Secant and conjugate direction methods in optimisation

G. A. Ringwood

SECANT AND CONJUGATE DIRECTION METHODS IN OPTIMISATION

by G A Ringwood

Thesis submitted for the degree of MSc Department of Mathematics, University of Durham

September 1978

SYNOPSIS

Criterion for the comparison of numerical methods for optimising a function of a finite number of variables are suggested firstly in general terms, then more specifically. The general criteria lead one to examine algorithms which are hybrids of line searches and methods for solving nonlinear equations.

Methods for solving nonlinear equations are to a greater or lesser extent based on methods which solve affine equations. It is shown that for two of these general methods while apparently having the same effect, namely diagonal matrix reduction, one essentially performs reduction to lower triangular form and the other reduction to upper triangular form. It is pointed out that the specific criteria are sometimes qualified by misconceptions about Newton's algorithm for solving nonlinear equations.

The specific criteria are used to compare some well known algorithms which are discussed in the text. It is argued that algorithms which posses all the desirable qualities are likely to perform better than those which only posses a few. On this basis Davidon's(1975) algorithm should rival modified Newton algorithms. Davidon's algorithm may be unecessarily complicated and simpler forms are suggested which also possess all the desired qualities.

CONTENTS

CHAPTER 0 Preliminaries

0.1 Mathematical programming

Mathematical programming, the optimisation of a function of many variables, possibly subject to constraints, is often the final step in the solution of problems in engineering design and operations research e.g.
minimise waste;
minimise cost;
maximise output;
maximise profit.
It is to be hoped that the function to be optimised gives a good description of the real system, waste, cost etc., in response to the variables under the control of the decision maker. If the function to be optimised gives a sufficiently good description of the situation it is supposed to model, the position of the optimum of the function will approximate the 'best' choice of the decision variables.

In general, only a numerical approximate to the position of the optimum of a function (if one exists at all) can be found and this only by an iterative process in which successively better approximations are constructed. The function may be so intractable (e.g. one thousand variables) that an analytic proof of the existence of an optimum is almost impossible. Intuition, however, may suggest that such an optimum exists and in addition provide initial estimates to start the

iterative process. It is to be hoped that the intuitive estimates are good enough so that the algorithm converges.

The engineer or operations researcher has to choose one of a proliferation of computer programs available today to optimise the function. The choice between the different algorithms depends on the problem and on the criteria. Two reasonable criteria are:
rapid convergence and
robustness.
The first criterion is fundamental, computer time is costly. It is, however, in competition with the second criterion, the ability to perform well on all problems. This second discriminant is an expression of idealism, the search for the panacea to all problems. This is too much to hope for but there is a possibility of reducing the choice to just a few algorithms. The reasons for wanting to restrict the choice are
as follows. The engineer or operations researcher will most likely want to optimise different functions from time to time, so familiarity and confidence with a particular program will speed up the process and reduce the number of errors. Familiarity with the mechanism of a particular algorithm will also aid the detection of reasons for failure of the program.

Two more criteria could possibly be added to the above:
minimal storage;
simplicity.
The first could really be said to come under the heading of robustness. One wants the program to work on functions with a large number of variables as well as on small problems. Advances in computer design will allow larger storage with more rapid access so the criterion should be measured, perhaps, as the fractional storage for a given size of problem. The difficulty of storage in extremely large optimisation problems at the

moment may necessitate familiarity with at least two algorithms: one for moderately sized problems, the other for very large problems. One can, however, imagine the development of computers with rapid recall storage so large that the formulation of models which required the maximum amount of storage would not be feasible or even desirable. The second supplementary criterion, simplicity, is attractive on two accounts. Firstly, the detection of sources of program failure is made that much easier; this is especially desirable if the user is not a mathematician. Secondly, simplicity leads to an easier understanding of the process and so to confidence in the final result. Brodlie(1977) states: "In my opinion any modification which complicates an algorithm has to justify itself by a significant improvement in performance. Similarly any modification which simplifies an algorithm —without of course impairing its performance— is especially welcome."

A distinction is usually drawn between constrained and unconstrained optimisation. Algorithms for constrained optimisation are in general adaptations of methods for unconstrained optimisation. Furthermore, constrained problems can be solved by the direct use of unconstrained techniques. It is sometimes possible to take account of a constraint by a change of variables. By careful choice it can happen that the new variables are not subject to any constraint. Unfortunately this is not always the case but by the use of penalty functions a constrained problem can always be converted into an unconstrained one. The penalty function method consists of adding to the objective another function, the penalty, which takes very small values in the constrained variable subspace but very large values outside it. From the desire for robustness it is reasonable that interest should be concentrated on the efficient solution of the unconstrained problem.

Perhaps the most obvious class of algorithms for unconstrained optimisation is that which uses alternating directions. Each iteration

consists of searches made in turn along a complete set of linearly independent directions. The process is then repeated, not necessarily with the same set of directions. The direction set can be updated in the light of previous information. Another direct search strategy, which is useful for functions which are not well defined e.g. observations subject to experimental error, is design search. A maximal initial set of linearly independent points (points which form a simplex) is systematically altered, replacing points with improved estimates of the optimum while retaining the simplex structure.

Direct methods are attractive because of their simplicity and because they make only modest demands on storage. Although the methods are heuristic they have proved to be robust and rarely fail to reach a local minimum. The drawback is that the rate of convergence is sometimes painfully slow. In many applications the function to be optimised is continuously differentiable and without too much difficulty the derivatives can be supplied to the program. In such cases, at the optimum of the function the derivative is zero. This gives a second handle to the solution of the optimisation problem which can only help speed convergence: the optimum is also the solution to a set of nonlinear equations. This suggests some hybrid of methods for solving nonlinear equations with strategies for optimisation. In general the use of derivative information leads to a substantial increase in the speed of convergence.

The speed of convergence of algorithms which use gradient information can be compared theoretically by means of order of convergence, number of function evaluations and housekeeping operations arguements. The order of convergence comparisons are, however, *confined* to certain classes of functions which are sometimes so restrictive that most applications do not fall into one of the classes or the problem is so complicated

that it is impractical to show that it falls into one of the classes. (Dixon's(1972) proof that with accurate line searches a subset of the Huang(1970) algorithms generate identical points is an exception.)

The speed of convergence and robustness of different algorithms may be compared experimentally by finding the time to reach a prescribed degree of precision on certain test functions e.g. Himmelblau(1972). This means of comparison has the drawback that the efficiency of an algorithm depends to a considerable extent on the details of the implementation. Broyden(1972) says "We realize that the reputation of an update may well be due as much to an artful choice of checks, safeguards and program constants with which it is surrounded as to the properties inherent in the update itself, and are only too conscious that a good update may be enhanced, and a poor one disguised, by such devices." This philosophy is exemplified by Dixon's(1972) Theorem: experimental comparison of Huang(1970) algorithms showed marked differences to one another. Because of Dixon's Theorem this result can be put down to sensitivity to inaccuracies in the line searches.

A recent trend (Larichev and Gorvits(1974)), "the middle way" is to compare algorithms at different stages of the optimisation process. Almost all test functions have, topographically speaking, a narrow, steep walled curving valley, the floor of which gently descends to a unimodal minimum. Stages of optimisation can be classified as descent into the valley, advance along the floor of the valley and search in the vicinity of the extremum.

The present work investigates a similar approach to the comparison of algorithms (which have been given various names such as quasi-Newton and variable metric) which are distinguished by the fact that they are a hybrid between the solution of a set of nonlinear

equations and the direct search method of alternating directions. The idea is to motivate and promote certain properties of algorithms and to suggest that algorithms which possess most of the properties are more likely to perform better than algorithms which possess only one or two of the properties. These desirable attributes of algorithms are far from new but will in some cases be motivated in less usual ways. The analysis reveals ways of constructing algorithms which have all the recommended properties.

All present methods for the solution of nonlinear equations are based to some extent on the solution of linear equations. Chapter One examines this basis. Two of the desirable properties of updates, positive definiteness and symmetry are usually motivated by appealing to Newton's method. It is argued that this is to some extent misleading and other possible motivations are put forward. Chapters Two and Three synthesise two classes of algorithm which solve linear equations in a finite number of steps, the secant and conjugate direction methods. These classes are presented in their most fundamental form. It is shown that the conjugate gradient method is essentially reduction to lower triangular form and the secant method is associated with reduction to upper triangular form. Chapter Four proposes the desirable properties and compares some well known algorithms on their basis. The synthesis of Chapters Two and Three suggest ways of constructing other algorithms which possess all the desirable properties. An appendix lists all the algorithms compared and pages in the text which refer to them.

0.2 Notation

Unless otherwise specified the notation in the text will be as follows. Lower case Greek letters will denote scalars. Lower case Latin letters from the beginning and end of the alphabet will be used for column vectors, elements of real Euclidean space R^n. Letters from the middle of the alphabet will be used as subscripts and superscripts. Thus, for example, x^i will be the ith component of the vector x while x_i is the ith vector in a sequence $\{x_i\}$ of vectors. If f is a function from n-dimensional Euclidean space to n-dimensional Euclidean space, denoted $f:R^n \to R^n$, and $\{x_i\}$ is a sequence of vectors in R^n, then $\{f_i\}$ is the sequence of vectors $\{f(x_i)\}$.

The notation (x,y) stands for the convex subset of points expressible in the form $\lambda x + (1-\lambda)y$ where λ is an element of the open unit interval of the real line. This is thus a generalization of the notatiion used for open intervals of the real line; closed and half open intervals are defined correspondingly.

Upper case Latin letters will be used for matrices. The subset, ran(A) will denote the space spanned by the columns of A and nul(A) will denote the null space of A. The matrix A^T stands for the transpose of A whereas $x^T y$ stands for the inner product of vectors x and y. The symbol $\perp$ will be used for the operation of taking the orthogonal complement e.g. $\text{nul}(A) = \text{ran}(A^T)^\perp$. The notation $A^{\overline{k}}$, $A^{|k}$, $A^{\underline{k}}$ and $A^{\underline{|}k}$ will represent the following submatrices of A respectively: the first k rows, the first k columns, the last k rows and the last k columns. Thus, $A^{\sqrt{k}}$ is the kth principle submatrix of A. Square brackets enclosing matrices and vectors separated by commas e.g. $[B,x,C]$ will represent a matrix whose columns are the columns of are the columns of the constituent matrices in left to right order.

