5. Search Methods

Search methods include line and multidimensional searches. The basic idea of the methods is as follows:

1. Start with a reasonable estimate for the minimum $\mathbf{x}_0$.
2. Determine a search direction $\mathbf{d}_k$.
3. Determine a step size α_k.
4. Get a new estimate for the minimum $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$.
5. Iterate until convergence, tested by, for example: $\left\| f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) \right\| < \varepsilon_1$, $\left\| \mathbf{x}_{k+1} - \mathbf{x}_k \right\| < \varepsilon_2$ and/or $\left\| \nabla f(\mathbf{x}_k) \right\| < \varepsilon_3$.

With the exception of random (or pseudo-random) search methods, all search methods converge on a local minimum. If the function to be optimized is not unimodal, then several optimizations from different starting points are required if the global minimum is to be located with any certainty.

5.1 Line Searches (1-D)

The simplest and most time consuming method is to select a trial solution x_0 and keep incrementing it by a small amount δx as follows:

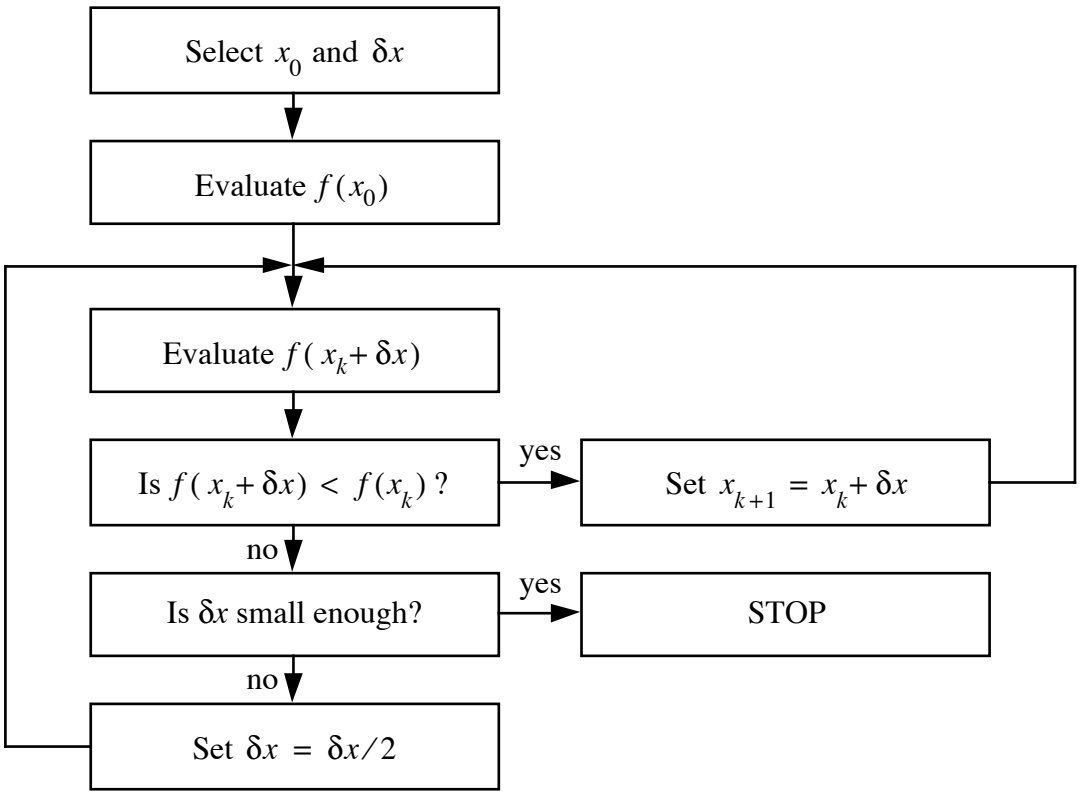
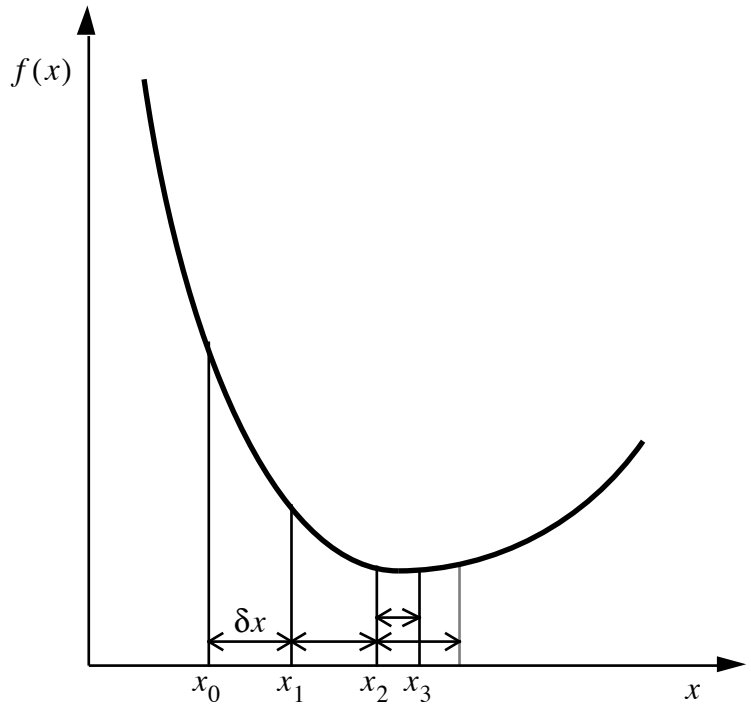


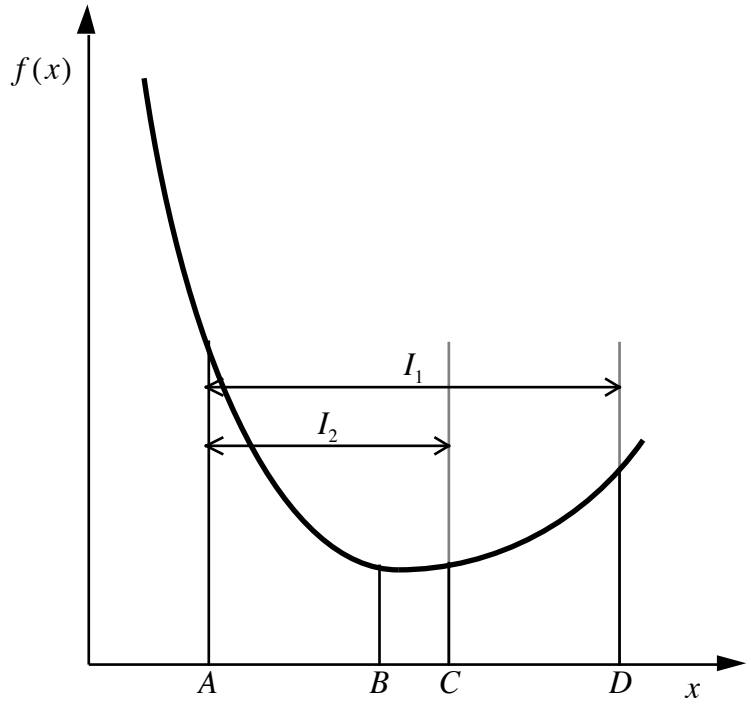
Figure 5.1: An Example of Simple 1-D Line Search.



5.2 Interval Reduction

A more efficient method is by interval reduction. The aim is to keep reducing the interval I within which the minimum is known to lie until the interval is acceptably small.

Figure 5.2: An Example of Interval Reduction Line Search.



Method:

1. Assume the minimum lies between A and D.
2. Evaluate $f(x)$ at two interior points B and C.
3. If $f(B) < f(C)$, then the new interval is $A - C$.
   If $f(B) > f(C)$, then the new interval is $B - D$.
   If $f(B) = f(C)$, then the new interval is either $A - C$ or $B - D$.

To achieve the greatest interval reduction per step, select B and C very close to the centre of the interval (but not too close). This almost halves the interval at each step.

5.2.1 Measures of Convergence

The *rate of convergence* of an algorithm is measured by the number of times $f(\mathbf{x})$ needs to be evaluated to locate the optimum with acceptable accuracy. For interval reduction methods, this means reducing the interval to an acceptably small value.

