GRADIENT UNCONSTRAINED MINIMIZATION METHODS

Recall, the Necessary and Sufficient Conditions for an *extremum* (x*) of an unconstrained Function F(**x**) where $\mathbf{x}=(x_1, x_2,\ldots,x_p)^T$.

Necessary Condition: F(**x**) is twice differentiable at **x*** and $\nabla F(\mathbf{x}^*)=\mathbf{0}$, that is, a stationary point exists at **x***.

Sufficient Condition: Hessian matrix $H(\mathbf{x}^*)=\nabla^2 F(\mathbf{x}^*)$ is positive definite ($\mathbf{x}^T\mathbf{H}\mathbf{x}$ is >0 for all $\mathbf{x}\neq\mathbf{0}$.) for a minimum to exist at **x*** or negative definite ($\mathbf{x}^T\mathbf{H}\mathbf{x}$ is <0 for all $\mathbf{x}\neq\mathbf{0}$.) for a maximum to exist at **x***.

Recall also the utility of the eigenvalues of H(**x**):

Test for strict concavity: All *eigenvalues* of H(**x**) are negative (<0).
Test for strict convexity: All *eigenvalues* of H(**x**) are positive (>0).[1]

Type equation here.

The basic problem in optimization is *to find vector x* that minimizes* F(**x**) where F(**x**) is called the *Objective Function* or *Performance Index*. The variables $x_1, x_2, \ldots, x_p$ are called *Decision Variables*.

Generally, we need an iterative scheme because it is not possible to find the exact solutions of the equation that gives the stationary points of F(**x**)

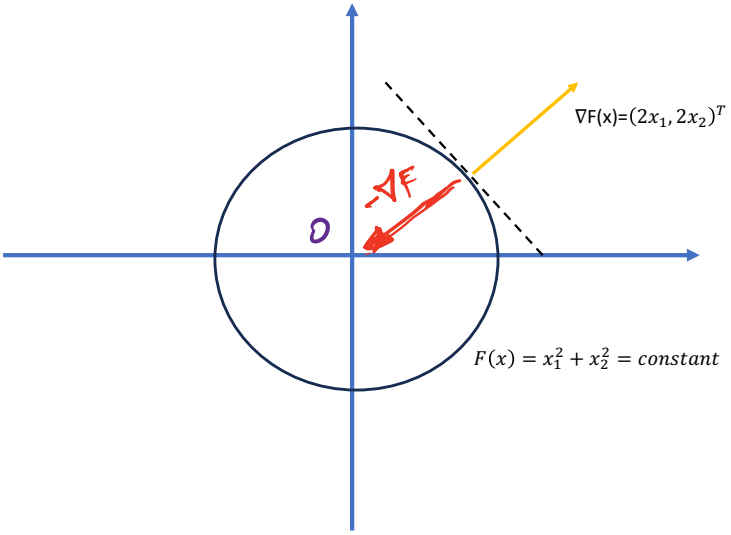
$$\nabla F(\mathbf{x}) \equiv \frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{0}$$

where the operator $\nabla \equiv \left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \ldots, \frac{\partial}{\partial x_p}\right)^T$ is applied to the scalar function F(**x**) yielding the vector, ∇F(**x**), known as the *gradient vector* **g**(**x**). The gradient vector contains the first partial derivatives of the objective function F(**x**) with respect to **x** .

$$\mathrm{g}(\mathrm{x}) = \nabla F(x) = \left(\frac{\partial F(x)}{\partial x_1} + \frac{\partial F(x)}{\partial x_2} + \cdots + \frac{\partial F(x)}{\partial x_p}\right)^T$$

[1] Bellman, R Introduction to Matrix Analysis, 2nd edition, SIAM< 1997

NOTE: The gradient vector $\nabla F(x)$is perpendicular to the surface *F*(**x**)=c (constant) [2].This is seen in the figure below for a two-variable function. In addition, the gradient vector points to the direction of maximum rate of increase of function *F*(**x**).



Minimization: We start from an initial guess for **x**, $\mathbf{x}^{(0)}=[x_1^{(0)}, x_2^{(0)}, \ldots, x_p^{(0)}]^T$. Some heuristic rules can be used in order to obtain an initial guess[3].

At the start of the j^{th} iteration we denote by $\mathbf{x}^{(j)}$ the current estimate of **x**. The j^{th} iteration consists of the computation of a search vector $\Delta\mathbf{x}^{(j+1)}$ from which we obtain the new estimate $\mathbf{x}^{(j+1)}$ according to the following equation

$$\mathbf{x}^{(j+1)} = \mathbf{x}^{(j)} + \mu^{(j)}\Delta\mathbf{x}^{(j+1)}$$

where $\mu^{(j)}$ is the *stepping parameter* also known as *dumping* or *relaxation factor*.

Based on the method to calculate the search vector, $\Delta x^{(j+1)}$, different solution methods to the minimization problem arise.

Convergence: $$\frac{1}{p}\sum_{i=1}^{p}\left|\frac{\Delta x_i^{(j+1)}}{x_i^{(j)}}\right| \leq 10^{-NSIG}$$

where NSIG is the number of desired significant digits in the **x** values. It is assumed that no x_i converges to zero.

---

[2] Spiegel, M.R. Vector Analysis, Schaum's 1959.

[3] Englezos, P., and N.E. Kalogerakis, *Applied Parameter Estimation for Chemical Engineers*, Marcel-Dekker, 2001

The minimization method must be ***computationally efficient*** and ***robust*** (Edgar and Himmelblau, 1988).

