5.4 or 8.3 Search Direction Determination:
Steepest Descent (gradient) Method (تندترین کاهش)

The basic requirement for search direction ***d*** is that the cost function be reduced if we make a small move along ***d*** (The descent condition of $c^{(k)}.d^{(k)}<0$ be satisfied). This will be called the descent direction.

Several methods are available for determining a descent direction for unconstrained optimization problems.

The steepest descent method or the gradient method is the simplest, the oldest, and probably the best known numerical method for unconstrained optimization.

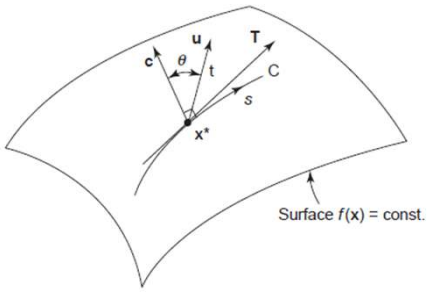
 1/69

Properties of the Gradient Vector

Property 1 The gradient vector $\vec{c}$ of a function $f(x_1,x_2,...,x_n)$ at the point $x^*=(x_1^*,x_2^*,...,x_n^*)$ is orthogonal (normal) to the tangent hyperplane for the surface $f(x_1,x_2,...,x_n)$=constant.

Property 2 The gradient represents a direction of maximum rate of increase for the function $f(x)$ at the point x^*.

Property 3 The maximum rate of change of $f(x)$ at any point x^* is the magnitude of the gradient vector.



Gradient vector for the surface $f(x)$ = constant at the point x^*.

2/69

The philosophy of the method is to find the direction $\vec{d}$ at the current iteration in which the cost function *f(x)* decreases most rapidly, at least locally.

The most important property is that the gradient at a point ***x*** points in the direction of maximum increase in the cost function.

Thus the direction of maximum decrease is opposite to that, i.e., negative of the gradient vector.

Any small move in the negative gradient direction will result in the maximum local rate of decrease in the cost function.

The negative gradient vector then represents a direction of steepest descent for the cost function and is written as

$$\vec{d} = -\vec{c}, \quad \text{or} \quad d_i = -c_i = -\frac{\partial f}{\partial x_i}; \ \text{i=1 to n}$$

The Steepest Descent Algorithm

Step 1. Estimate a starting design $x^{(0)}$ and set the iteration counter k=0. Select a convergence parameter ε>0.

Step 2. Calculate the gradient of *f(x)* at the point $x^{(k)}$ as $c^{(k)}=\nabla f(x^{(k)})$.

Step 3. Calculate $\left\|c^{(k)}\right\|$. If $\left\|c^{(k)}\right\| < \varepsilon$, then stop the iterative process because $x^*=x^{(k)}$ is a minimum point. Otherwise, continue.

Step 4. Let the search direction at the current point *x(k)* be $d^{(k)}=-c^{(k)}$.

Step 5. Calculate a step size α_k that minimizes $f(x^{(k)}+\alpha_k d^{(k)})$. Any one-dimensional search algorithm may be used to determine α_k.

Step 6. Update the design as $x^{(k+1)}=x^{(k)}+\alpha_k d^{(k)}$. Set $k=k+1$, and go to Step 2.

9.2.2 Orthogonality of Steepest Descent Directions

The successive directions of steepest descent are normal to one another, i.e., $(c^{(k)}.c^{(k+1)})=0$.

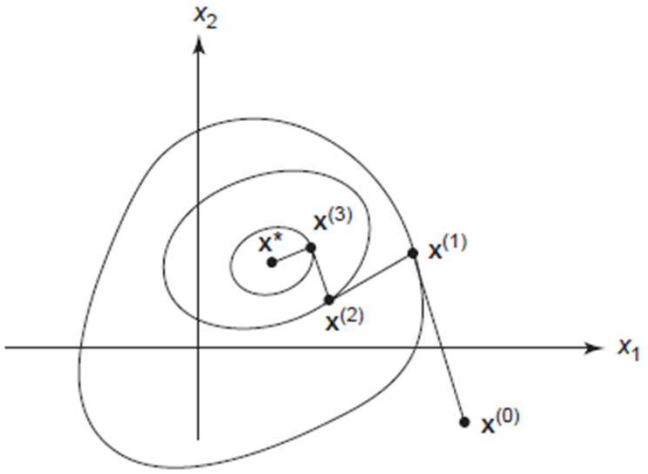
Proof: The step size determination problem is to compute α_k that minimizes $f(x^{(k)}+\alpha_k d^{(k)})$.
The necessary condition for this is $df/d\alpha_k=0$. Using the chain rule of differentiation, we get

$$\frac{df(x^{(k+1)})}{d\alpha_k}=\left[\frac{\partial f(x^{(k+1)})}{\partial x}\right]^T\frac{\partial x^{(k+1)}}{\partial \alpha_k}=0 \qquad (9.6a)$$

Since $\frac{\partial f(x^{(k+1)})}{\partial x}=c^{(k+1)}$ and $\frac{\partial (x^{(k+1)})}{\partial \alpha_k}=\frac{\partial}{\partial \alpha_k}(x^{(k)}+\alpha_k d^{(k)})=d^{(k)}$ (9.6c)

Therefore:

$$(c^{(k+1)}.d^{(k)})=0 \quad \text{or} \quad (c^{(k+1)}.c^{(k)})=0 \qquad (9.6b)$$



Orthogonal steepest descent paths.

EXAMPLE 8.4 Use of Steepest Descent Algorithm

$$\min f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1x_2 \quad \text{(a)}$$

using the steepest descent method starting from the point (1,0).

Solution. To solve the problem, we follow the steps of the steepest descent algorithm.

1. The starting design is given as $x^{(0)}=(1,0)$.
2. $c^{(0)} = (2x_1-2x_2, 2x_2-2x_1)=(2,-2)$.
3. $\|c^{(0)}\| = 2\sqrt{2} \neq 0$
4. Set $d^{(0)} = -c^{(0)}=(-2,2)$.
5. Calculate α to minimize $f(x^{(0)}+\alpha d^{(0)})$ where $x^{(0)}+\alpha d^{(0)}=(1-2\alpha,2\alpha)$:

$$f(x^{(0)} + \alpha d^{(0)}) = (1-2\alpha)^2 + (2\alpha)^2 - 2(1-2\alpha)(2\alpha)$$
$$= 16\alpha^2 - 8\alpha + 1 = f(\alpha) \quad \text{(b)}$$

Since this is a simple function of α, we can use necessary and sufficient conditions to solve for the optimum step length.

In general, a numerical one-dimensional search will have to be used to calculate α.

Using the analytic approach to solve for optimum α, we get

$$\frac{df(\alpha)}{d\alpha} = 0; \quad (32\alpha - 8) = 0 \quad \text{or} \quad \alpha_0 = 0.25 \quad \text{(c)}$$

$$\frac{d^2 f(\alpha)}{d\alpha^2} = 32 > 0. \quad \text{(d)}$$

Therefore, the sufficiency condition for a minimum for $f(\alpha)$ is satisfied.

6. Updating the design

$(x^{(0)}+\alpha_0 d^{(0)})$: $x_1^{(1)}=1-0.25(2)=0.5$, $x_2^{(1)}=0+0.25(2)=0.5$

Solving for $c^{(1)}$ from the expression in Step 2, we see that $c^{(1)}=(0,0)$, which satisfies the stopping criterion.

