Statistics 580

Optimization Methods

Introduction

Let $f(\mathbf{x})$ be a given real-valued function on $\Re^p$. The general optimization problem is to find an $\mathbf{x} \epsilon \Re^p$ at which $f(\mathbf{x})$ attain a maximum or a minimum. It is of great interest to statisticians because both maximum likelihood estimation and estimation by least-squares method are special optimization problems. The function f is called the *objective function*. Since a maximum of $f(\mathbf{x})$ is a minimum of $-f(\mathbf{x})$, discussion can be restricted to *function minimization* only . In the following discussion it will be assumed that f is twice differentiable and has continuous derivatives with respect to each of the x's over its range.

If the function f is differentiable at its minimum value with continuous derivatives, then its *gradient vector*

$$\mathbf{f}'(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_p} \right)^T$$

(sometimes denoted by $\nabla \mathbf{f}(\mathbf{x})$ in the optimization literature) has a zero at this minimum value. Hence, solving the minimization problem is equivalent to solving the nonlinear system of equations $\mathbf{f}'(\mathbf{x}) = \mathbf{0}$. Therefore, methods available for the solution of nonlinear systems of equations become a special class of optimization methods.

However, those methods may not solve the minimization problem for several reasons. One of these is that, although at a *stationery point* of f the gradient function $\mathbf{f}'(\mathbf{x})$ vanishes, a necessary condition for the solution to $\mathbf{f}'(\mathbf{x}) = \mathbf{0}$ to be a minimum is that the *Hessian matrix* $H = \mathbf{f}''(\mathbf{x})$ (sometimes denoted by $\nabla^2 \mathbf{f}(\mathbf{x})$), where

$$\mathbf{f}''(\mathbf{x}) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right)_{p \times p}$$

be positive definite when evaluated at that solution.

`Example 1:` Consider minimization of the function

$$f(\mathbf{x}) = 3x_1^2 + 3x_2^2 + 6x_3^2 - 2x_1x_2 - 4x_1x_3 - 3x_1 - 4x_2$$

whose first derivatives are:

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= 6x_1 - 2x_2 - 4x_3 - 3 \\ \frac{\partial f}{\partial x_2} &= 6x_2 - 2x_1 - 4 \\ \frac{\partial f}{\partial x_3} &= 12x_3 - 4x_1 \end{aligned}$$

Setting

$$\mathbf{f}'(\mathbf{x}) = \begin{bmatrix} 6x_1 - 2x_2 - 4x_3 - 3 \\ -2x_1 + 6x_2 - 4 \\ -4x_1 + 12x_3 \end{bmatrix} = 0$$

and solving, we get $x_1 = 13/12,\ x_2 = 37/36,\ x_3 = 13/36$. Thus $\mathbf{x}^* = (13/12,\ 37/36,\ 13/36)^T$ is a stationery point of f. Evaluating the Hessian matrix at $\mathbf{x}^*$

$$\frac{\partial^2 f}{\partial x_1^2} = 6 \qquad \frac{\partial^2 f}{\partial x_2^2} = 6$$

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = -2 = \frac{\partial^2 f}{\partial x_2 \partial x_1} \qquad \frac{\partial^2 f}{\partial x_2 \partial x_3} = 0 = \frac{\partial^2 f}{\partial x_3 \partial x_2}$$

$$\frac{\partial^2 f}{\partial x_1 \partial x_3} = -4 = \frac{\partial^2 f}{\partial x_3 \partial x_1} \qquad \frac{\partial^2 f}{\partial x_3^2} = 12$$

we get:

$$\mathbf{H} = \mathbf{f}''(\mathbf{x}) = \begin{bmatrix} 6 & -2 & -4 \\ -2 & 6 & 0 \\ -4 & 0 & 12 \end{bmatrix}$$

$\mathbf{H}$ is positive definite showing that $\mathbf{x}^*$ is a minimum of f. □

The minimization problem thus became a problem of solving a set of simultaneous equations. In general, $\mathbf{f}'(\mathbf{x})$ is a nonlinear system of equations, transforming the minimization problem into a root finding problem. The secant method and Newton-Raphson, two classical algorithms for root finding, and other methods were referenced in an earlier section. Algorithms for root finding thus fall into the class of *search methods* for the minimization problem discussed below.

Note that, the solution to $\mathbf{f}'(\mathbf{x}) = \mathbf{0}$, even if $\mathbf{f}''(\mathbf{x})$ is positive definite at that value, may represent only a *local minimum* and not a *global minimum*. Note also that these methods require both the gradient vector $\mathbf{f}'(\mathbf{x})$ and the Hessian matrix $\mathbf{f}''(\mathbf{x})$ to be computed, and when neither is available analytically, one must replace these quantities with appropriate discrete approximations. For the purpose of this discussion, let $\mathbf{g}(\mathbf{x})$ denote the gradient vector $\mathbf{f}'(\mathbf{x})$ and $\mathbf{H}(\mathbf{x})$ denote the Hessian matrix $\mathbf{f}''(\mathbf{x})$, both evaluated at $\mathbf{x}$.

Descent Methods

The optimization algorithms considered here are all iterative and of the form

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} + \alpha_{(i)}\mathbf{d}_{(i)} \tag{1}$$

and are designed to improve on an initial guess $\mathbf{x}_{(0)}$ and converge to a solution. Computations at the i^{th} iteration involve determining the vector $\mathbf{d}_{(i)}$, called the *step direction* or the *direction vector*, and $\alpha_{(i)}\|\mathbf{d}_{(i)}\|$, called the *step size*. Directions $\mathbf{d}_{(i)}$ are called *admissible* if $f(\mathbf{x}_{(i+1)}) \leq f(\mathbf{x}_{(i)})$ as $\alpha_{(i)} \to 0$ i.e., the new iterate $\mathbf{x}_{(i)} + \alpha_{(i)}\mathbf{d}_{(i)}$decreases the objective function in magnitude at $\alpha = 0$. Those algorithms that employ only admissible directions in each iteration are called *descent* methods. Iterative algorithms for optimization thus involve two steps:

1. to determine a search direction $\mathbf{d}_{(i)}$, usually a descent direction, and

2. to search along the line to determine a step size, $\alpha_{(i)}$ that minimizes f along that line.

In the next section, strategies for determining step size $\alpha_{(i)}$ in Equation 1, defining part of an iterative procedure for finding the minimum of f are discussed.

Linear Search Methods

The basic problem is to find α which minimizes $\rho(\alpha) = f(\mathbf{x}_{(i)} + \alpha\, \mathbf{d}_{(i)})$ for a fixed direction $\mathbf{d}_{(i)}$. There are two basic types of methods for linear search:

- one depends on reducing the size of the interval that contains the minimum, and
- the other attempts to approximate $\rho(\alpha)$ by a polynomial (say, a quadratic) whose minimum is easily computed using a well-known formula.

To use any of these methods, initial interval $[a, b]$ must be found such that $\rho(\cdot)$ is unimodal in that interval.

The *Golden-section search* is a bracketing algorithm that begins with an interval $[a, b]$ where ρ is unimodal and reduces the length of the interval by a fixed factor in each iteration. Algorithms for the Golden-section search are described in Kennedy and Gentle (1980) and Thisted (1988). Suppose that the interval of uncertainty is $[a, b]$ and ρ is unimodal in $[a, b]$. Evaluate $\rho(\alpha)$ at $\alpha_1 \,\epsilon\, (0, 1)$ where

$$\frac{1}{\alpha_1} = \frac{\alpha_1}{1 - \alpha_1}.$$

This gives $\alpha_1 = (\sqrt{5} - 1)/2 = .618033989$, that is, evaluate ρ at $\alpha_1 = a + .618034(b - a)$ and at $\alpha_2 = a + b - \alpha_1$, symmetrically on the other side of the interval $[a, b]$, respectively. The interval of uncertainty is reduced to a length $L = .618$ depending on which value of ρ is smaller. In general after n evaluations, $L = (.618)^{n-1}$. For example, after 5 evaluations L reduces to $.146(b - a)$. Fibonacci search is another similar search method.

Approximating $\rho(\alpha)$ by a polynomial in α, for example, by quadratic interpolation, is another approach for minimizing $\rho(\alpha)$. Let $\rho(\alpha) \approx A\alpha^2 + B\alpha + C$. By evaluating $\rho(\alpha)$ at pre-specified values α_1, α_2 and α_3, A, B, and C can be determined exactly by solving a system of equations. Since the optimum of $\rho(\alpha)$ occurs at $\alpha = -B/2A$, a closed form formula for the minimum point is:

$$\alpha = \frac{(\alpha_3^2 - \alpha_2^2)\rho(\alpha_1) + (\alpha_2^2 - \alpha_1^2)\rho(\alpha_3) + (\alpha_1^2 - \alpha_3^2)\rho(\alpha_2)}{2[(\alpha_3 - \alpha_2)\rho(\alpha_1) + (\alpha_2 - \alpha_1)\rho(\alpha_3) + (\alpha_1 - \alpha_3)\rho(\alpha_2)]}$$

This is then improved by reapplying the quadratic interpolation using this value as one of the starting values. Kennedy and Gentle (1980) also describes an algorithm due to Davidon (1959) where a cubic polynomial is fitted to $\rho(\cdot)$ using 3 points. Thisted (1988) describes a method called *successive approximation* due to Berman (1966).

