Chapter 2

LINE SEARCH DESCENT METHODS FOR UNCONSTRAINED MINIMIZATION

2.1 General line search descent algorithm for unconstrained minimization

Over the last 40 years many powerful *direct search algorithms* have been developed for the unconstrained minimization of general functions. These algorithms require an initial estimate to the optimum point, denoted by $\mathbf{x}^0$. With this estimate as starting point, the algorithm generates a sequence of estimates $\mathbf{x}^0$, $\mathbf{x}^1$, $\mathbf{x}^2$, ..., by successively searching *directly* from each point in a direction of *descent* to determine the next point. The process is terminated if either no further progress is made, or if a point $\mathbf{x}^k$ is reached (for smooth functions) at which the first necessary condition in (1.24), i.e. $\nabla f(\mathbf{x}) = \mathbf{0}$ is sufficiently accurately satisfied, in which case $\mathbf{x}^* \cong \mathbf{x}^k$. It is usually, although not always, required that the function value at the new iterate $\mathbf{x}^{i+1}$ be lower than that at $\mathbf{x}^i$.

An important sub-class of direct search methods, specifically suitable for smooth functions, are the so-called *line search* descent methods. Basic to these methods is the selection of a descent direction $\mathbf{u}^{i+1}$ at each iterate $\mathbf{x}^i$ that ensures descent at $\mathbf{x}^i$ in the direction $\mathbf{u}^{i+1}$, i.e. it is required that the directional derivative in the direction $\mathbf{u}^{i+1}$ be negative:

$$\left.\frac{df(\mathbf{x}^i)}{d\lambda}\right|_{\mathbf{u}^{i+1}} = \nabla^T f(\mathbf{x}^i)\mathbf{u}^{i+1} < 0. \tag{2.1}$$

The general structure of such descent methods is given below.

2.1.1 General structure of a line search descent method

1. Given starting point $\mathbf{x}^0$ and positive tolerances ε_1, ε_2 and ε_3, set $i = 1$.
2. Select a descent direction $\mathbf{u}^i$ (see descent condition (2.1).
3. Perform a *one-dimensional line search* in direction $\mathbf{u}^i$: i.e.

   $$\min_{\lambda} F(\lambda) = \min_{\lambda} f(\mathbf{x}^{i-1} + \lambda \mathbf{u}^i)$$

   to give minimizer λ_i.
4. Set $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i$.
5. Test for convergence:

   if $\|\mathbf{x}^i - \mathbf{x}^{i-1}\| < \varepsilon_1$, or $\|\nabla f(\mathbf{x}^i)\| < \varepsilon_2$, or $|f(\mathbf{x}^i) - f(\mathbf{x}^{i-1})| < \varepsilon_3$, then STOP and $\mathbf{x}^* \cong \mathbf{x}^i$,

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

In testing for termination in step 5, a combination of the stated termination criteria may be used, i.e. instead of *or*, *and* may be specified. The structure of the above descent algorithm is depicted in Figure 2.1.

Different descent methods, within the above sub-class, differ according to the way in which the descent directions $\mathbf{u}^i$ are chosen. Another important consideration is the method by means of which the one-dimensional line search is performed.

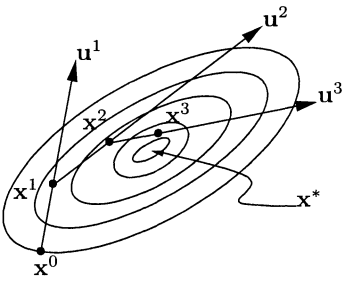

Figure 2.1: Sequence of line search descent directions and steps

2.2 One-dimensional line search

Clearly, in implementing descent algorithms of the above type, the one-dimensional minimization problem:

$$\min_{\lambda} F(\lambda), \ \lambda \in \mathbb{R} \tag{2.2}$$

is an important sub-problem. Here the minimizer is denoted by λ^*, i.e.

$$F(\lambda^*) = \min_{\lambda} F(\lambda).$$

Many one-dimensional minimization techniques have been proposed and developed over the years. These methods differ according to whether they are to be applied to smooth functions or poorly conditioned functions. For smooth functions *interpolation methods*, such as the quadratic interpolation method of Powell (1964) and the cubic interpolation algorithm of Davidon (1959), are the most efficient and accurate methods. For poorly conditioned functions, *bracketing methods*, such as the Fibonacci search method (Kiefer, 1957), which is optimal with respect to the number of function evaluations required for a prescribed accuracy, and the golden section method (Walsh, 1975), which is near optimal but much simpler and easier to implement, are preferred. Here *Powell's quadratic interpolation* method and the *golden section* method, are respectively presented as representative of the two different approaches that may be adopted to one-dimensional minimization.

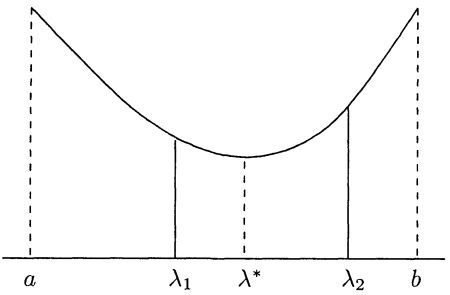

Figure 2.2: Unimodal function $F(\lambda)$ over interval $[a, b]$

2.2.1 Golden section method

It is assumed that $F(\lambda)$ is *unimodal* over the interval $[a, b]$, i.e. that it has a minimum λ^* within the interval and that $F(\lambda)$ is strictly descending for $\lambda < \lambda^*$ and strictly ascending for $\lambda > \lambda^*$, as shown in Figure 2.2.

Note that if $F(\lambda)$ is unimodal over $[a, b]$ with λ^* in $[a, b]$, then to determine a sub-unimodal interval, at least *two* evaluations of $F(\lambda)$ in $[a, b]$ must be made as indicated in Figure 2.2.

If $F(\lambda_2) > F(\lambda_1) \Rightarrow$ new unimodal interval $= [a, \lambda_2]$, and set $b = \lambda_2$ and select new λ_2; otherwise new unimodal interval $= [\lambda_1, b]$ and set $a = \lambda_1$ and select new λ_1.