If the sequence of estimates $\{\mathbf{x}_k\}$ converges to a minimum $\mathbf{x}^*$ such that

$$\lim_{k \to \infty} \frac{\left\| \mathbf{x}_{k+1} - \mathbf{x}^* \right\|}{\left\| \mathbf{x}_k - \mathbf{x}^* \right\|^p} = \beta \,, \tag{5.1}$$

then we get asymptotically that

$$\left\| \mathbf{x}_{k+1} - \mathbf{x}^* \right\| = \beta \left\| \mathbf{x}_k - \mathbf{x}^* \right\|^p \tag{5.2}$$

and the sequence $\{\mathbf{x}_k\}$ is said to converge with *convergence ratio* β and *order of convergence* p.

Note that larger values of the order p imply faster convergence, while larger convergence ratios β imply slower convergence.

For interval reduction line searches, the order of convergence p can also be expressed as

$$p = \left[\frac{I_1}{I_k} \right]^{\frac{1}{n}} \tag{5.3}$$

where I_1 and I_k are the sizes of the initial and k-th intervals and n is the number of evaluations performed.

For the interval reduction method, it takes two function evaluations to reduce the interval by a half (nearly), so the order of convergence is

$$p \approx \left[\frac{I_1}{\frac{1}{2} I_1} \right]^{\frac{1}{2}} = \sqrt{2} \,. \tag{5.4}$$

5.2.2 Golden Section Interval Reduction

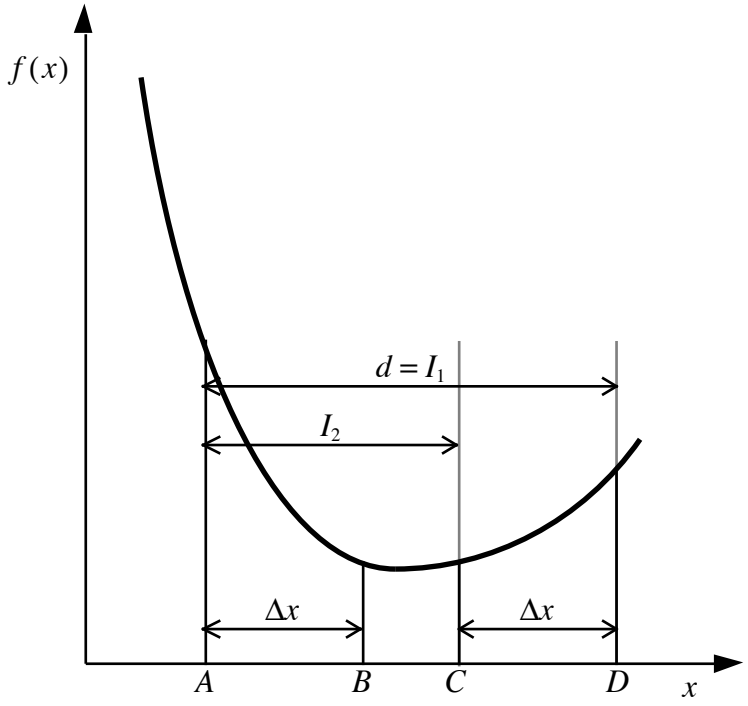
By being a little more subtle, we can increase the rate of convergence of the interval reduction method. This method makes use of the fact that we have already evaluated the function at one point in the new interval.

We position B and C such that

$$\frac{\Delta x}{d-\Delta x} = \frac{d-\Delta x}{d} \text{ , i.e. } \frac{\Delta x}{d} = 0.382 \text{ ,} \tag{5.5}$$

where d and Δx are defined as shown in Figure 5.3. This means that one of the interior points remains an interior point for the new interval.

Figure 5.3: An Example of Golden Section Interval Reduction Line Search.

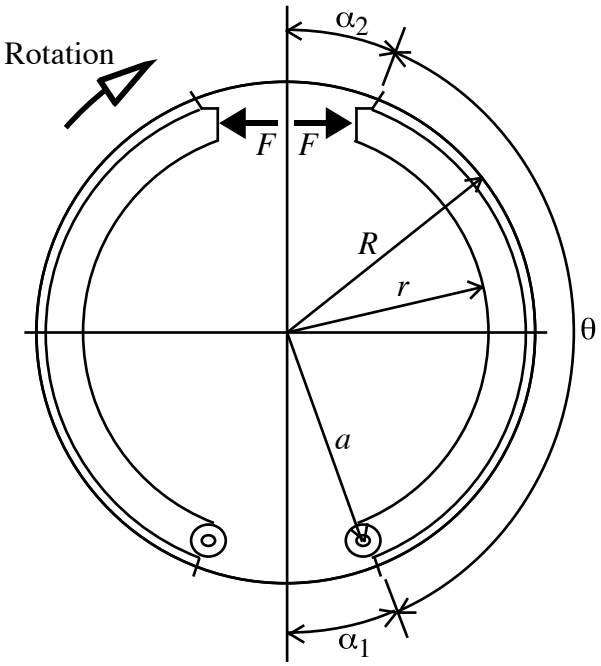


The order of convergence is now

$$p = \left[\frac{I_1}{I_2}\right]^{\frac{1}{1}} = \frac{d}{d-\Delta x} = 1.618 \text{ .} \tag{5.6}$$

This order of convergence p is called the *Golden Section ratio*.

5.2.3 Example: Brake with Internal Expanding Shoes



$\alpha_1 \approx \alpha_2$

$a = 123$ mm

$r = 115$ mm

$R = 150$ mm

$p_a = 1$ MPa (pressure limit)

$\mu = 0.45$

$b = 32$ mm (face width)

Problem: Fixing all other variables, find the brake span θ which minimises the self-actuation from the leading shoe.

Let M_f be the moment of the frictional forces on the leading shoe and M_n be the moment of the normal forces on the leading shoe:

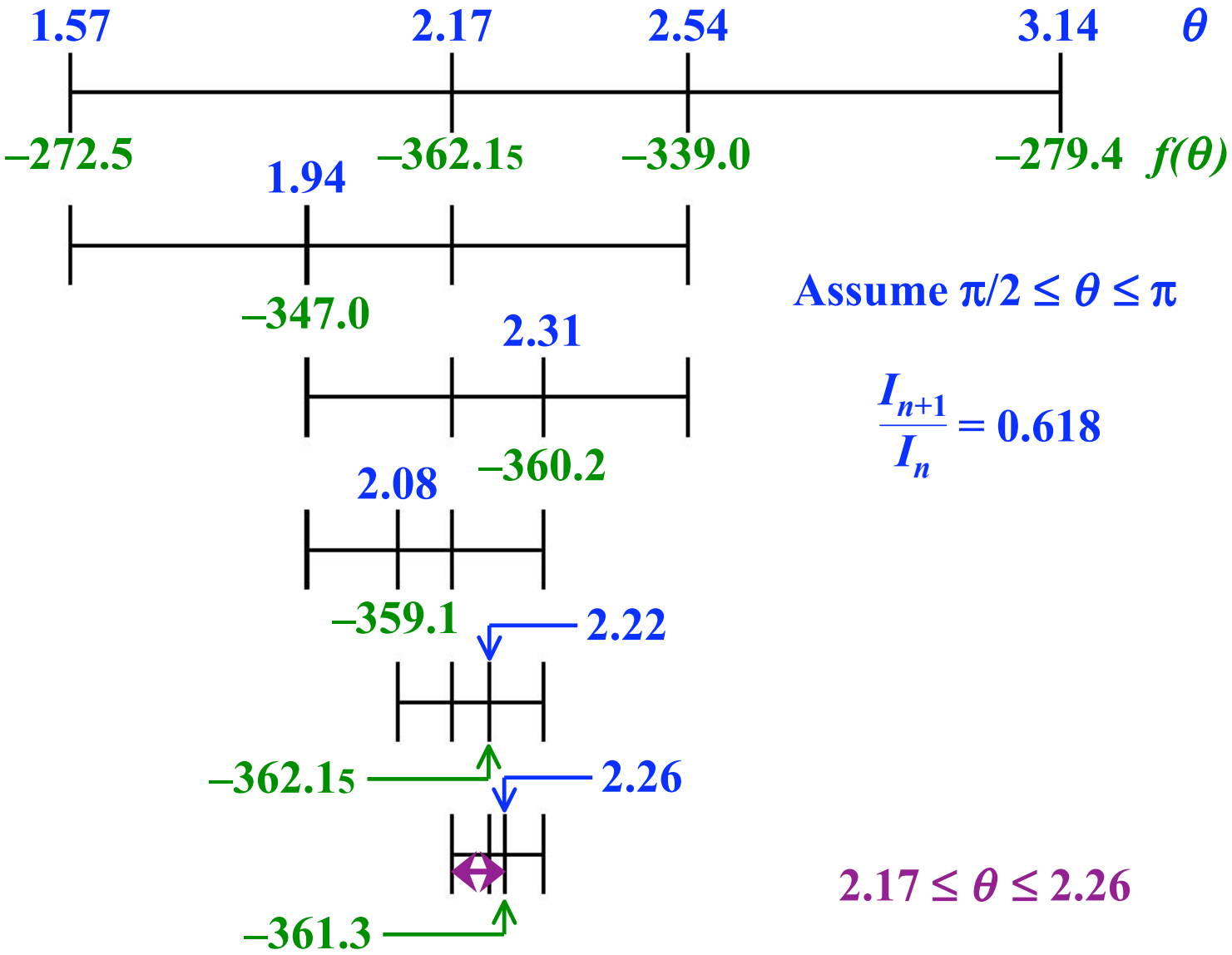
$$M_f = \mu p_a b R \left(R - R\cos\theta - \frac{a}{2}\sin^2\theta \right)$$