- ***Robustness*** refers to the ability to arrive at a solution.
- ***Computational efficiency*** is important since iterative procedures are employed. The speed with which convergence to the optimal values, **x***, is reached is defined with the *asymptotic rate of convergence* (Scales, 1986). An algorithm has a θ^{th} order rate of convergence when **θ** is the largest integer for which the following <u>limit</u> exists.

$$0 \le \lim_{j\to\infty} \frac{\left\| \mathbf{x}^{(j+1)} - \mathbf{x}^* \right\|}{\left\| \mathbf{x}^{(j)} - \mathbf{x}^* \right\|^{\theta}} < 1 \quad \text{or} \quad \frac{\|x^{j+1} - x^*\|}{\|x^j - x^*\|^{\theta}} \le c < 1, j\ large \qquad \text{(A)}$$

In the above equation, the norm $\|\cdot\|$ is usually the Euclidean norm $\|\mathbf{x}\|_2 = \left(x_1^2 + x_2^2 + \ldots + x_n^2\right)^{1/2} = (\mathbf{x}^T\mathbf{x})^{1/2}$. (~~Note replace n with p~~)

- We have a *linear convergence rate* when **θ** is equal to 1. In this case for large j, $\mathbf{x}_j \rightarrow \mathbf{x}^*$. Then $\|x^{j+1} - x^*\| \le c\|x^j - x^*\|$ $j\ large$. This means that at iteration j+1 the error is bounded by c times the error at iteration j, where c<1. If c=0.1 then the error is replaced by a factor of 10 at each iteration. Constant c is called the convergence ratio
  - *Superlinear convergence rate* refers to the case where **θ**=1 and the limit is equal to zero. (USUALLY FAST IN PRACTICE)
- When **θ**=2 the convergence rate is called *quadratic*. In this case, $\|x^{j+1} - x^*\| \le c\|x^j - x^*\|$ $j\ large$. Considering the case where $\|x^j - x^*\| = 10^{-1}$ we have $\|x^{j+1} - x^*\| \le c\ 10^{-2}$. Also, $\|x^{j+2} - x^*\| \le c^2\ 10^{-4}$ and $\|x^{j+3} - x^*\| \le c^3\ 10^{-6}$ and so on. Hence if c=1 the error decreases very rapidly (only a few iterations are needed before the limits of accuracy are reached in (A).

In general, the value of **θ** depends on the algorithm while the value of the limit depends upon the function that is being minimized.

LINEAR: $\frac{\|x^{j+1} - x^*\|}{\|x^j - x^*\|} \le c \quad 0 \le c \le 1$ — usually slow in practice

QUADRATIC: $\frac{\|x^{j+1} - x^*\|}{\|x^j - x^*\|^2} \le c, \quad c \ge 0$ — FASTEST IN PRACTICE

SUPERLINEAR: $\lim_{j\to\infty} \frac{\|x^{j+1} - x^*\|}{\|x^j - x^*\|} \to 0$ (or c_j and $c_j \to 0$ as $j \to \infty$)

Steepest Descent Method. Context. Recall that the gradient of the scalar function F(**x**) is an n-dimensional vector ($\nabla F(\mathbf{x})$) and also the necessary condition for a minimum is that $\nabla F(\mathbf{x}^*)=\mathbf{0}$. A good search direction to find $\mathbf{x}^*$ should reduce (for minimization) the objective function F(**x**) so that if $\mathbf{x}^0$ is the original point and $\mathbf{x}^1$ is the new point $F(\mathbf{x})<F(\mathbf{x}^0)$. Such a direction $\boldsymbol{s}$ is called a descent direction and satisfies the following requirement at any point $\nabla^T F(\mathbf{x}).\mathbf{s}< 0$. To see why, recall $\nabla^T F(\mathbf{x}).\boldsymbol{s}^k=\left|\nabla F(\mathbf{x}^k)\right|\left|\boldsymbol{s}^k\right| cos\ \theta$ and see the figure below[4].
Note that ϑ is the angle between vectors $|\nabla F(\mathbf{x})|$ and $\left|\mathbf{s}^k\right|$. If ϑ =90° then steps along $\mathbf{s}^k$ do not reduce the value of f(x). In other words, at right angles (θ=90°) dF (x)= $\nabla^T F(\mathbf{x}).d\boldsymbol{s}$ =0 and the gradient is perpendicular to the contour lines. If $0\leq\vartheta$ <90° then F(**x**) increases. Only if ϑ >90° F(**x**) decreases.

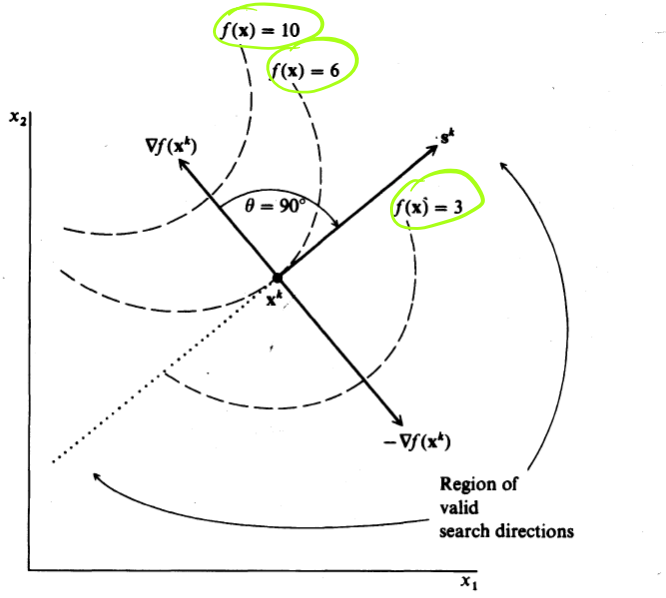


FIGURE 6.5
Identification of the region of possible search directions.

As we see, if we fix the magnitude of $|\boldsymbol{s}|$ and search around various directions (that is varying θ), the maximum change in F(**x**) occurs when θ=0 ($\cos\theta$=1). That is for a fixed distance $|\boldsymbol{s}|$, dF(**x**) is greatest when I move in the same direction as $\nabla F(\mathbf{x})$. Thus, the gradient $\nabla F(\mathbf{x})$ points in the direction of maximum increase of the function F(**x**) or maximum *ascent*. Moreover, the magnitude of $|\nabla F(\mathbf{x})|$ gives the slope (rate of increase) along this maximum direction. We also conclude that the direction of maximum descent (maximum decrease of the function F(**x**)) is opposite to the direction of maximum ascent.