Selecting Descent Directions

Several methods for function minimization via iterative methods are described here. Some methods are just variations of a basic method. No single method will be able to solve every minimization problem that arise in practice. But a good understanding of available methods is necessary for the user to be able to either select an appropriate method that is suitable for a particular objective function or to fine tune an algorithm to take advantage of special features of the problem at hand.

Method of Steepest Descent

In this method the choice for $\mathbf{d}$ is taken as the negative of the gradient vector, $-\mathbf{f}'(\mathbf{x})$ since, obviously, at any given point $\mathbf{x}_{(i)}$ where $\mathbf{f}'(\mathbf{x})$ is nonzero, the gradient vector defines the direction in which f is decreasing most rapidly. Thus the iteration becomes

$$\mathbf{x}_{(i+1)}) = \mathbf{x}_{(i)} - \alpha_{(i)}\,\mathbf{g}_{(i)}.$$

where $\mathbf{g}_{(i)} \equiv \mathbf{g}(\mathbf{x}_{(i)}) = \mathbf{f}'(\mathbf{x}_{(i)})$. In actual implementation of the algorithm, $\mathbf{g}$ is normalized at each step, so that $\mathbf{g}_{(i)}$ is replaced in above by $\mathbf{g}^*_{(i)}$ where $\mathbf{g}^*_{(i)} = \mathbf{g}_{(i)}/(\mathbf{g}^T_{(i)}\mathbf{g}_{(i)})^{1/2}$. This algorithm is seldom used today because it is too slow to converge; however, it may serve a purpose as a start-up routine for other algorithms, by providing good starting values after a few iterations.

`Example 2:` Minimize the function:

$$f(\mathbf{x}) = 100(x_1 - 15)^2 + 20(28 - x_1)^2 + 100(x_2 - x_1)^2 + 20(38 - x_1 - x_2)^2$$

The first derivative is:

$$f'(\mathbf{x}) = \begin{pmatrix} 200(x_1 - 15) - 40(28 - x_1) - 200(x_2 - x_1) - 40(38 - x_1 - x_2) \\ 200(x_2 - x_1) - 40(38 - x_1 - x_2) \end{pmatrix}$$

Using starting values

$$\mathbf{x}_{(0)} = \begin{pmatrix} 10 \\ 14 \end{pmatrix}$$

we compute

$$\mathbf{g}_{(0)} = f'(\mathbf{x})\big|_{(10,14)} = \begin{pmatrix} -3080 \\ +240 \end{pmatrix}$$

and hence

$$\mathbf{d} = -\mathbf{g}_{(0)} = \begin{pmatrix} 3080 \\ -240 \end{pmatrix}$$

Compute $f(\mathbf{x}_{(0)} - \alpha\mathbf{g}_{(0)})$ by substituting

$$\begin{aligned} x_1^{(0)} &= 10 + 3080\alpha \\ x_2^{(0)} &= 14 - 240\alpha \end{aligned}$$

in $f(\mathbf{x})$ above. Perform a one-dimensional search on $f(\mathbf{x}_{(0)} - \alpha \mathbf{g}_{(0)})$ to find α which minimizes it. For this example we use calculus i.e., by setting $\partial f / \partial \alpha = 0$ we get $\alpha_{(0)} = .00199$, giving

$$\mathbf{x}_{(1)} = \begin{pmatrix} 10 \\ 14 \end{pmatrix} + .00199 \begin{pmatrix} 3080 \\ -240 \end{pmatrix} = \begin{pmatrix} 16.12 \\ 13.52 \end{pmatrix}$$

Now compute $\mathbf{g}_{(1)}$ and then $\alpha_{(1)}$ and so on. □

Convergence of the Method of Steepest Descent of Example 2						
Iteration	x_1	x_2	$-g_1$	$-g_2$	α_i	f
1	10.00	14.00	3080	-240.0	0.00199	14,500
2	16.12	13.52	66.51	853.5	0.00462	5019
3	16.43	17.46	549.4	42.82	0.00199	3328
4	17.52	17.38	11.87	152.3	0.00462	3026
5	17.57	18.08	98.02	-7.64	0.00199	2972
6	17.77	18.06	2.12	27.16	0.00462	2963
10	17.82	18.21	0.071	0.861	0.00464	2960.71
14	17.821	18.214	0.00488	0.02808	0.00535	2960.71
15	17.821	18.214	0.01953	0.00244	0.00191	2960.71
16	17.821	18.214	0.00244	0.00610	0.00644	2960.71

Newton-Raphson Algorithm

The most natural direction to take seems to be that suggested by the Newton-Raphson iteration introduced for solving systems of nonlinear equations. Recall that we used Newton-Raphson to solve $f(x) = 0$ using the iteration $x_{(i+1)} = x_{(i)} - f(x_{(i)})/f'(x_{(i)})$. We first extend the N-R iterative formula to solve $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, where $\mathbf{f}$ is a vector valued function, $\mathbf{f} : \Re^n \longrightarrow \Re^n$ i.e., suppose $\mathbf{f}(\mathbf{x})$ is a system of n nonlinear simultaneous equations in n unknowns, and continuously differentiable at $\mathbf{x} \, \epsilon \, \Re^p$

$$\underset{n \times 1}{\mathbf{f}(\mathbf{x})} = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{pmatrix}$$

$$\underset{n \times n}{\mathbf{f}'(\mathbf{x})} = \begin{pmatrix} \mathbf{f}_1'(\mathbf{x})^T \\ \mathbf{f}_2'(\mathbf{x})^T \\ \vdots \\ \mathbf{f}_n'(\mathbf{x})^T \end{pmatrix} = \left(\frac{\partial f_i}{\partial x_j} \right)_{n \times n}$$

where $\mathbf{f}'(\mathbf{x})$ is an $n \times n$ matrix called the Jacobian of $\mathbf{f}$ evaluated at some $\mathbf{x}$ and is usually denoted by $\mathbf{J}(\mathbf{x})$. Thus the Newton-Raphson iteration for solving $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ is given by:

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} - \left[\mathbf{J}(\mathbf{x}_{(i)}) \right]^{-1} \mathbf{f}(\mathbf{x}_{(i)})$$

where $\mathbf{J}$ is the $n \times n$ Jacobian matrix of $\mathbf{f}$ given by $\mathbf{f}'(\mathbf{x})$.
`Example 3:` Let

$$\mathbf{f}: \ R^2 \longrightarrow R^2$$
$$f_1(\mathbf{x}) = e^{x_1} - x_2$$
$$f_2(\mathbf{x}) = x_1^2 - 2x_2$$

Then

$$\mathbf{f}'(\mathbf{x}) \ = \begin{pmatrix} e^{x_1} & -1 \\ 2x_1 & -2 \end{pmatrix} = \mathbf{J}.$$

The N-R iteration for solving the system of nonlinear equations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ as given by

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} - \left[\mathbf{J}(\mathbf{x}_{(i)})\right]^{-1} \mathbf{f}(\mathbf{x}_{(i)})$$

where $\mathbf{J}$ is the $n \times n$ Jacobian matrix of $\mathbf{f}$, is applied to solve the system $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ in the optimization problem, where $\mathbf{g}$ is the gradient vector. The required iteration then becomes

$$\begin{aligned} \mathbf{x}_{(i+1)} &= \mathbf{x}_{(i)} - \left[\mathbf{f}''(\mathbf{x}_{(i)})\right]^{-1} \mathbf{g}(\mathbf{x}_{(i)}) \\ &= \mathbf{x}_{(i)} - \left[\mathbf{H}(\mathbf{x}_i)\right]^{-1} \mathbf{g}(\mathbf{x}_{(i)}), \\ &= \mathbf{x}_{(i)} - \mathbf{H}_{(i)}^{-1} \mathbf{g}_{(i)}. \end{aligned}$$

Note that in this case, the Jacobian matrix $\mathbf{J}$ is replaced by the Hessian $\mathbf{H}$ and the vector-valued function $\mathbf{f}(\mathbf{x})$ is replaced by the gradient vector $\mathbf{g}$.

`Example 4:` Minimize the function:

$$f(\mathbf{x}) = 100(x_1 - 15)^2 + 20(28 - x_1)^2 + 100(x_2 - x_1)^2 + 20(38 - x_1 - x_2)^2$$

using the Newton-Raphson algorithm. Start with:

$$\mathbf{x}_{(0)} = \begin{pmatrix} 10 \\ 14 \end{pmatrix}$$

The gradient vector is:

$$\mathbf{f}'(\mathbf{x}) = \begin{pmatrix} 200(x_1 - 15) - 40(28 - x_1) - 200(x_2 - x_1) - 40(38 - x_1 - x_2) \\ 200(x_2 - x_1) - 40(38 - x_1 - x_2) \end{pmatrix}$$

The Hessian is:

$$\mathbf{f}''(\mathbf{x}) = \begin{pmatrix} 200 + 40 + 200 + 40 & -200 + 40 \\ -200 + 40 & 200 + 40 \end{pmatrix} = \begin{pmatrix} 480 & -160 \\ -160 & 240 \end{pmatrix} = 80 \begin{pmatrix} 6 & -2 \\ -2 & 3 \end{pmatrix}.$$

The inverse of the Hessian and the gradient vector evaluated at $\mathbf{x}_{(0)}$ are:

$$\mathbf{H}^{-1} = \frac{1}{80 \times 14} \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix} \quad \mathbf{g}_{(0)} = \begin{pmatrix} -3080 \\ +240 \end{pmatrix}$$

Thus:

$$\mathbf{H}^{-1}\mathbf{g}_{(0)} = \frac{1}{80 \times 14}\begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}\begin{pmatrix} -3080 \\ +240 \end{pmatrix} = \begin{pmatrix} -7.821 \\ -4.214 \end{pmatrix}$$

and the next iterate is, therefore:

$$\mathbf{x}_{(1)} = \begin{bmatrix} 10 \\ 14 \end{bmatrix} - \begin{bmatrix} -7.821 \\ -4.214 \end{bmatrix} = \begin{bmatrix} 17.821 \\ 18.214 \end{bmatrix}.\square$$

Again examining the N-R iteration

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} - \mathbf{H}_{(i)}^{-1}\mathbf{g}_{(i)} \ ,$$

note that the step-size for this iterative method is fixed at 1 and thus the direction $\mathbf{H}_{(i)}^{-1}\mathbf{g}_{(i)}$ becomes a descent direction when $\mathbf{H}_{(i)}$ is positive definite. However, a particular Newton direction may not be admissible because $\mathbf{H}_{(i)}$ is not positive definite for an $\mathbf{x}_{(i)}$ that is not near the minimum. A related problem is that $\mathbf{H}_{(i)}$ may become nearly singular, causing the search to move away from the current iterate because the size of the direction will be quite large. Thus, Newton steps are not a good choice at the beginning of an optimization iteration, even though they are optimal near the solution. These practical problems led to some important modifications of the original method.