Therefore, (0.5, 0.5) is a minimum point for $f(x)$ and $f^*=0$.

EXAMPLE 8.5 Use of Steepest Descent Algorithm

Minimize $f(x_1,x_2,x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$ (a) using the steepest descent method with a starting design as (2,4,10). Select the convergence parameter ε as 0.005. Perform a line search by golden section search with initial step length δ=0.05 and an accuracy of 0.0001.

Solution.

1. The starting point is set as $\boldsymbol{x}^{(0)}$=(2, 4, 10).

2. $\boldsymbol{c} = \nabla f = (2x_1+2x_2,\ 4x_2+2x_1+2x_3,\ 4x_3+2x_2)$; $\boldsymbol{c}^{(0)}$=(12,40, 48).

3. $\|c^{(0)}\| = \sqrt{4048} = 63.6 > \varepsilon$ (continue)

4. $\boldsymbol{d}^{(0)} = -\boldsymbol{c}^{(0)}$ = (-12, -40, -48).

5. Calculate α_0 by golden section search to minimize $f(\boldsymbol{x}^{(0)}+\alpha\boldsymbol{d}^{(0)})$; α_0=0.1587.

9/69

6. Update the design as $\boldsymbol{x}^{(1)}=\boldsymbol{x}^{(0)}+\alpha_0\boldsymbol{d}^{(0)}$=(0.0956,-2.348,2.381). At the new design, $\boldsymbol{c}^{(1)}$=(-4.5, -4.438, 4.828), $\|c^{(1)}\| = 7.952 > \varepsilon$.

Note that $c^{(1)}\cdot d^{(0)}$=0, which verifies the exact line search termination criterion given in Eq. (8.11): $c^{(k+1)}.d^{(k)} = 0$.

The steps in steepest descent algorithm should be repeated until the convergence criterion is satisfied.

Appendix *D* contains the computer program and user supplied subroutines FUNCT and GRAD to implement steps of the steepest descent algorithm.

The optimum results for the problem with the program are given in Table 8-2. The true optimum cost function value is 0.0 and the optimum point is (0,0,0).

Note that large numbers of iterations and function evaluations are needed to reach the optimum.

TABLE 8-2 Optimum Solution for Example 8.5 with Steepest Descent Method: $f(x_1,x_2,x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$

Starting values of design variables: 2, 4, 10

Optimum design variables:

8.04787E-03,-6.81319E-03, 3.42174E-03

Optimum cost function value: 2.473 47E-05

Norm of gradient of the cost function at optimum:

4.970 71E-03

Number of iterations: 40

Total number of function evaluations: 753

Some Drawbacks of Steepest Descent Method

1. Even if convergence of the method is guaranteed, a large number of iterations may be required for the minimization of even positive definite quadratic forms, i.e., the method can be quite slow to converge to the minimum point.
2. Information calculated at the previous iterations is not used. Each iteration is started independent of others, which is inefficient.
3. Only first-order information about the function is used at each iteration to determine the search direction.

   This is one reason that convergence of the method is slow.

   It can further deteriorate if an inaccurate line search is used.

   Moreover, the rate of convergence depends on the condition number (the condition number of a matrix is calculated as the ratio of the largest to the smallest eigenvalues of the matrix) of the Hessian of the cost function at the optimum point.

   If the condition number is large, the rate of convergence of the method is slow.

4. Practical experience with the method has shown that a substantial decrease in the cost function is achieved in the initial few iterations and then it decreases quite slowly in later iterations.

5. The direction of steepest descent (direction of most rapid decrease in the cost function) may be good in a local sense (in a small neighborhood) but not in a global sense.

Scaling of Design Variables for reducing the Condition No. and Increasing the Rate of Convergence of SD

The rate of convergence of the steepest descent method is at best linear even for a quadratic cost function.

It is possible to accelerate this rate of convergence of the steepest descent method if the condition number of the Hessian of the cost function can be reduced by scaling the design variables.

For a quadratic cost function it is possible to scale the design variables such that the condition number of the Hessian matrix with respect to the new design variables, is unity.

The steepest descent method converges in only one iteration for a positive definite quadratic function with a unit condition number.

To obtain the optimum point with the original design variables, we could then unscale the transformed design variables.

Thus the main objective of scaling the design variables is to define transformations such that the condition number of the Hessian with respect to the transformed variables is 1.

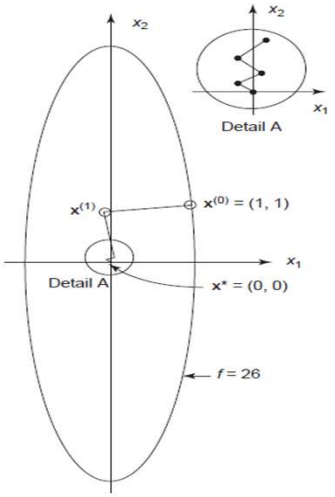
EXAMPLE 9.4 Effect of Scaling of Design Variables

Minimize $f(x_1,x_2)=25x_1^2+x_2^2$ with a starting design (1,1) by the steepest descent method.
How would you scale the design variables to accelerate the rate of convergence?

Solution. Let us solve the problem by the computer program for the steepest descent method given in Appendix *D*. The results are summarized in Table 9-1.

TABLE 9-1 Optimum Solution for Example 9.4 with Steepest Descent Method: $f(x) = 25x_1^2 + x_2^2$

Starting values of design variables:	1, 1
Optimum design variables:	−2.35450E−06, 1.37529E−03
Optimum cost function value:	1.89157E−06
Norm of gradient at optimum:	2.75310E−03
Number of iterations:	5
Number of function evaluations:	111



Note the inefficiency of the method on such a simple quadratic cost function; the method takes 5 iterations and 111 function evaluations.

FIGURE 9-6 Iteration history for Example 9.4 with the steepest descent method.

17/69

The Hessian of $f(x_1,x_2)$ is a diagonal matrix given as

$$H = \begin{bmatrix} 50 & 0 \\ 0 & 2 \end{bmatrix}$$

The condition number of the Hessian is 50/2=25 since its eigenvalues are 50 and 2. Now let us introduce new design variables y_1 and y_2 such that

$$x = Dy \quad \text{where} \quad D = \begin{bmatrix} \frac{1}{\sqrt{50}} & 0 \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

Note that, in general, we may use $D_{ii} = 1/\sqrt{H_{ii}}$, for $i = 1$ to n if the Hessian is a diagonal matrix (the diagonal elements are the eigenvalues of ***H***). The previous transformation gives

$$x_1 = y_1/\sqrt{50} \text{ and } x_2 = y_2/\sqrt{2} \qquad \text{and } f(y_1,y_2) = \frac{1}{2}(y_1^2 + y_2^2)$$

The minimum point of $f(y_1,y_2)$ is found in just one iteration by the steepest descent method compared with the five iterations for the original function since the condition number of the transformed Hessian is 1.
The optimum point is (0,0) in the new design variable space.
To obtain the minimum point in the original design space, we have to unscale the transformed design variables as

$$x_1^* = y_1/\sqrt{50} = 0 \text{ and } x_2^* = y_2/\sqrt{2} = 0$$

Thus for this example, the use of design variable scaling is quite beneficial.