$$M_n = p_a b R a \left(\frac{\theta}{2} - \frac{\sin 2\theta}{4} \right)$$

Thus, we need to maximize $(M_n - M_f)$ or minimize $f(\theta) = -(M_n - M_f)$ where, substituting values:

$$f(\theta) = -295.2\theta + 147.8\sin 2\theta - 324.0\cos\theta - 132.8\sin^2\theta + 324.0$$

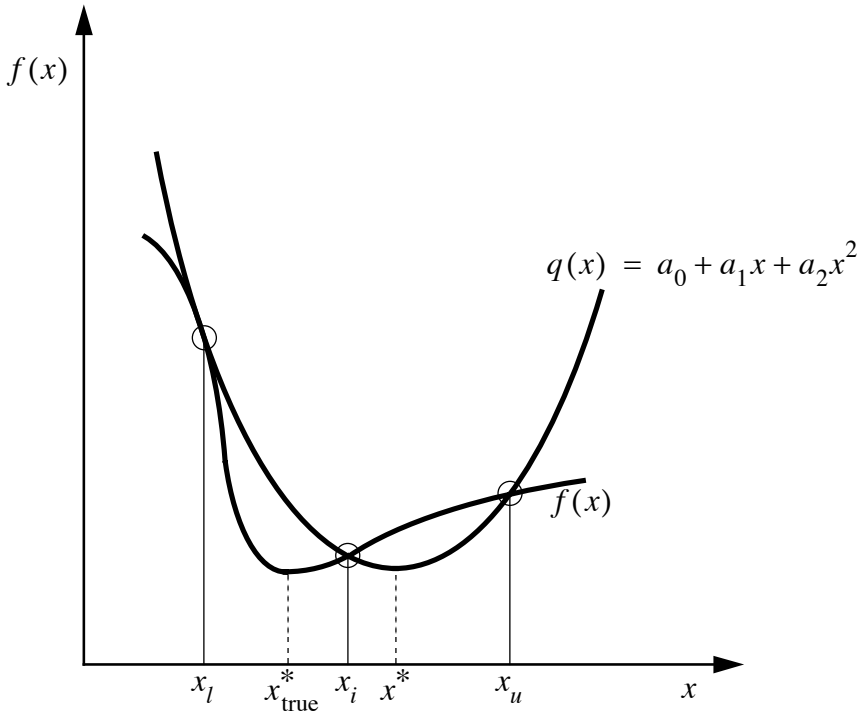
Example: Brake with Internal Expanding Shoes (continued)



5.3 Polynomial Fitting

Any continuous function can be approximated by a polynomial, and then the minimum value can be determined analytically.

It is often sufficient to fit a *quadratic* (second-order polynomial). For this, we need to know the value of the objective function at 3 points.



Let the quadratic be:

$$q(x) = a_0 + a_1 x + a_2 x^2 \,, \tag{5.7}$$

where a_0, a_1 and a_2 are unknown constants. At the three known points, we get:

$$\begin{aligned} f(x_l) &= a_0 + a_1 x_l + a_2 x_l^{\,2} \\ f(x_i) &= a_0 + a_1 x_i + a_2 x_i^{\,2} \\ f(x_u) &= a_0 + a_1 x_u + a_2 x_u^{\,2} \end{aligned} \tag{5.8}$$

Hence,

$$a_2 = \frac{1}{x_u - x_l}\left[\frac{f(x_u) - f(x_l)}{x_u - x_l} - \frac{f(x_i) - f(x_l)}{x_i - x_l}\right] \tag{5.9}$$

$$a_1 = \frac{f(x_i) - f(x_l)}{x_i - x_l} - a_2(x_u - x_i) \tag{5.10}$$

$$a_0 = f(x_l) - a_1 x_l - a_2 x_l^{\,2} \tag{5.11}$$

The minimum of the quadratic equation, x^*, is given by:

$$\frac{\mathrm{d}q}{\mathrm{d}x}(x^*) = 0 \;\Rightarrow\; x^* = -\frac{a_1}{2a_2}\,, \tag{5.12}$$

subject to

$$\frac{\mathrm{d}^2q}{\mathrm{d}x^2}(x^*) > 0 \;\Rightarrow\; a_2 > 0\,. \tag{5.13}$$

5.4 Newton's Method

A further increase in efficiency can be obtained by making use of the first and second derivatives of the function. Such methods are called *second-order methods*, of which the most popular ones are Newton's method and its variants.

The underlying idea of Newton's method is the following. From the Taylor series of $f(x)$:

$$f(x) = f(\hat{x}) + (x-\hat{x})f'(\hat{x}) + \frac{1}{2}(x-\hat{x})^2 f''(\hat{x}) + R\,, \tag{5.14}$$

approximate $f(x)$ as a quadratic, i.e. let $R = 0$.

Differentiate with respect to x to get

$$f'(x) = f'(\hat{x}) + (x-\hat{x})f''(\hat{x})\,. \tag{5.15}$$

If x is the location of the minimum of $f(x)$, then $f'(x) = 0$, and therefore

$$x = \hat{x} - \frac{f'(\hat{x})}{f''(\hat{x})}\,. \tag{5.16}$$

Let x_{k+1} be a candidate for the minimum (x_k being the previous guess), then $f'(x_{k+1}) = 0$, and hence, identifying x_{k+1} with x and x_k with $\hat{x}$:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}\,, \tag{5.17}$$

provided $f''(x_k) \neq 0$.

This can then be used as the basis for an iterative procedure, making x_{k+1} the new value of x_k. This iteration will refine the estimate of the minimum — unless the function is a quadratic, in which case the minimum is found immediately.

There are several variations of Newton's method, but as they apply to functions of any dimension, they will be discussed in detail in the next section on multidimensional search methods.

6. Multidimensional Search Methods

Multidimensional searches can be divided into three types:

- *Random Search*: Step direction and length are chosen at random; inefficient but given time it should eventually find the answer — like monkeys trying to type Shakespeare.
- *Direct Search*: No attempt is made to evaluate the local gradient.
- *Gradient Search*: The local gradient (and possibly the Hessian) is evaluated or estimated.

6.1 Direct Search Methods

Direct search methods can be described as *systematic trial and error* methods. They are mostly used when

- the analytical relationship between the control variables and $f(\mathbf{x})$ is not known, but $f(\mathbf{x})$ can be evaluated experimentally at individual points (e.g. in industrial processes);
- $f(\mathbf{x})$ is known but gradient information cannot be obtained in closed form.

These methods will be covered in Module 4M17.