Levenberg-Marquardt Adjustment

It is clear that the computation of the Newton step discussed above reduces to solving the linear system of equations

$$\mathbf{H}_{(i)}\mathbf{d}_{(i)} = \mathbf{g}_{(i)}$$

to determine the step direction $\mathbf{d}_{(i)}$. When $\mathbf{H}_{(i)}$ is not positive definite, it could be substituted by a modified Hessian that is an approximation that is positive definite. One approach is to replace the negative eigenvalues of $\mathbf{H}_{(i)}$ by their absolute values and zero eigenvalues $\mathbf{H}_{(i)}$ by small positive values. Thus the modified algorithm is

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} - \alpha_{(i)}\overline{\mathbf{H}}_{(i)}^{-1}\mathbf{g}_{(i)}$$

where $\alpha_{(i)}$ is a scalar introduced as the step length. Although convergence may be achieved, this method could be computationally quite expensive. A possibly less expensive alternative suggested independently by Levenberg (1944) and Marquardt (1963), substitutes

$$\mathbf{H}_{(i)}^{*} = \mathbf{H}_{(i)} + \tau_{(i)}\mathbf{I}_p$$

in place of $\mathbf{H}_{(i)}$ where $\tau_{(i)} \geq 0$ is chosen at each iteration such that the adjusted matrix $\mathbf{H}_{(i)}^{*}$ is positive definite. Note that this will define a descent direction whenever $\tau_{(i)} > \lambda_{\min}$ where $\lambda_{\min}$ is the smallest eigenvalue of $\mathbf{H}_{(i)}$. Thus the choice of τ depends on the current value of $\mathbf{H}(\mathbf{x})$ and therefore has to be recomputed at each step. When τ is small, the modified direction remains close to the Newton direction. As iteration proceeds, the sequence of $\tau_{(i)}$'s should become progressively smaller in order to achieve quadratic convergence. Dennis and

Schnabel (1983) discuss methods used in practice to achieve this as well give corresponding algorithms.

Other Methods for Adjusting the Step Direction
A method due to Gill and Murray (1974) consider the use of a Cholesky-type decomposition to solve the equations

$$\mathbf{H}\,\mathbf{d} = \mathbf{g}$$

for determining step direction. This is described in detail in Kennedy an Gentle (1980).

Quasi-Newton Methods

Davidon (1959) first suggested the use of approximations to the Hessian matrix to overcome difficulties associated with the exact computation of the Hessian for many objective functions. Quasi-Newton methods are characterized by the computation of an updated matrix as a by-product of each iteration, to be used as an approximation to the Hessian matrix (or its inverse) in the following step.

The quasi-Newton methods (also called *variable metric* methods) discussed here also apply generally to the problem of solving nonlinear systems of equations where Newton iterations are used. Also note that the two most popular quasi-Newton algorithms were proposed for solving optimization problems associated with scalar objective functions (such as those that occur in maximum likelihood (ML) and nonlinear least squares estimation (LS) problems). Consider the standard Newton-Raphson iteration

$$\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} - \alpha_{(i)}\mathbf{H}_{(i)}^{-1}\mathbf{g}_{(i)}.$$

To reduce the computational effort in obtaining $\mathbf{H}_{(i+1)}^{-1}$, it can be shown that under quite general conditions $\mathbf{H}_{(i+1)}^{-1}$ can be obtained by *updating* it with a low-rank matrix. i.e.

$$\mathbf{H}_{(i+1)}^{-1} = \mathbf{H}_{(i)}^{-1} + \mathbf{M}_{(i)}$$

where $\mathbf{M}_{(i)}$ is typically a matrix of rank one or two.

The justification of this is that by using the first order approximation

$$\mathbf{g}_{(i+1)} \approx \mathbf{g}_{(i)} + \mathbf{H}_{(i)}(\mathbf{x}_{(i+1)} - \mathbf{x}_{(i)})$$

it should be possible to obtain *curvature* information at the point $\mathbf{x}_{(i)}$ using

$$(\mathbf{x}_{(i+1)} - \mathbf{x}_{(i)})^T\mathbf{H}_{(i)}(\mathbf{x}_{(i+1)} - \mathbf{x}_{(i)}) \approx (\mathbf{x}_{(i+1)} - \mathbf{x}_{(i)})^T(\mathbf{g}_{(i+1)} - \mathbf{g}_{(i)})$$

and thus use this information to update $\mathbf{H}_{(i)}$ directly (or its inverse using the Sherman-Morrison formula). It is easy to observe that the relationship (called the *secant condition* or the quasi-Newton condition)

$$\mathbf{g}_{(i+1)} - \mathbf{g}_{(i)} = \mathbf{B}_{(i)}(\mathbf{x}_{(i+1)} - \mathbf{x}_{(i)})$$

must be satisfied by a matrix $\mathbf{B}_{(i)}$ that is an approximation to the Hessian matrix. Recall that in the univariate case for solving nonlinear equations, the Newton-Raphson was generalized

to the secant method by solving for B (one-dimensional). For the optimization case this would reduce to

$$f''(x_{(i+1)}) \approx (f'(x_{(i+1)}) - f'(x_{(i)}))/(x_{(i+1)} - x_{(i)})$$

However, in the multidimensional case, $\mathbf{B}$ cannot be uniquely determined without imposing some conditions. For example, an algorithm must find a solution to the above equation that is closest to the current estimate i.e., $\min_{\mathbf{B}} \|\mathbf{B} - \mathbf{B}_{(i)}\|$ and satisfies the *curvature condition* i.e., $(\mathbf{x}_{(i+1)} - \mathbf{x}_{(i)})^T \mathbf{B} (\mathbf{x}_{(i+1)} - \mathbf{x}_{(i)}) > 0$, and also must maintain the symmetry and positive definiteness of $\mathbf{B}$. These algorithms also include a line search in each step i.e., must estimate step length parameter $\alpha_{(i)}$ in each step. If the line search is not an exact minimizer (because of the need for efficient computation), there are certain conditions (called Wolfe conditions) that an inexact estimate of $\alpha_{(i)}$ must satisfy, so that the resulting matrix $\mathbf{B}$ remains positive definite. Some step length algorithms such as that based on cubic interpolation satisfies these conditions.

Rank one updates (e.g., **SR1** or *symmetric rank one method*) maintains the symmetry but does not guarantee a positive definite update and therefore will not be considered in this discussion. Of the quasi-Newton methods based on rank 2 matrix updating, two are of particular interest in statistical computation:the Davidon-Fletcher-Powell (DFP) method and the BFGS algorithm. The BFGS is widely used, more popular and included in software systems and libraries because it is less affected by the innacuracy of the line search estimate. Both these methods ensure that $\mathbf{H}_{(i)}$ remains positive definite if $\mathbf{H}_{(0)}$ is positive-definite, which is true in the case of ML and LS. Under some mild conditions, the convergence rate is super-linear near the minimum. The Broyden class of algorithms is defined by a linear combination of the DFP and the BFGS updates. Both DFP and the BFGS algorithm can be expressed in terms of $\mathbf{H}_{(i)}$ or $\mathbf{H}_{(i)}^{-1}$; however the updates are expressed below in the most convenient way first.

D-F-P Algorithm

The Davidon-Fletcher-Powell method (Davidon (1959); Fletcher and Powell (1963)) is fairly well-known in statistical computing, and is widely used. Let

$$\mathbf{d}_{(i)} = \mathbf{x}_{(i+1)} - \mathbf{x}_{(i)} = -\alpha_{(i)} \mathbf{H}_{(i)}^{-1} \mathbf{g}_{(i)}$$

and

$$\mathbf{e}_{(i)} = \mathbf{g}_{(i+1)} - \mathbf{g}_{(i)} \, .$$

Then the algorithm uses the update $\mathbf{H}_{(i+1)}^{-1}$ where

$$\mathbf{H}_{(i+1)}^{-1} = \mathbf{H}_{(i)}^{-1} + \frac{\mathbf{d}_{(i)} \mathbf{d}_{(i)}^T}{\mathbf{d}_{(i)}^T \mathbf{e}_{(i)}} - \frac{\mathbf{H}_{(i)}^{-1} \mathbf{e}_{(i)} \mathbf{e}_{(i)}^T \mathbf{H}_{(i)}^{-1}}{\mathbf{e}_{(i)}^T \mathbf{H}_{(i)}^{-1} \mathbf{e}_{(i)}}$$

The complete computational algorithm of using the update for the optimization problem is given in Kennedy and Gentle (1980) p.456. It also includes a discussion of numerical problems associated with this algorithm and possible solutions.

