Unconstrained Optimization 4

In this chapter we study mathematical programming techniques that are commonly used to extremize nonlinear functions of single and multiple (n) design variables subject to no constraints. Although most structural optimization problems involve constraints that bound the design space, study of the methods of unconstrained optimization is important for several reasons. First of all, if the design is at a stage where no constraints are active then the process of determining a search direction and travel distance for minimizing the objective function involves an unconstrained function minimization algorithm. Of course in such a case one has constantly to watch for constraint violations during the move in design space. Secondly, a constrained optimization problem can be cast as an unconstrained minimization problem even if the constraints are active. The penalty function and multiplier methods discussed in Chapter 5 are examples of such indirect methods that transform the constrained minimization problem into an equivalent unconstrained problem. Finally, unconstrained minimization strategies are becoming increasingly popular as techniques suitable for linear and nonlinear structural analysis problems (see Kamat and Hayduk[1]) which involve solution of a system of linear or nonlinear equations. The solution of such systems may be posed as finding the minimum of the potential energy of the system or the minimum of the residuals of the equations in a least squared sense.

4.1 Minimization of Functions of One Variable

In most structural design problems the objective is to minimize a function with many design variables, but the study of minimization of functions of a single design variable is important for several reasons. First, some of the theoretical and numerical aspects of minimization of functions of n variables can be best illustrated, especially graphically, in a one dimensional space. Secondly, most methods for unconstrained minimization of functions $f(\mathbf{x})$ of n variables rely on sequential one-dimensional minimization of the function along a set of prescribed directions, $\mathbf{s}_k$, in the multi-dimensional design space $\mathbf{R}^n$. That is, for a given design point $\mathbf{x}_0$ and a specified search direction at that point $\mathbf{s}_0$, all points located along that direction can be expressed in terms of a single variable α by

$$\mathbf{x} = \mathbf{x}_0 + \alpha \mathbf{s}_0 \,, \tag{4.1.1}$$

where α is usually referred to as the step length. The function $f(\mathbf{x})$ to be minimized can, therefore, be expressed as

$$f(\mathbf{x}) = f(\mathbf{x}_0 + \alpha \mathbf{s}_0) = f(\alpha) \ . \tag{4.1.2}$$

Thus, the minimization problem reduces to finding the value α^* that minimizes the function, $f(\alpha)$. In fact, one of the simplest methods used in minimizing functions of n variables is to seek the minimum of the objective function by changing only one variable at a time, while keeping all other variables fixed, and performing a one-dimensional minimization along each of the coordinate directions of an n-dimensional design space. This procedure is called the *univariate search* technique.

In classifying the minimization algorithms for both the one-dimensional and multi-dimensional problems we generally use three distinct categories. These categories are the zeroth, first, and second order methods. Zeroth order methods use only the value of the function during the minimization process. First order methods employ values of the function and its first derivatives with respect to the variables. Finally, second order methods use the values of the function and its first and second derivatives. In the following discussion of one-variable function minimizations, the function is assumed to be in the form $f = f(\alpha)$. However, the methods to be discussed are equally applicable for minimization of multivariable problems along a preselected direction, $\mathbf{s}$, using Eq. (4.1.1).

4.1.1 *Zeroth Order Methods*

Bracketing Method. As the name suggests, this method brackets the minimum of the function to be minimized between two points, through a series of function evaluations. The method begins with an initial point α_0, a function value $f(\alpha_0)$, a step size β_0, and a step expansion parameter $\gamma > 1$. The steps of the algorithm [2] are outlined as

1. Evaluate $f(\alpha_0)$ and $f(\alpha_0 + \beta_0)$.

2. If $f(\alpha_0 + \beta_0) < f(\alpha_0)$, let $\alpha_1 = \alpha_0 + \beta_0$ and $\beta_1 = \gamma\beta_0$, and evaluate $f(\alpha_1 + \beta_1)$. Otherwise go to step 4.

3. If $f(\alpha_1 + \beta_1) < f(\alpha_1)$, let $\alpha_2 = \alpha_1 + \beta_1$ and $\beta_2 = \gamma\beta_1$, and continue incrementing the subscripts this way until $f(\alpha_k + \beta_k) > f(\alpha_k)$. Then, go to step 8.

4. Let $\alpha_1 = \alpha_0$ and $\beta_1 = -\xi\beta_0$, where ξ is a constant that satisfies $0 < \xi < 1/\gamma$, and evaluate $f(\alpha_1 + \beta_1)$.

5. If $f(\alpha_1 + \beta_1) > f(\alpha_1)$ go to step 7.

6. Let $\alpha_2 = \alpha_1 + \beta_1$ and $\beta_2 = \gamma\beta_1$, and continue incrementing the subscripts this way until $f(\alpha_k + \beta_k) > f(\alpha_k)$. Then, go to step 8.

7. The minimum has been bracketed between points $(\alpha_0 - \xi\beta_0)$ and $(\alpha_0 + \beta_0)$. Go to step 9.

8. The last three points satisfy the relations $f(\alpha_{k-2}) > f(\alpha_{k-1})$ and $f(\alpha_{k-1}) < f(\alpha_k)$, and hence, the minimum is bracketed.

9. Use either one of the two end points of the bracket as the initial point. Begin with a reduced step size and repeat steps 1 through 8 to locate the minimum to a desired degree of accuracy.

Quadratic Interpolation. The method known as quadratic interpolation was first proposed by Powell [3] and uses the values of the function f to be minimized at three points to fit a parabola

$$p(\alpha) = a + b\alpha + c\alpha^2 \,, \tag{4.1.3}$$

through those points. The method starts with an initial point, say, $\alpha = 0$ with a function value $p_0 = f(\mathbf{x}_0)$, and a step size β. Two more function evaluations are performed as described in the following steps to determine the points for the polynomial fit. In general, however, we start with a situation where we have already bracketed the minimum between $\alpha_1 = \alpha_l$ and $\alpha_2 = \alpha_u$ by using the bracketing method described earlier. In that case we will only need an intermediate point α_0 in the interval (α_l, α_u).

1. Evaluate $p_1 = p(\beta) = f(\mathbf{x}_0 + \beta\mathbf{s})$

2. If $p_1 < p_0$, then evaluate $p_2 = p(2\beta) = f(\mathbf{x}_0 + 2\beta\mathbf{s})$. Otherwise evaluate $p_2 = p(-\beta) = f(\mathbf{x}_0 - \beta\mathbf{s})$. The constants $a, b,$ and c in equation Eq. (4.1.3) can now be uniquely expressed in terms of the function values $p_0, p_1,$ and p_2 as

$$a = p_0 \,,$$

$$b = \frac{4p_1 - 3p_0 - p_2}{2\beta}, \quad \text{and} \quad c = \frac{p_2 + p_0 - 2p_1}{2\beta^2}, \qquad \text{if } p_2 = f(\mathbf{x}_0 + 2\beta\mathbf{s}) \,, \tag{4.1.4}$$

or

$$b = \frac{p_1 - p_2}{2\beta}, \quad \text{and} \quad c = \frac{p_1 - 2p_0 + p_2}{2\beta^2}, \qquad \text{if } p_2 = f(\mathbf{x}_0 - \beta\mathbf{s}) \,. \tag{4.1.5}$$

3. The value of $\alpha = \alpha^*$ at which $p(\alpha)$ is extremized for the current cycle is then given by

$$\alpha^* = -\frac{b}{2c} \,. \tag{4.1.6}$$

4. α^* corresponds to a minimum of p if $c > 0$, and the prediction based on Eq. (4.1.3) is repeated using $(\mathbf{x}_0 + \alpha^*\mathbf{s})$ as the initial point for the next cycle with $p_0 = f(\mathbf{x}_0 + \alpha^*\mathbf{s})$ until the desired accuracy is obtained.

5. If the point $\alpha = \alpha^*$ corresponds to a maximum of p rather than a minimum, or if it corresponds to a minimum of p which is at a distance greater than a prescribed maximum $\beta_{\max}$ (possibly meaning α^* is outside the bracket points), then the maximum allowed step is taken in the direction of decreasing f and the point furthest away from this new point is discarded in order to repeat the process.

In step 4, instead of starting with $(\mathbf{x}_0 + \alpha^*\mathbf{s})$ as the initial point and repeating the previous steps, there is a cheaper alternative in terms of the number of function evaluations. The point $(\mathbf{x}_0 + \alpha^*\mathbf{s})$ and the two points closest to it from the left and

right can be used in another quadratic interpolation to give a better value of α^*. Other strategies for improving the accuracy of the prediction will be discussed later in Section 4.1.4.

Fibonacci and the Golden Section Search. Like bracketing, the Fibonacci and the golden section search techniques are very reliable, if not the most efficient, line search techniques for locating the unconstrained minimum of a function $f(\alpha)$ within the interval $a_0 \leq \alpha \leq b_0$. It is assumed that the function f is *unimodal*, or that it has only one minimum within the interval. Unimodal functions are not necessarily continuous or differentiable, nor convex (see Figure 4.1.1). A function is said to be unimodal [3] in the interval $\mathcal{I}_0$ if there exist an $\alpha^* \in \mathcal{I}_0$ such that α^* minimizes f on $\mathcal{I}_0$, and for any two points $\alpha_1, \alpha_2 \in \mathcal{I}_0$ such that $\alpha_1 < \alpha_2$ we have

$$\alpha_2 \leq \alpha^* \qquad \text{implies that} \qquad f(\alpha_1) > f(\alpha_2)\,, \tag{4.1.7}$$

$$\alpha_1 \geq \alpha^* \qquad \text{implies that} \qquad f(\alpha_2) > f(\alpha_1)\,. \tag{4.1.8}$$

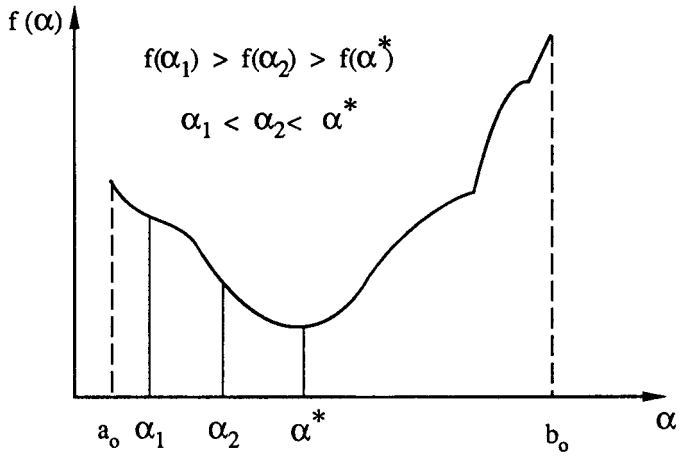

Figure 4.1.1 A typical unimodal function.

The assumption of unimodality is central to the Fibonacci search technique which seeks to reduce the interval of uncertainty within which the minimum of the function f lies.

The underlying idea behind the Fibonacci and the golden section search techniques can be explained as follows. Consider the minimization of f in the interval (a_0, b_0). Let us choose two points in the interval (a_0, b_0) at $\alpha = \alpha_1$ and at $\alpha = \alpha_2$ such that $\alpha_1 < \alpha_2$, and evaluate the function f at these two points. If $f(\alpha_1) > f(\alpha_2)$, then since the function is unimodal the minimum cannot lie in the interval (a_0, α_1). The new interval is (α_1, b_0) which is smaller than the original interval. Similarly, if $f(\alpha_2) > f(\alpha_1)$, then the new interval will be (a_0, α_2). The process can be repeated to

reduce the interval to any desired level of accuracy. Only one function evaluation is required in each iteration after the first one, but we have not specified how to choose the locations where f is evaluated. The best placement of these points will minimize the number of function evaluations for a prescribed accuracy requirement (i.e., reduction of the interval of uncertainty to a prescribed size). If the number of function evaluations is n the most efficient process is provided by a symmetric placement of the points provided by the relations [4]

$$\alpha_1 = a_0 + \frac{f_{n-1}}{f_{n+1}} l_0 \,, \tag{4.1.9}$$

$$\alpha_2 = b_0 - \frac{f_{n-1}}{f_{n+1}} l_0 \,, \tag{4.1.10}$$

and

$$\alpha_{k+1} \quad = \quad a_k + \frac{f_{n-(k+1)}}{f_{n-(k-1)}} l_k \quad = \quad b_k - \frac{f_{n-(k+1)}}{f_{n-(k-1)}} l_k \,, \tag{4.1.11}$$

where f_n are Fibonacci numbers defined by the sequence $f_0 = 1$, $f_1 = 1$, $f_n = f_{n-2} + f_{n-1}$, and l_k is the length of the kth interval (a_k, b_k). The total number of required function evaluations n may be determined from the desired level of accuracy. It can be shown that the interval of uncertainty after n function evaluations is $2\epsilon l_0$ where

$$\epsilon = \frac{1}{f_{n+1}} \,. \tag{4.1.12}$$

A disadvantage of the technique is that the number of function evaluations has to be specified in advance in order to start the Fibonacci search. To eliminate this undesirable feature a quasi-optimal technique known as the golden section search technique has been developed. The golden section search technique is based on the finding that for sufficiently large n, the ratio

$$\frac{f_{n-1}}{f_{n+1}} \to 0.382 \,. \tag{4.1.13}$$

Thus, it is possible to approximate the optimal location of the points given by Eqs. (4.1.9 - 4.1.11) by the following relations

$$\alpha_1 = a_0 + 0.382 l_0 \,, \tag{4.1.14}$$

$$\alpha_2 = b_0 - 0.382 l_0 \,, \tag{4.1.15}$$

and

$$\alpha_{k+1} \quad = \quad a_k + 0.382 l_k \quad = \quad b_k - 0.382 l_k \,. \tag{4.1.16}$$

Example 4.1.1

Determine the value of α, to within $\epsilon = \pm 0.1$, that minimizes the function $f(\alpha) = \alpha(\alpha - 3)$ on the interval $0 \leq \alpha \leq 2$ using the golden section search technique.

From Eqs. (4.1.14) and (4.1.15) we can calculate

$$\alpha_1 = 0 + 0.382(2) = 0.764, \qquad f(\alpha_1) = -1.708\,,$$
$$\alpha_2 = 2 - 0.382(2) = 1.236, \qquad f(\alpha_2) = -2.180\,.$$

Since $f(\alpha_2) < f(\alpha_1)$ we retain $(\alpha_1, 2)$. Thus, the next point is located at

$$\alpha_3 = 2 - 0.382(2 - 0.764) = 1.5278, \qquad f(\alpha_3) = -2.249\,.$$

Since $f(\alpha_3) < f(\alpha_2)$ we reject the interval (α_1, α_2). The new interval is $(\alpha_2, 2)$. The next point is located at

$$\alpha_4 = 2 - 0.382(2 - 1.236) = 1.7082, \qquad f(\alpha_4) = -2.207\,.$$

Since $f(\alpha_4) < f(\alpha_2) < f(2)$ we reject the interval $(\alpha_4, 2)$ and retain (α_2, α_4) as the next interval and locate the point α_5 at

$$\alpha_5 = 1.236 + 0.382(1.7082 - 1.236) = 1.4164, \qquad f(\alpha_5) = -2.243\,.$$

Since $f(\alpha_5) < f(\alpha_4) < f(\alpha_2)$ we retain the interval (α_5, α_4). The next point is located at

$$\alpha_6 = 1.7082 + 0.382(1.7082 - 1.4164) = 1.5967, \qquad f(\alpha_6) = -2.241\,.$$

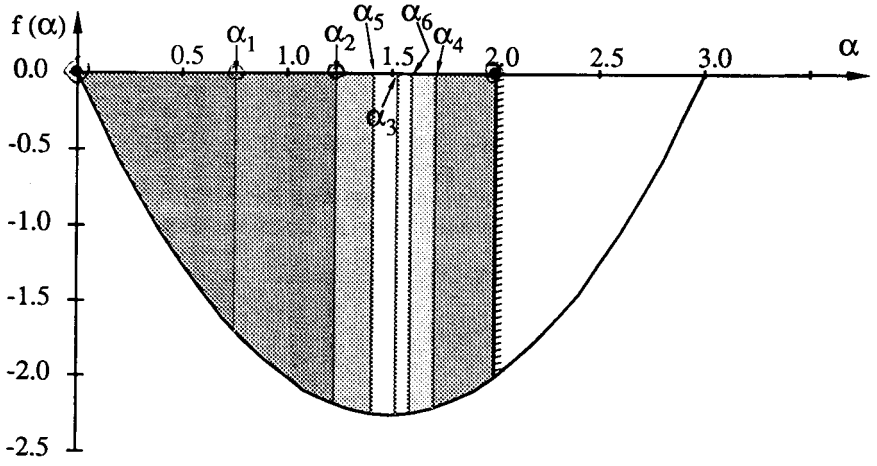

Figure 4.1.2 Iteration history for the function minimization $f(\alpha) = \alpha(\alpha - 3)$.

