OPTIMIZATION TECHNIQUES AND THEIR APPLICATION

J.T. HENDERSON

THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN THE UNIVERSITY OF LEICESTER

APRIL 1978

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

For my wife and children

ABSTRACT

The problem of optimizing a nonlinear function of one or more variables, in the sense of locating the values of the variables which give the greatest or least value of the function, is considered from two points of view. First, the development of two new and improved techniques for optimization is described. Second, the ways in which the available techniques can be applied are discussed with reference to case studies of practical significance.

The two new techniques are for unconstrained optimization problems of a type which frequently occur in curve-fitting and modelling applications and also in the solution of sets of nonlinear equations. The first of these is a new two-part algorithm for minimizing a sum of squares objective function; it uses a new descent method in combination with a modified Gauss-Newton search to give an algorithm which has proved extremely reliable even when applied to difficult problems. The second technique is a hybrid algorithm for minimizing a sum of moduli objective function; it makes novel use of the methods of parametric linear programming.

Ten case studies of the application of optimization techniques are described, ranging from problems involving a single variable up to a problem with several hundred variables. The areas from which the applications are drawn include biochemistry, engineering, statistics and theoretical physics; the problems themselves are mainly concerned with curve-fitting or the solution of nonlinear equations.

Acknowledgements

The Author is indebted to his supervisor, Dr. O.P.D. Cutteridge, for his advice and encouragement; to Mr. M.A. Wolfe for his interest and helpful discussions; to his many colleagues at the Universities of Leicester and St Andrews for providing him with interesting optimization problems; and lastly to Seonaid for her skilled work in typing this thesis.

C O N T E N T S

1. INTRODUCTION

Optimization in the sense of trying to find the maximum or minimum value of a function of one or more variables has been a topic of interest to mathematicians from the first. Perhaps the earliest optimization problems were solved by Euclid who showed [1] for example that of all parallelograms of a given perimeter the square is the one with maximum area. Coming closer to the present day, problems arising from studies such as kinematics and dynamics were solved using the methods of calculus and calculus of variations developed during the seventeenth and eighteenth centuries. At this time also the technique of Lagrange multipliers [2] was introduced to handle explicit constraints which the variables were required to satisfy.

While aids to computation have been available [3] since pre-Christian days, and numeric methods of solving optimization problems could thereby be attempted in the absence of analytical solutions, with the advent of the programmable digital computer within the last thirty years it has become computationally-feasible to apply numeric methods to problems of a complexity far in excess of previously-attempted problems. At the same time, the acceleration in the growth of both the scale and the inter-relationships of human activities has led man to seek ways of controlling his activities in an optimum fashion. One useful approach is to create mathematical models of real situations; the variables of the model represent factors under human control, the constraint equations represent physical limitations on the values these variables can adopt and the objective function represents the measure of cost or benefit which it is

sought to optimize appropriately. A classical example of such a model is the transportation problem of Hitchcock [4]; although the objective function and the constraints are all linearly-dependent upon the variables in Hitchcock's model, the scale of the problem necessitates a computer-aided solution in most cases.

Dantzig [5] formulated the Simplex method for solving such problems which he termed "linear programming" problems; by programming he was referring to the planning process which was to be optimized on the basis of these models. The Simplex method and its variants have been, and continue to be, used with beneficial results. However, a linear representation of a real situation is inadequate in many areas of interest and so the more general mathematical programming problem has to be considered. This takes the form

$$\begin{array}{lll} \text{Minimize} & f(\underline{x}) & \\ \text{Subject to} & g_i(\underline{x}) \geqslant 0 & i=1, \ldots p \\ & h_j(\underline{x}) = 0 & j=1, \ldots q \end{array}$$

The term *mathematical programming* is credited to Dorfman by Himmelblau [6]. The variables are denoted by the vector $\underline{x}^T = (x_1, \ldots x_n)$; the objective function by $f(\underline{x})$ and the constraints are subdivided into p inequality constraints and q equality constraints. Note that this form is completely general since the maximization of $f(\underline{x})$ can be carried out by minimizing $-f(\underline{x})$. Similarly, an inequality constraint of the type $g_i(\underline{x}) \leqslant 0$ can be expressed as $-g_i(\underline{x}) \geqslant 0$.

Many special cases of the mathematical programming problem have been identified, such as the integer and quadratic programming problems as well as the linear programming problem already discussed. In this thesis, the

concern is with the nonlinear programming problem, obtained when at least one of the terms $f(\underline{x})$, $g_i(\underline{x})$ and $h_j(\underline{x})$ is nonlinear. Some of the early fundamental theory on the nonlinear problem was done by Kuhn and Tucker [7] who derived the necessary and sufficient conditions for $\underline{x}^*$ to be a local minimum when $f(\underline{x}^*)$ is a convex function. By a local minimum is meant that for any value $\underline{x}$ within a small but finite distance of $\underline{x}^*$, measured according to some suitable norm, the relation $f(\underline{x}) > f(\underline{x}^*)$ holds. A global minimum will have been reached at $\underline{x}^*$ if this relation holds for any point $\underline{x}$ in n-dimensional Euclidean space.

As indicated by its title, this thesis is concerned with not only optimization techniques but also their application to problems of practical importance. Chapters 2 and 3 describe two new iterative methods for unconstrained optimization, the first of which was developed by the author in collaboration with his supervisor, Dr.O.P.D. Cutteridge, and the second of which was developed solely by the author. The restriction to unconstrained problems is not so limiting as might appear. Many methods for solving the constrained problem make use of methods for unconstrained optimization, for example the Sequential Unconstrained Minimization Technique described by Fiacco and McCormick [8]. Also, it is often found that a constrained problem can be simplified to an unconstrained problem by an appropriate transformation of the variables, as described for example by Kowalik and Osborne [9].

In developing these two new methods, the emphasis was laid on the effective implementation of ideas, derived mainly from intuitive reasoning, by following a course dictated by the results of numerical experiments. Many methods have been successfully used long before the underlying mathematical theory was complete; they would not have been available for use before that time if their implementation had been delayed until their

performance could be theoretically predicted. In addition, it will always be the case that as far as optimization methods are concerned the proof of the pudding is in the eating; no matter how much theory is given about them, most new methods are published with numerical details of their performance on test problems and comparisons with existing methods.

Chapter 4 is concerned with the application of optimization methods. There can be no hard and fast rules for determining which is the best method for a particular application. There will always be a test problem, not necessarily of pathological interest, which will defeat a given method. The author's belief is that optimization techniques are like any other branch of numerical analysis, the problem must first be formulated in the most effective way before considering the choice of optimization method. It is the author's experience that in many instances the selection of method becomes of secondary importance. The case studies given in Chapter 4 illustrate these points.

1.1 Minimizing a function of one variable

The problem of minimizing a function of one variable is often encountered in practice and is of interest in its own right. However, many optimization methods for problems of more than one variable make use of line-searches in which the minimum of the objective function is sought along a line in n-dimensional Euclidean space. Consequently special attention has been devoted to these methods.

A class of simple methods described by Swann [10] are of use when the minimimum is known to be located between lower and upper bounds on the independent variable. Within these bounds, a set of trial points can then be generated either with some random distribution or to form a regular grid.

The function is then evaluated at each point and the lowest value of the function so found is taken as the minimum. Alternatively, new bounds with a smaller range could be deduced and the process repeated until the position of the minimum is known to within a required accuracy. Such methods have the advantages of being simple to implement and make no assumptions about the way in which the function varies. However they are expensive in terms of function evaluations.

If it can be safely-assumed, as often is the case, that the function is unimodal within given bounds about the minimum, then a more systematic and efficient method can be adopted. Suppose that the four points $a < b < c < d$ are found such that the minimum is at x^* where $a < x^* < d$. Then by a comparison of the function values at each point it is possible to reduce the bounds to $a < x^* < c$ or $b < x^* < d$. It is then only necessary to compute one more function value, at some point within the new bounds, to be able to repeat the process. The most efficient method of generating the trial points, in the sense that for a fixed number of function evaluations it gives the greatest reduction in the initial range, is based on the Fibonacci numbers as shown by Kiefer [11]. A directly-related method is the Golden Section search, described by Kowalik and Osborne [9]. At worst, the Golden Section search will require one more function evaluation than the Fibonacci search for the same interval reduction; but the Golden Section search is easier to implement and is therefore preferred.

While these last two methods are a great improvement on a random or grid search, they still do not utilise all the available information since they simply compare function values. The actual values of the function can be employed to derive an approximate function which roughly follows the behaviour of the original function but whose minimum can be predicted

analytically to give an estimate of the minimum x* of the original function. Normally a quadratic or cubic polynomial is used for the approximating function. Davies, Swann and Campey [12] use a method which first locates three points $a < b < c$, where $b = (a + c)/2$, such that $f(a) > f(b) < f(c)$. A single quadratic interpolation gives a new estimate of the minimum. This forms the starting point for the next search for a bracket followed by an interpolation, and so on with the bracket on the minimum being progressively reduced. Powell [13] described a method in which the bracket is found once and then reduced by the results of repeated quadratic interpolations. A cubic function was used by Fletcher and Reeves [14], based on the function and its derivative at two points. Some further suggestions for univariate searches are given by Bard [15].

1.2 Minimizing a function of more than one variable

Many books have been published on iterative methods for unconstrained optimization, including those by Kowalik and Osborne [9], Murray [16] and Himmelblau [6]. Some of the more commonly-met methods will now be outlined. These can be split into those which require function values only and those which require in addition first, and possibly second, partial derivatives of $f(\underline{x})$ with respect to the variables.

1.2.1 Methods using function values only

Most of the methods in this category can be termed direct search methods; they use a systematic series of trials, based on function comparisons, which is intended to lead ultimately to the minimum. They have the advantages of being simple and do not make any assumption on $f(\underline{x})$ other than it is continuous. However, they have a slow final rate of convergence when they get close to the minimum.

A simple, but not very effective, strategy is the alternating variable method described by Swann [10]; a univariate search is used on one variable at a time to locate the minimum of $f(\underline{x})$ with respect to that variable. The Fibonacci search has been extended to the n-dimensional case by Krolak and Cooper [17]. In the pattern search of Hooke and Jeeves [18], the basic operation is a series of exploratory moves followed by a pattern move, which is predicted by the exploratory moves to be a profitable direction of search as regards reducing $f(\underline{x})$. Exploratory moves are also a feature of the method of Rosenbrock [19]; he used a set of n orthogonal directions $\underline{v}_i$ (i=1, .. n) where $\underline{v}_i^T \underline{v}_j = 0$ $(i \neq j)$ and adjusted these directions as the search progressed. The razor search of Bandler and MacDonald [20] adjusts the length of each move to allow for sharp peaks and troughs in the objective function. The Simplex method of Nelder and Mead [21] uses a set of n+1 points to form the vertices of a regular polyhedron (the Simplex) in n-dimensional Euclidean space. On the basis of comparisons of the function values at the vertices, the Simplex is then expanded, contracted or reflected about a vertex. The process is repeated until a minimum is located.

1.2.2 Methods requiring derivatives

If first derivatives of $f(\underline{x})$ are available in the form of the gradient vector $\underline{g}^T(\underline{x}) = (\partial f(\underline{x})/\partial x_1, \ldots \partial f(\underline{x})/\partial x_n)$ then it is possible to ensure that at each iteration of a search the corrections $\Delta\underline{x}$ to be made to the current estimate $\underline{x}^k$ of the minimum are downhill by finding a $\Delta\underline{x}$ such that $\underline{g}^T(\underline{x}^k)\,\Delta\underline{x} < 0$. This test applies at $\underline{x}^k$ and as we move away from $\underline{x}^k$ in the direction of $\Delta\underline{x}$ it will not normally remain downhill. Consequently many methods use a line

search to find the scalar α which minimizes $f(\underline{x}^k + \alpha\Delta\underline{x})$ and then set $\underline{x}^{k+1} = \underline{x}^k + \alpha\Delta\underline{x}$. In some cases, all that is required is that an α is found to give a sufficient reduction in $f(\underline{x})$, as discussed by Fletcher [22].

An early derivative method is that of steepest descent due to Cauchy [23]; the corrections at any iteration are given by $\Delta\underline{x} = -\underline{g}(\underline{x}^k)$. This method will always converge to a stationary point, although not necessarily a minimum, but the final rate of convergence can be very slow. Consequently, the method is not often used in practice.

The classical Newton method for solving nonlinear equations can be extended to the minimization of $f(\underline{x})$ and is the basis of a large number of methods. At a minimum we will have $g_i(\underline{x}^*)=0$ $(i=1, .. n)$. We can approximate $\underline{g}(\underline{x})$ about the point $\underline{x}^k$ by $\underline{g}(\underline{x}^k+ \Delta\underline{x}) \simeq \underline{g}(\underline{x}^k) + G(\underline{x}^k)\ \Delta\underline{x}$, where $G(\underline{x}^k)$ is the Hessian matrix $G_{ij} = \partial^2 f(\underline{x})/\partial x_i \partial x_j$ evaluated at $\underline{x}^k$. The corrections required on the basis of this approximation to reduce the gradient to zero are then given by the solution of the system of linear equations $G(\underline{x}^k)\ \Delta\underline{x} = -g(\underline{x}^k)$. For a positive definite quadratic function, the method will terminate in one iteration; this property is of relevance since in the vicinity of $\underline{x}^*$ it is often the case that $f(\underline{x})$ can be approximated by a positive-definite quadratic form.

Two drawbacks to Newton's method are that it requires second derivatives, which can be expensive to compute, and it involves the solution of linear equations. The main drawback is that G is not necessarily positive-definite and so the corrections $-G^{-1}\underline{g}$ may not be downhill. Greenstadt [24] overcomes this difficulty by using an eigenvalue and eigenvector analysis of G. He sets to a small positive value all negative eigenvalues and eigenvalues close to zero. Then he computes G^{-1} using the

modified eigenvalues and the eigenvectors. Murray [25] believes that it is more efficient to assume G is positive-definite and use a Cholesky factorisation as described by Wilkinson [26] to compute G^{-1}. Only if the factorisation fails, because G is not positive-definite, need Greenstadt's approach be used. In general, it will be necessary to carry out a line search to ensure that a reduction in $f(\underline{x})$ is obtained; Gill, Murray and Picken [27] give one such implementation of Newton's method.

Davidon [28] originally proposed the class of variable metric or quasi-Newton methods. He proposed that an initial estimate of G^{-1} be made and then updated as the search progressed using first derivatives only. It can be ensured, theoretically at least, that the estimate of the matrix G^{-1} remains positive-definite. Fletcher and Powell [29] gave a more exhaustive treatment of the method. Broyden [30] later developed a related family of rank one methods, so called because the matrix used to update the estimate of G^{-1} at each iteration is of rank one (the Davidson-Fletcher-Powell matrix is of rank two). Gill and Murray [31] proposed that the estimate of G^{-1} be stored in factored form to ensure that numerical difficulties do not cause the matrix to become negative-definite; they also give some consideration to the use of numeric estimates of the derivatives.

A final class of methods is based on the use of conjugate directions; the set of n vectors $\underline{v}_i (i=1, .. n)$ are conjugate with respect to G if $\underline{v}_i^T G \underline{v}_j = 0 \ (i \neq j)$. These methods have the property of quadratic termination in that if $f(\underline{x})$ is a positive definite quadratic they will terminate in a finite number of steps. This property depends upon exact line searches being used, which is not practicable. Two well-known methods of this type are the conjugate gradient method of Fletcher and Reeves [14] and the method of Powell [13] which does not in fact require derivatives.

1.3 Methods for least squares problems

Methods have been developed to handle the special case of the least squares problem when the objective function to be minimised is in the form of a sum of squares $\sum_{i=1}^{m} f_i^2(\underline{x})$. Such problems frequently arise in curve-fitting and modelling where some physical quantity Y is assumed or known to depend on one or more variable quantities $\underline{X}^T = (X_1, \ldots X_p)$ according to a given functional relation $Y = F(\underline{X}, \underline{x})$. The vector $\underline{x}^T = (x_1, \ldots x_n)$ represents fixed parameters appearing in the function F. A set of experimental measurements is taken to give the observed value Y_i of Y at each of m points $\underline{X}_i$; it is then desired to find the parameters $\underline{x}$ which give closest agreement between the observed and predicted values of Y.

At each measured point, the difference between the observed and predicted value can be denoted by the residual $f_i(\underline{x}) = Y_i - F(\underline{X}_i, \underline{x})$. A common measure of fit is the sum of the squares of the residuals (the l_2-norm) which is minimized to give the best fit according to this criterion. Other norms can be used; thus Barrodale and Roberts [32] give a method for minimizing the sum of the absolute values of the residuals (the l_1-norm) and Osborne and Watson [33] give a method for minimizing the maximum magnitude residual (the l_∞-norm).

Least squares methods also have an important use in the solution of systems of nonlinear equations. Given the system $f_i(\underline{x}) = 0$ (i=1, .. n) then if the sum of the squares $\sum_{i=1}^{n} f_i^2(\underline{x})$ is minimized and the minimum found to be zero the vector $\underline{x}^*$ at the minimum will be a solution of the equations. Sometimes a nonzero minimum will be found; in this case further investigations are necessary to determine if this is in fact the global minimum, and hence the nearest approach to a solution, or if the optimization method is failing to locate the global minimum.

Reviews of the methods available for least squares problems have been given by Lill [34] and Dennis [35]. Most of the methods derive from a modification of Newton's method first proposed by Gauss [36] and hereafter referred to as the Gauss-Newton method. As shown earlier, for a general objective function the Newton corrections are $-G^{-1}\underline{g}$. For a sum of squares objective function differentiation gives

$$G_{ij} = \sum_{k=1}^{m} 2\left(\frac{\partial f_k}{\partial x_i}\,\frac{\partial f_k}{\partial x_j} + f_k\,\frac{\partial^2 f_k}{\partial x_i \partial x_j}\right)$$

from which it will be seen that the Hessian is made up of two terms, one of which does not involve second derivatives. If the residuals f_k are small, or they are varying slowly with $\underline{x}$, then we may approximate G_{ij} by ignoring the terms involving second derivatives. This approximation is likely to get better as a minimum of the sum of squares is approached. If we denote the Jacobian matrix by J such that $J_{ij} = \partial f_i/\partial x_j$, then the Hessian approximation can be written as $2J^TJ$ and the gradient vector is given exactly by $2J^T\underline{f}$. The Gauss-Newton corrections then become $(J^TJ)^{-1}J^T\underline{f}$, where the factors of 2 have been cancelled.

Provided that J is nonsingular, the Gauss-Newton corrections will always be downhill. However, unless $f(\underline{x}^*)=0$, the final rate of convergence of the Gauss-Newton method is of first order as compared with the second order rate of convergence of Newton's method. In most problems, an iterative search using the Gauss-Newton corrections as they stand would diverge; Hartley [37] used a line search to ensure that the sum of squares decreases at every iteration.

The difficulty met when J becomes singular has been considered by Levenberg [38] and later Marquardt [39]. They proposed the corrections

$\underline{\Delta x} = -(J^TJ + \lambda W)^{-1}J^T\underline{f}$ where W is a positive-definite matrix and λ is a parameter whose value is to be chosen at each iteration to give a reduction in the sum of squares. A suitable choice for W, suggested by Marquardt, is I, the unit diagonal matrix. The corrections can be seen to be hybrid since when λ is increased from zero, the corrections initially equal the Gauss-Newton corrections and change to the direction of the steepest descent corrections. Many methods have been subsequently derived using the Levenberg-Marquardt corrections including those of Goldfeld , Quandt and Trotter [40], Fletcher [41], Meyer and Roth [42] and Nash [43]. A similar hybrid approach was used by Powell [44] and Jones [45].

A recent comparison of some least squares methods was given by Wolfe [46]. Bard [15] carried out a set of comparisons, on least squares problems, of special-purpose methods and variable metric methods; he concluded that the former were to be preferred. However, McKeown [47] has shown that for some least squares problems it may be better to use one of the standard unconstrained optimization methods.

1.4 Methods for global optimization

Before progressing to the next chapter, it is relevant to outline some of the methods available for global optimization, since the algorithm described in Chapter 2 and two of the case studies of Chapter 4 consider this problem. A review of methods for global optimization was given by Dixon, Gomulka and Hersom [48]. A probabilistic approach is often used. In the multistart method, a number of minimizations are carried out, each starting from a different point chosen in some random fashion. The

lowest minimum (often the same minimum is found every time) is taken as the global minimum. A second technique is to generate a large number of trial points at random in n-dimensional Euclidean space. Price [49] recently gave one such method.

Other methods are available which do not use a probabilistic approach. The descent from a minimum method of Goldstein and Price [50] works well for polynomial functions. The trajectory approach of Branin [51] uses a closed curve obtained by integrating the equation $\dot{\underline{g}}(\underline{x}) = -\underline{g}(\underline{x})$ and which, it is hoped, will pass through all the stationary points of the objective function.

2. A TWO-PART ALGORITHM FOR MINIMIZING SUMS OF SQUARES OF NONLINEAR FUNCTIONS

This chapter describes the development of a two-part algorithm and corresponding computer program for the minimization of sums of squares of nonlinear functions through iteration. The work was done by the author in conjunction with Cutteridge. The latter's contributions to the research are acknowledged in the text where appropriate; otherwise the work described and the views expressed are the author's own.

2.1 Background to the research

The synthesis of lumped linear electrical networks of arbitrary structure has for long been a research topic of interest to Cutteridge; a particular area of his concern has been the synthesis of networks which do not have a series-parallel equivalent [52]. For a given network topology, one practical method of quantifying the discrepancy between the desired characteristics of the network and its actual characteristics is that of coefficient matching. This method, described by Calahan [53], measures the discrepancy by a set of nonlinear functions of the values of the elements making up the network. At an exact match, all these functions are zero and the element values can be varied in an attempt to achieve this situation. In general, it may be necessary to alter the network topology to obtain a match; this aspect of the problem is discussed by Cutteridge and di Mambro [54].

The process of searching for the element values to give the best match is equivalent to the solution of m nonlinear equations (the functions) in n variables (the element values). The approach used by Cutteridge was to minimize the sum of the squares of the functions (or residuals of the equations) using standard techniques for optimization. While normally m=n

for Cutteridge's problems, it was found by Cutteridge and Krzeczkowski [55] that the optimization could be assisted by introducing excess functions, chosen suitably and which are zero at a solution of the original system. Also, although a solution may not exist for the current topology an approximate solution will always be obtained which minimizes the sum of squares. Thus the algorithm described in this chapter is directed towards the minimization of sums of squares; it should be borne in mind that it could equally be applied to solving nonlinear equations.

Cutteridge found from experience that for some of his problems existing methods of optimization were not sufficiently reliable; it was important in the synthesis process to know, in the event of failing to reach a solution, whether this failure was due to deficiencies in the opimization or indicated the need to change the network topology. Thus Cutteridge developed a reliable two-part algorithm [56] of his own. The concept of a two-part algorithm can in fact be traced back to 1847 when Cauchy [23] presented a paper which is best-known today as the first exposition of the method of steepest descent. At the time, Cauchy wished to solve sets of nonlinear equations which he encountered in his calculations on planetary orbits. To do this he used Newton's method which he found did not always converge. This failure was caused by starting the Newton iterations from variable values which are too far removed from the solution. To overcome this difficulty, Cauchy proposed that several iterations of his method of steepest descent should be first carried out until the variables are close to the solution as indicated by the residuals of the equations becoming small. Newton's method would then locate the solution values accurately and rapidly if started from the values at the end of the steepest descent search.

Two-part algorithms are just one instance of the polyalgorithm approach to optimization. This method has not found great favour with many researchers, possibly due to the fact that a theoretical treatment of polyalgorithms is

made difficult by the presence of empirical switching criteria determining when to change from one algorithm to another during the search. Published work on polyalgorithms is thus biased towards a practical, rather than theoretical, treatment; for example Phillips [57], Chien [58] and the work of Cutteridge cited earlier. The underlying aim of polyalgorithms, namely the combination of the best features of two or more algorithms, has not been neglected since the concept of hybrid algorithms has received a lot of attention. The essential difference in a hybrid algorithm is that, at each iteration, the change in the variables is given by an interpolation between the two correction vectors produced by two different algorithms. The interpolation process usually depends upon parameter values some of which must be set by the user of the algorithm; this dependence upon parameter values is one disadvantage of the method. The most common hybrid algorithms are those which interpolate between the steepest descent corrections and the Newton (or Gauss-Newton) corrections. The original work on such algorithms was by Levenberg [38] and Marquardt [39] and has been followed up by many others including Powell [44], Fletcher [40], Jones [45] Goldfeld, Quandt and Trotter [41], Meyer and Roth [42] and Nash [43].