Starting with $\mathbf{x}_{(0)}$, $\mathbf{H}_{(0)} = \mathbf{I}$, and $i = 0$

1. Compute $\mathbf{d}_{(i)} = -\mathbf{H}_{(i)}^{-1}\mathbf{g}_{(i)}$.
2. Compute $\alpha_{(i)}$ to minimize $\rho(\alpha) = f(\mathbf{x}_{(i)} + \alpha\mathbf{d}_{(i)})$.
3. Set

$$\begin{aligned} \mathbf{x}_{(i+1)} &= \mathbf{x}_{(i)} + \alpha_{(i)}\mathbf{d}_{(i)} \\ \mathbf{e}_{(i)} &= \mathbf{g}_{(i+1)} - \mathbf{g}_{(i)} \end{aligned}$$

4. Check $\mathbf{x}_{(i+1)}$, $f(\mathbf{x}_{(i+1)})$ and $\mathbf{g}_{(i+1)}$ for termination.
5. Compute

$$\mathbf{H}_{(i+1)}^{-1} = \mathbf{H}_{(i)}^{-1} + \frac{\mathbf{d}_{(i)}\mathbf{d}_{(i)}^T}{\mathbf{d}_{(i)}^T\mathbf{e}_{(i)}} - \frac{\mathbf{H}_{(i)}^{-1}\mathbf{e}_{(i)}\mathbf{e}_{(i)}^T\mathbf{H}_{(i)}^{-1}}{\mathbf{e}_{(i)}^T\mathbf{H}_{(i)}^{-1}\mathbf{e}_{(i)}}$$

The algorithm for the update $\mathbf{H}_{(i+1)}$ is more complicated:

$$\mathbf{H}_{(i+1)} = \mathbf{H}_{(i)} - \frac{\mathbf{H}_{(i)}\mathbf{d}_{(i)}\mathbf{d}_{(i)}^T\mathbf{H}_{(i)}}{\mathbf{d}_{(i)}^T\mathbf{H}_{(i)}\mathbf{d}_{(i)}} + \frac{\mathbf{e}_{(i)}\mathbf{e}_{(i)}^T}{\mathbf{e}_{(i)}^T\mathbf{d}_{(i)}} + (\mathbf{d}_{(i)}^T\mathbf{H}_{(i)}\mathbf{d}_{(i)})\mathbf{w}_{(i)}\mathbf{w}_{(i)}^T$$

where

$$\mathbf{w}_{(i)} = \frac{\mathbf{e}_{(i)}}{\mathbf{e}_{(i)}^T\mathbf{d}_{(i)}} - \frac{\mathbf{H}_{(i)}\mathbf{d}_{(i)}}{\mathbf{d}_{(i)}^T\mathbf{H}_{(i)}\mathbf{d}_{(i)}}$$

BFGS Algorithm

The second rank two update is known as the Broyden, Fletcher, Goldfarb, Shanno (BFGS) algorithm, proposed independently by Broyden (1970), Fletcher (1970), Goldfarb (1970) and Shanno (1970), and is generally considered superior to the Devidon-Fletcher-Powell algorithm. It is given in terms of $\mathbf{H}_{(i)}$, below:

$$\mathbf{H}_{(i+1)} = \mathbf{H}_{(i)} - \frac{\mathbf{H}_{(i)}\mathbf{d}_{(i)}\mathbf{d}_{(i)}^T\mathbf{H}_{(i)}}{\mathbf{d}_{(i)}^T\mathbf{H}_{(i)}\mathbf{d}_{(i)}} + \frac{\mathbf{e}_{(i)}\mathbf{e}_{(i)}^T}{\mathbf{d}_{(i)}^T\mathbf{e}_{(i)}}$$

Using the Sherman-Morrison formula, the BFGS update of the inverse is given as:

$$\mathbf{H}_{(i+1)}^{-1} = \mathbf{H}_{(i)}^{-1} + \left[1 + \frac{\mathbf{e}_{(i)}^T\mathbf{H}_{(i)}^{-1}\mathbf{e}_{(i)}}{\mathbf{d}_{(i)}^T\mathbf{e}_{(i)}}\right]\frac{\mathbf{d}_{(i)}\mathbf{d}_{(i)}^T}{\mathbf{d}_{(i)}^T\mathbf{e}_{(i)}} - \frac{\mathbf{d}_{(i)}\mathbf{e}_{(i)}^T\mathbf{H}_{(i)}^{-1} + \mathbf{H}_{(i)}^{-1}\mathbf{e}_{(i)}\mathbf{d}_{(i)}^T}{\mathbf{d}_{(i)}^T\mathbf{e}_{(i)}}$$

`Example 5:` Recall that

$$\begin{aligned} f(\mathbf{x}) &= 100(x_1 - 15)^2 + 20(28 - x_1)^2 + 100(x_2 - x_1)^2 + 20(38 - x_1 - x_2)^2 \\ \mathbf{g}(\mathbf{x}) &= \mathbf{f}'(\mathbf{x}) = \begin{pmatrix} 200(x_1 - 15) - 40(28 - x_1) - 200(x_2 - x_1) - 40(38 - x_1 - x_2) \\ 200(x_2 - x_1) - 40(38 - x_1 - x_2) \end{pmatrix} \end{aligned}$$

Let

$$\mathbf{x}_{(0)} = \begin{pmatrix} 10 \\ 14 \end{pmatrix}$$

as before and

$$\mathbf{g}_{(0)} = \mathbf{g}(\mathbf{x}_0) = \begin{pmatrix} -3080 \\ 240 \end{pmatrix}$$

Start with $\mathbf{H}_{(0)}^{-1} = \mathbf{I}_2$. Compute first direction $\mathbf{d}_{(0)} = -\mathbf{H}_{(0)}^{-1} = \mathbf{g}_{(0)} = \begin{pmatrix} 3080 \\ -240 \end{pmatrix}$ or in normalized form $\mathbf{d}_{(0)} = \begin{pmatrix} .997 \\ -0.078 \end{pmatrix}$. At this stage it is needed to minimize

$$f(\mathbf{x}_{(0)} + \alpha \mathbf{d}_{(0)}) = 100(10 + .997\alpha - 15)^2 + 20(28 - 10 - .997\alpha)^2 + \cdots$$

with respect to α. This would normally involve a cubic polynomial search. Here, for obtaining a minimum to continue with the iteration for illustrating the procedure, we will set $\frac{\partial f}{\partial \alpha} = 0$ and solve the resulting equation, giving $\alpha_{(0)} = 6.136$. Thus the next iterate is:

$$\begin{aligned} \mathbf{x}_{(1)} &= \mathbf{x}_{(0)} + \alpha_{(0)} \mathbf{d}_{(0)} \\ &= \begin{pmatrix} 10 \\ 14 \end{pmatrix} + 6.136 \begin{pmatrix} .997 \\ .078 \end{pmatrix} = \begin{pmatrix} 16.12 \\ 13.52 \end{pmatrix} \end{aligned}$$

To continue, compute $\mathbf{g}_{(1)}$ and $\mathbf{e}_{(0)}$:

$$\begin{aligned} \mathbf{g}_{(1)} &= \mathbf{g}(\mathbf{x}_{(1)}) = \begin{pmatrix} -65.6 \\ -854.4 \end{pmatrix} \\ \mathbf{e}_{(0)} &= \mathbf{g}_{(1)} - \mathbf{g}_{(0)} = \begin{pmatrix} -65.6 \\ -854.4 \end{pmatrix} - \begin{pmatrix} -3080 \\ 240 \end{pmatrix} = \begin{pmatrix} 3014.4 \\ -1094.4 \end{pmatrix} \end{aligned}$$

So

$$\begin{aligned} \mathbf{H}_{(1)}^{-1} &= \mathbf{H}_{(0)}^{-1} + \frac{\mathbf{d}_{(0)} \mathbf{d}_{(0)}^T}{\mathbf{d}_{(0)}^T \mathbf{e}_{(0)}} - \frac{\mathbf{H}_{(0)}^{-1} \mathbf{e}_{(0)} \mathbf{e}_{(0)}^T \mathbf{H}_{(0)}^{-1}}{\mathbf{e}_0^T \mathbf{H}_{(0)}^{-1} \mathbf{e}_{(0)}} \\ &= \begin{pmatrix} 6.21543 & -.154468 \\ -.154468 & .920572 \end{pmatrix} \end{aligned}$$

This ends the first iteration. Computing the second direction:

$$\begin{aligned} \mathbf{d}_{(1)} &= -\mathbf{H}_{(1)}^{-1} \mathbf{g}_{(1)} \\ &= -\begin{pmatrix} 6.21543 & -1544 \\ & .920572 \end{pmatrix} \begin{pmatrix} -65.6 \\ -854.4 \end{pmatrix} = \begin{pmatrix} 275.755 \\ 776.404 \end{pmatrix} \end{aligned}$$

or in normalized form equal to $\begin{pmatrix} .3357 \\ .942 \end{pmatrix}$. To minimize $f(\mathbf{x}_{(1)} + \alpha \mathbf{d}_{(1)})$ with respect to α, set $\frac{\partial f}{\partial \alpha} = 0$ giving $\alpha_{(1)} = 4.990$. Thus

$$\mathbf{x}_{(2)} = \mathbf{x}_{(1)} + \alpha_{(1)}\mathbf{d}_{(1)} = \begin{pmatrix} 17.82 \\ 18.21 \end{pmatrix},$$

and therefore

$$\mathbf{g}_{(2)} = \mathbf{g}(\mathbf{x}_{(2)}) = \begin{pmatrix} -0.017 \\ -0.001 \end{pmatrix}$$

Theoretically, at this stage $\mathbf{g}_{(2)}$ should equal $\mathbf{0}$ because $f(\mathbf{x})$ is quadratic. $f(\mathbf{x}_{(2)}) = 2960.716$ is the minimum of $f(\mathbf{x})$ if we stop the iteration at this stage.

