasi-Newton method of solution:

```
-->[fopt,xopt] = leastsq(1,f,Df,'b',[0;0;0],[10;10;10],x0,'qn')
 *********** qnbd ****************
 qnbd : pb dans ajour. mode= -1
 qnbd : indqn=       8
 stop during calculation of estimated hessian

 xopt  =

!   0. !
!   1. !
!   0. !
 fopt  =

    1.
```

Exercises

[1]. Minimize the objective function

$$C = 3.6x_1+3.08x_2+2.6x_3+2.7x_4$$

subject to the constraints:

$$x_1 + x_2 \le 240$$
$$x_3 + x_4 \le 300$$
$$x_1 + x_3 \le 200$$
$$x_2 + x_4 \le 200$$

with the design variables being $x_i \ge 0$, for $i = 1,2,3,4$.

[2]. Minimize the objective function

$$C = 3.5x_1 + 3.0x_2 + 6x_3 + 4.8x_4 + 1.8x_5 + x_6$$

subject to the constraints

$$x_1 + x_2 = 800$$
$$x_3 + x_4 = 500$$
$$x_5 + x_6 = 1500$$
$$5x_1 + 6x_3 + 3x_5 \le 6000$$
$$5x_2 + 6x_4 +3x_6 \le 8000$$

with the design variables being $x_i \ge 0$, for $i = 1,2,\ldots,6$.

[3]. Minimize the objective function

$$C = 0.7x_1 + x_2 + 0.6x_3 + 0.6x_4 + 0.8x_5 + x_6$$

subject to the constraints

$$x_1 + x_4 = 4000$$
$$x_2 + x_5 = 3000$$
$$x_3 + x_6 = 4500$$
$$x_1 + x_3 + x_3 \leq 6000$$
$$x_4 + 6x_5 + x_6 \leq 8000$$

with the design variables being $x_i \geq 0$, for $i = 1,2,\ldots,6$.

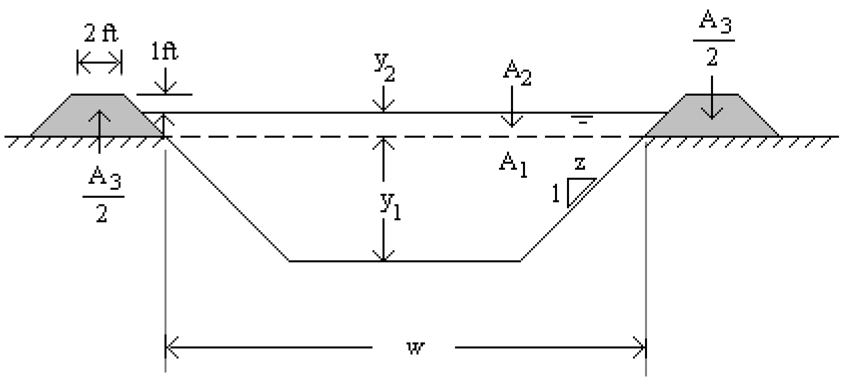
[4]. It is required to build an open-top parallelepipedal container for the transportation of sand in a construction operation. The construction costs for building the container are:

- Sides: $50/ft^2
- Ends: $60/ft^2
- Bottom: $90/ft^2

Salvage value = 10% of construction cost
Useful life = 15 years
Yearly maintenance = $15/ft^2 of outside surface area
Minimum volume needed = 150 ft^3
Interest rate = 8.5% per year

The problem is to determine the container dimensions (x_1,x_2,x_3) for minimum cost. Formulate the problem and find the container dimensions.

[5]. It is necessary to design a trapezoidal cross-section channel with a flow cross-sectional area of *150 ft*^{2}. To minimize the construction costs the amount of excavated material should be equal to the material used to build the dykes of area $A_3/2$, each, on both ends of the cross-section. Formulate the problem to minimize the excavated area A_1 if $z = 1$. Notice that the design allows for a *1-ft* freeboard elevation and that the crest of the dykes is *2-ft* long. Solve the problem for the values of y_1, y_2, and w that minimize A_1.



[6]. Minimize the objective function

$$C = (x_1-3)^2 + (x_2-3)^2$$

subject to $x_1 + x_2 \leq 4$, $x_1 \geq 0$, and $x_2 \geq 0$.