Returning to the two-part algorithm, Cutteridge used for this two algorithms combined in the end-on manner of Cauchy. However, Cutteridge introduced an additional facility such that, if the second part of the algorithm fails to converge, then control is passed back to the first part with the variables reset to their values on entry to the second part; further progress is then made in reducing the sum of squares before the second part is re-entered. In the first part, Cutteridge used a reliable descent algorithm; he tried, among others, the conjugate gradient method of Fletcher and Reeves [14] and his own modification [59] of the Levenberg-Marquardt hybrid algorithm. In the second part, since he was dealing with sums of squares, he used a modified Gauss-Newton search similar to that described by Hartley [37]. He improved the efficiency of the method by using

empirical criteria to anticipate, as far in advance as possible, the onset of failure in the Gauss-Newton search if this should happen. In this way, he was able to return to the first part before failure actually took place and save on wasted Gauss-Newton iterations.

Cutteridge applied his two-part algorithm to solve many problems of interest. When used in network synthesis, it is not always possible to supply close estimates of the solution values. Consequently, Cutteridge was interested in developing an algorithm which would be efficient yet converge to a solution from a wide range of starting points and for difficult problems. This task he gave to the author whose work in improving the two-part algorithm will now be described. During the development, a single test problem was used; this was a set of eight nonlinear equations involving exponentials, formulated by Skwirzynski [60] and shown in the Appendix. Prior to this work, Cutteridge had been able to solve this problem only from a few starting-points close to the solution.

2.2 An improved descent algorithm

Cutteridge suggested that, for the first part of his algorithm, an improved descent algorithm could be developed if second derivatives of the functions were used in addition to first derivatives. He proposed that a correction vector $\Delta\underline{x}$ be obtained by following a "curve of steepest descent" given for iteration k by the expression

$$\Delta\underline{x} = -\alpha \ (\underline{g}^k + G^k \ \Delta\underline{x}) \qquad (2.1)$$

Here $\underline{g}^k$ is the gradient vector the sum of squares objective function $F = \sum_{i=1}^{m} f_i^2 \ (\underline{x})$ and G^k is the Hessian matrix of F, both evaluated at $\underline{x}^k$, the value of $\underline{x}$ at iteration k. The locus of the curve is formed by variation of the positive scalar α, noting that at $\alpha = 0$ the correction vector is zero. At each iteration, α was to be chosen so as to minimize the function

$F(\alpha) = F(\underline{x}^k + \Delta\underline{x})$.

2.2.1 Initial investigations

The author pointed out that equation (2.1) is not the true curve of steepest descent because, for this to be so, the direction $d\underline{x}/d\alpha$ (or $d\Delta\underline{x}/d\alpha$ since $\underline{x} = \underline{x}^k + \Delta\underline{x}$) must be in the same direction as $-\underline{g}(\underline{x})$ at any point along the curve. It should also be noted that the definition of "steepest" depends upon the norm used to define distance, as discussed by Murray [25]. It will be seen that Cauchy's method of steepest descent can be derived by approximating $\underline{g}(\underline{x})$ by its value $\underline{g}^k$ at $\underline{x}^k$. If the differential equation $d\Delta\underline{x}/d\alpha = -\underline{g}^k$ is solved, subject to $\Delta\underline{x} = 0$ at $\alpha = 0$, then the steepest descent corrections $\Delta\underline{x} = -\alpha\underline{g}^k$ are obtained. If the Hessian is available, a better approximation is given by $\underline{g}(\underline{x}) \simeq \underline{g}^k + G^k\Delta\underline{x}$ to give

$$\frac{d}{d\alpha}\Delta\underline{x} = -(\underline{g}^k + G^k\Delta\underline{x}) \qquad (2.2)$$

with the same boundary conditions of $\Delta\underline{x}=0$ at $\alpha=0$. Differential equations have been used by other workers such as Branin [51] and Ramsay [61] in optimization algorithms.

A program was written using corrections defined by equation (2.2) since it was believed that this gave a better approximation to the curve of steepest descent than equation (2.1). For a given value of α, the corrections $\Delta\underline{x}$ were calculated using a Runge-Kutta [62] numerical integration. It was found that the computation time required by the integration was much greater than that required to evaluate F, $\underline{g}^k$ and G^k. This time could have been reduced by using a less accurate integration and also, when searching for α to minimize $F(\alpha)$, by using intermediate results for $\Delta\underline{x}$ obtained at values of α covered by previous integrations. However, it was believed that even with these economies an efficient algorithm would not be obtained and equation (2.1) was re-considered.

In passing, it should be observed that an analytic solution of equation (2.2) is possible given that the eigenvalues of G^k are first calculated numerically. In the simplest case, when G^k possesses n distinct, non-zero eigenvalues ϕ_i (i=1, .. n) the solution is

$$\Delta \underline{x} = -(G^k)^{-1} \underline{g}^k \text{ diag } (1 - \exp[-\phi_i \alpha]) \qquad (2.3)$$

where diag (a_i) denotes a diagonal matrix such that the diagonal element on row i is a_i and all off-diagonal elements are zero. Complications arise when there are repeated or zero eigenvalues. The general analytical solution of first order, linear differential equations is discussed by Braun [63]. Some trials were made at a later date using equation (2.3); these were encouraging but, for reasons of time, have not been followed up for the present work.

While equation (2.1) is not the true path of the steepest descent, use of it may be justified from other considerations. One way is to consider the approximate solution to equation (2.2) given by assuming the right-hand side is a constant quantity; in this case equation (2.1) is obtained. The assumption will be roughly valid for small α, when $\underline{g}^k$ will outweigh $G^k \Delta \underline{x}$. Better approximate solutions could be obtained with different assumptions. However, the most convincing reason for using equation (2.1) is that it has a parallel with the expression used by Levenberg and Marquardt, as will now be shown.

Since α is an arbitrary scalar parameter, whose value is to be determined by a line search, then the algorithm is not affected if we replace α in equation (2.1) by $1/\lambda$, where λ is another parameter. Rearrangement then gives

$$(G^k + \lambda I) \Delta \underline{x} = -\underline{g}^k \qquad (2.4)$$

where I is the unit diagonal matrix. The expression used by Levenberg

and Marquardt can be written

$$(J^TJ + \lambda'W)\ \Delta\underline{x} = -J^T\underline{f} \qquad (2.5)$$

where J is the Jacobian matrix of the residuals $\underline{f}$ and W is a pre-determined diagonal matrix. One choice for W suggested by Marquardt is the unit matrix. The scalar parameter λ' is chosen at each iteration by a search on $F(\underline{x}^k + \Delta\underline{x})$. Taking W=I and replacing λ' by $\lambda''/2$, multiplication of both sides of equation (2.5) by 2 gives

$$(2J^TJ + \lambda''I)\ \Delta\underline{x} = -2J^T\underline{f} \qquad (2.6)$$

Now for a sum of squares objective function, the gradient vector $\underline{g}^k$ is given by $2J^T\underline{f}$; furthermore, the Hessian matrix G can be replaced by its Gauss-Newton approximation $2J^TJ$. Thus equations (2.4) and (2.6) are equivalent except that the former uses the exact Hessian. Other workers for example Bard [15] have used equation (2.4) as the basis for an optimization algorithm.

Inspection of equation (2.4) gives

$$\lim_{\lambda \to 0} \Delta\underline{x} = -(G^k)^{-1}\ \underline{g}^k \qquad (2.7a)$$

$$\lim_{\lambda \to \infty} \Delta\underline{x} = -\underline{g}^k/\lambda \qquad (2.7b)$$

while similarily equation (2.6) yields

$$\lim_{\lambda'' \to 0} \Delta\underline{x} = -(J^TJ)^{-1}\ J^T\underline{f} \qquad (2.8a)$$

$$\lim_{\lambda'' \to \infty} \Delta\underline{x} = -2J^T\underline{f}/\lambda'' \qquad (2.8b)$$

These results shown that, for small values of the parameters λ and λ'', the values of $\Delta\underline{x}$ tend to the Newton and Gauss-Newton corrections respectively.

Further, in both cases, when the values of the parameters tend to large, positive values then the corrections become small and in the direction of steepest descent. Thus, by taking a sufficiently large value for λ or λ'' it should always be possible to generate a correction $\Delta\underline{x}$ for which $F(\underline{x}^k + \Delta\underline{x}) < F(\underline{x}^k)$, except of course if a local minimum has been reached. The validity of this argument is also dependent upon round-off error not having a dominant effect on the calculation, which can be the case in some instances.

A program was written, using the original expression for the corrections as given by equation (2.1); α was chosen to minimise $F(\alpha)$. Note that in this case $\Delta\underline{x}$ tends to the steepest descent direction as α tends to zero. A single iteration of the algorithm consisted of the following steps.

Step 1 — Compute $F(\underline{x}^k)$, $\underline{g}^k$ and G^k
Set $\alpha_1=0$, $F_1=F(\underline{x}^k)$
Set $\alpha_2=\alpha_0$ where α_0 is a given value

Step 2 — Compute $\Delta\underline{x}$ from equation (2.1) for $\alpha=\alpha_2$;
set $F_2=F(\underline{x}^k + \Delta\underline{x})$

Step 3 — If $F_2 > F_1$ then set $\alpha_2=\alpha_2/2$ and return to Step 2. Otherwise ....

Step 4 — Set $\alpha_3=2\alpha_2$; compute $\Delta\underline{x}$ for $\alpha=\alpha_3$ and set $F_3=F(\underline{x}^k + \Delta\underline{x})$.

Step 5 — If $F_3 < F_2$ then set $\alpha_1=\alpha_2$, $\alpha_2=\alpha_3$, $F_1=F_2$, $F_2=F_3$ and return to Step 4. Otherwise ....

Step 6 — The minimum of $F(\alpha) = F(\underline{x}^k+\Delta\underline{x})$ is at α_m where $\alpha_1 < \alpha_m < \alpha_3$; use a Golden Section search to locate α_m more precisely.

A similar doubling-process to bracket a minimum was used by Davies,

Swann and Campey [12] .

It quickly became apparent that the behaviour of the search was significantly affected by the choice of α_0 at step 1. Assuming that $F(\alpha)$ is unimodal, changing α_0 will alter the values of α_1 and α_3 when the minimum is bracketed at step 6; this in turn will, in general, cause the final value of α_m to be slightly different. It is well-known, for example Dixon [64] , that changing the accuracy of a line-search can greatly affect the overall behaviour of an optimization algorithm. However this, was not the main reason for the differences found here. Detailed printout of the calculations showed that completely different minima of $F(\alpha)$ were often found at the same iteration by choosing α_0 differently. Tabulation of $F(\alpha)$ over a wide range of values of α showed that it was in fact highly non-unimodal with many sharp peaks and troughs. This behaviour is not so marked in the Levenberg-Marquardt algorithm for the following reasons.

The Gauss-Newton approximation $2J^TJ$ in equation (2.6) is always positive semidefinite, and most likely is positive definite, since for any vector $\underline{y}$ we have $\underline{y}^T (2J^TJ)\underline{y} = 2(J\underline{y})^T(J\underline{y}) \geq 0$. Thus the matrix $(2J^TJ+\lambda''I)$ will be positive definite for any $\lambda'' > 0$ and $\Delta\underline{x}$ will be downhill with respect to $F(\underline{x}^k)$. Furthermore, the modified matrix cannot be singular for any positive λ''. Thus it is likely that $F(\underline{x}^k+\Delta\underline{x})$ will be unimodal, or nearly so, in the range $\lambda'' > 0$. In the case of equation (2.4) these arguments do not apply. There is no guarantee that the true Hessian G^k, even when augmented by the addition of a positive value λ to its diagonal, will be positive definite. Thus $\Delta\underline{x}$ need not be downhill and if λ equals $-\phi_i$, where ϕ_i is an eigenvalue of G^k, then the matrix $G^k + \lambda I$ is singular and the corrections become infinite.

In view of the multimodal nature of $F(\alpha)$, the author decided to use a search which would look for the global minimum of this function. To draw a parallel with the Levenberg-Marquardt algorithm it was also decided to use equation (2.4) to evaluate the corrections and search over λ, rather than α. This change of itself does not affect the results since there is a one-to-one correspondence between the corrections at $\lambda=a$ and those at $\alpha=1/a$. Some minor differences will be introduced by the effects of roundoff error and the fact that the global minimum cannot be located exactly. For the sake of completeness, it was further decided to search over all real values of λ, rather than restrict the search to positive values only.

2.2.2 The multimodal search

The search for the global minimum of $F(\lambda)$ could be considered as a special case of the more general problem of global optimization of a function of n variables. Most existing algorithms for this problem, such as the descent from a minimum method of Goldstein and Price [50] assume that the function and its first (and sometimes higher) order derivatives are continuous; this is not the case with $F(\lambda)$. It was decided to develop a special-purpose search, which could to its advantage take note of the known properties of $F(\lambda)$.

A simple strategy was adopted, similar to a method which was used by Bandler and MacDonald [20]. The range of λ is covered by a grid of trial values λ_i; it will be assumed that $\lambda_i > \lambda_{i-1}$. The first grid-point is sufficiently large and negative, and the final grid-point similarily large and positive, so that the corrections at the ends of the grid are close to zero. The values $F_i=F(\lambda_i)$ are evaluated; if three successive grid-points show that $F_{i-1} > F_i < F_{i+1}$ then a local minimum must exist at λ_m where

$\lambda_{i-1} < \lambda_m < \lambda_{i+1}$. These bounds on λ_m are then reduced to an acceptable size using a standard method for finding the minimum of a unimodal function. The global minimum is then taken as the lowest of all the local minima detected. The strategy depends for its success on a judicious choice of grid; a balance must be struck such that the grid is sufficiently fine to sieve out the local minima but not so fine as to lead to an excessive amount of computation.

The first grid tried contained 2N+1 grid-points defined as follows:

$$\lambda_{N+i+1} = r\lambda_{N+i} \qquad i=2, \ldots N \qquad (2.9a)$$

$$\lambda_{N+1} = 0 \qquad (2.9b)$$

$$\lambda_i = -\lambda_{2N+2-i} \qquad i=1, \ldots N \qquad (2.9c)$$

It will be seen that the grid is symmetrical about the midpoint $\lambda_{N+1}=0$. There are many ways in which the grid could be specified. It was decided to first fix the smallest and largest positive grid-points, λ_{N+2} and λ_{2N+1} respectively, and then fix either N or r. As will be seen, N and r are related by $\lambda_{2N+1} = r^{N-1}\lambda_{N+2}$. While, as will be shown, λ_{N+2} and λ_{2N+1} were chosen for individual multimodal searches, the values of N or r were constants specified by the user of the two-part algorithm. Experiments showed that if r was used to fix the grid, this could often result in an excessively fine or too coarse a grid. By using N to fix the grid, these problems were less frequently met, there being no risk of extreme values of N being generated as can happen when r is used to fix the grid.

When fixing λ_{N+2}, it is desirable that the corrections are close to the Newton corrections $-(G^k)^{-1}\underline{g}^k$ at $\lambda=0$ but not so close that grid-points will be wasted in covering a region of little variation in $F(\lambda)$. An initial trial value λ_p for λ_{N+2} is calculated, which is predicted to

give a 1% difference in the values of $\Delta\underline{x}^T\Delta\underline{x}$ at $\lambda=\lambda_p$ and $\lambda=0$. By differentiating equation (2.4) we get

$$\frac{d}{d\lambda}\Delta\underline{x} = -(G^k + \lambda I)^{-1}\Delta\underline{x} \qquad (2.10)$$

A small change $\delta\lambda$ in λ will give, to a first approximation, the change $\delta\Delta\underline{x}$ in $\Delta\underline{x}$ where, from equation (2.10), $\delta\Delta\underline{x} \simeq -(G^k + \lambda I)^{-1}\Delta\underline{x}\delta\lambda$. The change in $\Delta\underline{x}^T\Delta\underline{x}$ is then given by

$$\delta(\Delta\underline{x}^T\Delta\underline{x}) \simeq -2\Delta\underline{x}^T(G^k+\lambda I)^{-1}\Delta\underline{x}\,\delta\lambda \qquad (2.11)$$

Setting $\lambda=0$ and $\delta\lambda=\lambda_p$, the required prediction is

$$\lambda_p = \Delta\underline{x}^T\Delta\underline{x} \;/\; 100\left|2\,\Delta\underline{x}^T(G^k)^{-1}\Delta\underline{x}\right| \qquad (2.12)$$

where $\Delta\underline{x}$ is evaluated at $\lambda=0$. The corrections are then evaluated at $\lambda=\lambda_p$. If it is found that the variation in $\Delta\underline{x}^T\Delta\underline{x}$ is not more than 2% then λ_{N+2} is set to λ_p. Otherwise, λ_p is halved until this requirement is met, which it eventually must be.

At $\lambda=\lambda_{2N+1}$, the corrections must be small and close to the steepest descent direction; however, λ_{2N+1} must not be so large that it leads to wasted grid-points. Again, an initial value λ_p is predicted, this time to give $F(\lambda_p) = 0.99\,F(\infty)$, where the value $F(\infty)$ equals $F(\underline{x}^k)$. From equation (2.7b) it will be seen that for large λ the corrections are approximately $-\underline{g}^k/\lambda$; by a linear approximation the corresponding reduction in $F(\underline{x}^k)$ will be $(\underline{g}^k)^T\,\underline{g}^k/\lambda$. For the required 1% reduction

$$\lambda_p \simeq 100\,(\underline{g}^k)^T\,\underline{g}^k/F(\underline{x}^k) \qquad (2.13)$$

If on evaluation of $F(\lambda_p)$ it is found that the reduction is not more than 2% then λ_{2N+1} is set to λ_p. Otherwise, λ_p is doubled until this condition is satisfied.

This grid was used in conjunction with a Golden Section search to refine the bounds on the minima; encouraging results were obtained. An important feature of the computer implementation was that, following the approach previously-adopted by Cutteridge, a step limit was set such that no component of the vector $\underline{x}$ was allowed to change by more than this limit from one iteration to the next. This aspect of the algorithm will be described more fully later.

It was apparent that the algorithm could be improved in several areas. The corrections were found from equation (2.4) using a method [65] based on Crout factorisation; if the matrix $G^k + \lambda I$ is singular, or nearly-so, then this method fails. The more accurate method of Choleski decomposit--ion could not be used since it requires that $G^k + \lambda I$ is always positive definite. At a singularity, although $\Delta\underline{x}$ is infinite, the step limit would ensure that the corrections are finite provided that the direction of $\Delta\underline{x}$ could be found. The program logic has to have safeguards built in to cope with the Crout factorisation method failing, either when fixing λ_{N+2} and λ_{2N+1} or when evaluating $F(\lambda_i)$; this cannot be done in a wholly satisfactory manner. The calculation of $\Delta\underline{x}$ for any λ is time-consuming and, as Cutteridge pointed out, could be speeded up by a technique described by Jones [45] . This technique uses the eigenvalues and eigenvectors of G^k which can be found numerically by standard methods. The eigenvalues would have a second important use in that they specify the positions of the singularities in the grid search.

The method of Jones can be explained as follows. Since the matrix G^k is both real and symmetric then it will have n real eigenvalues; for example, Wilkinson [26] gives a proof of this property. Let these eigenvalues be denoted by ϕ_i (i=1, .., n) where $\phi_i > \phi_{i-1}$. Further, let the eigenvector corresponding to each eigenvalue be the vector $\underline{v}_i$ which, without loss of generality, can be taken as a unit vector. Because $\underline{v}_i^T \underline{v}_j = 0$

for $i \neq j$ (the eigenvectors form an orthogonal basis) for any value of $\Delta\underline{x}$ there will exist n real scalars α_i such that

$$\Delta\underline{x} = \sum_{i=1}^{n} \alpha_i \, \underline{v}_i \tag{2.14}$$

Substituting $\Delta\underline{x}$ from equation (2.14) into equation (2.4) we obtain

$$\sum_{i=1}^{n} \alpha_i \, (G^k + \lambda I) \, \underline{v}_i = -\underline{g}^k$$

and since, by definition, $G^k \underline{v}_i = \phi_i \underline{v}_i$ this last expression simplifies to

$$\sum_{i=1}^{n} \alpha_i (\phi_i + \lambda) \, \underline{v}_i = -\underline{g}^k \tag{2.15}$$

If both sides of equation (2.15) are pre-multiplied by $\underline{v}_i^T$ it follows from the orthogonality of the eigenvectors that

$$\alpha_i (\phi_i + \lambda) = -\underline{v}_i^T \, \underline{g}^k \tag{2.16}$$

noting that $\underline{v}_i^T \underline{v}_i = 1$. If α_i from equation (2.16) is substituted into equation (2.14) we obtain

$$\Delta\underline{x} = -\sum_{i=1}^{n} \frac{1}{\phi_i + \lambda} \, \underline{v}_i^T \underline{v}_i \, \underline{g}^k \tag{2.17}$$

Thus for any value of λ, the corresponding correction vector can be found directly from equation (2.17). A further saving in processor time is made by programming the expression for $\Delta\underline{x}$ as

$$\Delta\underline{x} = Z \, \underline{w} \tag{2.18a}$$

$$Z = [\underline{z}_1, \underline{z}_2, \ldots \underline{z}_n] \tag{2.18b}$$

$$\underline{z}_i = -\underline{v}_i^T \, \underline{v}_i \, \underline{g}^k \tag{2.18c}$$

$$w_i = \frac{1}{\phi_i + \lambda}$$

Thus if Z is evaluated once at the start of the nonunimodal search, the corrections are thereafter given by a single matrix-vector multiplication.

Before describing how the eigenvalues were used to specify the grid, it is necessary to discuss the effect on the search of the limitation on the size of the corrections as mentioned earlier. Cutteridge had found from past experience that it was beneficial to limit the magnitude of the components of $\Delta\underline{x}$; this he found improved the global convergence properties of the algorithms he used and prevented numerical difficulties which could often be caused by one iteration taking a large step in the variables from which subsequent iterations tried to recover. He used one of two basic methods. The first was a simple "cut-off" such that if $|\Delta x_i| > p_i$ then Δx_i is set to p_i multiplied by the sign of Δx_i. The parameter p_i is a pre-set value; it will be assumed that the same value p will be used for all the p_i values but some further comments on the possibility of using differing p_i values will be made later. The second method of limitation is by "scaling-down" in which $\Delta\underline{x}$ is multiplied by the scalar $s \leq 1$ where

$$s = \min \left(1, \frac{p}{|\Delta x_1|}, \ldots \frac{p}{|\Delta x_n|}\right) \qquad (2.19)$$

It will be seen that s is defined such that if the maximum absolute correction component does not exceed p then s=1 and $\Delta\underline{x}$ is unchanged; otherwise $\Delta\underline{x}$ is reduced to make this maximum correction exactly equal to p in magnitude. Both methods of limitation were tried in the present algorithm; scaling-down was more consistently successful and was therefore used in the final version of the program. One possible explanation for the greater success of the scaling-down method is that in the multimodal search there are many regions in which scaling applies. The cut-off method changes both direction and length, whereas the scaling down method changes the length only. Consequently if a downhill direction is generated i.e. $(\underline{g}^k)^T \Delta\underline{x} < 0$ the cut-off method could make this into an uphill direction.

It will be noted from equations (2.18d) that w_i becomes infinite when $\lambda = -\phi_i$ corresponding to a singularity of $G^k + \lambda I$. Thus in equation (2.18a) the i^{th} column $\underline{z}_i$ of the matrix Z will be dominant, all other columns being multiplied by finite values of $\underline{w}$. The corrections $\Delta\underline{x}$ will therefore have the same direction as $\underline{z}_i$. Scaling-down will be in effect and so at such a singularity

$$s = \min \left(\frac{p}{|Z_{1i}|}, \ \ldots \ \frac{p}{|Z_{ni}|} \right) \tag{2.20a}$$

$$\Delta\underline{x} = \pm \, s \, \underline{z}_i \tag{2.20b}$$

The sign chosen in equation (2.20b) will depend upon the direction from which the singularity is approached. If λ tends to $-\phi_i$ from values greater than $-\phi_i$, then the sign will be positive; otherwise the sign will be negative. Thus although the corrections are finite at $\lambda=-\phi_i$, they still exhibit a discontinuity in their values.