Since $f(\alpha_6) < f(\alpha_4)$ we reject the interval (α_6, α_4) and retain the interval (α_5, α_6) of length 0.18, which is less than the interval of specified accuracy, $2\epsilon = 0.2$. The iteration history for the problem is shown in Figure 4.1.2. Hence, the minimum has been bracketed to within a resolution of ± 0.1. That is, the minimum lies between $\alpha_5 = 1.4164$ and $\alpha_6 = 1.5967$. We can take the middle of the interval, $\alpha = 1.5066 \pm 0.0902$ as the solution. The exact location of the minimum is at $\alpha = 1.5$ where the function has the value -2.25. • • •

4.1.2 *First Order Methods*

Bisection Method. Like the bracketing and the golden section search techniques which progressively reduce the interval where the minimum is known to lie, the bisection technique locates the zero of the function f' by reducing the interval of uncertainty. Beginning with the known interval (a, b) for which $f'(a)f'(b) < 0$, an approximation to the root of f' is obtained from

$$\alpha^* = \frac{a+b}{2}\,, \tag{4.1.17}$$

which is the point midway between a and b. The value of f' is then evaluated at α^*. If $f'(\alpha^*)$ agrees in sign with $f'(a)$ then the point a is replaced by α^* and the new interval of uncertainty is given by (α^*, b). If on the other hand $f'(\alpha^*)$ agrees in sign with $f'(b)$ then the point b is replaced by α^* and the new interval of uncertainty is (a, α^*). The process is then repeated using Eq. (4.1.17).

Davidon's Cubic Interpolation Method. This is a polynomial approximation method which uses both the function values and its derivatives for locating its minimum. It is especially useful in those multivariable minimization techniques which require the evaluation of the function and its gradients.

We begin by assuming the function to be minimized $f(\mathbf{x}_0 + \alpha\mathbf{s}_0)$ to be approximated by a polynomial in the form

$$p(\alpha) = a + b\alpha + c\alpha^2 + d\alpha^3\,, \tag{4.1.18}$$

with constants $a, b, c,$ and d to be determined from the values of the function, p_0 and p_1, and its derivatives, g_0 and g_1, at two points, one located at $\alpha = 0$ and the other at $\alpha = \beta$.

$$p_0 = p(0) = f(\mathbf{x}_0), \qquad p_1 = p(\beta) = f(\mathbf{x}_0 + \beta\mathbf{s})\,, \tag{4.1.19}$$

and

$$g_0 = \frac{\mathrm{d}p}{\mathrm{d}\alpha}(0) = \mathbf{s}^T\nabla f(\mathbf{x}_0), \qquad g_1 = \frac{\mathrm{d}p}{\mathrm{d}\alpha}(\beta) = \mathbf{s}^T\nabla f(\mathbf{x}_0 + \beta\mathbf{s})\ . \tag{4.1.20}$$

After substitutions, Eq. (4.1.18) takes the following form

$$p(\alpha) = p_0 + g_0\alpha - \frac{g_0 + e}{\beta}\alpha^2 + \frac{g_0 + g_1 + 2e}{3\beta^2}\alpha^3\,, \tag{4.1.21}$$

where

$$e = \frac{3}{\beta}(p_0 - p_1) + g_0 + g_1\ . \tag{4.1.22}$$

We can now locate the minimum, $\alpha = \alpha_m$, of Eq. (4.1.21) by setting its derivative with respect to α to be zero. This results in

$$\alpha_m = \beta\left(\frac{g_0 + e \pm h}{g_0 + g_1 + 2e}\right)\,, \tag{4.1.23}$$

where

$$h = (e^2 - 2g_0 g_1)^{1/2} \ . \tag{4.1.24}$$

It can be easily verified, by checking $\mathrm{d}^2 p/\mathrm{d}\alpha^2$, that the positive sign must be retained in Eq. (4.1.23) for α_m to be a minimum rather than a maximum. Thus, the algorithm for Davidon's cubic interpolation [5] may be summarized as follows.

1. Evaluate $p_0 = f(\mathbf{x}_0)$ and $g_0 = \mathbf{s}^T \nabla f(\mathbf{x}_0)$ and make sure that $g_0 < 0$.

2. In the absence of an estimate of the initial step length β, we may calculate it on the basis of a quadratic interpolation derived using p_0, g_0 and an estimate of $p_{\min}$. Thus,

$$\beta = \frac{2(p_{\min} - p_0)}{g_0} \ . \tag{4.1.25}$$

3. Evaluate $p_1 = f(\mathbf{x}_0 + \beta \mathbf{s})$ and $g_1 = \dfrac{df(\mathbf{x}_0 + \beta \mathbf{s})}{d\beta}$

4. If $g_1 > 0$ or if $p_1 > p_0$ go to step 6, or else go to step 5.

5. Replace β by 2β and go to step 3.

6. Calculate α_m using Eq. (4.1.23) with a positive sign.

7. Use the interval $(0, \alpha_m)$ if

$$g_{\alpha_m} = \frac{df(\mathbf{x}_0 + \alpha_m \mathbf{s})}{d\alpha_m} \geq 0 \, , \tag{4.1.26}$$

or else use the interval (α_m, β) and return to step 4.

8. If α_m corresponds to a maximum, restart the algorithm by using new points. Selection of the new points may be performed by using a strategy similar to that described for the quadratic interpolation technique.

4.1.3 *Second Order Method*

The problem of minimizing the function $f(\alpha)$ is equivalent to obtaining the root of the nonlinear equation

$$f'(\alpha) = 0 \, , \tag{4.1.27}$$

because this is the necessary condition for the extremum of f. A convenient method for solving (4.1.27) is Newton's method. This method consists of linearizing $f'(\alpha)$ about a point $\alpha = \alpha_i$ and then determining the point α_{i+1} at which the linear approximation

$$f'(\alpha_{i+1}) = f'(\alpha_i) + f''(\alpha_i)(\alpha_{i+1} - \alpha_i) \, , \tag{4.1.28}$$

vanishes. This point

$$\alpha_{i+1} = \alpha_i - \frac{f'(\alpha_i)}{f''(\alpha_i)} \, , \tag{4.1.29}$$

serves as a new approximation for a repeated application of Eq. (4.1.29) with i replaced by $i+1$. For a successful convergence to the minimum it is necessary that the second derivative of the function f be greater than zero. Even so the method may diverge depending on the starting point. Several strategies exist [6] which modify Newton's method to make it globally convergent (that is, it will converge to a minimum regardless of the starting point) for multivariable functions; some of these will be covered in the next section.

The reason this method is known as a second order method is not only because it uses second derivative information about the function f, but also because it has a rate of convergence to the minimum that is quadratic. In other words, Newton's algorithm converges to the minimum α^* such that

$$\lim_{i\to\infty} \frac{|\alpha_{i+1} - \alpha^*|}{(\alpha_i - \alpha^*)^2} = \beta\,, \tag{4.1.30}$$

where α_i and α_{i+1} are the ith and the $(i+1)$st estimates of the minimum value of the α^*, β is a non-zero constant.

4.1.4 *Safeguarded Polynomial Interpolation* [7], p. 92

Polynomial interpolations such as the *Quadratic interpolation* and the *Davidon's cubic interpolation* are sometimes found to be quite inefficient and unreliable for locating the minimum of a function along a line. If the interpolation function is not representative of the behavior of the function to be minimized within the interval of uncertainty, the minimum may fall outside the interval, or become unbounded below, or the successive iterations may be too close to one another without achieving a significant improvement in the function value. In such cases, we use what are known as safeguarded procedures. These procedures consist of combining polynomial interpolations with a simple bisection technique or the golden section search technique described earlier. At the end of the polynomial interpolation, the bisection technique would be used to find the zero of the derivative of the function f. The golden section search, on the other hand, would work with the function f itself using the known interval of uncertainty (a, b) and locate the point α^* which corresponds to the minimum of f within the interval.

4.2 Minimization of Functions of Several Variables

4.2.1 *Zeroth Order Methods*

Several methods exist for minimizing a function of several variables using only function values. However, only two of these methods may be regarded as being useful. These are the *sequential simplex* method of Spendley, Hext and Himsworth [8] and

Powell's conjugate direction method [3]. Both of these methods require that the function $f(\mathbf{x}), \mathbf{x} \in \mathbf{R}^n$, be unimodal; that is the function f has only one minimum. The sequential simplex does not require that the function f be differentiable, while the differentiability requirement on f is implicit in the exact line searches of Powell's method. It appears from tests by Nelder and Mead [9] that for most problems the performance of the sequential simplex method is comparable to if not better than Powell's method. Both of these methods are considered inefficient for $n \geq 10$; Powell's method may fail to converge for $n \geq 30$. A more recent modification of the simplex method by Chen, et al. [10] extends the applicability of this algorithm for high dimensional cases. If the function is differentiable, it is usually more efficient to use the more powerful first and second order methods with derivatives obtained explicitly or from finite difference formulae.

Sequential Simplex Method. The sequential simplex method was originally proposed by Spendley, Hext and Himsworth [8] and was subsequently improved by Nelder and Mead [9]. The method begins with a regular geometric figure called the simplex consisting of $n+1$ vertices in an n-dimensional space. These vertices may be defined by the origin and by points along each of the n coordinate directions. Such a simplex may not be geometrically regular. The following equations are suggested in Ref. 8 for the calculation of the positions of the vertices of a regular simplex of size a in the n-dimensional design space

$$\mathbf{x}_j = \mathbf{x}_0 + p\mathbf{e}_j + \sum_{\substack{k=1 \\ k \neq j}}^{n} q\mathbf{e}_k, \qquad j = 1, \ldots, n\,, \tag{4.2.1}$$

with

$$p = \frac{a}{n\sqrt{2}}(\sqrt{n+1} + n - 1), \qquad \text{and} \qquad q = \frac{a}{n\sqrt{2}}(\sqrt{n+1} - 1)\,, \tag{4.2.2}$$

where $\mathbf{e}_k$ is the unit base vector along the kth coordinate direction, and $\mathbf{x}_0$ is the initial base point. For example, for a problem in two-dimensional design space Eqs. (4.2.1) and (4.2.2) lead to an equilateral triangle of side a.

Once the simplex is defined, the function f is evaluated at each of the $n+1$ vertices $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n$. Let $\mathbf{x}_h$ and $\mathbf{x}_l$ denote the vertices where the function f assumes its maximum and minimum values, respectively, and $\mathbf{x}_s$ the vertex where it assumes the second highest value. The simplex method discards the vertex $\mathbf{x}_h$ and replaces it by a point where f has a lower value. This is achieved by three operations namely *reflection, contraction*, and *expansion.*

The reflection operation creates a new point $\mathbf{x}_r$ along the line joining $\mathbf{x}_h$ to the centroid $\bar{\mathbf{x}}$ of the remaining points defined as

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=0}^{n} \mathbf{x}_i, \qquad i \neq h\ . \tag{4.2.3}$$

The vertex at the end of the reflection is calculated by

$$\mathbf{x}_r = \bar{\mathbf{x}} + \alpha(\bar{\mathbf{x}} - \mathbf{x}_h)\,, \tag{4.2.4}$$

with α being a positive constant called the reflection coefficient which is usually assumed to be unity. Any positive value of the reflection coefficient in Eq. (4.2.4) guarantees that $\mathbf{x}_r$ is on the other side of the $\bar{\mathbf{x}}$ from $\mathbf{x}_h$. If the value of the function at this new point, $f_r = f(\mathbf{x}_r)$, satisfies the condition $f_l < f_r \leq f_s$, then $\mathbf{x}_h$ is replaced by $\mathbf{x}_r$ and the process is repeated with this new simplex. If, on the other hand, the value of the function f_r at the end of the reflection is less than the lowest value of the function $f_l = f(\mathbf{x}_l)$, then there is a possibility that we can still decrease the function by going further along the same direction. We seek an improved point $\mathbf{x}_e$ by the expansion technique using the relation

$$\mathbf{x}_e = \bar{\mathbf{x}} + \beta(\mathbf{x}_r - \bar{\mathbf{x}})\,, \tag{4.2.5}$$

with the expansion coefficient β often being chosen to be 2. If the value of the function f_e is smaller than the value at the end of the reflection step, then we replace $\mathbf{x}_h$ by $\mathbf{x}_e$ and repeat the process with the new simplex. However, if the expansion leads to a function value equal to or larger than f_r, then we form the new simplex by replacing $\mathbf{x}_h$ by $\mathbf{x}_r$ and continue.

Finally, if the process of reflection leads to a point x_r such that, $f_r < f_h$, then we replace $\mathbf{x}_h$ by $\mathbf{x}_r$ and perform contraction. Otherwise ($f_r \geq f_h$), we perform contraction without any replacement using

$$\mathbf{x}_c = \bar{\mathbf{x}} + \gamma(\mathbf{x}_h - \bar{\mathbf{x}})\,, \tag{4.2.6}$$

with the contraction coefficient γ, $0 < \gamma < 1$, usually chosen to be 1/2. If $f_c = f(\mathbf{x}_c)$ is greater than f_h, then we replace all the points by a new set of points

$$\mathbf{x}_i = \mathbf{x}_i + \frac{1}{2}(\mathbf{x}_l - \mathbf{x}_i), \qquad i = 0, 1, \ldots, n\,, \tag{4.2.7}$$

and restart the process with this new simplex. Otherwise, we simply replace $\mathbf{x}_h$ by $\mathbf{x}_c$ and restart the process with this simplex. The operation in Eq. (4.2.7) causes the distance between the points of the old simplex and the point with the lowest function value to be halved and is therefore referred to as the *shrinkage* operation. The flow chart of the complete method is given in Figure 4.2.1. For the convergence criterion to terminate the algorithm Nelder and Mead [9] proposed the following

$$\left\{ \frac{1}{1+n} \sum_{i=0}^{n} [f_i - f(\bar{\mathbf{x}})]^2 \right\}^{\frac{1}{2}} < \epsilon\,, \tag{4.2.8}$$

where ϵ is some specified accuracy requirement.