EXAMPLE 9.5 Effect of Scaling of Design Variables

$$\min f(x_1,x_2) = 6x_1^2 - 6x_1x_2 + 2x_2^2 - 5x_1 + 4x_2 + 2 \quad \text{(a)}$$

with a starting design (-1,-2) by the steepest descent method. Scale the design variables to have a condition number of unity for the Hessian matrix of the function with respect to the new design variables.

Solution. Note that unlike the previous example the function f in this problem contains the cross term x_1x_2.
Therefore the Hessian matrix is not a diagonal matrix, and we need to compute its eigenvalues and eigenvectors to find a suitable scaling or transformation of the design variables.
The Hessian H of the function f is given as

$$H = \begin{bmatrix} 12 & -6 \\ -6 & 4 \end{bmatrix} \quad \text{(b)}$$

The eigenvalues of the Hessian: 0.7889 and 15.211 (condition number=15.211/0.7889=19.3).
The corresponding eigenvectors are:

For $\lambda_1 = 0.7889$

$$\begin{bmatrix} 12-0.7889 & -6 \\ -6 & 4-0.7889 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0, \begin{cases} 11.2111x_1 - 6x_2 = 0 \\ -6x_1 - 3.2111x_2 = 0 \end{cases}$$

$$x^{(1)} = \frac{1}{\sqrt{(11.2111)^2 + 36}}\begin{bmatrix} 11.2111 \\ -6 \end{bmatrix} = \begin{bmatrix} 0.8817 \\ -0.4718 \end{bmatrix}$$

$$x^{(2)} = \frac{1}{\sqrt{(3.2111)^2 + 36}}\begin{bmatrix} -6 \\ -3.2111 \end{bmatrix} = \begin{bmatrix} -0.8817 \\ 0.4718 \end{bmatrix}$$

For $\lambda_2 = 15.211$

$$\begin{bmatrix} 12-15.211 & -6 \\ -6 & 4-15.211 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0, \begin{cases} -3.2111x_1 - 6x_2 = 0 \\ -6x_1 - 11.2111x_2 = 0 \end{cases}$$

$$x^{(1)} = \frac{1}{\sqrt{(3.2111)^2 + 36}}\begin{bmatrix} -3.2111 \\ -6 \end{bmatrix} = \begin{bmatrix} -0.4718 \\ -0.8817 \end{bmatrix}$$

$$x^{(2)} = \frac{1}{\sqrt{(11.2111)^2 + 36}}\begin{bmatrix} -6 \\ -11.2111 \end{bmatrix} = \begin{bmatrix} -0.4718 \\ -0.8817 \end{bmatrix}$$

21/69

(0.4718, 0.8817) and (-0.8817, 0.4718).
Now let us define new variables y_1 and y_2 by the following transformation

$$X = Qy \quad \text{where} \quad Q = \begin{bmatrix} 0.4718 & -0.8817 \\ 0.8817 & 0.4718 \end{bmatrix} \qquad \text{(c)}$$

Note that the columns of Q are the eigenvectors of the Hessian matrix H. The transformation of variables defined by Eq. (c) gives the function in terms of y_1 and y_2 as

$$f(y_1, y_2) = 0.5(0.7889y_1^2 + 15.211y_2^2) + 1.678y_1 + 6.2957y_2 + 2 \qquad \text{(d)}$$

The condition number of the Hessian matrix in the new design variables y_1 and y_2 is still not unity. To achieve the condition number equal to unity for the Hessian, we must define another transformation of y_1 and y_2 using the eigenvalues of the Hessian matrix as

$$y = Dz \quad \text{where} \quad D = \begin{bmatrix} \frac{1}{\sqrt{0.7889}} & 0 \\ 0 & \frac{1}{\sqrt{15.211}} \end{bmatrix} \tag{e}$$

where z_1 and z_2 are the new design variables that can be calculated from the equations:

$$y_1 = \frac{z_1}{\sqrt{0.7889}} \quad \text{and} \quad y_2 = \frac{z_2}{\sqrt{15.211}} \tag{f}$$

and $f(z_1,z_2)=0.5(z_1^2+z_2^2)+1.3148z_1+1.6142z_2$

The condition number of the Hessian of $f(z_1,z_2)$ is 1.

The steepest descent method converges to the solution of $f(z_1,z_2)$ in just one iteration as (-1.3158, -1.6142). The minimum point in the original design space is found by defining the inverse transformation as $x=QDz$. This gives the minimum point in the original design space as (-1/3,-3/2).

مسائل زیر را حل کرده و تا دو هفته دیگر تحویل فرمایید:

5) 55, 66

تمرین‌های قبلی:

5) 1, 15, 18, 51

5.5 or 8.4 Search Direction Determination: Conjugate Gradient Method

The conjugate gradient method is a very simple and effective modification of the steepest descent method.

The steepest descent directions at two consecutive steps are orthogonal to each other.

This tends to slow down the steepest descent method although it is convergent.

The conjugate gradient directions are not orthogonal to each other. Rather, these directions tend to cut diagonally through the orthogonal steepest descent directions. Therefore, they improve the rate of convergence of the steepest descent method considerably.

The conjugate gradient algorithm

Step 1. Estimate a starting design as $x^{(0)}$. Set the iteration counter k=0. Select the convergence parameter ε. Calculate

$$d^{(0)}=-c^{(0)}=-\nabla f(x^{(0)}) \qquad (8.21a)$$

Check stopping criterion.
If $\|c^{(0)}\| < \varepsilon$, then stop.
Otherwise, go to Step 5.

Note that Step 1 of the conjugate gradient and the steepest descent methods is the same.

Step 2. Compute the gradient of the cost function as $c^{(k)}=\nabla f(x^{(k)})$.

Step 3. Calculate $\|c^{(k)}\|$. If $\|c^{(k)}\| < \varepsilon$, then stop; otherwise continue.

Step 4. Calculate the new conjugate direction as

$$d^{(k)} = -c^{(k)} + \beta_k d^{(k-1)}; \qquad \beta_k = \left(\|c^{(k)}\| / \|c^{(k-1)}\|\right)^2 \qquad (8.21b)$$

The conjugate gradient algorithm (Cont'd)

Step 5. Compute a step size $\alpha_k=\alpha$ to minimize $f(x^{(k)}+\alpha d^{(k)})$.

Step 6. Change the design as follows, set $k=k+1$ and go to Step 2.

$$x^{(k+1)}=x^{(k)}+\alpha_k\, d^{(k)} \qquad (8.22)$$

The conjugate gradient method should always be preferred over the steepest descent method.

EXAMPLE 8.6 Use of Conjugate Gradient Algorithm

Consider the problem solved in Example 8.5:

Minimize $f(x_1,x_2,x_3)=x_1^2+2x_2^2+2x_3^2+2x_1x_2+2x_2x_3$ (a)

Carry out two iterations of the conjugate gradient method starting from the design (2,4,10).

Solution. The first iteration of the conjugate gradient method is the same as given in Example 8.5:

$c^{(0)}$ = (12, 40, 48); $\|c^{(0)}\|=63.6$, $f(x^{(0)})$=332.0 (b)

$x^{(1)}$ = (0.0956, - 2.348, 2.381) (c)

The second iteration starts from Step 2 of the conjugate gradient algorithm:

2. $c^{(1)}$=(-4.5, - 4.438, 4.828), $f(x^{(1)})$=10.75 (d)

3. $\|c^{(1)}\|=7.952>\varepsilon$, so continue.