[7]. Maximize the objective function

$$C = x_1 + x_2 + 2x_3$$

subject to $3x_2 - 2x_3 = 6$, $1 \leq x_1 \leq 4$, $x_2 \geq 0$, and $-1 \leq x_3 \leq 2$.

[8]. Identify a matrix associated with the quadratic form

$$Q = x_1^2 + 4x_1x_2 + 2x_1x_3 - 7x_2^2 - 6x_2x_3 + 5x_3^2$$

and determine the unconstrained minimum or maximum for this function.

[9]. Minimize the objective function

$$C = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 2$$

subject to $-2x_1 - x_2 + 4 \leq 0$.

[10]. Maximize the objective function

$$C = 2x_1 + 5x_2 - 4.5x_3 + 1.5x_4$$

subject to

$$\begin{gathered} 5x_1 + 3x_2 + 1.5x_3 \leq 8 \\ 1.8x_1 - 6x_2 + 4x_3 + x_4 \geq 3 \\ -3.6x_1 + 8.2x_2 + 7.5x_3 + 5x_4 = 15 \\ x_i \geq 0,\ i=1,2,3,4. \end{gathered}$$

[11]. Minimize the objective function

$$C = 8x_1 - 3x_2 + 15x_3$$

subject to

$$\begin{gathered} 5x_1 - 1.8x_2 - 3.6x_3 \geq 2 \\ 3x_1 + 6x_2 + 8.2x_3 \geq 5 \\ -3.6x_1 + 8.2x_2 + 7.5x_3 + 5x_4 = 15 \\ -3.6x_1 + 8.2x_2 + 7.5x_3 + 5x_4 = 15 \\ x_1 \geq 0,\ x_2 \geq 0, \\ \text{no restriction on sign for } x_3 \end{gathered}$$

REFERENCES (for all SCILAB documents at InfoClearinghouse.com)

Abramowitz, M. and I.A. Stegun (editors), 1965,"*Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*," Dover Publications, Inc., New York.

Arora, J.S., 1985, "*Introduction to Optimum Design*," Class notes, The University of Iowa, Iowa City, Iowa.

Asian Institute of Technology, 1969, "*Hydraulic Laboratory Manual*," AIT - Bangkok, Thailand.

Berge, P., Y. Pomeau, and C. Vidal, 1984,"*Order within chaos - Towards a deterministic approach to turbulence*," John Wiley & Sons, New York.

Bras, R.L. and I. Rodriguez-Iturbe, 1985,"*Random Functions and Hydrology*," Addison-Wesley Publishing Company, Reading, Massachussetts.

Brogan, W.L., 1974,"*Modern Control Theory*," QPI series, Quantum Publisher Incorporated, New York.

Browne, M., 1999, "*Schaum's Outline of Theory and Problems of Physics for Engineering and Science*," Schaum's outlines, McGraw-Hill, New York.

Farlow, Stanley J., 1982, "*Partial Differential Equations for Scientists and Engineers*," Dover Publications Inc., New York.

Friedman, B., 1956 (reissued 1990), "*Principles and Techniques of Applied Mathematics*," Dover Publications Inc., New York.

Gomez, C. (editor), 1999, "*Engineering and Scientific Computing with Scilab*," Birkhäuser, Boston.

Gullberg, J., 1997, "*Mathematics - From the Birth of Numbers*," W. W. Norton & Company, New York.

Harman, T.L., J. Dabney, and N. Richert, 2000, "*Advanced Engineering Mathematics with MATLAB® - Second edition*," Brooks/Cole - Thompson Learning, Australia.

Harris, J.W., and H. Stocker, 1998, "*Handbook of Mathematics and Computational Science*," Springer, New York.

Hsu, H.P., 1984, "*Applied Fourier Analysis*," Harcourt Brace Jovanovich College Outline Series, Harcourt Brace Jovanovich, Publishers, San Diego.

Journel, A.G., 1989, "*Fundamentals of Geostatistics in Five Lessons*," Short Course Presented at the 28th International Geological Congress, Washington, D.C., American Geophysical Union, Washington, D.C.

Julien, P.Y., 1998,"*Erosion and Sedimentation*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Keener, J.P., 1988, "*Principles of Applied Mathematics - Transformation and Approximation*," Addison-Wesley Publishing Company, Redwood City, California.

Kitanidis, P.K., 1997,"*Introduction to Geostatistics - Applications in Hydogeology*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Koch, G.S., Jr., and R. F. Link, 1971, "*Statistical Analysis of Geological Data - Volumes I and II*," Dover Publications, Inc., New York.