Given functions $\lambda: R^n \to R$ and $f: R^n \to R^n$ the derivatives at x_0 $D\lambda|_{x_0}$ and $Df|_{x_0}$ will denote those linear functions which best approximate $\lambda(x) - \lambda(x_0)$ and $f(x) - f(x_0)$ i.e.

$\lambda(h+x_0) = \lambda(x_0) + D\lambda|_{x_0}(h) + o(h)$

$f(h+x_0) = f(x_0) + Df|_{x_0}(h) + o(h)$

That is $D\lambda|_x$ can be thought of as a vector and $Df|_x$ can be thought of as a matrix. Thus, $D\lambda$ is the gradient vector and $DD\lambda$ the Hessian matrix. (The notation $O(h)$ represents some quantity such that $\lim_{|h|\to 0} O(h)/|h| =$ constant where $|h|$ is some norm of h; when the constant is zero a lower case 'o' is used.)

CHAPTER 1 Optimisation and nonlinear equations

1.1 Statement of the problem

Mathematical programming is concerned with finding numerical approximations to x^* (for purposes herein x^* is an element of n-dimensional Euclidian space R^n) which is specified by a subspace $\Gamma \subset R^n$ which contains it and a function $\lambda:\Gamma \to R$ for which $\lambda(x^*) < \lambda(x)$ for all $x \in \Gamma - x^*$.
That is, $\lambda(x^*)$ is the minimum value the function λ takes on Γ: $\lambda(x^*) = \inf(\lambda(\Gamma))$. (If a maximum is sought this can be achieved by using $-\lambda$.) If $\Gamma = R^n$ the problem is said to be unconstrained, otherwise it is constrained. The constraints are usually given in functional form and fall into two classes: equality constraints, $e(x^*) = 0$; $e \in R^m$; $m<n$, which reduce the dimension of the problem and inequality constraints, $b(x^*) = 0$; $b \in R^p$; $p \leq n$, which give rise to boundaries. In many applications the constrained subspace is a reasonable geometric object. If for example with equality constraints, the derivative, De has maximal rank the constrained subspace will be a manifold with dimension n-m. (For inequality constraints the function $b:R^p \to R^n$ has to be transversal (Guillemin and Poliack (1974).)

If, as will usually be assumed, λ is continuously differentiable and the problem is unconstrained, the optimum also satisfies $D\lambda|_{x^*} = 0$, a set of 'n' nonlinear equations. Points at which the gradient is zero are not necessarily global minima. They may, for example, be maxima, saddle points or just local minima. It is thus not sufficient just to solve the set of nonlinear equations; some hybrid with a

search technique not only speeds convergence, it also tends to prevent convergence to any point which is not, at least, a local minimum.

Given a set of nonlinear equations $f(x^*) = 0$ to be solved the problem can be turned into one of optimisation. For example, minimise $f^T f$ subject to $f(x^*) = 0$. There is thus a formal similarity between the following problems:

equality constrained optimisation— minimise λ subject to $e = 0$;
unconstrained optimisation— " λ " $D\lambda = 0$;
set of nonlinear equations— " $f^T f$ " $f = 0$.

Thus, any method for solving one problem can be adapted to solving the others. There appears to be a difference in dimensions. The first problem, equality constraint, is in reasonable cases (when the subspace is a manifold), say, an n-m dimensional problem. If $\lambda(x^*)$ is an optimum on the subspace, then the projection of the gradient $D\lambda|_{x^*}$ onto the tangent space at x^* must be zero. The tangent space will have dimension n-m so together with the m constraint equations, $e = 0$, there is a total of 'n' nonlinear equations.

1.2 Nonlinear equations

This section attempts to examine the bases of methods for solving sets of nonlinear(nonaffine) equations $f(x^*) = 0$. Since primary interest is in optimisation it will be assumed that f has 'n' components. There are a number of well developed methods for solving sets of affine(linear) equations so it is natural to try and formulate the nonaffine problem so as to appear affine. If λ is continuously differentiable,

by the mean value theorem,

$$f(x_1) = f(x_2) + J(x_1 - x_2)$$

where the ith row of the matrix J is

$$Df^i\Big|_{\theta_i x_1 + (1-\theta_i)x_2} \quad , \; 0 \leq \theta_i \leq 1, \quad i = 1, \ldots, n .$$

A matrix $J = [j_1, j_2, \ldots, j_n]$ may alternatively be constructed as follows (Hicks(1971)). Define

$$\tilde{f}_i(x_1; x_2; \alpha) = f(x_2^1, x_2^2, \ldots, x_2^{i-1}, (\alpha x_2^i + (1-\alpha)x_1^i), x_1^{i+1}, \ldots, x_1^n)$$

then

$$f(x_1) - f(x_2) = \sum_{i=1}^{n} \tilde{f}_i(x_1; x_2; \alpha)\Big|_0^1 = \sum_{i=1}^{n} \int_0^1 d\alpha \, \partial \tilde{f}_i / \partial \alpha$$

$$= J(x_2; x_1)(x_2 - x_1)$$

where $j_i = \int_0^1 d\alpha \, \partial f / \partial x^i \big|_{x_2^1, \ldots x_2^{i-1}, (\alpha x_2^i - (1-\alpha)x_1^i), x_1^{i+1}, \ldots x_1^n}$.

The matrix $J(x_2, x_1)$ is not necessarily symmetric in its arguments nor is it in general a symmetric matrix. It can be seen from its construction that

$\lim_{x_2 \to x_1} J(x_2; x_1) = Df\big|_{x_1}$

so that when $f = D\lambda$

$\lim_{x_2 \to x_1} J(x_2; x_1) = D(D\lambda\big|_{x_1})\big|_{x_1}$

the Hessian matrix.

The nonlinear problem is then to solve the set of equations

$0 = f(x^*) = f(x) + J(x^*; x)(x^* - x)$

for x^*. If $J(x^*; x)$ were a known function of x, the solution would be given by

$x^* = x - J^{-}(x^*; x) f(x)$

where J^- is a particular generalised inverse of J (Rao and Mitra(1971))
As, of course, $J(x^*;x)$ is unknown all that can be hoped for is an approximation B to J or B^- to J^- so that

$$x_2 = x_1 - B^- f_1 \qquad (1.2.1)$$

is a better approximation to x^* than x_1 was.

By "better approximation" is usually meant $|x_2-x^*| < |x_1-x^*|$ for some norm $|.|$. From a practical point of view (and, in fact, the only thing it is possible to check) "better" means $\lambda(x_2) < \lambda(x_1)$. Functions λ for which $\lambda(x_2) < \lambda(x_1)$ implies that $|x_2-x^*| < |x_1-x^*|$ would certainly make convergence proofs easier. Broader classes of functions for which $\lambda(x)-\lambda(x^*)$ has some of the properties of $|x-x^*|$ could be expected to make some convergence proofs possible. Almost by definition, the set $\Gamma_\varepsilon = \{x/\ |x-x^*| \leqslant \varepsilon\}$ is bounded (Munkres (1975)), thus one possible class of functions is given by those for which the level sets $L_\varepsilon = \{x/\lambda(x) \leqslant \varepsilon\}$ are bounded. From the properties of the Euclidean metric, if x_3 is an element of the line segment (x_2,x_1) then

$|x_3-x^*| \leqslant \max\{|x_1-x^*|,|x_2-x^*|\}$.

A function for which

$\lambda(x_3) \leqslant \max[\lambda(x_1),\lambda(x_2)\}$, $x_3 \in (x_2,x_1)$

is said to be quasiconvex and this provides another useful class of functions.

The approximation 1.2.1 may be improved by a line search $x_2 = x_1 - \mu B^- f_1$ where μ is chosen so that $\lambda(x_2)$ is sufficiently less than $\lambda(x_1)$. This is a most natural hybrid between the method of solution of nonlinear equations and direct search techniques. Having found one improved value the procedure is iterated. The distinguishing feature of the different algorithms is the way in which the approximations B or B^- at each iteration are chosen. An important factor in the speed of convergence is the amount of work needed in each iteration. A trend which began with Fletcher's and Reeves'(1964) implementation of the conjugate gradient method

was to perform exact line searches, for this is necessary for the success of the conjugate gradient philosophy. If μ is taken to be unity the amount of work required is minimal; if an exact search is required the computing cost of each iteration could be high. A paper by Fletcher(1970) marks the recent tendancy to inexact line searches which give a sufficient decrease in the objective function for the expenditure of a moderate amount of work.

The amount of work will be reduced if the direction of search, $-\bar{B_1}f_1$, can be guaranteed, by suitable choice of $\bar{B_1}$, to be downhill; that is, a descent direction. From the Taylor series

$$\lambda(x_2) = \lambda(x_1) - \mu f_1^T \bar{B_1} f_1 + O(\mu^2 |\bar{B_1} f_1|^2)$$

it is clear that if $f_1^T \bar{B_1} f_1$ is nonzero then for sufficiently small μ the objective function can certainly be reduced. The reduction would be expected to vary as $|f_1^T \bar{B_1} f_1|^2 / |\bar{B_1} f_1|^2$. If $f_1^T \bar{B_1} f_1$ is positive the search can be restricted to positive μ with a consequent reduction in computer time. Thus, in order to speed convergence, a desirable property of the approximations $\bar{B}$ is that they are positive definite. With 1.2.1 in mind, a further restriction —μ is limited to a preassigned neighbourhood of unity— could be considered.