Thus, the unimodal interval may successively be reduced by inspecting values of $F(\lambda_1)$ and $F(\lambda_2)$ at interior points λ_1 and λ_2.

The question arises: How can λ_1 and λ_2 be chosen in the most economic manner, i.e. such that a least number of function evaluations are required for a prescribed accuracy (i.e. for a specified uncertainty interval)? The most economic method is the Fibonacci search method. It is however a complicated method. A near optimum and more straightforward method is the golden section method. This method is a limiting form of the Fibonacci search method. Use is made of the golden ratio r when selecting the values for λ_1 and λ_2 within the unimodal interval. The value of r corresponds to the positive root of the quadratic equation: $r^2 + r - 1 = 0$, thus $r = \frac{\sqrt{5}-1}{2} = 0.618034$.

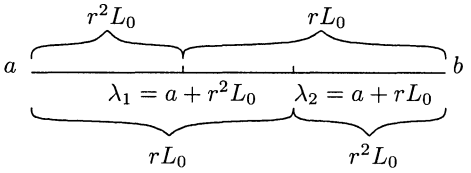

Figure 2.3: Selection of interior points λ_1 and λ_2 for golden section search

The details of the selection procedure are as follows. Given initial unimodal interval $[a, b]$ of length L_0, then choose interior points λ_1 and λ_2 as shown in Figure 2.3.

Then, if $F(\lambda_1) > F(\lambda_2) \Rightarrow$ new $[a, b] = [\lambda_1, b]$ with new interval length $L_1 = rL_0$, and

if $F(\lambda_2) > F(\lambda_1) \Rightarrow$ new $[a, b] = [a, \lambda_2]$ also with $L_1 = rL_0$.

The detailed formal algorithm is stated below.

2.2.1.1 Basic golden section algorithm

Given interval $[a, b]$ and prescribed accuracy ε; then set $i = 0$; $L_0 = b - a$, and perform the following steps:

1. Set $\lambda_1 = a + r^2L_0$; $\lambda_2 = a + rL_0$.
2. Compute $F(\lambda_1)$ and $F(\lambda_2)$; set $i = i + 1$.
3. *If* $F(\lambda_1) > F(\lambda_2)$ then
   set $a = \lambda_1$; $\lambda_1 = \lambda_2$; $L_i = (b - a)$; and $\lambda_2 = a + rL_i$,
   else
   set $b = \lambda_2$; $\lambda_2 = \lambda_1$; $L_i = (b - a)$; and $\lambda_1 = a + r^2L_i$.
4. *If* $L_i < \varepsilon$ then
   set $\lambda^* = \dfrac{b + a}{2}$; compute $F(\lambda^*)$ and STOP,
   else go to Step 2.

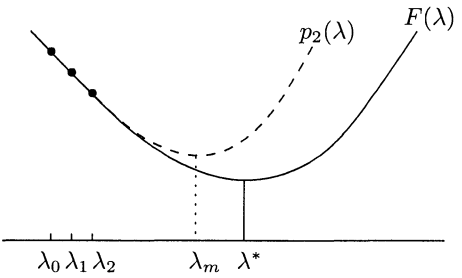

Figure 2.4: Approximate minimum λ_m via quadratic interpolation

2.2.2 Powell's quadratic interpolation algorithm

In Powell's method successive quadratic interpolation curves are fitted to function data giving a sequence of approximations to the minimum point λ^*.

With reference to Figure 2.4, the basic idea is the following. Given three data points $\{(\lambda_i, F(\lambda_i)),\ i = 1,2,3\}$, then the interpolating quadratic polynomial through these points $p_2(\lambda)$ is given by

$$p_2(\lambda) = F(\lambda_0) + F[\lambda_0, \lambda_1](\lambda - \lambda_0) + F[\lambda_0, \lambda_1, \lambda_2](\lambda - \lambda_0)(\lambda - \lambda_1) \quad (2.3)$$

where $F[\ ,\ ]$ and $F[\ ,\ ,\ ]$ respectively denote the first order and second order divided differences.

The turning point of $p_2(\lambda)$ occurs where the slope is zero, i.e. where

$$\frac{dp_2}{d\lambda} = F[\lambda_0, \lambda_1] + 2\lambda F[\lambda_0, \lambda_1, \lambda_2] - F[\lambda_0, \lambda_1, \lambda_2](\lambda_0 + \lambda_1) = 0$$

which gives the turning point λ_m as

$$\lambda_m = \frac{F[\lambda_0, \lambda_1, \lambda_2](\lambda_0 + \lambda_1) - F[\lambda_0, \lambda_1]}{2F[\lambda_0, \lambda_1, \lambda_2]} \cong \lambda^* \quad (2.4)$$

with the further condition that for a minimum the second derivative must be non-negative, i.e. $F[\lambda_0, \lambda_1, \lambda_2] > 0$.

The detailed formal algorithm is as follows.

2.2.2.1 Powell's interpolation algorithm

Given starting point λ_0, stepsize h, tolerance ε and maximum stepsize H; perform following steps:

1. Compute $F(\lambda_0)$ and $F(\lambda_0 + h)$.
2. *If* $F(\lambda_0) < F(\lambda_0 + h)$ evaluate $F(\lambda_0 - h)$,
   else evaluate $F(\lambda_0 + 2h)$. (The three initial values of λ so chosen constitute the initial set $(\lambda_0, \lambda_1, \lambda_2)$ with corresponding function values $F(\lambda_i)$, $i = 0, 1, 2$.)
3. Compute turning point λ_m by formula (2.4) and test for minimum or maximum.
4. *If* λ_m a minimum point *and* $|\lambda_m - \lambda_n| > H$, where λ_n is the nearest point to λ_m, then discard the point furthest from λ_m and take a step of size H from the point with lowest value in direction of descent, and go to Step 3;
   if λ_m a maximum point, then discard point nearest λ_m and take a step of size H from the point with lowest value in the direction of descent and go to Step 3;
   else continue.
5. *If* $|\lambda_m - \lambda_n| < \varepsilon$ then $F(\lambda^*) \cong \min[F(\lambda_m), F(\lambda_n)]$ and STOP,
   else continue.
6. Discard point with highest F value and replace it by λ_m; go to Step 3

Note: It is always safer to compute the next turning point by interpolation rather than by extrapolation. Therefore in Step 6: if the maximum value of F corresponds to a point which lies alone on one side of λ_m, then rather discard the point with highest value on the other side of λ_m.