The scaling-down can itself introduce discontinuities to the slope of the corrections. We may write

$$\frac{d}{d\lambda} (s\Delta\underline{x}) = s \frac{d}{d\lambda} \Delta\underline{x} + \Delta\underline{x} \frac{ds}{d\lambda} \tag{2.21}$$

It will be apparent from equation (2.17) that $d\Delta\underline{x}/d\lambda$ is continuous in the range $-\phi_i < \lambda < -\phi_{i-1}$, assuming that $\phi_i > \phi_{i-1}$ (i=2, .. n). However in this range the value of $ds/d\lambda$ can be discontinuous as seen from equation (2.19). In a region where scaling-down is not in effect, s=1 and $ds/d\lambda=0$; when scaling-down is in effect, then $s=p/|\Delta x_{imax}|$ where $|\Delta x_{imax}|=\max(|\Delta x_1|,.. |\Delta x_n|)$ and $ds/d\lambda \neq 0$. At the junction of two such regions, $ds/d\lambda$ is therefore discontinuous. Similarly, a discontinuity occurs at the boundary of two regions in which scaling-down is in effect but the index imax changes. Dowson [66] observed that the discontinuities in $ds/d\lambda$ could have a very

marked effect on the form of $F(\underline{x}^k + \underline{\Delta x})$ and developed an improved grid search to take account of this fact. Some further work done by the author on this aspect of the multimodal search is discussed in section 4.2.4.

One further point that must be borne in mind when choosing the grid is that, in general, the eigenvalues of G^k may not all be distinct. In the event of repeated eigenvalues, then equations (2.18a) - (2.18d) must be slightly modified by reducing the number of columns of Z, by adding eigenvectors which correspond to equal eigenvalues, and at the same time reducing the length of the vector $\underline{w}$. The effect of rounding errors must always be allowed for in testing for equality of eigenvalues. Hence two eigenvalues were assumed to be equal if $\phi_i - \phi_{i-1} \leqslant \varepsilon_\phi$ where ε_ϕ is the maximum likely rounding error. The choice of a suitable value for ε_ϕ is difficult. Since the effect of small values of λ on the corrections depends upon their effect on the diagonal of the matrix G^k, ε_ϕ as used in the program was taken, somewhat arbitrarily, as

$$\varepsilon_\phi = 10^{-8} \max(10^{-8}, |G_{11}^k|, .. |G_{nn}^k|)$$

The grid was set up as shown in Figure 2.1; the description which follows needs to be slightly modified if repeated eigenvalues are present. There are n-1 interior regions, bounded by the discontinuities in $\underline{\Delta x}$ at the points $\lambda = -\phi_i$. In addition there are two end-regions, the descent region in which $\lambda > -\phi_1$ and the ascent region in which $\lambda < -\phi_n$. Each region is considered in isolation from the others by the multimodal search. The interior regions are each subdivided into N equal intervals of λ by the grid points λ_j (j=1, .. N+1) where

$$\lambda_j = -\phi_i + (j-1)(\phi_i - \phi_{i-1})/N \qquad (2.22)$$

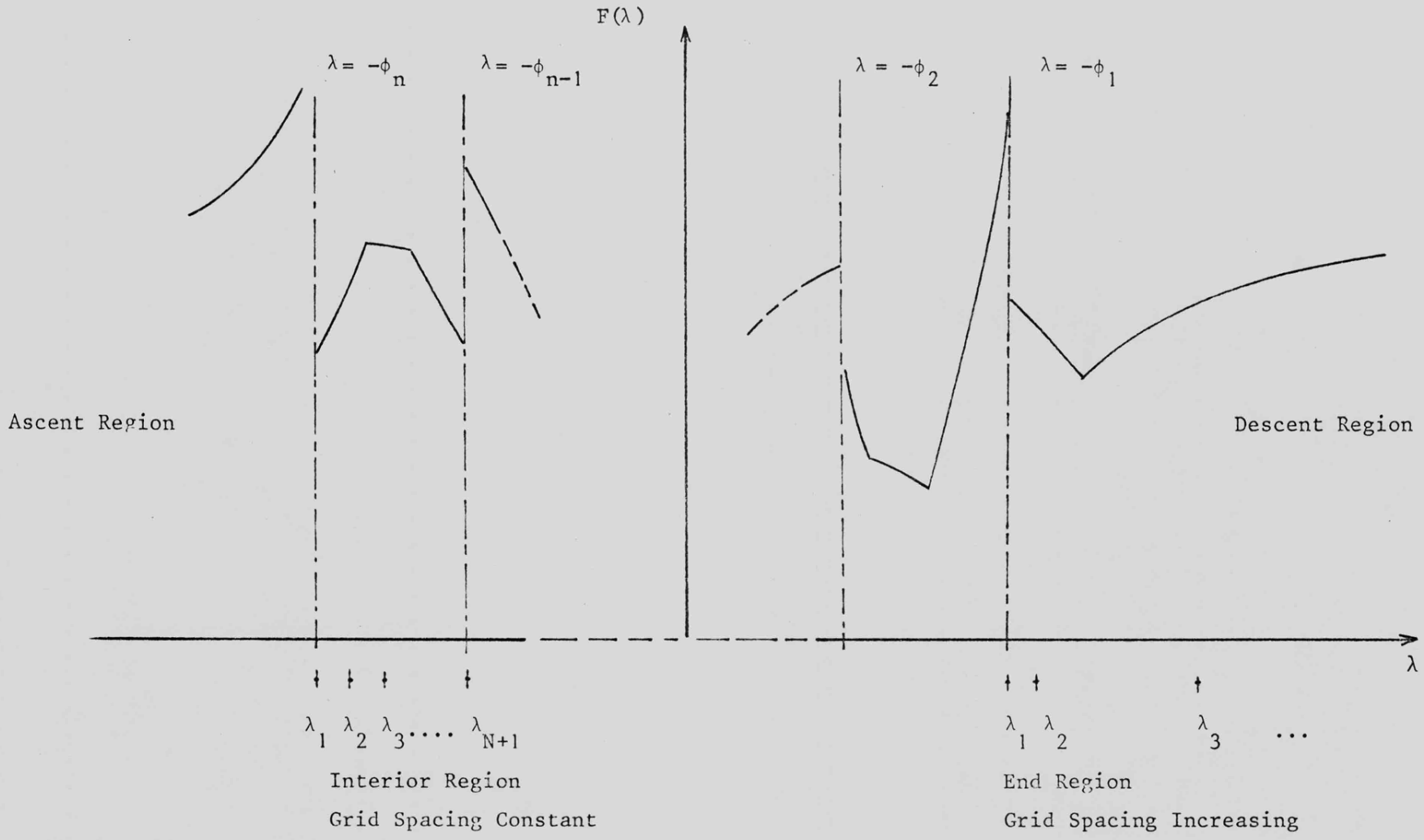


Figure 2.1 Multimodal Function (schematic) showing choice of grid

It should be noted that λ_1 and λ_{N+1} coincide with the values $-\phi_i$ and $-\phi_{i-1}$ respectively and so the corrections must be evaluated from equations (2.20a) and (2.20b) in this instance. The end regions use a different grid; for the descent region this is defined as λ_i (i=1, ... M) where

$$\lambda_1 = -\phi_1 \tag{2.23a}$$

$$\lambda_2 = \lambda_1 + (\phi_2-\phi_1)/N \tag{2.23b}$$

$$\lambda_i = \lambda_{i-1} + 10(\lambda_{i-1}-\lambda_{i-2}) \quad i=3, .. M \tag{2.23c}$$

The grid is terminated at the first value $\lambda=\lambda_M$ for which $0.95F(\underline{x}^k)<F(\underline{x}^k+\Delta\underline{x})<F(\underline{x}^k)$. A similar scheme is used for the ascent region. If there is only one distinct eigenvalue then equation (2.23b) will be replaced by $\lambda_2=\lambda_1 + 10^{-3}$ min $(1,|G^k_{11}|, .. |G^k_{nn}|)$ in the program.

Two situations exist when looking for minima of $F(\lambda)$. First, a true minimum is found if on three successive grid points $F_{i-1} > F_i > F_{i+1}$; in this event the minimum is located precisely using the safeguarded quadratic interpolation method described below. Second, a quasi-minimum is assumed to exist if $F_1 < F_2$ or, for interior regions only, $F_{N+1} < F_N$; in these cases the program assumes that $dF(\underline{x}^k+\Delta\underline{x})/d\lambda > 0$ at $\lambda=-\lambda_1$ or $\lambda=\lambda_{N+1}$ and accepts these as minima. Dowson [66] found that occasionally this assumption was erroneous and that in fact a true minimum did exist between the singularity and the adjacent grid point.

A safeguarded quadratic interpolation illustrated in Figure 2.2a was used in preference to a Golden Section or Fibonacci search because it would make full use of the three values making up the bracket on each minimum found. If the initial bracket is represented by the points (λ_1,F_1), (λ_2,F_2) and (λ_3,F_3) (where the subscripts now indicate adjacent points in the grid) then we can fit a

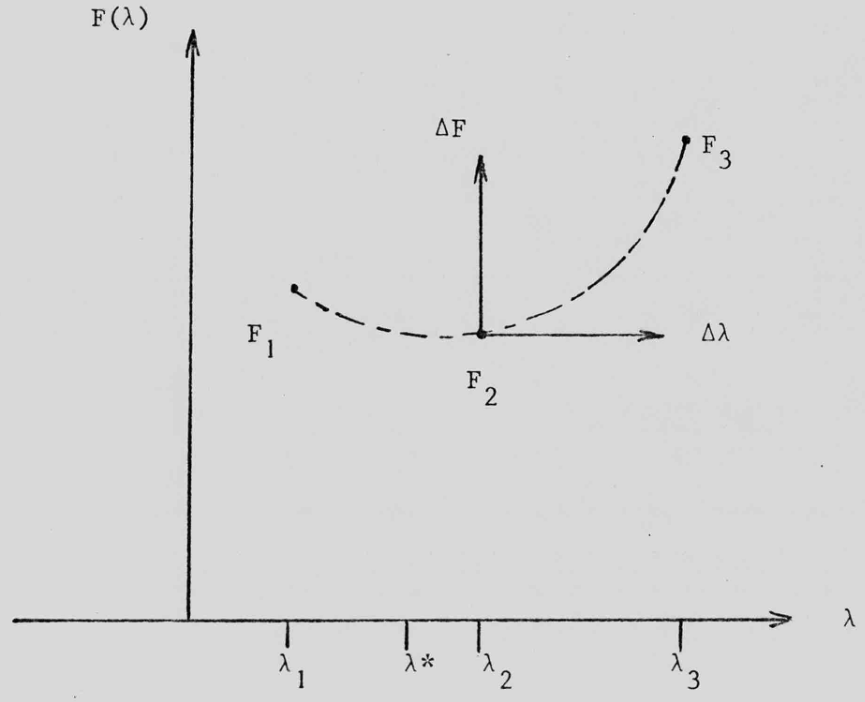


Figure 2.2a Quadratic Interpolation

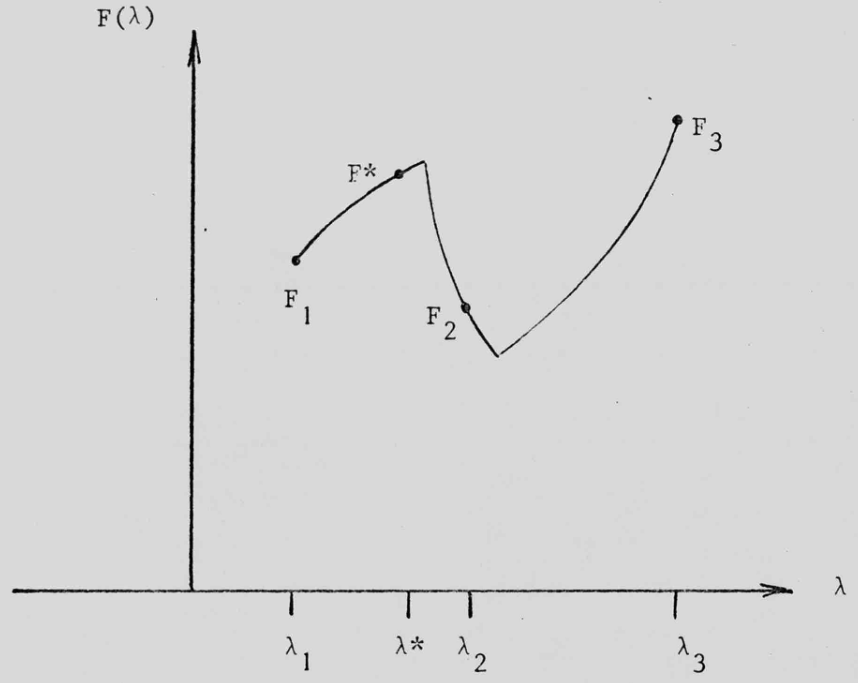


Figure 2.2b Detection of Nonunimodality

quadratic through these points by the expression

$$\Delta F = A\,\Delta\lambda + B\Delta\lambda^2 \qquad (2.24)$$

where $\Delta F=F-F_2$ and $\Delta\lambda=\lambda-\lambda_2$. The constants A and B are readily found by solving the two linear equations given by the requirement that the curve passes through the points (λ_1,F_1) and (λ_3,F_3). Since $\Delta F=0$ and $\Delta\lambda=0$ at (λ_2,F_2), there is no constant term in equation (2.24). By differentiating ΔF with respect to $\Delta\lambda$ and equating to zero, the minimum of ΔF must lie at $\Delta\lambda=\Delta\lambda^*$ where $\Delta\lambda^*=-A/2B$. Substituting the values for A and B, and writing $\Delta F_i=F_i-F_2$, $\Delta\lambda_i=\lambda_i-\lambda_2$, we obtain

$$\Delta\lambda^* = \tfrac{1}{2}\,\frac{\Delta F_1\Delta\lambda_3^{\,2} - \Delta F_3\ \Delta\lambda_1^2}{\Delta F_1\Delta\lambda_3 - \Delta F_3\ \Delta\lambda_1} \qquad (2.25a)$$

Simple algebraic manipulation of equation (2.25a) gives

$$\Delta\lambda^* = \frac{\Delta\lambda_1}{2} + \frac{\Delta F_1\ \Delta\lambda_3(\Delta\lambda_3-\Delta\lambda_1)}{\Delta F_1\ \Delta\lambda_3 - \Delta F_3\Delta\lambda_1} \qquad (2.25b)$$

and also

$$\Delta\lambda^* = \frac{\Delta\lambda_3}{2} + \frac{\Delta F_3\Delta\lambda_1(\Delta\lambda_3-\Delta\lambda_1)}{\Delta F_1\Delta\lambda_3 - \Delta F_3\Delta\lambda_1} \qquad (2.25c)$$

Initially, it is known that $\Delta\lambda_1 < 0$ and $\Delta F_1 > 0$, $\Delta F_3 > 0$ and $\Delta\lambda_3 > 0$; as will be shown, these relations are maintained throughout the quadratic interpolation. Thus, from equations (2.25b) and (2.25c) it follows that

$$\frac{\Delta\lambda_1}{2} < \Delta\lambda^* < \frac{\Delta\lambda_3}{2} \qquad (2.26)$$

showing that predicted minimum will never be more than halfway from the middle point to either of the outer points. If $\Delta\lambda^*$ is close to zero then the

new trial value of $\lambda^* = \lambda_2 + \Delta\lambda^*$ would be too close to λ_2 to provide useful additional information. To safeguard against this, $\Delta\lambda^*$ was restricted so that if $\Delta\lambda_1/10 < \Delta\lambda^* < 0$ then $\Delta\lambda^*$ is set to $\Delta\lambda_1/10$ and if $\Delta\lambda_3/10 > \Delta\lambda^* > 0$ then $\Delta\lambda^*$ is set to $\Delta\lambda_3/10$.

Having obtained λ^*, the value $F^* = F(\lambda^*)$ is calculated and the three points in the bracket adjusted as follows. If $\lambda^* < \lambda_2$ then the new bracket is formed by $(\lambda^*, \lambda_2, \lambda_3)$ if $F^* \geq F_2$ and by $(\lambda_1, \lambda^*, \lambda_2)$ if $F^* < F_2$. Similarly, if $\lambda^* > \lambda_2$ then the bracket is changed to $(\lambda_1, \lambda_2, \lambda^*)$ if $F^* \geq F_2$ and to $(\lambda_2, \lambda^*, \lambda_3)$ if $F^* < F_2$. In all cases, one of the two outer points is discarded and the bound on the minimum is thereby reduced. The whole process is repeated until the bound is acceptably small, as given by $\lambda_2 - \lambda_1 < \varepsilon_1|\lambda_2|$ and $\lambda_3 - \lambda_2 < \varepsilon_1|\lambda_2|$. A second convergence criterion was provided such that the interpolation terminated if $F_1 - F_2 < \varepsilon_2 F_2$ and $F_3 - F_2 < \varepsilon_2 F_2$. This was only used if F_2 was greater than the lowest minimum so far found by the multimodal search on the current descent iteration, on the assumption that the minimum bracketed is unlikely to be the global minimum. A fail-safe limit of 50 iterations was imposed. Also, a further safeguard to ensure that the bound steadily decreased was included whereby if, on three successive interpolations, one outer point remained the same then the next value of λ^* was taken as midway between this outer point and the middle point λ_2. It should be observed that if $F(\lambda)$ is in fact not unimodal in the range of the bracket λ_1 to λ_3, then the search will still converge to a minimum, although not necessarily the lowest minimum if there are should be two or more within the bracket. Nonunimodality is detected when $\lambda^* < \lambda_2$ and $F^* > F_1$ or when $\lambda^* > \lambda_2$ and $F^* > F_3$, as illustrated in Figure 2.2b; such a situation certainly indicates a maximum, but not necessarily a further minimum.

2.3 The modified Gauss-Newton algorithm

For the second part of his two-part algorithm, Cutteridge used a modified Gauss-Newton search. The classical Gauss-Newton method takes the corrections $\Delta\underline{x} = -(J^TJ)^{-1} J^T\underline{f}$ at iteration k where $\underline{f}$ is the vector of functions and J is the Jacobian of $\underline{f}$, both evaluated at $\underline{x}^k$. If the number of functions m equals the number of variables n then the corrections are identical to those of the classical Newton (or Newton-Raphson) iterative method for solving sets of nonlinear equations. In the latter case, the corrections have the simpler form of $\Delta\underline{x} = -J^{-1}\underline{f}$. In the modified Gauss-Newton method, the iteration $\underline{x}^{k+1} = \underline{x}^k + \alpha_m \Delta\underline{x}$ is performed where α_m is found from a line search to minimize the function $F(\alpha) = F(\underline{x}^k+\alpha\Delta\underline{x})$; this modification was used earlier by Hartley [37] to improve the global convergence of the Gauss-Newton method.

This section describes the particular implementation of the modified Gauss-Newton search which was developed by the author. The two main aspects of interest are the line-search used to locate α_m and the criteria used to terminate the Gauss-Newton search.

2.3.1 The search for α_m

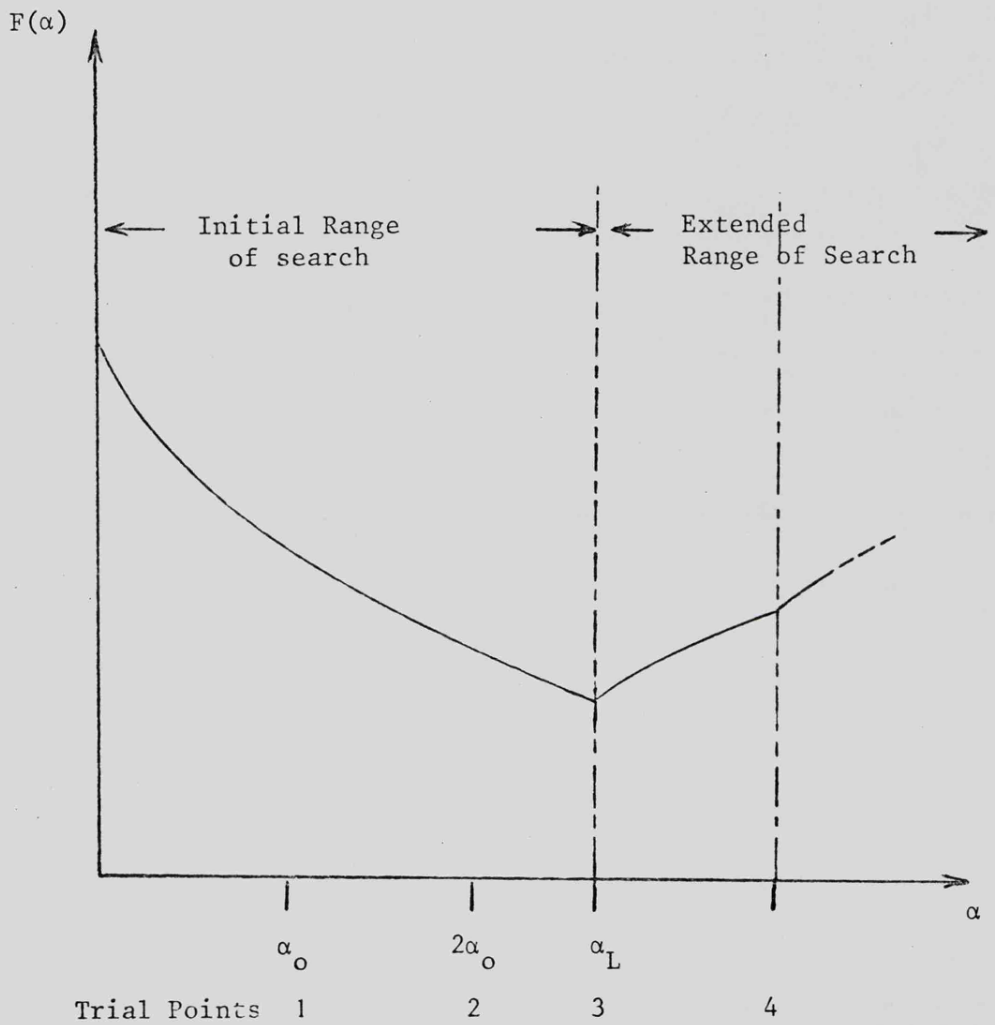
IF $F(\alpha)$ is expanded by a Taylor series and differentiated it is found that $dF(\alpha)/d\alpha = (\underline{g}^k)^T \Delta\underline{x}$ at $\alpha = 0$; as before $\underline{g}^k$ is the gradient vector of $F(\underline{x})$ evaluated at $\underline{x}^k$. Now $J^T\underline{f} = \underline{g}^k$ and from the definition of $\Delta\underline{x}$ we have $J^T\underline{f} = -J^TJ\Delta\underline{x}$. We can therefore replace $\underline{g}^k$ by $-J^TJ\Delta\underline{x}$ and obtain $dF(\alpha)/d\alpha = -(J^TJ\Delta\underline{x})^T \Delta\underline{x}$. Some manipulation finally gives for $\alpha = 0$

$$\frac{dF}{d\alpha}(\alpha) = -\Delta\underline{x}^T J^T J \Delta\underline{x} \qquad (2.27)$$

Equation (2.27) shows that the Gauss-Newton corrections are downhill and that there will exist an $\alpha_m > 0$ for which $F(\underline{x}^{k+1}) < F(\underline{x}^k)$ provided that J^TJ is not singular.

Unfortunately, it is often found in practice that J^TJ does become singular. Since round-off error will always be present in the calculations, this situation is manifested by $\Delta\underline{x}$ becoming very large with the corresponding value of α_m becoming very small; ultimately, break-down will occur of the numerical method for solving linear simultaneous equations which must be used to compute $\Delta\underline{x}$. The program used a method [65] based on the Crout factorisation of J^TJ; a much more accurate method is that due to Golub [67]. The author is not convinced that it is preferable to use a more accurate method. It often happens that the Gauss-Newton search will converge even when started from a point at which J^TJ is almost singular; in such cases, a less accurate method might fail at the outset to compute $\Delta\underline{x}$ because J^TJ is effectively singular within the accuracy of the method. On the other hand, a more common occurrence is for the Gauss-Newton search to fail to converge and the matrix J^TJ to progressively get closer to singularity as the search continues; in this case a more accurate method will simple prolong the onset of failure and result in wasted computation.