An improvement in the performance of the simplex algorithm for those cases with large number of design variables, n, is achieved by Chen, Saleem, and Grace [10]. A modified simplex search procedure proposed in Ref. [10] executes the *reflection, expansion, contraction*, and, *shrinkage* operations on more than one vertex of the simplex at a given step. This is achieved by first separating the vertices of the simplex

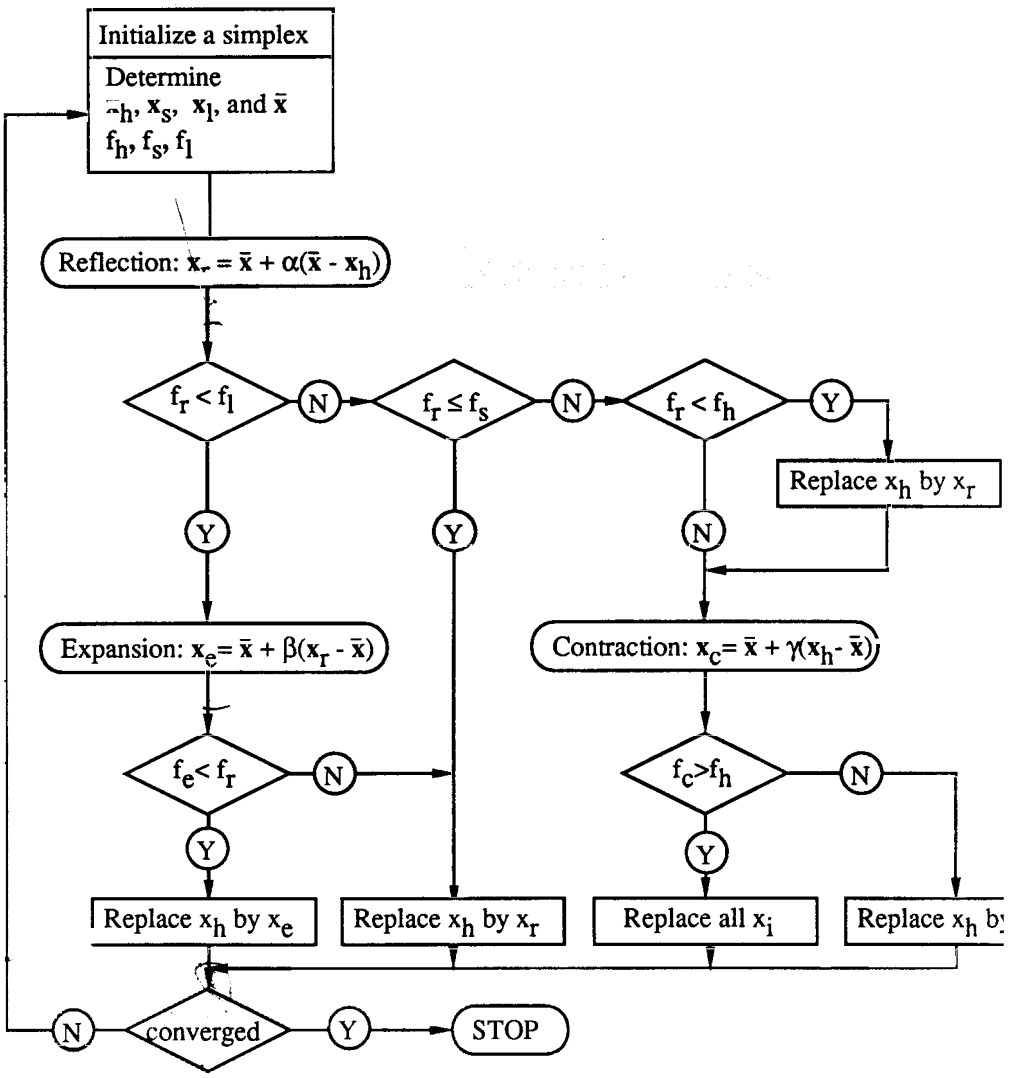

Figure 4.2.1 Flow chart of the Sequential Simplex Algorithm.

into two groups by defining a cutting value (CV) of the function, $f_{\rm cv}$. The cutting value is defined by the relation

$$f_{\rm cv} = \frac{(f_h + f_l)}{2} + \eta s\,, \tag{4.2.9}$$

where s is the standard deviation of the values of the function corresponding to the

vertices of the simplex,

$$s = \left[\sum_{i=0}^{n} (f_i - \bar{f})^2 / (n+1) \right]^{\frac{1}{2}} , \tag{4.2.10}$$

and η is a parameter (discussed below) that controls the number of vertices to be operated on. The $\bar{f}$ value in Eq. (4.2.10) is the average of the function values over the entire current simplex.

The vertices with function values higher than the cutting value form the group to be reflected (and to be dropped). The other vertices serve as reference points. If the parameter η is sufficiently large, all the vertices of the simplex except the $\mathbf{x}_h$ stay in the group to be used as the reference points and, therefore, the algorithm is equivalent to the original form. For sufficiently small values of the parameter η, all points except the $\mathbf{x}_n$ are dropped. The selection of the parameter η depends on the difficulty of the problem as well as the number of variables. Recommended values for η are given in Table II of Ref. [10]. Among the $n+1$ vertices of the current simplex, we rearrange and number the vertices from largest to smallest function values as $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\text{cv}}, \ldots, \mathbf{x}_n$ where $i = 0, \ldots, n_{\text{cv}}$ are the elements of the group to be reflected next. The centroid of the vertices in the reference group is defined as

$$\bar{\mathbf{x}} = \frac{1}{n - n_{\text{cv}}} \sum_{i=n_{\text{cv}}+1}^{n} \mathbf{x}_i \ . \tag{4.2.11}$$

The performance of this modified simplex method has been compared [10] with the simplex method proposed by Nelder and Mead, and also with more powerful methods such as the second order Davidon-Fletcher-Powell (DFP) method which will be discussed later in this chapter. For high dimensional problems the modified simplex algorithm was found to be more efficient and robust than the DFP algorithm. Nelder and Mead [9] have also provided several illustrations of the use of their algorithm in minimizing classical test functions and compared its performance with Powell's conjugate directions method which will be discussed next.

Powell's Conjugate Directions Method and its Subsequent Modification. Although most problems have functions which are not quadratic, many unconstrained minimization algorithms are developed to minimize a quadratic function. This is because a function can be approximated well by a quadratic function near a minimum. Powell's conjugate directions algorithm is a typical example. A quadratic function in $\mathbf{R}^n$ may be written as

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{b}^T \mathbf{x} + \mathbf{c} \ . \tag{4.2.12}$$

A set of directions $\mathbf{s}_i, i = 1, 2 \ldots$ are said to be Q-conjugate if

$$\mathbf{s}_i^T Q \mathbf{s}_j = 0, \qquad \text{for } i \neq j \ . \tag{4.2.13}$$

Furthermore, it can be shown that if the function f is minimized once along each direction of a set $\mathbf{s}$ of linearly independent Q-conjugate directions then the minimum

of f will be located at or before the nth step regardless of the starting point provided that no round-off errors are accumulated. This property is commonly referred to as the quadratic termination property. Powell provided a convenient method for generating such conjugate directions by a suitable combination of the simple univariate search and a pattern search technique [3]. However, in certain cases Powell's algorithm generates directions which are linearly dependent and thereby fails to converge to the minimum. Hence, Powell modified his algorithm to make it robust but at the expense of its quadratic termination property.

Powell's strategy for generating conjugate directions is based on the following property (see Ref. 3 for proof). If $\mathbf{x}_1$ and $\mathbf{x}_2$ are any two points and $\mathbf{s}$ a specified direction, and $\mathbf{x}_{1s}$ corresponds to the minimum point of a quadratic function f on a line starting at $\mathbf{x}_1$ along $\mathbf{s}$ and $\mathbf{x}_{2s}$ is the minimum point on a line starting at $\mathbf{x}_2$ along $\mathbf{s}$, then the directions $\mathbf{s}$ and $(\mathbf{x}_{2s} - \mathbf{x}_{1s})$ are Q-conjugate. The basic steps of Powell's modified method are based on a cycle of univariate minimizations. For each cycle we use the following steps.

1. Minimize f along each of the coordinate directions (univariate search) starting at $\mathbf{x}_0^k$ and generating the points $\mathbf{x}_1^k, \ldots, \mathbf{x}_n^k$ where k is the cycle number.

2. After completing the univariate cycle find the index m corresponding to the direction of the univariate search which yields the largest function decrease in going from $\mathbf{x}_{m-1}^k$ to $\mathbf{x}_m^k$.

3. Calculate the "pattern" direction $\mathbf{s}_p^k = \mathbf{x}_n^k - \mathbf{x}_0^k$ (which is the sum of all the univariate moves) and determine the value of α from $\mathbf{x}_0^k$ along $\mathbf{s}_p^k$ that minimizes f. Denote this new point by $\mathbf{x}_0^{k+1}$.

4. If

$$|\alpha| < \left[\frac{f(\mathbf{x}_0^k) - f(\mathbf{x}_0^{k+1})}{f(\mathbf{x}_{m-1}^k) - f(\mathbf{x}_m^k)}\right]^{\frac{1}{2}}, \tag{4.2.14}$$

then use the same old directions again for the next univariate cycle (that is do not discard any of the directions of the previous cycle in preference to the pattern direction $\mathbf{s}_p^k$). If Eq. (4.2.14) is not satisfied then replace the mth direction by the pattern direction $\mathbf{s}_p^k$.

5. Begin the next univariate cycle with the directions decided in step 4, and repeat the steps 2 through 4 until convergence to a specified accuracy. Convergence is assumed to be achieved when the Euclidean norm $\|\mathbf{x}^{k-1} - \mathbf{x}^k\|$ is less than a prespecified quantity ϵ.

Although Powell's original method does possess a quadratic termination property, his modified algorithm does not [3]. The modified method will now be illustrated on the following simple example from structural analysis.

Example 4.2.1

The problem of determination of the maximum deflection and tip-rotation of a cantilever beam of length l shown in Figure (4.2.2) loaded at its tip is considered. Solution of this problem is formulated as a minimization of the total potential energy of the beam which is modelled using a single cubic beam finite element. For a two-noded beam element with two degrees of freedom at each node, the displacement field is assumed to be

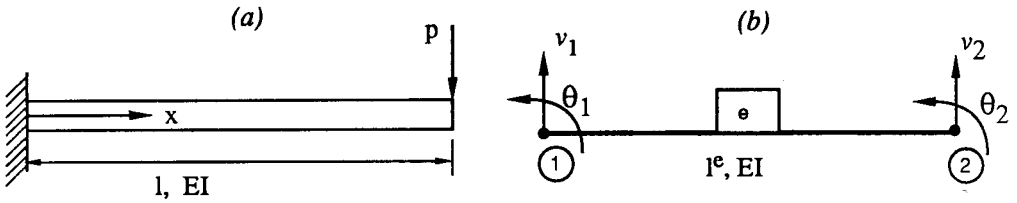

Figure 4.2.2 Tip loaded cantilever beam and its finite element model.

$$v(\xi) = \begin{bmatrix} (1 - 3\xi^2 + 2\xi^3) & l(\xi - 2\xi^2 + \xi^3) & (3\xi^2 - 2\xi^3) & l(-\xi^2 + \xi^3) \end{bmatrix} \begin{Bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{Bmatrix} , \tag{4.2.15}$$

where $\xi = x/l$. The corresponding potential energy of the beam model is given by

$$\Pi = \frac{EI}{2l^3} \int_0^l \left(\frac{\mathrm{d}^2 v}{\mathrm{d}\xi^2} \right)^2 \mathrm{d}\xi + p v_2 \ . \tag{4.2.16}$$

Because of the cantilever end condition at $\xi = 0$, the first two degrees of freedom in Eq. (4.2.15) are zero. Therefore, substituting Eq. (4.2.15) into Eq. (4.2.16) we obtain

$$\Pi = \frac{EI}{2l^3}(12v_2^2 + 4\theta_2^2 l^2 - 12 v_2 \theta_2 l) + p v_2 \ . \tag{4.2.17}$$

Defining $f = 2\Pi l^3/EI,\ x_1 = v_2,\ x_2 = \theta_2 l$, and choosing $pl^3/EI = 1$, the problem of determining the tip deflection and rotation of the beam reduces to an unconstrained minimization of

$$f = 12x_1^2 + 4x_2^2 - 12 x_1 x_2 + 2x_1 \ . \tag{4.2.18}$$

Starting with an initial point of $\mathbf{x}_0^1 = (-1,\ -2)^T$ and $f(\mathbf{x}_0^1) = 2$ we will minimize f using Powell's conjugate directions method. The exact solution of this problem is at $\mathbf{x}^* = (-1/3,\ -1/2)^T$.

Since we have an explicit relation for the objective function f, the one dimensional minimizations along a given direction will be performed exactly without resorting to any of the numerical techniques discussed in the previous section. However, if these minimizations were done numerically, one of the zeroth order techniques would be sufficient. We use superscripts to denote the univariate cycle number and subscripts to denote the iteration number within a cycle.

First, we perform the univariate search along the x_1 and x_2 directions. Choosing $\mathbf{s}_1^1 = (1,\ 0)^T$ we have

$$\mathbf{x}_1^1 = \begin{Bmatrix} -1 \\ -2 \end{Bmatrix} + \alpha \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} = \begin{Bmatrix} -1+\alpha \\ -2 \end{Bmatrix} , \tag{4.2.19}$$

and

$$f(\alpha) = 12(-1+\alpha)^2 + 4(-2)^2 - 12(-1+\alpha)(-2) + 2(-1+\alpha) \ . \tag{4.2.20}$$

Taking the derivative of Eq. (4.2.20) with respect to α, we obtain the value of α which minimizes f to be $\alpha = -1/12$. Hence,

$$\mathbf{x}_1^1 = \begin{Bmatrix} \frac{-13}{12} \\ -2 \end{Bmatrix} \qquad \text{and} \quad f(\mathbf{x}_1^1) = 1.916666667 \ .$$

Choosing $\mathbf{s}_2^1 = (0,\ 1)^T$, we obtain

$$\mathbf{x}_2^1 = \begin{Bmatrix} \frac{-13}{12} \\ -2 \end{Bmatrix} + \alpha \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} = \begin{Bmatrix} \frac{-13}{12} \\ -2+\alpha \end{Bmatrix} , \tag{4.2.21}$$

and

$$f(\alpha) = 12\left(\frac{-13}{12}\right)^2 + 4(-2+\alpha)^2 - 12\left(\frac{-13}{12}\right)(-2+\alpha) + 2\left(\frac{-13}{12}\right) , \tag{4.2.22}$$

which is minimum at $\alpha = 3/8$. Therefore, at the end of the univariate search we have

$$\mathbf{x}_2^1 = \begin{Bmatrix} \frac{-13}{12} \\ \\ \frac{-13}{8} \end{Bmatrix} \qquad \text{and} \quad f(\mathbf{x}_2^1) = 1.354166667 \ .$$

At this point we construct a pattern direction as

$$\mathbf{s}_p^1 = \mathbf{x}_2^1 - \mathbf{x}_0^1 = \begin{Bmatrix} \frac{-13}{12} \\ \\ \frac{-13}{8} \end{Bmatrix} - \begin{Bmatrix} -1 \\ -2 \end{Bmatrix} = \begin{Bmatrix} \frac{-1}{12} \\ \\ \frac{3}{8} \end{Bmatrix} , \tag{4.2.23}$$

and minimize the function along this direction by

$$\mathbf{x}_0^2 = \begin{Bmatrix} -1 \\ -2 \end{Bmatrix} + \alpha \begin{Bmatrix} \frac{-1}{12} \\ \\ \frac{3}{8} \end{Bmatrix} = \begin{Bmatrix} -1 - \frac{\alpha}{12} \\ \\ -2 + \frac{3\alpha}{8} \end{Bmatrix} , \tag{4.2.24}$$

which attains its minimum value for $\alpha = 40/49$ at

$$\mathbf{x}_0^2 = \begin{Bmatrix} \frac{-157}{147} \\ \\ \frac{-83}{49} \end{Bmatrix} \qquad \text{and} \quad f(\mathbf{x}_0^2) = 1.319727891 \ .$$

The direction that corresponds to the largest decrease in the objective function f during the first cycle of the univariate search is associated with the second variable. We can now decide whether we want to replace the second ($m = 2$) univariate search direction by the pattern direction or not by checking the condition stated in step 4 of the algorithm, Eq. (4.2.24). That is, Powell's criterion

$$|\alpha| = \frac{40}{49} < \left[\frac{2 - 1.319727891}{1.916666667 - 1.354166667}\right]^{\frac{1}{2}} \ . \tag{4.2.25}$$

is satisfied, therefore, we retain the old univariate search directions for the second cycle and restart the procedure by going back to step 2 of the algorithm. The results of the second cycle are tabulated in Table 4.2.1.