2.2.3 Exercises

Apply the golden section method and Powell's method to the problems below. Compare their respective performances with regard to the num-

ber of function evaluation required to attain the prescribed accuracies.

(i) minimize $F(\lambda) = \lambda^2 + 2e^{-\lambda}$ over $[0, 2]$ with $\varepsilon = 0.01$.

(ii) maximize $F(\lambda) = \lambda \cos \lambda$ over $[0, \pi/2]$ with $\varepsilon = 0.001$.

(iii) minimize $F(\lambda) = 4(\lambda-7)/(\lambda^2+\lambda-2)$ over $[-1.9; 0.9]$ by performing only 10 function evaluations.

(iv) minimize $F(\lambda) = \lambda^4 - 20\lambda^3 + 0.1\lambda$ over $[0; 20]$ with $\varepsilon = 10^{-5}$.

2.3 First order line search descent methods

Line search descent methods (see Section 2.1.1), that use the gradient vector $\nabla f(\mathbf{x})$ to determine the search direction for each iteration, are called *first order methods* because they employ *first order* partial derivatives of $f(\mathbf{x})$ to compute the search direction at the current iterate. The simplest and most famous of these methods is the *method of steepest descent*, first proposed by Cauchy in 1847.

2.3.1 The method of steepest descent

In this method the direction of steepest descent is used as the search direction in the line search descent algorithm given in section 2.1.1. The expression for the direction of steepest descent is derived below.

2.3.1.1 The direction of steepest descent

At $\mathbf{x}'$ we seek the unit vector $\mathbf{u}$ such that for $F(\lambda) = f(\mathbf{x}' + \lambda\mathbf{u})$, the directional derivative

$$\left.\frac{df(\mathbf{x}')}{d\lambda}\right|_{\mathbf{u}} = \frac{dF(0)}{d\lambda} = \nabla^T f(\mathbf{x}')\mathbf{u}$$

assumes a minimum value with respect to all possible choices for the unit vector $\mathbf{u}$ at $\mathbf{x}'$ (see Figure 2.5).

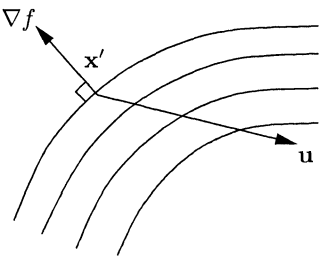

Figure 2.5: Search direction $\mathbf{u}$ relative to gradient vector at $\mathbf{x}'$

By Schwartz's inequality:

$$\nabla^T f(\mathbf{x}')\mathbf{u} \geq -\|\nabla f(\mathbf{x}')\|\|\mathbf{u}\| = -\|\nabla f(\mathbf{x}')\| = \text{ least value.}$$

Clearly for the particular choice $\mathbf{u} = \dfrac{-\nabla f(\mathbf{x}')}{\|\nabla f(\mathbf{x}')\|}$ the directional derivative at $\mathbf{x}'$ is given by

$$\frac{dF(0)}{d\lambda} = -\nabla^T f(\mathbf{x}')\frac{\nabla f(\mathbf{x}')}{\|\nabla f(\mathbf{x}')\|} = -\|\nabla f(\mathbf{x}')\| = \text{ least value.}$$

Thus this particular choice for the unit vector corresponds to the direction of steepest descent.

The search direction

$$\mathbf{u} = \frac{-\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|} \tag{2.5}$$

is called the *normalized steepest descent direction* at $\mathbf{x}$.

2.3.1.2 Steepest descent algorithm

Given $\mathbf{x}^0$, do for iteration $i = 1, 2, \ldots$ until convergence:

1. set $\mathbf{u}^i = \dfrac{-\nabla f(\mathbf{x}^{i-1})}{\|\nabla f(\mathbf{x}^{i-1})\|}$
2. set $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i$ where λ_i is such that

$$F(\lambda_i) = f(\mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i) = \min_{\lambda} f(\mathbf{x}^{i-1} + \lambda \mathbf{u}^i) \text{ (line search).}$$

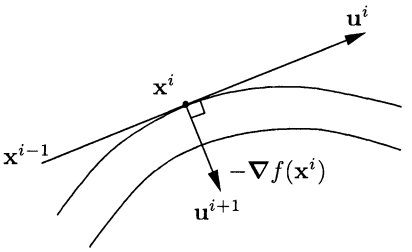

Figure 2.6: Orthogonality of successive steepest descent search directions

2.3.1.3 Characteristic property

Successive steepest descent search directions can be shown to be *orthogonal*. Consider the line search through $\mathbf{x}^{i-1}$ in the direction $\mathbf{u}^i$ to give $\mathbf{x}^i$. The condition for a minimum at λ_i, i.e. for optimal descent, is

$$\left.\frac{df(\mathbf{x}^{i-1}+\lambda_i\mathbf{u}^i)}{d\lambda}\right|_{\mathbf{u}^i} = \left.\frac{dF(\lambda_i)}{d\lambda}\right|_{\mathbf{u}^i} = \nabla^T f(\mathbf{x}^i)\mathbf{u}^i = 0$$

and with $\mathbf{u}^{i+1} = -\dfrac{\nabla f(\mathbf{x}^i)}{\|\nabla f(\mathbf{x}^i)\|}$ it follows that $\mathbf{u}^{i+1^T}\mathbf{u}^i = 0$ as shown in Figure 2.6.