As with the descent algorithm, the change in any value x_i at an iteration of the Gauss-Newton search was restricted to a set limit of p. This was accounted for by the line-search used to locate α_m. The search is illustrated in Figure 2.3 and consisted of a preliminary search to bracket α_m followed by a quadratic interpolation to locate α_m to greater accuracy. The preliminary search is confined to the range $0 \leqslant \alpha \leqslant \alpha_L$ where $\alpha_L = p/\max(|\Delta x_i|, .. |\Delta x_n|)$. Starting with a given value α_o, values of α are calculated from the Fibonacci series α_o, $2\alpha_o$, $3\alpha_o$, $5\alpha_o$, ... with a cut-off at α_L. The values of $F(\alpha)$ are computed and inspected to see whether three successive values of α bracket a minimum; note that the value of $F(\alpha)$ at $\alpha = 0$ is known and included. There is no significance in the use of a Fibonacci series other than it is a convenient way of generating values



1. Initial Guess
2. Second term in Fibonacci series
3. Largest correction at limit
4. Second largest correction at limit

Figure 2.3 Search for the minimum of $F(\alpha)$

of α at reasonable spacing. Also, the line-search aims at choosing an α_o which is close to α_m; if this aim is achieved then the bracket on α_m is likely to be given by the three equally-spaced α values $(o,\alpha_o, 2\alpha_o)$ or $(\alpha_o, 2\alpha_o, 3\alpha_o)$.

Should the cut-off point be reached without obtaining a bracket on α_m, then up to n-1 further trial values of α are used. These further values correspond to the remaining corrections in turn reaching the limit of $\pm$ p. There will be less than n-1 values if any correction is zero or if there should be identical corrections. In this extended range, $\alpha\Delta\underline{x}$ changes direction in a discontinuous fashion at each trial value of α; as a result, $F(\alpha)$ is often nonunimodal. Although no special provision was made to cope with nonunimodality, the extended range often gave a worthwhile reduction in $F(\alpha)$ below its value at α_L.

The safeguarded quadratic interpolation algorithm that was used to locate α_m more accurately, given a bracket, was the same as that described in section 2.2.2 for the descent algorithm. It should be noted that sometimes it was found that $F(\alpha)$ decreased up to the maximum possible value of α = p/min $(|\Delta x_1|, \ldots |\Delta x_n|)$ at which all the modified corrections equal $\pm$ p; no interpolation is needed in this event.

The choice of α_o is based on simple considerations. At the first iteration of a Gauss-Newton search, α_o is taken as the lower of the two values 0.4 and $0.4p/|\Delta x_{imax}|$. The first value will be used when $|\Delta x_{imax}| < p$; since α_m will usually be less than unity, a minimum is likely to be bracketed by the values of α of (0, 0.4, 0.8). When the second value is used, the Fibonacci series will be terminated at the third term. On the second and subsequent Gauss-Newton iterations, α_o is set to the lowest of the three values 0.4, $0.4p/|\Delta x_{imax}|$ and $2\alpha_m/3$, where α_m is the value of α_m obtained at the previous iteration. Since α_m does not

change greatly from one iteration to the next, the third option can, if used, be expected to bracket the minimum by the points (0, $2\alpha_m/3$, $4\alpha_m/3$).

Having chosen α_o in the way described, if $F(\alpha_o) < F(\underline{x}^k)$ then the value of α_o is accepted; otherwise α_o is divided by 10, and this division repeated if necessary, until either $F(\alpha_o) < F(\underline{x}^k)$ or the modified corrections $\alpha_o \Delta\underline{x}$ become negligibly small. In the latter case, the search for α_m fails; however, since this situation normally occurs when J^TJ is nearly singular the Gauss-Newton search itself would fail.

2.3.2 The criteria used to terminate the Gauss-Newton search

The Gauss-Newton search can be terminated in two ways. Either it converges, and a solution to the problem will have been obtained, or it fails, in which case control is passed back to the descent algorithm after the variables have been reset to the values they had on entry to the Gauss-Newton search. It is crucial that reliable criteria are used to detect the occurrence of either event. Also, in the case of failure, it improves the efficiency of the two-part algorithm if the onset of failure can be predicted in advance. The criteria incorporated into the program were evolved by numerical experiment; they fall into four categories.

First, convergence to a minimum of $F(\underline{x})$ is assumed when $|\Delta x_i| < \varepsilon$ (i=1, .. n). This is a simple-to-apply criterion but depends upon the value specified for ε being attainable within the accuracy of the computations. A relative, rather than absolute, criterion could be used such that convergence is assumed when $|\Delta x_i| < \varepsilon |x_i|$ (i=1, .. n); this is to be preferred if the variables differ greatly in size when a single value would not be appropriate. Provided that the variables are roughly the

same size, the first criterion is adequate. In the tests on the difficult problem, the variables used in the search were the natural logarithms of the problem variables; this was to ensure that the solution values would all be positive. With such a transformation, when $|\Delta x_i| < \varepsilon$ then the change in the problem variable corresponding to the change $|\Delta x_i|$ in the search variable will be less than $(\exp(\pm\varepsilon)-1)$ times the problem variable; the sign of ε will depend upon the sign of Δx_i. If ε is small, then the change will be approximately $\pm\varepsilon$ times the problem variable and so the convergence criterion is, in this case, a relative one.

The second category consists of three criteria to determine immediate failure. The Gauss-Newton search is entered prior to each iteration of the descent algorithm. If less than N_D descent iterations have been performed, if the objective function is greater than F_E, or if the Gauss-Newton correction of maximum magnitude $|\Delta x_{imax}|$ exceeds Δ_L, then immediate failure is assumed and control passes back to the descent algorithm. The first two criteria are consistent with the philosophy of two-part algorithms in that several iterations are performed in the first part before the second part is entered. Since it cannot be known in advance when the descent part will be sufficiently close to a solution for the Gauss-Newton search to converge, these two criteria were not used for the tests; they were left in the program as a means whereby a user could ensure that a minimum amount of progress is made in the descent algorithm if he so desired. The third of these criteria was not as useful as first thought. Trials showed that convergence to a solution was attained in some cases where the value $|\Delta x_{imax}|$ on entry was as high as 10^8. Consequently, this criterion too was not used in the tests, although it was retained as an option in the program. The attitude adopted by the author was that the

more times the Gauss-Newton search is attempted, the more likely it is that a solution of a difficult problem will be obtained; if the problem is an easy one, then an early attempt will be successful anyway and so no computation will be wasted.

The third category consists of two criteria for forced failure. If J^TJ is singular, as evinced by a failure of the numerical method of solution for the corrections, or if the search on α_m cannot find α_o such that $F(\alpha_o) < F(\underline{x}^k)$ then forced failure has occurred. The second mode of failure is caused by the matrix J^TJ approaching singularity. The effect of round-off in the calculations determines which mode of failure occurs first.

The fourth and final category contains three criteria for the prediction of ultimate failure. The first of these is a straightforward limit N_G on the number of iterations allowed in a single Gauss-Newton search. The other two criteria are prediction criteria in a truer sense; both were used by Cutteridge in previous algorithms. If $|\Delta x_{imax}|$ increases to 100 times, or more, its value on entry, or if $|\Delta x_{imax}|$ increases on ten successive iterations and the increase gets larger at each successive iteration, then it is assumed that J^TJ will eventually become singular and that the search will fail. Note that the second of these two criteria indicates an acceleration in the rate of increase of $|\Delta x_{imax}|$.

2.4 The restart facility

Using the descent algorithm and modified Gauss-Newton algorithm described in the preceding sections, the two-part program obtained solutions to Skwirzynski's problem from a wide range of starting values. However, the program was not completely reliable and in some cases reached a local

minimum of $F(\underline{x})$ in the descent part without converging to the global minimum of zero in the Gauss-Newton part. The author and Cutteridge jointly came to the conclusion that a facility for the automatic restart of the descent algorithm at a different starting value of $\underline{x}$ would improve the reliability. The restart approach has been used by other workers, see for example Dixon [48] ; commonly the new starting value is generated in some random manner. The restart facility developed by Cutteridge and the author does not formally use a random process, although it still has an element of randomness to it.

It will be recalled that the descent algorithm at each iteration, chooses the value of λ corresponding to the global minimum of $F(\lambda)$. As a by-product of the multimodal search, one or more other values of λ are normally found corresponding to local minima which are greater than the global minimum. The corrections obtaining at these local minima can often be far removed in direction and magnitude from those at the global minimum. The new restart facility uses only those local minima which, if used in place of the global minimum would still give a reduction, albeit smaller, in the objective function. By restarting from the value $\underline{x}$ corresponding to a local minimum, the subsequent path followed by the descent algorithm will diverge from the originally-obtained path.

This facility is implemented in the program as follows. On the original descent path, a list of potential restarts is built up by adding to the list at each descent iteration the values of $\underline{x}$ corresponding to local minima as just discussed. The list is kept in ascending order of descent iteration and, for the local minima at a given iteration, in ascending order of size of objective function. Should the algorithm fail, as evinced by $F(\underline{x})$ changing by less than 1% on three successive descent iterations, then it is restarted from the first point on the list. Further restarts are made as necessary until either convergence is achieved or else the list is exhausted.

2.5 The final form of the algorithm

Before proceeding to the next section, which discusses the results obtained with the program, it is worthwhile to summarise the main steps of the two-part algorithm, as in its final form. These steps are:

Step 1 Initial entry; set k=0 and $\underline{x}^o$ to a given estimate of the solution variables $\underline{x}^*$

Step 2 Calculate the functions $\underline{f}$, Jacobian J and the second derivatives at the current point $\underline{x}^k$. Form the Hessian G^k and gradient vector $\underline{g}^k$ of $F(\underline{x})$.

Step 3 Enter the Gauss-Newton search. If any of the immediate ejection criteria are satisfied, then continue from step 10; otherwise, store $\underline{x}^s=\underline{x}^k$, $k^s=k$ and continue with the Gauss-Newton search at step 4.

Step 4 If J^TJ is singular, continue from step 9; otherwise compute the Gauss-Newton corrections $\Delta\underline{x}=-(J^TJ)^{-1}J^T\underline{f}$

Step 5 If $\max(|\Delta x_1|, \ldots |\Delta x_n|)<\varepsilon$ then convergence to a solution is assumed, set $\underline{x}^* = \underline{x}^k$ and terminate the optimization.

Step 6 If any of the predicted failure criteria are satisfied, continue from Step 9.

Step 7 Find the value α_m of α to minimize $F(\underline{x}^k+\alpha\Delta\underline{x})$, allowing for the effect of the limit p on the corrections Δx. If $F(\underline{x}^k+\alpha_m\Delta\underline{x}) \nmid F(\underline{x}^k)$ then continue from Step 9.

Step 8 Set $\underline{x}^{k+1} = \underline{x}^{k} + \alpha_m \Delta\underline{x}$ and k=k+1. Compute $\underline{f}$ and J at $\underline{x}^{k}$ and continue the Gauss-Newton search at Step 4.

Step 9 Failure exit from the Gauss-Newton search; set $k=k^s$ and $\underline{x}^{k}=\underline{x}^{s}$.

Step 10 Perform a single iteration of the descent algorithm. Calculating the corrections by $\Delta\underline{x} = -(G^k+\lambda I)^{-1}\underline{g}^{k}$, if necessary scaled down to ensure that no correction $|\Delta x_i|$ exceeds the step limit p, use the multimodal search on λ to find the global minimum of $F(\underline{x}^{k}+\Delta\underline{x})$. If the descent algorithm has not yet been restarted, add to the list of restart points using any suitable local minima found by the multimodal search.

Step 11 Set $\underline{x}^{k+1} = \underline{x}^{k} + \Delta\underline{x}$ where $\Delta\underline{x}$ corresponds to the global minimum found at Step 9.

Step 12 If progress in the descent algorithm is tailing off, then either restart the algorithm and continue from Step 2 or, if no more restarts are available, terminate the optimization. Otherwise, if the descent algorithm is still making good progress, continue from Step 2 with the current values for $\underline{x}^{k}$.

The program has default values for the various parameters it employs; these can, if desired, be changed by the user. To date, the only parameter which it has been necessary to alter is the convergence parameter ε, which has a default of 10^{-8}. The step limit p is set to 0.5 for both the descent and Gauss-Newton corrections. In the multimodal search, the number of

intervals N used to subdivide the interior ranges is set to 3; this value can be increased, as Dowson [66] did, to increase the probability that the multimodal search will locate all the minima. The convergence criteria in the quadratic interpolation use the values ε_1 and ε_2. For the descent algorithm, these are set to 10^{-3} and 10^{-2} respectively, for the Gauss-Newton algorithm they are set to 10^{-2} and 0. Note that the effect of setting either parameter to zero is to ensure that the test for convergence using that parameter will not be operative

2.6 Numerical experience with the algorithm

Originally the algorithm was developed as an ALGOL 60 program on an ICL (Elliot) 4130 computer. The final version was later translated into FORTRAN IV on the same computer and has, more recently, been transferred to an IBM 360/44. Unless stated to the contrary, the results given in this section were obtained using the IBM 360/44 version. Note that double precision floating-point arithmetic was employed; for this the IBM uses a 56-bit mantissa which enables real numbers to be stored to approximately 14 significant decimal digits of accuracy.

This section first discusses the performance of the program on the difficult test problem. Then a modified form of the descent algorithm is discussed which does not require second derivatives. The effect of using numerical approximations of the derivatives rather than analytical expressions is described. Some statistics are given which summarise the behaviour of the important internal features of the two-part algorithm. Lastly, some comparisons are made with other algorithms, using the difficult test problem together with a further eight test problems.

2.6.1 Performance on the difficult test problem

The difficult test problem due to Skwirzynski was used as the touchstone by which to gauge the reliability achieved by the two-part algorithm. Details of this problem are given in the Appendix, where it is referred to as Problem 1. As can be seen, the problem is one of eight nonlinear equations involving eight variables and has an exact solution. There is a pronounced nonlinearity in the equations due to the presence of exponential terms. With some algebraic manipulation the problem could be simplified so that fewer variables are present; this was not done since it was desired to preserve the difficulty of the problem.

The equations arise from the use of the Ebers-Moll[68] model of transistor junctions and, from physical considerations, the variables cannot be negative. The problem is thus one of minimizing a sum of squares subject to non-negativity constraints on the variables. However, it can be transformed to an unconstrained problem by using new search variables $x'_i = \log_e x_i$; this technique is discussed further in [9, p.82] . It will be seen that the original variables $\underline{x}$ will always be non-negative whatever the values adopted by $\underline{x}'$ during the optimization.

From the programming viewpoint, the author believes that all such transformations should always be kept separate from both the optimization algorithm and the subroutine, supplied by the user, to compute function values and derivatives. Earlier programs used by Cutteridge had the logarithmic transformation embedded in the relevant places in the algorithm; the general usefulness of the program was thereby reduced. The present program used a FORTRAN subroutine to act as an interface between the algorithm and user-supplied subroutine. The optimization algorithm calls

this interface subroutine as necessary and supplies it with the current values of the search variables $\underline{x}'$, together with an integer to specify whether the function values only are required or whether first and/or second derivatives are required as well. The interface subroutine then carries out the reverse transformation $x_i = \exp(x'_i)$ to obtain the values of the problem variables corresponding to the current search variables. Then the user-supplied subroutine is called and supplied with the values of $\underline{x}$ with which to calculate the required function and derivative information. The interface subroutine then, as appropriate, transforms the derivative information to apply to the variables $\underline{x}'$, before returning the function and derivative values to the optimization algorithm.

This approach may appear cumbersome; however it is relatively easy to incorporate different transformations by simply changing the interface routine. Other facilities, such as checking of the consistency of the function and derivative calculations in the user-supplied routine or the option to use numeric estimates of the derivatives, can also be included as user-options in the interface routine. Note the modular structure of FORTRAN IV is suited to the programming of optimization algorithms, since only the user-supplied subroutine need be compiled at run-time provided that the subroutines making up the algorithm are kept in compiled form on disk and loaded with the user-subroutine.

It should be noted that for the logarithmic transformation the derivatives with respect to the variables x'_i are calculated from the expressions

$$\frac{\partial f_k}{\partial x'_i} = x_i \frac{\partial f_k}{\partial x_i}$$

$$\frac{\partial^2 f_k}{\partial x'_i \partial x'_j} = x_i x_j \frac{\partial^2 f_k}{\partial x_i \partial x_j} + \delta_{ij} x_i \frac{\partial f_k}{\partial x_i}$$

where $\delta_{ij} = 0$ if $i \neq j$ and $\delta_{ii} = 1$.

Using the standard version of the program summarised in section 2.5, the results shown in Table 2.1 were obtained. Fifteen different starting values were tried; for each starting value the initial estimate x_i^o of each variable was kept the same for each variable. It will be seen that values of x_i^o in the range 0.1 to 10 were used. At the top end of this range, the function values become very large and although solutions were obtained with $x_i^o = 11$ and $x_i^o = 12$ the algorithm breaks down at higher values of x_i^o because of numerical overflows. However, such high values would not be realistic starting values anyway. At the bottom end, solutions were obtained for values of x_i^o less than 0.1 but only with difficulty and the use of many restarts. In Table 2.1, the number of descent iterations shown is the number of steps taken by the descent algorithm from the initial guess on $\underline{x}$ to the point at which the Gauss-Newton search converges; if there are any restarts, then only the descent path which leads to convergence is considered.

The computational effort n_c is defined by $n_c = n_F + n \times n_D + n(n+1) \times n_{DD}/2$ where n_F, n_D and n_{DD} are the total numbers of function, first-derivative and second-derivative evaluations respectively. These figures exclude any failed descent paths. This definition tacitly assumes that the same computational effort is required to calculate the value of a function as is required for a derivative. Patently this is not so; for example, common expressions can be stored from a function evaluation for use in a subsequent derivative calculation; also some derivatives may be constant or zero. Murray [25, p.70] uses a measure involving processor time; as Himmelblau

Initial Estimate x_i^o	Initial Sum of Squares $F(\underline{x}^o)$	Number of Descent Iterations	Sum of squares on transfer to Gauss-Newton	Number of Gauss-Newton Iterations	Final Sum of Squares	Computational Effort	Number of Restarts
0.1	$1.1_{10}5$	10	$1.9_{10}2$	38	$5.1_{10}-26$	4852	2
0.3	$6.7_{10}4$	8	$1.9_{10}2$	26	$1.8_{10}-14$	4069	0
0.5	$3.5_{10}4$	6	$1.9_{10}2$	20	$9.4_{10}-23$	4222	0
0.7	$1.4_{10}4$	4	$1.9_{10}2$	10	$2.5_{10}-15$	2202	1
0.9	$3.2_{10}3$	6	$6.7_{10}1$	10	$4.6_{10}-26$	3062	0
1.0	$2.1_{10}3$	4	$1.9_{10}2$	23	$7.8_{10}-16$	2153	0
2.0	$1.3_{10}5$	6	$7.3_{10}1$	13	$2.9_{10}-24$	2756	0
3.0	$5.3_{10}5$	5	$1.0_{10}2$	17	$4.0_{10}-24$	2260	0
4.0	$1.8_{10}6$	11	$1.1_{10}2$	120	$1.9_{10}-26$	5848	0
5.0	$1.2_{10}7$	9	$3.8_{10}1$	29	$2.3_{10}-21$	3251	0
6.0	$2.5_{10}8$	9	$6.1_{10}1$	97	$5.1_{10}-16$	4170	0
7.0	$9.8_{10}9$	20	$3.8_{10}1$	92	$2.0_{10}-14$	11014	0
8.0	$7.3_{10}11$	15	$4.7_{10}1$	60	$7.0_{10}-14$	8665	0
9.0	$1.3_{10}14$	16	$6.1_{10}1$	60	$1.2_{10}-22$	6707	0
10.0	$5.6_{10}16$	22	$3.6_{10}1$	79	$6.8_{10}-15$	9221	0

Table 2.1 Results for Problem 1 - True Hessian, Analytic Derivatives

[6] points out this can be suspect since there are many imponderables e.g. compiler efficiency and multiprogramming which can markedly effect such measures. Bard [15] observed that computational effort should also take into account other major calculations such as the solution of linear equations and eigenvalue analysis. Wolfe [46] introduced an index of computational efficiency $\log_e(r_F)/n_c$ where r_F is the ratio of initial to final sum of squares; when the final sum of squares is zero this index is infinite and when there is no reduction the index is zero. Many sum of squares problems do not have a zero minimum; consequently this index could class a search, which did not reach a solution but go some way towards doing so with modest computation, as being more efficient than a search which reached the solution but with a greater amount of computation.

It will be observed that a solution was obtained from all fifteen starting-points and that only in two cases were restarts necessary. Thus it can be justifiably claimed that the results show that the original aim of developing a reliable algorithm with good global convergence properties was achieved. Note that by global convergence is meant the ability to reach a single solution from a wide range of starting points rather than the ability to locate the global minimum of a function with several minima.

2.6.2 A modification which does not need second derivatives

In view of the parallel which was drawn between the Levenberg-Marquardt algorithm and the descent algorithm, it was decided to investigate the effect of using equation (2.6) in place of equation (2.4) when evaluating the descent corrections. This is equivalent to replacing the Hessian G^k in equation (2.4) by its Gauss-Newton approximation $2J^TJ$. The major benefit is that second derivatives are thereby no longer needed.

One point of note arose in connection with the logarithmic transformation of the variables. For the untransformed variables the elements of the approximate Hessian are

$$\frac{\partial^2 F}{\partial x_i \partial x_j} \simeq \sum_{k=1}^{m} 2 \frac{\partial f_k}{\partial x_i} \frac{\partial f_k}{\partial x_j} \tag{2.28}$$

For the transformed variables, the Hessian becomes

$$\frac{\partial^2 F}{\partial x'_i \partial x'_j} \simeq \sum_{k=1}^{m} 2 \frac{\partial f_k}{\partial x'_i} \frac{\partial f_k}{\partial x'_j} = \sum_{k=1}^{m} 2 x_i x_j \frac{\partial f_k}{\partial x_i} \frac{\partial f_k}{\partial x_j} \tag{2.29}$$

However, one could start from the exact relation

$$\frac{\partial^2 F}{\partial x'_i \partial x'_j} = x_i x_j \frac{\partial^2 F}{\partial x_i \partial x_j} + \delta_{ij} x_i \frac{\partial F}{\partial x_i} \tag{2.30}$$

where $\delta_{ij}=0$ if $i \neq j$ and $\delta_{ii}=1$. If the first term on the right-hand side of equation (2.30) is replaced by its approximation as given by equation (2.28) then a different expression for $\partial^2 F/\partial x'_i \partial x'_j$ from that given by equation (2.29) is obtained. Although this alternative form could have some merit, the author used the form of equation (2.29) since this is consistent with the assumptions of the Gauss-Newton approximation.

The results for Problem 1 using the approximate Hessian are shown in Table 2.2. Comparison with the earlier results using the exact Hessian shows that the approximate method results in an overall increase in computational effort and a greater use of the restart facility. However, the modified algorithm is still very reliable and has the advantage of not requiring second derivatives.