Conjugate Gradient Methods

Two directions $\mathbf{u}$ and $\mathbf{v}$ are conjugate with respect to the positive definite matrix A if $\mathbf{u}'A\mathbf{v} = 0$. The two directions are said to be A-conjugate.

Theorem: If $\boldsymbol{\xi_1}$, $\boldsymbol{\xi_2}$, $\ldots \boldsymbol{\xi_r}$ are a set of vectors mutually conjugate with respect to the positive definite $r \times r$ matrix A, then the minimum of the quadratic form

$$f = a + \mathbf{b}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T A\mathbf{x}$$

can be reached from an arbitrary starting point $\mathbf{x}_{(0)}$ by a finite descent computation in which each of the vectors $\boldsymbol{\xi_i}$ $(i = 1, \ldots, r.)$ is used as a descent direction only once. The order in which $\boldsymbol{\xi_i}$ are used is immaterial.

This implies that if the function to be minimized is the given quadratic form, then the minimum could be located in r *successive one dimensional searches*, in the r conjugate directions. Since any function which is not necessarily a quadratic form, approaches a quadratic surface near the minimum, convergence is assured if we start near enough to the minimum point. Expand $f(\mathbf{x})$ about an arbitrary point $\mathbf{x}_0$ in a Taylor series and set

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \mathbf{f}'(\mathbf{x}_0)^T(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \mathbf{f}''(\mathbf{x}_0)\,(\mathbf{x} - \mathbf{x}_0)$$

where $\mathbf{f}'(\mathbf{x}_0) = \mathbf{g}(\mathbf{x}_0)$ and $\mathbf{f}''(\mathbf{x}_0) = \mathbf{H}(\mathbf{x}_0)$ as before. If $\mathbf{x}_0$ is a minimum, $\mathbf{g}(\mathbf{x}_0) = \mathbf{0}$ implying that

$$f(\mathbf{x}) \approx f(\mathbf{x}_{\min}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_{\min})^T \mathbf{H}(\mathbf{x}_{\min})(\mathbf{x} - \mathbf{x}_{\min})$$

Because of the fact that near the optimum the function $f(\mathbf{x})$ is nearly quadratic, in light of the above theorem, we can use the above approximation to conjecture that using conjugate directions with respect to $\mathbf{H}(\mathbf{x}_{\min})$ would lead to a good algorithm for the optimization

problem. Of course since $\mathbf{x}_{\min}$ is unknown, $\mathbf{H}(\mathbf{x}_{\min})$ cannot be evaluated and thus it is impossible to calculate the conjugate directions in advance. In the method of Fletcher and Reeves, the directional vectors $\boldsymbol{\xi}_{(0)}$, $\boldsymbol{\xi}_{(1)}, \ldots$ are generated in a sequential fashion such that the descent direction at the current point, $\boldsymbol{\xi}_{(i+1)}$, is a linear combination of the negative gradient at the current point, $-\mathbf{g}_{(i)}$, and the previous descent directions, $\boldsymbol{\xi}_{(0)}$, $\boldsymbol{\xi}_{(1)}, \ldots \boldsymbol{\xi}_{(i)}$, so that H-orthogonality, namely,

$$\boldsymbol{\xi}_{(i)}^T \mathbf{H} \boldsymbol{\xi}_{(j)} = 0, \qquad j = 0, \ldots, i-1$$

is satisfied. Let us begin by setting $\boldsymbol{\xi}_{(0)} = -\mathbf{g}_{(0)}$. We have

$$\mathbf{x}_{(1)} = \mathbf{x}_{(0)} + \alpha_{(0)} \boldsymbol{\xi}_{(0)}$$

where, $\alpha_{(0)}$ is found by minimizing $f(\mathbf{x}_{(0)} + \alpha \boldsymbol{\xi}_{(0)})$ as a function of α. Now let $\boldsymbol{\xi}_{(1)}$ be a linear combination of the negative gradient at the new point, $-\mathbf{g}_{(1)}$ and the previous direction, that is

$$\boldsymbol{\xi}_{(1)} = -\mathbf{g}_{(1)} + \gamma_{(1)} \boldsymbol{\xi}_{(0)},$$

$\gamma_{(1)}$ is chosen so that $\boldsymbol{\xi}_{(0)}$ and $\boldsymbol{\xi}_{(1)}$ are H-conjugate. This involves solving $\boldsymbol{\xi}_{(0)}^T \mathbf{H} \boldsymbol{\xi}_{(1)} = 0$ and the above relation simultaneously, and results in

$$\gamma_{(1)} = \frac{\mathbf{g}_{(1)}^T \mathbf{g}_{(1)}}{\mathbf{g}_{(0)}^T \mathbf{g}_{(0)}}.$$

In general we have

$$\boldsymbol{\xi}_{(i+1)} = -\mathbf{g}_{(i+1)} + \gamma_{(i+1)} \boldsymbol{\xi}_{(i)},$$

where $\gamma_{(i+1)}$ is chosen so that $\boldsymbol{\xi}_{(i+1)}$ and $\boldsymbol{\xi}_{(i)}$ are H-conjugate. Then $\gamma_{(i+1)}$ is given by

$$\gamma_{(i+1)} = \frac{\mathbf{g}_{(i+1)}^T \mathbf{g}_{(i+1)}}{\mathbf{g}_{(i)}^T \mathbf{g}_{(i)}}.$$

Hestenes and Stiefel(1953) proposed this solution for quadratic functions and Fletcher and Reeves(1964) applied it for the general case. In general, this leads to the following algorithm:

Fletcher and Reeves Algorithm

1. Select starting point $\mathbf{x}_{(0)}$
2. Compute $\mathbf{g}_{(0)}$ and set $\boldsymbol{\xi}_{(0)} = -\mathbf{g}_{(0)}^*$ where $\mathbf{g}_{(0)}^* = \mathbf{g}_{(0)} / (\mathbf{g}_{(0)}^{\mathrm{T}} \mathbf{g}_{(0)})^{1/2}$ (i.e., normalized $\mathbf{g}_{(0)}$)
3. For $i = 0, \ldots, p-1$ do
   (a) Set $\mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} + \alpha_{(i)} \boldsymbol{\xi}_{(i)}$, where $\alpha_{(i)}$ is found by minimizing $f(\mathbf{x}_{(i)} + \alpha_{(i)} \boldsymbol{\xi}_{(i)})$ using a linear search method, usually cubic polynomial interpolation.
   (b) Compute $\mathbf{g}_{(i+1)} = f'(\mathbf{x}_{(i+1)})$.
   (c) Set $\boldsymbol{\xi}_{(i+1)} = -\mathbf{g}_{(i+1)} + \frac{\mathbf{g}_{(i+1)}^T \mathbf{g}_{(i+1)}}{\mathbf{g}_{(i)}^T \mathbf{g}_{(i)}} \boldsymbol{\xi}_{(i)}$, and return to a).

4. Replace $\mathbf{x}_{(0)}$ by $\mathbf{x}_{(p)}$ and restart iteration.

The last step 4.) was suggested by Fletcher and Reeves and ensures that when the conditions for quadratic convergence becomes valid near the minimum, the construction of conjugate directions is restarted.

`Example 7:` Again consider the minimization of

$$f(\mathbf{x}) = 100(x_1 - 15)^2 + 20(28 - x_1)^2 + 100(x_2{-}_1)^2 + 20(38 - x_1 - x_2)^2$$

As before,

$$\mathbf{f}'(\mathbf{x}) = \mathbf{g}(\mathbf{x}) = \begin{pmatrix} 200(x_1 - 15) + 40(28 - x_1) - 200(x_2\ x_1) - 40(38 - x_1 - x_2) \\ 200(x_2 - x_1 - 40(38 - x_1 - x_2) \end{pmatrix}$$

Starting with

$$\mathbf{x}_{(0)} = \begin{pmatrix} 10 \\ 14 \end{pmatrix} \qquad \mathbf{g}_{(0)} = \begin{pmatrix} -3080 \\ 240 \end{pmatrix}$$

and

$$\boldsymbol{\xi}_{(0)} = -\mathbf{g}_{(0)} = \begin{pmatrix} +3080 \\ -240 \end{pmatrix} \text{ normalized to } \boldsymbol{\xi}_{(0)} = \begin{pmatrix} .997 \\ -.078 \end{pmatrix}$$

Thus

$$\mathbf{x}_{(1)} = \mathbf{x}_{(0)} + \alpha_{(0)}\ \boldsymbol{\xi}_{(0)}$$

Repeating the process, $\alpha_{(0)} = 6.136$ minimizes $f(\mathbf{x}_{(0)} + \alpha\, \boldsymbol{\xi}_{(0)})$ as before, giving

$$\mathbf{x}_{(1)} = \mathbf{x}_{(0)} + 6.136\ \boldsymbol{\xi}_{(0)} = \begin{pmatrix} 16.12 \\ 13.52 \end{pmatrix} \text{ and therefore } \mathbf{g}_{(1)} = \begin{pmatrix} -65.6 \\ -854.4 \end{pmatrix}.$$