[4] Edgar, T.F. and D.M. Himmelblau, *Optimization of Chemical Processes*, 2nd ed McGraw-Hill, New York, 2001

The steepest descent method. Recall, the j^{th} iteration consists of the computation of a search vector $\Delta\mathbf{x}^{(j+1)}$ from which we obtain the new estimate $\mathbf{x}^{(j+1)}$ according to the following equation

$$\mathbf{x}^{(j+1)} = \mathbf{x}^{(j)} + \mu^{(j)}\Delta\mathbf{x}^{(j+1)}$$

where $\mu^{(j)}$ is the *stepping parameter* also known as *dumping* or *relaxation factor*. For steepest descent method the Search vector is:

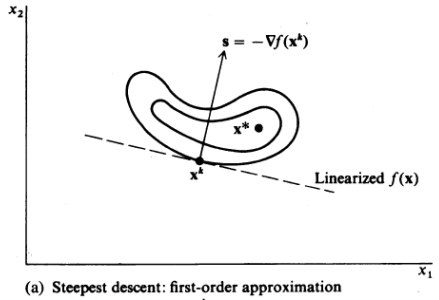
$$\Delta\boldsymbol{x}^{(j+1)} = -\nabla F\left(x^{(j)}\right) = -\boldsymbol{g}\left(\boldsymbol{x}^{(j)}\right)$$

The Steepest Descent method is of interest only for historical and theoretical reasons because its convergence is very slow and oscillations in the **x**-space can easily occur. It is not a viable method for the general purpose minimization of nonlinear functions.

Algorithm - Implementation Steps

1. Input the initial guess for the parameters, $\mathbf{x}^{(0)}$ and NSIG
2. For j=0,1, 2,…, repeat
3. Compute $\Delta\mathbf{x}^{(j+1)}$
4. Determine μ using the *bisection rule*[5] and obtain $\mathbf{x}^{(j+1)}=\mathbf{x}^{(j)}+\mu^{(j)}\Delta\mathbf{x}^{(j+1)}$
5. Continue until the maximum number of iterations is reached or convergence is achieved i.e., $\dfrac{1}{p}\sum_{i=1}^{p}\left|\dfrac{\Delta x_i^{(j+1)}}{x_i^{(j)}}\right| \le 10^{-NSIG}$

The idea of the S.D. method is to minimize at each iteration the linear approximation of F around the current point $\mathbf{x}_j$[6].



(a) Steepest descent: first-order approximation (linearization) of $f(\mathbf{x})$ at $\mathbf{x}^k$

[5] Englezos and Kalogerakis, 2001: *bisection rule* for μ: start with μ=1 and keep on halving μ until $F(x^{J+1})<F(x^J)$

[6] Edgar and Himmelblau, 2001

NEXT NEWTON'S METHOD

UNIVARIATE CASE. $F(x) = 0$

Necessary Condition: $F'(x) = 0$ and to Find the stationary points we must solve for x:

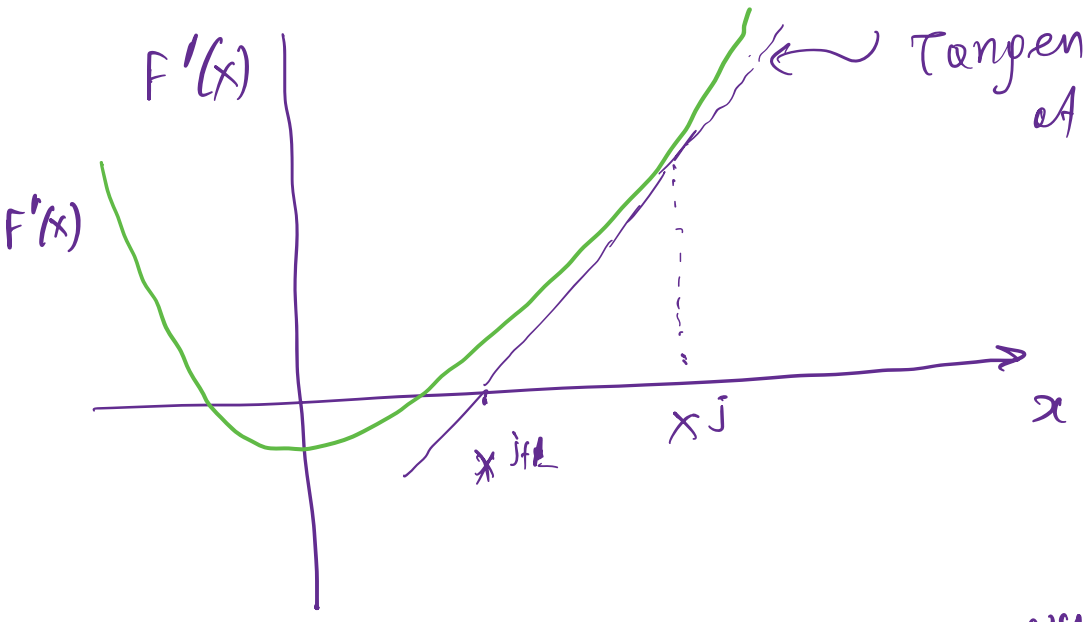
$F'(x) = 0$.

Newton's method:

$$x^{j+1} = x^j - \frac{F'(x^j)}{F''(x^j)}$$

making sure that at each stage j

$F(x^{j+1}) < F(x^j)$ for a minimum.