Initial Estimate x_i^o	Initial Sum of Squares $F(\underline{x}^o)$	Number of Descent Iterations	Sum of squares on transfer to Gauss-Newton	Number of Gauss-Newton Iterations	Final Sum of Squares	Computational Effort	Number of Restarts
0.1	$1.1_{10}5$	22	$5.8_{10}1$	40	$1.6_{10}-13$	15730	5
0.3	$6.7_{10}4$	4	$1.9_{10}2$	40	$3.3_{10}-16$	3242	5
0.5	$3.5_{10}4$	5	$1.9_{10}2$	17	$4.3_{10}-16$	2637	0
0.7	$1.4_{10}4$	3	$1.9_{10}2$	94	$1.4_{10}-23$	3260	0
0.9	$3.2_{10}3$	3	$1.9_{10}2$	18	$3.8_{10}-15$	1402	2
1.0	$2.1_{10}3$	4	$1.9_{10}2$	15	$3.4_{10}-18$	1594	0
2.0	$1.3_{10}5$	16	$7.9_{10}1$	117	$8.4_{10}-26$	33534	3
3.0	$5.3_{10}5$	12	$4.3_{10}1$	68	$9.2_{10}-15$	4574	0
4.0	$1.8_{10}6$	4	$9.4_{10}1$	13	$1.4_{10}-23$	1242	0
5.0	$1.2_{10}7$	8	$8.7_{10}1$	14	$3.5_{10}-22$	2483	0
6.0	$2.5_{10}8$	9	$4.7_{10}1$	27	$1.6_{10}-19$	2977	0
7.0	$9.8_{10}9$	17	$3.8_{10}1$	151	$2.4_{10}-18$	11223	0
8.0	$7.3_{10}11$	13	$4.0_{10}1$	53	$3.9_{10}-27$	4419	0
9.0	$1.3_{10}14$	14	$7.6_{10}1$	45	$1.7_{10}-26$	6114	0
10.0	$5.6_{10}16$	14	$5.1_{10}1$	158	$2.5_{10}-18$	7225	0

Table 2.2 Results for Problem 1 - Approximate Hessian, Analytic Derivatives

2.6.3 The use of numerical estimates of the derivatives

A number of different formulae could be used to estimate the derivatives numerically; Gill and Murray [31] discuss this subject more fully. In the present program, a forward difference formula was tried as follows. Let it be desired to find an estimate of $\partial h(\underline{x})/\partial x_i$ for the function $h(\underline{x})$ at $\underline{x}=\underline{x}^k$. First $h(\underline{x})$ is evaluated at $\underline{x}^k$, then a displacement vector $\delta\underline{x}$ is defined where $\delta x_i \neq 0$ and $\delta x_j = 0$ $(i \neq j)$ and $h(\underline{x})$ evaluated at $\underline{x}^k + \delta\underline{x}$. The required estimate is then

$$\frac{\partial h}{\partial x_i}(\underline{x}^k) \simeq \frac{h(\underline{x}^k + \delta\underline{x}) - h(\underline{x}^k)}{\delta x_i}$$

To estimate the Jacobian matrix will require n evaluations of $\underline{f}(\underline{x}^k + \delta\underline{x})$, one for each variable x_i, in addition to the evaluation of $\underline{f}(\underline{x}^k)$. This latter quantity will normally be required anyway by the algorithm at the time the Jacobian is required. Curtis, Powell and Reid [69] show how the computation can be reduced when the Jacobian is known to be sparse.

A similar process is used for estimation of the second derivative terms, provided that analytic first derivatives are available (it is not possible to estimate second derivatives accurately from function values only). Note that since the matrix of second derivatives for a given function f is symmetric, the truncation error in the approximation can be reduced by averaging two independent estimates using the formula

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(\underline{x}^k) \simeq \tfrac{1}{2}\left[\frac{\frac{\partial f}{\partial x_i}(\underline{x}^k + \delta\underline{x}^1) - \frac{\partial f}{\partial x_i}(\underline{x}^k)}{\delta x^1{}_j} + \frac{\frac{\partial f}{\partial x_j}(\underline{x}^k + \delta\underline{x}^2) - \frac{\partial f}{\partial x_j}(\underline{x}^k)}{\delta x^2{}_i}\right]$$

where $\delta x^1{}_j$ and $\delta x^2{}_i$ are the only non-zero components of $\delta\underline{x}^1$ and $\delta\underline{x}^2$.

The major difficulty is choosing a suitably small value for the nonzero component of the displacement vectors. Gill and Murray [31] recommend that a suitable choice is $2^{-t/2}$ where t is the number of bits in the mantissa, provided that the components of $\underline{x}$ are of order unity and the function is well-behaved. Intuitively, this strikes a balance between reducing truncation error without increasing roundoff error.

In the two-part program, δx_i was taken as 10^{-7} $(1 + |x_i|)$ following Gill and Murray's reasoning but also allowing for the possibility of large values of $|x_i|$. Two sets of results were obtained. The first is shown in Table 2.3 and corresponds to the use of numerical estimates of the second derivatives and should be compared with Table 2.1. The second set in Table 2.4 was obtained with numerical estimates of the first derivatives and using the approximation of the Hessian in the descent algorithm; it should be compared with Table 2.2. It will be seen that there is little difference in the results obtained using the exact Hessian, with and without numerical second derivatives. However, there are major differences between the two cases using the approximate Hessian. Logically this could be expected. If numeric estimates of the second derivatives are used only the descent algorithm is affected; the effect may be slight if G^k is dominated by the term $2J^TJ$. On the other hand, when the modified descent algorithm is used in conjunction with numeric estimates of the first derivatives, both the descent algorithm and the Gauss-Newton algorithm are affected since they both depend upon the value $2J^TJ$. The effect of errors in the numeric estimates will be more marked if $2J^TJ$ is nearly singular; this explains why the differences between Tables 2.2 and 2.4 are more marked for low starting-values of x_i^o (it was found that $2J^TJ$ is singular for $x_i^o = 1$).

Initial Estimate x_i^o	Initial Sum of Squares $F(\underline{x}^o)$	Number of Descent Iterations	Sum of squares on transfer to Gauss-Newton	Number of Gauss-Newton Iterations	Final Sum of Squares	Computational Effort	Number of Restarts
0.1	$1.1_{10}5$	10	$1.9_{10}2$	47	$1.7_{10}-15$	5014	2
0.3	$6.7_{10}4$	8	$1.9_{10}2$	26	$1.8_{10}-14$	4071	0
0.5	$3.5_{10}4$	6	$1.9_{10}2$	20	$9.2_{10}-23$	4220	0
0.7	$1.4_{10}4$	4	$1.9_{10}2$	10	$2.5_{10}-15$	2203	1
0.9	$3.2_{10}3$	6	$6.7_{10}1$	10	$2.7_{10}-26$	3062	0
1.0	$2.1_{10}3$	4	$1.9_{10}2$	23	$7.8_{10}-16$	2116	0
2.0	$1.3_{10}5$	6	$7.3_{10}1$	13	$3.0_{10}-24$	2756	0
3.0	$5.3_{10}5$	5	$1.0_{10}2$	17	$4.1_{10}-24$	2261	0
4.0	$1.8_{10}6$	11	$1.1_{10}2$	120	$3.0_{10}-27$	5848	0
5.0	$1.2_{10}7$	9	$3.8_{10}1$	29	$2.3_{10}-21$	3252	0
6.0	$2.5_{10}8$	9	$6.1_{10}1$	97	$5.2_{10}-16$	4172	0
7.0	$9.8_{10}9$	20	$3.8_{10}1$	92	$1.8_{10}-14$	11024	0
8.0	$7.3_{10}11$	15	$4.6_{10}1$	60	$7.0_{10}-14$	8665	0
9.0	$1.3_{10}14$	16	$6.1_{10}1$	60	$1.4_{10}-22$	6704	0
10.0	$5.6_{10}16$	22	$3.6_{10}1$	79	$5.5_{10}-18$	9224	0

Table 2.3 Results for Problem 1 - True Hessian, Numeric Second Derivatives

Initial Estimate x_i^o	Initial Sum of Squares $F(\underline{x}^o)$	Number of Descent Iterations	Sum of squares on transfer to Gauss-Newton	Number of Gauss-Newton Iterations	Final Sum of squares	Computational Effort	Number of Restarts
0.1	$1.1_{10}5$	6	$1.9_{10}2$	72	$1.4_{10}-18$	3979	0
0.3	$6.7_{10}4$	4	$1.9_{10}2$	43	$2.2_{10}-14$	2990	0
0.5	$3.5_{10}4$	5	$1.9_{10}2$	17	$4.6_{10}-16$	4759	0
0.7	$1.4_{10}4$	3	$1.9_{10}2$	94	$1.7_{10}-22$	3139	0
0.9	$3.2_{10}3$	3	$1.9_{10}2$	18	$2.5_{10}-17$	1389	2
1.0	$2.1_{10}3$	5	$1.8_{10}2$	17	$4.0_{10}-24$	3765	0
2.0	$1.3_{10}5$	16	$8.0_{10}1$	117	$4.8_{10}-25$	43542	3
3.0	$5.3_{10}5$	13	$3.9_{10}1$	77	$5.3_{10}-15$	4975	0
4.0	$1.8_{10}6$	4	$9.4_{10}1$	13	$4.6_{10}-23$	1242	0
5.0	$1.2_{10}7$	8	$8.7_{10}1$	14	$6.6_{10}-23$	2483	0
6.0	$2.5_{10}8$	9	$4.7_{10}1$	27	$1.3_{10}-19$	2977	0
7.0	$9.8_{10}9$	17	$3.8_{10}1$	151	$3.3_{10}-18$	12783	0
8.0	$7.3_{10}11$	13	$4.0_{10}1$	53	$1.1_{10}-24$	4418	0
9.0	$1.3_{10}14$	14	$7.6_{10}1$	45	$1.1_{10}-27$	9126	0
10.0	$5.6_{10}16$	14	$5.1_{10}1$	158	$2.2_{10}-18$	9028	0

Table 2.4 Results for Problem 1 - Approximate Hessian, Numeric First Derivatives

2.6.4 Some statistics concerning the program

During the early stages of development, various statistics were output by the program to monitor its performance. These statistics were not included in the final FORTRAN version. However, some of the more interesting statistics produced using the ALGOL version with the exact Hessian for Problem 1 will now be given. The values quoted are averaged over the fifteen starting-points.

In the descent algorithm, on average 9-10 minima were found at each application of the multimodal search; the largest and smallest numbers of minima found were 14 and 8 respectively. Of the minima, about 30% were quasi-minima. If the grid search is examined, it will be apparent that there must always be at least one minimum or quasi-minimum for each of the n-1 interior ranges. Also, there must be a minimum in the descent region. Thus there will always be an absolute lower bound of n minima per multimodal search, which agrees with the value of 8 just quoted. Taking account of both the grid search and the quadratic interpolation, approximately 12 evaluations of $F(\lambda)$ were required per minimum found. If a Golden Section search had been used, statistics showed that to get the same accuracy as obtained by the quadratic interpolation a further 4 evaluations of $F(\lambda)$ per minimum would be required. In about 5% of the quadratic interpolations, nonunimodality of $F(\lambda)$ was detected.

In the Gauss-Newton algorithm, on average 25 iterations per search (including failed searches) were performed. The line-search at each iteration required on average 10 evaluations of $F(\alpha)$; of these 3 evaluations were needed to bracket the minimum. This latter figure indicates the suitability of the choice of α_o since at least 2 evaluations of $F(\alpha)$, in addition to the known value at $\alpha=0$, will always be necessary. Roughly one

third of the quadratic interpolations detected nonunimodality in $F(\alpha)$; this high figure could be expected since no special provision for nonunimodality is made in the bracketing process, as is done for the descent algorithm. Of all the Gauss-Newton searches, 7% resulted in convergence; 2% failed because the matrix J^TJ became singular; 1% failed because of failure in the search on $F(\alpha)$; 3% because the upper limit of 200 iterations was reached; 52% because the rate of increase of the maximum magnitude correction increased 10 times; and 35% because this same correction became 100 times greater than its value on entry to the search. These last two figures show the benefits of using predicted failure criteria; of those searches that converged, the largest number of increases in the rate of increase of the MMC was 4 and the greatest increase in its value was 22 times its value on entry; the corresponding figures used of 10 and 100 thus have a good margin for error.

2.6.5 Comparisons with other algorithms

In order to assess the worth of the two-part algorithm in relation to existing algorithms, some comparative tests were carried out against three other algorithms in common use. The chosen algorithms were the methods of Powell [44], Fletcher [41] and Gill, Murray and Picken [27], as implemented in the widely-distributed NAG [70] Library. The first two methods are hybrid algorithms specifically for sum of squares objective functions; Powell's method uses function values only and computes numerical estimates of the first derivatives while Fletcher's method requires that first derivatives be supplied. The third method, that of Gill *et al*, is an implementation of Newton's method for the solution of a general objective function; it requires both the first and second derivatives of the function.

To make the comparisons, Problem 1 was used together with an additional eight test problems, given as Problems 2-9 in the Appendix. These additional problems were cited by Meyer and Roth [42] as being good tests of the performance of a sum of squares minimization algorithm. Nash [43] has recently published results obtained with Meyer and Roth's problems using his own implementation of Marquardt's method.

Before progressing to describe the results of the comparisons it is necessary to say a little about the way in which the test runs were performed. In the case of Problem 1, a logarithmic transformation of the variables was used, as before, to ensure that the non-negativity constraints were satisfied; for the remaining eight problems, which were not subject to such constraints, a linear transformation to variables $x'_i = x_i/s_i$ was used. The value of s_i for each variable is set at the start so that $x'_i=1$ when $x_i=x_i^o$; this requires that $s_i=x_i^o$. Provision is made by the transformation for the case of $x_i^o=1$; in this event, s_i is set to 1 and the initial value x'_i to 0. Both transformations ensure that the same step limit p can be used for all the variables in the two-part algorithm. The maximum change of $\pm p$ in any transformed variable x'_i at one iteration of the optimization will produce a corresponding change in value x_i of the problem variable of $(\exp(\pm p)-1)\, x_i$ or $\pm p\, s_i$, for the logarithmic and linear transformations respectively. The choice of p=0.5 will thus be a reasonable one. A similar argument applies to the methods of Powell and Gill *et al* which both require that an upper limit on the length of any correction $\Delta\underline{x}$ be specified. The value of 0.5 was used for both these algorithms for the tests; in the absence of the transformations, different step limits would have been required for each problem according to the magnitude of the problem variables.

Powell's method requires the specification of a step size for use in

the numerical estimation of first derivatives. The value of 10^{-7} was used, in line with the comments in section 2.3.3. Fletcher's method uses the Marquardt-type correction vector $\underline{\Delta x} = -(J^TJ + \lambda W)^{-1}J^T\underline{f}$ where W is a specified matrix; two separate tests were made one with W=I, the unit diagonal matrix, and the other with $W=J^TJ$. No changes were made to the two-part algorithm other than to relax the convergence criterion for Problems 2-9 by increasing ε from its default of 10^{-8} to the less stringent value of 10^{-5}.

The performance of all four algorithms on Problem 1 is summarised in Table 2.5. Of the two entries shown for the two-part algorithm, the first refers to the use of the true Hessian and the second to the use of the approximation $2J^TJ$ in place of the Hessian (they therefore correspond to Tables 2.1 and 2.2 respectively). For Fletcher's algorithm, the first entry is for W=I and the second for $W=J^TJ$. Table 2.5 is intended to compare the abilities of the four algorithms to solve a difficult problem; in all but one instance, the NAG algorithms fail to do so. The manner in which each test ended is indicated by the symbols G for the global minimum (which is zero and corresponds to the solution), L for a local minimum and D for a dead-end in which the variables are wildly-removed from the solution and no further progress can be made. In the case of $x_i^o=3$ when Powell's algorithm reached the global solution, the required computational effort n_c was 192. To check on the effect of the step limit on the performance of the methods of Powell and Gill *et al*, a further survey was made using values for this step limit of 0.1, 0.2, 0.5, 1.0, 2.0, 5.0 and 10.0. The results indicated that no significant improvement in reliability from that shown could have been achieved by changing the step limit for these two methods. It should be noted that all algorithms could easily find the solution to Problem 1 containing negative x_i values when

Starting Value x_i^o	Two-part algorithm	Powell algorithm	Fletcher algorithm	Gill, Murray and Picken algorithm
0.1	G*/G*	L	D/D	L
0.3	G/G*	L	D/D	L
0.5	G/G	L	D/D	D
0.7	G*/G	L	D/D	D
0.9	G/G*	L	D/D	D
1.0	G/G	L	D/D	D
2.0	G/G*	L	D/D	D
3.0	G/G	G	D/D	D
4.0	G/G	L	D/D	D
5.0	G/G	L	D/D	D
6.0	G/G	L	D/D	D
7.0	G/G	L	D/D	D
8.0	G/G	L	D/D	D
9.0	G/G	L	D/D	D
10.0	G/G	L	D/D	D

G	Global Minimum Located (solution)
L	Local Minimum Located
D	Dead-end from which no further progress can be made
*	Denotes use of restart facility

Table 2.5 Comparisons of the four algorithms using Problem 1

the logarithmic transformation was removed. The results obtained with Problems 2-9 are shown in Table 2.6. The figures given are the values of computational effort n_c defined in Section 2.6.1. The pairs of figures, given for both the two-part algorithm and the Fletcher algorithm, have the same significance as for Table 2.5. Problems 2-9 are much less difficult than Problem 1; however note that Powell's method failed on Problem 7 and the method of Gill *et al* failed on Problems 7 and 8.

It will be apparent from these results that the two-part algorithm is superior in reliability to the other three algorithms. Even if the restart feature were to be removed, since it could be argued that this gives the two-part algorithm an unfair advantage, then the success rate would still be 80% on Problem 1 as compared with, at best, 7% for the other three algorithms on the same problem. This reliability is not achieved at too great a cost in efficiency since, for Problems 2-9 the two-part algorithm requires a similar amount of computational effort to the other algorithms. It should be noted that, for Problems 2, 3, 4, 5 and 9 and the Gauss-Newton search in the two-part algorithm converged from the initial starting point. Also, similar results were obtained when numerical approximations were used for the derivatives by the two-part algorithm.

Of the other three algorithms, Powell's method performed best overall in terms of reliability and efficiency. It consistently located the local minimum of Problem 1 apart from the one instance when it found the global minimum and so it could justifiably be claimed that it always achieved its aim of minimizing the sum of squares. Fletcher's method is very efficient when it is successful but it performed poorly on Problem 1. The comparison is slightly unfair to the method of Gill *et al* in that it is designed for a general objective function; on the other hand it does have the benefit of an exact Hessian matrix of the objective function.

Problem	Two-part algorithm	Powell algorithm	Fletcher algorithm	Gill, Murray and Picken algorithm
2	59/59	31	30/49	216
3	162/162	29	46/46	147
4	166/166	25	49/50	147
5	88/88	30	25/217	110
6	405/294	37	29/247	220
7	250/244	failed	276/269	failed
8	233/227	282	267/255	failed
9	76/76	65	243/569	2168

N.B. The figures given for each algorithm are of the computational effort n_c

Table 2.6 Comparisons of the four algorithms using Problems 2-9

3. A HYBRID ALGORITHM FOR SOLVING SETS OF NONLINEAR EQUATIONS

This chapter describes work carried out by the author with the aim of developing a new hybrid algorithm for solving sets of nonlinear equations involving n variables. As shown in the previous chapter, considerable success was achieved by using a two-part algorithm to solve such problems. The major source of inefficiency in the two-part algorithm is wasted computational effort expended when the second part is entered prematurely. To avoid this, the author sought a new method which would, in a single algorithm, combine the robustness of a descent algorithm with the fast rate of convergence of Newton's method for solving nonlinear equations. The method which was developed was later extended to handle overdetermined systems of equations, for which case the optimal solution defined by the new method is that which minimizes the sum of the absolute values of the residuals, rather than the sum of squares.

3.1 Newton's method for solving nonlinear equations

In the early stages of development of the two-part algorithm, the author was concerned with solving sets of nonlinear equations. Thus the Newton (or Newton-Raphson) corrections were originally used in the second part of the algorithm. At a later stage the Newton corrections were replaced by the more general Gauss-Newton corrections; this final form

of the two-part algorithm is the one described in the previous chapter. However, for this historical reason, the original ideas of the author for the hybrid algorithm developed from a consideration of the causes of the breakdown of the Newton method, which will now be described.

Given the set of n nonlinear equations with residuals $f_i(\underline{x})$ (i=1, ... n) for the values of the variables $\underline{x}^T = (x_1, \ldots x_n)$, it is required to find the value $\underline{x}^*$ at which $\underline{f}(x^*)=0$. Suppose that at iteration k the current estimate of $\underline{x}^*$ is $\underline{x}^k$. The functions $\underline{f}(\underline{x})$ at the point $\underline{x}^k + \Delta\underline{x}$ are approximately given by a Taylor series expansion as

$$\underline{f}(\underline{x}^k + \Delta\underline{x}) \simeq \underline{f}(\underline{x}^k) + J\,\Delta\underline{x} \qquad (3.1)$$

where J is the Jacobian matrix $J_{ij} = \partial f_i/\partial x_j$ evaluated at $\underline{x}^k$. This expansion ignores terms involving second and higher-order derivatives of $\underline{f}(\underline{x})$. On the basis of this linear approximation, the Newton corrections $\Delta\underline{x}$ are obtained by setting the left-hand side of equation (3.1) to zero to give

$$\Delta\underline{x} = -J^{-1}\, f(\underline{x}^k) \qquad (3.2)$$

Since this prediction is not, in general, exact it is normally necessary to set $\underline{x}^{k+1} = \underline{x}^k + \Delta\underline{x}$ and repeat the process until convergence is achieved; the final rate of convergence can be shown to be quadratic.

In practice, Newton's method often diverges. To cope with this, one can set

$$\underline{x}^{k+1} = \underline{x}^k + \alpha\Delta\underline{x} \qquad (3.3)$$

and choose α such that a "sufficient" reduction is obtained in the objective function $F(\underline{x}) = \sum_{i=1}^{n} f_i^2(\underline{x})$ from one iteration to the next.

It will usually be possible to find an α such that $F(\underline{x}^{k+1}) < F(\underline{x}^{k})$ since the correction vector is downhill. This follows from the observation that $dF(\underline{x}^{k})/d\alpha = \underline{g}^{T}\Delta\underline{x}$ where $\underline{g}$ is the gradient vector of $F(\underline{x})$ at $\underline{x}^{k}$. Since $\underline{g}=2J^{T}\underline{f}$ and $\Delta\underline{x}=-J^{-1}\underline{f}$, direct substitution of these values leads to $dF(\underline{x}^{k})/d\alpha = -2\underline{f}^{T}\underline{f}$, demonstrating that $\Delta\underline{x}$ is downhill. Various methods e.g. [37] have been proposed for the choice of α. Although the modification prevents divergence to extreme values of $\underline{x}$, the method can still fail due to the Jacobian J becoming singular, when the corrections given by equation (3.2) will be infinite. Sometimes the method will fail because a value of α cannot be found which gives $F(\underline{x}^{k+1}) < F(\underline{x}^{k})$. Such a breakdown is a symptom of the Jacobian tending to singularity. The effect of roundoff error in the calculations determines which of the two forms of breakdown occurs first. This topic was discussed in the previous chapter in connection with the Gauss-Newton search.

Some means of circumventing the difficulties caused by a singular Jacobian was sought so that continued reduction in $F(\underline{x})$ could be ensured, thus producing a robust algorithm. Consider the situations in which the Jacobian becomes singular. At a stationary point of $F(\underline{x})$ the gradient vector $\underline{g}(\underline{x})$ will be zero. Since $\underline{g}(\underline{x}) = J^{T}\underline{f}(\underline{x})$ it follows that if $F(\underline{x}) > 0$ at the stationary point, then at least one value $f_{i}(\underline{x})$ will be nonzero and consequently J^{T} (and hence J) must be singular. If a local minimum has been reached, it will not be possible to generate a downhill direction; this should be possible however, if the stationary point is a saddle-point. When $F(\underline{x}) = 0$, the stationary point corresponding to the solution, no similar inferences on the Jacobian can be drawn. Also it will be possible for the Jacobian to become singular away from a stationary point. In view of these observations, the more limited goal was adopted of developing a Newton-type algorithm which would locate a

minimum of $F(\underline{x})$ (not necessarily the solution of $f(\underline{x})=0$) and which would not fail if the Jacobian becomes singular.