4. $$\beta_1=\left(\|c^{(1)}\|/\|c^{(0)}\|\right)^2=(7.952/63.3)^2=0.015633 \qquad (e)$$

$$d^{(1)} = -c^{(1)} + \beta_1 d^{(0)} = \begin{bmatrix} 4.500 \\ 4.438 \\ -4.828 \end{bmatrix} + (0.015633) \begin{bmatrix} -12 \\ -40 \\ -48 \end{bmatrix} = \begin{bmatrix} 4.31241 \\ 3.81268 \\ -5.57838 \end{bmatrix} \quad \text{(f)}$$

5. Step size in the direction $d^{(1)}$ is calculated as α=0.3156.
6. The design is updated as

$$x^{(2)} = \begin{bmatrix} 0.0956 \\ -2.348 \\ 2.381 \end{bmatrix} + \alpha \begin{bmatrix} 4.31241 \\ 3.81268 \\ -5.57838 \end{bmatrix} = \begin{bmatrix} 1.4566 \\ -1.1447 \\ 0.6205 \end{bmatrix} \quad \text{(g)}$$

Calculating the gradient at this point, we get $c^{(2)}$=(0.6238, -0.4246, 0.1926). $\|c^{(2)}\| = 0.7788 > \varepsilon$, so we need to continue the iterations.
Note that $c^{(2)} \cdot d^{(1)}$=0.

The problem is solved using the conjugate gradient method available in the IDESIGN software with ε=0.005 (Arora and Tseng, 1987a,b).
Table 8-3 summarizes performance results for the method.

It can be seen that a very precise optimum is obtained in only 4 iterations and 10 function evaluations.
Comparing these with the steepest descent method results given in Table 8-2, we conclude that the conjugate gradient method is superior for this example.

TABLE 8-3 Optimum Solution for Example 8.6 with the Conjugate Gradient Method: $f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$

Starting values of design variables:	2, 4, 10
Optimum design variables:	−6.4550E−10, −5.8410E−10, 1.3150E−10.
Optimum cost function value:	6.8520E−20.
Norm of the gradient at optimum:	3.0512E−05.
Number of iterations:	4
Number of function evaluations:	10

TABLE 8-2 Optimum Solution for Example 8.5 with Steepest Descent Method: $f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$

Starting values of design variables: 2, 4, 10
Optimum design variables: 8.04787E−03, −6.81319E−03, 3.42174E−03
Optimum cost function value: 2.473 47E−05
Norm of gradient of the cost function at optimum: 4.970 71E−03
Number of iterations: 40
Total number of function evaluations: 753

The true optimum cost function value is 0.0 and the optimum point is (0, 0, 0).

30/69

Geometric interpretation of the Steepest descent and Conjugate gradient method (Example from Vanderplaats)

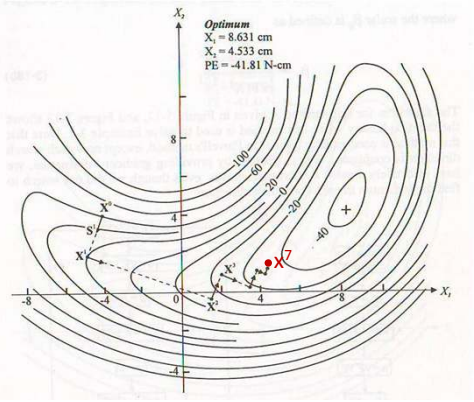


Figure 3-11 Steepest descent method

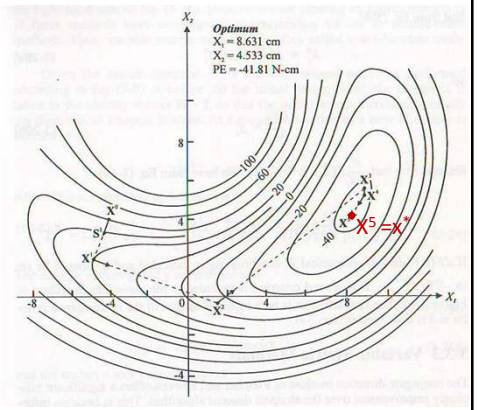


Figure 3-13 Conjugate direction method of Fletcher Reeves

مسائل زیر را حل کرده و تا دو هفته دیگر تحویل فرمایید:

5) 70, 81 (with C or Matlab)

تمرین‌های قبلی:

5) 1, 15, 18, 51, 55, 66

9.4 Search Direction Determination: Newton's Method

Basic idea: Use a second-order Taylor's expansion of the function about the current design point. This gives a quadratic expression for the change in design Δx.

The necessary condition for minimization of this function then gives an explicit calculation for design change.

$$f(x+\Delta x)=f(x)+c^T\Delta x+0.5\Delta x^T H\Delta x \qquad (9.7)$$

Equation (9.7) is a quadratic function in terms of Δx.

The theory of convex programming problems in Chapter 4 guarantees that if H is positive semidefinite, then there is a Δx that gives a global minimum for the function of Eq. (9.7).

In addition, if H is positive definite, then the minimum for Eq. (9.7) is unique. Writing optimality conditions $[\partial f/\partial(\Delta x)=0]$ for the function of Eq. (9.7),

$$c+H\Delta x=0 \qquad (9.8)$$

Assuming H to be nonsingular, we get an expression for Δx as

$$\Delta x=-H^{-1}c \qquad (9.9)$$

Using this value for Δx, the design is updated as

$$x^{(1)}=x^{(0)}+\Delta x \qquad (9.10)$$

Since Eq. (9.7) is just an approximation for f at the point $x^{(0)}$, $x^{(1)}$ will probably not be the precise minimum point of $f(x)$.

Therefore, the process will have to be repeated to obtain improved estimates until the minimum is reached.

Each iteration of Newton's method requires computation of the Hessian of the cost function.

Since it is a symmetric matrix, it needs computation of $n(n+1)/2$ second-order derivatives of $f(x)$ (recall that n is the number of design variables).

This can require considerable computational effort.

9.4.2 Modified Newton's Method

Note that the classical Newton's method does not have a step size associated with the calculation of design change Δx in Eq. (9.9); ($\Delta x=-H^{-1}c$), i.e., step size is taken as one (step of length one is called an ideal step size or Newton's step).

Therefore, there is no way to ensure that the cost function will be reduced at each iteration; i.e., to ensure that $f(x^{(k+1)})<f(x^{(k)})$.

Thus, the method is not guaranteed to converge to a local minimum point even with the use of second order information that requires large calculations.

35/69

This situation can be corrected if we incorporate the use of a step size in the calculation of the design change Δx.
In other words, we treat the solution of Eq. (9.9) as the search direction and use any of the one-dimensional search methods to calculate the step size in the search direction.
This is called the modified Newton's method and is stated as follows.

Step 1. Make an engineering estimate for a starting design $x^{(0)}$. Set iteration counter k=0. Select a tolerance ε for the stopping criterion.

Step 2. Calculate $c_i^{(k)}=\partial f(x^{(k)})/\partial x_i$ for i=1 to n. If $\left\|c^{(k)}\right\|<\varepsilon$, stop the iterative process. Otherwise, continue.