The different algorithms are classified by the way in which the approximations B or $\bar{B}$ are updated. They may depend on information at the previous step and may even depend on information gathered at all previous steps. The less information an update requires the less storage will be required. If the approximate is symmetric this could mean a substantial saving in storage for large problems. An important requirement of an algorithm is that it should not fail

due to updates becoming singular. For some algorithms there is a tendency that once an approximate becomes singular all subsequent approximations are singular and subsequent steps are restricted to a subspace of R^n (Broyden (1972)). Even if in theory the updates are positive definite, in practice they may become singular due to round off error and inexact line searches. The way to avoid this problem is, of course, to keep a check on the singularity of the update and make corrections when necessary. This in general could be a complicated, time consuming business unless the updates are stored in a fashion that makes it immediately apparent what their condition is. If the updates are symmetric they may be stored in terms of factors e.g. $B^- = L^T L$ where L is lower triangular and from the diagonal elements of L it is clear if there is a danger of singularity. A singularity can easily be cured by amending the offending diagonal element. Not only does storage in this form give a check on the condition of the update, it also allows one to easily check for positive definiteness. Symmetry of the update is, thus, a desirable property both from the point of view of storage, but perhaps more importantly, because of its factorisations. Recent work (Goldfarb(1976); Brodlie, Gourlay and Greenstadt(1973)) has been concerned with updating the factors directly.

Two of the oldest updates are described in the remaining part of this Chapter. The following two Chapters are devoted to methods which use progressively more assumptions about the linearity of the function to be optimised.

1.3 Newton's and the steepest descent approximation

If the initial point x_1 is 'near' to the optimum x^* and $J(x;x_1)$ is a slowly varying function of x,
$J(x_1;x_1) = Df|_{x_1} = DD\lambda|_{x_1}$,
the Hessian at x_1, could be expected to be a reasonable approximation to $J(x^*;x_1)$. This is Newton's approximation; it is symmetric but is only positive definite if x_1 is sufficiently close to x^*. This objection can be overcome by factorising and modifying the factors if required, as described in Section 1.2; this is called the modified Newton method.

Another objection to Newton's method is the requirement that the Hessian be determined at each step. This involves the evaluation of n^2 scalar functions, the equivalent of n evaluations of f, and this depending on the complexity of the functions could be a costly operation. The cost is not only in computer time but also manhours to evaluate the correct analytic expressions for the second derivative and computer code them. In a problem with a large number of variables this could be a formidable task. The latter problem can be overcome by finite difference approximations to the second derivative. This may be even more desirable when one realises that what is really required is an approximation to $J(x^*;x_1)$ the rows of which are given by $Df^i|_{\theta_i x^*+(1-\theta_i)x_1}$:

A problem that cannot be avoided with Newton's method is the need to solve a set of linear equations:

$$f(x_1) + J(x_1;x_1)(x_2-x_1) = 0$$

involving $O(n^3)$ arithmetic operations per iteration. (This of course can be done by factorising the Hessian which neatly ties in with any modification which is needed.) Many authors have suggested that in looking for alternatives to

Newton's method, one is trying to approximate the Hessian matrix and in consequence the approximation should be both symmetric and positive definite. It is clear from Section 1.2 that one is really trying to approximate $J(x^*;x)$ which in general is neither symmetric nor positive definite. What could be said to be reasonable is that the approximation B should tend to $J(x^*;x^*)$, the Hessian at x^* which is both symmetric and positive definite.

A long way from the optimum and without any prior knowledge, the most reasonable choice of B^{-} is the unit matrix, for this gives the search direction as the direction of steepest descent. This is the line along which the function initially decreases most rapidly. The steepest descent approximation is transparently symmetric and positive definite. In practice the function decreases rapidly at first. As the algorithm progresses to the second stage, advance along a steep sided gently descending valley, the search directions will be almost at right angles to the line of the valley floor. Thus, consecutive points tend to zigzag along the valley sides making little progress. What is apparently needed is an algorithm that starts out life as steepest descent and as it approaches the optimum develops into Newton's algorithm.

If the boundaries of the level sets of the function λ are concentric hyperspheres about the optimum, the steepest descent algorithm reaches the optimum in a single step. This suggests changing the coordinates so as to make the level sets as near as possible spherical. This is known as scaling. If a steepest descent step is made in coordinates y with $T^T y = x$ then in the x coordinates the step made is

$$x_2 - x_1 = -\mu T^T T f_1 .$$

This in effect is a symmetric positive definite approximation T^TT to $J(x^*;x)$. Since any symmetric positive definite matrix may be factored in the form T^TT, a step made with such an approximation can be thought of as a change of scale.

CHAPTER 2 Secant approximations

This Chapter begins the analysis of secant approximations which appear in various guises. The philosophy is to make a linear fit to known function values. Newton's algorithm can be seen to lie just at the edge of this broad definition as it is the limit in which the points where the value is sampled approach the intial point. When a difference approximation is made to the Hessian this implementation lies within the secant philosophy; the approximation is fitted to function values taken along the coordinate directions. Various other secant updates will be discussed.

2.1 The basic idea

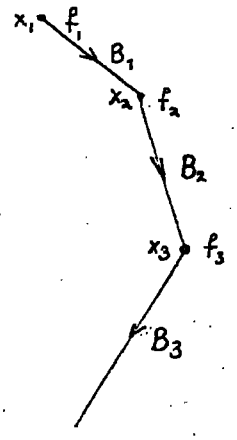
Suppose a step is generated by B_1 an approximate to J, $x_2 = x_1 - \mu B_1^{-1} f_1$; from Section 1.2 $\Delta f_1 = J_{21}\Delta x_1$, where $\Delta f_2 = f_2 - f_1$ and $\Delta x_1 = x_2 - x_1$. If x_1 and x_2 are close to the optimum and J is slowly varying $J(x^*;x_2)$ will be approximately equal to J_{21}. This suggests making the new approximate , B_2, fit the same equation as J_{21}: $\Delta f_1 = B_2\Delta x_1$.

As with Newton's algorithm this is hardly justifiable at large distances from the optimum. It does, however, present the possibility of starting an algorithm with the steepest direction and then making a minimal change from the unit matrix in order that the update fits the previous step.

The strategy of secant approximations is, then, to approximate J by linear fits to known function values. The number of previous steps which are fitted distinguish,

to some extent, the possible secant algorithms. Let the kth step be generated by B_k

$$B_k \Delta x_k = -\mu_k f_k$$



where $\Delta x_i = x_{i+1} - x_i$. This is shown symbolically in the Figure. The approximate B_{k+1} is chosen to fit the previous k steps

$$B_{k+1} \Delta X|^k = \Delta F|^k \qquad (2.1.1)$$

where $\Delta X|^k = [\Delta x_1, \Delta x_2, \ldots, \Delta x_k]$ and $\Delta F|^k = [\Delta f_1, \Delta f_2 \ldots \Delta f_k]$. The x_i will for the moment be assumed linearly independent. A general solution of 2.1.1 can be written in the form (Rao and Mitra (1971))

$$B_k = \Delta F^k (\Delta X|^k)^{-L} + C_k, \qquad (2.1.2)$$

where $(\Delta X|^k)^{-L}$ is a left inverse of $\Delta X|^k$ (not unique) and the null space of C_k contains the range of $\Delta X|^k$. A general form of left inverse is (Rao and Mitra (1971))

$$(\Delta X|^k)^{-L} = (Y_k \Delta X|^k)^{-1} Y_k,$$

where Y_k is a k×n matrix such that the rank of $Y_k \Delta X|^k$ is equal to the rank of $\Delta X|^k$ which is assumed to be k. If Z_k is a k×n matrix for which $\text{rank}(Z_k \Delta X|^k) = k$, $\Delta X|^k (Z_k \Delta X|^k)^{-1} Z_k$ is idempotent and therefore a projector. It projects onto the range of $\Delta X|^k$ along the null space of Z_k (the orthogonal complement of the range of Z_k^T). One form of C_k is then

$$C_k = H_k (I - \Delta X|^k (Z_k \Delta X|^k)^{-1} Z_k)$$

where H_k is arbitrary and the second factor projects onto the null space of Z_k along the range of $\Delta X^{|k}$.

A canonical choice for 2.1.2 is

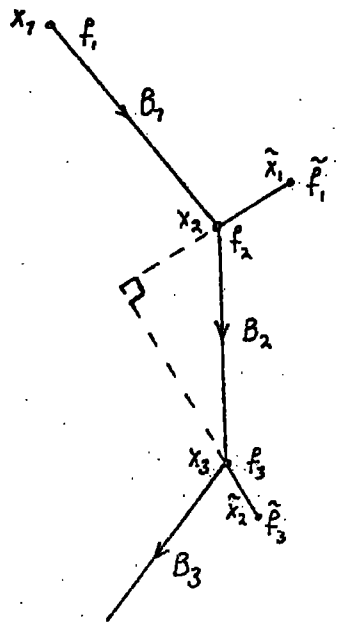
$$B_{k+1} = B_1 + (\Delta F^{|k} - B_1 \Delta X^{|k})(\Delta X^{|k T} \Delta X^{|k})^{-1} \Delta X^{|k} \qquad (2.1.3)$$

that is $Y_k = Z_k = \Delta X^{|k}$ and $H_k = B_1$. This represents the minimal departure from B_1, the initial approximation, for if c is orthogonal to the range of $\Delta X^{|k}$ then $B_{k+1}c = B_1 c$. This is the generalised secant approximation of Barnes (1965). It is neither symmetric nor positive definite.

If 'n' linearly independent steps Δx_i can be taken with the secant approximation 2.1.2 the inverse $(\Delta X^{|n})^{-L}$ is unique and $C_k = 0$. If, moreover, the system of equations $f = 0$ is affine J is a constant and B_n becomes equal to J after n steps. A further step gives the exact solution. The n-step secant algorithm can be started with the steepest descent step and if applied to an affine equation, as described, the final step is the Newton step. For the nonlinear case, having made a fit to the previous n steps the algorithm can be restarted or the n directions can be systematically updated by more recent information.