Now,

$$\boldsymbol{\xi}_{(1)} = -\mathbf{g}_{(1)} + \frac{\mathbf{g}_{(1)}^T\, \mathbf{g}_{(1)}}{\mathbf{g}_{(0)}^T\, \mathbf{g}_{(0)}}\ \boldsymbol{\xi}_{(0)} = \begin{pmatrix} 302.57 \\ 835.94 \end{pmatrix} \text{ which is normalized to } \begin{pmatrix} 0.340 \\ 0.940 \end{pmatrix}.$$

Thus $\mathbf{x}_{(2)} = \mathbf{x}_{(1)} + \alpha_1\, \boldsymbol{\xi}_{(\mathbf{1})}$ and minimizing $f(\mathbf{x}_{(1)} + \alpha_1\, \boldsymbol{\xi}_{(1)})$ gives $\alpha_1 = 4.994$. Hence

$$\mathbf{x}_{(2)} = \begin{pmatrix} 17.82 \\ 18.21 \end{pmatrix}$$

giving

$$\mathbf{g}_{(2)} = \begin{pmatrix} -.017 \\ -.001 \end{pmatrix}. \qquad \square$$

Direct Search Algorithms

Main feature of direct search algorithms are that they do not require evaluation of derivatives of the objective function, thus readily applicable to problems for which analytical derivatives are difficult to compute.

Nelder-Mead Simplex Algorithm

A simplex is a geometrical shape formed by joining $(p+1)$ points in Euclidean p-space. For example, a triangle in 2-dimensional space is a 2-dimensional simplex. This algorithm attempts to move a simplex on the objective function surface using *reflections*, *extensions*, *contractions* and *shrinking* of the sides forming the simplex. Simplex procedure attempts to produce moves in opposite directions from high values of the objective function rather than moving linearly towards a minimum, thus finding a zigzag path to the minimum.

Let $\mathbf{x}$ be $p \times 1$ and $\mathbf{x}_0, \ldots, \mathbf{x}_p$ be starting values.

Initialize:

Evaluate f at each $\mathbf{x}_0, \ldots, \mathbf{x}_p$:

$$f_i = f(\mathbf{x}_i) \quad i = 0, \ldots, p$$

Let the 3 largest values of f, $f_h \geq f_m \geq f_\ell$ correspond to $\mathbf{x}_h$, $\mathbf{x}_m$, $\mathbf{x}_\ell$, respectively. Thus the new simplex is $\mathbf{x}_h$, $\mathbf{x}_m$, $\mathbf{x}_\ell$.

Start:

Step 0: Use a `reflection` to move away from $\mathbf{x}_h$. Compute centroid $\mathbf{x}_g$ of the other p-points:

$$\mathbf{x}_g = \frac{1}{p} \sum_{i \neq h} \mathbf{x}_i$$

and $\mathbf{x}_r = \mathbf{x}_h + 2(\mathbf{x}_g - \mathbf{x}_h) = 2\mathbf{x}_g - \mathbf{x}_h$. Set $f_r = f(\mathbf{x}_r)$

Step 1: **If** $f_r < f_\ell$
Use an `extension` to move further from the minimum $\mathbf{x}_r$ in the same direction. Compute

$$\mathbf{x}_e = \mathbf{x}_r + (\mathbf{x}_r - \mathbf{x}_g) = 2\mathbf{x}_r - \mathbf{x}_g$$

and set $f_e = f(\mathbf{x}_e)$

Step 1a: **If** $f_e < f_r$

Construct new simplex with $\mathbf{x}_h \leftarrow \mathbf{x}_e$

Else

Construct new simplex with $\mathbf{x}_h \leftarrow \mathbf{x}_r$

Test for convergence and restart.

Step 2: **Else if** $f_r < f_m$
(i.e., $\mathbf{x}_r$ is not the minimum, but better than $\mathbf{x}_m$ and $\mathbf{x}_h$)

Construct new simplex with $\mathbf{x}_h \leftarrow \mathbf{x}_r$. Test for convergence and restart.

Step 3: **Else if** $f_h < f_r$.
($i.e.$, $\mathbf{x}_r$ is worse than $\mathbf{x}_m$ and $\mathbf{x}_\ell$)
Use a `contraction` from $\mathbf{x}_g$. Compute

$$\mathbf{x}_c = \mathbf{x}_h + \frac{1}{2}(\mathbf{x}_g - \mathbf{x}_h) = \frac{1}{2}(\mathbf{x}_h + \mathbf{x}_g)$$

and set $f_c = f(\mathbf{x}_c)$, go to Step 4.

Else

($i.e.$, $\mathbf{x}_r$ is better than $\mathbf{x}_h$ but worse than $\mathbf{x}_m$ or $\mathbf{x}_\ell$).
Use a `contraction` from $\mathbf{x}_r$. Compute

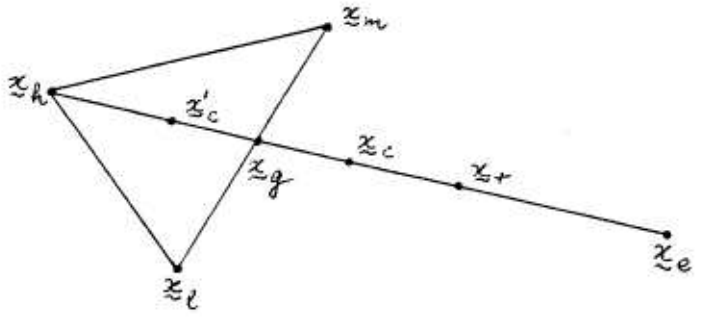
$$\mathbf{x}_c = \mathbf{x}_g + \frac{1}{2}(\mathbf{x}_r - \mathbf{x}_g) = \frac{1}{2}(\mathbf{x}_g + \mathbf{x}_r)$$

and set $f_c = f(\mathbf{x}_c)$, go to Step 4.

Step 4: **If** $f_c < f_h$
Construct new simplex with $\mathbf{x}_h \leftarrow \mathbf{x}_c$. Test for convergence and restart.

else

Replace original simplex $\mathbf{x}_0, \ldots, \mathbf{x}_p$ by $\mathbf{x}'_i = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_\ell)$ i.e., halve the distance of each point from $\mathbf{x}_\ell$. Recalculate $f_i = f(\mathbf{x}'_i)$. Test for convergence and restart.



Test for convergence: Generally based on the standard deviation of the $(p+1)$ function values. If

$$s = \left\{ \frac{1}{p} \sum_{i=0}^{p} (f_i - \bar{F})^2 \right\}^{\frac{1}{2}} < e$$

claim convergence achieved.

Nelder-Mead Sample Computations

	x	Point	Function Value	
Original Simplex	(10,14)	T_2	14,500	
	(10,8)	T_3	17,380	
	(7,10)	T_1	24,940	
	(10,11)	T_4		Centroid of T_2 and T_3
	(13,12)	T_5	8,380	T_1 Reflected to T_5
	(16,13)	T_6	5,500	Expand to T_6
New Simplex	(16,13)	T_6	5,500	Replace T_1 with T_6
	(10,14)	T_2	14,500	
	(10,8)	T_3	17,380	
	(13,13.5)	T_7		Centroid of T_2 and T_6
	(16,19)	T_8	4,060	T_3 Reflected to T_8
	(19,24.5)	T_9	6,850	Expand to T_9 which fails
New Simplex	(16,19)	T_8	4,060	Replace T_3 with T_8
	(16,13)	T_6	5,500	
	(10,14)	T_2	14,500	
	(16,16)	T_{10}		Centroid of T_6 and T_8
	(22,18)	T_{11}	7,300	T_2 Reflected to T_{11}
	(19,17)	T_{12}	3,700	Contraction from T_{11}
New Simplex	(19,17)	T_{12}	3,700	Replace T_2 with T_{12}
	(16,19)	T_8	4,060	
	(16,13)	T_6	5,500	

The process is repeated, Each new simplex formed has lower average function value than preceding ones. After 40 calculations of the objective function, the simplex is

(17.8,18.4)	2,965
(18.0,18.3)	2,965
(17.7,18.0)	2,966

with $s = (\sum(f_i - \overline{F})^2)^{1/2} \approx 0.6$. After 55 calculations of the objective function,

(17.8,18.3)	2,961
(17.8,18.2)	2,961
(17.9,18.2)	2,961

with $s \approx 0.07$. Note: $E(N) = 3.16(n+1)^{2.11}$ for $2 \leq n \leq 10$.