6.2 Gradient Search Methods

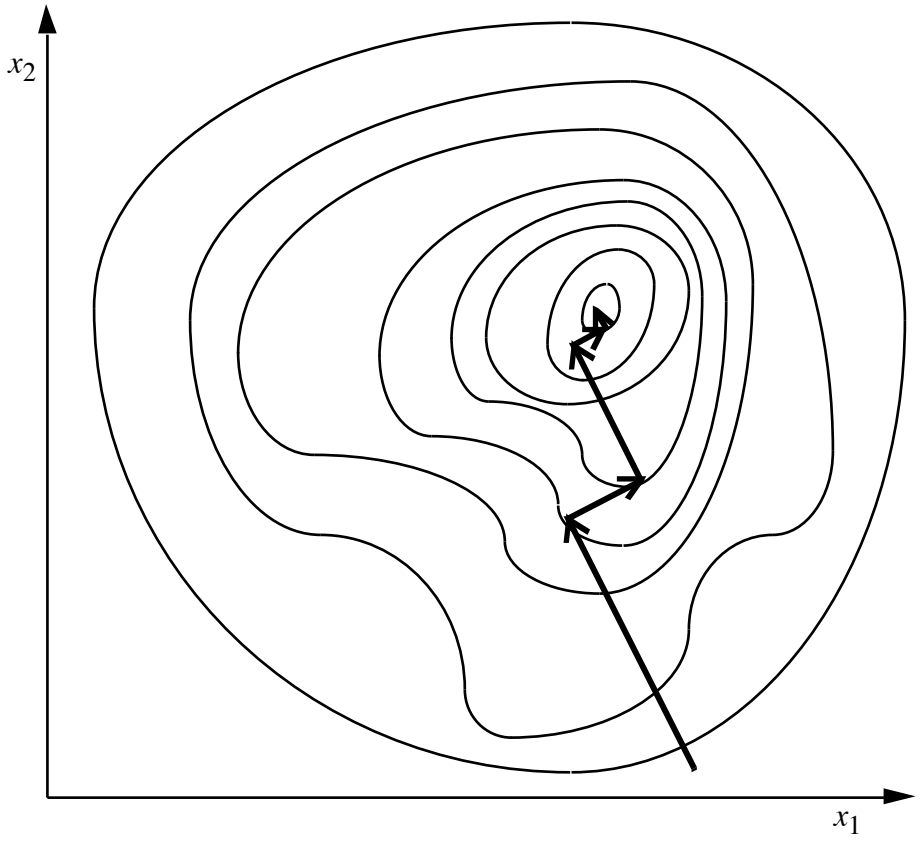
These methods involve evaluating the local gradient to determine the search direction (not necessarily the gradient itself), and then doing a line search to determine the minimum value of the function along that line. At this new point, the local gradient is evaluated again, and the process is repeated.

6.2.1 Method of Steepest Descent

The Steepest Descent method was devised by Cauchy in 1847. It involves evaluating the *gradient* at each point and then doing a line search in that direction to minimize the function. Each successive search direction is orthogonal to the previous one.

Although this appears sensible, it has, in fact, a slow rate of convergence (see below).

Figure 6.1: An Example of Steepest Descent Search.



Method:

1. Select a starting point $\mathbf{x}_0$.
2. Determine the search direction $\mathbf{d}_k$ which is the gradient at $\mathbf{x}_k$, i.e. $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$ (negative since *descent*).
3. Determine the step size α_k along $\mathbf{d}_k$ to minimize the function.
4. Update the estimate of the minimum $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$.
5. Check for convergence. If not converged, go to step 2.

To determine the step size α_k either use a line search or start with the Taylor series of the function

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) = f(\mathbf{x}_k) + \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k + \frac{\alpha_k^2}{2} \mathbf{d}_k^T \boldsymbol{H}(\mathbf{x}_k) \mathbf{d}_k + R \ . \tag{6.1}$$

Differentiating with respect to α_k and neglecting R gives

$$\frac{\partial f}{\partial \alpha_k}(\mathbf{x}_k + \alpha_k \mathbf{d}_k) = \nabla f(\mathbf{x}_k)^T \mathbf{d}_k + \alpha_k \mathbf{d}_k^T \boldsymbol{H}(\mathbf{x}_k) \mathbf{d}_k \ , \tag{6.2}$$

so that the minimum (when this expression equals 0) is when

$$\alpha_k = -\frac{\nabla f(\mathbf{x}_k)^T \mathbf{d}_k}{\mathbf{d}_k^T \boldsymbol{H}(\mathbf{x}_k)\mathbf{d}_k} = \frac{\mathbf{d}_k^T \mathbf{d}_k}{\mathbf{d}_k^T \boldsymbol{H}(\mathbf{x}_k)\mathbf{d}_k} . \tag{6.3}$$

This value of α_k thus gives the distance to the point where the gradient of the function is perpendicular to the line of search.

It should be noted that if the Hessian $\boldsymbol{H}$ is available, then a number of search methods which are superior to the Steepest Descent method can be used.

6.2.2 Convergence of the Steepest Descent Method

The method of steepest descent gives a sequence of objective values $f(\mathbf{x}_k)$ which converges *linearly* to $f(\mathbf{x}^*)$ (i.e. the order of convergence p is approximately one).

Since most methods have linear convergence, we must look at the convergence ratio β to compare them.

Linear convergence implies

$$\left\| \mathbf{x}_{k+1} - \mathbf{x}^* \right\| = \beta \left\| \mathbf{x}_k - \mathbf{x}^* \right\| . \tag{6.4}$$

In the case of the Steepest Descent method, the value of β is given by

$$\beta \leq \left[\frac{A-a}{A+a} \right]^2 , \tag{6.5}$$

where $A > 0$ and $a > 0$ are the largest and smallest eigenvalues of $\boldsymbol{H}(\mathbf{x}^*)$.

Thus, the convergence of the Steepest Descent method is best when the largest and smallest eigenvalues of $\boldsymbol{H}(\mathbf{x}^*)$ are both large and close in value.

6.2.3 Example: Steepest Descent

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

with the convergence condition $\varepsilon = \left| \nabla f(\mathbf{x}^*) \right| < 0.005$.

$$\nabla f = \begin{bmatrix} 2x_1 + 2x_2 \\ 4x_2 + 2x_1 + 2x_3 \\ 4x_3 + 2x_2 \end{bmatrix}$$

$$\underline{\underline{H}} = \begin{bmatrix} 2 & 2 & 0 \\ 2 & 4 & 2 \\ 0 & 2 & 4 \end{bmatrix}$$

constant since $f(x)$ is a second-order polynomial

Let $\underline{x}_0 = (2 \quad 4 \quad 10)^T$

1st iteration $\nabla f(\underline{x}_0) = (12 \quad 40 \quad 48)^T$

$\therefore \underline{d}_0 = -\nabla f(\underline{x}_0) = (-12 \quad -40 \quad -48)^T$

$\underline{\underline{H}}(\underline{x}_0) = \underline{\underline{H}}$

$$\alpha_0 = \frac{\underline{d}_0^T \underline{d}_0}{\underline{d}_0^T \underline{\underline{H}} \underline{d}_0} = \frac{4048}{25504} = 0.159$$

$$\underline{x}_1 = \underline{x}_0 + \alpha_0 \underline{d}_0$$
$$= (2 \quad 4 \quad 10)^T + 0.159\,(-12 \quad -40 \quad -48)^T$$
$$= (0.096 \quad -2.35 \quad 2.38)^T$$

$\nabla f(\underline{x}_1) = (-4.50 \quad -4.44 \quad 4.83)^T$

$|\nabla f(\underline{x}_1)| = 7.95$ i.e. not yet converged

2nd iteration $\underline{d}_1 = -\nabla f(\underline{x}_1) = (4.50 \quad 4.44 \quad -4.83)^T$

$$\alpha_1 = \frac{\underline{d}_1^T \underline{d}_1}{\underline{d}_1^T \underline{\underline{H}} \underline{d}_1} = 0.305$$

$\underline{x}_2 = \underline{x}_1 + \alpha_1 \underline{d}_1 = (1.47 \quad -0.99 \quad 0.91)^T$