The difficulty with the n-step secant approximation is that it cannot in general be guarenteed that the steps $\{\Delta x_i\}$ generated by the $\{B_i\}$ will be linearly independent. One way to avoid this problem is to take side evaluations. This is shown symbolically in the Figure over the page. The steps are generated as before

$$B_k \Delta x_k = -\mu_k f_k$$



but B_{k+1} is chosen to fit side evaluations

$$B_{k+1}\Delta\tilde{X}|^k = \Delta\tilde{F}|^k$$

where $\Delta\tilde{x}_i = x_{i+1} - \tilde{x}_i$ and $\Delta\tilde{f}_i = f_{i+1} - \tilde{f}_i$. The $\Delta\tilde{x}_i$ may be chosen as before to be the steps Δx_i or members of a linearly independent set $\{r_i\}$, say. If the latter is the case, twice as many function evaluations are needed. The update B_{k+1}, as above, is given by

$$B_{k+1} = \Delta\tilde{F}|^k(Y_k\Delta\tilde{X}|^k)^{-1}Y_k + \tilde{C}_k \qquad (2.1.4)$$

where the rank of $Y_k\Delta\tilde{X}|^k$ is k and the null space of $\tilde{C}_k$ contains the range of $\Delta X|^k$. An obvious choice it to take $\{\Delta x_i\}$ as the coordinate directions $\{e_i\}$ possibly scaled, then $B_n = \Delta\tilde{F}(\Delta\tilde{X})^{-1}$ where $(\Delta\tilde{X})^{-1}$ is diagonal. This is a difference approximation to the Jacobian, albeit a poor one. (The difference approximation to the Jacobian can be made either in the rows or the columns, as here. The difference approximation here is not, however, taken at one point but is gradually built up. For example, starting with the unit matrix the difference approximation to $\partial f/\partial x^1|_{x_2}$ can be found and this used to replace the first column of the unit matrix; a step is then generated by this new approximation. A difference approximation to $\partial f/\partial x^2|_{x_3}$ is made and this used to replace the second column. and so on.)

In general the updates of the above form are not symmetric nor positive definite. Updating those n-step secant algorithms which are symmmetric and positive definite(if B^{-} is to be

symmetric positive definite then so must its inverse B), if such exist, is likely to be a complicated business. For most of the rest of this Chapter interest will be in those algorithms which fit only the previous step. Forms of these exist which are symmetric and positive definite. The final section in Chapter Three will return to algorithms which solve affine equations in n+1 steps.

If the update has only to fit the previous step, a general form is

$$B_2 = \Delta f y^T / y^T \Delta x + C_1 \qquad (2.1.5)$$

where $C_1 \Delta x = 0$. As before, one choice of C_1 is $H_1(I - \Delta x z^T / z^T \Delta x)$ with H_1 arbitrary. If $H_1 = B_1$, the initial approximate, a matrix multiplication is avoided since $B_1 \Delta x = -\mu \Delta f$. When $y = z$, the initial approximate is only modified by a matrix of rank one:

$$B_2 = B_1 + (\Delta f - B_1 \Delta x) z^T / z^T \Delta x \qquad (2.1.6)$$

The canonical choice $y = z = \Delta x$, gives Broyden's (1965) algorithm, for which $B_2 c = B_1 c$ for any c orthogonal to the previous search direction Δx. The Broyden (1965) update is neither symmetric nor positive definite. If the initial approximation is poor the algorithm is unstable (it generates directions which are not descent directions). Powell (1970) has given a symmetric adaptation of Broyden's update in which B^- is updated by a matrix of rank two.

A symmetric form of 2.1.6 is obtained by taking $z = (\Delta f - B_1 \Delta x)$, which gives the complement of Davidon's (1968) symmetric rank one update. Not only is it the complement (Δf is interchanged with Δx and B with B^-), it is the inverse of Davidon's (1968) as can be seen in Section 2.2 using the Sherman-Morrison formula.

This algorithm is not positive definite and is notoriously unstable. It is, however, attractive for reasons explained in Sections 2.2 and 3.3.

If $C_1 = B_1(I - \Delta x z^T/z^T\Delta x)$ the most general symmetric update of the form 2.1.5 is

$$B_2 = B_1 + \Delta f\Delta f^T/\Delta f^T\Delta x + B_1\Delta x\Delta x^T B_1/\Delta x^T B_1\Delta x + \zeta(\Delta f/\Delta f^T\Delta x - B_1\Delta x/\Delta x^T B_1\Delta x)(\Delta f/\Delta f^T\Delta x - B_1\Delta x/\Delta x^T B_1\Delta x)^T$$

where ζ is arbitrary. This is the complementary single parameter rank two Broyden(1967) family. This family is the family of inverses of the Broyden(1967) family but the inverse of a particular member is not equal to its complement. A subclass of this family has the potential for positive definiteness (provided the line searches are carried out to a prescribed accuracy) as will be discussed in Section 2.2.

2.2 The complementary secant approximations

To find the next step in the secant approximation, as with Newton's approximation, it is necessary to solve a set of linear equations. This in general requires $O(n^3)$ operations. (An exception is the implementation by Gill and Murray (1972): the approximate B is stored in terms of its Cholesky factors and these are updated directly.) If B_2 and B_1 are nonsingular approximations which differ by an elementary matrix (rank one), 2.1.5, then the Sherman-Morrison formula gives a simple expression for $\bar{B_2}$, the updated approximate to $\bar{J}$, in terms of the initial approximate $\bar{B_1}$ to $\bar{J}$:

$$\bar{B_2} = \bar{B_1} + (\Delta x - \bar{B_1}\Delta f)z^T\bar{B_1}/z^T\bar{B_1}\Delta f.$$

The update as might be expected fits the previous step,

$$\bar{B_2}\Delta f = \Delta x.$$

This suggests that instead of making a linear fit to J, a linear fit to J^- could be made. That is, for the k step complementary secant approximation B^-_{k+1} is determined by fitting the k previous steps

$$\Delta X^{|k} = B^-_{k+1}\Delta F^{|k}$$

Then as for the usual secant approximation

$$B^-_{k+1} = \Delta X^{|k}(Y_k\Delta F^{|k})^{-1}Y_k \quad + C_k$$

where the rank of $Y_k\Delta F^{|k}$ is k and the null space of C_k contains the range of $\Delta F^{|k}$. It is clear that little is to be achieved by taking side evaluations since if the Δx_i are taken to be linearly independent there is no guarantee the the Δf_i will be linearly independent.

The one step complementary secant update is

$$B_2^- = \Delta x y^T/y^T\Delta f + C_1 \qquad (2.2.1)$$

where $C_1\Delta f = 0$. If $C_1 = B_1^-(I - \Delta f z^T/z^T\Delta f)$ a matrix multiplication is not avoided as it was for the one step secant update. However, $0(n^2)$ operations are needed for matrix multiplication as opposed to $0(n^3)$ for solving a system of linear equations. Taking $z = y$ gives the rank one updates and the canonical choice $z = y = \Delta f$ is the complement of Broyden's(1965) algorithm

$$B_2^- = B_1^- + (\Delta x - B^-\Delta f)\Delta f^T/\Delta f^T\Delta f$$

for which $B_2^-c = B_1^-c$ for all c orthogonal to Δf. This, like its counterpart is neither symmetric nor positive definite.

The symmetric rank one update, Davidon(1968)

$$B_2^- = B_1^- + (\Delta x - B_1^-\Delta f)(\Delta x - B_1^-\Delta f)^T/(\Delta x - B_1^-\Delta f)^T\Delta f$$

is attractive because, for an affine function, not only does it fit the *immediately* previous step, it fits all previous steps. This will be discussed further in Section 3.3. As stated above the algorithm is not stable but various implementations have been proposed which greatly improve its reliability.

A symmetric rank two form of 2.2.1 is the single parameter rank two Broyden (1967) family.

$$B_2^- = B_1^- + \Delta x\Delta x^T/\Delta x^T\Delta f - B_1^-\Delta f\Delta f^T B_1^-/\Delta f^T B_1^-\Delta f + \xi(\Delta x/\Delta x^T\Delta f - B^-\Delta f/\Delta f^T B^-\Delta f)(\Delta x/\Delta x^T\Delta f - B^-\Delta f/\Delta f^T B^-\Delta f)^T \quad (2.2.2)$$

This family includes the most widely used optimization algorithms such as DFP (Davidon-Fletcher-Powell (1963)) with $\xi = 0$, and BFGS (Broyden-Fletcher-Goldfarb-Shanno (1970)) with $\xi = \Delta f^T B_2^- \Delta f$. It is a subclass of the Huang (1970) family: members of the Huang family solve an affine equation in n+1 steps when exact line searches are used. The reason for this is that they generate a set of conjugate directions . The Huang family is generalised in Section 3.1. Dixon (1972) has shown that a subclass of the Huang algorithms, which includes the Broyden family, generate the same sequence of points when exact line searches are made. If exact line searches are performed, the subclass of the Broyden family with $\xi \geq 0$ are positive definite. For inexact line searches the same subclass is positive definite if and only if $\Delta f^T\Delta x > 0$.

The symmetric rank one algorithm (Davidon (1963)) solves an affine equation in n+1 steps with arbitrary step size. The drawback with the algorithm is that is that the approximations to J^- are not positive definite. The subclass of the Broyden (1967) family for which $\xi \geq 0$ gives positive definite updates if the

line searches are performed sufficiently accurately so that $\Delta f^T \Delta x > 0$. To reach this degree of accuracy could be expensive. Only if the line searches are accurate will this class solve an affine equation in a finite number of steps. What would be desirable is a symmetric positive definite algorithm which solves an affine equation in a finite number of steps with arbitrary step length. Davidon(1975) has produced just such an algorithm which has a form similar to 2.2.2:

$$B_2^- = B_1^- + zz^T/z^Ty - B_1^-yy^TB_1^-/y^TB_1^-y + \phi(z/z^Ty - B_1^-y/y^TB_1^-y)(z/z^Ty - B_1^-y/y^TB_1^-y)^T$$

where $z = \Delta x - B_1^-\Delta f + B_1^-y$ and $y^Tz = y^T\Delta x = \Delta f^Tz$. This can be rearranged into the form 2.2.1: $B_2^- = \Delta xz^T/z^T\Delta f + C_1$ where

$$C_1 = B_1^-(I - \Delta fz^T/z^T\Delta f) + (B_1^-y + \phi z/z^T\Delta f - \phi B_1^-y/y^TB_1^-\Delta f)(z/z^T\Delta f - B_1^-y/y^TB_1^-\Delta f)^T$$

showing that it is a complimentary one step secant algorithm. Davidon chooses y so that B_2^- can fit all previous steps for an affine function. Exactly how this can happen is investigated in Chapter Three. The parameter ϕ is chosen so that the update is positive definite and the condition number of $B_1B_2^-$ is minimised.