Step 3. Calculate the Hessian matrix $H^{(k)}$ at the current point $x^{(k)}$.

Step 4. Calculate the search by solving Eq. (9.9) as

گردآوری و تنظیم: محمدحسین ابوالبشری

$$d^{(k)} = -[H^{(k)}]^{-1}c^{(k)} \qquad (9.11)$$

36/69

Note that the calculation of $d^{(k)}$ in the above equation is symbolic. For computational efficiency, the linear equation $H^{(k)}d^{(k)}=-c^{(k)}$ is solved directly instead of evaluating the inverse of the Hessian matrix.

Step 5. Update the design as $x^{(k+1)}=x^{(k)}+\alpha_k d^{(k)}$, where α_k is calculated to minimize $f(x^{(k)}+\alpha d^{(k)})$. Any one-dimensional search procedure may be used to calculate α.

Step 6. Set $k=k+1$ and go to Step 2.

It is important to note here that unless H is positive definite, the direction $d^{(k)}$ determined from Eq. (9.11) ($d^{(k)} = -[H^{(k)}]^{-1}c^{(k)}$) may not be that of descent for the cost function. To see this, we substitute $d^{(k)}$ from Eq. (9.11) into the descent condition of Eq. (8.8) ($c^{(k)}.d^{(k)}<0$) to obtain

$$-c^{(k)T}H^{-1}c^{(k)} < 0 \qquad (9.12)$$

The foregoing condition will always be satisfied if H is positive definite.

If H is negative definite or negative semidefinite, the condition is always violated.

With H as indefinite or positive semidefinite, the condition may or may not be satisfied, so we must check for it.

If the direction obtained in Step 4 is not that of descent for the cost function, then we should stop there because a positive step size cannot be determined.

Based on the foregoing discussion, it is suggested that the descent condition of Eq. (8.8) should be checked for Newton's search direction at each iteration before calculating the step size.

EXAMPLE 9.6 Use of Modified Newton's Method

Minimize $f(x)=3x_1^2+2x_1x_2+2x_2^2+7$ (a)
using the modified Newton's algorithm starting from the point (5,10). Use ε=0.0001 as the stopping criterion.

Solution. We will follow the steps of the modified Newton's method.

1. $x^{(0)}$ is given as (5, 10).
2. The gradient vector $c^{(0)}$ at the point (5,10) is given as

$$c^{(0)}=(6x_1+2x_2,\ 2x_1+4x_2)=(50,50) \quad \text{(b)}$$

$$\left\|c^{(0)}\right\| = \sqrt{50^2+50^2} = 50\sqrt{2} > \varepsilon \quad \text{(c)}$$

Therefore, the convergence criterion is not satisfied.

3. The Hessian matrix at the point (5,10) is given as

$$H^{(0)} = \begin{bmatrix} 6 & 2 \\ 2 & 4 \end{bmatrix} \quad \text{(d)}$$

Note that the Hessian does not depend on design variables and is positive definite (since its eigenvalues are 7.24 and 2.76). Therefore Newton's direction satisfies the descent condition at each iteration.

4. The direction of design change is

$$d^{(0)} = -H^{-1}c^{(0)} = \frac{-1}{20}\begin{bmatrix} 4 & -2 \\ -2 & 6 \end{bmatrix}\begin{bmatrix} 50 \\ 50 \end{bmatrix} = \begin{bmatrix} -5 \\ -10 \end{bmatrix} \quad \text{(e)}$$

5. Step size α is calculated to minimize $f(x^{(0)}+\alpha d^{(0)})$:

$$x^{(1)} = x^{(0)} + \alpha d^{(0)} = \begin{bmatrix} 5 \\ 10 \end{bmatrix} + \alpha \begin{bmatrix} -5 \\ -10 \end{bmatrix} = \begin{bmatrix} 5-5\alpha \\ 10-10\alpha \end{bmatrix} \quad \text{(f)}$$

$$\frac{df}{d\alpha} = 0; \quad \text{or} \quad \nabla f(x^{(1)}) \cdot d^{(0)} = 0 \quad \text{(g)}$$

Using the Step 2 calculations, $\nabla f(x^{(1)})$ and the dot product $\nabla f(x^{(1)}) \cdot d^{(0)}$ are calculated as

$$\nabla f(x^{(1)}) = \begin{bmatrix} 6(5-5\alpha)+2(10-10\alpha) \\ 2(5-5\alpha)+4(10-10\alpha) \end{bmatrix} = \begin{bmatrix} 50-50\alpha \\ 50-50\alpha \end{bmatrix} \quad \text{(h)}$$

$$\nabla f(x^{(1)}) \cdot d^{(0)} = (50-50\alpha, 50-50\alpha)\begin{bmatrix} -5 \\ -10 \end{bmatrix} = 0 \quad \text{(i)}$$

$$\text{or} \quad -5(50-50\alpha)-10(50-50\alpha) = 0 \quad \text{(j)}$$

Solving the preceding equation, we get α=1.

Note that the golden section search also gives α=1. Therefore,

$$x^{(1)} = \begin{bmatrix} 5-5\alpha \\ 10-10\alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{(k)}$$

The gradient of the cost function at $x^{(1)}$ is calculated as

$$c^{(1)} = \begin{bmatrix} 50-50\alpha \\ 50-50\alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{(l)}$$

Since $\|c^{(k)}\| < \varepsilon$, the Newton's method has given the solution in just one iteration.

This is because the function is a positive definite quadratic form (the Hessian of f is positive definite everywhere).

Note that the condition number of the Hessian is not 1; therefore the steepest descent method will not converge in one iteration, as was the case in Examples 9.4 and 9.5.

EXAMPLE 9.7 Use of Modified Newton's Method

Minimize $f(x)=10x_1^4-20x_1^2x_2+10x_2^2+x_1^2-2x_1+5$ (a)

using the computer program for the modified Newton's method given in Appendix *D.*

Starting point: (-1,3)

Step size determination: Golden section search with: δ=0.05

Line search accuracy=0.0001

Stopping criterion: ε=0.005

Solution. Note that *f(x)* is not a quadratic function in terms of the design variables. Thus, we cannot expect the Newton's method to converge in one iteration.

The gradient and the Hessian matrix of *f(x)* are:

$$c = \nabla f(x) = (40x_1^3 - 40x_1x_2 + 2x_1 - 2, -20x_1^2 + 20x_2) \quad \text{(b)}$$

$$H = \nabla^2 f(x) = \begin{bmatrix} 120x_1^2 - 40x_2 + 2 & -40x_1 \\ -40x_1 & 20 \end{bmatrix} \quad \text{(c)}$$

TABLE 9-2 Optimum Solution for Example 9.7 with Modified Newton's Method:

$f(x)=10x_1^4-20x_1^2x_2+10x_2^2+x_1^2-2x_1+5$

Starting point: -1, 3

Optimum design variables:
9.99880E-01, 9.99681E-01
Optimum cost function value: 4.0
Norm of gradient at optimum:
3.26883E-03
Number of iterations: 8
Number of function evaluations: 198

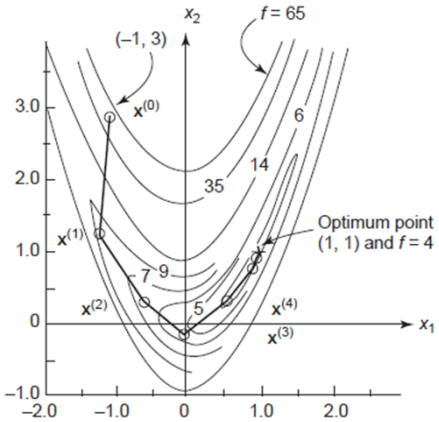


FIGURE 9-7 Iteration history for Example 9.7 with Newton's method.