Table 4.2.1. Solution of the beam problem using Powell's conjugate directions method

CycleNo.	x_1	x_2	f
0	-1.0	-2.0	2.0
1	-1.083334	-2.0	1.916667
1	-1.083334	-1.625	1.354167
2	-0.895834	-1.625	0.9322967
2	-0.895834	-1.34375	0.6158854
2	-0.33334	-0.499999	-0.333333

The effectiveness of Powell's modified method can be seen to be much more pronounced on the minimization of the following function considered by Avriel [2],

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

and left as an exercise for the reader (see Exercise 2). • • •

Before we proceed to the discussion of first order methods, it is worthwhile to consider when zeroth order method should be used. The sequential simplex method can be used for non differentiable functions where first order methods are not appropriate. For those unconstrained minimization problems with differentiable functions, it is preferable to calculate the exact derivatives, or generate such derivatives by using finite differences and subsequently use a first order method for minimization when these derivatives can be calculated accurately. Zeroth order methods such as Powell's conjugate directions algorithm may still have a place for problems with a highly nonlinear objective functions where the accuracy of the function evaluations may be poor. The poor accuracy in function evaluations may call for high order finite difference formulae to be used for derivative calculations, therefore, the use of a zeroth order method for minimization may be a prudent alternative.

4.2.2 *First Order Methods*

First order methods for unconstrained minimization of a function f in $\mathbf{R}^n$ use the gradient of the function as well as its value in calculating the move direction for the function minimization. These methods possess a linear or a superlinear rate of convergence. A sequence $\mathbf{x}_k, k = 0, 1, 2, \ldots$, is said to be q-superlinear convergent to $\mathbf{x}^*$ of order at least p if

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq c_k \|\mathbf{x}_k - \mathbf{x}^*\|^p \,, \tag{4.2.26}$$

where c_k converges to zero. If c_k in Eq.(4.2.26) is a constant then the convergence is said to be a simple q-order convergence of order at least p. Thus, if $p = 1$ with c_k equal to a constant then we have a linear convergence rate, whereas if $p = 1$ and c_k is a sequence that converges to zero then the convergence is said to be superlinear (see Ref. 6 for additional definitions).

Perhaps the oldest known method for minimizing a function of n variables is the steepest descent method first proposed by Cauchy [11] for solving a system of linear equations. It can be used for function minimization as follows. The direction of move is obtained by minimizing the directional derivative of f

$$\nabla f^T \mathbf{s} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} s_i \,, \tag{4.2.27}$$

subject to the condition that $\mathbf{s}$ be a unit vector in $\mathbf{R}^n$ in the Euclidean sense.

$$\mathbf{s}^T \mathbf{s} = 1 \,. \tag{4.2.28}$$

It can easily be verified (see Exercise 6) that the steepest descent direction is given by

$$\mathbf{s} = -\frac{\nabla f}{\|\nabla f\|} \,, \tag{4.2.29}$$

where $\|\ \|$ denotes the Euclidean norm, and it provides the largest decrease in the function f. Starting with a point $\mathbf{x}_k$ at the kth iteration of the minimization process, we obtain the next point $\mathbf{x}_{k+1}$ as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{s} \,. \tag{4.2.30}$$

Here $\mathbf{s}$ is given by Eq. (4.2.29) and α is determined such that f is minimized along the chosen direction by using any one of the one-dimensional minimization techniques covered in the previous section. If the function to be minimized is quadratic in $\mathbf{R}^n$ and expressed as

$$f = \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{b}^T \mathbf{x} + c \,, \tag{4.2.31}$$

the step length can be determined directly by substituting Eq. (4.2.30) into Eq. (4.2.31) for the $(k+1)$st iteration followed by a minimization of f with respect to α which yields

$$\alpha^* = -\frac{({\mathbf{x}_k}^T \mathbf{Q} + \mathbf{b}^T)\mathbf{s}}{(\mathbf{s}^T \mathbf{Q} \mathbf{s})} \,. \tag{4.2.32}$$

In obtaining Eq. (4.2.32) we assume that the Hessian matrix **Q** of the quadratic form is available explicitly, and we make use of the symmetry of **Q**.

The performance of the steepest descent method depends on the condition number of the Hessian matrix **Q**. The condition number of a matrix is the ratio of the largest to the smallest eigenvalue. A large condition number implies that the contours of the function to be minimized form an elongated design space, and therefore the progress made by the steepest descent method is very slow and proceeds in a zigzag pattern known as hemstitching. This is even true for quadratic functions, and can be improved by re-scaling the variables.

Example 4.2.2

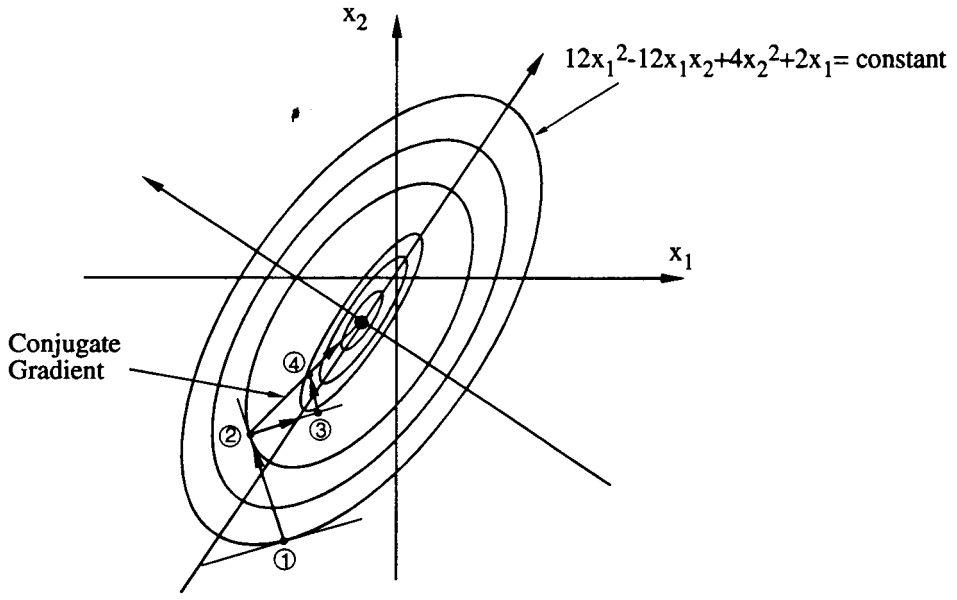

Figure 4.2.3 Contours of the cantilever beam potential energy function.

The cantilever problem discussed in the previous example illustrates this behavior most vividly. The steepest descent method when applied to this problem may exhibit the typical hemstitching phenomenon as shown in Figure 4.2.3 for certain initial starting points. However, a simple transformation of variables to improve the scaling of the variables causes the steepest descent method to converge to the minimum in a single step. For example, consider the following transformation

$$y_1 = (x_1 - \frac{1}{2}x_2), \qquad y_2 = \frac{1}{\sqrt{12}}x_2 \ . \tag{4.2.33}$$

The function f may now be expressed in terms of the new variables y_1 and y_2 as

$$f(y_1, y_2) = y_1^2 + y_2^2 + \frac{1}{6}(y_1 + \sqrt{3}y_2) \ . \tag{4.2.34}$$

As a result of the scaling and elimination of the cross-product term, the condition number of the Hessian of f is unity. Contours of the function f in the $y_1 - y_2$ plane will appear as circles. Beginning with any arbitrary starting point $\mathbf{y}_0$ and applying the steepest descent method we have

$$\mathbf{y}_1 = \mathbf{y}_0 + \alpha \begin{Bmatrix} 2y_{10} + \frac{1}{6} \\ 2y_{20} + \frac{\sqrt{3}}{6} \end{Bmatrix} . \tag{4.2.35}$$

It can be easily verified that the value of α^* that minimizes f is 0.5. Therefore,

$$\mathbf{y}_1 = \begin{Bmatrix} \frac{-1}{12} \\ \frac{-\sqrt{3}}{12} \end{Bmatrix} ,$$

at which the gradient of f is zero, implying that it is a minimum point. The corresponding values of the original variables x_1^*, and x_2^* are $-1/3$ and $-1/2$, respectively. This simple demonstration clearly shows the effectiveness of scaling in convergence of the steepest descent algorithm to the minimum of a function in $\mathbf{R}^n$. It can be shown [6] that the steepest descent method has only a linear rate of convergence in the absence of an appropriate scaling. • • •

Unfortunately, in most multivariable function minimizations it is not easy to determine the appropriate scaling transformation that leads to a one step convergence to the minimum of a general quadratic form in $\mathbf{R}^n$ using the steepest descent algorithm. This would require calculating the Hessian matrix and then performing an expensive eigenvalue analysis of the matrix. Hence, we are forced to look at other alternatives for rapid convergence to the minimum of a quadratic form. One such alternative is provided by minimizing along a set of conjugate gradient directions which guarantees a quadratic termination property. Hestenes and Stiefel [12] and later Fletcher and Reeves [13] offered such an algorithm which will be covered next.

Fletcher-Reeves' Conjugate Gradient Algorithm. This algorithm begins from an initial point $\mathbf{x}_0$ by first minimizing f along the steepest descent direction, $\mathbf{s}_0 = -\nabla f(\mathbf{x}_0) = \mathbf{g}_0$, to the new iterate $\mathbf{x}_1$. The direction for the next iteration $\mathbf{s}_1$ must be constructed so that it is Q-conjugate to $\mathbf{s}_0$ where $\mathbf{Q}$ is the Hessian of the quadratic f. The function is then minimized along $\mathbf{s}_1$ to yield the next iterate $\mathbf{x}_2$. The next direction $\mathbf{s}_2$ from $\mathbf{x}_2$ is constructed to be Q-conjugate to the previous directions $\mathbf{s}_0$ and $\mathbf{s}_1$, and the process is continued until convergence to the minimum is achieved. By virtue of Powell's theorem on conjugate directions for quadratic functions, convergence to the minimum is theoretically guaranteed at the end of the minimization of the function f along the conjugate direction $\mathbf{s}_{n-1}$. For functions which are not quadratic, conjugacy of the directions $\mathbf{s}_i, i = 1, \ldots, n$ loses its meaning since the Hessian of the functions is not a matrix of constants. However, it is a common practice to use this algorithm for non-quadratic functions. Since, for such functions, convergence to the minimum will rarely be achieved in n steps or less, the algorithm is restarted after every n steps. The basic steps of the algorithm at the $(k+1)$th iterate is as follows

1. Calculate $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_{k+1}\mathbf{s}_k$ where α_{k+1} is determined such that

$$\frac{\mathrm{d}f(\alpha_{k+1})}{\mathrm{d}\alpha_{k+1}} = 0 . \tag{4.2.36}$$

2. Let $\mathbf{s}_k = \mathbf{g}_k = -\nabla f(\mathbf{x}_k)$ if $k = 0$; and $\mathbf{s}_k = \mathbf{g}_k + \beta_k \mathbf{s}_{k-1}$ if $k > 0$ with

$$\beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}}, \quad \text{and} \quad \mathbf{g}_k = -\nabla f(\mathbf{x}_k) \; . \tag{4.2.37}$$

3. If $\|\mathbf{g}_{k+1}\|$ or $|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)|$ is sufficiently small, then stop. Otherwise
4. If $k < n$ go to step number 1, or else restart

Example 4.2.3

We will show the effectiveness of this method on the cantilever beam problem for which we minimize

$$f = 12x_1^2 + 4x_2^2 - 12x_1x_2 + 2x_1 \, ,$$

starting with the initial design point $\mathbf{x}_0^T = (-1, -2)$. The initial move direction is calculated from the gradient

$$\nabla f(\mathbf{x}_0) = \left\{ \begin{array}{c} 24x_1 - 12x_2 + 2 \\ 8x_2 - 12x_1 \end{array} \right\}_{\mathbf{x}=\mathbf{x}_0} ,$$

$$\mathbf{s}_0 = -\nabla f(\mathbf{x}_0) = \left\{ \begin{array}{c} -2 \\ 4 \end{array} \right\} ,$$

and at the end of the first step we have

$$\mathbf{x}_1 = \left\{ \begin{array}{c} -1 \\ -2 \end{array} \right\} + \alpha_1 \left\{ \begin{array}{c} -2 \\ 4 \end{array} \right\} ,$$

$$f(\alpha_1) = 12(-1 - 2\alpha_1)^2 + 4(-1 + 4\alpha_1)^2 - 12(-1 - 2\alpha_1)(-2 + 4\alpha_1) + 2(-1 - 2\alpha_1) \; .$$

The value of α_1 for which the function f is a minimum is obtained from the condition $\mathrm{d}f/\mathrm{d}\alpha_1 = 0$, or $\alpha_1 = 0.048077$. The new design point and the gradient at that point are

$$\mathbf{x}_1 = \left\{ \begin{array}{c} -1.0961 \\ -1.8077 \end{array} \right\}, \qquad \text{and} \quad \nabla f(\mathbf{x}_1) = \left\{ \begin{array}{c} -2.6154 \\ -1.3077 \end{array} \right\} \; .$$

Next, let $\mathbf{s}_1 = -\nabla f(\mathbf{x}_1) + \beta_1 \mathbf{s}_0$ with β_1 from Eq. (4.2.37), or

$$\beta_1 = \frac{(-2.6154)^2 + (-1.3077)^2}{(-2)^2 + (4)^2} \quad = \quad 0.4275 \, ,$$

The new move direction is

$$\mathbf{s}_1 = -\left\{ \begin{array}{c} -2.6154 \\ -1.3077 \end{array} \right\} + 0.4275 \left\{ \begin{array}{c} -2 \\ 4 \end{array} \right\} = \left\{ \begin{array}{c} 1.76036 \\ 3.0178 \end{array} \right\} ,$$

and

$$\mathbf{x}_2 = \left\{ \begin{array}{c} -1.0961 \\ -1.8077 \end{array} \right\} + \alpha_2 \left\{ \begin{array}{c} 1.76036 \\ 3.0178 \end{array} \right\} \; .$$

Again setting $\mathrm{d}f(\alpha_2)/\mathrm{d}(\alpha_2) = 0$ we obtain $\alpha_2 = 0.4334$,

$$\mathbf{x}_2 = \begin{Bmatrix} -0.3334 \\ -0.50 \end{Bmatrix}, \qquad \text{and} \quad \nabla f(\mathbf{x}_2) = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} .$$

Finally, since

$$\begin{Bmatrix} -2 \\ 4 \end{Bmatrix}^T \begin{bmatrix} 24 & -12 \\ -12 & 8 \end{bmatrix} \begin{Bmatrix} 1.76036 \\ 3.0178 \end{Bmatrix} \simeq 0 .$$

we have verified the Q-conjugacy of the two directions $\mathbf{s}_0$ and $\mathbf{s}_1$. The progress of minimization using this method is illustrated in Figure (4.2.3). • • •

Beale's Restarted Conjugate Gradient Technique. In minimizing non-quadratic functions using the conjugate gradient method, restarting the method after every n steps is not always a good strategy. Such a strategy seems to be insensitive to the nonlinear character of the function being minimized. Beale [14] and later Powell [15] have proposed restart techniques that take the nonlinearity of the function into account in deciding when to restart the algorithm. Numerical experiments with minimization of several general functions have led to the following algorithm by Powell [15].

1. Given $\mathbf{x}_0$, define $\mathbf{s}_0$ to be the steepest descent direction,

$$\mathbf{s}_0 = -\nabla f(\mathbf{x}_0) = \mathbf{g}_0 ,$$

let $k = t = 0$, and begin iterations by incrementing k.

2. For $k \geq 1$ the direction $\mathbf{s}_k$ is defined by Beale's formula [14]

$$\mathbf{s}_k = -\mathbf{g}_k + \beta_k \mathbf{s}_{k-1} + \gamma_k \mathbf{s}_t, \quad \text{and} \quad \mathbf{g}_k = -\nabla f(\mathbf{x}_k) , \tag{4.2.38}$$

where

$$\beta_k = \frac{\mathbf{g}_k^T[\mathbf{g}_k - \mathbf{g}_{k-1}]}{\mathbf{s}_{k-1}^T[\mathbf{g}_k - \mathbf{g}_{k-1}]} , \tag{4.2.39}$$

and

$$\gamma_k = \frac{\mathbf{g}_k^T[\mathbf{g}_{t+1} - \mathbf{g}_t]}{\mathbf{s}_t^T[\mathbf{g}_{t+1} - \mathbf{g}_t]}, \quad \text{if } k > t+1 , \tag{4.2.40}$$

$$\gamma_k = 0, \qquad \text{if } k = t+1 . \tag{4.2.41}$$

3. For $k \geq 1$ test the inequality

$$|\mathbf{g}_{k-1}^T \mathbf{g}_k| \geq 0.2 \|\mathbf{g}_k\|^2 . \tag{4.2.42}$$

If this inequality holds, then it is taken to be an indication that enough orthogonality between $\mathbf{g}_{k-1}$and $\mathbf{g}_k$ has been lost to warrant a restart. Accordingly, t is reset $t = k - 1$ to imply restart.