$|\nabla f(\underline{x}_2)| = 2.06$ i.e. not yet converged

Takes 33 iterations for convergence

Reason for slow convergence

$$|\underline{\underline{H}} - \lambda \underline{\underline{I}}| = 0$$

$$\Rightarrow \lambda^3 - 14\lambda^2 + 28\lambda - 8 = 0$$

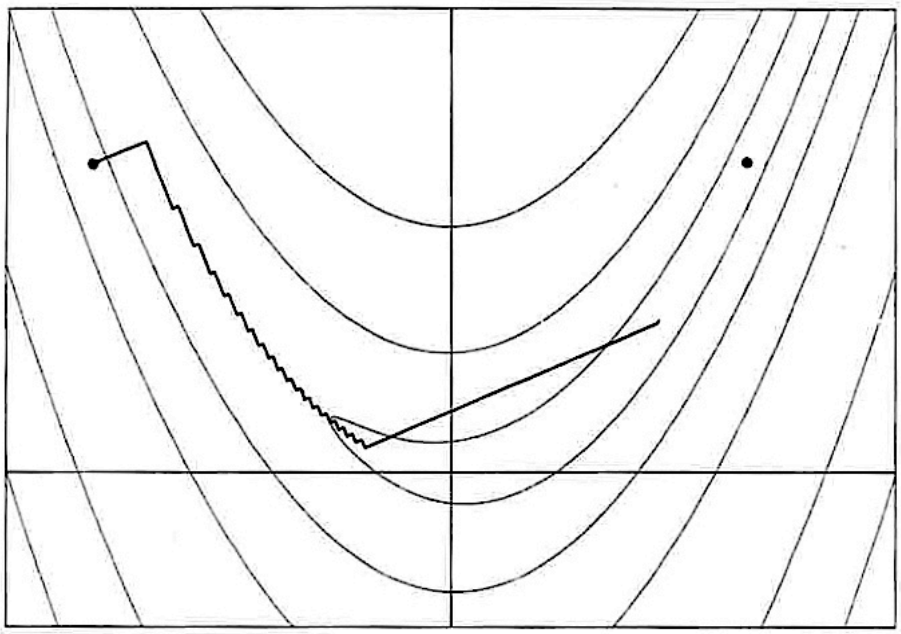
$$\Rightarrow \lambda = 6.49, 3.10, 0.39$$

$$\beta \leq \left[\frac{A-a}{A+a}\right]^2 = \left[\frac{6.49-0.39}{6.49+0.39}\right]^2$$

$$= 0.79$$

6.2.4 Example: Rosenbrock's Function $f(\mathbf{x}) = 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2$

Figure 6.2: Solution Path of a Steepest Descent Algorithm on Rosenbrock's Function.



The linear segments of Figure 6.2 correspond to the step taken within a given iteration. Note that the algorithm would have failed in the vicinity of the point (−0.3, 0.1) but for the fact that the line search found (by chance) the *second* minimum along the search direction. Several hundred iterations were performed close to the new point without any perceptible change in the objective function.

6.2.5 Newton's Method

Direct search methods simply evaluate the function and make no use of the gradient information. The Steepest Descent method evaluates the local gradient to determine the search direction (a first-order method), but takes no account of the rate at which the gradient is changing. Newton's method makes use of this *second-order* information.

Suppose that we want to minimize $f(\mathbf{x})$ and that, at a point $\mathbf{x}_k$, it is possible to evaluate $f(\mathbf{x}_k)$, $\nabla f(\mathbf{x}_k)$ and $\boldsymbol{H}(\mathbf{x}_k)$.

We can then write the Taylor series of $f(\mathbf{x}_{k+1})$ as

$$f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T (\mathbf{x}_{k+1} - \mathbf{x}_k) + \frac{1}{2} (\mathbf{x}_{k+1} - \mathbf{x}_k)^T \boldsymbol{H}(\mathbf{x}_k) (\mathbf{x}_{k+1} - \mathbf{x}_k) + R \,, \quad (6.6)$$

and approximate f as a quadratic function by setting $R = 0$.

Differentiating with respect to $\mathbf{x}_{k+1}$,

$$\nabla f(\mathbf{x}_{k+1}) = \nabla f(\mathbf{x}_k) + \boldsymbol{H}(\mathbf{x}_k) (\mathbf{x}_{k+1} - \mathbf{x}_k) \,. \quad (6.7)$$

The minimum is reached at $\mathbf{x}_{k+1}$ when $\nabla f(\mathbf{x}_{k+1}) = 0$, i.e.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \boldsymbol{H}(\mathbf{x}_k)^{-1} \nabla f(\mathbf{x}_k)$$

or

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \ , \tag{6.8}$$

where

$$\alpha_k = 1 \ \text{ and } \ \mathbf{d}_k = -\boldsymbol{H}(\mathbf{x}_k)^{-1} \nabla f(\mathbf{x}_k) \ . \tag{6.9}$$

If $\boldsymbol{H}$ is positive definite and f is quadratic, only one iteration is, of course, required to reach the minimum from any starting point (note that the step length α_k is unity). Therefore, we expect good convergence from Newton's method when the function is closely approximated by a quadratic.

6.2.6 Convergence of Newton's Method

For a general nonlinear function f, Newton's method converges *quadratically* to $\mathbf{x}^*$ (i.e. the order of convergence p is approximately two) provided $\mathbf{x}_0$ is sufficiently close to $\mathbf{x}^*$, $\boldsymbol{H}(\mathbf{x}^*)$ is positive definite and the step lengths $\{\alpha_k\}$ converge to unity (which is always the case for the basic Newton's method). That is

$$\left\| \mathbf{x}_{k+1} - \mathbf{x}^* \right\| \leq \beta \left\| \mathbf{x}_k - \mathbf{x}^* \right\|^2 \ , \tag{6.10}$$

where β is the (constant) convergence ratio.

The local properties of Newton's method make it a very attractive algorithm for unconstrained optimization. In fact, it is often regarded as the standard against which other algorithms are measured.

However, difficulties, even failure, may occur if the quadratic model is a poor approximation to f near the current point. The search direction found at each iteration $\mathbf{d}_k$ will always be a descent direction, but, because the step length α_k is unity, if f is not well modelled by a quadratic, it is possible that the search will badly overshoot the minimum. At the next iteration the search may overshoot again coming back, and, thus, Newton's Method can end up oscillating indefinitely rather than converging.

6.2.7 Newton-Raphson Method

One common modification of Newton's method is therefore to choose

$$\mathbf{d}_k = -\boldsymbol{H}(\mathbf{x}_k)^{-1} \nabla f(\mathbf{x}_k) \tag{6.11}$$

and

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \tag{6.12}$$

where α_k is the step size which minimizes $f(\mathbf{x}_{k+1})$, which is found using a line search.

This modification has a number of advantages over the basic Newton method which make it worth the comparatively small additional effort.

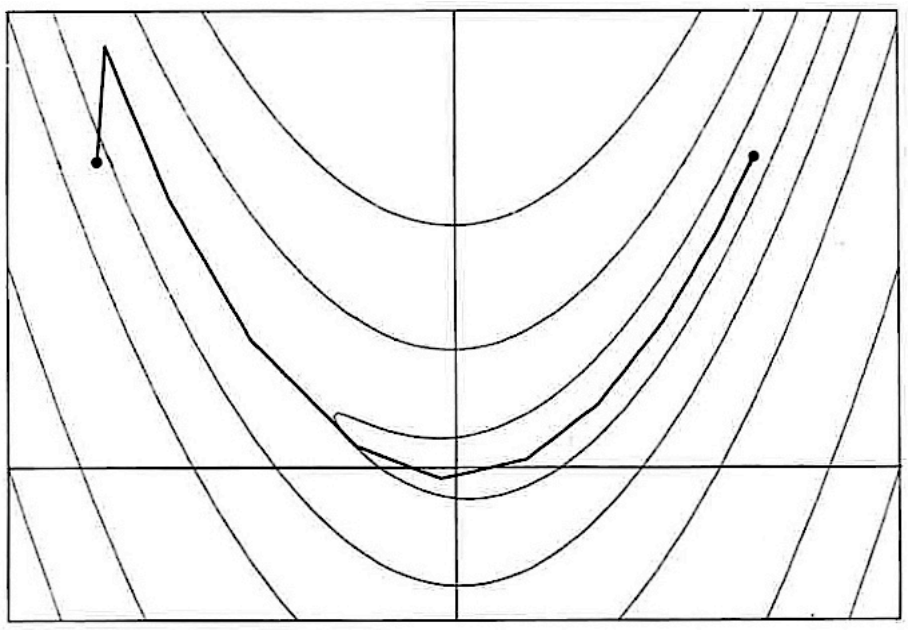
It will usually

- speed up the convergence (it will remain of order two but will have a smaller convergence ratio β, but only if the steps $\{\alpha_k\}$ converge sufficiently fast to unity) — it can sometimes even secure convergence when the basic method diverges;
- avoid convergence to a saddle point or maximum, i.e. the method is tailored for minimization problems.