2.3.1.4 Convergence criteria

In practice the algorithm is terminated if some convergence criterion is satisfied. Usually termination is enforced at iteration i if one, or a combination, of the following criteria is met:

(i) $\|\mathbf{x}^i - \mathbf{x}^{i-1}\| < \varepsilon_1$

(ii) $\|\nabla f(\mathbf{x}^i)\| < \varepsilon_2$

(iii) $|f(\mathbf{x}^i) - f(\mathbf{x}^{i-1})| < \varepsilon_3$.

where ε_1, ε_2 and ε_3 are prescribed small positive tolerances.

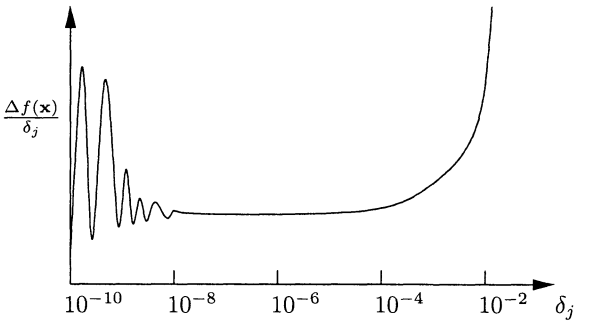

Figure 2.7: Sensitivity of finite difference approximation to δ_j

2.3.1.5 Gradients by finite differences

Often the components of the gradient vector is not analytically available in which case they may be approximated by forward finite differences:

$$\frac{\partial f(\mathbf{x})}{\partial x_j} \cong \frac{\Delta f(\mathbf{x})}{\delta_j} = \frac{f(\mathbf{x} + \boldsymbol{\delta}_j) - f(\mathbf{x})}{\delta_j} \tag{2.6}$$

where $\boldsymbol{\delta}_j = [0, 0, \ldots \delta_j, 0, \ldots, 0]^T$, $\delta_j > 0$ in the j-th position.

Often $\delta_j \equiv \delta$ for all $j = 1, 2, \ldots, n$. A typically choice is $\delta = 10^{-6}$. If however "*numerical noise*" is present in the computation of $f(\mathbf{x})$, special care should be taken in selecting δ_j. This may require doing some numerical experiments such as, for example, determining the sensitivity of approximation (2.6) to δ_j, for each j. Typically the sensitivity graph obtained is as depicted in Figure 2.7, and for the implementation of the optimization algorithm a value for δ_j should be chosen which corresponds to a point on the plateau as shown in Figure 2.7. Better approximations, at of course greater computational expense, may be obtained through the use of central finite differences.

2.3.2 Conjugate gradient methods

In spite of its local optimal descent property the method of steepest descent often performs poorly, following a zigzagging path of ever decreas-

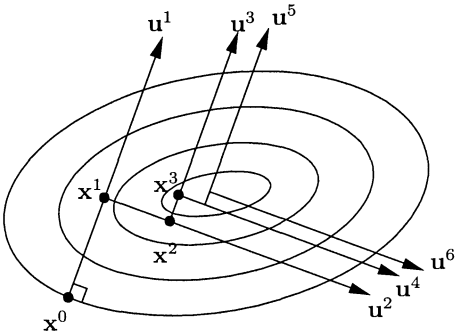

Figure 2.8: Orthogonal zigzagging behaviour of the steepest descent method

ing steps. This results in slow convergence and becomes extreme when the problem is poorly scaled, i.e. when the contours are extremely elongated. This poor performance is mainly due to the fact that the method enforces successive orthogonal search directions (see Section 2.3.1.3) as shown in Figure 2.8. Although, from a theoretical point of view, the method can be proved to be convergent, in practice the method may not effectively converge within a finite number of steps. Depending on the starting point this poor convergence also occurs when applying the method to positive-definite quadratic functions.

There is, however, a class of first order line search descent methods, known as *conjugate gradient methods*, for which it can be proved that whatever the scaling, a *method from this class will converge exactly in a finite number of iterations when applied to a positive-definite quadratic function*, i.e. to a function of the form

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c \tag{2.7}$$

where $c \in \mathbb{R}$, $\mathbf{b}$ is a real n-vector and $\mathbf{A}$ is a positive-definite $n \times n$ real symmetric matrix. Methods that have this property of *quadratic termination* are highly rated, because they are expected to also perform well on other non-quadratic functions in the neighbourhood of a local minimum. This is so, because by the Taylor expansion (1.21), it can be seen that many general differentiable functions approximate the form (2.7) near a local minimum.

2.3.2.1 Mutually conjugate directions

Two vectors $\mathbf{u}$, $\mathbf{v} \neq \mathbf{0}$ are defined to be *orthogonal* if the scalar product $\mathbf{u}^T\mathbf{v} = (\mathbf{u}, \mathbf{v}) = 0$. The concept of *mutual conjugacy* may be defined in a similar manner. Two vectors $\mathbf{u}$, $\mathbf{v} \neq \mathbf{0}$, are defined to be *mutually conjugate* with respect to the matrix $\mathbf{A}$ in (2.7) if $\mathbf{u}^T\mathbf{A}\mathbf{v} = (\mathbf{u}, \mathbf{A}\mathbf{v}) = 0$. Note that $\mathbf{A}$ is a positive definite symmetric matrix.

It can also be shown (see Theorem 6.5.1 in Chapter 6) that if the set of vectors $\mathbf{u}^i$, $i = 1, 2, \ldots, n$ are *mutually conjugate*, then they form a *basis* in $\mathbb{R}^n$, i.e. any $\mathbf{x} \in \mathbb{R}^n$ may be expressed as

$$\mathbf{x} = \sum_{i=1}^{n} \lambda_i \mathbf{u}^i \tag{2.8}$$

where

$$\lambda_i = \frac{(\mathbf{u}^i, \mathbf{A}\mathbf{x})}{(\mathbf{u}^i, \mathbf{A}\mathbf{u}^i)}. \tag{2.9}$$

2.3.2.2 Convergence theorem for mutually conjugate directions

Suppose $\mathbf{u}^i$, $i = 1, 2, \ldots, n$ are mutually conjugate with respect to positive-definite $\mathbf{A}$, then the optimal line search descent method in Section 2.1.1, using $\mathbf{u}^i$ as search directions, converges to the unique minimum $\mathbf{x}^*$ of $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$ in less than or equal to n steps.