3.2 A modification of Newton's method

At any iteration k of the Newton method, provided that $\underline{x}^k$ is not a stationary point of $F(\underline{x})$, there will be an infinite number of possible downhill directions. The Newton corrections $\Delta\underline{x}$ specify that downhill direction along which, on the basis of a linear approximation, the residuals $\underline{f}(\underline{x})$ will decrease in the same proportion. Consider the modified Newton corrections $\alpha\Delta\underline{x}$. A Taylor-series expansion about $\underline{f}(\underline{x}^k)$ gives

$$\underline{f}(\underline{x}^k + \alpha\Delta\underline{x}) \simeq \underline{f}(\underline{x}^k) + \alpha J\Delta\underline{x} \qquad (3.4)$$

Substitution of the Newton corrections $-J^{-1}f(\underline{x})$ for $\Delta\underline{x}$ on the right-hand side of equation (3.4) gives

$$\underline{f}(\underline{x}^k + \alpha\Delta\underline{x}) \simeq (1-\alpha)\ \underline{f}(\underline{x}^k) \qquad (3.5)$$

The parameter α defines a point on the line $\underline{x}^k + \alpha\Delta\underline{x}$. From equation (3.5) for each residual f_i we have

$$f_i(\underline{x}^k + \alpha\Delta\underline{x})/f_i(\underline{x}^k) \simeq 1-\alpha \qquad (3.6)$$

Thus, on the basis of the linear approximation, each residual is reduced by the proportional amount $1-\alpha$; for $\alpha=0$, corresponding to no departure from $\underline{x}^k$, there is no reduction in the residuals; for $\alpha=1$, corresponding to the full Newton corrections, the residuals will all be zero.

Even if the Jacobian becomes singular, a downhill direction will still exist provided that a local minimum has not been reached. A similar analysis will now be used to that of section 2.2.2 in the derivation of equation (2.17). Consider the matrix J^TJ; since it is real and symmetric, it will have n real eigenvalues ϕ_i and the corresponding n eigenvectors $\underline{v}_i$ will form an orthonormal set. The Newton corrections can then be expressed as

$$\Delta\underline{x} = \sum_{i=1}^{n} \beta_i \underline{v}_i \qquad (3.7)$$

where the scalar values β_i (i=1, .. n) are to be determined.

Noting that $J\Delta\underline{x} = -\underline{f}$ by definition of the Newton corrections, it follows that $J^TJ\ \Delta\underline{x} = -J^T\underline{f}$. If we replace $\Delta\underline{x}$ by the expression given by equation (3.7) we obtain

$$J^TJ \sum_{i=1}^{n} \beta_i \underline{v}_i = -J^T\underline{f} \qquad (3.8)$$

Since by definition $J^TJ\ \underline{v}_i = \phi_i \underline{v}_i$, equation (3.8) simplifies to

$$\sum_{i=1}^{n} \beta_i \phi_i \underline{v}_i = -J^T\underline{f} \qquad (3.9)$$

If both sides of equation (3.9) are pre-multiplied by $\underline{v}_i^T$, the orthonormality of the eigenvectors gives

$$\beta_i \phi_i = -\underline{v}_i^T J^T \underline{f} \qquad (3.10)$$

Substitution of the values of β_i given by equation (3.10) into equation (3.7) gives finally

$$\Delta\underline{x} = -\sum_{i=1}^{n} \underline{v}_i^T (J^T\underline{f})\ \underline{v}_i / \phi_i \qquad (3.11)$$

When any value ϕ_i tends to zero in equation (3.11), the corrections $\Delta\underline{x}$ will be dominated by the i^{th} term of the summation. If $\phi_i=0$, the corrections are infinite; however their direction is known to be $\underline{v}_i^T J^T \underline{f}\, \underline{v}_i$. This direction will be downhill but in practice is not likely to be a worthwhile search direction since the linear approximation used predicts that an infinite displacement is required to reduce the residuals to zero. Greenstadt [24] considered Newton's method as it applies to a general objective function; in this case the corrections are $\Delta\underline{x} = -G^{-1}\underline{g}$, where G and $\underline{g}$ are respectively the Hessian and gradient of the function. He used a similar expression to equation (3.11) to evaluate $\Delta\underline{x}$ but, before doing so, set $\phi_i = \max(|\phi_i|,\varepsilon)$ for i=1, .. n. By setting ε to a suitable small value, he ensured that the corrections were always downhill.

The author took a different approach and investigated the possibility of relaxing, in some way to be defined, the criterion used to define the Newton corrections, with the aim of avoiding the problem of J becoming singular. It was shown by equation (3.6) that, on the basis of a linear approximation, the residuals decrease in the same proportion $1-\alpha$ along the path $\underline{x}^k + \alpha\Delta\underline{x}$ where $\Delta\underline{x}$ equals the Newton corrections $-J^{-1}\underline{f}$. The predicted decrease in the sum of squares will be $F(\underline{x}^k+\alpha\Delta\underline{x})\simeq(1-\alpha)^2F(\underline{x}^k)$; however the true sum of squares will depart from this prediction, although in a finite range $0 \leqslant \alpha \leqslant \alpha_m$ the true sum of squares will decrease as α is increased.

The relaxed form of the Newton corrections will be denoted by $\Delta\underline{x}(\alpha)$; when α is varied from 0 up to 1 the predicted residuals will decrease, in some manner, along the path $\underline{x}^k+ \Delta\underline{x}(\alpha)$. The requirement that all the predicted residuals decrease by the same proportional amount $1-\alpha$ was removed; if it would be beneficial, a residual would even be allowed to increase along part of the path. As will become apparent, the chosen

form for $\Delta\underline{x}(\alpha)$ would involve nonlinear algebra if the sum of squares $F(\underline{x})$ was used to measure the proximity of the residuals to the solution. Thus the measure $S(\underline{x}) = \sum_{i=1}^{n} |f_i(\underline{x})|$ was adopted; as is the case with $F(\underline{x})$, the global minimum of $S(\underline{x})$ will be zero if a solution to $f(\underline{x}) = 0$ exists and $S(\underline{x}) \geq 0$ for all $\underline{x}$. The relaxed corrections were chosen to be such that $S(\underline{x}^k + \Delta\underline{x}(\alpha)) = (1-\alpha)\, S(\underline{x}^k)$, on the basis of a linear approximation to $f(\underline{x})$; this gives

$$\sum_{i=1}^{n} \left| f_i(\underline{x}^k) + \sum_{j=1}^{n} J_{ij}\, \Delta x_j(\alpha) \right| = (1-\alpha) \sum_{i=1}^{n} |f_i(\underline{x}^k)| \qquad (3.12)$$

The desired condition that $\Delta\underline{x}(\alpha) = 0$ at $\alpha=0$ is satisfied by equation (3.12). At $\alpha=1$, the only possible value for $\Delta\underline{x}(\alpha)$ will be the Newton corrections. Note that the modified Newton corrections $-\alpha J^{-1}\underline{f}$ would satisfy equation (3.12) over the complete range $0 \leq \alpha \leq 1$. There are an infinite number of possible solutions of equation (3.12); the relaxed Newton corrections were taken as that solution $\Delta\underline{x}(\alpha)$ which minimizes $\sum_{i=1}^{n} |\Delta x_i(\alpha)|$. The reasoning for this is that, for a given α, the smaller the size of the corrections the smaller will be the error introduced by the linear approximation.

At this stage, the effect of using a step limit p_i on individual corrections was incorporated. In Chapter 2 it was described how such a limit often improved the global convergence of the descent and Gauss-Newton algorithms. For these earlier algorithms, the step limit was applied posthumously. In other words the correction vector $\Delta\underline{x}$ was first calculated, without taking the limit into consideration and only afterwards was $\Delta\underline{x}$ reduced, if necessary, so as not to exceed the limit. In the present case, posthumous application of the limit to the calculated $\Delta\underline{x}(\alpha)$ could result in a smaller predicted reduction in $S(\underline{x})$ than could

have been obtained if the limit was taken into account by the process used to evaluate $\Delta\underline{x}(\alpha)$. With the step limit incorporated, the corrections $\Delta\underline{x}(\alpha)$ are obtained as the solution $\Delta\underline{x}$ of the problem

$$\text{P1:} \qquad \text{Minimize} \sum_{i=1}^{n} |\Delta x_i|$$

$$\text{Subject to} \sum_{i=1}^{n} \left|f_i + \sum_{j=1}^{n} J_{ij}\Delta x_j\right| = (1-\alpha) \sum_{i=1}^{n} |f_i|$$

$$|\Delta x_i| \leq p_i \quad (i=1, \ldots n)$$

The residuals f_i and the Jacobian elements J_{ij} are those applying at $\underline{x}^k$. No saving in computational labour would be achieved by introducing the restriction that the value of p_i must be the same for each correction; therefore the facility to use different values of p_i was retained. Solutions of problem P1 will exist in the range $0 \leq \alpha \leq \alpha_{max}$ where $\alpha_{max} \leq 1$; solutions with $\alpha < 0$ also exist but are of no interest since they correspond to an increase in the residuals.

Early trials with an algorithm using the corrections $\Delta\underline{x}(\alpha)$ and choosing α at each iteration so as to minimize $S(\underline{x}^k + \Delta\underline{x}(\alpha))$ gave encouraging results. Consideration was therefore given to other ways in which the corrections might be defined. This led to the alternative corrections $\Delta\underline{x}(\beta)$ which are defined as the solution for given β of the problem

$$\text{P2:} \qquad \text{Minimize} \sum_{i=1}^{n} \left|f_i + \sum_{j=1}^{n} J_{ij}\Delta x_j\right|$$

$$\text{Subject to} \sum_{i=1}^{n} |\Delta x_i| = \beta$$

$$|\Delta x_i| \leq p_i \quad (i=1, \ldots n)$$

It will be seen that the point corresponding to β on the path $\underline{x}^k + \Delta\underline{x}(\beta)$ is such that β specifies the displacement of the point from $\underline{x}^k$ as measured by the sum of the magnitudes of the corrections. The step limits p_i set an upper limit $\beta_{max} = \sum_{i=1}^{n} p_i$ such that no solution to problem P2 can exist for $\beta > \beta_{max}$. Within the range $0 \leq \beta \leq \beta_{max}$ the corrections $\Delta\underline{x}$ are specified uniquely by the requirement that they minimize $S(\underline{x}^k + \Delta\underline{x})$ as represented by the linear approximation used in the objective function of problem P2.

This new formulation when tried was immediately found to give corrections equivalent, in the following sense, to those obtained using problem P1. Suppose that the optimal solution to problem P1 for $\alpha = \alpha_o$ is given by the vector $\Delta\underline{x}$ and the objective function $\sum_{i=1}^{n} |\Delta x_i|$ equals β_o. Then the same vector $\Delta\underline{x}$ is the optimal solution of problem P2 when $\beta = \beta_o$ and the objective function $\sum_{i=1}^{n} |f_i + \sum_{j=1}^{n} J_{ij} \Delta x_j|$ will have the value $(1-\alpha_o) \sum_{i=1}^{n} |f_i|$.

With hindsight, such an equivalence might have been intuitively deduced by comparison of both problems; a formal proof is given later in Section 3.4. It will also be shown that the converse does not apply in that the existence of a solution to problem P2 does not necessarily imply the existence of a solution to problem P1. Because of the greater scope of the corrections $\Delta\underline{x}(\beta)$ they were adopted in place of $\Delta\underline{x}(\alpha)$ and used as the basis of a hybrid algorithm.

3.3 Evaluation of the corrections

The corrections $\Delta\underline{x}(\beta)$ corresponding to any given value of β are obtained by solving the minimization problem P2 which, as will be shown, can be formulated in the form of the linear programming problem. Standard methods, mainly based on the Simplex method of Dantzig [5], exist for the solution of the linear programming problem. At each iteration, it will be necessary to solve problem P2 for one or more values of β, depending upon the manner of choosing the corrections to be used to set $\underline{x}^{k+1} = \underline{x}^{k} + \Delta\underline{x}(\beta)$. Rather than solve P2 afresh for each value, it is more efficient to use the techniques of parametric linear programming. Also, advantage can be taken of the known features of P2 to make improvements on the standard method for solving parametric problems. First a brief outline must be given of linear programming techniques before an explanation can be given of the way in which these techniques are applied to the solution of problem P2.

3.3.1 Linear programming techniques

The treatment given here is based on that given by Beale [71]; however it should be noted that Beale considered the maximization, rather than minimization, of the objective function. A more rigorous theoretical treatment is given by Gass [72]. The linear programming problem can be written as

$$\text{P3} \qquad \text{Minimize} \quad \sum_{i=1}^{n} C_i x_i$$

$$\text{Subject to} \quad \sum_{j=1}^{n} A_{ij} x_j = B_i \qquad (i=1, \ldots m)$$

$$x_i \geq 0 \qquad (i=1, \ldots n)$$

The objective function is a linear function of n independent variables x_i; a constant term C_o is omitted from the objective function since the solution variables $\underline{x}$ will be the same whether or not a constant term is present. The variables are subject to m equality constraints; in addition, as with all linear programming problems, the variables are constrainted to be non-negative. The coefficients A_{ij}, B_i and C_i are supplied constants for any given problem.

A feasible solution to P3 is a value of $\underline{x}$ which satisfies both the equality and non-negativity constraints. Usually a feasible solution can exist only when $m \leqslant n$; when $m = n$ there will be at most one feasible solution, unless there is degeneracy; when $m < n$ there will be an infinite number of feasible solutions. Linear programming theory proves that the minimum feasible solution, which will be the required solution of P3, must be a basic feasible solution which is defined as a feasible solution in which at most m values of x_i are greater than zero. The values of any chosen set of m basic variables are given uniquely by solving the constraint equations, with the remaining m - n nonbasic variables each set to zero. The total number of basic feasible solutions equals the number of ways in which m basic variables can be chosen from a population of n variables, which equals $n!/((n-m)!\ m!)$. For many problems, it will not be computationally-feasible to enumerate all the basic feasible solutions in order to find that which gives the minimum value for the objective function; the computation can be dramatically reduced by using the Simplex algorithm as follows.

The algorithm starts by finding a basic feasible solution. Then an iterative procedure is performed in which, at each iteration, one of the nonbasic variables is exchanged with a basic variable, the choice of these

variables being made so as to give a reduction in the objective function When no further reduction is possible, the minimum solution will have been attained. In addition to reducing the number of basic feasible solutions which have to be considered, the calculations can be arranged so that it is not necessary to completely solve the constraint equations for each new basic feasible solution. Instead a tableau of values is maintained containing information on the current solution, and this tableau is updated at each iteration by a set of row operations. Various forms of the Simplex method exist according to the organization of the tableau and the calculations.

The parametric form of the linear programming problem which will be used in the present case is given by

P4: Minimize $\sum_{i=1}^{n} C_i x_i$

$$\text{Subject to } \sum_{j=1}^{n} A_{ij} x_j = B_i + \theta B'_i \quad (i=1, .. m)$$

$$x_i \geq 0 \quad (i=1, .. n)$$

It is assumed that solutions are required for values of the parameter θ in the range $0 \leq \theta \leq \theta_{max}$; the upper limit θ_{max} may be infinite. These solutions are found as follows.

First, the minimum feasible solution to P4 is found for $\theta=0$; this can be done in the manner described previously for the non-parametric problem. Denote the basic variables by $\underline{X}^T = (X_1, \ldots X_m)$ and the nonbasic variables by $\underline{Y}^T = (Y_1, \ldots Y_{n-m})$. Using this notation we can express the equality constraints as

$$X_i = \sum_{j=1}^{n-m} a_{ij} (-Y_j) + b_i + \theta b'_i \quad (3.13)$$

$$(i=1, \ldots m)$$

and the objective function by z which, in terms of the nonbasic variables, is given as

$$z = \sum_{i=1}^{n-m} c_i(-Y_i) + c_o + \theta c'_o \qquad (3.14)$$

The values of coefficients appearing in the last two equations are obtained from the basic feasible solution at $\theta=0$ and are stored in the tableau. Note that additional information has to be stored in order to keep track of which variables are represented by $\underline{X}$ and $\underline{Y}$. Following the convention adopted by Beale, a minus sign is used in conjunction with $\underline{Y}$ in equations (3.13) and (3.14).

Since the solution at $\theta=0$ is optimal, then we must have $c_i \leq 0$ for $i=1, .. n-m$; otherwise if any c_i were positive, the objective function could be further reduced by introducing the corresponding variable into the basis. Furthermore, as θ is increased from zero the solution will remain optimal as long as the values for $\underline{X}$ remain feasible. However, if one or more values b'_i are negative, then the corresponding variables will eventually become negative and the solution will no longer be feasible. An upper limit is set at $\theta=\theta_1$ where

$$\theta_1 = \min_{i,\ b'_i < 0} \left(-\frac{b_i}{b'_i}\right) \qquad (3.15)$$

If no value b'_i is negative then the current solution given by equations (3.13) and (3.14) will remain feasible and optimal for all $\theta > 0$.

Assume that a finite limit $\theta_1 = -b_p/b'_p$ exists; the basic variable X_p is thus zero at $\theta = \theta_1$ and will need to be replaced by a non-basic variable in order that the solution will continue to be feasible for $\theta > \theta_1$. The rule for choosing a nonbasic variable Y_q to enter the basis

such that the solution remains optimal for $\theta > \theta_1$ is

$$\frac{c_q}{a_{pq}} = \min_{i,\ a_{pi} < 0} \left(\frac{c_i}{a_{pi}}\right) \tag{3.16}$$

If there is no i such that $a_{pi} < 0$ then no feasible solutions exist for $\theta > \theta_1$; otherwise the variables X_p and Y_q are exchanged and the tableau updated by a set of row operations which use the elements a_{pq} as a pivot. These operations can be summarised in algorithmic form as:

Step 1 Set $r = 1/a_{pq}$

Step 2 For row p of the tableau

Set $a_{pj} = r a_{pj} \qquad j \neq q$

$a_{pq} = r$

Step 3 For each of the remaining rows

Set $a_{ij} = a_{ij} - a_{iq}\, a_{pj} \qquad j \neq q$

$a_{iq} = -r\, a_{iq}$

The coefficients b_i and b'_i are stored in the two last columns of the tableau and the coefficients c_i, c_o and c'_o are stored in the last row; the row operations are carried out on these elements of the tableau as well.

Having updated the tableau in this way, the upper limit θ_2 of the second range $\theta_1 \leq \theta \leq \theta_2$ is found and the process repeated until the complete set of solutions in the range $0 \leq \theta \leq \theta_{max}$ has been obtained.

3.3.2 Parametric solution of problem P2

Before problem P2 can be expressed in the form of the parametric linear programming problem P4, it is first necessary to remove the modulus terms. One widely-used method [73] is to replace any modulus expression such as $|y|$ by two new variables y^+ and y^-. As long as it is ensured that y^+ and y^- can never be in the basis simultaneously, then we can write $y = y^+ - y^-$ and $|y| = y^+ + y^-$. For problem P2 new variables were introduced in this way such that

$$|\Delta x_i| = \Delta x_i^+ + \Delta x_i^- \qquad (i=1, .. n) \qquad (3.17a)$$

$$\left|f_i + \sum_{j=1}^{n} J_{ij}\, \Delta x_i\right| = f_i^+ + f_i^- \qquad (i=1, .. n) \qquad (3.17b)$$

Problem P2 can then be expressed as the following parametric linear programming problem

P5: Minimize $\sum_{i=1}^{n} f_i^+ + f_i^-$

Subject to $\sum_{i=1}^{n} \Delta x_i^+ + \Delta x_i^- = \beta$

$$\Delta x_i^+ + \Delta x_i^- + \Delta x_i^s = p_i \qquad (i=1, .. n)$$

$$f_i^+ - f_i^- - \sum_{j=1}^{n} J_{ij}(\Delta x_j^+ - \Delta x_j^-) = f_i^o \qquad (i=1, .. n)$$

$$\Delta x_i^+, \Delta x_i^-, \Delta x_i^s, f_i^+, f_i^- \geqslant 0 \qquad (i=1, .. n)$$

The inequality constraints $|\Delta x_i| \leqslant p_i$ have been transformed to equality constraints by the well-known technique of introducing slack variables. Also, further equality constraints have been added to define the relationship between f^+ and f^-. It will be seen that the problem has 5n variables and 2n+1 constraints and it is parametric with respect to β.

Note that the value f_i^o has been used in place of f_i where $f_i^o = |f_i|$. If at $\underline{x}^k$ it happens that any residual f_i is negative, then the corresponding row i of the Jacobian is multiplied by -1 to be consistent with this definition of f_i^o. The corrections $\Delta\underline{x}(\beta)$ are not affected since this is equivalent to replacing equation $f_i(\underline{x})=0$ by $-f_i(\underline{x})=0$ in the set of equations being solved.

To find the minimum basic feasible solution of P5 at $\beta=0$, it would normally be necessary to solve the corresponding non-parametric linear programming problem obtained by setting $\beta=0$. However, the solution can be found very much more quickly as follows. Since for $\beta=0$ it is known that $\Delta\underline{x}=0$, it follows from the constraint equations that the variables Δx_i^s, f_i^+ (i=1, .. n) must be included in the basis. The tableau shown in Figure 3.1 can then be set up. The coefficients refer to the quantities shown at the head of each column and the basic variables are shown down the left hand side. The first 2n+1 rows correspond to the equality constraints in the form given by equation (3.13); the last row corresponds to the expression for the objective function given by equation (3.14). The equality constraints appear in the same order in the tableau as they do in problem P5. Note that the minus sign associated with the nonbasic variables in equations (3.13) and (3.14) is not shown.

It will be seen from the first row of the tableau that one more variable must be chosen to complete the basis. Since $\Delta\underline{x} \neq 0$ for $\beta > 0$, it follows that the chosen variable must be one of the 2n variables Δx_i^+ and Δx_i^-. The chosen variable must be that which gives the greatest rate of reduction in z as β increases; it will therefore correspond to the maximum coefficient c_{imax} of the coefficients c_i stored in the bottom row of the tableau. Note that $c_{i+n}=-c_i$ (i=1, .. n) and that therefore

NONBASIC VARIABLES

BASIC VARIABLES	Δx_1^+	Δx_2^+	.	Δx_n^+	Δx_1^-	Δx_2^-	.	Δx_n^-	f_1^-	f_2^-	.	f_n^-	1	β
-	1	1	.	1	1	1	.	1	0	0	.	0	0	1
Δx_1^s	1	0	.	0	1	0	.	0	0	0	.	0	p_1	0
Δx_2^s	0	1	.	0	0	1	.	0	0	0	.	0	p_2	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Δx_n^s	0	0	.	1	0	0	.	1	0	0	.	0	p_n	0
f_1^+	$-J_{11}$	$-J_{12}$	.	$-J_{1n}$	J_{11}	J_{12}	.	J_{1n}	-1	0	.	0	f_1^o	0
f_2^+	$-J_{21}$	$-J_{22}$	.	$-J_{2n}$	J_{21}	J_{22}	.	J_{2n}	0	-1	.	0	f_2^o	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
f_n^+	$-J_{n1}$	$-J_{n2}$	.	$-J_{nn}$	J_{n1}	J_{n2}	.	J_{nn}	0	0	.	-1	f_n^o	0
OBJECTIVE FUNCTION z	$-\sum_i J_{i1}$	$-\sum_i J_{i2}$	.	$-\sum_i J_{in}$	$\sum_i J_{i1}$	$\sum_i J_{i2}$	.	$\sum_i J_{in}$	-2	-2	.	-2	$\sum_i f_i^o$	0

Figure 3.1 Tableau for optimum solution at $\beta=0$

$c_{imax} \geq 0$. If $c_{imax} = 0$ then a stationary value of $S(\underline{x})$ has been reached at $\underline{x}^k$; this situation will be discussed further in section 3.6.2. The usual situation will be that $c_{imax} > 0$; in this case, if $imax \leq n$ then the variable Δx^+_{imax} is chosen to enter the basis and if $imax > n$ then variable Δx^-_{imax-n} is chosen. Note that $imax \neq 2n$ since the remaining coefficients, corresponding to the variables f_i^- are all negative.

Once the last basic variable has been chosen, it is entered into the basis and the tableau adjusted using a similar process to that already described. Lastly, the tableau is contracted in order to remove the column previously occupied by the variable which entered the basis. The tableau of 2n+2 rows and 2n+1 columns will then be obtained, corresponding to the optimal solution of problem P5 at $\beta=0$.