It is noted (does not show) that the step size was approximately equal to one (0.957247) in the last phase of the iterative process. This is because the function resembles a quadratic function sufficiently close to the optimum point and step size is equal to unity for a quadratic function.

The Drawbacks of the Modified Newton's Method for General Applications

1. It requires calculations of second-order derivatives at each iteration, which is usually quite time consuming. In some applications it may not even be possible to calculate such derivatives.
   Also, a linear system of equations in Eq. (9.11) ($d^{(k)} = -[H^{(k)}]^{-1}c^{(k)}$) needs to be solved. Therefore, each iteration of the method requires substantially more calculations compared with the steepest descent or conjugate gradient method.
2. The Hessian of the cost function may be singular at some iterations. Thus, Eq. (9.11) cannot be used to compute the search direction.
   Also, unless the Hessian is positive definite, the search direction cannot be guaranteed to be that of descent for the cost function.
3. The method is not convergent unless the Hessian remains positive definite and a step size is calculated along the search direction to update design.
   However, the method has a quadratic rate of convergence when it converges. For a strictly convex quadratic function, the method converges in just one iteration from any starting design.

45/69

EXAMPLE 9.8 Comparison of Steepest Descent, Conjugate Gradient, and Modified Newton Methods

Minimize $f(x)=50(x_2-x_1^2)^2+(2-x_1)^2$ starting from the point (5,-5). Use the steepest descent, Newton, and conjugate gradient methods, and compare their performance.

Solution. The minimum point for the function is known as (2,4) with $f(2,4)=0$.
We use exact gradient expressions and $\varepsilon=0.005$ to solve the problem using the steepest descent and Newton's method programs given in Appendix *D* and the conjugate gradient method available in IDESIGN. Table 9-3 summarizes final results with the three methods.
For the steepest descent method, $\delta_0=0.05$ and a line search termination criterion of 0.00001 are used.
For the Newton's method, they are 0.05 and 0.0001 respectively. Golden section search is used with both methods.

46/69

TABLE 9-3 Evaluation of Three Methods for Example 9.8:

$f(x)=50(x_2-x_1^2)^2+(2-x_1)^2$

	Steepest descent	Conjugate gradient	Modified Newton
x_1	1.9941	2.0000	2.0000
x_2	3.9765	3.9998	3.9999
f	3.4564E-05	1.0239E-08	2.5054E-10
$\|c\|$	3.3236E-03	1.2860E-04	9.0357E-04
No. of function Evaluations	138236	65	349
No. of iterations	9670	22	13

Conclusion for the present example

most inefficient most efficient

****

Therefore, the conjugate gradient method is recommended for general applications.

47/69

9.4.3 Marquardt Modification

As noted before the modified Newton's method has several drawbacks that can cause numerical difficulties.

For example, if the Hessian *H* of the cost function is not positive definite, the direction found from Eq. (9.11) ($d^{(k)}=[H^{(k)}]^{-1}c^{(k)}$) may not be that of descent for the cost function. In that case, a step cannot be executed along the direction.

Marquardt (1963) suggested a modification to the direction finding process that has the desirable features of the steepest descent and Newton's methods.

It turns out that far away from the solution point, the method behaves like the steepest descent method, which is quite good there. Near the solution point, it behaves like the Newton's method, which is very effective there.

48/69

In the modified procedure, the Hessian is modified as *(H+λI),* where *λ* is a positive constant. *λ* is initially selected as a large number that is reduced as iterations progress. The search direction is computed from Eq. (9.11) as

$$d^{(k)} = -[H^{(k)} + \lambda_k I]^{-1} c^{(k)} \qquad (9.13)$$

Note that when *λ* is large, the effect of *H* essentially gets neglected and $d^{(k)}$ is essentially $-(1/\lambda)c^{(k)}$, which is the steepest descent direction with $1/\lambda$ as the step size.

As the algorithm proceeds, *λ* is reduced (i.e., step size is increased). When *λ* becomes sufficiently small, then the effect of *λI* is essentially neglected and the Newton direction is obtained from Eq. (9.13).

If the direction $d^{(k)}$ of Eq. (9.13) does not reduce the cost function, then *λ* is increased (step size is reduced) and the search direction is recomputed.

49/69

Marquardt's Algorithm

Step 1. Make an engineering estimate for starting design $x^{(0)}$. Set iteration counter k=0. Select a tolerance ε as the stopping criterion, and λ_0 as a large constant (say 1000).

Step 2. Calculate $c_i^{(k)} = \partial f(x^{(k)})/\partial x_i$ for i=1 to n. If $\|c^{(k)}\| < \varepsilon$, stop. Otherwise, continue.

Step 3. Calculate the Hessian matrix $H(x^{(k)})$.

Step 4. Calculate the search direction by solving Eq. (9.13)

$$d^{(k)} = -[H^{(k)} + \lambda_k I]^{-1} c^{(k)}.$$

Step 5. If $f(x^{(k)}+d^{(k)}) < f(x^{(k)})$, then continue (α=1 because the step size is controlled by λ; see the previous slide). Otherwise, increase λ_k (to say $2\lambda_k$), and go to Step 4.

Step 6. Reduce λ_k, say, $\lambda_{k+1} = 0.5\lambda_k$. Set $k=k+1$ and go to Step 2.

50/69

مسائل زیر را حل کرده و تا دو هفته دیگر تحویل فرمایید:

5) 82, 93(with C or Matlab)

تمرین‌های قبلی: 5) 1, 15, 18, 51, 55, 66, 70, 81

9.5 Search Direction Determination: Quasi-Newton Methods

One drawback of the steepest descent method was that the method has a poor rate of convergence because only first-order information is used.

This flaw was corrected with Newton's method where second-order derivatives were used. Newton's method has very good convergence properties.
However, the method can be inefficient because it requires calculation of *n(n+1)/2* second-order derivatives to generate the Hessian matrix (recall that *n* is the number of design variables).

For most engineering design problems, calculation of second-order derivatives may be tedious or even impossible.
Also, Newton's method runs into difficulties if the Hessian of the function is singular at any iteration.

The methods presented in this section overcome these drawbacks by generating an approximation for the Hessian matrix or its inverse at each iteration.

Only the first derivatives of the function are used to generate these approximations.

Therefore the methods have desirable features of both the steepest descent and the Newton's methods. They are called quasi-Newton methods.

9.5.1 Inverse Hessian Updating: DFP Method

This method, initially proposed by Davidon (1959), was modified by Fletcher and Powell (1963) and that method is presented here.
This is one of the most powerful methods for the minimization of a general function *f(x)*.

The method builds an approximate inverse of the Hessian of *f(x)* using only the first derivatives.

It is often called the DFP (Davidon-Fletcher-Powell) method.