NEWTON'S METHOD IS EQUIVALENT TO USING A QUADRATIC MODEL FOR A FUNCTION IN MINIMIZATION (OR MAXIMIZATION) AND APPLYING THE NECESSARY CONDITIONS.

$$F(x) = F(x^j) + F'(x^j)(x - x^j) + \frac{1}{2} F''(x^j)(x - x^j)^2$$

$$\frac{dF(x)}{dx} = 0 \quad \therefore \quad F'(x^j) + \frac{1}{2}(2) F''(x^j)(x - x^j) = 0$$

$$\therefore \quad x = x^{j+1} = x^j - \frac{F'(x^j)}{F''(x^j)}$$

Newton's method

Quasi-Newton method

If analytical derivative of $f(x)$ cannot be obtained, we employ Numerical derivative:

$$x^{j+1} = x^j - \frac{[F(x_j+h) - F(x_j-h)]/2h}{[F(x_j+h) - 2F(x) + F(x^j-h)]/h^2}$$

EXAMPLES

- $F(x) = x^2 - x$, $F'(x) = 2x - 1$, $F''(x) = 2 > 0$

  $x_0 = 3 \therefore \quad x_1 = x_0 - \frac{f'(x^0)}{f''(x^0)} = 3 - \frac{5}{2} = 0.5$

  Because $F(x)$ is quadratic, $F'(x)$ is Linear and minimum is obtained in one step

- $F(x) = x^4 - x + 1 \quad \therefore F'(x) = 4x^3 - 1, \; F''(x) = 12x^2$

$$x_1 = x_0 - \frac{4x_0^3 - 1}{12x_0^2} = 3 - \frac{4(3)^3 - 1}{12 * 3^2} = 3 - \frac{107}{108} = 2.009259$$

$$x_2 = 2.009259 - \frac{31.4466}{48.4454} = 1.36015$$

$x_3 = 0.9518103$, $X_4 = 0.7265204$, $X_5 = 0.6422266$

$X_6 = 0.6301933$, $X_7 = 0.6299606$, $X_8 = 0.6299606$

$x_9 = 0.6299605$ Convergence

Newton's Method.

The step-size parameter $\mu^{(j)}$ is taken equal to 1 and the search vector is $\Delta \boldsymbol{x}^{(j=1)} = -[\nabla^2 F(\boldsymbol{x})]^{-1} \nabla F(x^{(j)})$

where $\nabla^2 F(\mathbf{x})$ is the Hessian matrix of $F(\mathbf{x})$ evaluated at $\mathbf{x}^j$ denoted as $\mathbf{H}(\mathbf{x})$. Hence,

$$\Delta \mathbf{x}^{(j+1)} = -\left[\mathbf{H}(\mathbf{x}^{(j)})\right]^{-1} \mathbf{g}(\mathbf{x}^{(j)})$$

There is no need to obtain the inverse of the Hessian matrix because it is better to solve the following linear system of equations

$$\left[\nabla^2 F(\mathbf{x})\right] \Delta \mathbf{x}^{(j+1)} = -\nabla F(\mathbf{x})$$

or equivalently

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

As seen the steepest-descent method arises from Newton's method if we assume that the *Hessian* matrix of $F(\mathbf{x})$ is approximated by the *identity* matrix.

COMMENTS: Newton's method:

(i) It is the most rapidly convergent method when the Hessian matrix of $F(\mathbf{x})$ is available.

(ii) There is no guarantee that it will converge to a minimum from an arbitrary starting point.

(iii) Problems arise when the Hessian matrix is indefinite or singular.

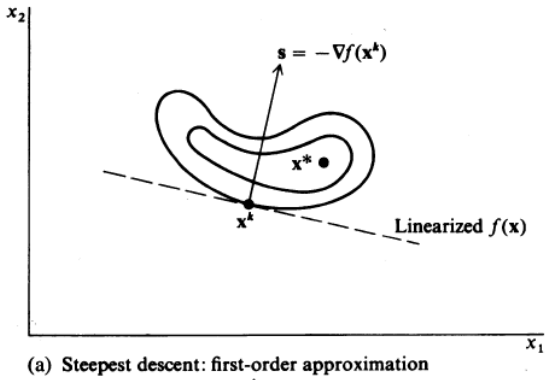
(iv) The method requires analytical first and second order derivatives which may not be practical to obtain. In that case, finite difference techniques may be employed.

(v) Newton's method is not a satisfactory general-purpose algorithm for function minimization, even when a stepping parameter μ is introduced. Fortunately, it can be modified to provide extremely reliable algorithms with the same asymptotic rate of convergence. *The ratio of the largest to the smallest eigenvalue of the Hessian matrix at the minimum is defined as*

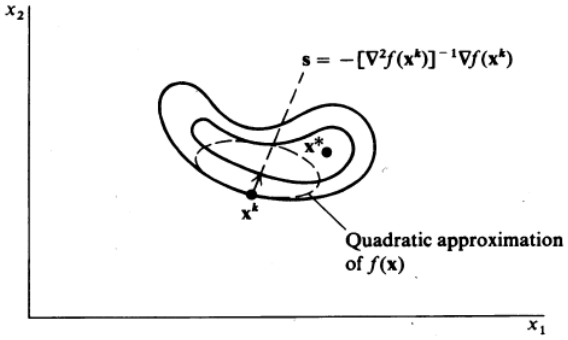
*the **condition number**. The larger the condition number, the more difficult it is for the minimization to converge.*

Solution of the linear Equations: One approach is the Cholesky factorization of H as follows[7] : $\mathbf{H} = \mathbf{L}\,\mathbf{D}\,\mathbf{L}^T$. Matrix D is a diagonal one and L is a lower triagonal matrix with diagonal elements of unity. We prefer to perform an *eigenvalue decomposition.*

NOTE: The idea of the Newton method is to minimize at each iteration the quadratic approximation of F around the current point $\mathbf{x}_j$[8][9].



(a) Steepest descent: first-order approximation (linearization) of $f(\mathbf{x})$ at $\mathbf{x}^k$



(b) Newton's method: second-order (quadratic) approximation of $f(\mathbf{x})$ at $\mathbf{x}^k$

FIGURE 6.9
Comparison of steepest descent with Newton's method from the viewpoint of objective function approximation.

---

[7] Gill et al., 1981, p. 108.