Korn, G.A. and T.M. Korn, 1968, "*Mathematical Handbook for Scientists and Engineers*," Dover Publications, Inc., New York.

Kottegoda, N. T., and R. Rosso, 1997, "*Probability, Statistics, and Reliability for Civil and Environmental Engineers*," The Mc-Graw Hill Companies, Inc., New York.

Kreysig, E., 1983, "*Advanced Engineering Mathematics - Fifth Edition*," John Wiley & Sons, New York.

Lindfield, G. and J. Penny, 2000, "*Numerical Methods Using Matlab®,*" Prentice Hall, Upper Saddle River, New Jersey.

Magrab, E.B., S. Azarm, B. Balachandran, J. Duncan, K. Herold, and G. Walsh, 2000, "*An Engineer's Guide to MATLAB®*", Prentice Hall, Upper Saddle River, N.J., U.S.A.

McCuen, R.H., 1989,"*Hydrologic Analysis and Design - second edition*," Prentice Hall, Upper Saddle River, New Jersey.

Middleton, G.V., 2000, "*Data Analysis in the Earth Sciences Using Matlab®,"* Prentice Hall, Upper Saddle River, New Jersey.

Montgomery, D.C., G.C. Runger, and N.F. Hubele, 1998, "*Engineering Statistics*," John Wiley & Sons, Inc.

Newland, D.E., 1993, "*An Introduction to Random Vibrations, Spectral & Wavelet Analysis - Third Edition*," Longman Scientific and Technical, New York.

Nicols, G., 1995, "*Introduction to Nonlinear Science*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Parker, T.S. and L.O. Chua, , "*Practical Numerical Algorithms for Chaotic Systems*," 1989, Springer-Verlag, New York.

Peitgen, H-O. and D. Saupe (editors), 1988, "*The Science of Fractal Images*," Springer-Verlag, New York.

Peitgen, H-O., H. Jürgens, and D. Saupe, 1992, "*Chaos and Fractals - New Frontiers of Science*," Springer-Verlag, New York.

Press, W.H., B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, 1989, "*Numerical Recipes - The Art of Scientific Computing (FORTRAN version)*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Raghunath, H.M., 1985, "*Hydrology - Principles, Analysis and Design*," Wiley Eastern Limited, New Delhi, India.

Recktenwald, G., 2000, "*Numerical Methods with Matlab - Implementation and Application*," Prentice Hall, Upper Saddle River, N.J., U.S.A.

Rothenberg, R.I., 1991, "*Probability and Statistics*," Harcourt Brace Jovanovich College Outline Series, Harcourt Brace Jovanovich, Publishers, San Diego, CA.

Sagan, H., 1961,"*Boundary and Eigenvalue Problems in Mathematical Physics*," Dover Publications, Inc., New York.

Spanos, A., 1999,"*Probability Theory and Statistical Inference - Econometric Modeling with Observational Data*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Spiegel, M. R., 1971 (second printing, 1999), "*Schaum's Outline of Theory and Problems of Advanced Mathematics for Engineers and Scientists*," Schaum's Outline Series, McGraw-Hill, New York.

Tanis, E.A., 1987, "*Statistics II - Estimation and Tests of Hypotheses*," Harcourt Brace Jovanovich College Outline Series, Harcourt Brace Jovanovich, Publishers, Fort Worth, TX.

Tinker, M. and R. Lambourne, 2000, "*Further Mathematics for the Physical Sciences*," John Wiley & Sons, LTD., Chichester, U.K.

Tolstov, G.P., 1962, "*Fourier Series*," (Translated from the Russian by R. A. Silverman), Dover Publications, New York.

Tveito, A. and R. Winther, 1998, "*Introduction to Partial Differential Equations - A Computational Approach*," Texts in Applied Mathematics 29, Springer, New York.

Urroz, G., 2000, "*Science and Engineering Mathematics with the HP 49 G - Volumes I & II*", www.greatunpublished.com, Charleston, S.C.

Urroz, G., 2001, "*Applied Engineering Mathematics with Maple*", www.greatunpublished.com, Charleston, S.C.

Winnick, J., , "*Chemical Engineering Thermodynamics - An Introduction to Thermodynamics for Undergraduate Engineering Students*," John Wiley & Sons, Inc., New York.