Step 1. Estimate an initial design $x^{(0)}$. Choose a symmetric positive definite $n*n$ matrix $A^{(0)}$ as an estimate for the inverse of the Hessian of the cost function. In the absence of more information, $A^{(0)}=I$ may be chosen.
Also, specify a convergence parameter ε. Set k=0.
Compute the gradient vector as $c^{(0)} = \nabla f(x^{(0)})$.

Step 2. Calculate the norm of the gradient vector as $\|c^{(k)}\|$.
If $\|c^{(k)}\| < \varepsilon$, then stop the iterative process. Otherwise continue.

Step 3. Calculate the search direction as $d^{(k)}=-A^{(k)}c^{(k)}$ (a)

Step 4. Compute optimum step size $\alpha_k=\alpha$ to minimize $f(x^{(k)}+\alpha d^{(k)})$.

Step 5. Update the design as $x^{(k+1)}=x^{(k)}+\alpha_k d^{(k)}$ (b)

Step 6. Update the matrix $A^{(k)}$-approximation for the inverse of the Hessian of the cost function-as

$$A^{(k+1)} = A^{(k)} + B^{(k)} + C^{(k)} \quad \text{(c)}$$

where the correction matrices $B^{(k)}$ and $C^{(k)}$ are calculated using the quasi-Newton condition mentioned earlier, as

$$B^{(k)} = \frac{s^{(k)} s^{(k)^T}}{(s^{(k)} \cdot y^{(k)})} \qquad C^{(k)} = \frac{-z^{(k)} z^{(k)^T}}{(y^{(k)} \cdot z^{(k)})} \quad \text{(d)}$$

$$s^{(k)} = \alpha_k d^{(k)} \quad \text{(Change in design)}$$
$$y^{(k)} = c^{(k+1)} - c^{(k)} \quad \text{(Change in gradient)} \quad \text{(e)}$$

$$c^{(k+1)} = \nabla f(X^{(k+1)}) \qquad z^{(k)} = A^{(k)} y^{(k)} \quad \text{(f)}$$

Step 7. Set $k = k+1$ and go to Step 2.
Note that the first iteration of the method is the same as that for the steepest descent method.

Fletcher and Powell (1963) prove that this algorithm has the following properties:

1. The matrix $A^{(k)}$ is positive definite for all k. This implies that the method will always converge to a local minimum point, since

$$\frac{d}{dx} f\left(x^{(k)} + \alpha d^{(k)}\right)\Big|_{\alpha=0} = -c^{(k)T} A^{(k)} c^{(k)} < 0 \qquad \text{(g)}$$

as long as $c^{(k)} \neq 0$. This means that $f(x^{(k)})$ may be decreased by choosing $\alpha > 0$ if $c^{(k)} \neq 0$ (i.e., $d^{(k)}$ is a direction of descent).

2. When this method is applied to a positive definite quadratic form, $A^{(k)}$ converges to the inverse of the Hessian of the quadratic form.

9.5.2 Direct Hessian Updating: BFGS Method

It is possible to update the Hessian rather than its inverse at every iteration.

Several such updating methods are available; however, we shall present a popular method that has proven to be most effective in applications.

It is known as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, which is summarized in the following algorithm:

Step 1. Estimate an initial design $x^{(0)}$.
Choose a symmetric positive definite $n*n$ matrix $H^{(0)}$ as an estimate for the Hessian of the cost function. In the absence of more information, let $H^{(0)}=I$.
Choose a convergence parameter ε. Set k=0, and compute the gradient vector as

$$c^{(0)} = \nabla f(x^{(0)}).$$

Step 2. Calculate the norm of the gradient vector as $\left\|c^{(k)}\right\|$.
If $\left\|c^{(k)}\right\| < \varepsilon$, then stop the iterative process. Otherwise continue.

Step 3. Solve the linear system of equations $H^{(k)}d^{(k)}=-c^{(k)}$ to obtain the search direction.

Step 4. Compute optimum step size $\alpha_k=\alpha$ to minimize $f(x^{(k)}+\alpha d^{(k)})$.

Step 5. Update the design as $x^{(k+1)}=x^{(k)}+\alpha_k d^{(k)}$

Step 6. Update the Hessian approximation for the cost function as

$$H^{(k+1)} = H^{(k)} + D^{(k)} + E^{(k)} \qquad \text{(a)}$$

where the correction matrices $D^{(k)}$ and $E^{(k)}$ are given as

$$D^{(k)} = \frac{y^{(k)} y^{(k)^T}}{(y^{(k)} \cdot s^{(k)})} \qquad E^{(k)} = \frac{c^{(k)} c^{(k)^T}}{(c^{(k)} \cdot d^{(k)})} \qquad \text{(b)}$$

$$s^{(k)} = \alpha_k d^{(k)} \quad \text{(Change in design)}$$

$$y^{(k)} = c^{(k+1)} - c^{(k)} \quad \text{(Change in gradient)}$$

$$c^{(k+1)} = \nabla f(X^{(k+1)}) \qquad \text{(c)}$$

Step 7. Set $k=k+1$ and go to Step 2.

9.7 Solution of Constrained Problems Using Unconstrained Optimization Methods

The basic idea is to construct a composite function using the cost and constraint functions.

It also contains certain parameters-called the penalty parameters-that penalize the composite function for violation of constraints. The larger the violation, the larger the penalty.

Once the composite function is defined for a set of penalty parameters, it is minimized using any of the unconstrained optimization techniques.

The penalty parameters are then adjusted based on certain conditions, and the composite function is redefined and minimized. The process is continued until there is no significant improvement in the estimate for the optimum point.

61/69

Methods based on the foregoing philosophy have been generally called sequential unconstrained minimization techniques (SUMT).

All transformation methods convert this constrained optimization problem into an unconstrained problem using a transformation function of the form:

$$\phi(X, r) = f(X) + P\left[h(X), g(X), r\right] \qquad (9.14)$$

where *r* is a vector of penalty parameters and *P* is a real valued function whose action of imposing the penalty on the cost function is controlled by *r*.

The form of penalty function *P* depends on the method used.

The basic procedure is to choose an initial design estimate $x^{(0)}$ and define the function φ of Eq. (9.14).

The penalty parameters *r* are also initially selected.

The function φ is minimized for *x*, keeping *r* fixed.

Then the parameters *r* are adjusted (increased) and the procedure is repeated until no further improvement is possible.

9.7.1 Sequential Unconstrained Minimization Techniques (SUMT)

Sequential unconstrained minimization techniques consist of two different types of penalty functions.