CHAPTER 3 Conjugate directions

This Chapter examines the use of conjugate directions in optimisation with special interest in their relation to and use within the secant framework. The analysis is more general than is usual when discussing the use of conjugate directions in optimisation. It is argued that most of these generalisations are unlikely to be helpful in producing algorithms. Section 3.3 does, however, give some ideas for constructing algorithms with the same properties as Davidon's(1975) algorithm.

3.1 Conjugacy

The secant approximations involve fitting a hyperplane to known function values. The approximations in the next section assume linearity in the equations to be solved: $f = Ax - b = 0$. The solution of this set of equations is essentially the problem of finding an inverse for the matrix A. The inverse of certain matrices, for example lower triangular matrices, are relatively easy to find. Methods which find the generalised inverse in a finite number of steps depend on reducing the matrix to a form for which the inverse is easy to find. The reduction of a matrix, A, is essentially the problem of finding pre and post multiplying matrices R^T and S such that R^TAS has zeros in prescribed positions. This can alternatively be thought of as choosing vectors r and s so that $r^TAs = 0$. If this condition is true s is said to be A-conjugate to r or the pair (s,r) are said to be A-conjugate. If also (r,s) is A-conjugate, or equivalently (s,r) is A^T-conjugate, the pair (s,r) are said to be A-biconjugate. Orthogonality is clearly a special case of conjugacy.

3.2 Reduction to lower triangular form

Suppose that A, $S = [s_1, s_2, \ldots\ldots s_n]$ and $R = [r_1, r_2, \ldots\ldots r_n]$ are nonsingular matrices and $R^T AS = L$ is a lower triangular matrix, that is $r_i^T A s_j = 0$ $i < j$, then motivated by the usual method of conjugate directions Stewart(1973) gives the following algorithm for solving $Ax^* - b = 0$. Given arbitrary x_1, the sequence $\{x_k\}_{k\ 1}^{n+1}$ generated by

$$x_{i+1} = x_i - s_i r_i^T f_i / r_i^T A s_i \qquad (3.2.1)$$

terminates on x^*. By the assumptions of nonsingularity, the algorithm can always be carried to completion. The easiest way to understand the performance of the algorithm is the following. The matrix can be regarded as a linear transformation from R^n to R^n. Let the domain of A be equipped with a basis formed from the columns of S and the range with a basis formed from the columns of $(R^T)^{-1}$. With these bases the transformation is given by the matrix L. If $x_I = 0$ the algorithm is nothing more than the usual recursive method for solving the triangular system of equations:

$$Ly = z, \quad x = Sy, \quad z = R^T b.$$

A second proof that is pertinent to optimisation is the following. It is easy to show inductively that f_i is orthogonal to $r_1, r_2, \ldots r_{i-1}$ and since the range of R is R^n, f_{n+1} must be identically zero. From the point of view of minimising the quadratic

$\lambda(x) = \frac{1}{2}x^T Ax - bx$

where A is symmetric positive definite, after $i \leqslant n$ steps λ

is minimised on the variety through the current point spanned by $r_1, r_2, \ldots . r_{i-1}$ (since λ is strictly convex). This, of course, is the emergence of the parallel subspace theorem (which is used in some algorithms for optimisation without derivatives):

> If x and y are minima of the quadratic λ on parallel subspaces (e.g. varieties generated by $r_1, r_2, \ldots . r_{i-1}$) then x-y (proportional to s_i) is A-conjugate to the subspace.

In the usual conjugate direction method (e.g. Luenberger(1973)) the step directions s_i serve the additional purpose of describing the variety containing the minimum. The above generalisation, thus, provides a second set $\{r_i\}_{i=1}^{n}$ to delineate the variety containing the minimum.

The second proof can be restated in a way which will prove useful later. It can be seen that

$f_{k+1} = F_k f_1$ where

$$F_k = (I - As_k r_k^T / r_k^T As_k)(I - As_{k-1} r_{k-1}^T / r_{k-1}^T As_{k-1}) \ldots . (I - As_1 r_1^T / r_1^T As_1).$$

From conjugacy it follows that

$$(As_i r_i^T / r_i^T As_i) F_k = 0 \quad \text{for } i \leq k \qquad (3.2.2)$$

so that F_k is idempotent ($F_k F_k = F_k$). That is, F_k is a projector. From 3.2.2 the range of F_k is the orthogonal complement of the range of $R^{|k}$. In a similar way it can be shown that the range of F_k^T is the orthogonal complement of the range of $AS^{|k}$. Since any projector projects along the orthogonal complement of the range of its transpose, F_k must project along the range

of $AS|_k$. That is,

$$F_k = I - AS|_k(R|_{kT}AS|_k)^{-1}R|_{kT}$$
$$= I - AS|_k(L\overline{|_k})^{-1}R|_{kT}$$
$$F_n = 0.$$

The residues $e_i = x_i - x^*$ are related in a similar fashion $e_{k+1} = E_k e_1$ where

$$E_k = (I - s_k r_k^T A/r_k^T A s_k)(I - s_{k-1} r_{k-1}^T A/r_{k-1}^T A s_{k-1})\ldots(I - s_1 r_1^T A/r_1^T A s_1)$$

is a projector which projects onto the orthogonal complement of the range of $A^T R|_k$ (which by conjugacy is the same as the range of $S^{n-k}|$) along the range of $S|_k$. (The proof is the same as for F_k and is given in detail by Stewart(1973).) That is,

$$E_k = I - S|_k(L\overline{|_k})^{-1}R|_{kT}A$$
$$= S^{n-k}|(S^{-1})\underline{^{n-k}}$$
$$E_n = 0.$$

The algorithm 3.2.1 is, of course, useless unless matrices R and S can be found. Chapter One suggests taking s_i proportional to $\bar{B_i}f_i$ when 3.2.1 becomes

$$x_{i+1} = x_i - r_i^T f_i \bar{B_i} f_i / r_i^T A \bar{B_i} f_i. \qquad (3.2.3)$$

The requirement $r_j^T A s_k = 0$ for $j < k$ implies that

$$r_j^T A \bar{B_k} f_k = 0, \text{ for } j < k.$$

The algorithm is constructed so that $r_j^T f_k = 0$, $j < k$, as shown above, which suggests that

$$B_k^{-T} A^T r_j = \rho r_j, \quad j < k \tag{3.2.4}$$

for some constant ρ. If this is the case

$$\Delta B_k^{-T} A^T r_j = 0, \quad j < k$$

where $\Delta B_k^- = B_{k+1}^- - B_k^-$, and

$$\Delta B_k^{-T} A^T r_k = \rho r_k - B_k^{-T} A r_k.$$

One possibility is then

$$B_{k+1}^{-T} = B_k^{-T} + \rho r_k u_k^T / u_k^T A^T r_k - B_k^{-T} A^T r_k t_k^T / t_k^T A^T r_k \tag{3.2.5}$$

for some u_k and t_k in the orthogonal complement of $\text{ran}(A^T R^{|k-1})$. Since s_i is proportional to Δx_i

$$\Delta f_i^T B_k^{-T} A^T r_j = \Delta x_i^T A^T B_k^{-T} A^T r_j = \rho \Delta x_i^T A^T r_j = 0, \quad j < k, i \text{ , so}$$

Δx_k and $B_k^- \Delta f_k$ are both members of the orthogonal complement of the range of $A^T R^{|k-1}$ (which is the range of $S^{n-k+1|}$). If the update is just to depend on the information from the previous step (to minimise storage) then it is reasonable to take u_k and t_k as linear combinations of Δx_k and $B_k^- \Delta f_k$. With this choice 3.2.5 is a generalisation of the Huang (1970) class of updates. Setting $r_k = \Delta x_k$ reproduces the Huang class.

Condition 3.2.4 rewritten as

$$B_k^{-T} \Delta \tilde{f}_j = \Delta x_j \quad , \quad j < k$$

where $r_j = \Delta\bar{x}_j$ and $\Delta\tilde{f}_j = \rho^{-1}A^T r_j$, can be compared with fitting of the hyperplane in the complementary secant approximation with side evaluations, Section 2.2,

$$\bar{B}_k\Delta\tilde{f}_j = \Delta\bar{x}_j, \qquad j < k .$$

Thus, the conjugate direction algorithm 3.2.3, 3.2.5 can be seen to fit a hyperplane to the transpose of $\rho^{-1}A$ using side evaluations along r_j. If A and $\bar{B}_k$ are symmetric and $\rho = 1$ this is a complementary n-step secant approximation with side evaluations.

Huang (1970) showed that all the Huang class generate the same directions when applied to an affine equation. That is, the s_k are the same up to modulous. The following shows that this is true for all members of the above generalised Huang class. Firstly, since $r_j^T f_k = 0$ for $j < k$

$$\bar{B}_{k+1}f_{k+1} = (I - t_k r_k^T A/r_k^T A t_k)(I - t_{k-1} r_{k-1}^T A/r_{k-1}^T A t_{k-1}) \ldots\ldots \bar{B}_1 f_{k+1}$$

so that the directions are independent of u_k and ρ. Because of this ρ could be iteration dependant, ρ_k, without affecting the conclusion. Since t_i is a linear combination of Δx_i and $\bar{B}_i\Delta f_i$ it is clear that t_i is a linear combination of $\bar{B}_1 f_1, \bar{B}_1 f_2, \ldots\ldots \bar{B}_1 f_i$ so that s_k is also a linear combination of $\bar{B}_1 f_1, \bar{B}_1 f_2, \ldots\ldots \bar{B}_1 f_k$. This can be written as

$$R^T A \bar{B}_1 F U^{-1} = L \quad \text{or} \quad R^T A \bar{B}_1 F = LU$$

where $F = [f_1, f_2, \ldots\ldots f_n]$ and U^{-1} is an upper triangular matrix. That is, $R^T A \bar{B}_1 F$ can be factored into the product of upper and lower triangular matrices. Since R, A and $\bar{B}_1 F$ are assumed nonsingular, U is uniquely determined up to the scaling of its rows

(Householder(1964) §1.4). Thus, given s_k a linear combination of $B_1^{-}f_1$, $B_1^{-}f_2$,.....$B_1^{-}f_k$ it is unique up to modulous. Apart from scaling, then,

$$s_1 = -B_1^{-}f_1$$
$$s_k = -B_1^{-}f_k + \Sigma_{j\ 1}^{k-1}\sigma_{kj}s_j \quad k > 1.$$

The requirement $r_j^T A s_i = 0$ for $j < i$ implies that $\sigma_{ij} = r_j^T A B_1^{-}f_i / r_j^T A s_j$:

$$s_k = -(I - \Sigma_{j\ 1}^{k-1} s_j r_j^T A / r_j^T A s_j) B_1^{-}f_i$$

$$= -E_{k-1}^T B_1^{-}f_k$$

where E_k is the projection matrix defined earlier in this section. The projector E_k^T projects onto the orthogonal complement of $S^{|k}$ along the range of $A^T R^{|k}$. This shows the relationship to the projection implementation of the conjugate direction method e.g. Zoutendijk(1960).