Proof:

If $\mathbf{x}^0$ the starting point, then after i iterations:

$$\begin{aligned} \mathbf{x}^i &= \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i = \mathbf{x}^{i-2} + \lambda_{i-1}\mathbf{u}^{i-1} + \lambda_i \mathbf{u}^i = \ldots \\ &= \mathbf{x}^0 + \sum_{k=1}^{i} \lambda_k \mathbf{u}^k. \end{aligned} \tag{2.10}$$

The condition for *optimal descent* at iteration i is

$$\begin{aligned} \frac{dF(\lambda_i)}{d\lambda} = \left.\frac{df(\mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i)}{d\lambda}\right|_{\mathbf{u}^i} &= [\mathbf{u}^i, \nabla f(\mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i)] = 0 \\ &= [\mathbf{u}^i, \nabla f(\mathbf{x}^i)] = 0 \end{aligned}$$

i.e.

$$\begin{aligned} 0 &= (\mathbf{u}^i, \mathbf{A}\mathbf{x}^i + \mathbf{b}) \\ &= \left(\mathbf{u}^i, \mathbf{A}\left(\mathbf{x}^0 + \sum_{k=1}^{i} \lambda_k \mathbf{u}^k\right) + \mathbf{b}\right) = (\mathbf{u}^i, \mathbf{A}\mathbf{x}^0 + \mathbf{b}) + \lambda_i (\mathbf{u}^i, \mathbf{A}\mathbf{u}^i) \end{aligned}$$

because $\mathbf{u}^i$, $i = 1, 2, \ldots, n$ are mutually conjugate, and thus

$$\lambda_i = -(\mathbf{u}^i, \mathbf{A}\mathbf{x}^0 + \mathbf{b})/(\mathbf{u}^i, \mathbf{A}\mathbf{u}^i). \tag{2.11}$$

Substituting (2.11) into (2.10) above gives

$$\begin{aligned} \mathbf{x}^n &= \mathbf{x}^0 + \sum_{i=1}^{n} \lambda_i \mathbf{u}^i = \mathbf{x}^0 - \sum_{i=1}^{n} \frac{(\mathbf{u}^i, \mathbf{A}\mathbf{x}^0 + \mathbf{b})\mathbf{u}^i}{(\mathbf{u}^i, \mathbf{A}\mathbf{u}^i)} \\ &= \mathbf{x}^0 - \sum_{i=1}^{n} \frac{(\mathbf{u}^i, \mathbf{A}\mathbf{x}^0)\mathbf{u}^i}{(\mathbf{u}^i, \mathbf{A}\mathbf{u}^i)} - \sum_{i=1}^{n} \frac{(\mathbf{u}^i, \mathbf{A}(\mathbf{A}^{-1}\mathbf{b}))\mathbf{u}^i}{(\mathbf{u}^i, \mathbf{A}\mathbf{u}^i)}. \end{aligned}$$

Now by utilizing (2.8) and (2.9) it follows that

$$\mathbf{x}^n = \mathbf{x}^0 - \mathbf{x}^0 - \mathbf{A}^{-1}\mathbf{b} = -\mathbf{A}^{-1}\mathbf{b} = \mathbf{x}^*.$$

The implication of the above theorem for the case $n = 2$ and where mutually conjugate line search directions $\mathbf{u}^1$ and $\mathbf{u}^2$ are used, is depicted in Figure 2.9.

2.3.2.3 Determination of mutually conjugate directions

How can mutually conjugate search directions be found? One way is to determine all the eigenvectors $\mathbf{u}^i$, $i = 1, 2, \ldots, n$ of $\mathbf{A}$. For $\mathbf{A}$ positive-definite, all the eigenvectors are mutually orthogonal and since $\mathbf{A}\mathbf{u}^i = \mu_i \mathbf{u}^i$ where μ_i is the associated eigenvalue, it follows directly that for all $i \neq j$ that $(\mathbf{u}^i, \mathbf{A}\mathbf{u}^j) = (\mathbf{u}^i, \mu_j \mathbf{u}^j) = \mu_j (\mathbf{u}^i, \mathbf{u}^j) = 0$, i.e. the eigenvectors are mutually conjugate with respect to $\mathbf{A}$. It is, however, not very practical to determine mutually conjugate directions by finding all the eigenvectors of $\mathbf{A}$, since the latter task in itself represents a computational problem of magnitude equal to that of solving the original unconstrained optimization problem via any other numerical algorithm. An easier method for obtaining mutually conjugate directions, is by means of the Fletcher-Reeves formulae (Fletcher and Reeves, 1964).

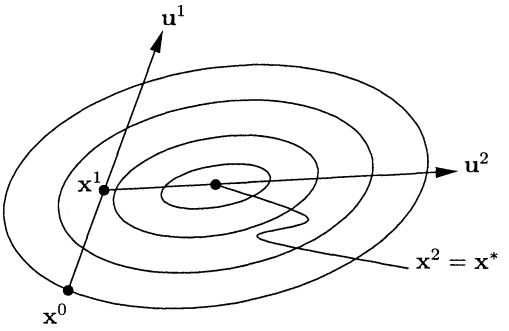

Figure 2.9: Quadratic termination of the conjugate gradient method in two steps for the case $n = 2$

2.3.2.4 The Fletcher-Reeves directions

The *Fletcher-Reeves directions* $\mathbf{u}^i$, $i = 1, 2, \ldots, n$, that are listed below, can be shown (see Theorem 6.5.3 in Chapter 6) to be mutually conjugate with respect to the matrix $\mathbf{A}$ in the expression for the quadratic function in (2.7) (for which $\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$). The explicit directions are:

$$\mathbf{u}^1 = -\nabla f(\mathbf{x}^0)$$

and for $i = 1, 2, \ldots, n-1$

$$\mathbf{u}^{i+1} = -\nabla f(\mathbf{x}^i) + \beta_i \mathbf{u}^i \tag{2.12}$$

where $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i$, and λ_i corresponds to the optimal descent step in iteration i, and

$$\beta_i = \frac{\|\nabla f(\mathbf{x}^i)\|^2}{\|\nabla f(\mathbf{x}^{i-1})\|^2}. \tag{2.13}$$

The *Polak-Ribiere* directions are obtained if, instead of using (2.13), β_i is computed using

$$\beta_i = \frac{(\nabla f(\mathbf{x}^i) - \nabla f(\mathbf{x}^{i-1}))^T \nabla f(\mathbf{x}^i)}{\|\nabla f(\mathbf{x}^{i-1})\|^2}. \tag{2.14}$$

If $f(\mathbf{x})$ is quadratic it can be shown (Fletcher, 1987) that (2.14) is equivalent to (2.13).