[8] Bertstekas, D.P. Nonlinear Programming, 3rd edition, p. 32; Gill et al., Practical Optimization, 1981, p. 105

[9] Edgar and Himmelblau, 2001

Example[10]

EXAMPLE 6.4 APPLICATION OF NEWTON'S METHOD AND QUADRATIC CONVERGENCE

If we minimize the nonquadratic function

$$f(\mathbf{x}) = (x_1 - 2)^4 + (x_1 - 2)^2 x_2^2 + (x_2 + 1)^2$$

from the starting point of (1, 1), can you show that Newton's method exhibits quadratic convergence? *Hint*: Show that

$$\frac{\|\mathbf{x}^{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}^k - \mathbf{x}^*\|^2} < c \quad \text{(see Section 5.3)}$$

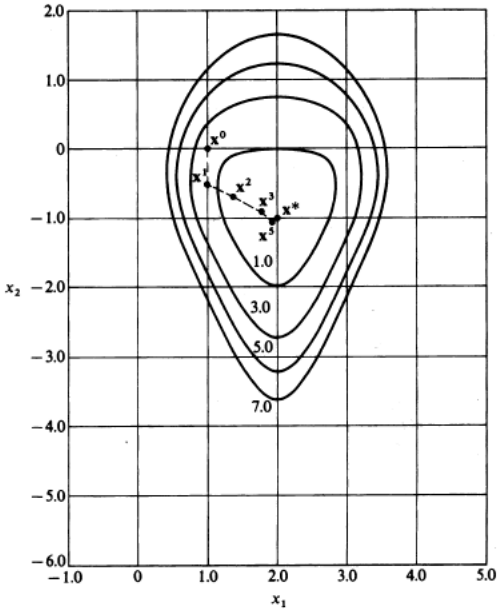


FIGURE E6.4

Solution. Newton's method produces the following sequences of values for x_1, x_2, and $[f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)]$ (you should try to verify the calculations shown in the following table; the trajectory is traced in Figure E6.4).

Iteration	x_1	x_2	$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)$
0	1.000000	1.000000	6.000
1	1.000000	−0.500000	1.500
2	1.391304	−0.695652	4.09×10^{-1}
3	1.745944	−0.948798	6.49×10^{-2}
4	1.986278	−1.048208	2.53×10^{-3}
5	1.998734	−1.000170	1.63×10^{-6}
6	1.9999996	−1.000002	2.75×10^{-12}

You can calculate between iterations 2 and 3 that $c = 0.55$; and between 3 and 4 that $c \approx 0.74$. Hence, quadratic convergence can be demonstrated numerically.

[10] Edgar, T.F., D.M. Himmelblau, L.S. Lasdon, *Optimization of Chemical Processes*, 2nd edition, McGraw-Hill, New York, 2001.

Modified Newton's Methods

- Modified Newton methods attempt to alleviate the deficiencies of Newton's method. The problem arises if the Hessian matrix, **G**, is not positive definite.
- If any of the eigenvalues are not positive then a procedure proposed by Marquardt (1963) based on earlier work by Levenberg (1944) should be followed[11][12]. A positive value $\boldsymbol{\gamma}$ can be added to all the eigenvalues such that the resulting positive quantities, $\lambda_i+\boldsymbol{\gamma}$, i=1,2,…,m are the eigenvalues of a positive matrix $\mathbf{H}_{LM}$, given by

$$\mathbf{H}_{LM} = \mathbf{G} + \boldsymbol{\gamma}\,\mathbf{I}$$

where **I** is the identity matrix.

Modified Newton methods require calculation of second derivatives. If derivatives are not available analytically, we calculate them by finite differences (considerable number of gradient evaluations if the number of parameters, p, is large). Finite difference approximations of derivatives are prone to truncation and round-off errors and require a considerable number of gradient evaluations if the number of parameters, p, is large

[11] Marquardt, D.W., J. Soc. Indust. Appl. Math., 11(2), 431-441, June 1963))

[12] Levenberg, K., Quart. Appl. Math., II92), 164-168, 1944

Conjugate Gradient Methods

Conjugate gradient-type methods form a class of minimization procedures that accomplish two objectives:

(a) There is no need for calculation of second order derivatives;

(b) they have relatively small computer storage requirements.

These methods are suitable for problems with a very large number of parameters. Two versions of the method have been formulated:

(a) *Fletcher-Reeves* (FR) version: the earliest conj grad. method

(b) *Polak-Ribiere* (PR) version: recommended for its slightly better convergence properties

Algorithms[13,14]: We consider the general case where we need to minimize a function F (**x**) where **x** is a n-dimensional vector.

Step 1. At $\mathbf{x}_0$ calculate $F(\mathbf{x}_0)$. Let $\mathbf{p}_0 = -\nabla F(\mathbf{x}_0)$.

Step 2. Save $\nabla F(\mathbf{x}_0)$ and compute $\mathbf{x}_1 = \mathbf{x}_0 + \lambda_0 \mathbf{p}_0$

Step 3. Calculate $F(\mathbf{x}_1)$, $\nabla F(\mathbf{x}_1)$. The new search direction is a linear combination of $\mathbf{p}_0$ and $\nabla F(\mathbf{x}_1)$:

$$\mathbf{p}_1 = -\nabla F(\mathbf{x}_1) + \mathbf{p}_0 \frac{\nabla^T F(\mathbf{x}_1).\nabla F(\mathbf{x}_1)}{\nabla^T F(\mathbf{x}_0).\nabla F(\mathbf{x}_0)}$$

For the k-th iteration

FR method $$\mathbf{p}_{k+1} = -\nabla F(\mathbf{x}_{k+1}) + \mathbf{p}_k \frac{\nabla^T F(\mathbf{x}_{k+1}).\nabla F(\mathbf{x}_{k+1})}{\nabla^T F(\mathbf{x}_k).\nabla F(\mathbf{x}_k)}$$

PR method $$\mathbf{p}_{k+1} = -\nabla F(\mathbf{x}_{k+1}) + \mathbf{p}_k \frac{\left[\nabla F(\mathbf{x}_{k+1}) - \nabla F(\mathbf{x}_k)\right]^T.\nabla F(\mathbf{x}_{k+1})}{\nabla^T F(\mathbf{x}_k).\nabla F(\mathbf{x}_k)}$$

Step 4. Test for convergence to the minimum of F(x). If convergence is not attained go to step 3.