The method described in the previous section can then be applied to generate the complete set of optimal solutions in the range $0 \leq \beta \leq \beta_{max}$. The corrections $\Delta\underline{x}(\beta)$ are derived using the values of $\Delta\underline{x}^+$ and $\Delta\underline{x}^-$ from the tableaux to give expressions of the form

$$\Delta\underline{x}(\beta) = \underline{U}^l + \beta\underline{V}^l \qquad (l=1, \ldots l_{max}) \qquad (3.18)$$

In each range $\beta_{l-1} \leq \beta \leq \beta_l$ the vectors $\underline{U}^l$ and $\underline{V}^l$ are constant; these vectors, together with the values β_l, are stored and completely define the solutions; note that $\beta_o=0$. The number of ranges l_{max} must be finite since the step limit constraints require that $\beta_{max} = \sum_{i=1}^{n} p_i$.

3.4 Proof of the equivalence of problems P1 and P2

It will now be shown that if at $\alpha=\alpha_o$ there exists an optimal solution to problem P1 such that the variables are $\underline{\Delta x}$ and the corresponding objective function $\sum_{i=1}^{n} |\Delta x_i|$ equals β_o, then the optimal solution to problem P2 for $\beta=\beta_o$ is given by the same variables and will have an objective function $\sum_{i=1}^{n} |f_i + \sum_{j=1}^{n} J_{ij} \Delta x_j|$ equal to $(1-\alpha_o) \sum_{i=1}^{n} |f_i|$.

In a similar way to that used in section 3.3.2 for problem P2, we can express P1 as the parametric linear programming problem

$$\text{P6: Minimize} \quad \sum_{i=1}^{n} \Delta x_i^+ + \Delta x_i^-$$

$$\text{Subject to} \quad \sum_{i=1}^{n} f_i^+ + f_i^- = (1-\alpha) \sum_{i=1}^{n} f_i^o$$

$$\Delta x_i^+ + \Delta x_i^- + \Delta x_i^s = p_i \quad (i=1, \ldots n)$$

$$f_i^+ - f_i^- - \sum_{j=1}^{n} J_{ij}(\Delta x_j^+ - \Delta x_j^-) = f_i^o \quad (i=1, \ldots n)$$

$$\Delta x_i^+, \Delta x_i^-, \Delta x_i^s, f_i^+, f_i^- \geqslant 0 \quad (i=1. \ldots n)$$

The corrections $\underline{\Delta x}(\alpha)$ will therefore have the same piecewise-linear form shown by equation (3.18) for $\underline{\Delta x}(\beta)$. In each range $\alpha_{l-1} \leqslant \alpha \leqslant \alpha_l$ the gradient $d\beta_o/d\alpha$ of the optimal objective function is constant; there will be a discontinuity in $d\beta_o/d\alpha$ at the end α_l of a range corresponding to the change in the basic variables. It will first be necessary to prove that $d\beta_o/d\alpha \geqslant 0$ over the range $0 \leqslant \alpha \leqslant \alpha_{max}$ in which feasible solutions of problem P6 exist.

In figure 3.2 , the variation of β_o with α in the range $\alpha_{l-1} \leqslant \alpha \leqslant \alpha_l$ is shown by the line EB; it will be assumed that $d\beta_o/d\alpha > 0$. Either

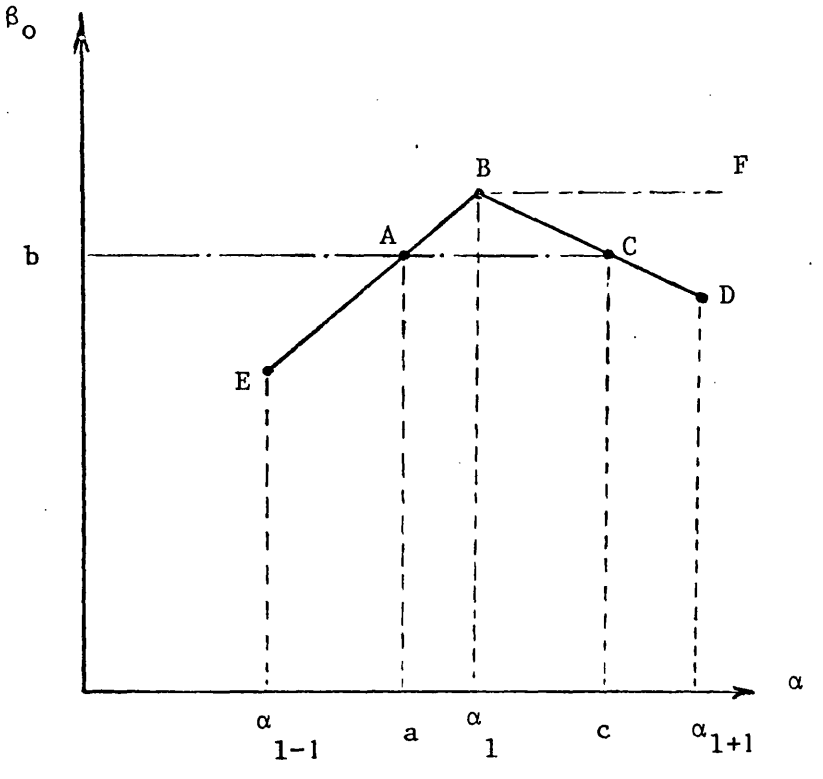


Figure 3.2 Geometric proof that $d\beta_o/d\alpha \not< 0$

$\alpha_1 = \alpha_{max}$ in which case no feasible solutions exist for $\alpha > \alpha_1$ or a further range $\alpha_1 \leqslant \alpha \leqslant \alpha_{l+1}$ of feasible solutions can be obtained by a change in the basic variables. Assume that the latter case applies and that in the new range $d\beta_o/d\alpha < 0$; since $\alpha_{l+1} > \alpha_1$ then the angle DBF must be acute. There must therefore exist two points A and C given by the intersection of the line $\beta_o = b$ with the lines EB and BD at $\alpha = a$ and $\alpha = c$, and such that b is less than the value of β_o at b. Now consider the two vectors $\Delta\underline{x}(a)$ and $\Delta\underline{x}(c)$ corresponding to the optimal solutions at A and C respectively. By linear interpolation we can obtain new corrections

$$\Delta\underline{x}'(\alpha) = \Delta\underline{x}(a) + (\Delta\underline{x}(c) - \Delta\underline{x}(a)) \frac{(\alpha-a)}{(c-a)}$$

These corrections $\Delta\underline{x}'$ must be a feasible solution of P6 provided that $a \leqslant \alpha \leqslant c$; note that this is not necessarily so for values of α outside this range. Similarily, by linear interpolation the value of the objective function corresponding to $\Delta\underline{x}'(\alpha)$ must be constant at $\beta_o = b$; this is less than the optimal solution which, by definition, is given by the lines AB and BC in the range $a \leqslant \alpha \leqslant c$. Consequently we cannot have $d\beta_o/d\alpha < 0$ for the range $\alpha_1 \leqslant \alpha \leqslant \alpha_{l+1}$. The argument can be suitably modified to show that if $d\beta_o/d\alpha = 0$ in this last range, and possibly further ranges, then at the next range for which $d\beta_o/d\alpha \neq 0$ we must have $d\beta_o/d\alpha > 0$.

Now in the first range $0 \leqslant \alpha \leqslant \alpha_1$ it is known that $d\beta_o/d\alpha > 0$ unless no feasible solutions exist for $\alpha > 0$. Thus in the second range, and by induction all succeeding ranges up to α_{max}, we must have $d\beta_o/d\alpha \geqslant 0$.

Return now to the original proposition concerning the equivalence of Problems P1 and P2. Comparison of their respective parametric formulations

P6 and P5 shows that they use the same set of variables and the two sets of equality constraints differ only with respect to the first constraint. It will then be seen if for given α the basic feasible solution $\underline{X}$ gives the objective function $\sum_{i=1}^{n} |\Delta x_i| = \beta_o$ for P6, then for $\beta=\beta_o$ the same vector $\underline{X}$ is also a basic feasible solution of P5 and the objective function will be $(1-\alpha) \sum_{i=1}^{n} f_i^o$. It remains to be shown that if $\underline{X}$ is the optimal solution of P6 then it is also the optimal solution of P5.

Assume that in the range $\alpha_{l-1} \leqslant \alpha \leqslant \alpha_l$ for P6 and the range $\beta_{l-1} \leqslant \beta \leqslant \beta_l$ for P5 that both problems have the same optimal solution vector $\underline{X}$; this is shown in Figure 3.3 by the line AB. The graph can be thought of as showing the variation either of the objective function (β) with α for problem P6 or of objective function $((1-\alpha) \sum_{i=1}^{n} f_i^o)$ with β for problem P5. At B one of the basic variables $\underline{X}$ leaves the solution; there will be a finite number of choices of nonbasic variable which can enter the basis; corresponding to each choice the objective functions would follow lines such as BD. The optimal choice for both problems will be that which minimizes the angle FBD made with the α-axis. Three situations can occur.

First, there may be no feasible solutions, so that B then corresponds to the upper limits α_{max} and β_{max} of the range of feasible solutions for P6 and P5 respectively. Second, one or more feasible solutions exist of which the minimum angle, shown as FBC, is less than a right angle; in this case the same solution remains optimal for both problems for a further range. Note that FBC cannot be less than zero from the earlier proof. Third, one or more feasible solutions exist of which the minimum angle, shown as FBE, is greater than a right angle; in this case a further range of optimal solutions exists for P5 but not P6. Note that in this last event, the objective function of P5 will attain its minimum possible value at B and further

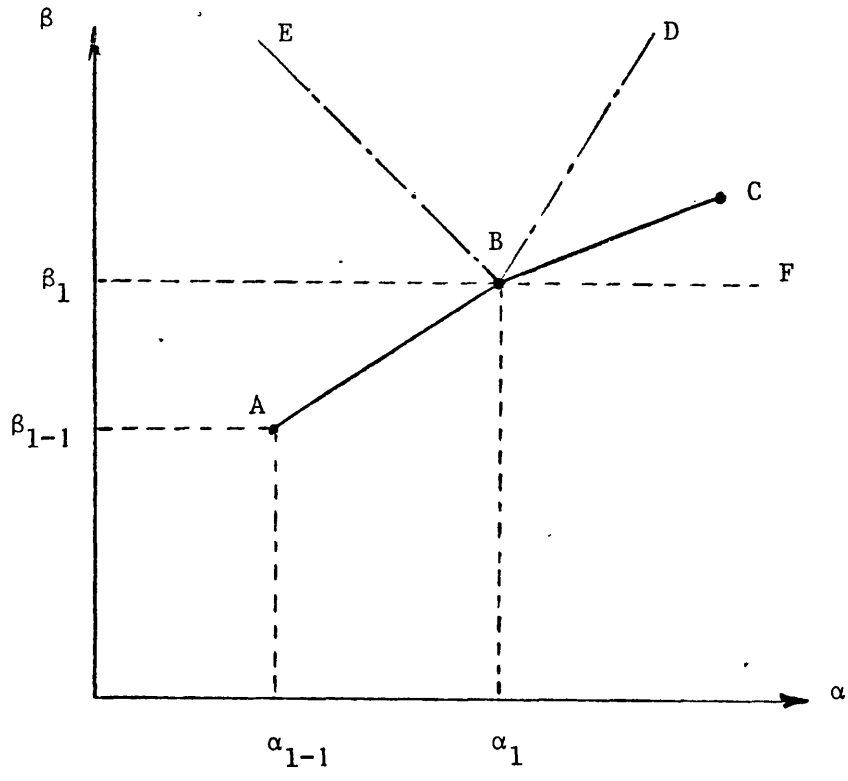


Figure 3.3 Geometric demonstration that P5 and P6 make the same change in the basis

increase of β will produce an increase in the optimal value of the objective function of P5.

Consider now the optimal solutions at $\alpha=0$ and $\beta=0$ for the two problems. The basic set of variables must contain the variables Δx_i^s, f_i^+ (i=1, ... n) in both cases, leaving one more variable to be chosen. From the arguments of the preceding paragraph, the same variable will be chosen in both cases. By induction, at subsequent changes in the basis the same variables will be exchanged to preserve optimality. Thus the equivalence of the two formulations is proved.

3.5 The hybrid property of the corrections

The hybrid corrections discussed in Chapter 2 were obtained by an interpolation between a descent direction and a Newton, or Gauss-Newton direction. The classical steepest descent corrections for a continuous function $F(\underline{x})$ with continuous derivatives can be defined as the solution of the problem

$$\text{Minimize} \quad \sum_{i=1}^{n} \frac{\partial F}{\partial x_i} \Delta x_i$$

$$\text{Subject to} \quad \sum_{i=1}^{n} \Delta x_i^2 = \gamma$$

Murray [25] discusses this further. For any length γ of corrections, using the distance norm $||\Delta\underline{x}|| = \sum_{i=1}^{n} \Delta x_i^2$, the steepest descent corrections are uniquely specified as those that minimize the increase (or maximize the decrease) in $F(\underline{x})$ as approximated by the linear objective function used in the definition. It will be seen that the definition of the corrections $\Delta\underline{x}(\beta)$ using problem P2 has an exact parallel in that, in the absence of the step limits p_i, the corrections of length β using the norm $||\Delta\underline{x}|| = \sum_{i=1}^{n} |\Delta x_i|$ are chosen to minimize the function $S(\underline{x})$. Thus the corrections $\Delta\underline{x}(\beta)$ are also steepest descent corrections.

Furthermore, when β is increased from zero upwards the corrections tend towards the Newton corrections. They will ultimately equal the Newton corrections at $\beta=\beta_{max}$ provided that J is nonsingular and the step limits are not exceeded. The parameter β therefore effects an interpolation between a steepest descent direction and the Newton corrections.

Finally, to emphasise the hybrid nature of the corrections, consider the Marquardt [39] hybrid corrections for least squares problems defined as the solution of

$$\text{Minimize} \quad \sum_{i=1}^{n} \left(f_i + \sum_{j=1}^{n} J_{ij}\, \Delta x_j\right)^2$$

$$\text{Subject to} \quad \sum_{i=1}^{n} \Delta x_i^2 = \gamma$$

Using the method of Lagrange multipliers [2] the Marquardt corrections are $-(J^TJ + \lambda I)^{-1}J^T\underline{f}$ where the value $\lambda \geq 0$ must be determined to satisfy the length γ. Apart from the step-limits, the definition of corrections using Problem P2 differs only in the norm used to measure distance, both for the length of the corrections and the departure of the residuals from zero.

A major disadvantage of the Marquardt corrections is that it is not computationally practical to specify a value for γ and then obtain the corrections; instead one or more values of λ may have to be tried before corrections of a suitable length are obtained. On the other hand, the corrections $\Delta\underline{x}(\beta)$ can be directly computed for any specified length β; further the step limits can be incorporated into the calculation of $\Delta\underline{x}(\beta)$ to produce the most beneficial effect.

3.6 Implementation of the algorithm

An algorithm based on the corrections $\Delta\underline{x}(\beta)$ was developed and has the following structure:

Step 1 — Set k=0; $\underline{x}^0$ to a supplied starting value
Specify appropriate step lengths p_i (i=1, .. n)

Step 2 — Evaluate $\underline{f}$, J at $\underline{x}^k$

Step 3 — Solve problem P5 and obtain all the sets of corrections $\Delta\underline{x}(\beta)$ within the range $0 \leq \beta \leq \beta_{max}$

Step 4 — If a stationary point of $S(\underline{x})$ has been reached at $\underline{x}^k$ then halt

Step 5 — Choose a value β' which must give

$$S(\underline{x}^k + \Delta\underline{x}(\beta')) < S(\underline{x}^k)$$

Step 6 — Set $\underline{x}^{k+1} = \underline{x}^k + \Delta\underline{x}(\beta')$; k=k+1
Return to Step 2

The important features of the implementation of this algorithm will now be discussed.

3.6.1 Computational aspects of solving problem P5

Some aspects of the computations involved in the solution of problem P5 will now be discussed with reference to a worked example. The data used is for the starting-point of Problem 3 in the Appendix; this well-known test problem has two equations in two variables. Step limits of $p_1=p_2=0.5$ were used.

Figure 3.4 illustrates the generation of the optimal solution at $\beta=0$. Since the first equation has a negative residual, the first row of the Jacobian was multiplied by -1 before being used to set up tableau A as shown; this tableau should be compared with its general form in Figure 3.1. The final variable to enter the basis is Δx_1^+ since this has the maximum coefficient c_i on the bottom row. The tableau is then modified by row operations and contracted to remove the column originally occupied by the coefficients of Δx_1^+; this gives tableau B which shows the optimal solution at $\beta=0$.

This solution remains optimal until $\beta=0.18$, when the basic variable f_1^+ becomes zero and is exchanged with variable Δx_2^- to give tableau C in Figure 3.5. The next change of basis is required at $\beta=0.89$ where $\Delta x_2^s=0$, corresponding to the constraint $|\Delta x_2| \leq p_2$ becoming active; Δx_2^s is exchanged with f_1^- to give tableau D. The solution then remains optimal up to $\beta=1.00$, when $\Delta x_1^s=0$; both corrections are then at their limits and no further solutions exist as this is confirmed by the fact that there is no negative value of a_{2j} (j=1, .. 5) in tableau D.

If the criterion of equation (3.16) was used as it stands, at $\beta=0.89$ the variable chosen to enter the basis would be Δx_1^-. Since Δx_1^+ is already in the basis, this would invalidate the assumption made when formulating P5 that the variables Δx_1^+ and Δx_1^- (and other similar pairs) are never in the basis at the same time. Thus a safeguard is necessary to ensure that the choice of nonbasic variable to enter the basis is restricted to only those variables which would not lead to such a conflict; thus in the example f_1^- is chosen instead of Δx_1^-. Note that if Δx_1^- was permitted to enter the basis, in the range $0.89 < \beta < 1.00$ the value of Δx_1 given by $\Delta x_1^+ - \Delta x_1^-$ would remain constant at -0.40 while its modulus $\Delta x_1^+ + \Delta x_1^-$ would equal $-0.50 + \beta$ and would therefore increase with β.

	Δx_1^+	Δx_2^+	Δx_1^-	Δx_2^-	f_1^-	f_2^-	1	β
-	1.00	1.00	1.00	1.00	0.00	0.00	0.00	1.00
Δx_1^s	1.00	0.00	1.00	0.00	0.00	0.00	0.50	0.00
Δx_2^s	0.00	1.00	0.00	1.00	0.00	0.00	0.50	0.00
f_1^+	24.00	10.00	-24.00	-10.00	-1.00	0.00	4.40	0.00
f_2^+	1.00	0.00	-1.00	0.00	0.00	-1.00	2.20	0.00
Z	25.00	10.00	-25.00	-10.00	-2.00	-2.00	6.60	0.00

A: Tableau before inclusion of Δx_1^+ in basis

	Δx_2^+	Δx_1^-	Δx_2^-	f_1^-	f_2^-	1	β
Δx_1^+	1.00	1.00	1.00	0.00	0.00	0.00	1.00
Δx_1^s	-1.00	0.00	-1.00	0.00	0.00	0.50	-1.00
Δx_2^s	1.00	0.00	1.00	0.00	0.00	0.50	0.00
f_1^+	-14.00	-48.00	-34.00	-1.00	0.00	4.40	-24.00
f_2^+	-1.00	-2.00	-1.00	0.00	-1.00	2.20	-1.00
Z	-15.00	-50.00	-35.00	-2.00	-2.00	6.60	-25.00

B: Tableau after inclusion of Δx_1^+ in basis and contraction

Data: Starting point of Problem 3 in the Appendix

$$\underline{x} = \begin{bmatrix} -1.20 \\ 1.00 \end{bmatrix} \qquad \underline{f} = \begin{bmatrix} 4.40 \\ 2.20 \end{bmatrix} \qquad J = \begin{bmatrix} -24.00 & -10.00 \\ -1.00 & 0.00 \end{bmatrix}$$

Step limits : $p_1 = p_2 = 0.5$

Figure 3.4 Generation of the initial tableau at β=0

	Δx_2^+	Δx_1^-	f_1^+	f_1^-	f_2^-	1	β
Δx_1^+	0.59	-0.41	0.03	-0.03	0.00	0.13	0.29
Δx_1^s	-0.59	1.41	-0.03	0.03	0.00	0.37	-0.29
Δx_2^s	0.59	-1.41	0.03	-0.03	0.00	0.63	-0.71
Δx_2^-	0.41	1.41	-0.03	0.03	0.00	-0.13	0.71
f_2^+	-0.59	-0.59	-0.03	0.03	-1.00	2.07	-0.29
Z	-0.59	-0.59	-1.03	-0.97	-2.00	2.07	-0.29

C : Tableau after exchange of f_1^+ and Δx_2^- at β=0.18

	Δx_2^+	Δx_1^-	f_1^+	Δx_2^s	f_2^-	1	β
Δx_1^+	0.00	1.00	0.00	-1.00	0.00	-0.50	1.00
Δx_1^s	0.00	0.00	0.00	1.00	0.00	1.00	-1.00
f_1^-	-20.00	48.00	-1.00	-34.00	0.00	-21.40	24.00
Δx_2^-	1.00	0.00	0.00	1.00	0.00	0.50	0.00
f_2^+	0.00	-2.00	0.00	1.00	-1.00	2.70	-1.00
Z	-20.00	46.00	-2.00	-33.00	-2.00	-18.70	23.50

D : Tableau after exchange of Δx_2^s and f_1^- at β=0.89

Data : As given in Figure 3.4

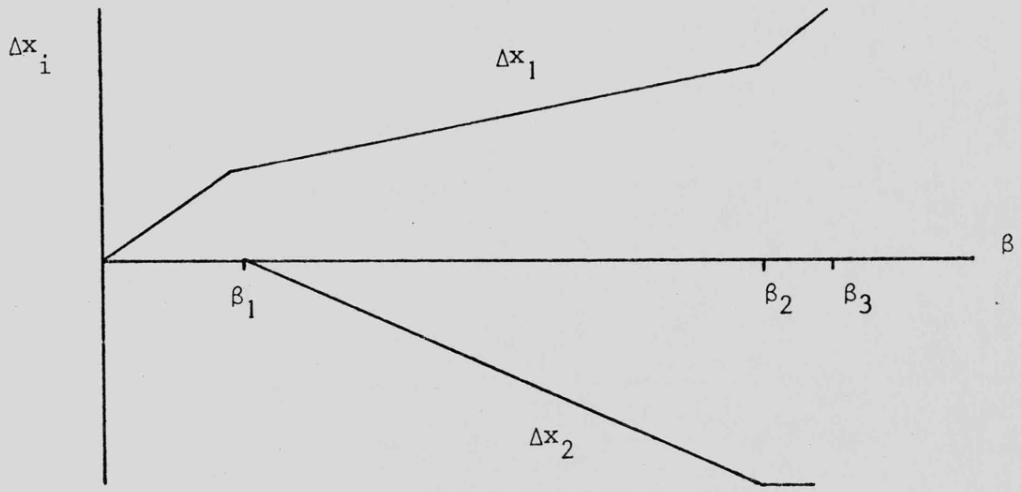
Figure 3.5 Updated tableaux at changes in basis

Figure 3.6 shows the variation with β of the corrections $\Delta x_i(\beta) = \Delta x_i^+ - \Delta x_i^-$; the predicted residuals $f_i^+ - f_i^-$ and the predicted value $\sum_{i=1}^{n} f_i^+ + f_i^-$ of $S(\underline{x})$ using the results in the tableaux of Figures 3.4 and 3.5. The computed values $\underline{f}(\underline{x}^k + \Delta\underline{x}(\beta))$ and $S(\underline{x}^k + \Delta\underline{x}(\beta))$ are shown by way of comparison. Note that the change in basis at β=0.89 makes $S(\underline{x}^k + \Delta\underline{x}(\beta))$ nonunimodal. As would be expected from problem 3, the second residual is predicted exactly. The example shown is not typical in so far as the global minimum of $S(\underline{x}^k + \Delta\underline{x}(\beta))$ occurs at a greater value of β than the predicted minimum.