2.3.2.5 Formal Fletcher-Reeves conjugate gradient algorithm for general functions

Given $\mathbf{x}^0$ perform the following steps:

1. Compute $\nabla f(\mathbf{x}^0)$ and set $\mathbf{u}^1 = -\nabla f(\mathbf{x}^0)$.
2. For $i = 1, 2, \ldots, n$ do:
   2.1 set $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i$ where λ_i such that
   $$f(\mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i) = \min_{\lambda} f(\mathbf{x}^{i-1} + \lambda \mathbf{u}^i) \text{ (line search)},$$
   2.2 compute $\nabla f(\mathbf{x}^i)$,
   2.3 *if* convergence criteria satisfied, then STOP and $\mathbf{x}^* \cong \mathbf{x}^i$, *else* go to Step 2.4.
   2.4 *if* $1 \leq i \leq n-1$, $\mathbf{u}^{i+1} = -\nabla f(\mathbf{x}^i) + \beta_i \mathbf{u}^i$ with β_i given by (2.13).
3. Set $\mathbf{x}^0 = \mathbf{x}^n$ and go to Step 2 (restart).

If β_i is computed by (2.14) instead of (2.13) the method is known as the *Polak-Ribiere* method.

2.3.2.6 Simple illustrative example

Apply the Fletcher-Reeves method to minimize

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

with $\mathbf{x}^0 = [10, -5]^T$.

Solution:

Iteration 1:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} x_1 + x_2 \\ x_1 + 2x_2 \end{bmatrix} \text{ and therefore } \mathbf{u}^1 = -\nabla f(\mathbf{x}^0) = \begin{bmatrix} -5 \\ 0 \end{bmatrix}.$$

$$\mathbf{x}^1 = \mathbf{x}^0 + \lambda \mathbf{u}^1 = \begin{bmatrix} 10 - 5\lambda \\ -5 \end{bmatrix} \text{ and}$$

$$F(\lambda) = f(\mathbf{x}^0 + \lambda \mathbf{u}^1) = \tfrac{1}{2}(10 - 5\lambda)^2 + (10 - 5\lambda)(-5) + 25.$$

For optimal descent

$$\frac{dF}{d\lambda}(\lambda) = \frac{df}{d\lambda}\bigg|_{\mathbf{u}^1} = -5(10 - 5\lambda) + 25 = 0 \text{ (line search).}$$

This gives $\lambda_1 = 1$, $\mathbf{x}^1 = \begin{bmatrix} 5 \\ -5 \end{bmatrix}$ and $\nabla f(\mathbf{x}^1) = \begin{bmatrix} 0 \\ -5 \end{bmatrix}$.

Iteration 2:

$$\mathbf{u}^2 = -\nabla f(\mathbf{x}^1) + \frac{\|\nabla f(\mathbf{x}^1)\|^2}{\|\nabla f(\mathbf{x}^0)\|^2}\mathbf{u}^1 = -\begin{bmatrix} 0 \\ -5 \end{bmatrix} + \tfrac{25}{25}\begin{bmatrix} -5 \\ 0 \end{bmatrix} = \begin{bmatrix} -5 \\ 5 \end{bmatrix}.$$

$$\mathbf{x}^2 = \mathbf{x}^1 + \lambda \mathbf{u}^2 = \begin{bmatrix} 5 \\ -5 \end{bmatrix} + \lambda \begin{bmatrix} -5 \\ 5 \end{bmatrix} = \begin{bmatrix} 5(1 - \lambda) \\ -5(1 - \lambda) \end{bmatrix} \text{ and}$$

$$F(\lambda) = f(\mathbf{x}^1 + \lambda \mathbf{u}^2) = \tfrac{1}{2}[25(1 - \lambda)^2 - 50(1 - \lambda)^2 + 50(1 - \lambda)^2].$$

Again for optimal descent

$$\frac{dF}{d\lambda}(\lambda) = \frac{df}{d\lambda}\bigg|_{\mathbf{u}^2} = -25(1 - \lambda) = 0 \text{ (line search).}$$

This gives $\lambda_2 = 1$, $\mathbf{x}^2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\nabla f(\mathbf{x}^2) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. Therefore STOP.

The two iteration steps are shown in Figure 2.10.

2.4 Second order line search descent methods

These methods are based on Newton's method (see Section 1.5.4.1) for solving $\nabla f(\mathbf{x}) = \mathbf{0}$ iteratively: Given $\mathbf{x}^0$, then

$$\mathbf{x}^i = \mathbf{x}^{i-1} - \mathbf{H}^{-1}(\mathbf{x}^{i-1})\nabla f(\mathbf{x}^{i-1}), \; i = 1, 2, \ldots \tag{2.15}$$

As stated in Chapter 1, the main characteristics of this method are:

1. In the neighbourhood of the solution it may converge very fast, in

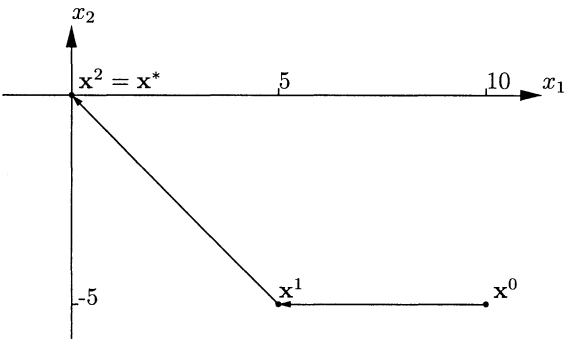

Figure 2.10: Convergence of Fletcher-Reeves method for illustrative example

fact it has the very desirous property of being quadratically convergent if it converges. Unfortunately convergence is not guaranteed and it may sometimes diverge, even from close to the solution.