Step 5. Terminate the algorithm when $\|\mathbf{p}_k\| <$ TOL (prescribed tolerance). e.g. TOL=0.00005

---

[13] Scales, L.E., *Introduction to Non-linear Optimization*, Springer-Verlag, New York, NY, 1985

[14] Edgar, T.F. and D.M. Himmelblau, *Optimization of Chemical Processes*, McGraw-Hill, New York, NY, 1988.

EXAMPLE (6.2)[15]

EXAMPLE 6.2 APPLICATION OF THE FLETCHER–REEVES CONJUGATE GRADIENT ALGORITHM

We solve the problem known as Rosenbrock's function

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

starting at $\mathbf{x}^{(0)} = [-1.2 \;\; 1.0]^T$. The first few stages of the Fletcher–Reeves procedure are listed in Table E6.2. The trajectory as it moves toward the optimum is shown in Figure E6.2.

TABLE E6.2
Results for Example 6.2 using the Fletcher–Reeves method

Iteration	Number of function calls	$f(\mathbf{x})$	x_1	x_2	$\frac{\partial f(\mathbf{x})}{\partial x_1}$	$\frac{\partial f(\mathbf{x})}{\partial x_2}$
0	1	24.2	−1.2	1.0	−215.6	−88.00
1	4	4.377945	−1.050203	1.061141	−21.65	−8.357
5	14	3.165142	−0.777190	0.612232	−1.002	−1.6415
10	28	1.247687	−0.079213	−0.025322	−3.071	−5.761
15	41	0.556612	0.254058	0.063189	−1.354	−0.271
20	57	0.147607	0.647165	0.403619	3.230	−3.040
25	69	0.024667	0.843083	0.710119	−0.0881	−0.1339
30	80	0.0000628	0.995000	0.989410	0.2348	−0.1230
35	90	1.617×10^{-15}	1.000000	1.000000	-1.60×10^{-8}	-3.12×10^{-8}

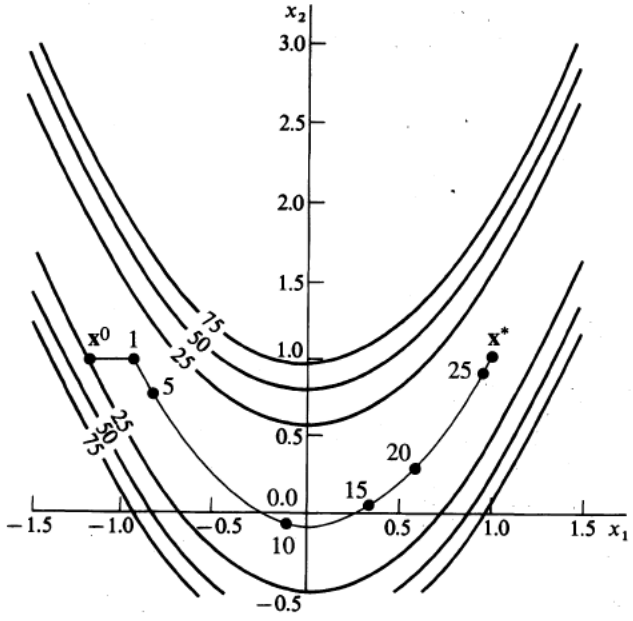


FIGURE E6.2
Search trajectory for the Fletcher–Reeves algorithm (the numbers designate the iteration).

[15] Edgar, T.F., D.M. Himmelblau, L.S. Lasdon, *Optimization of Chemical Processes*, 2nd edition, McGraw-Hill, New York, 2001.

Quasi-Newton or Variable Metric or Secant Methods[16]

These methods avoid calculation of the elements of the ($p \times p$) Hessian matrix but they rely on formulas that approximate the Hessian and its inverse.

Two algorithms have been developed:

(a) The Davidon-Fletcher-Powell Formula (DFP)

(b) The Broyden-Fletcher-Goldfard-Shanno Formula (BFGS) .

The BFGS method is considered to be superior to DFP because it is less prone to loss of positive definiteness or to singularity problems through round off errors

Algorithms.

We consider the minimization of a function F($\mathbf{x}$). Recall Newton's method:

$\mathbf{x}_{k+1}=\mathbf{x}_k+\mathbf{p}_k$ where $\mathbf{p}_k= - \mathbf{G}_k^{-1}\mathbf{g}_k$ or equivalently $\mathbf{G}_k \mathbf{p}_k= -\mathbf{g}_k$.

where $\mathbf{g}_k=\nabla F(\mathbf{x}_k)$ (*gradient* vector) and $\mathbf{G}_k=\nabla^2 F(\mathbf{x}_k)$ (*Hessian* matrix).

Instead of using $\mathbf{G}_k$ and $\mathbf{G}_k^{-1}$ i.e. the true Hessian and inverse of the Hessian matrix we use the approximate matrices $\mathbf{B}_k$ and $\mathbf{H}_k$ respectively.

Hence, equations to be solved: $\mathbf{p}_k= - \mathbf{H}_k \mathbf{g}_k$ or $\mathbf{B}_k \mathbf{p}_k= -\mathbf{g}_k$.

If $\mathbf{H}_k$ and $\mathbf{B}_k$ are available how do we obtain $\mathbf{H}_{k+1}$ and $\mathbf{B}_{k+1}$?

The matrix $\mathbf{H}_{k+1}$ can be written as following: $\mathbf{H}_{k+1}=\mathbf{H}_k+(\mathbf{H}_{k+1}-\mathbf{H}_k)\equiv\mathbf{H}_k+\mathbf{Q}_k^H$ where $\mathbf{Q}_k^H$ is a matrix which is calculated from the values of $\mathbf{x}_k$, $\mathbf{x}_{k+1}$, $\mathbf{g}_k$, $\mathbf{g}_{k+1}$, and $\mathbf{H}_k$. The initial approximation $\mathbf{H}_0$ can be any positive definite matrix.

Similarly, we want $\mathbf{B}_{k+1}= \mathbf{B}_k+\mathbf{Q}_k^B$ with $\mathbf{B}_k \Delta\mathbf{x}_k=-\mathbf{g}_k$. (with $\mathbf{p}_k=\mathbf{x}_{k+1}-\mathbf{x}_k=\Delta\mathbf{x}_k$) in accordance with equation $\mathbf{G}_k \Delta\mathbf{x}_k=-\mathbf{g}_k$.