1) Penalty function method
2) Barrier function method

Penalty function method:

The basic idea of the penalty function approach is to define the function *P* in Eq. (9.14) $\phi(X,r)=f(X)+P[h(X),g(X),r]$ in such a way that if there are constraint violations, the cost function *f(x)* gets penalized by addition of a positive value. Several penalty functions can be defined. The most popular one is called the quadratic loss function defined as

$$P[h(X),g(X),r]=r\left\{\sum_{i=1}^{p}[h_i(X)]^2+\sum_{i=1}^{m}\left[\max\left(0,g_i(X)\right)\right]^2\right\} \quad (9.15)$$

a scalar penalty parameter

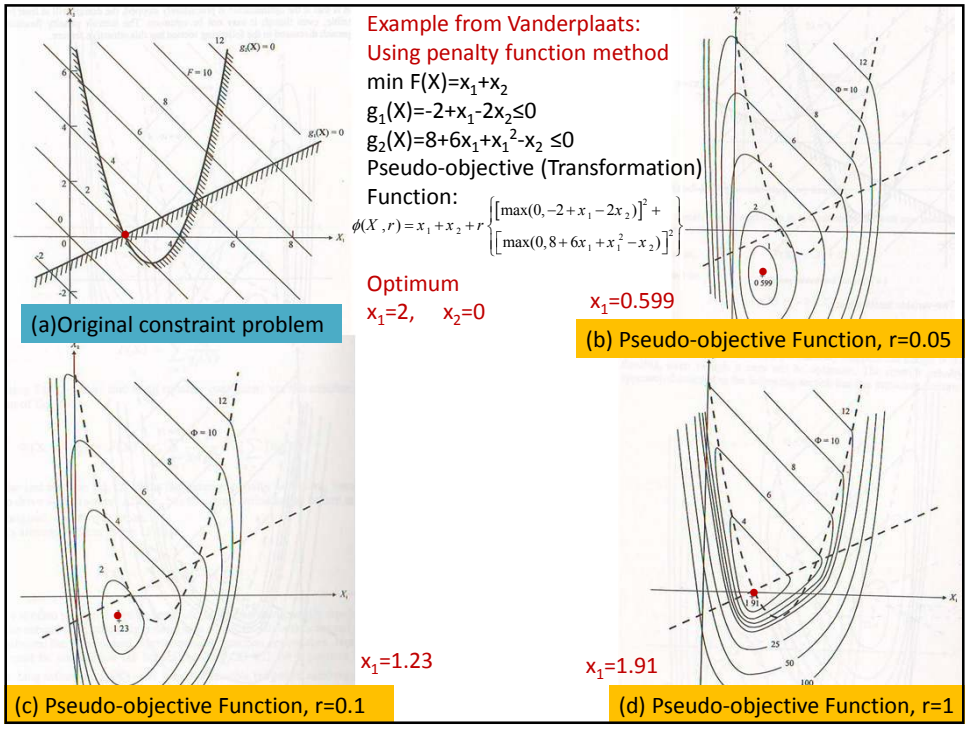
it is zero if the inequality is inactive ($g_i(x)<0$)
and it is positive if the inequality is violated

It can be seen that if the equality constraint is not satisfied, i.e., $h_i(x)\neq 0$, or the inequality is violated, i.e., $g_i(x)>0$, then Eq. (9.15) gives a positive value to the function *P*, and the cost function is penalized, $\phi(X,r)=f(X)+P[h(X),g(X),r]$ The starting design point for the method can be arbitrary. The methods based on the philosophy of penalty functions are sometimes called the exterior methods because they iterate through the infeasible region.

The advantages and disadvantages of the penalty function method are:

1. It is applicable to general constrained problems with equality and inequality constraints.
2. The starting design point can be arbitrary.
3. The method iterates through the infeasible region where the problem functions may be undefined.
4. If the iterative process terminates prematurely, the final design may not be feasible and hence not usable.

64/69



Barrier function methods:

The barrier function methods are applicable only to the inequality constrained problems. Popular barrier functions are:

1. Inverse barrier function $$P\left[g(x),r\right]=\frac{1}{r}\sum_{i=1}^{m}\frac{-1}{g_i(x)}$$

1. Log barrier function $$P\left[g(x),r\right]=\frac{1}{r}\sum_{i=1}^{m}\log\left[-g_i(x)\right]$$

x	0	0.01,	0.1	1
log(x)	∞	-2	-1	0

اگر قید نقض نشود *g(x)<0* و داخل پرانتز مثبت و تابع کمی جریمه می‌شود. ولی اگر به مرز نزدیک شود تابع جریمه سنگینی می‌شود.

These are called the barrier function methods because a large barrier is constructed around the feasible region. In fact, the function *P* becomes infinite if any of the inequalities is active. Thus, when the iterative process is started from a feasible point, it cannot go into the infeasible region because the iterative process cannot cross the huge barrier.

For both penalty function and barrier function methods, it can be shown that as $r\to\infty$, $x(r)\to x^*$, where $x(r)$ is a point that minimizes the transformed function $\varphi(x,r)$ of Eq. (9.14) and x^* is a solution of the original constrained optimization problem.

66/69

The advantages and disadvantages of the barrier function method

1. The method is applicable to inequality constrained problems only.

2. The starting design point must be feasible. It turns out, however, that the method itself can be used to determine the starting point (Haug and Arora, 1979).

3. The method always iterates through the feasible region, so if it terminates prematurely, the final design is feasible and hence usable.

The sequential unconstrained minimization techniques have certain weaknesses that are most serious when r is large. The penalty and barrier functions tend to be ill-behaved near the boundary of the feasible set where the optimum points usually lie.
There is also a problem of selecting the sequence $r^{(k)}$. The choice of $r^{(0)}$ and the rate at which $r^{(k)}$ tends to infinity can seriously affect the computational effort to find a solution.
Furthermore, the Hessian matrix of the unconstrained function becomes ill-conditioned as $r \to \infty$.

67/69

9.7.2 Multiplier (Augmented Lagrangian) Methods

To alleviate (make less sever) some of the difficulties of the methods presented in the previous section, a different class of transformation methods has been developed in the literature. These are called the multiplier or augmented Lagrangian methods.
In these methods, there is no need for the penalty parameters r to go to infinity. As a result the transformation function $\emptyset$ has good conditioning with no singularities. The multiplier methods are convergent as are the SUMTs.
That is, they converge to a local minimum starting from any point. It has been proven that they possess a faster rate of convergence than the two methods of the previous subsection.
With multiplier methods, the penalty function is given as

68/69

$$P[h(x),g(x),r,\theta]=\frac{1}{2}\sum_{i=1}^{p}r_i'(-h_i+\theta_i')^2+\frac{1}{2}\sum_{i=1}^{m}r_i[(g_i+\theta_i)^+]^2 \quad (9.18)$$

where $\theta_i>0$, $r_i>0$, and θ'_i, r'_i are parameters associated with the ith inequality and equality constraints, respectively, and $(g_i+\theta_i)^+=\max(0,g_i+\theta_i)$. If $\theta_i=\theta'_i=0$ and $r_i=r'_i=r$, then Eq. (9.18) reduces to the well-known quadratic loss function given in Eq. (9.15)

$$P[h(X),g(X),r]=r\left\{\sum_{i=1}^{p}[h_i(X)]^2+\sum_{i=1}^{m}\left[\max\left(0,g_i(X)\right)\right]^2\right\}$$

where convergence is enforced by letting $r\to\infty$.

However, the objective of the multiplier methods is to keep each r_i and r'_i finite.

The idea of multiplier methods is to start with some r_i, r'_i, θ'_i, and θ_i and minimize the transformation function of Eq. (9.14).

$$\phi(X,r)=f(X)+P[h(X),g(X),r]$$

The parameters r_i, r'_i, θ'_i, and θ_i are then adjusted using some procedures and the entire process is repeated until optimality conditions are satisfied.

69/69

مسائل زیر را حل کرده و تا دو هفته دیگر تحویل فرمایید:

5) 94, 105, 106 (with C or Matlab)

تمرین‌های قبلی:

5) 1, 15, 18, 51, 55, 66, 70, 81, 82, 93