2. The implementation of the method requires that $\mathbf{H}(\mathbf{x})$ be evaluated at each step.

3. To obtain the Newton step, $\boldsymbol{\Delta} = \mathbf{x}^i - \mathbf{x}^{i-1}$ it is also necessary to solve a $n \times n$ linear system $\mathbf{H}(\mathbf{x})\boldsymbol{\Delta} = -\boldsymbol{\nabla} f(\mathbf{x})$ at each iteration. This is computationally very expensive for large n, since an order n^3 multiplication operations are required to solve the system numerically.

2.4.1 Modified Newton's method

To avoid the problem of convergence (point 1. above), the computed Newton step $\boldsymbol{\Delta}$ is rather used as a search direction in the general line search descent algorithm given in Section 2.1.1. Thus at iteration i: select $\mathbf{u}^i = \boldsymbol{\Delta} = -\mathbf{H}^{-1}(\mathbf{x}^{i-1})\boldsymbol{\nabla} f(\mathbf{x}^{i-1})$, and minimize in that direction to obtain a λ_i such that

$$f(\mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i) = \min_{\lambda} f(\mathbf{x}^{i-1} + \lambda \mathbf{u}^i)$$

and then set $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i$.

2.4.2 Quasi-Newton methods

To avoid the above mentioned computational problems (2. and 3.), methods have been developed in which approximations to $\mathbf{H}^{-1}$ are applied at each iteration. Starting with an approximation $\mathbf{G}_0$ to $\mathbf{H}^{-1}$ for the first iteration, the approximation is updated after each line search. An example of such a method is the Davidon-Fletcher-Powell (DFP) method.

2.4.2.1 DFP quasi-Newton method

The structure of this (rank-1 update) method (Fletcher, 1987) is as follows.

1. Choose $\mathbf{x}^0$ and set $\mathbf{G}_0 = \mathbf{I}$.
2. Do for iteration $i = 1, 2, \ldots, n$:
   - 2.1 set $\mathbf{x}^i = \mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i$, where $\mathbf{u}^i = -\mathbf{G}_{i-1} \nabla f(\mathbf{x}^{i-1})$ and λ_i is such that $f(\mathbf{x}^{i-1} + \lambda_i \mathbf{u}^i) = \min_{\lambda} f(\mathbf{x}^{i-1} + \lambda \mathbf{u}^i)$, $\lambda_i \geq 0$ (line search),
   - 2.2 if stopping criteria satisfied then STOP, $\mathbf{x}^* \cong \mathbf{x}^i$,
   - 2.3 set $\mathbf{v}^i = \lambda_i \mathbf{u}^i$ and set $\mathbf{y}^i = \nabla f(\mathbf{x}^i) - \nabla f(\mathbf{x}^{i-1})$,
   - 2.4 set
   $$\mathbf{G}_i = \mathbf{G}_{i-1} + \mathbf{A}_i + \mathbf{B}_i \text{ (rank 1-update)} \tag{2.16}$$
   where $\mathbf{A}_i = \dfrac{\mathbf{v}^i \mathbf{v}^{iT}}{\mathbf{v}^{iT} \mathbf{y}^i}$, $\mathbf{B}_i = \dfrac{-\mathbf{G}_{i-1} \mathbf{y}^i (\mathbf{G}_{i-1} \mathbf{y}^i)^T}{\mathbf{y}^{iT} \mathbf{G}_{i-1} \mathbf{y}^i}$.
3. Set $\mathbf{x}^0 = \mathbf{x}^n$; $\mathbf{G}_0 = \mathbf{G}_n$ (or $\mathbf{G}_0 = \mathbf{I}$), and go to Step 2 (restart).

2.4.2.2 Characteristics of DFP method

1. The method does not require the evaluation of $\mathbf{H}$ or the explicit solution of a linear system.

2. If $\mathbf{G}_{i-1}$ is positive-definite then so is $\mathbf{G}_i$ (see Theorem 6.6.1).

3. If $\mathbf{G}_i$ is positive-definite then descent is ensured at $\mathbf{x}^i$ because

$$\left.\frac{df(\mathbf{x}^i)}{d\lambda}\right|_{\mathbf{u}^{i+1}} = \nabla^T f(\mathbf{x}^i)\mathbf{u}^{i+1}$$
$$= -\nabla^T f(\mathbf{x}^i)\mathbf{G}_i \nabla f(\mathbf{x}^i) < 0, \text{ for all } \nabla f(\mathbf{x}) \neq \mathbf{0}.$$

4. The directions $\mathbf{u}^i$, $i = 1, 2, \ldots, n$ are mutually conjugate for a quadratic function with $\mathbf{A}$ positive-definite (see Theorem 6.6.2). The method therefore possesses the desirable property of *quadratic termination* (see Section 2.3.2).

5. For quadratic functions: $\mathbf{G}_n = \mathbf{A}^{-1}$ (see again Theorem 6.6.2).

2.4.2.3 The BFGS method

The state-of-the-art quasi-Newton method is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method developed during the early 1970s (see Fletcher, 1987). This method uses a more complicated rank-2 update formula for $\mathbf{H}^{-1}$. For this method the update formula to be used in Step 2.4 of the algorithm given in Section 2.4.2.1 becomes

$$\mathbf{G}_i = \mathbf{G}_{i-1} + \left[1 + \frac{\mathbf{y}^{iT}\mathbf{G}_{i-1}\mathbf{y}^i}{\mathbf{v}^{iT}\mathbf{y}^i}\right]\left[\frac{\mathbf{v}^i\mathbf{v}^{iT}}{\mathbf{v}^{iT}\mathbf{y}^i}\right] - \left[\frac{\mathbf{v}^i\mathbf{y}^{iT}\mathbf{G}_{i-1} + \mathbf{G}_{i-1}\mathbf{y}^i\mathbf{v}^{iT}}{\mathbf{v}^{iT}\mathbf{y}^i}\right]. \tag{2.17}$$

2.5 Zero order methods and computer optimization subroutines

This chapter would not be complete without mentioning something about the large number of so-called *zero order* methods that have been developed. These methods are called such because they do not use either first order or second order derivative information, but only function values, i.e. only zero order derivative information.