4. For $k > t + 1$ the direction $\mathbf{s}_k$ is also checked to guarantee a sufficiently large gradient by testing the inequalities

$$-1.2\|\mathbf{g}_k\|^2 \leq \mathbf{s}_k^T \mathbf{g}_k \leq -0.8\|\mathbf{g}_k\|^2 . \tag{4.2.43}$$

If these inequalities are not satisfied, the algorithm is restarted by setting $t = k - 1$.

5. Finally, the algorithm is also restarted by setting $t = k - 1$, if $k - t \geq n$ as in the case of the Fletcher-Reeves method.

6. The process is terminated if $\|\mathbf{g}_{k-1}\|$ or $|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)|$ is sufficiently small. If not, k is incremented by one and the process is repeated by going to step 2.

Powell [15] has examined in great detail the effectiveness of the new restart procedure using Beale's basic algorithm on a variety of problems. These experiments clearly establish the superiority of the new procedure over the algorithms of Fletcher-Reeves and Polak-Ribiere [16]. The only disadvantage of this new algorithm appears to be its slightly increased storage requirements arising from the need for storing the vectors $\mathbf{s}_t$ and $(\mathbf{g}_{t+1} - \mathbf{g}_t)$ after a restart. More recent enhancement for the first order conjugate gradient type algorithms [17, 18] involve inclusion of certain preconditioning schemes to improve the rate of convergence.

4.2.3 *Second Order Methods*

The oldest second order method for minimizing a nonlinear multivariable function in $\mathbf{R}^n$ is Newton's method. The motivation behind Newton's method is identical to the steepest descent method. In arriving at the steepest descent direction, $\mathbf{s}$, we minimized the directional derivative, Eq. (4.2.27), subject to the condition that the Euclidean norm of $\mathbf{s}$ was unity, Eq. (4.2.28). The Euclidean norm, however, does not consider the curvature of the surface. Hence, it motivates the definition of a different norm or a metric of the surface. Thus, we pose the problem as finding the direction $\mathbf{s}$ that minimizes

$$\nabla f^T \mathbf{s} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} s_i \,, \tag{4.2.44}$$

subject to the condition that

$$\mathbf{s}^T \mathbf{Q} \mathbf{s} = 1 \,. \tag{4.2.45}$$

The solution of this problem is provided by Newton direction (see Exercise 6) to within a multiplicative constant, namely

$$\mathbf{s} = -\mathbf{Q}^{-1} \nabla f \,, \tag{4.2.46}$$

where $\mathbf{Q}$ is the Hessian of the objective function. The general form of the update equation of Newton's method for minimizing a function in $\mathbf{R}^n$ is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_{k+1} \mathbf{Q}_k^{-1} \nabla f(\mathbf{x}_k) \,, \tag{4.2.47}$$

where α_{k+1} is determined by minimizing f along the Newton direction. For $\mathbf{Q} = \mathbf{I}$, Eq. (4.2.47) yields the steepest descent solution since the norm in Eq. (4.2.45) reduces to the Euclidean norm. For quadratic functions it can be shown that the update equation reaches the optimum solution in one step with $\alpha = 1$

$$\mathbf{x}^* = \mathbf{x}_0 - [\mathbf{Q}(\mathbf{x}_0)]^{-1} [\nabla f(\mathbf{x}_0)] \,, \tag{4.2.48}$$

regardless of the initial point $\mathbf{x}_0$.

Newton's method can also shown to have a quadratic rate of convergence (see for example [4] or [8]), but the serious disadvantages of the method are the need to evaluate the Hessian $\mathbf{Q}$ and then solve the system of equations

$$\mathbf{Q}\mathbf{s} = -\nabla f\,, \tag{4.2.49}$$

to obtain the direction vector $\mathbf{s}$. For every iteration (if $\mathbf{Q}$ is non-sparse), Newton's method involves the calculation of $n(n+1)/2$ elements of the symmetric $\mathbf{Q}$ matrix, and n^3 operations for obtaining $\mathbf{s}$ from the solution of Eqs. (4.2.49). It is this feature of Newton's method that has led to the development of methods known as quasi-Newton or variable-metric methods which seek to use the gradient information to construct approximations for the Hessian matrix or its inverse.

Quasi-Newton or Variable Metric Algorithms . Consider the Taylor series expansion of the gradient of f around $\mathbf{x}_{k+1}$

$$\nabla f(\mathbf{x}_{k+1}) \simeq \nabla f(\mathbf{x}_k) + \mathbf{Q}(\mathbf{x}_{k+1} - \mathbf{x}_k)\,, \tag{4.2.50}$$

where $\mathbf{Q}$ is the actual Hessian of the function f. Assuming $\mathbf{A}_k (\mathbf{A}_k \equiv \mathbf{A}(\mathbf{x}_k))$ to be an approximation to the Hessian at the kth iteration, we may write equation (4.2.50) in a more compact form as

$$\mathbf{y}_k = \mathbf{A}_k \mathbf{p}_k\,, \tag{4.2.51}$$

where

$$\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k), \qquad \text{and} \qquad \mathbf{p}_k = \mathbf{x}_{k+1} - \mathbf{x}_k\ , \tag{4.2.52}$$

Similarly, the solution of Eq. (4.2.51) for $\mathbf{p}_k$ can be written as

$$\mathbf{B}_{k+1}\mathbf{y}_k = \mathbf{p}_k\,, \tag{4.2.53}$$

with $\mathbf{B}_{k+1}$ being an approximate inverse of the Hessian $\mathbf{Q}$. If $\mathbf{B}_{k+1}$ is to behave eventually as $\mathbf{Q}^{-1}$ then $\mathbf{B}_{k+1}\mathbf{A}_k = \mathbf{I}$. Equation (4.2.53) is known as the quasi-Newton or the secant relation. The basis for all variable-metric or quasi-Newton methods is that, the formulae which update the matrix $\mathbf{A}_k$ or its inverse $\mathbf{B}_k$ must satisfy Eq. (4.2.53) and, in addition, maintain the symmetry and positive definiteness properties. In other words, if $\mathbf{A}_k$ or $\mathbf{B}_k$ are positive definite then $\mathbf{A}_{k+1}$ or $\mathbf{B}_{k+1}$ must remain so.

A typical variable-metric algorithm with an inverse Hessian update may be stated as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_{k+1}\mathbf{s}_k\,, \tag{4.2.54}$$

where

$$\mathbf{s}_k = -\mathbf{B}_k \nabla f(\mathbf{x}_k)\,, \tag{4.2.55}$$

with $\mathbf{B}_k$ being a positive definite symmetric matrix.

Rank-One Updates. In the class of rank-one updates we have the well-known symmetric Broyden's update [19] for $\mathbf{B}_{k+1}$ given as

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{(\mathbf{p}_k - \mathbf{B}_k\mathbf{y}_k)(\mathbf{p}_k - \mathbf{B}_k\mathbf{y}_k)^T}{(\mathbf{p}_k - \mathbf{B}_k\mathbf{y}_k)^T\mathbf{y}_k}\ . \tag{4.2.56}$$

To start the algorithm, an initial positive definite symmetric matrix $\mathbf{B}_0$ is assumed and the next point $\mathbf{x}_1$ is calculated from Eq. (4.2.54). Then, Eq. (4.2.56) is used to calculate the updated approximate inverse Hessian matrix. It is easy to verify that the columns of the second matrix on the right-hand side of Eq. (4.2.56) are multiples of each other. In other words, the update matrix has a single independent column and, hence is rank-one. Furthermore, if $\mathbf{B}_k$ is symmetric then $\mathbf{B}_{k+1}$ will also be symmetric. It is, however, not guaranteed that $\mathbf{B}_{k+1}$ will remain positive definite even if $\mathbf{B}_k$ is. This fact can lead to a breakdown of the algorithm especially when applied to general non-quadratic functions. Broyden [19] suggests choosing the step lengths α_{k+1} in Eq. (4.2.54) by either (i) an exact line search, or by (ii) $\alpha_{k+1} = 1$ for all steps, or by (iii) choosing α_{k+1} such that $\|\nabla f\|$ is minimized or reduced.

Irrespective of the type of line search used, Broyden's update guarantees a quadratic termination property. However, because of the lack of robustness in minimizing general non-quadratic functions, rank-one updates have been superseded by rank-two updates which guarantee both symmetry and positive definiteness of the updated matrices.

Rank-Two Updates. Rank-two updates for the inverse Hessian approximation may generally be written as

$$\mathbf{B}_{k+1} = \left[\mathbf{B}_k - \frac{\mathbf{B}_k\mathbf{y}_k\mathbf{y}_k^T\mathbf{B}_k}{\mathbf{y}_k^T\mathbf{B}_k\mathbf{y}_k} + \theta_k\mathbf{v}_k\mathbf{v}_k^T\right]\rho_k + \frac{\mathbf{p}_k\mathbf{p}_k^T}{\mathbf{p}_k^T\mathbf{y}_k}\,, \tag{4.2.57}$$

where

$$\mathbf{v}_k = (\mathbf{y}_k^T\mathbf{B}_k\mathbf{y}_k)^{\frac{1}{2}}\left[\frac{\mathbf{p}_k}{\mathbf{p}_k^T\mathbf{y}_k} - \frac{\mathbf{B}_k\mathbf{y}_k}{\mathbf{y}_k^T\mathbf{B}_k\mathbf{y}_k}\right]\,, \tag{4.2.58}$$

and θ_k and ρ_k are scalar parameters that are chosen appropriately. Updates given by Eqs. (4.2.57) and (4.2.58) are subsets of Huang's family of updates [20] which guarantee that $\mathbf{B}_{k+1}\mathbf{y}_k = \mathbf{p}_k$ for all choices of θ_k and ρ_k. If we set $\theta_k = 0$ and $\rho_k = 1$ for all k we obtain the Davidon-Fletcher-Powell's (DFP) update formula [21, 22] which is given as

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k\mathbf{y}_k\mathbf{y}_k^T\mathbf{B}_k}{\mathbf{y}_k^T\mathbf{B}_k\mathbf{y}_k} + \frac{\mathbf{p}_k\mathbf{p}_k^T}{\mathbf{p}_k^T\mathbf{y}_k}\,. \tag{4.2.59}$$

The DFP update formula preserves the positive definiteness and symmetry of the matrices $\mathbf{B}_k$, and has some other interesting properties as well. When used for minimizing quadratic functions, it generates Q-conjugate directions and, therefore, at the nth iteration $\mathbf{B}_n$ becomes the exact inverse of the Hessian $\mathbf{Q}$. Thus, it has the features of the conjugate gradient as well as the Newton-type algorithms. The DFP algorithm can be used without an exact line search in determining α_{k+1} in Eq. (4.2.54). However, the step length must guarantee a reduction in the function value, and must be such that $\mathbf{p}_k^T\mathbf{y}_k > 0$ in order to maintain positive definiteness of $\mathbf{B}_k$. The performance of the algorithm, however, was shown to deteriorate as the accuracy of the line search decreases [20]. In most cases the DFP formula works quite successfully. In a few cases the algorithm has been known to break down because $\mathbf{B}_k$ became singular. This has led to the introduction of another update formula developed simultaneously

by Broyden [19], Fletcher [23], Goldfarb [24], and Shanno [25] and known known as BFGS formula. This formula can be obtained by putting $\theta_k = 1$ and $\rho_k = 1$ in Eq. (4.2.57) which reduces to

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \left[1 + \frac{\mathbf{y}_k^T \mathbf{B}_k \mathbf{y}_k}{\mathbf{p}_k^T \mathbf{y}_k}\right] \frac{\mathbf{p}_k \mathbf{p}_k^T}{\mathbf{p}_k^T \mathbf{y}_k} - \frac{\mathbf{p}_k \mathbf{y}_k^T \mathbf{B}_k}{\mathbf{p}_k^T \mathbf{y}_k} - \frac{\mathbf{B}_k \mathbf{y}_k \mathbf{p}_k^T}{\mathbf{p}_k^T \mathbf{y}_k} \ . \tag{4.2.60}$$

Equation (4.2.60) can also be written in a more compact manner as

$$\mathbf{B}_{k+1} = \left[\mathbf{I} - \frac{\mathbf{p}_k \mathbf{y}_k^T}{\mathbf{p}_k^T \mathbf{y}_k}\right] \mathbf{B}_k \left[\mathbf{I} - \frac{\mathbf{y}_k \mathbf{p}_k^T}{\mathbf{p}_k^T \mathbf{y}_k}\right] + \frac{\mathbf{p}_k \mathbf{p}_k^T}{\mathbf{p}_k^T \mathbf{y}_k} \ . \tag{4.2.61}$$

Using $\mathbf{A}_{k+1} = \mathbf{B}_{k+1}^{-1}$ and $\mathbf{A}_k = \mathbf{B}_k^{-1}$ we can invert the above formula to arrive at an update for the Hessian approximations. It is found that this update formula reduces to

$$\mathbf{A}_{k+1} = \mathbf{A}_k - \frac{\mathbf{A}_k \mathbf{p}_k \mathbf{p}_k^T \mathbf{A}_k}{\mathbf{p}_k^T \mathbf{A}_k \mathbf{p}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{p}_k} \, , \tag{4.2.62}$$

which is the analog of the DFP formula (4.2.59) with $\mathbf{B}_k$ replaced by $\mathbf{A}_k$, and $\mathbf{p}_k$ and $\mathbf{y}_k$ interchanged. Conversely, if the inverse Hessian $\mathbf{B}_k$ is updated by the DFP formula then the Hessian $\mathbf{A}_k$ is updated according to an analog of the DFP formula. It is for this reason that the BFGS formula is often called the complementary DFP formula. Numerical experiments with BFGS algorithm [26] suggest that it is superior to all known variable-metric algorithms. We will illustrate its use by minimizing the potential energy function of the cantilever beam problem.

Example 4.2.4

Minimize $f(x_1, x_2) = 12x_1^2 + 4x_2^2 - 12x_1x_2 + 2x_1$ by using the BFGS update algorithm with exact line searches starting with the initial guess $\mathbf{x}_0^T = (-1, -2)$.