[16] Edgar, T.F. and D.M. Himmelblau, *Optimization of Chemical Processes*, McGraw-Hill, New York, NY, 1988; Scales, L.E., *Introduction to Non-linear Optimization*, Springer-Verlag, New York, NY, 1985;.

The Davidon-Fletcher-Powell Formula (DFP)[17]

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \left\{ \frac{\Delta\mathbf{x}_k \Delta\mathbf{x}_k^T}{\Delta\mathbf{x}_k^T \Delta\mathbf{g}_k} - \frac{\mathbf{H}_k \Delta\mathbf{g}_k \Delta\mathbf{g}_k^T \mathbf{H}_k^T}{\Delta\mathbf{g}_k^T \mathbf{H}_k \Delta\mathbf{g}_k} \right\} \qquad (A)$$

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \left\{ \frac{\Delta\mathbf{g}_k \Delta\mathbf{g}_k^T}{\Delta\mathbf{g}_k^T \Delta\mathbf{x}_k} - \frac{\mathbf{B}_k \Delta\mathbf{x}_k \Delta\mathbf{x}_k^T \mathbf{B}_k}{\Delta\mathbf{x}_k^T \mathbf{B}_k \Delta\mathbf{x}_k} + \Delta\mathbf{x}_k^T \mathbf{B}_k \Delta\mathbf{x}_k \mathbf{V}_k \mathbf{V}_k^T \right\} \qquad (B)$$

where $\mathbf{V}_k = \frac{\Delta\mathbf{g}_k}{\Delta\mathbf{g}_k^T \Delta\mathbf{x}_k} - \frac{\mathbf{B}_k \Delta\mathbf{x}_k}{\Delta\mathbf{x}_k^T \mathbf{B}_k \Delta\mathbf{x}_k}$. $\mathbf{B}_{k+1}$ is also given by

$$\mathbf{B}_{k+1} = \left[\mathbf{I} - \frac{\Delta\mathbf{g}_k \Delta\mathbf{x}_k^T}{\Delta\mathbf{g}_k^T \Delta\mathbf{x}_k} \right] \mathbf{B}_k \left[\mathbf{I} - \frac{\Delta\mathbf{g}_k \Delta\mathbf{x}_k^T}{\Delta\mathbf{g}_k^T \Delta\mathbf{x}_k} \right]^T + \frac{\Delta\mathbf{g}_k \Delta\mathbf{g}_k^T}{\Delta\mathbf{g}_k^T \Delta\mathbf{x}_k} \qquad (C)$$

The Broyden-Fletcher-Goldfard-Shanno (BFGS) Formula[15]

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \left\{ \frac{\Delta\mathbf{x}_k \Delta\mathbf{x}_k^T}{\Delta\mathbf{x}_k^T \Delta\mathbf{g}_k} - \frac{\mathbf{H}_k \Delta\mathbf{g}_k \Delta\mathbf{g}_k^T \mathbf{H}_k^T}{\Delta\mathbf{g}_k^T \mathbf{H}_k^T \Delta\mathbf{g}_k} + \Delta\mathbf{g}_k^T \mathbf{H}_k \Delta\mathbf{g}_k \mathbf{W}_k \mathbf{W}_k^T \right\} \qquad (D)$$

where $\mathbf{W}_k = \frac{\Delta\mathbf{x}_k}{\Delta\mathbf{x}_k^T \Delta\mathbf{g}_k} - \frac{\mathbf{H}_k \Delta\mathbf{g}_k}{\Delta\mathbf{g}_k^T \mathbf{H}_k \Delta\mathbf{g}_k}$. $\mathbf{H}_{k+1}$ is also given by

$$\mathbf{H}_{k+1} = \left[\mathbf{I} - \frac{\Delta\mathbf{x}_k \Delta\mathbf{g}_k^T}{\Delta\mathbf{x}_k^T \Delta\mathbf{g}_k} \right] \mathbf{H}_k \left[\mathbf{I} - \frac{\Delta\mathbf{x}_k \Delta\mathbf{g}_k^T}{\Delta\mathbf{x}_k^T \Delta\mathbf{g}_k} \right]^T + \frac{\Delta\mathbf{x}_k \Delta\mathbf{x}_k^T}{\Delta\mathbf{x}_k^T \Delta\mathbf{g}_k} \qquad (E)$$

The matrix $\mathbf{B}_{k+1}$ is given by

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \left\{ \frac{\Delta\mathbf{g}_k \Delta\mathbf{g}_k^T}{\Delta\mathbf{g}_k^T \Delta\mathbf{x}_k} - \frac{\mathbf{B}_k \Delta\mathbf{x}_k \Delta\mathbf{x}_k^T \mathbf{B}_k}{\Delta\mathbf{x}_k^T \mathbf{B}_k \Delta\mathbf{x}_k} \right\} \qquad (F)$$

NOTE: Since $\mathbf{B}_k\Delta\mathbf{x}_k = \mathbf{B}_k(\mathbf{x}_{k+1}-\mathbf{x}_k) = \mathbf{B}_k\lambda_k\mathbf{p}_k = -\lambda_k\mathbf{g}_k$ i.e. $\mathbf{B}_k\Delta\mathbf{x}_k = -\lambda_k\mathbf{g}_k$, the matrix - vector products in eqns (B), (C) or (F) can be eliminated. This saves computation. There is no corresponding effect possible for the term $\mathbf{H}_k\Delta\mathbf{g}_k$ in eqns (A), (D) and (E).

[17] Scales, L.E., *Introduction to Non-linear Optimization*, Springer-Verlag, New York, NY, 1985.

EXAMPLE 6.6[18]

EXAMPLE 6.6 APPLICATION OF THE BFGS METHOD

Apply the BFGS method to find the minimum of the function $f(\mathbf{x}) = x_1^4 - 2x_2x_1^2 + x_2^2 + x_1^2 - 2x_1 + 5$.