However, a major disadvantage of all Newton methods is that it may be difficult or indeed impossible to compute the Hessian at each step. Also, inverting the Hessian is usually a major undertaking, which is most impractical for large or complicated problems.

6.2.8 Example: Rosenbrock's Function $f(\mathbf{x}) = 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2$

Figure 6.3: Solution Path of a Modified Newton Algorithm on Rosenbrock's Function.



Except for the first iteration, the method follows the base of the valley in an almost 'optimal' number of steps.

6.2.9 Conjugate Gradient Method

In the Steepest Descent method, consecutive steps are orthogonal. In the Conjugate Gradient method, consecutive steps follow *conjugate* directions, i.e. $\mathbf{d}_i^T \boldsymbol{H} \mathbf{d}_j = 0 \quad \forall\ i \neq j$, where $\boldsymbol{H}$ is the Hessian.

If the objective function is quadratic, the solution will be found after a number of iterations equal to the number of variables in the function. For higher-order functions, more steps may be needed since the method is essentially fitting a quadratic to the function at each stage.

The first step is the same as for the Steepest Descent method, but subsequent search directions are different — they 'remember' a bit of the previous direction.

This means that the directions tend to cut diagonally through the orthogonal steepest descent directions. Thus, they improve considerably the rate of convergence.

The search direction for $k > 0$ is defined as

$$\mathbf{d}_{k+1} = -\nabla f(\mathbf{x}_{k+1}) + \beta_k \mathbf{d}_k \,, \tag{6.14}$$

where

$$\beta_k = \left[\frac{\left| \nabla f(\mathbf{x}_{k+1}) \right|}{\left| \nabla f(\mathbf{x}_k) \right|} \right]^2 , \tag{6.15}$$

with step size α_k chosen to minimize $f(\mathbf{x}_{k+1})$ in the search direction (as in the Steepest Descent method), i.e.

$$\alpha_k = -\frac{\nabla f(\mathbf{x}_k)^T \mathbf{d}_k}{\mathbf{d}_k^T \boldsymbol{H}(\mathbf{x}_k) \mathbf{d}_k} \,. \tag{6.16}$$

Conjugate Gradient Method:

1. Choose $\mathbf{x}_0$, and compute $\mathbf{d}_0 = -\nabla f(\mathbf{x}_0)$, $\boldsymbol{H}(\mathbf{x}_0)$ and $\alpha_0 = \dfrac{\mathbf{d}_0^T \mathbf{d}_0}{\mathbf{d}_0^T \boldsymbol{H}(\mathbf{x}_0)\, \mathbf{d}_0}$.

2. Determine $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$, $\nabla f(\mathbf{x}_{k+1})$ and $\boldsymbol{H}(\mathbf{x}_{k+1})$.

3. Evaluate
$$\beta_k = \left[\frac{\left| \nabla f(\mathbf{x}_{k+1}) \right|}{\left| \nabla f(\mathbf{x}_k) \right|} \right]^2$$
$$\mathbf{d}_{k+1} = -\nabla f(\mathbf{x}_{k+1}) + \beta_k \mathbf{d}_k$$
$$\alpha_{k+1} = -\frac{\mathbf{d}_{k+1}^T \nabla f(\mathbf{x}_{k+1})}{\mathbf{d}_{k+1}^T \boldsymbol{H}(\mathbf{x}_{k+1})\, \mathbf{d}_{k+1}}$$

4. Go to step 2, until the algorithm has converged.

It is quite common for implementations of the Conjugate Gradient method to restart periodically (in particular every n iterations, where n is the number of control variables), i.e. to follow the steepest descent direction and then successively compute new conjugate directions. The rationale for this strategy is as follows:

Because the Conjugate Gradient method will converge after n iterations if the objective function is quadratic (and therefore the Hessian is constant), if the method has not converged after n iterations, the objective function is not quadratic and the Hessian is not constant. If the Hessian is not constant, then successive search directions are only approximately conjugate (see the next section for more details). By restarting periodically, memory of former search directions (which are no longer conjugate) is effectively erased, and new search directions reflecting more accurately the behaviour of the objective function around the current search location can be determined.

Conjugate Gradient Method with Restarts:

1. Choose $\mathbf{x}_0$, and compute $\mathbf{d}_0 = -\nabla f(\mathbf{x}_0)$, $\boldsymbol{H}(\mathbf{x}_0)$ and $\alpha_0 = \dfrac{\mathbf{d}_0^T \mathbf{d}_0}{\mathbf{d}_0^T \boldsymbol{H}(\mathbf{x}_0)\, \mathbf{d}_0}$.

2. Determine $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$, $\nabla f(\mathbf{x}_{k+1})$ and $\boldsymbol{H}(\mathbf{x}_{k+1})$.

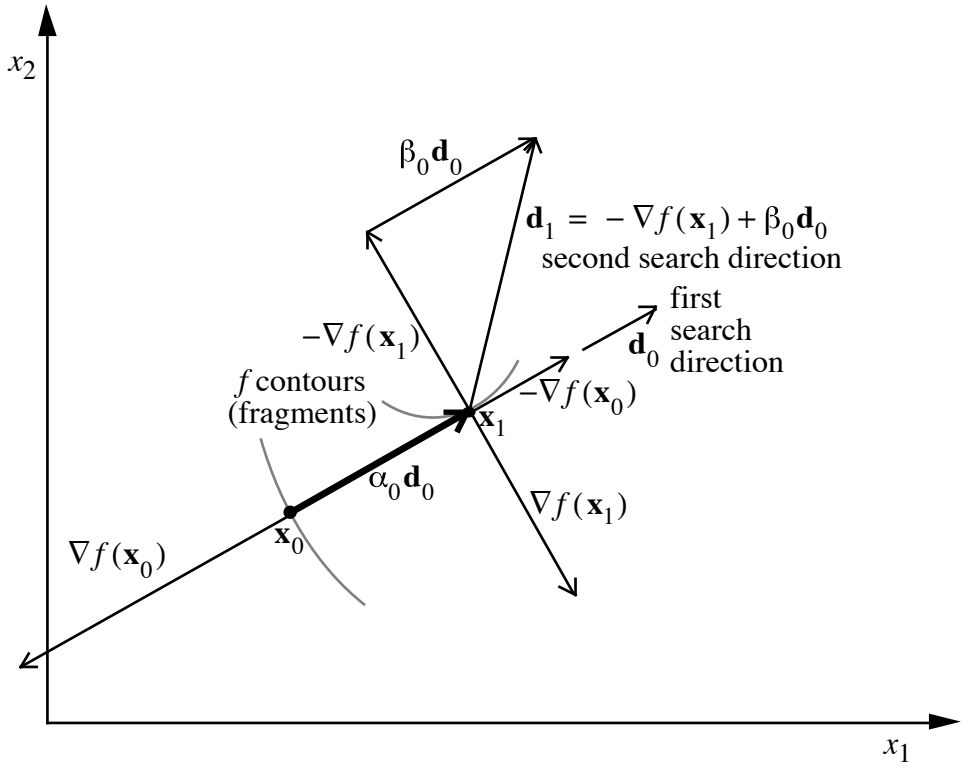
3. If restarting, compute $\mathbf{d}_{k+1} = -\nabla f(\mathbf{x}_{k+1})$ and $\alpha_{k+1} = \dfrac{\mathbf{d}_{k+1}^T \mathbf{d}_{k+1}}{\mathbf{d}_{k+1}^T \boldsymbol{H}(\mathbf{x}_{k+1})\, \mathbf{d}_{k+1}}$,

   else evaluate
   $$\beta_k = \left[\frac{\left| \nabla f(\mathbf{x}_{k+1}) \right|}{\left| \nabla f(\mathbf{x}_k) \right|} \right]^2$$
   $$\mathbf{d}_{k+1} = -\nabla f(\mathbf{x}_{k+1}) + \beta_k \mathbf{d}_k$$
   $$\alpha_{k+1} = -\frac{\mathbf{d}_{k+1}^T \nabla f(\mathbf{x}_{k+1})}{\mathbf{d}_{k+1}^T \boldsymbol{H}(\mathbf{x}_{k+1})\, \mathbf{d}_{k+1}}$$

4. Go to step 2, until the algorithm has converged.

$$\beta_k = \frac{\nabla f(\mathbf{x}_{k+1})^T \boldsymbol{Q}\, \mathbf{d}_k}{\mathbf{d}_k^T \boldsymbol{Q}\, \mathbf{d}_k} \,. \tag{6.17}$$

Figure 6.4: Two Variable Conjugate Gradient Search Pattern.