We initiate the algorithm with a line search along the steepest descent direction. This is associated with the assumption that $\mathbf{B}_0 = \mathbf{I}$ which is symmetric and positive definite. The resulting point is previously calculated in example 4.2.3 to be

$$\mathbf{x}_1 = \begin{Bmatrix} -1.0961 \\ -1.8077 \end{Bmatrix}, \qquad \text{and} \quad \nabla f(\mathbf{x}_1) = \begin{Bmatrix} -2.6154 \\ -1.3077 \end{Bmatrix} \ .$$

From Eq. (4.2.52) we calculate

$$\mathbf{p}_0 = \begin{Bmatrix} -1.0961 \\ -1.8077 \end{Bmatrix} - \begin{Bmatrix} -1 \\ -2 \end{Bmatrix} = \begin{Bmatrix} -0.0961 \\ 0.1923 \end{Bmatrix} ,$$

$$\mathbf{y}_0 = \begin{Bmatrix} -2.6154 \\ -1.3077 \end{Bmatrix} - \begin{Bmatrix} 2 \\ -4 \end{Bmatrix} = \begin{Bmatrix} -4.6154 \\ 2.6923 \end{Bmatrix} \ .$$

Substituting the terms

$$\mathbf{p}_0^T \mathbf{y}_0 = (-0.0961)(-4.6154) + (0.1923)(2.6923) = 0.96127 \, ,$$

$$\mathbf{p}_0\mathbf{y}_0^T = \left\{ \begin{array}{c} -0.0961 \\ 0.1923 \end{array} \right\} [-4.6154 \quad 2.6923] = \left[\begin{array}{cc} 0.44354 & -0.25873 \\ -0.88754 & 0.51773 \end{array} \right],$$

into Eq. (4.2.61), we obtain

$$\mathbf{B}_1 = \left(\left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] - \frac{1}{0.96127} \left[\begin{array}{cc} 0.44354 & -0.25873 \\ -0.88754 & 0.51773 \end{array} \right] \right) \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right]$$

$$\times \left(\left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right] - \frac{1}{0.96127} \left[\begin{array}{cc} 0.44354 & -0.88754 \\ -0.25873 & 0.51773 \end{array} \right] \right) + \frac{1}{0.96127} \left[\begin{array}{cc} 0.00923 & -0.01848 \\ -0.01848 & 0.03698 \end{array} \right],$$

$$= \left[\begin{array}{cc} 0.37213 & 0.60225 \\ 0.60225 & 1.10385 \end{array} \right].$$

Next, we calculate the new move direction from Eq. (4.2.55)

$$\mathbf{s}_1 = - \left[\begin{array}{cc} 0.37213 & 0.60225 \\ 0.60225 & 1.10385 \end{array} \right] \left\{ \begin{array}{c} -2.6154 \\ -1.3077 \end{array} \right\} = \left\{ \begin{array}{c} 1.7608 \\ 3.0186 \end{array} \right\},$$

and obtain

$$\mathbf{x}_2 = \left\{ \begin{array}{c} -1.0961 \\ -1.8077 \end{array} \right\} + \alpha_2 \left\{ \begin{array}{c} 1.7608 \\ 3.0186 \end{array} \right\}.$$

Setting the derivative of $f(\mathbf{x}_2)$ with respect to α_2 to 0 yields the value $\alpha_2 = 0.4332055$, and

$$\mathbf{x}_2 = \left\{ \begin{array}{c} -0.3333 \\ -0.5000 \end{array} \right\}, \quad \text{with} \quad \nabla f(\mathbf{x}_2) \simeq \left\{ \begin{array}{c} 0 \\ 0 \end{array} \right\}.$$

This implies convergence to the exact solution. It is left to the reader to verify that if $\mathbf{B}_1$ is updated once more we obtain

$$\mathbf{B}_2 = \left[\begin{array}{cc} 0.1667 & 0.25 \\ 0.25 & 0.5 \end{array} \right],$$

which is the exact inverse of the Hessian matrix

$$\mathbf{Q} = \left[\begin{array}{cc} 24 & -12 \\ -12 & 8 \end{array} \right].$$

It can also be verified that, as expected, the directions $\mathbf{s}_0$ and $\mathbf{s}_1$ are Q-conjugate. • • •

Q-conjugacy of the directions of travel has meaning only for quadratic functions, and is guaranteed for such problems in the case of variable-metric algorithms belonging to Huang's family only if the line searches are exact. In fact, Q-conjugacy of the directions is not necessary for ensuring a quadratic termination property [26]. This realization has led to the development of methods based on the DFP and BFGS formulae that abandon the computationally expensive exact line searches. The line searches must be such that they guarantee positive definiteness of the $\mathbf{A}_k$ or $\mathbf{B}_k$ matrices while reducing the function value appropriately. Positive definiteness is

guaranteed as long as $\mathbf{p}_k^T \mathbf{y}_k > 0$. To ensure a wide radius of convergence for a quasi-Newton method, it is also necessary to satisfy the following two criteria. First, a sufficiently large decrease in the function f must be achieved for the step taken and, second, the rate of decrease of f in the direction $\mathbf{s}_k$ at $\mathbf{x}_{k+1}$ must be smaller than the rate of decrease of f at $\mathbf{x}_k$ [26]. In view of this observations, most algorithms with inexact line searches require the satisfaction of the following two conditions.

$$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k) + 10^{-4} \alpha \mathbf{s}^T \nabla f(\mathbf{x}_k) \,, \tag{4.2.63}$$

and

$$\left| \frac{\mathbf{s}^T \nabla f(\mathbf{x}_{k+1})}{\mathbf{s}^T \nabla f(\mathbf{x}_k)} \right| < 0.9 \ . \tag{4.2.64}$$

The convergence of the BFGS algorithm under these conditions has been studied by Powell [27]. Similar convergence studies with Beale's restarted conjugate gradient method under the same two conditions have been carried out by Shanno [28].

4.2.4 *Applications to Analysis*

Several of the algorithms for unconstrained minimization of functions in $\mathbf{R}^n$ can also be used for solving a system of linear or nonlinear equations. In some cases, like the problems of nonlinear structural analysis, the necessary condition for the potential energy to be stationary is that its gradient vanish. The latter can be construed as solving a system of equations of the type

$$\nabla f(\mathbf{x}) = g(\mathbf{x}) = 0 \,, \tag{4.2.65}$$

where the Hessian of f and the Jacobian of $\mathbf{g}$ are the same. In cases where the problems are posed directly as

$$\mathbf{g}(\mathbf{x}) = 0 \,, \tag{4.2.66}$$

Dennis and Schnabel [6] and others solve Eq. (4.4.2) by minimizing the nonlinear least squares function

$$f = \frac{1}{2} \mathbf{g}^T \mathbf{g} \ . \tag{4.2.67}$$

In this case, however, the Hessian of f and the Jacobian of $\mathbf{g}$ are not identical but a positive definite approximation to the Hessian of f appropriate for most minimization schemes can be easily generated from the Jacobian of $\mathbf{g}$ [6]. Minimization of f then permits the determination of not only stable but also unstable equilibrium configurations provided the minimization does not converge to a local minimum. In the case of convergence to a local minimum, certain restart [6] or deflation and tunnelling techniques [29, 30] can be invoked to force convergence to the global minimum of f at which $\|\mathbf{g}\| = 0$.

4.3 Specialized Quasi-Newton Methods

4.3.1 *Exploiting Sparsity*

The rank-one and rank-two updates that we discussed in the previous section yield updates which are symmetric but not necessarily sparse. In other words the Hessian or Hessian inverse updates lead to symmetric matrices which are fully populated. In most structural analysis problems using the finite element method it is well known that the Hessian of the potential energy (the tangent stiffness matrix) is sparse. This may be also true of many structural optimization problems. For such sparse systems the solution phase for finite element models exploits the triple $\mathbf{LDL}^{\mathrm{T}}$ factorization. Thus the Hessian or the Hessian inverse updates discussed previously are not appropriate for solving large-scale structural analysis problems which involve sparse Hessians.

In applying the BFGS method for solving large-scale nonlinear problems of structural analysis Matthies and Strang [31] have proposed an alternate implementation of the method suitable for handling large sparse problems by storing the vectors

$$\mathbf{p}_k = \mathbf{x}_{k+1} - \mathbf{x}_k \,, \tag{4.3.1}$$

and

$$\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k) \,, \tag{4.3.2}$$

and reintroducing them to compute the new search directions. After a sequence of five to ten iterations during which the BFGS updates are used, the stiffness matrix is recomputed and the update information is deleted.

Sparse updates for solving large-scale problems were perhaps first proposed by Schubert [32], who proposed a modification of Broyden's method [33] according to which the ith row of the Hessian $\mathbf{A}_{k+1}$ is updated by using

$$\mathbf{A}_{k+1}^{(i)} = \mathbf{A}_k^{(i)} + \frac{[\mathbf{g}_i(\mathbf{x}_{k+1}) - (1-\alpha_k)\mathbf{g}(\mathbf{x}_k)]}{\alpha_k \hat{\mathbf{p}}_k^T \hat{\mathbf{p}}_k} \,, \tag{4.3.3}$$

with $\hat{\mathbf{p}}_k$ obtained from $\mathbf{p}_k$ by setting to zero those components corresponding to known zeros in $\mathbf{A}_k^{(i)}$. The method has the drawback, however, that it cannot retain symmetry of the resulting matrix even when starting with a symmetric, positive definite matrix. Not only does this result in slightly increased demands on storage, but it also requires special sparse linear equation solvers. Toint [34] and Shanno [35] have recently proposed algorithms which find updating formulae for symmetric Hessian matrices that preserve known sparsity conditions. The update is obtained by calculating the smallest correction subject to linear constraints that include the sparsity conditions. This involves the solution of a system of equations with the same sparsity pattern as the Hessian.

Curtis, Powell and Reid [36], and Powell and Toint [37] have proposed finite difference strategies for the direct evaluation of sparse Hessians of functions. In addition to using the finite difference operations, they used concepts from graph theory that minimize the number of gradient evaluations required for computing the few non-zero entries of a sparse Hessian. By using these strategies, we can exploit the sparsity not only in the computation of the Newton direction but also in the formation of Hessians [38, 39]

The Curtis-Powell-Reid (CPR) strategy exploits sparsity, but not the symmetry of the Hessian. It divides the columns of the Hessian into groups, so that in each group the row numbers of the unknown elements of the column vectors are all different. After the formation of the first group, other groups are formed successively by applying the same strategy to columns not included in the previous groups. The number of such groups for sparse or banded matrices is usually very small by comparison with n. To evaluate the Hessian of f at $\mathbf{x}_0$ we evaluate the gradient of f at $\mathbf{x}_0$. After this initial gradient evaluation, only as many more gradient evaluations as the number of groups are needed to evaluate all the non-zero elements of the Hessian using forward difference approximation. Thus

$$a_{ij} = \frac{\partial g_i}{\partial x_j} = \frac{\mathbf{g}_i(\mathbf{x}_0 + h_j \mathbf{e}_j) - \mathbf{g}_i(\mathbf{x}_0)}{h_j}\,, \tag{4.3.4}$$

where $\mathbf{e}_j$ is the jth coordinate vector and h_j is a suitable step size. Each step size may be adjusted such that the greatest ratio of the round-off to truncation error for any column of the Hessian falls within a specified range. However, such an adjustment of step sizes would require a significantly large number of gradient evaluations. Hence, to economize on the number of gradient evaluations the step sizes are not allowed to leave the range

$$[\max(\epsilon|\mathbf{x}_j|, \eta h_{uj}), h_{uj}]\,, \tag{4.3.5}$$

where ϵ is the greatest relative round-off in a single operation, η is the relative machine precision, and h_{uj} is an upper bound on h_j [36].

Powell and Toint [37] extended the CPR strategy to exploit symmetry of the Hessian. They proposed two methods, one of which is known as the substitution method. According to this, the CPR strategy is first applied to the lower triangular part, $\mathbf{L}$, of the symmetric Hessian, $\mathbf{A}$. Because, all the elements of $\mathbf{A}$ computed this way will not be correct, the incorrect elements are corrected by a back-substitution scheme. Details of this back-substitution scheme may be found in Ref. 37.

The Powell-Toint (PT) strategy of estimating sparse Hessians directly appears to be a much better alternative to Toint's sparse update algorithm [38]. One major drawback of Toint's update algorithm is that the updated Hessian approximation is not guaranteed to remain positive definite even if the initial Hessian approximation was positive definite.

4.3.2 *Coercion of Hessians for Suitability with Quasi-Newton Methods*

In minimizing a multivariable function f using a discrete Newton method or the Toint's update algorithm we must ensure that the Hessian approximation is positive

definite. If this is not so, then Newton's direction is not guaranteed to be a descent direction. There are several strategies for coercing an indefinite Hessian to a positive definite form. Prominent among these strategies is the one proposed by Gill and Murray [40]. The most impressive feature of this strategy is that the coercion of the Hessian takes place during its $\mathbf{LDL}^T$ decomposition for the computation of the Newton direction. The diagonal elements of the $\mathbf{D}$ matrix are forced to be sufficiently positive to avoid numerical difficulties while the off-diagonal terms of $\mathbf{LD}^{1/2}$ are limited by a quantity designed to guarantee positive definiteness of the resulting matrix. This is equivalent to modifying the original non-positive definite Hessian matrix by the addition of an appropriate diagonal matrix. Because this matrix modification is carried out during its $\mathbf{LDL}^T$ decomposition, the strategy for the computation of Newton's descent direction does not entail a great deal of additional computations.

4.3.3 *Making Quasi-Newton Methods Globally Convergent*

It is well known that despite a positive definite Hessian approximation, Newton's method can diverge for some starting points. Standard backtracking along the Newton direction by choosing shorter step lengths can achieve convergence to the minimum. However, backtracking along the Newton direction fails to use the n-dimensional quadratic model of the function f. Dennis and Schnabel [7] have proposed a strategy called the double-dogleg strategy which uses the full n-dimensional quadratic model to choose a new direction obtained by a linear combination of the steepest descent and the Newton direction. This new direction is a function of the radius of the trust region within which the n-dimensional quadratic model of the function approximates the true function well. The double-dogleg strategy not only makes Newton's method globally convergent (that is converge to the minimum of the function irrespective of the starting point) but also makes it significantly more efficient for certain poorly scaled problems. For details about the double-dogleg strategy readers are advised to consult Ref. 7. More recent attempts to widen the domain of convergence of the quasi-Newton method or make it globally convergent for a wide class of problems are studied in Refs. [41, 42].

4.4 Probabilistic Search Algorithms

A common disadvantage of most of the algorithms discussed so far is their inability to distinguish local and global minima. Many structural design problems have more than one local minimum, and depending on the starting point, these algorithms may converge to one of these local minima. The simplest way to check for a better local solution is to restart the optimization from randomly selected initial points to check if other solutions are possible. However, for problems with a large number of variables the possibility of missing the global minimum is large unless unpractically large number of optimization runs are performed. The topic of global optimization is an area of active research where new algorithms are emerging and old algorithms are constantly being improved [43–45].

Dealing with the problem of local minima becomes even worse if the design variables are required to take discrete values. First of all, for such problems the design space is discontinuous and disjointed, therefore derivative information is either useless or is not defined. Secondly, the use of discrete values for the design variables introduces multiple minima corresponding to various combinations of the variables, even if the objective function for the problem has a single minimum for continuous variables. A methodical way of dealing with multiple minima for discrete optimization problems is to use either random search techniques that would sample the design space for a global minimum or to employ enumerative type algorithms. In either case, the efficiency of the solution process deteriorates dramatically as the number of variables is increased.

Two algorithms, *Simulated Annealing* and *Genetic Algorithms* (see, Laarhoven [46] and Goldberg [47], respectively), have emerged more recently as tools ideally suited for optimization problems where a global minimum is sought. In addition to being able to locate near global solutions, these two algorithms are also powerful tools for problems with discrete-valued design variables. Both algorithms rely on naturally observed phenomena and their implementation calls for the use of a random selection process which is guided by probabilistic decisions. In the following sections brief descriptions of the two algorithms are presented. Application of the algorithms to structural design will be demonstrated for laminated composites in Chapter 11.

4.4.1 *Simulated Annealing*

The development of the simulated annealing algorithm was motivated by studies in statistical mechanics which deal with the equilibrium of large number of atoms in solids and liquids at a given temperature. During solidification of metals or formation of crystals, for example, a number of solid states with different internal atomic or crystalline structure that correspond to different energy levels can be achieved depending on the rate of cooling. If the system is cooled too rapidly, it is likely that the resulting solid state would have a small margin of stability because the atoms will assume relative positions in the lattice structure to reach an energy state which is only locally minimal. In order to reach a more stable, globally minimum energy state, the process of annealing is used in which the metal is reheated to a high temperature and cooled slowly, allowing the atoms enough time to find positions that minimize a steady state potential energy. It is observed in the natural annealing process that during the time spent at a given temperature it is possible to have the system jump to a higher energy state temporarily before the steady state is reached. As will be explained in the following paragraphs, it is this characteristic of the annealing process which makes it possible to achieve near global minimum energy states.