It is difficult to justify the implementation of the above generalised conjugate direction algorithm, 3.2.1, as a hybrid search method. Only in the case $r_i = s_i$, the usual conjugate direction method, is the step taken to the minimum of a quadratic along the search direction. Even then, the search has to be exact to maintain conjugacy and achieve termination in n+1 steps. Van Wyk(1977) suggests line searches along the side steps r_i. This seems hardly likely to accelerate the convergence for a general nonlinear function as the searches are not in the step directions.

A feature of the generalised conjugate direction algorithm is that the choice of r_k can be deferred until after s_k has been determined.

Stewart(1973) shows how different choices of R lead to other well known matrix reductions. The usual method of conjugate directions, $r_i = s_i$, leads to diagonal reduction. Another way to achieve diagonal reduction is described in the remainder of this section.

Given matrices A and $B_1^{-}F$ are nonsingular, let V be another nonsingular matrix such that $V^TAB_1^{-}F$ can be factored into LDU where L is lower triangular, U is upper triangular and D is diagonal. Then

$$R^TAS = D$$

where $R^T = L^{-1}V^T$ and $S = B^{-}FU^{-1}$. That is, the pairs (s_i, r_j) are biconjugate for $i \neq j$ with r_k a linear combination of $v_1, \ldots\ldots v_k$. Again S and R are determined uniquely up to scaling (Householder(1964). Thus, up to modulous s_k is determined as in 3.2.6 and r_k can be given by a similar expression:

$$r_1 = v_1$$

$$r_k = v_k - \sum_{j=1}^{k-1}\rho_{kj}r_j = (I - \sum_{j=1}^{k-1} r_j s_j^T A^T / r_j^T A s_j)v_k = F_{k-1}^T v_k .$$

In addition to the sequence generated by 3.2.1, the sequence generated by

$$\bar{x}_{k+1} = \bar{x}_k - \bar{f}_k^T s_k r_k / r_k^T A s_k \qquad (3.2.7)$$

also terminates on x^*, when A is symmetric, for arbitrary initial point $\bar{x}_1$. This is because for symmetric A, S^TAR is lower triangular. If $B_1^{-} = I$ (i.e. the initial step is the

the steepest descent) and $v_k = \bar{f}_k$ then since $f_k(\bar{f}_k)$ is orthogonal to $r_1, r_2, \ldots r_{k-1}$ $(s_1, s_2, \ldots s_{k-1})$ which are linear combinations of $\bar{f}_1, \bar{f}_2, \ldots \bar{f}_{k-1}$ $(f_1, f_2, \ldots f_{k-1})$ it follows that $\bar{f}_i^T f_k = 0$ $(\bar{f}_k^T f_i = 0)$ for $i < k$. Hence $\sigma_{ij} = \rho_{ij} = 0$ for $i-1 \neq j$ so in this situation the algorithms 3.2.6 and 3.2.7 reduce to

$$s_{i+1} = -f_{i+1} - \bar{f}_{i+1}^T f_{i+1} s_i / \bar{f}_i^T s_i$$

$$r_{i+1} = -\bar{f}_{i+1}^T - \bar{f}_{i+1}^T f_{i+1} r_i / r_i^T f_i$$

After a little translation in coding this can be seen to be Fletcher's(1975) biconjugate gradient algorithm which is a generalisation of the Hestenes-Stiefel(1952) conjugate gradient algorithm. A discussion of biconjugation applied to affine problems is given in Wilkinson's(1965 Ch.6) book. It is to be noted that this type of algorithm suffers from numerical instabilities $(\bar{f}_i \neq f_i)$ because of lack of symmetry. It is hardly likely that the generalisation will prove useful for the nonlinear case.

3.3 Reduction to upper triangular form

The n-step complimentary secant update

$$B_{k+1}^- = \Delta X|_k (Y_k \Delta F|_k)^{-1} Y_k + B_1^- (I - \Delta F|_k (Z_k \Delta F|_k)^{-1} Z_k) + H_k \qquad (3.3.1)$$

where $H_k \Delta F|_k = 0$ is somewhat attractive because with exact arithmatic it will solve an affine system at the (n+1)st step without exact line searches. The one step complementary secant update

$$B_{k+1}^- = \Delta x_k y_k^T / y_k^T \Delta f_k + B_k^- (I - \Delta f_k z_k^T / z_k^T \Delta f_k) + G_k \qquad (3.3.2)$$

where $G_k\Delta f_k = 0$, is attractive because of its simplicity: this allows one to identify those members which are symmetric and positive definite *e.g.* some members of the Broyden family. Furthermore, another subfamily the Huang class, which contains the Broyden family, are able to solve an affine system at the (n+1)st step with accurate line searches. It is natural to ask if there is a subclass which will solve an affine system in a finite number of steps but without exact line searches. That is, under what choices of y_k and z_k will 3.3.2 generate 3.3.1. Tracing 3.3.2 back to B_1^- gives

$$B_{k+1}^- = \Delta X^{|k} Q^{|kT} + B_1^- P_{1k} + \Sigma_{i=2}^{k} G_{i-1} P_{ik} + G_k$$

where

$$P_{jk} = (I - \Delta f_j z_j^T / z_j^T \Delta f_j)(I - \Delta f_{j+1} z_{j+1}^T / z_{j+1}^T \Delta f_{j+1}) \ldots\ldots (I - \Delta f_k z_k^T / z_k^T \Delta f_k)$$

and

$$Q^{|k} = \left[P_{2k}^T y_1 / y_1^T \Delta f_1, P_{3k}^T y_2 / y_2^T \Delta f_2, \ldots\ldots, P_{kk}^T y_{k-1} / y_{k-1}^T \Delta f_{k+1}, y_k\right]$$

The discussion in Section 2.1 shows that P_{1k} must be a projector but from its construction

$P_{1k} = I - \Delta F^{|k} U_k Z^{|kT}$ where $Z = [z_1, z_2, \ldots\ldots z_n]$ and where U_k is upper triangular. Thus $U_k = (Z^{|kT} \Delta F^{|k})^{-1}$ and since the inverse of an upper triangular matrix is triangular it follows that a necessary condition for 3.3.2 to generate 3.3.1 is that $Z^{|kT} \Delta F^{|k}$ is upper triangular. It can be shown that P_{jk} is a projector if $Z^T \Delta F$ is upper triangular i.e. if $z_j^T \Delta f_i = 0$ for $j > i$. The proof follows that which showed F_k was a projector Section 3.2. The matrix P_{jk} projects onto the orthogonal complement of the subspace generated by $z_j, \ldots\ldots z_k$ along the the subspace generated by $\Delta f_j \ldots\ldots \Delta f_k$. Note that, because of orthogonality the range of P_{jk} includes the span of $\Delta f_1, \ldots\ldots \Delta f_{j-i}$.

From Section 2.1 it is clear that $Q^{|kT}$ must be a left inverse of $\Delta F^{|k}$ or alternatively

$$y_j^T P_{j+1,k} \Delta f_i / y_j^T \Delta f_i = \begin{cases} 1 & i > j \\ 0 & \text{otherwise} \end{cases} .$$

If $Z^T \Delta F$ is upper triangular then as described P_{jk} is a projector and

$$P_{j+1,k} \Delta f_i = \begin{cases} 0 & 1 \leqslant j < i \leqslant k \\ \Delta f_i & k > j \geqslant i \geqslant 1 \end{cases} .$$

Thus, if Q^T is to be a left inverse of ΔF, $y_j^T \Delta f_i = 0$ for $j > i$ or $Y^T \Delta F$ must be upper triangular. Thus necessary conditions for 3.3.1 to be generated from 3.3.2 are that $Z^T \Delta F$ and $Y^T \Delta F$ are upper triangular. It is easy to convince oneself that these conditions are also sufficient.

Section 3.2 shows how, given a matrix T, to construct a matrix S, with s_k a linear combination of $t_1, t_2, \ldots . t_k$ so that R and S reduce A to lower triangular form. Inspired by this it should be possible, given a matrix $V = [v_1, \ldots . v_n]$, to find a matrix Y, with y_i a linear combination of $v_1, \ldots . v_i$, such that $Y^T \Delta F$ is upper triangular. If $y_1 = v_1$ and

$$y_k = (I - y_{k-1} \Delta f_{k-1}^T / \Delta f_{k-1}^T y_{k-1}) \ldots \ldots \ldots (I - y_1 \Delta f_1^T / \Delta f_1^T y_1) v_k \qquad (3.3.3)$$

then by induction $z_k^T \Delta f_j = 0$ for $k > j$. This is precisely the technique adopted by Davidon (1975). Davidon, however, chooses v_k to be a vector z_{k-1} which for an affine system is already orthogonal to $\Delta f_1, \Delta f_2, \ldots . \Delta f_{k-2}$ then y_k becomes

$$y_k = (I - y_{k-1} \Delta f_{k-1}^T / \Delta f_{k-1}^T y_{k-1}) z_{k-1} . \qquad (3.3.4)$$

For the affine system of equations in Section 3.2 (the method of conjugate directions) the problem is to reduce A to lower triangular form. For an affine system the problem here, the n-step secant approach, is to find a matrix Z such that $Z^T\Delta F = Z^TA\Delta X$ is upper triangular. That is, it is necessary to reduce A to upper triangular form. Suppose there is a matrix V such that there is a factorisation of $V^TA\Delta X$ into LU where L is lower triangular and U is upper triangular then $Z^T = L^{-1}V^T$. By an arguement similar to the one in Section 3.2, Z is unique up to scaling of its columns.