6.2.10 Convergence of the Conjugate Gradient Method

The conjugate direction is nothing but a deflected steepest descent direction, which improves substantially the rate of convergence.

As for the Steepest Descent method, the Conjugate Gradient method converges *linearly*, i.e.

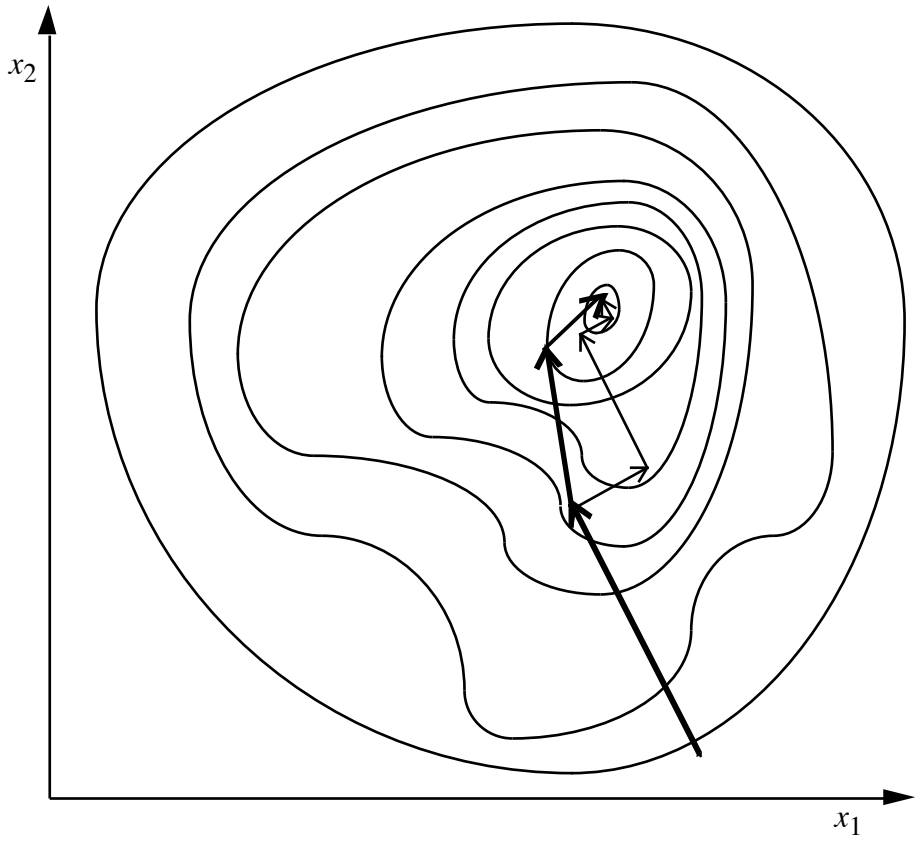
$$\left\| \mathbf{x}_{k+1} - \mathbf{x}^* \right\| = \beta \left\| \mathbf{x}_k - \mathbf{x}^* \right\|, \tag{6.18}$$

with, in theory, convergence ratios near zero (i.e. $\beta \approx 0$, called *superlinear convergence*).

In practice, however, numerical errors in the computation tend to make $\beta > 0$. Nevertheless, this method always yields faster convergence than the Steepest Descent method.

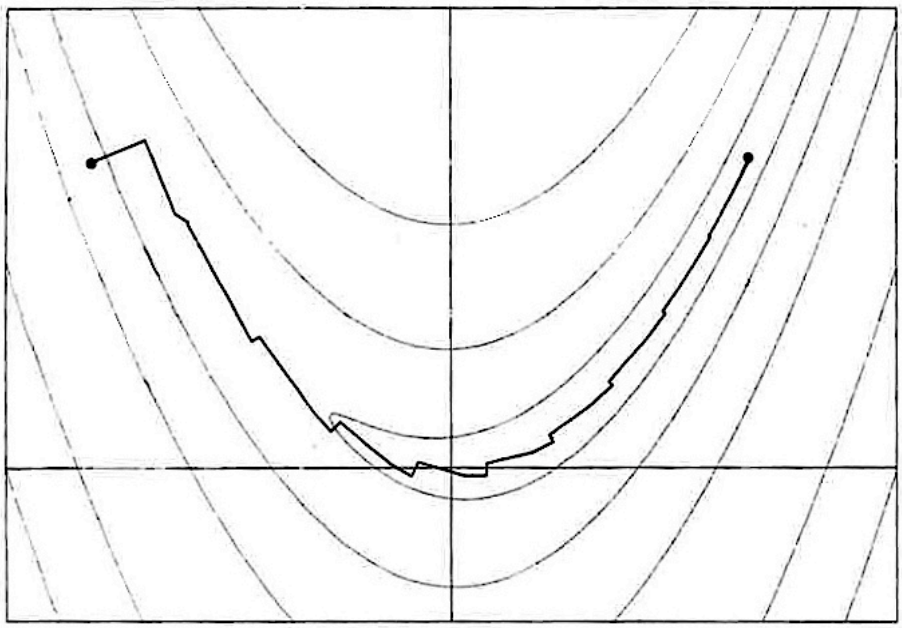
The major advantage of the Conjugate Gradient method over Newton methods is that the former does not require the Hessian to be inverted.

Figure 6.5: An Example of Conjugate Gradient Search.



6.2.11 Example: Rosenbrock's Function $f(\mathbf{x}) = 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2$

Figure 6.6: Solution Path of a Conjugate Gradient Algorithm on Rosenbrock's Function.



In this case the Conjugate Gradient algorithm was restarted every two iterations. Although the method is not intended for problems with so few control variables, Figure 6.6 illustrates the cyclic nature of the traditional Conjugate Gradient method on problems which are far from quadratic.

6.2.12 Example: Conjugate Gradient

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

with the convergence condition $\varepsilon = |\nabla f(\mathbf{x}^*)| < 0.005$.