A computational algorithm that simulates the annealing process was proposed by Metropolis et al. [48], and is referred to as the *Metropolis algorithm*. At a given temperature, T, the algorithm perturbs the position of an atom randomly and computes the resulting change in the energy of the system, ΔE. If the new energy state is lower than the initial state, then the new configuration of the atoms is accepted. If, on the other hand $\Delta E \geq 0$, the perturbed state causes an increase in the energy,

the new state might be accepted or rejected based on a random probabilistic decision. The probability of acceptance, $\mathcal{P}(\Delta E)$, of a higher energy state is computed as

$$\mathcal{P}(\Delta E) = e^{\left(\frac{-\Delta E}{k_B T}\right)}, \tag{4.4.1}$$

where k_B is the Boltzmann's constant. If the temperature of the system is high, then the probability of acceptance of a higher energy state is close to one. If, on the other hand, the temperature is close to zero, then the probability of acceptance becomes very small.

The decision to accept or reject is made by randomly selecting a number in an interval $(0, 1)$ and comparing it with $\mathcal{P}(\Delta E)$. If the number is less than $\mathcal{P}(\Delta E)$, then the perturbed state is accepted, if it is greater than $\mathcal{P}(\Delta E)$, the state is rejected. At each temperature, a pool of atomic structures would be generated by randomly perturbing positions until a steady state energy level is reached (commonly referred to as thermal equilibrium). Then the temperature is reduced to start the iterations again. These steps are repeated iteratively while reducing the temperature slowly to achieve the minimal energy state.

The analogy between the simulated annealing and the optimization of functions with many variables was established recently by Kirkpatrick et al. [49], and Cerny [50]. By replacing the energy state with an objective function f, and using variables $\mathbf{x}$ for the the configurations of the particles, we can apply the Metropolis algorithm to optimization problems. The method requires only function values. The moves in the design space from one point, $\mathbf{x}^i$ to another $\mathbf{x}^j$ causes a change in the objective function, Δf^{ij}. The temperature T now becomes a control parameter that regulates the convergence of the process. Important elements that affect the performance of the algorithm are the selection of the initial value of the "temperature", T_0, and how to update it. In addition, the number of iterations (or combinations of design variables) needed to achieve "thermal equilibrium" must be decided before the T can be reduced. These parameters are collectively referred to as the "cooling schedule".

A flow chart of a typical simulated annealing algorithm is shown in Figure 4.4.1. The definition of the cooling schedule begins with the selection of the initial temperature. If a low value of T_0 is used, the algorithm would have a low probability of reaching a global minimum. The initial value of T_0 must be high enough to permit virtually all moves in the design space to be acceptable so that almost a random search is performed. Typically, T_0 is selected such that the acceptance ratio $\mathcal{X}$ (defined as the ratio of the number of accepted moves to total number of proposed moves) is approximately $\mathcal{X}_0 = 0.95$ [51]. Johnson et al. [52] determined T_0 by calculating the average increase in the objective function, $\overline{\Delta f}^{(+)}$, over a predetermined number of moves and solved

$$\mathcal{X}_0 = e^{\left(\frac{-\overline{\Delta f}^{(+)}}{T_0}\right)}, \tag{4.4.2}$$

leading to

$$T_0 = \frac{\overline{\Delta f}^{(+)}}{\ln(\mathcal{X}_0^{-1})}. \tag{4.4.3}$$

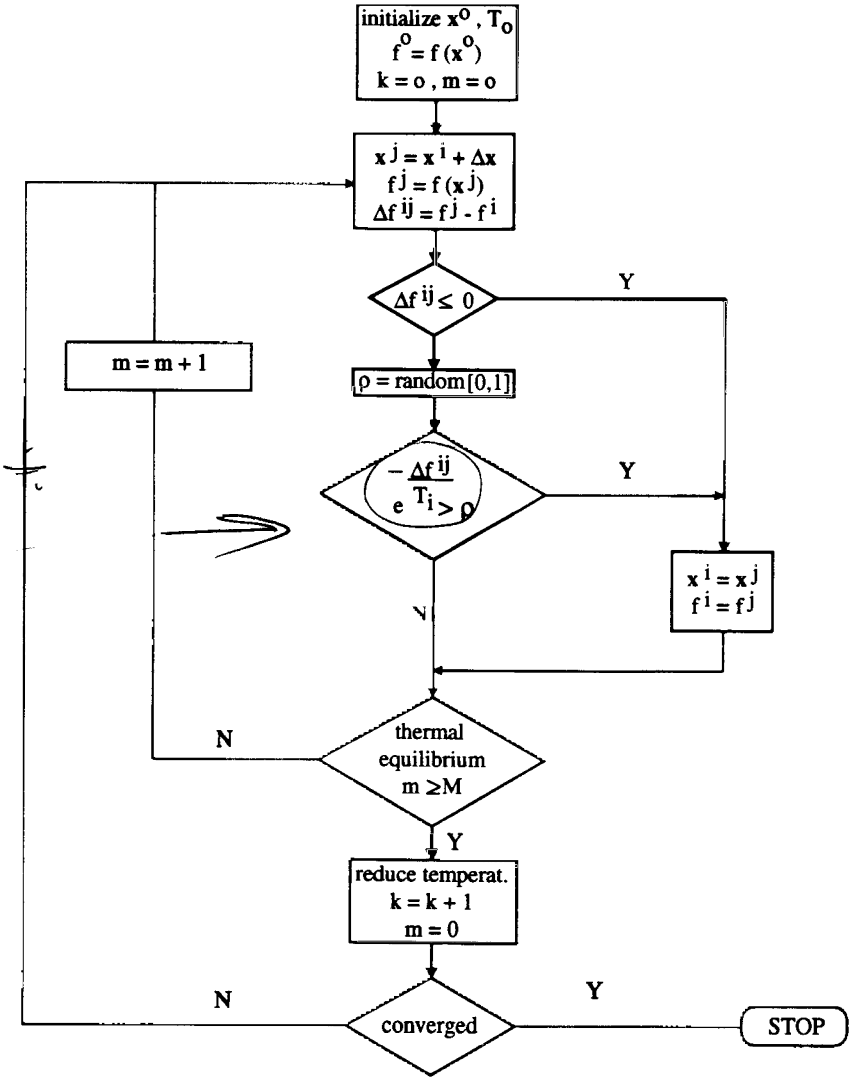

Figure 4.4.1 Flow chart of the simulated annealing algorithm.

Once the temperature is set, a number of moves in the variable space is performed by perturbing the design. The number of moves at a given temperature must be large enough to allow the solution to escape from a local minimum. One possibility is to move until the value of the objective function does not change for a specified number, M, of successive iterations. Another possibility suggested by Aarts [53] for discrete valued design variables is to make sure that every possible combinations of design variables in the neighborhood of a steady state design is visited at least once with a

probability of P. That is, if there are S neighboring designs, then

$$M = S \ln \left(\frac{1}{1-P} \right) , \tag{4.4.4}$$

where $P = 0.99$ for $S > 100$, and $P = 0.995$ for $S < 100$. For discrete valued variables there are often many options for defining the neighborhood of the design. One possibility is to define it as all the designs that can be obtained by changing one design variable to its next higher or lower value. A broader immediate neighborhood can be defined by changing more than one design variables to their next higher or lower values. For an n variable problem, the immediate neighborhood has

$$S = 3^n - 1 \ . \tag{4.4.5}$$

Once convergence is achieved at a given temperature, generally referred to as thermal equilibrium, the temperature is reduced and the process is repeated.

Many different schemes have been proposed for updating the temperature. A frequently used rule is a constant cooling update

$$T_{k+1} = \alpha T_k, \qquad k = 0, 1, 2, \ldots, K \, , \tag{4.4.6}$$

where $0.5 \leq \alpha \leq 0.95$. Nahar [54] fixes the number of decrement steps K, and suggests determination of the values of the T_k experimentally. It is also possible to divide the interval $[0, T_0]$ into a fixed K number of steps and use

$$T_K = \frac{K-k}{K} T_0, \qquad k = 1, 2, \ldots, K \ . \tag{4.4.7}$$

The number of intervals typically ranges from 5 to 20.

The use of simulated annealing for structural optimization has been quite recent. Elperin [55] applied the method to the design of a ten-bar truss problem where member cross-sectional dimensions were to be selected from a set of discrete values. Kincaid and Padula [56] used it for minimizing the distortion and internal forces in a truss structure. A 6-story 156 member frame structure with discrete valued variables was considered by Balling and May [57]. Optimal placement of active and passive members in a truss structure was investigated by Chen et al. [58] to maximize the finite-time energy dissipation to achieve increased damping properties.

4.4.2 *Genetic Algorithms*

Genetic algorithms use techniques derived from biology, and rely on the principle of Darwin's theory of survival of the fittest. When a population of biological creatures is allowed to evolve over generations, individual characteristics that are useful for survival tend to be passed on to the future generations, because individuals carrying them get more chances to breed. Those individual characteristics in biological populations are stored in chromosomal strings. The mechanics of natural genetics is based on operations that result in structured yet randomized exchange of genetic

information (i.e., useful traits) between the chromosomal strings of the reproducing parents, and consists of *reproduction*, *crossover*, occasional *mutation*, and *inversion* of the chromosomal strings.

Genetic algorithms, developed by Holland [59], simulate the mechanics of natural genetics for artificial systems based on operations which are the counterparts of the natural ones (even called by the same names), and are extensively used as multivariable search algorithms. As will be described in the following paragraphs, these operations involve simple, easy to program, random exchanges of location of numbers in a string, and, therefore, at the outset look like a completely random search of extremum in the parameter space based on function values only. However, genetic algorithms are experimentally proven to be robust, and the reader is referred to Goldberg [47] for further discussion of the theoretical properties of genetic algorithms. Here we discuss the genetic representation of a minimization problem, and focus on the mechanics of three commonly used genetic operations, namely; reproduction, crossover, and mutation.

Application of the operators of the genetic algorithm to a search problem first requires the representation of the possible combinations of the variables in terms of bit strings that are counterparts of the chromosomes. Naturally, the measure of goodness of a specific combination of genes is represented in an artificial system by the objective function of the search problem. For example, if we have a minimization problem

$$\textbf{minimize} \qquad f(\mathbf{x}), \qquad \mathbf{x} = \{x_1, x_2, x_3, x_4\}\,, \tag{4.4.8}$$

a binary string representation of the variable space could be of the form

$$\underbrace{0\,1\,1\,0}_{x_1}\ \underbrace{1\,0\,1}_{x_2}\ \underbrace{1\,1}_{x_3}\ \underbrace{1\,0\,1\,1}_{x_4}, \tag{4.4.9}$$

where string equivalents of the individual variables are connected head-to-tail, and, in this example, base 10 values of the variables are $x_1 = 6, x_2 = 5, x_3 = 3, x_4 = 11$, and their ranges correspond to $\{15 \geq x_1, x_4 \geq 0\}, \{7 \geq x_2 \geq 0\}$, and $\{3 \geq x_3 \geq 0\}$. Because of the bit string representation of the variables, genetic algorithms are ideally suited for problems where the variables are required to take discrete or integer variables. For problems where the design variables are continuous values within a range $x_i^L \leq x_i \leq x_i^U$, one may need to use a large number of bits to represent the variables to high accuracy. The number of bits that are needed depends on the accuracy required for the final solution. For example, if a variable is defined in a range $\{0.01 \leq x_i \leq 1.81\}$ and the accuracy needed for the final value is $x^{\text{incr}} = 0.001$, then the number of binary digits needed for an appropriate representation can be calculated from

$$2^m \geq \left((x_i^U - x_i^L)/x^{\text{incr}} + 1\right)\,, \tag{4.4.10}$$

where m is the number of digits. In this example, the smallest number of digits that satisfy the requirement would be $m = 11$, which actually produces increments of 0.00087 in the value of the variable, instead of the required value of 0.001.

Unlike the search algorithms discussed earlier that move from one point to another in the design variable space, genetic algorithms work with a population of strings

(chromosomes). This aspect of the genetic algorithms is responsible for capturing near global solutions, by keeping many solution points that may have the potential of being close to minima (local or global) in the pool during the search process rather than singling out a point early in the process and running the risk of getting stuck at a local minimum. Working on a population of designs also suggests the possibility of implementation on parallel computers. However, the concept of parallelism is even more basic to genetic algorithms in that evolutionary selection can improve in parallel many different characteristics of the design. Also, the outcome of a genetic search is a population of good designs rather than a single design. This aspect can be very useful to the designer.

Initially the size of the population is chosen and the values of the variables in each string are decided by randomly assigning 0's and 1's to the bits. The next important step in the process is *reproduction*, in which individual strings with good objective function values are copied to form a new population, an artificial version of the survival of the fittest. The bias towards strings with better performance can be achieved by increasing the probability of their selection in relation to the rest of the population. One way to achieve this is to create a biased roulette wheel where individual strings occupy areas proportional to their function values in relation to the cumulative function value of the entire population. Therefore, the population resulting from the reproduction operation would have multiple copies of the highly fit individuals.

Once the new population is generated, the members are paired off randomly for crossover. The mating of the pair also involves a random process. A random integer k between 1 and $L-1$, where L is the string length, is selected and two new strings are generated by exchanging the 0's and 1's that comes after the kth location in the first parent with the corresponding locations of the second parent. For example, the two strings of length $L = 9$

$$\begin{array}{ll} \textbf{parent 1:} & 0\ 1\ 1\ 0\ 1 \| 0\ 1\ 1\ 1 \\ \textbf{parent 2:} & 0\ 1\ 0\ 0\ 1 \| 0\ 0\ 0\ 1 \end{array}, \tag{4.4.11}$$

are mated with a crossover point of $k = 5$, the offsprings will have the following composition,

$$\begin{array}{ll} \textbf{offspring 1:} & 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1 \\ \textbf{offspring 2:} & 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1 \end{array}. \tag{4.4.12}$$

Multiple point crossovers in which information between the two parents are swapped among more string segments are also possible, but because of the mixing of the strings the crossover becomes a more random process and the performance of the algorithm might degrade, De Jong [60]. Exception to this is the two-point crossover. In fact, the one point crossover can be viewed as a special case of the two point crossover in which the end of the string is the second crossover point. Booker [61] showed that by choosing the end-point of the segment to be crossed randomly, the performance of the algorithm can actually be improved.

Mutation serves an important task of preventing premature loss of important genetic information by occasional introduction of random alteration of a string. As

mentioned earlier, at the end of reproduction it is possible to have populations with multiple copies of the same string. In the worst scenario, it is possible to have the entire pool to be made of the same string. In such a case, the algorithm would be unable to explore the possibility of a better solution. Mutation prevents this uniformity, and is implemented by randomly selecting a string location and changing its value from 0 to 1 or vice versa. Based on small rate of occurrence in biological systems and on numerical experiments, the role of the mutation operation on the performance of a genetic algorithm is considered to be a secondary effect. Goldberg [49] suggests a rate of mutation of one in one thousand bit operations.

Application of genetic algorithms in optimal structural design has started only recently. The first application of the algorithm to a structural design was presented by Goldberg and Samtani [62] who used the 10-bar truss weight minimization problem. More recently, Hajela [63] used genetic search for several structural design problems for which the design space is known to be either nonconvex or disjoint. Rao et al. [64] address the optimal selection of discrete actuator locations in actively controlled structures via genetic algorithms.

In closing, the basic ideas behind the simulation of a natural phenomena is finding a more mathematically sound foundation in the area of probabilistic search algorithms, especially for discrete variables. Improvements in the performance of the algorithms are constantly being made. For example, modifications in the cooling schedule proposed by Szu [65] led to the development of a new algorithm know as the fast simulated annealing. Applications and analysis of other operations that mimic the natural biological genetics (such as inversion, dominance, niches, etc.) are currently being evaluated for genetic algorithms.

4.5 Exercises

1. Solve the problem of the cantilever beam problem of example 4.2.1 by

   (a) Nelder-Mead's simplex algorithm, and

   (b) Davidon-Fletcher-Powell's algorithm.

   Begin with $\mathbf{x}_0^T = (-1, -2)$. For the simplex algorithm assume an initial simplex of size a=2.0. Assume an initial base point $\mathbf{x}_0$ with the coordinates of the other vertices to be given by Eqs. (4.2.1) and (4.2.1).