Zero order methods are of the earliest methods and many of them are based on rough and ready ideas without very much theoretical background. Although these ad hoc methods are, as one may expect, much slower and computationally much more expensive than the higher order methods, they are usually reliable and easy to program. One of the most successful of these methods is the *simplex method* of Nelder and Mead (1965). This method should not be confused with the simplex method for linear programming. Another very powerful and popular method that only uses function values is the multi-variable method of Powell (1964). This method generates mutually conjugate directions by performing sequences of line searches in which only function evaluations are used. For this method Theorem 2.3.2.2 applies and the method therefore possesses the property if quadratic termination.

Amongst the more recently proposed and modern zero order methods, the method of *simulated annealing* and the so-called *genetic algorithms* (GAs) are the most prominent (see for example, Haftka and Gündel, 1992).

Computer programs are commercially available for all the unconstrained optimization methods presented in this chapter. Most of the algorithms may, for example, be found in the *Matlab Optimization Toolbox* and in the *IMSL* and *NAG* mathematical subroutine libraries.

2.6 Test functions

The efficiency of an algorithm is studied using standard functions with standard starting points $\mathbf{x}^0$. The total number of functions evaluations required to find $\mathbf{x}^*$ is usually taken as a measure of the efficiency of the algorithm. Some classical test functions (Rao, 1996) are listed below.

1. Rosenbrock's parabolic valley:

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2;\ \mathbf{x}^0 = \begin{bmatrix} -1.2 \\ 1.0 \end{bmatrix}\ \mathbf{x}^* = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

2. Quadratic function:

$$f(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2;\ \mathbf{x}^0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}\ \mathbf{x}^* = \begin{bmatrix} 1 \\ 3 \end{bmatrix}.$$

3. Powell's quartic function:

$$\begin{aligned} f(\mathbf{x}) &= (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4; \\ \mathbf{x}^0 &= [3, -1, 0, 1]^T; \quad \mathbf{x}^* = [0, 0, 0, 0]^T. \end{aligned}$$

4. Fletcher and Powell's helical valley:

$$\begin{aligned} f(\mathbf{x}) &= 100\left((x_3 - 10\theta(x_1, x_2))^2 \right. \\ &\quad \left. + \left(\sqrt{x_1^2 + x_2^2} - 1\right)^2\right) + x_3^2; \\ \text{where } 2\pi\theta(x_1, x_2) &= \begin{cases} \arctan \dfrac{x_2}{x_1} \text{ if } x_1 > 0 \\ \pi + \arctan \dfrac{x_2}{x_1} \text{ if } x_1 < 0 \end{cases} \\ \mathbf{x}^0 &= [-1, 0, 0]^T; \quad \mathbf{x}^* = [1, 0, 0]^T. \end{aligned}$$

5. A non-linear function of three variables:

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{1 + (x_1 - x_2)^2} + \sin\left(\frac{1}{2}\pi x_2 x_3\right) + \exp\left(-\left(\frac{x_1 + x_3}{x_2} - 2\right)^2\right); \\ \mathbf{x}^0 &= [0, 1, 2]^T; \quad \mathbf{x}^* = [1, 1, 1]^T. \end{aligned}$$

6. Freudenstein and Roth function:

$$\begin{aligned} f(\mathbf{x}) &= (-13 + x_1 + ((5 - x_2)x_2 - 2)x_2)^2 \\ &\quad + (-29 + x_1 + ((x_2 + 1)x_2 - 14)x_2)^2; \\ \mathbf{x}^0 &= [0.5, -2]^T; \quad \mathbf{x}^* = [5, 4]^T; \quad \mathbf{x}^*_{\text{local}} = [11.41\ldots, -0.8968\ldots]^T. \end{aligned}$$

7. Powell's badly scaled function:

$$\begin{aligned} f(\mathbf{x}) &= (10\,000 x_1 x_2 - 1)^2 + (\exp(-x_1) + \exp(-x_2) - 1.0001)^2; \\ \mathbf{x}^0 &= [0, 1]^T; \quad \mathbf{x}^* = [1.098\cdots \times 10^{-5}, 9.106\ldots]^T. \end{aligned}$$

8. Brown's badly scaled function:

$$\begin{aligned} f(\mathbf{x}) &= (x_1 - 10^6)^2 + (x_2 - 2 \times 10^{-6})^2 + (x_1 x_2 - 2)^2; \\ \mathbf{x}^0 &= [1, 1]^T; \quad \mathbf{x}^* = [10^6, 2 \times 10^6]^T. \end{aligned}$$

9. Beale's function:

$$\begin{aligned} f(\mathbf{x}) &= (1.5 - x_1(1 - x_2))^2 + (2.25 - x_1(1 - x_2^2))^2 \\ &\quad +(2.625 - x_1(1 - x_2^3))^2; \\ \mathbf{x}^0 &= [1, 1]^T; \quad \mathbf{x}^* = [3, 0.5]^T. \end{aligned}$$

10. Wood's function:

$$\begin{aligned} f(\mathbf{x}) &= 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 \\ &\quad +10(x_2 + x_4 - 2)^2 + 0.1(x_2 - x_4)^2 \\ \mathbf{x}^0 &= [-3, -1, -3, -1]^T; \quad \mathbf{x}^* = [1, 1, 1, 1]^T. \end{aligned}$$

http://www.springer.com/978-0-387-24348-1

Practical Mathematical Optimization
An Introduction to Basic Optimization Theory and
Classical and New Gradient-Based Algorithms
Snyman, J.
2005, XX, 258 p., Hardcover
ISBN: 978-0-387-24348-1