Suppose an update B^-_{k+1} when applied to an affine system fits the k previous steps

$$B^-_{k+1}\Delta F|^k = B^-_{k+1}A\Delta X|^k = \Delta X|^k$$

then for any vector v

$$v^T(AB^-_{k+1} - I)A\Delta X|^k = 0$$

so that one choice of z_{k+1} is

$$z_{k+1} = (I - B^{-T}_{k+1}A^T)v_{k+1}$$

When A is symmetric, as it is in optimisation, the natural choice is $v_{k+1} = \Delta x_{k+1}$ for then

$$z_{k+1} = \Delta x_{k+1} - B^{-T}_{k+1}\Delta f_{k+1}. \qquad (3.3.5)$$

This is the choice used by Davidon(1975). The symmetric rank one update (Davidon(1968)) also uses this form. Thus, both these

algorithms are n-step secant methods when applied to an affine function. An alternative to 3.3.5 is side evaluations

$$z_{k+1} = \Delta\tilde{x}_{k+1} - \bar{B}_{k+1}^{T}\Delta\tilde{f}_{k+1} \qquad (3.3.6)$$

or 3.3.6 can be used to generate a second orthogonal set.

A symmetric rank two form of 3.3.2 is (Powell(1977))

$$\bar{B}_{k+1} = \bar{B}_k + \theta_0(\Delta x_k - \bar{B}_k\Delta f_k)(\Delta x_k - \bar{B}_k\Delta f_k)^T$$

$$+ \theta_1((\Delta x_k - \bar{B}_k\Delta f_k)w_k^T + w_k(\Delta x_k - \bar{B}_k\Delta f_k)^T) + \theta_2 w_k w_k^T$$

where

$$\theta_0(\Delta x_k - \bar{B}_k\Delta f_k)^T\Delta f_k + \theta_1 w_k^T\Delta f_k = 1,$$

$$\theta_1(\Delta x_k - \bar{B}_k\Delta f_k)^T\Delta f_k + \theta_2 w_k^T\Delta f_k = 0$$

and w_k is **A**-conjugate to $\Delta x_1, \ldots\ldots \Delta x_{k-1}$. Any of the forms 3.3.3, 3.3.4 and 3.3.6 can be used for w_k. If θ_1 and θ_2 are zero this is just the symmetric rank one update(Davidon(1968)). The free parameter can be chosen so that the update is positive definite. For the Davidon(1975) algorithm $w_k = \bar{B}_k y_k$ so Davidon uses a complicated projection method to ensure conjugacy to previous steps.

CHAPTER 4 Summary and conclusions

Desirable general features of techniques for numerical optimisation of a function of a finite number of variables are:
rapid convergence;
robustness;
simplicity.
Minimal storage, which is useful for handling problems with a large number of variables, is to be considered as incorporated in robustness. In a large number of applications the function to be optimised is continuously differentiable which gives an additional means by which to achieve rapid convergence. In this situation the algorithm can be a hybrid between a line search and a method for solving nonlinear equations.

Methods for solving nonlinear equations are based to a greater or lesser extent on methods for solving affine equations. Most of these are in some way related to secant methods, Chapter Two. In the secant approach, an approximation to $J(x^*;x)$ or $J^-(x^*;x)$ (Chapter One) is made by making a linear fit to known function values. With Newton's method a linear fit to n+1 function values, in the vicinity of x, is made at one iteration. This is seen more clearly when a difference approximation is used for the Hessian. The simultaneous fit to n+1 points is then followed by a line search and this completes the iteration. In the n-step secant method, e.g. Barnes(1965) a linear fit is made to n+1 points but the fit is made over n iterations. The fits can be made to previous steps or to side evaluations. (Newon's method using difference approximations can be seen as

a fit to side evaluations made along the coordinate directions.)

The one-step secant algorithms, Chapter Two, are initially content with fitting just the previous step. This allows a great deal of arbitrariness but if the gradient is assumed affine the arbitrariness can be utilised so as to make an n-step secant fit. This involves the notion of conjugacy which is related to reduction of the fundamental (Hessian) matrix (Chapter Three). There are two basic approaches. The first, lower triangular reduction, makes a fit to the transpose of the inverse of the fundamental matrix, Section 3.2. This method requires exact step lengths. The second method, reduction to upper triangular form, is more closely related to the more usual secant method, Section 3.3. The step lengths for this method can be quite arbitrary. Note that when the matrix approximations are symmetric and the fundamental matrix is symmetric both methods lead to diagonal reduction and so appear similar. Thus, in general, the majority of algorithms which attempt to solve nonlinear equations are related to the secant method and a large number of these use conjugate directions in order to make a linear fit. This is the reason for the title of the Thesis and it is these algorithms which will be compared.

A successful algorithm should be able to cope adequately with each of the three stages of optimisation: descent into the valley, advance along the floor of the valley and search in the vicinity of the optimum, Chapter One. A long way from the optimum and without any previous information, the steepest descent direction is the natural initial choice. The update should differ from this by a minimal amount.

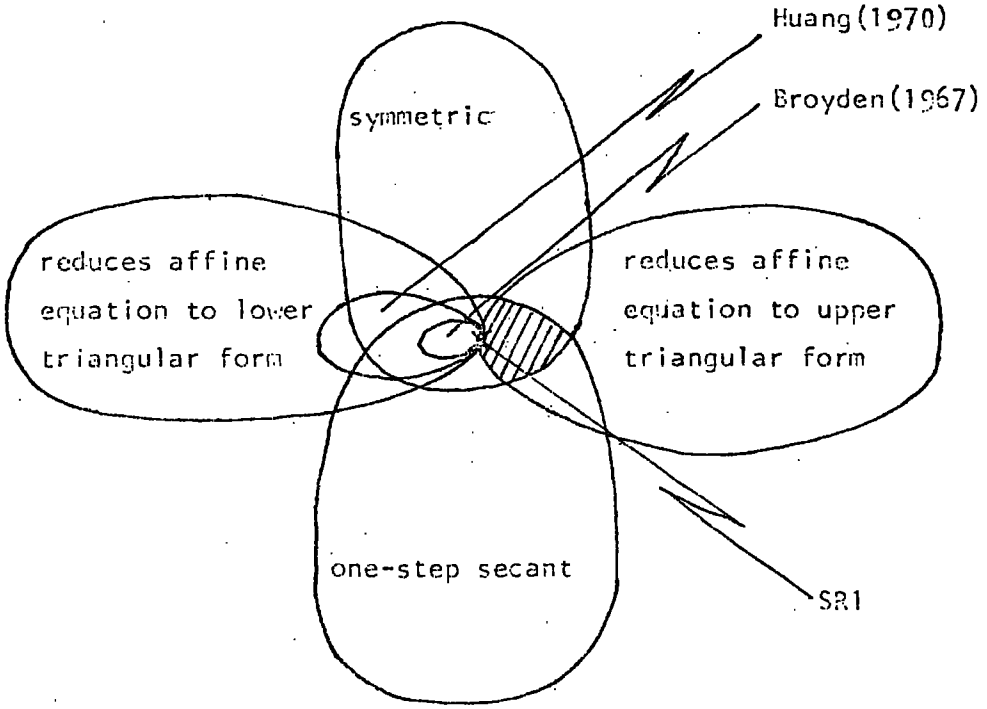
To ensure advance along the valley floor an improvement should be made in the objective function at each step. From Chapter

Two, this can be achieved if the approximating matrices are positive definite. Accurate line searches are costly so it is reasonable to terminate the search when a preassigned reduction in the objective function is achieved. The first procedure in the search should be to investigate the value of the function when the step parameter μ is set to unity and only if this fails to satisfy the search condition should the search be continued. This can be done by making a polynomial fit to the objective and gradient values.

Symmetry of the approximating matrices is attractive on a number of accounts. Firstly, for an affine equation lack of symmetry in the approximating matrix leads to numerical instability (Wilkinson(1965) Chapter 6). Secondly, a symmetric update can be thought of as a rescaling with a steepest descent step. With spherical symmetry the steepest descent step reaches the optimum. Thirdly, if the updates are symmetric only about half as much storage is required and this can be important for large problems. Finally, and perhaps the most important need for symmetry, the update can be stored and hopefully updated in factored form so that a check on nonsingularity and positive definiteness can be made and if necessary corrected.

In the final stage of the optimisation the gradient will be almost affine. The ability to solve an affine equation in a finite number of steps is thus an attractive feature of an algorithm. It should, however, be regarded as less important than positive definiteness and symmetry. "Quadratic termination is desirable if it is not achieved at the expense of stability" (Broyden(1972). Methods which solve affine equations exactly do so in general by deriving the fundamental matrix or its inverse. The approximating matrix then fits all previous steps or side evaluations.

Newton's algorithm only fits the previous steps when the equations to be solved are affine. The n-step secant algorithm fits the previous steps or side evaluations whether or not the equation to be solved is affine. Algorithms such as Davidon's(1975) and the Broyden family(1967) only fit all the previous steps when the equation to be solved is affine. The relationship between families of algorithms which are one-step secant is shown in the following Venn diagram.



Even though when the Hessian is symmetric and the updates are symmetric, algorithms which cause reduction to lower and upper triangular form have the same effect, namely diagonal reduction, their nature is very differnt. One requires precise step lengths but for the other arbitrary steps will do. The symmetric rank one update (Davidon(1968)) lies in both camps. Algorithms which lie in the shaded region, such as Davidon(1975), have up until recently been neglected.