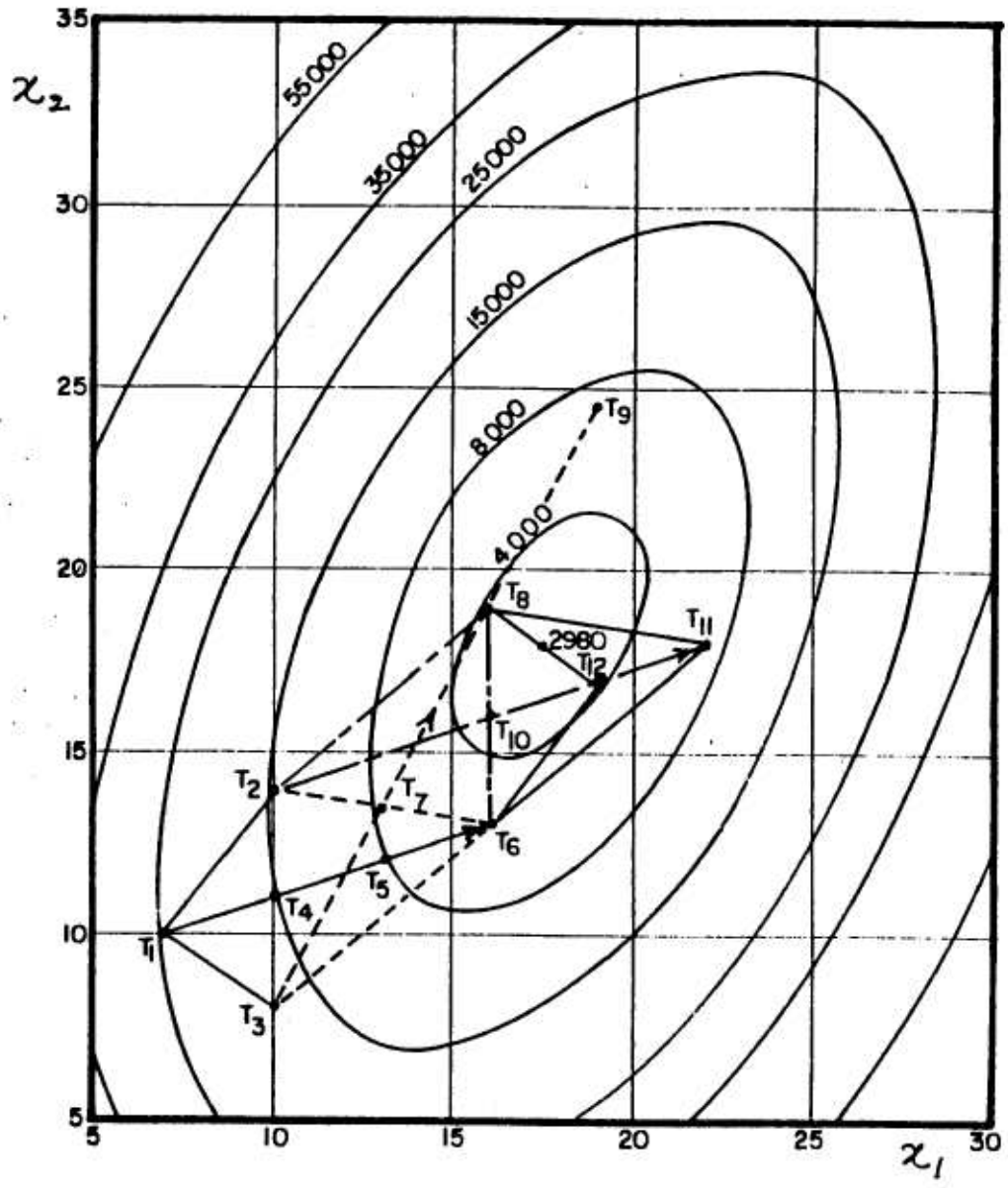


Figure 1: Nelder-Mead Simplex Method: Graphical Illustration of the Example

Hooke and Jeeves Pattern Search

A wide variety of techniques has been developed for the minimization of an unconstrained nonlinear function of multivariables. Among these techniques the direct search methods are the most easily adaptable for use on computers since they tend to use repeated identical arithmetic operations with a simple logic. One of these direct search techniques is the Hooke and Jeeves pattern search technique. It is among the simplest and most efficient methods for solving the unconstrained nonlinear minimization problems. The technique consists of searching the local nature of the objective function in the space and then moving in a favorable direction for reducing the functional value.

The direct search method of Hooke and Jeeves is a sequential search routine for minimizing a function $f(\mathbf{x})$ of more than one variable $\mathbf{x} = (x_1, x_2, \ldots, x_r)$. The argument $\mathbf{x}$ is varied until the minimum of $f(\mathbf{x})$ is obtained. The search routine determines the sequence of values for $\mathbf{x}$. The successive values of $\mathbf{x}$ can be interpreted as points in an r-dimensional space. The procedure consists of two types of moves: `Exploratory` and `Pattern`.

A `move` is defined as the procedure of going from a given point to the following point. A move is a `success` if the value of $f(\mathbf{x})$ decreases (for minimization); otherwise, it is a `failure`. The first type of move is exploratory move which is designed to explore the local behavior of the objective function, $f(\mathbf{x})$. The success or failure of the exploratory moves is utilized by combining it into a pattern which indicates a probable direction for a successful move.

The `exploratory move` is performed as follows:

1. Introduce a starting point $\mathbf{x}$ with a prescribed step length δ_i in each of the independent variables $x_i, i = 1, 2, \ldots, r$.

2. Compute the objective function, $f(\mathbf{x})$ where $\mathbf{x} = (x_1, x_2, \ldots, \ x_i, \ldots, \ x_r)$.
   Set $i = 1$.

3. Compute $f(\mathbf{x})$ at the trial point $\mathbf{x}_n = (x_1, x_2, \ldots, x_i + \delta_i, \ x_{i+1}, \ldots, x_r)$. Call it $f_i(\mathbf{x})$.

   Compare $f_i(\mathbf{x})$ with $f(\mathbf{x})$:

   (i) If $f_i(\mathbf{x}) < f(\mathbf{x})$, this move is a success, set this trial point as the starting point. Set $\mathbf{x} = \mathbf{x}_n = (x_1, x_2, \ldots, x_i + \delta_i, \ldots, x_r)$, $f(\mathbf{x}) = f_i(\mathbf{x})$, and $i = i + 1$, and repeat from step 3.

   (ii) If $f_i(\mathbf{x}) \geq f(\mathbf{x})$, the move is a failure. Go to Step 4.

4. Compute $f(\mathbf{x})$ at new trial point $\mathbf{x}_n = (x_1, x_2, \ldots, x_i - \delta_i, \ldots, x_r)$. Call it $f_i(\mathbf{x})$.

   Compare $f_i(\mathbf{x})$ with $f(\mathbf{x})$:

   (i) If $f_i(\mathbf{x}) < f(\mathbf{x})$, this move is a success, retain the new trial point as a starting point. Set $\mathbf{x} = \mathbf{x}_n = (x_1, x_2, \ldots, x_i - \delta_i, \ldots, x_r)$, $f(\mathbf{x}) = f_i(\mathbf{x})$, and $i = i + 1$, and repeat from Step 3.

(ii If $f_i(\mathbf{x}) \geq f(\mathbf{x})$, then the move is a failure and $\mathbf{x}$ remains unchanged, set $i = i+1$ and repeat from Step 3.

The point $\mathbf{x}_B$ obtained at the end of the exploratory moves, which is reached by repeating Step 3 and 4 until $i = r$, is defined as a `base point`. The starting point introduced in Step 1 of the exploratory move is a starting base point or point obtained by the pattern move.

The `pattern move` is designed to utilize the information acquired in the exploratory move, and executes the actual minimization of the function by moving in the direction of the established pattern. The pattern move is a simple step from the current base to the point

$$\mathbf{x} = \mathbf{x}_B + (\mathbf{x}_B - \mathbf{x}_B^*)$$

$\mathbf{x}_B^*$ is either the starting base point or the preceding base point.

Following the pattern move, a series of exploratory moves are performed to further improve the pattern. If the pattern move nor the exploratory moves brings any improvement, the pattern move is a failure. Then we return to the last base which becomes a starting base and the process is repeated. Otherwise, the pattern move is a success and a new base is established.

If the exploratory moves from any starting base do not yield a point which is better than this base, the lengths of all the steps are reduced and the moves are repeated. Convergence is assumed when the step lengths, δ_i, have been reduced below predetermined limits.

Hooke and Jeeves Example

n	$\underline{x}_B$	$\underline{\delta}$	$\underline{x}$	$f(\underline{x})$	$\underline{x}_n$	$f_i(\underline{x})$	Comments
1	B_0	(2,2)	(5,10)	33,660			Starting Base
2			(5,10)	33,660	(7,10)	24,940	Exp. succ.
3			(7,10)	24,940	(7,12)	24,940	Exp. fail.
4			(7,10)	24,940	(7,8)	25,900	Exp. fail.
2	B_1		(7,10)	24,940			$f(x_2) < f(x_1)$
5			(9,10)	18,140			Pattern
6			(9,10)	18,140	(11,10)	13,260	Exp. succ.
7			(11,10)	13,260	(11,12)	11,980	Exp. succ.
7	B_2		(11,12)	11,980			$f(x_7) < f(x_2)$
8			(15,14)	5,100			Pattern
9			(15,14)	5,100	(17,14)	4,700	Exp. succ.
10			(17,14)	4,700	(17,16)	3,420	Exp. succ.
10	B_3		(17,16)	3,420			$f(x_{10}) < f(x_7)$
11			(23,20)	8,300			Pattern
12			(23,20)	8,300	(25,20)	13,660	Exp. fail.
13			(23,20)	8,300	(21,20)	4,860	Exp. succ.
14			(21,20)	4,860	(21,22)	5,180	Exp. fail.
15			(21,20)	4,860	(21,18)	5,500	Exp. fail.
13			(21,20)	4,860			Pattern move fail. $f(x_{13}) > f(x_{10})$