2. Find the minimum of the function

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

using Powell's conjugate directions method, starting with $\mathbf{x}_0 = (0, 0, 0)^T$.

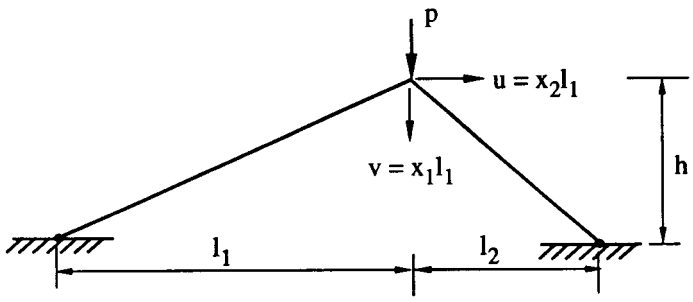

Figure 4.5.1 Two bar unsymmetric shallow truss.

3. Determine the minimum of

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \,,$$

using steepest descent method, starting with $\mathbf{x}_0 = (1.2, 1.0)^T$.

4. The stable equilibrium configuration of the two bar unsymmetric shallow truss of Figure 4.5.1 can be obtained by minimizing the potential energy function f of the non-dimensional displacement variables x_1, x_2 as

$$f(x_1, x_2) = \frac{1}{2} m\gamma(-\alpha_1 x_1 + \frac{1}{2}x_1^2 + x_2)^2 + \frac{1}{2}\left(-\alpha_1 x_1 + \frac{1}{2}x_1^2 - \frac{x_2}{\gamma}\right)\gamma^4 - \overline{p}\gamma x_1 \,,$$

where $m, \gamma, \alpha_1, \overline{p}$ are nondimensional quantities defined as

$$m = \frac{A_1}{A_2}, \qquad \gamma = \frac{l_1}{l_2}, \qquad \alpha_1 = \frac{h}{l_1}, \qquad \overline{p} = \frac{p}{EA_2} \,,$$

and E is the elastic modulus, A_1 and A_2 are the cross-sectional areas of the bars. Using the BFGS algorithm determine the equilibrium configuration in terms of x_1 and x_2 for $m = 5, \gamma = 4, \alpha_1 = 0.02, \overline{p} = 2 \times 10^{-5}$. Use $\mathbf{x}_0^T = (0, 0)$.

5. Continuing the analysis of the problem 4 it can be shown that the critical load p_{cr} at which the shallow truss is unstable (snap-through instability) is given by

$$p_{\text{cr}} = \frac{EA_1A_2\gamma(\gamma+1)^2}{(A_1 + A_2\gamma)} \frac{\alpha_1^3}{3\sqrt{3}} \,.$$

Suppose now that p_{cr} as given above is to be maximized subject to the condition that

$$A_1 l_1 + A_2 l_2 = v_0 = \text{constant} \,.$$

The exterior penalty formulation of Chapter 5 reduces the above problem to the unconstrained minimization of

$$p_{\text{cr}}^*(A_1, A_2, r) = \frac{EA_1A_2\gamma(\gamma+1)^2}{(A_1 + A_2\gamma)} \frac{\alpha_1^3}{3\sqrt{3}} + r(A_1 l_1 + A_2 l_2 - v_0)^2 \,,$$

where r is a penalty parameter. Carry out the minimization of an appropriately nondimensionalized form of $p^*_{\rm cr}$ for $l_1 = 200$ in, $l_2 = 50$ in, $h = 2.50$ in, $v_0 = 200$ in^3, $E = 10^6$ psi, $r = 10^4$ to determine an approximate solution for the optimum truss configuration and the corresponding value of $p_{\rm cr}$. Use the BFGS algorithm for unconstrained minimization beginning with an initial feasible guess of $A_1 = 0.952381$in^2 and $A_2 = 0.190476$in^2.

6. a) Minimize the directional derivative of f in the direction $\mathbf{s}$

$$\nabla f^T \mathbf{s} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} s_i \,,$$

subject to the condition

$$\sum_{i=1}^{n} s_i^2 = 1 \,,$$

to show that the steepest descent direction is given by

$$\mathbf{s} = -\frac{\nabla f}{\|\nabla f\|} \,. \tag{4.5.1}$$

b) Repeat the above with the constraint condition on $\mathbf{s}$ replaced by

$$\mathbf{s}^T \mathbf{Q} \mathbf{s} = 1 \,,$$

to show that the Newton direction is given by

$$\mathbf{s} = -\mathbf{Q}^{-1} \nabla f \,,$$

$\mathbf{Q}$ being the Hessian of the quadratic function f.

4.6 References

[1] Kamat, M.P. and Hayduk, R.J., "Recent Developments in Quasi–Newton Methods for Structural Analysis and Synthesis," AIAA J., 20 (5), 672–679, 1982.

[2] Avriel, M., Nonlinear Programming: Analysis and Methods. Prentice–Hall, Inc., 1976.

[3] Powell, M.J.D., "An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives," Computer J., 7, pp. 155–162, 1964.

[4] Kiefer, J., "Sequential Minmax Search for a Maximum," Proceedings of the American Mathematical Society, 4, pp. 502–506, 1953.

[5] Walsh, G.R., Methods of Optimization, John Wiley, New York, 1975.

[6] Dennis, J.E. and Schnabel, R.B., Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice–Hall, 1983.

[7] Gill, P.E., Murray, W. and Wright, M.H., Practical Optimization, Academic Press, New York, p. 92, 1981.

[8] Spendley, W., Hext, G. R., and Himsworth, F. R., "Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation," Technometrics, 4 (4), pp. 441–461, 1962.

[9] Nelder, J. A. and Mead, R., "A Simplex Method for Function Minimization," Computer J., 7, pp. 308–313, 1965.

[10] Chen, D. H., Saleem, Z., and Grace, D. W., "A New Simplex Procedure for Function Minimization," Int. J. of Modelling & Simulation, 6, 3, pp. 81–85, 1986.

[11] Cauchy, A., "Methode Generale pour la Resolution des Systemes D'equations Simultanees," Comp. Rend. l'Academie des Sciences Paris, 5, pp. 536–538, 1847.

[12] Hestenes, M.R. and Stiefel, E., "Methods of Conjugate Gradients for Solving Linear Systems," J. Res. Nat. Bureau Stand., 49, pp. 409–436, 1952.

[13] Fletcher, R. and Reeves, C.M., "Function Minimization by Conjugate Gradients," Computer J., 7, pp. 149–154, 1964.

[14] Gill, P.E. and Murray, W.,"Conjugate-Gradient Methods for Large Scale Nonlinear Optimization," Technical Report 79-15; Systems Optimization Lab., Dept. of Operations Res., Stanford Univ., pp. 10–12, 1979.

[15] Powell, M.J.D., "Restart Procedures for the Conjugate Gradient Method," Math. Prog., 12, pp. 241–254, 1975.

[16] Polak, E., Computational Methods in Optimization: A Unified Approach, Academic Press, 1971.

[17] Axelsson, O. and Munksgaard, N., "A Class of Preconditioned Conjugate Gradient Methods for the Solution of a Mixed Finite Element Discretization of the Biharmonic Operator," Int. J. Num. Meth. Engng., 14, pp. 1001–1019, 1979.

[18] Johnson, O.G., Micchelli, C.A. and Paul, G., "Polynomial Preconditioners for Conjugate Gradient Calculations," SIAM J. Num. Anal., 20 (2), pp. 362–376, 1983.

[19] Broyden, C.G., "The Convergence of a Class of Double–Rank Minimization Algorithms 2. The New Algorithm," J. Inst. Math. Appl., 6, pp. 222–231, 1970.

[20] Oren, S.S. and Luenberger, D., "Self–scaling Variable Metric Algorithms, Part I," Manage. Sci., 20 (5), pp. 845–862, 1974.

[21] Davidon, W.C., Variable Metric Method for Minimization. Atomic Energy Commission Research and Development Report, ANL–5990 (Rev.), November 1959.

[22] Fletcher, R. and Powell, M.J.D., "A Rapidly Convergent Descent Method for Minimization," Computer J., 6, pp. 163–168, 1963.

[23] Fletcher, R., "A New Approach to Variable Metric Algorithms," Computer J., 13 (3), pp. 317–322, 1970.

[24] Goldfarb, D., "A Family of Variable-metric Methods Derived by Variational Means," Math. Comput., 24, pp. 23–26, 1970.

[25] Shanno, D.F., "Conditioning of Quasi–Newton Methods for Function Minimization," Math. Comput., 24, pp. 647–656, 1970.

[26] Dennis, J.E., Jr. and More, J.J., "Quasi-Newton Methods, Motivation and Theory," SIAM Rev., 19 (1), pp. 46–89, 1977.

[27] Powell, M.J.D., "Some Global Convergence Properties of a Variable Metric Algorithm for Minimization Without Exact Line Searches," In: Nonlinear Programming (R.W.Cottle and C.E. Lemke, eds.), American Mathematical Society, Providence, RI, pp. 53–72, 1976.

[28] Shanno, D.F., "Conjugate Gradient Methods with Inexact Searches," Math. Oper. Res., 3 (2), pp. 244–256, 1978.

[29] Kamat, M.P., Watson, L.T. and Junkins, J.L., "A Robust Efficient Hybrid Method for Finding Multiple Equilibrium Solutions," Proceedings of the Third Intl. Conf. on Numerical Methods in Engineering, Paris, France, pp. 799–807, March 1983.

[30] Kwok, H.H., Kamat, M.P. and Watson, L.T., "Location of Stable and Unstable Equilibrium Configurations using a Model Trust Region, Quasi-Newton Method and Tunnelling," Computers and Structures, 21 (6), pp. 909–916, 1985.

[31] Matthies, H. and Strang, G., "The Solution of Nonlinear Finite Element Equations," Int. J. Num. Meth. Enging., 14, pp. 1613–1626, 1979.

[32] Schubert, L.K., "Modification of a Quasi-Newton Method for Nonlinear Equations with a Sparse Jacobian," Math. Comput., 24, pp. 27–30, 1970.

[33] Broyden, C.G., "A Class of Methods for Solving Nonlinear Simultaneous Equations," Math. Comput., 19, pp. 577–593, 1965.

[34] Toint, Ph.L., "On Sparse and Symmetric Matrix Updating Subject to a Linear Equation," Math. Comput., 31, pp. 954–961, 1977.

[35] Shanno, D.F., "On Variable-Metric Methods for Sparse Hessians," Math. Comput., 34, pp. 499–514, 1980.

[36] Curtis, A.R., Powell, M.J.D. and Reid, J.K., "On the Estimation of Sparse Jacobian Matrices," J. Inst. Math. Appl., 13, pp. 117–119, 1974.

[37] Powell, M.J.D. and Toint, Ph.L., "On the Estimation of Sparse Hessian Matrices," SIAM J. Num. Anal., 16 (6), pp. 1060–1074, 1979.

[38] Kamat, M.P., Watson, L.T. and VandenBrink, D.J., "An Assessment of Quasi-Newton Sparse Update Techniques for Nonlinear Structural Analysis," Comput. Meth. Appl. Mech. Enging., 26, pp. 363–375, 1981.

[39] Kamat, M.P. and VandenBrink, D.J., "A New Strategy for Stress Analysis Using the Finite Element Method," Computers and Structures 16 (5), pp. 651–656, 1983.

[40] Gill, P.E. and Murray, W., "Newton-type Methods for Linearly Constrained Optimization," In: Numerical Methods for Constrained Optimization (Gill & Murray, eds.), pp. 29–66. Academic Press, New York 1974.

[41] Griewank, A.O., Analysis and Modifications of Newton's Method at Singularities. Ph.D. Thesis, Australian National University, 1980.

[42] Decker, D.W. and Kelley, C.T., "Newton's Method at Singular Points, I and II," SIAM J. Num. Anal., 17, pp. 66–70; 465–471, 1980.

[43] Hansen, E., "Global Optimization Using Interval Analysis– The Multi Dimensional Case," Numer. Math., 34, pp. 247–270, 1980.

[44] Kao, J.-J., Brill, E. D., Jr., and Pfeffer, J. T., "Generation of Alternative Optima for Nonlinear Programming Problems," Eng. Opt., 15, pp. 233–251, 1990.

[45] Ge, R., "Finding More and More Solutions of a System of Nonlinear Equations," Appl. Math. Computation, 36, pp. 15-30, 1990.

[46] Laarhoven, P. J. M. van., and Aarts, E., Simulated Annealing: Theory and Applications, D. Reidel Publishing, Dordrecht, The Netherlands, 1987.

[47] Goldberg, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Co. Inc., Reading, Massachusetts, 1989.

[48] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E., "Equation of State Calculations by Fast Computing Machines," J. Chem. Physics, 21 (6), pp. 1087–1092, 1953.

[49] Kirkpatrick, S., Gelatt, C. D., Jr., and Vecchi, M. P., "Optimization by Simulated Annealing," Science, 220 (4598), pp. 671–680, 1983.

[50] Cerny, V., "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm," J. Opt. Theory Appl., 45, pp. 41–52, 1985.

[51] Rutenbar, R. A., "Simulated Annealing Algorithms: An Overview," IEEE Circuits and Devices, January, pp. 19–26, 1989.

[52] Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C., "Optimization by Simulated Annealing: An Experimental Evaluation. Part I. Graph Partitioning," Operations Research, 37, 1990, pp. 865–893.

[53] Aarts, E., and Korst, J., Simulated Annealing and Boltzmann Machines, A Stochastic Approach to Combinatorial Optimization and Neural Computing, John Wiley & Sons, 1989.

[54] Nahar, S., Sahni, S., and Shragowithz, E. V., in the Proceedings of 22nd Design Automation Conf., Las Vegas, June 1985, pp. 748-752.

[55] Elperin, T, "Monte Carlo Structural Optimization in Discrete Variables with Annealing ALgorithm," Int. J. Num. Meth. Eng., 26, 1988, pp. 815–821.

[56] Kincaid, R. K., and Padula, S. L., "Minimizing Distortion and Internal Forces in Truss Structures by Simulated Annealing," Proceedings of the AIAA/ASME /ASCE/AHS/ASC 31st Structures, Structural Dynamics, and Materials Conference, Long Beach, CA., 1990, Part 1, pp. 327–333.

[57] Balling, R. J., and May, S. A., "Large-Scale Discrete Structural Optimization: Simulated Annealing, Branch-and-Bound, and Other Techniques," presented at the AIAA/ASME/ASCE/AHS/ASC 32nd Structures, Structural Dynamics, and Materials Conference, Long Beach, CA., 1990,

[58] Chen, G.-S., Bruno, R. J., and Salama, M.,"Optimal Placement of Active/Passive Members in Structures Using Simulated Annealing," AIAA J., 29 (8), August 1991, pp. 1327–1334.

[59] Holland, J. H., Adaptation of Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, MI, 1975.

[60] De Jong, K. A., Analysis of the Behavior of a Class of Genetic Adaptive Systems (Doctoral Dissertation, The University of Michigan; University Microfilms No. 76-9381), Dissertation Abstracts International, 36 (10), 5140B, 1975.

[61] Booker, L., "Improving Search in Genetic Algorithms," in Genetic Algorithms and Simulated Annealing, Ed. L. Davis, Morgan Kaufmann Publishers, Inc., Los Altos, CA. 1987, pp. 61–73.

[62] Goldberg, D. E., and Samtani, M. P., "Engineering Optimization via Genetic Algorithm," Proceedings of the Ninth Conference on Electronic Computation, ASCE, February 1986, pp. 471–482.

[63] Hajela, P., "Genetic Search—An Approach to the Nonconvex Optimization Problem," AIAA J., 28 (7), July 1990, pp. 1205–1210.

[64] Rao, S. S., Pan, T.-S., and Venkayya, V. B., "Optimal Placement of Actuators in Actively Controlled Structures Using Genetic Algorithms," AIAA J., 29 (6), pp. 942–943, June 1991.

[65] Szu, H., and Hartley, R.L., "Nonconvex Optimization by Fast Simulated Annealing," Proceedings of the IEEE, 75 (11), pp. 1538–1540, 1987.