Use a starting point of (1,2) and terminate the search when f changes less than 0.00005 between iterations. The contour plot for the function was shown in Figure 5.7.

Solution. Using the Optimization Toolbox from MATLAB, the BFGS method requires 20 iterations before the search is terminated, as shown below.

TABLE E6.6
BFGS method

Iteration	x_1	x_2	$f(x_1, x_2)$	$\frac{\partial f}{\partial x_1}$	$\frac{\partial f}{\partial x_2}$
	1.00000	2.00000	5.00000	−4.00000	2.00000
1	1.29611	1.82473	4.10866	−0.15866	0.28966
2	1.29192	1.73556	4.08964	0.24022	0.13299
3	1.22980	1.63069	4.06680	−0.12218	0.23654
4	1.22409	1.54972	4.05285	0.19694	0.10263
5	1.17160	1.46528	4.03803	−0.09085	0.18524
6	1.16530	1.39587	4.02876	0.15372	0.07589
7	1.12318	1.33087	4.01998	−0.06513	0.13867
8	1.11718	1.27501	4.01446	0.11408	0.05383
9	1.08519	1.22728	4.00972	−0.04507	0.09927
10	1.08012	1.18504	4.00676	0.08077	0.03678
11	1.05705	1.15150	4.00442	−0.03024	0.06828
12	1.05314	1.12129	4.00297	0.05494	0.02438
13	1.03725	1.09861	4.00190	−0.01977	0.04544
14	1.03444	1.07795	4.00125	0.03623	0.01578
15	1.02386	1.06305	4.00079	−0.01269	0.02950
16	1.02195	1.04940	4.00051	0.02335	0.01005
17	1.01509	1.03981	4.00032	−0.00803	0.01882
18	1.01382	1.03100	4.00020	0.01482	0.00632
19	1.00945	1.02492	4.00012	−0.00503	0.01186
20	1.00863	1.01932	4.00008	0.00930	0.00395

[18] Edgar, T.F., D.M. Himmelblau, L.S. Lasdon, *Optimization of Chemical Processes*, 2nd edition, McGraw-Hill, New York, 2001.

Quasi-Newton Methods Using H-matrices

1. Input $\mathbf{x}_0$, $\mathbf{H}_0$, *gtol*
2. For k=0,1,…, repeat
3. Set $\mathbf{p}_k = -\mathbf{H}_k\mathbf{g}_k$
4. Compute λ_k
5. Set $\mathbf{x}_{k+1}=\mathbf{x}_k+\lambda_k\mathbf{p}_k$
6. Compute $\mathbf{Q}_k^H$
7. Set $\mathbf{H}_{k+1}=\mathbf{H}_k+\mathbf{Q}_k^H$
8. End until $\|\mathbf{g}_{k+1}\| < gtol$

Line	*Comments*
1	$\mathbf{H}_0 = \mathbf{I}$ if no better positive definite estimate is available
6,7	DFP method uses eqn (A) whereas the BFGS uses eqn (D) or (E).

Quasi-Newton Methods Using B-matrices

1. Input $\mathbf{x}_0$, $\mathbf{B}_0$, *gtol*
2. For k=0,1,…, repeat
3. Solve $\mathbf{B}_k\mathbf{p}_k = -\mathbf{g}_k$ to obtain $\mathbf{p}_k$
4. Compute λ_k
5. Set $\mathbf{x}_{k+1}=\mathbf{x}_k+\lambda_k\mathbf{p}_k$
6. Compute $\mathbf{Q}_k^B$
7. Set $\mathbf{B}_{k+1}=\mathbf{B}_k+\mathbf{Q}_k^B$
8. End until $\|\mathbf{g}_{k+1}\| < gtol$

Line	*Comments*
1	$\mathbf{B}_0 = \mathbf{I}$ if no better positive definite estimate is available
6,7	DFP method uses eqn (B) or (C) whereas the BFGS uses eqn (F).

Implementation of Numerically Stable Quasi-Newton Methods .

- There is a tendency for matrices $\mathbf{H}_k$ and $\mathbf{B}_k$ to become singular because of rounding errors (DFP more than BFGS). A drastic way out of this problem is to reset matrix $\mathbf{H}_k$ or $\mathbf{B}_k$ to the unit matrix $\mathbf{I}$ but any useful information accumulated up to that point is then lost. Gill and Murray (1972) have shown how to avoid singularity without undue loss of information[19].
- Although $\mathbf{H}_k$ and $\mathbf{B}_k$ are positive definite in theory, the DFP method has a tendency to produce $\mathbf{H}_k$ and $\mathbf{B}_k$'s that are not positive definite because of computer round off error, while the BFGS method does not.

[19] Gill, P.E., and Murray, W. (1972), *J. Inst. Math. Appl.*, 9, pp 91-108; Goldfarb, D., *Math. Comp.*, 30, 796 (1976)}

SUMMARY: Comments on Gradient methods.

Steepest-Descent

- It is of interest for historical and theoretical reasons
- Not a viable method for general purpose minimization

Newton

- Most rapidly convergent when $G(\mathbf{x})=\nabla^2 F(\mathbf{x})$ is available
- No guarantee to converge to a minimum from an arbitrary starting point
- Problems when $G(\mathbf{x})$ is indefinite or singular. *Modified Newton* bypass this problem by using a positive definite approximation of $G(\mathbf{x})$ (Marquardt - Levenberg)
- G(x) may be approximated by finite difference techniques if necessary and, if it is large and sparse great savings may be possible in the number of gradient evaluations required.

The remaining methods do away with the explicit need for second derivatives of function F(x).

Conjugate Gradient

- They are suitable for problems with a large number of variables since storage requirements are extremely modest
  - Polak-Ribiere is recommended

Quasi-Newton

- The Hessian and its inverse are approximated
- Better rates of convergence than conjugate gradient methods
- Similar storage requirements to Newton's method
- BFGS is superior to DFP in most cases:
  - less prone to loss of positive definiteness through round off errors;
  - less tendency for matrices Hk and Bk to become singular because of round off errors is less marked for the BFGS method
  - better theoretical convergence properties.