n	$\underline{x}_B$	$\underline{\delta}$	$\underline{x}$	$f(\underline{x})$	$\underline{x}_n$	$f_i(\underline{x})$	Comments
							Return to $x_{10}(= B_3)$
10	B_3		(17,16)	3,420			Starting base point
16			(17,16)	3,420	(19,16)	4,300	Exp. fail.
17			(17,16)	3,420	(15,16)	4,460	Exp. fail.
18			(17,16)	3,420	(17,18)	3,100	Exp. succ.
18	B_4		(17,18)	3,100			$f(x_{18}) < f(x_{10})$
19			(17,20)	3,740			Pattern
20			(17,20)	3,740	(19,20)	3,340	Exp. succ.
21			(19,20)	3,340	(19,22)	4,300	Exp. fail.
22			(19,20)	3,340	(19,18)	3,340	Exp. fail.
20			(19,20)	3,340			Pattern move fail.
							$f(x_{20}) > f(x_{18})$
							Return to $x_{18}(= B_4)$
18	B_4		(17,18)	3,100			Starting base point
23			(17,18)	3,100	(19,18)	3,340	Exp. fail.
24			(17,18)	3,100	(15,18)	4,780	Exp. fail.
25			(17,18)	3,100	(17,20)	3,740	Exp. fail.
26			(17,18)	3,100	(17,16)	3,420	Exp. fail.
18	B_4		(17,18)	3,100			No better base
							Exp. fail.
							$\delta(2,2) >$ (0.05, 0.05)
							Reduce δ(2,2)
							to δ(1,1)
18	B_4	(1,1)	(17,18)	3,100			Starting base point
27			(17,18)	3,100	(18,18)	2,980	Exp. succ.
28			(18,18)	2,980	(18,19)	3,020	Exp. fail.
29			(18,18)	2,980	(18,17)	3,180	Exp. fail.
27	B_5		(18,18)	2,980			$f(x_{27}) < f(x_{18})$
30			(19,18)	3,340			Pattern
31			(19,18)	3,340	(20,18)	4,180	Exp. fail.
32			(19,18)	3,340	(18,18)	2,980	Exp. succ.
33			(18,18)	2,980	(18,19)	3,020	Exp. fail.
34			(18,18)	2,980	(18,17)	3,180	Exp. fail.
32			(18,18)	2,980			$f(x_{32}) \not< f(x_{27})$
							Pattern move fail.
							Return to $x_{27}(= B_5)$

n	$\underline{x}_B$	$\underline{\delta}$	$\underline{x}$	$f(\underline{x})$	$\underline{x}_n$	$f_i(\underline{x})$	Comments
27	B_5		(18,18)	2,980			Starting base point
35			(18,18)	2,980	(19,18)	3,340	Exp. fail.
36			(18,18)	2,980	(17,18)	3,100	Exp. fail.
37			(18,18)	2,980	(18,19)	3,020	Exp. fail.
38			(18,18)	2,980	(18,17)	3,180	Exp. fail.
27			(18,18)	2,980			No better base Exp. fail. $\delta(1,1) >$ (0.05,0.05) Reduce δ (1,1) to δ(0.5, 0.5)
27	B_5	(0.5,0.5)	(18,18)	2,980			Starting base point
29			(18,18)	2,980	(18.5,18)	3,100	Exp. fail.
40			(18,18)	2,980	(17.5,18)	2,980	Exp. fail.
⋮							

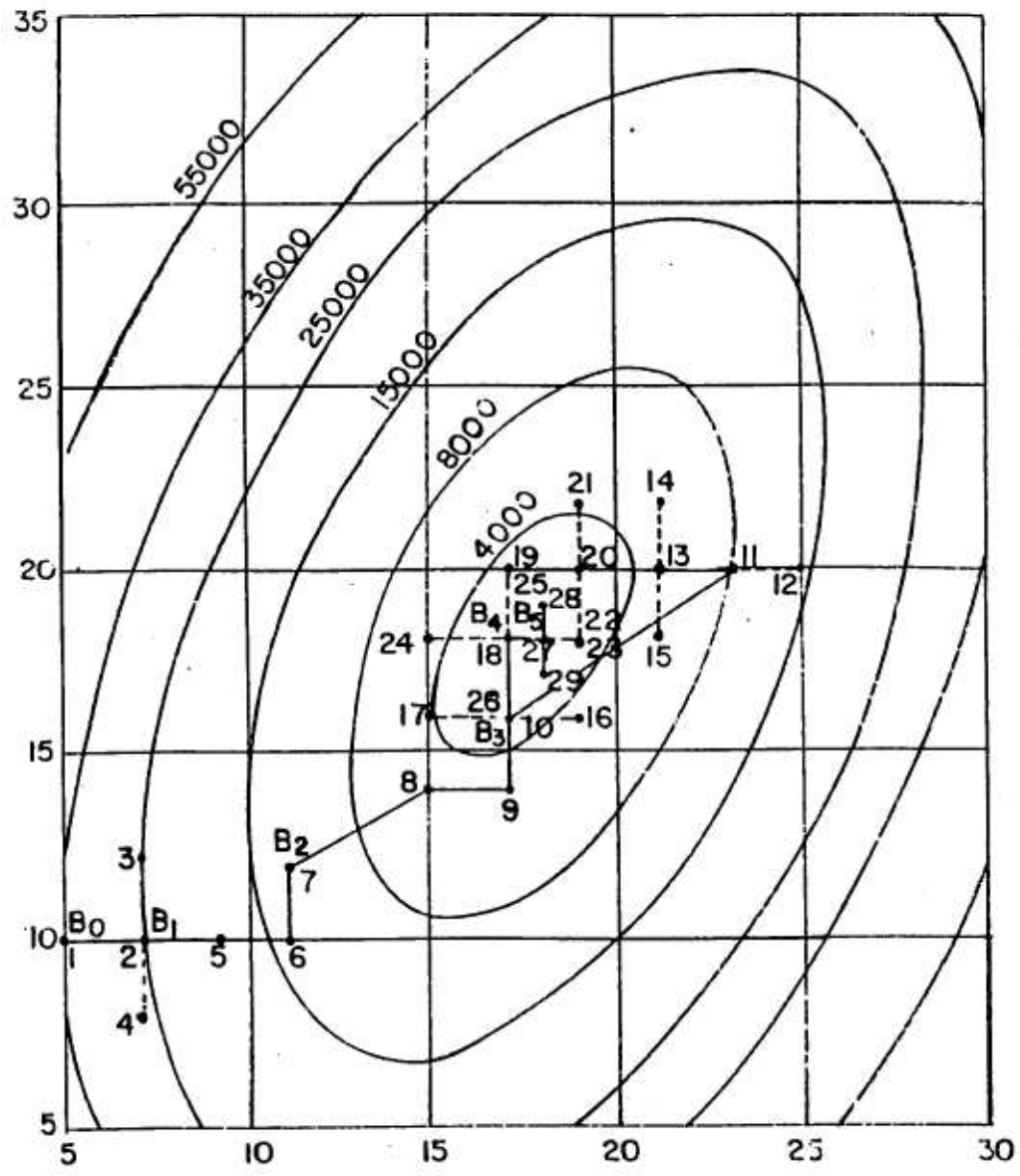


Figure 2: Hooke and Jeeves Pattern Search: Graphical Illustration of the Example

Reference List

Brent R.P. (1973). *Algorithms for Minimization Without Derivatives*. Prentice Hall: Englewood Cliffs, NJ.

Broyden C.G. (1967). "Quasi-Newton methods and their application to function minimization," *Mathematics of Computation*, **21**, 368-381.

Davidon, W.C. (1959). "Variable metric methods for minimization," *Argonne National Lab Report ANL-5990*.

Dennis, J.E., Jr. and Moré, J.J. (1977). "Quasi-Newton methods,motivation and theory," *SIAM Review*, **19**, 46-89.

Dennis, J.E., Jr. and Schnabel, R.B. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice Hall: Englewood Cliffs, NJ.

Fletcher, R. (1970). "A new approach to variable metric algorithms," *The Computer Journal*, **13**, 317-322.

Fletcher, R., and Powell, M.J.D. (1963). "A rapidly convergent descent method for mimimization," *The Computer Journal*, **6**, 163-168.

Fletcher, R., and Reeves, C.M. (1964). "Function mimimization by conjugate gradients," *The Computer Journal*, **7**, 149-154.

Gill, P.E., Murray, W., and Wright, M. (1981). *Practical Optimization*. Academic Press: London.

Goldfarb, D. (1970). "A family of variable metric methods derived by variational means," *Mathematics of Computation*, **24**, 23 -26.

Hooke, R., and Jeeves, T.A. (1961). "Direct search solution of numerical and statistical problems," *Journal of the Association of Computing Machinery*, **8**, 212-229.

Kennedy, W.J., Jr. and Gentle, J.E. (1980). *Statistical Computing*. Marcel Dekker: NY.

Nelder J.A., and Mead, R. (1965). "A Simplex Method for Function Minimization," *The Computer Journal*, **7**, 308-313.

Ortega, J.M. and Rheinboldt, W.C. (1970). *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press: NY.

Powell, M.J.D. (1962). "An iterative method for finding stationary values of a function of several variables," *The Computer Journal*, **5**, 147-151.

Powell, M.J.D. (1964). "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, **7**, 155-162.

Shanno, D.F. (1970). " Conditioning of quasi-Newton method for function minimization," *Mathematics of Computation*, **24**, 647-657.

Thisted, Ronald A. (1988). *Elements of Statistical Computing.* Chapman and Hall: NY.