In order to perform well at all stages of the optimisation process an algorithm should ideally start with the unit matrix (steepest descent) and progress through positive definite symmetric matrices until arriving at the fundamental matrix (Hessian) when applied to an affine system. Even though one is attempting to approximate $J(x^*;x)$, a matrix which is in general neither positive definite nor symmetric, positive definiteness and symmetry of the updates is of primary importance for they ensure reasonable performance at all stages of the optimisation process. Of secondary consideration is starting with steepest descent and deriving the Hessian or inverse Hessian for an affine equation. These enhance the algorithm in the initial and final stages. As described above accurate line searches are not very economical and if the algorithm has the above properties with arbitrary line searches this is an added bonus. If an algorithm derives the Hessian or its inverse for an affine equation then of course it fits the previous step. Algorithms which fit the previous step but do not derive the Hessian matrix may then be thought of as intermediate. They do tend to enhance convergence in the second and final stages but not as well as those algorithms which derive the Hessian.

Some important algorithms referred to in the text which are based on secant philosopy and conjugate directions are compared in the Table on the following page using the criteria of the above paragraph. Newton's modified algorithm refers to the incorporation of safeguards for positive definiteness. Although the algorithm does not in principle start with a steepest descent step, one or two such steps could be made so as to initialise the algorithm. The BFGS and DFP algorithms when used with inexact searches appear from experience to be more efficient than when used with exact searches. This shows that having to make exact line searches is not compensated by quadratic convergence. Note, however, the difference between these two algorithms and the steepest descent algorithm. An entry in the third row of the table obviously makes a significant difference in the performance so it must tend to reduce zigzagging behaviour which is a danger in the second stage (Section 1.3).

	Stage of optimisation	Modified Newton	Steepest Descent	Barnes Secant 1965	Broyden 1965	SR1 Davidon 1968	Powell 1970	BFGS, DFD exact search	BFGS,DFP inexact search $\Delta f^T \Delta x > 0$	Davidon 1975
Starts with steepest descent	1	✓	✓	✓	✓	✓	✓	✓	✓	✓
stable { pos def	1, 2, 3	✓	✓					✓	✓	✓
stable { sym						✓	✓			
One-step secant	2, 3				✓		✓		✓	
Derives Hessian or inverse for affine equation		✓		✓		✓		✓		✓
inexact line search		✓	✓	✓	✓	✓	✓		✓	✓

In the Broyden family the BFGS(Broyden(1970), Fletcher(1970), Goldfarb(1970) and Shanno(1970)) update when used with inexact line searches is reportedly the best. The reason for this is not completely clear but is related to conditioning. The definition of conditioning used by Shanno(1970) — the update that makes $z^T B^- z$ a maximum for arbitrary z is optimal — picks out the BFGS update. This, however, is not the usual definition of the condition number of a matrix. Davidon's(1975) definition is much closer; he tries to minimise the ratio of the largest to the smallest eigenvalues of $B_k B_{k+1}^-$. This definition picks out the BFGS update but also the DFP(Davidon(1959), Fletcher and Powell(1963)) update. This, while suggesting the superiority of the DFP and BFGS updates over the rest does not explain the difference between them. In the opinion of the author the defintion of 'conditioning' should bear in mind the criteria which terminate the search. For example, Section 2.1 suggests that the search could be terminated when $\lambda_k - \lambda_{k+1} > \varepsilon |f_k B_k^- f_k|^2 / |B_k^- f_k|^2$ for some preassigned ε. This is clearly going to be affected by the conditioning.

The table suggests that Davidon's(1975) update and the like, Section 3.3, if necessary optimally conditioned, could rival the modified Newton's update (with positive definite corrections when necessary). Davidon's algorithm has the complication of requiring a projection matrix within each iteration. Some of the alternatives of Section 3.3 may prove equally as good. One of these could possibly be chosen so as to make factorisation easier or so as to make direct update of the factors easier. There is more freedom than the Broyden(1967) family because, not only is there a free parameter, the choice of the second conjugate set $\{w_k\}$ is also free. (If side evaluations are used to generate the second set the algorithm will have the same number of function calls as Newton's algorithm when difference approximations to the second derivative are made.) In any event this class of algorithms should prove of current interest.

APPENDIX

The algorithms compared in the Table, page 45, are referred to on the following pages of text:

Newton 15, 16, 18, 20, 21, 23, 40, 43, 44, 46

Steepest descent 16, 41, 42, 44

DFP(Davidon(1959), Fletcher and Powell(1963)) 25, 44, 46

Barnes(1965) 20, 40

Broyden(1965) 22, 24

SR1 Symmetric rank one (Davidon 1968) 22, 24, 25, 38, 39, 43

Powell(1970) 22

BFGS(Broyden(1970), Fletcher(1970), Goldfarb(1970) and Shanno(1970)) 25, 44, 46

Davidon(1975) 26, 27, 37, 38, 39, 43, 46

Note the Huang family contains the Broyden family which contains the BFGS and DFG algorithms. These families of algorithms are referred to on the following pages:

Broyden(1967) 23, 25, 36, 41, 43, 46

Huang(1970) 5, 25, 31, 32, 36, 43

Bibliography

Barnes JGP(1965) An algorithm for solving nonlinear equations based on the secant method, Comput J 8 66-72

Beale EML(1972) A derivation of conjugate gradients in Numerical methods for nonlinear optimisation ed FA Lootsma, Academic Press 39-43

Brodlie KW(1977) Unconstrained minimisation in The state of the art in numerical analysis ed DAH Jacobs, Academic Press 229-268

Brodlie KW Gourlay AR Greenstadt J(1973) Rank-one and rank-two corrections to positive definite matrices expressed in product form, J Inst Maths Applics 11 73-82

Broyden CG(1965) A class of methods for solving nonlinear simultaneous equations, Math Comp 19 577-593

Broyden CG(1967) Quasi-Newton methods and their application to function minimisation, Math Comp 21 368-381

Broyden CG(1970) The convergence of a class of double rank minimisation algorithms, J Inst Math Applic 6 76-90 and 222-231

Broyden CG(1972) Quasi-Newton Methods in Numerical methods for unconstrained optimisation ed W Murray, Academic Press 87-106

Davidon WC(1959) Variable metric method for minimisation, AEC research and development report ANL-5990 (Rev), Argonne National Laboratory, Argonne Illinois

Davidon WC(1968) Variance algorithms for minimisation, Comput J 10 406-410

Davidon WC(1975) Optimally conditioned optimisation algorithms without line searches, Math Prog 9 1-30

Dennis JE(1972) On some methods based on Broyden's secant approximation in Numerical methods for nonlinear optimisation ed FA Lootsma, Academic Press

Dennis JE Jorge JR More J(1976) Quasi-Newton methods motivation and theory, SIAM Rev 19 46-89

Dixon LCW(1972) Quasi-Newton algorithms generate identical points, Math Prog 2 383-387

Fletcher R(1969) A review of methods for unconstrained optimisation in Optimisation ed R Fletcher, Academic Press

Fletcher R(1970) A new approach to variable metric algorithms, Comput J 13 317-322

Fletcher R(1972) An algorithm for solving linearly constrained optimisation problems, Math Prog 2 133-165

Fletcher R(1975) Conjugate gradient methods for indefinite systems in Dundee conference on numerical analysis ed GA Watson, Springer Verlag

Fletcher R Powell MJD(1963) A rapidly convergent descent method for minimisation, Comput J 6 163-168

Fletcher R Reeves CM(1964) Function minimisation by conjugate gradients, Comput J 7 149-154

Gill PE Golub GH Murray W Saunders MA(1974) Methods for modifying matrix factorisations, Comput J 23 505-535

Gill PE Murray W(1972) Quasi-Newton methods for unconstrained optimisation, J Inst Math Applic 9 91-108

Gill PE Murray W(1977) Modification of matrix factorizations after a rank-one change in The state of the art in numerical analysis ed DAH Jacobs, Academic Press

Goldfarb D(1970) A family of variable metric methods derived by variational means, Math Comp 24 23-26

Golfarb D(1976) Factorised variable metric methods for unconstrained optimisation, Math Comp 30 796-811

Guillemin V Pollack A(1974) Differential topology, Prentice Hall

Hestenes MR Stiefel E(1952) Methods of conjugate gradients for solving linear systems, Journal of Research of the National Bureau of Standards B49 409-436

Hicks NJ(1971) Notes on differential geometry, Van Nostrand Reinhold

Himmelblau DM(1972) A uniform evaluation of unconstrained optimization techniques in Numerical methods for nonlinear optimisation ed FA Lootsma, Academic Press

Householder AS(1964) The theory of matrices in numerical analysis, Blaisdell

Huang HY(1970) Unified approach to quadratically convergent algorithms for function minimisation, JOTA 6 405-423

Jacobs DAH(1977) ed The state of the art in numerical analysis, Academic Press

Larichev OI Gorvits CG(1974) New approach to the comparison of search methods used in nonlinear programming problems, JOTA 13 635-59

Lootsma FA(1972) ed Numerical methods for nonlinear optimisation, Academic Press

Luenberger DG(1973) Introduction to linear and nonlinear programming, Addison-Wesley

Munkres JR(1975) Topology a first course, Prentice Hall

Murray W(1972) ed Numerical methods for unconstrained optimisation, Academic Press

Nazareth L(1977) Unified approach to unconstrained minimisation via basic matrix factorisation, Lin Alg Applic 17 197-232

Powell MJD(1969) Rank-one methods for unconstrained optimisation in Nonlinear and integer programming ed J Abadie, North Holland 139-156

Powell MJD(1970) A new algorithm for unconstrained optimisation, Report TP 393 AERE Harwell

Powell MJD(1977) Quadratic termination properties of Davidon's new variable metric algorithm, Math Prog 12 141-147

Rao CR Mitra SK(1971) Generalised inverse of matrices and its applications, John Wiley

Ringwood GA(1978) Biconjugate directions (submitted JOTA)

Shanno DF(1970) Conditioning of quasi-Newton methods for function minimization, Math Comp 24 647-657

Stewart GW(1973) Conjugate direction methods for solving systems of linear equations, Num Math 21 285-297

Van Wyk DJ(1977) Generalisation of conjugate direction methods in the optimisation of functions, JOTA 21 435-56

Wilkinson JH(1965) The algebraic eigenvalue problem, Carendon Press

Wolfe P(1959) The secant method for simultaneous nonlinear equations, Comm ACM 2 12-13

Zoutendijk G(1960) Methods of feasible directions, Elsevier