Conjugate Gradient example

Iteration 1 : same as for Steepest Descent example
(see page 25)

$$\therefore \underline{x}_1 = (0.096 \quad -2.35 \quad 2.38)^T$$

$$\nabla f(\underline{x}_1) = (-4.50 \quad -4.44 \quad 4.83) \quad \text{again}$$

Iteration 2 : $\beta_0 = \left(\frac{|\nabla f(\underline{x}_1)|}{|\nabla f(\underline{x}_0)|} \right)^2 = \left(\frac{7.95}{63.3} \right)^2 = 0.0156$

$$\underline{d}_1 = -\nabla f(\underline{x}_1) + \beta_0 \underline{d}_0$$
$$= (4.50 \quad 4.44 \quad -4.83)^T + 0.0156 (-12 \quad -40 \quad -48)^T$$
$$= (4.31 \quad 3.81 \quad -5.58)^T$$

$$\alpha_1 = -\frac{\nabla f(\underline{x}_1)^T \underline{d}_1}{\underline{d}_1^T \underline{\underline{H}} \underline{d}_1} = 0.316$$

$$\underline{x}_2 = \underline{x}_1 + \alpha_1 \underline{d}_1 = (1.46 \quad -1.15 \quad 0.62)^T$$

$$\nabla f(\underline{x}_2) = (0.62 \quad -0.43 \quad 0.19)^T$$
$$|\nabla f(\underline{x}_2)| = 0.78 \quad \text{i.e. not converged}$$

Iteration 3 : $\beta_1 = \left(\frac{|\nabla f(\underline{x}_2)|}{|\nabla f(\underline{x}_1)|} \right)^2 = \left(\frac{0.78}{7.95} \right)^2 = 0.0095$

$$\underline{d}_2 = -\nabla f(\underline{x}_2) + \beta_1 \underline{d}_1 = (-0.58 \quad 0.46 \quad -0.25)^T$$

$$\alpha_2 = -\frac{\nabla f(\underline{x}_2)^T \underline{d}_2}{\underline{d}_2^T \underline{\underline{H}} \underline{d}_2} = 2.44$$

$$\underline{x}_3 = \underline{x}_2 + \alpha_2 \underline{d}_2 = (0 \quad 0 \quad 0)^T$$

$$\nabla f(\underline{x}_3) = (0 \quad 0 \quad 0)^T$$
$$|\nabla f(\underline{x}_3)| = 0.0 \quad \text{i.e. converged!}$$

in 3 iterations, as expected

Example: Conjugate Gradient (continued)

6.2.13 Gradient Search Methods Summary

Method	Advantages	Disadvantages
Steepest Descent	• Only need to evaluate or estimate the gradient	• Order of convergence is only linear • Convergence can be very slow if convergence ratio is high
Newton's Method	• Quadratic convergence	• Can oscillate indefinitely if objective function is not well modelled by a quadratic • Need to compute and invert Hessian
Newton Raphson Method	• Quadratic convergence • More reliable convergence than Newton Method • Tailored to find minima	• Need to compute and invert Hessian
Conjugate Gradient Method	• Faster than Steepest Descent method • No need to invert Hessian	• Need to compute Hessian • Order of convergence is only linear — though convergence ratio is low

6.3 Nonlinear Least Squares

This class of methods is used for fitting hypothesized models to data.

For example, let $y(t_i)$, $i = 1, 2, \ldots, m$ be the measured data and $\phi(t, \mathbf{x})$ be the model, where the independent parameters $\mathbf{x}^T = [x_1, \ldots, x_n]$ are to be manipulated in order to adjust the model to the data.

(We are assuming that the number of data points m is much larger than the number of model parameters n.)

The *residual* is given by

$$r_i(\mathbf{x}) = \phi(t_i, \mathbf{x}) - y(t_i), \tag{6.19}$$

so that the squared error is

$$f(\mathbf{x}) = \sum_{i=1}^{m} r_i^2(\mathbf{x}) = \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x}) \tag{6.20}$$

where $\mathbf{r}(\mathbf{x})^T = [\, r_1(\mathbf{x}), \ldots, r_m(\mathbf{x}) \,]$.

The nonlinear least-squares problem is to minimize $f(\mathbf{x})$.

Although $f(\mathbf{x})$ can be minimized by a general unconstrained method, in most circumstances it is worthwhile to use methods designed specifically for least-squares problems. In particular, it is useful to make use of the special structure of the gradient and Hessian matrix of $f(\mathbf{x})$.

The gradient of $f(\mathbf{x})$ can be expressed as

$$\nabla f(\mathbf{x}) = 2 \sum_{i=1}^{m} r_i(\mathbf{x}) \nabla r_i(\mathbf{x}) = 2 \boldsymbol{J}(\mathbf{x})^T \mathbf{r}(\mathbf{x}), \tag{6.21}$$

where $\boldsymbol{J}(\mathbf{x})$ is the *Jacobian* matrix of $\mathbf{r}(\mathbf{x})$

$$\boldsymbol{J}(\mathbf{x}) = \begin{bmatrix} \nabla r_1(\mathbf{x})^T \\ \vdots \\ \nabla r_m(\mathbf{x})^T \end{bmatrix} = \begin{bmatrix} \dfrac{\partial r_1}{\partial x_1} & \cdots & \dfrac{\partial r_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial r_m}{\partial x_1} & \cdots & \dfrac{\partial r_m}{\partial x_n} \end{bmatrix}. \tag{6.22}$$

The Hessian matrix $\boldsymbol{H}(\mathbf{x})$ of $f(\mathbf{x})$ is given by

$$\boldsymbol{H}(\mathbf{x}) = \nabla(\nabla f(\mathbf{x})) = 2 \boldsymbol{J}(\mathbf{x})^T \boldsymbol{J}(\mathbf{x}) + 2 \sum_{i=1}^{m} r_i(\mathbf{x}) \boldsymbol{R}_i(\mathbf{x}), \tag{6.23}$$

where $\boldsymbol{R}_i(\mathbf{x})$ is the Hessian of $r_i(\mathbf{x})$, i.e. $\boldsymbol{R}_i(\mathbf{x}) = \nabla(\nabla r_i(\mathbf{x}))$.

6.3.1 Gauss-Newton Method for Least Squares

The Gauss-Newton method is usually the best choice for nonlinear least squares problems. It is based on the premise that, for data fitting, the residuals $r_i(\mathbf{x}^*)$ are small, so that

$$\boldsymbol{H}(\mathbf{x}) \approx 2\boldsymbol{J}(\mathbf{x})^T\boldsymbol{J}(\mathbf{x}) \tag{6.24}$$

for $\mathbf{x}$ near $\mathbf{x}^*$. This gives a positive definite Hessian $\boldsymbol{H}(\mathbf{x})$.

Like Newton's method, the Gauss-Newton method uses the search direction

$$\begin{aligned}
\mathbf{d}_k &= -\boldsymbol{H}(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k) \\
&= -[\,2\,\boldsymbol{J}(\mathbf{x}_k)^T\boldsymbol{J}(\mathbf{x}_k)\,]^{-1}\nabla f(\mathbf{x}_k) \\
&= -\frac{1}{2}[\,\boldsymbol{J}(\mathbf{x}_k)^T\boldsymbol{J}(\mathbf{x}_k)\,]^{-1}\,2\boldsymbol{J}(\mathbf{x}_k)^T\mathbf{r}(\mathbf{x}_k) \\
&= -[\,\boldsymbol{J}(\mathbf{x}_k)^T\boldsymbol{J}(\mathbf{x}_k)\,]^{-1}\boldsymbol{J}(\mathbf{x}_k)^T\mathbf{r}(\mathbf{x}_k)
\end{aligned} \tag{6.25}$$

to give the sequence of estimates

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k\ . \tag{6.26}$$

Here equation (6.21) has been used to substitute for $\nabla f(\mathbf{x})$ in equation (6.25).

Rewriting the final form of equation (6.25)

$$\mathbf{d}_k = -\boldsymbol{J}(\mathbf{x}_k)^{-1}\,[\,\boldsymbol{J}(\mathbf{x}_k)^T]^{-1}\,\boldsymbol{J}(\mathbf{x}_k)^T\mathbf{r}(\mathbf{x}_k) = -\boldsymbol{J}(\mathbf{x}_k)^{-1}\,\mathbf{r}(\mathbf{x}_k)$$

$$\therefore\ \boldsymbol{J}(\mathbf{x}_k)\,\mathbf{d}_k = -\mathbf{r}(\mathbf{x}_k)\ . \tag{6.27}$$

So the search direction $\mathbf{d}_k$ corresponds to the solution of the *linear* least squares problem:

$$\text{minimize}\left\|\,\boldsymbol{J}(\mathbf{x}_k)\,\mathbf{d}_k + \mathbf{r}(\mathbf{x}_k)\right\|^2 \tag{6.28}$$

and since linear least squares problems can be solved efficiently using *singular-value decomposition* or *orthogonal factorization*, this provides a computationally more accurate method of obtaining the search direction (and, as there is no need to invert matrices, avoids problems associated with ill-conditioned matrices).

The two main advantages of the Gauss-Newton method are that it only needs first derivatives and that it converges like Newton's method if the residuals $r_i(\mathbf{x})$ are small.