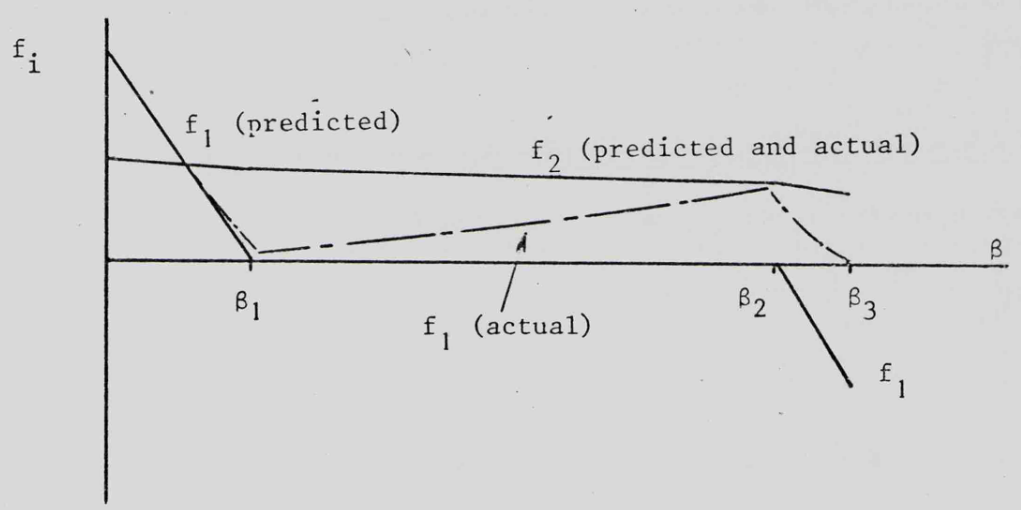
One last feature of note in the computation of $\Delta\underline{x}(\beta)$ concerns the assumption that the initial tableau, when generated in the way previously described, gives the optimal solution at β=0. Consider the situation when one of the residuals, say f_i, is zero. The initial tableau could be generated in two ways, both valid, such that in one the elements of J are used unaltered and in the other row i of J is first multiplied by -1. The choice of correction to complete the initial set of basic variables can thereby be affected, so that the two tableaux will then be different and cannot both be optimal.

This inconsistency is automatically resolved by the parametric linear programming method. It will be found in one or other case (perhaps both) that the basic variable f_i^+ decreases with β and therefore the first change of basis will be at β_1=0. One or more changes of basis may be necessary before a finite range $0 \leq \beta \leq \beta_1$ is obtained within which the solution remains optimal. For both cases, this solution will be the same although the series of changes in the basic variables will be different.

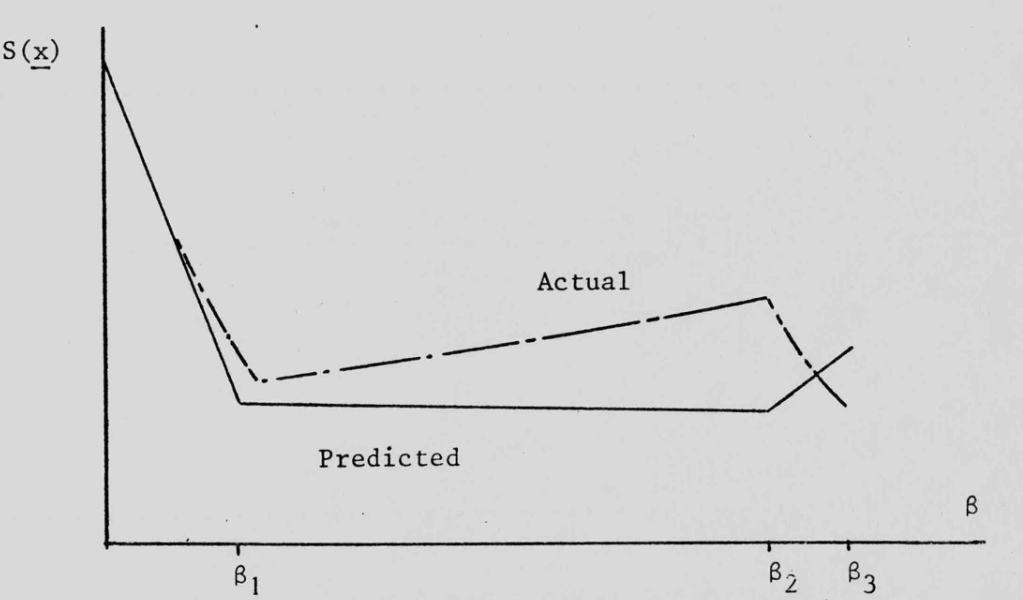
A related situation occurs when at least one of the residuals is zero. It has been assumed that the optimal solution at β=0 contains only



(a) Hybrid corrections



(b) Predicted and actual residuals



(c) Predicted and actual values of $S(\underline{x})$

Figure 3.6 Predicted and actual effects of the hybrid correction

one correction. It can happen, as will be discussed in the next section, that a downhill direction cannot be generated with a single nonzero component of $\Delta\underline{x}(\beta)$ while a downhill direction can be generated with two or more nonzero components. In this event, the initial tableau is not optimal and exchanges are made at $\beta=0$ to introduce additional corrections so that, if possible, a downhill direction is attained.

Both of the situations just described are illustrated in Figure 3.7. The two cases correspond to different choices of sign for the first row of the Jacobian; the correction chosen to complete the initial basis is Δx_1^+ for case A and Δx_1^- for case B. Neither of the solutions are feasible for $\beta>0$ since $f_1^+ = -20\beta$; in both cases f_1^+ is exchanged with Δx_2^-. The solution for case B is then optimal and feasible in the range $0 \leq \beta \leq 0.75$. However, in case A we have $\Delta x_1^+ = -\beta$ and this variable is exchanged with Δx_1^-; the solution is then identical to that for case B.

3.6.2 Convergence criterion

For a function $F(\underline{x})$ with continuous first and second derivatives, the well-known conditions for $\underline{x}^k$ to be a local minimum are that $\underline{x}^k$ must be a stationary point and that the Hessian matrix of $F(\underline{x})$ is positive definite. The objective function $S(\underline{x})$ used in this hybrid algorithm has discontinuous derivatives at points where at least one of the residuals is zero. Thus different criteria are required to determine when a local minimum has been reached.

Assume that at $\underline{x}^k$ the residuals are all non-negative; this can always be ensured by changing any equation $f_i(\underline{x})=0$ having a negative residual to the equivalent equation $-f_i(\underline{x}) = 0$. Now for a small change

Initial Basis	Basis at $\beta_1=0$	Basis at $\beta_2=0$
$\Delta x_1^+ = \beta$	$\Delta x_1^+ = -\beta$	$\Delta x_1^- = \beta/3$
$\Delta x_1^s = \frac{1}{2} - \beta$	$\Delta x_1^s = \frac{1}{2} + \beta$	$\Delta x_1^s = \frac{1}{2} - \beta/3$
$\Delta x_2^s = \frac{1}{2}$	$\Delta x_2^s = \frac{1}{2} - 2\beta$	$\Delta x_2^s = \frac{1}{2} - 2\beta/3$
$f_1^+ = -20\beta$	$\Delta x_2^- = 2\beta$	$\Delta x_2^- = 2\beta/3$
$f_2^+ = 2 + \beta$	$f_2^+ = 2 - \beta$	$f_2^+ = 2 - \beta/3$

Case A - First row of J unchanged

Initial Basis	Basis at $\beta_1=0$
$\Delta x_1^- = \beta$	$\Delta x_1^- = \beta/3$
$\Delta x_1^s = \frac{1}{2} - \beta$	$\Delta x_1^s = \frac{1}{2} - \beta/3$
$\Delta x_2^s = \frac{1}{2}$	$\Delta x_2^s = \frac{1}{2} - 2\beta/3$
$f_1^+ = -20\beta$	$\Delta x_2^- = 2\beta/3$
$f_2^+ = 2 - \beta$	$f_2^+ = 2 - \beta/3$

Case B - First row of J multiplied by -1

Data: Problem 3 in the Appendix at $\underline{x}^T = (-1, 1)$

Linear transformation of variables is used

Step Limits $p_1 = p_2 = \frac{1}{2}$

$$\underline{f} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \qquad J = \begin{bmatrix} -20 & 10 \\ 1 & 0 \end{bmatrix}$$

Figure 3.7 Generation of the tableau at $\beta=0$ when one of the residuals is zero.

$\delta\underline{x}$ in the variables at $\underline{x}^k$ the corresponding change δS in $S(\underline{x})$ will be given to a first-order approximation by

$$\delta S \simeq \sum_{i,f_i>0} \sum_{j=1}^{n} J_{ij}\, \delta x_j + \sum_{i,f_i=0} \left| \sum_{j=1}^{n} J_{ij}\, \delta x_j \right| \qquad (3.19)$$

For $\underline{x}^k$ to be a stationary-point, it will be seen that we must have for each change δx_j (j=1, .. n)

$$\sum_{i,f_i>0} J_{ij} = 0 \qquad (3.20a)$$

$$J_{ij} = 0 \qquad \text{for all i such that } f_i=0 \qquad (3.20b)$$

If these conditions hold, then the second derivatives of $\underline{f}(\underline{x})$ must be considered to determine whether $\underline{x}^k$ is a minimum. Note that the linear programming solution for $\Delta\underline{x}(\beta)$ would fail in the generation of the initial tableau if these conditions hold.

In practice, it is found that minima of $S(\underline{x})$ do not occur at a stationary point. In this event, it will be seen from equation (3.19) when a vector $\delta\underline{x}$ with a single nonzero component δx_j is considered, that a necessary condition for $\underline{x}^k$ to be a local minimum is that $\delta S > 0$ for $\delta\underline{x}$ or $-\delta\underline{x}$ and so

$$\left| \sum_{i,f_i>0} J_{ij} \right| < \sum_{i,f_i=0} |J_{ij}| \qquad (j=1, .. n) \qquad (3.21)$$

Note that this requires that at least one zero residual exists. However, this condition is not sufficient, since a vector $\delta\underline{x}$ may exist with two or more nonzero components for which $\delta S < 0$; this situation was in fact the case in the example shown in Figure 3.7, where two corrections in

the initial basis produced a downhill direction while one correction could not.

Although the sufficient conditions are complicated, the linear programming method itself determines the optimal direction at $\underline{x}^k$; consequently if the objective function $\Sigma f_i^+ + f_i^-$ of P5 is increasing with β at $\beta=0$, then a local minimum has been reached. Suppose that S_1 is the value of the objective function at $\beta=\beta_1$; the lowest value of the objective function must occur at one such point, say $\beta=\beta_{1min}$. If $S_{1min} > S(\underline{x}^k)$ a local minimum will have been attained. Roundoff error will be significant near a local minimum. It was stated that at least one residual must be zero (unless at a stationary point) and in practice an exact zero is unlikely. Consequently near to a local minimum the usual situation is for S_{1min} to be less than $S(\underline{x}^k)$ by a very small amount and for β_{1min} to be small also.

The convergence criterion adopted therefore was that a local minimum was reached at $\underline{x}^k$ when either $\beta_{1min} < \varepsilon_1$ or $S(\underline{x}^k) - S_{1min} < \varepsilon_2 S(\underline{x}^k)$. These two criteria proved to be adequate for the test problems used. Other workers such as Wolfe [46] carry out a final grid search about the assumed minimum; the objective function is evaluated at 2n points obtained by perturbing each variable in turn by a small amount both above and below its value at the minimum. If no further decrease in the objective function is found, then it can be confidently assumed that a local minimum has indeed been located. Such a check is of most use in checking that a stationary point is not a saddle-point; since $S(\underline{x})$ is unlikely to have any stationary points, this additional check was not used for the hybrid algorithm.

3.6.3 Method used to choose β'

At each iteration of the hybrid algorithms, a value $\beta=\beta'$ is required such that the corrections $\Delta\underline{x}(\beta')$ give a reduction in $S(\underline{x})$. The first method to be tried was that β' should correspond to the global minimum of the nonunimodal function $S(\underline{x}^k + \Delta\underline{x}(\beta))$. A multimodal search similar to that for the descent algorithm described in the previous chapter was used; the grid was made up of the series of values $.. \beta_{l-1}, (\beta_{l-1} + \beta_l)/2, \beta_l ... (l=1, .. l_{max})$. The computational labour required by the search was high; it was found that when a coarser search for β' was used the reliability of the algorithm was not impaired. Thus the adopted method of choosing β' was as follows.

Step 1 — Set $\beta'=\beta_{1min}/2^N$ where N is set to zero prior to the first search for β'

Step 2 — If $S(\underline{x}^k + \Delta\underline{x}(\beta')) < S(\underline{x}^k)$ then set $N=\max(0, N-1)$ and continue to the next iteration of the hybrid algorithm.

Step 3 — If $\beta' < \varepsilon_1$ then the search for β' has failed and the hybrid algorithm is terminated.

Step 4 — Set $\beta'=\beta'/2$; $N=N+1$; Return to Step 2.

At each search, the first choice for β' is β_{1min}, the position of the predicted minimum, divided by 2^N where N is set at the previous iteration. It is assumed that at successive iterations, the chosen values for β' will be roughly the same in proportion to the corresponding values for β_{1min}. It will be seen that if $\beta'=\beta_{1min}/2^{N_k}$ at iteration k

of the hybrid algorithm, then a realistic estimate of β' at the next iteration will be $\beta'=\beta_{1min}/2^{N_k-1}$ when $N_k > 1$ and β_{1min} when $N_k \leq 1$, as used by the search algorithm. Note that the estimate errs on the high side since β' cannot be increased above its initial estimate during each search.

The choice of β' is restricted to the range $0 \leq \beta' \leq \beta_{1min}$ and so there is no point in extending the parametric solutions beyond β_{1min} in this case, although as seen in Figure 3.6 the global minimum of $S(\underline{x}^k + \Delta\underline{x}(\beta))$ may lie in this region. At step 3, if failure occurs then it is likely that a local minimum of $S(\underline{x}^k)$ has been reached but not detected because the convergence criteria are too strict; note that ε_1 is the same parameter used in the convergence criterion based on β_{1min}. Lastly, it will be seen that the residuals at $\underline{x}^{k+1}$ will be available at the end of the search on β' (this is not necessarily the case on exit from a quadratic interpolation or other more refined search) and only the Jacobian needs to be evaluated before continuing with the next iteration of the hybrid algorithm.

3.7 Extension to overdetermined sets of equations

In its final form, the hybrid algorithm was modified to cope with overdetermined sets of m nonlinear equations in n variables. Barrodale and Roberts [32] give an algorithm which obtains the optimum solution of an overdetermined set of linear equations using the l_1-norm (sum of moduli) criterion. The main application of their algorithm is to linear regression. Often a fit obtained using the l_1-norm will be better than one obtained using the l_2-norm (sum of squares) criterion; this is especially likely when one or more rogue points occur in the data. Barrodale and Roberts believe that the reason why experimenters do not make better use of the l_1-norm is that few statistical measures of goodness-of-fit are available as compared with those for fits obtained using the l_2-norm.

The hybrid algorithm, when modified to cope with m equations, can thus be considered as complementary to the Barrodale and Roberts algorithm in that it handles nonlinear equations. Barrodale and Roberts used a non-parametric linear programming method of solution but took advantage of the specialised nature of the problem to reduce the size of the tableau and also save on the computations; possibly some of their ideas could also apply to the parametric solution used by the hybrid algorithm. The required modification to the algorithm is straightforward; the objective function of problem P2 must be changed to $\sum_{i=1}^{m} \left| f_i + \sum_{j=1}^{n} J_{ij} \Delta x_j \right|$. The equivalent parametric problem P5 then has $2m + 3n$ variables and $m + n + 1$ constraints and the tableau must therefore be adjusted suitably. In all other respects, the logic remains the same.

3.8 Numerical experience with the algorithm

The final version of the algorithm was implemented as a FORTRAN IV program, using double-precision arithmetic, on an IBM 360/44. In addition to the normal diagnostic printout of intermediate results during the development of the program, two independent checks on the correctness of the program were conveniently available.

First, it will be noted that if m=n and suitably large step limits p_i are used, then the minimum value of the objective function of P5 will be at β_{1min} and the corrections $\Delta\underline{x}(\beta_{1min})$ will be identical to the Newton corrections, provided that these exist. The hybrid algorithm was therefore used in this way to generate the Newton corrections for Problem 1, at x_i^o=5.0, and at the starting point of Problem 2; these estimates were found to be in agreement with the Newton corrections found by direction solution of the equations $J\Delta\underline{x} = -\underline{f}$.

Second, a linear test problem of 7 equations in 2 variables with known solution was solved using the hybrid algorithm. The problem is due to Spyropoulos, Kiountouzis and Young [74] who state that it is a good test of an algorithm's ability to cope with degeneracies arising in the linear programming tableau. The problem is to find the best straight line y=a+bx to fit, in the l_1-norm, the set of points (Y_i, X_i), i=1, .. 7 where $\underline{X}^T$ = (1,2,3,4,5,6,7) and $\underline{Y}^T$ = (1,12,3,4,5,12,7); the solution is a=1 and b=1. When the hybrid algorithm was applied to this problem, with x_1=a and x_2=b, it converged at the correct solution in one iteration as would be expected.

Once it had been confirmed that the hybrid algorithm had been implemented successfully, it was tested on the nonlinear problems in the

Appendix. The tests were conducted under similar conditions to those used for the two-part algorithm, namely a logarithmic transformation of the variables for Problem 1 and a linear transformation for the remaining problems; in every case, the step limit p_i was set at 0.5 for each variable. The parameters ε_1 and ε_2 used in the tests for convergence were set at 10^{-5} and 10^{-3} respectively.

The results obtained with Problems 2-9 are shown in Table 3.1. Problems 2, 6, 8 and 9 do not have an exact solution and so the optimum solution obtained is that which minimizes the sum of the moduli of the residuals. As would be expected, the solutions are very similar to those obtained using a least squares approach. Note that the final sum of moduli for the other problems were exactly zero showing how the algorithm can make fine adjustments near a solution. The computational effort n_c is defined as $n_F + n \times n_D$ where n_F and n_D are the number of evaluations of the residuals and first derivatives respectively. Two figures are shown for the number of iterations and the computational effort. The first is for the algorithm in its form described. The second is for tests carried out in which β' was set to β_{1min} at each iteration and no check was made that the residuals were reduced. In fact, in the majority of cases $S(\underline{x})$ did decrease at each iteration, although occasionally it did increase; in every case the search converged to the solution shown. The net effect of this change was to reduce the number of iterations required and thus the computational effort. The results shown compare favourably, both in terms of reliability and efficiency, with those obtained for the other four algorithms shown in Table 2.6.

The performance on Problem 1, the difficult system of eight equations, was disappointing. Convergence was achieved only when the hybrid algorithm was started from points slightly perturbed from the solution itself, such as when all the variables are set to 0.5 above their values at the solution.

Problem Number	Initial Sum of Moduli $S(\underline{x}^o)$	Number of Iterations *	Final Sum of Moduli	Computational Effort * n_c	Variables after optimization		
					x_1	x_2	x_3
2	0.2022	12/8	0.0093	59/36	2.861	15.12	0.8362
3	6.600	12/7	0.0	47/24	1.000	1.000	-
4	5.864	18/8	0.0	72/21	1.000	1.000	-
5	48.36	9/12	0.0	38/52	14.30	1.500	20.10
6	48.89	4/4	2.010	19/20	36.00	1.4853	20.00
7	1451_{10}^{11}	52/19	0.0	253/80	15.50	1.200	0.0200
8	1451_{10}^{11}	51/16	0.1717	250/68	15.78	0.8996	0.0236
9	1331_{10}^{5}	77/9	26.34	393/40	0.0057	6167.0	344.7

* Second of two figures obtained using $\beta' = \beta_{1min}$ at every iteration

Table 3.1 Results obtained with the hybrid algorithm on Problems 2-9

When started from any of the fifteen points used in the previous tests, the algorithm terminated at a local minimum. Although in a sense the algorithm had, in these situations, achieved its aim of finding a minimum of $S(\underline{x})$, the minima obtained were well-removed from the global minimum of zero.

The most commonly-encountered local minimum was at $S(\underline{x}) = 27.37$. This was achieved by any vector $\underline{x}$ for which $x_1 = 0.9044$ and $x_2 = 1.1057$; this minimum is independent of the values of the remaining six variables. Examination of the equations in the Appendix showed that, since in this case $x_1 x_2 = 1$ and the exponential terms include the multiplying factor $1-x_1 x_2$, this minimum corresponded to the minimization of

$$S(\underline{x}) = \sum_{i=1}^{4} |Y_{4i}\, x_2 - Y_{5i}| + |Y_{4i} - Y_{5i}\, x_1|$$

subject to the constraint $x_1 x_2 = 1$. It is interesting to note that a similar argument explains why, with the two-part algorithm, the descent algorithm made slow progress when $x_1 x_2 \simeq 1$ and $F(\underline{x}) \simeq 190$.

The lowest minimum found was at

$$\underline{x}^T = (0.9010,\ 0.8938,\ 3.755,\ 5.387,\ 10.97,\ 0.0,\ 1.103,\ 0.6729);$$

at this point $f_6 = 0.2737$ and all the remaining residuals are zero. This has a close similarity to the local minimum of $F(\underline{x})$ as shown in the Appendix. If the logarithmic constraints were removed, the alternative solution, with some variables negative, was readily found by the hybrid algorithm. For example, starting from $x_i^o = 5$ $(i=1, \ldots 8)$ this solution was found in 15 iterations at a computational effort of 144.

3.9 Some related algorithms

The hybrid algorithm can be modified to minimize in the l_2-norm (sum of squares) by choosing $\Delta\underline{x}(\beta)$ as the solution of the quadratic programming problem

$$\text{P7:} \qquad \text{Minimize} \quad \sum_{i=1}^{m} \left(f_i(\underline{x}^k) + \sum_{j=1}^{n} J_{ij}\, \Delta x_j\right)^2$$

$$\text{Subject to} \quad \sum_{i=1}^{n} |\Delta x_i| = \beta$$

$$|\Delta x_i| \leq p_i \qquad (i=1, \ldots n)$$

This is very similar to the definition of the Marquardt corrections given in section 3.5. Some experiments were carried out using this formulation. An algorithm due to Beale [75] was available in the NAG library for the solution of the quadratic programming problem with a convex objective function so this was used. Unfortunately, a parametric version was not available and so the quadratic programming problem had to be solved for every trial value of β.

The results from the tests with the hybrid algorithm showed that, at most iterations, the corrections at $\beta=\beta_{1min}$ corresponding to the lowest value of the objective function produced a reduction in $S(\underline{x})$. Consequently, the first constraint of P7 was changed to $\sum_{i=1}^{n} |\Delta x_i| \leq \beta$ and at each iteration P7 was solved with $\beta = \sum_{i=1}^{n} p_i$. The resulting corrections will correspond to the value of β_{1min} as defined previously but this time for the quadratic objective function. If these corrections do not reduce $F(\underline{x})$ then the next trial value of β is set to $\beta_{min}/2$ where β_{min} is found from the summation $\sum_{i=1}^{n} |\Delta x_i|$ using the corrections just obtained.

The results were inconclusive; least squares solutions of some of the problems in the Appendix were obtained, but at the cost of considerably more computational labour than for the hybrid algorithm. A fair comparison would require the development of a parametric version of Beale's algorithm, making use of the known optimal solution at $\beta=0$ and incorporating a safeguard to prevent the inclusion together in the basis of variables of the form y^+ and y^-. Some linear approximations of the sum of squares objective function were also tried using the parametric method of the hybrid algorithm; again, some fuller investigations are necessary before firm conclusions can be drawn on the worth of this approach.

Finally, it should be observed that a similar way of defining corrections has been used by Madsen [76] following earlier work by Osborne and Watson [33]. Madsen used the l_∞ or minimax norm and solved the problem

$$\text{Minimize} \quad \max_i \left| f_i(\underline{x}^k) + \sum_{j=1}^{n} J_{ij} \Delta x_j \right|$$

$$\text{Subject to} \quad \max_i \left| \Delta x_i \right| \leq p$$

A major difficulty which Madsen did not circumvent was the effect of scaling; as can be seen if a large change in one variable produces a similar change in the residuals to that produced by a small change in another variable, then the single value p used in this constraint cannot be chosen appropriately for both variables. Since the hybrid algorithm limits each correction individually it does not suffer from this difficulty since each limit can be set to a suitable value. The linear transformation of the variables if incorporated into Madsen's algorithm might enable it to cope with badly-scaled problems. At each iteration, Madsen tried one or more values for p and had to solve the linear programming problem *ab initio*

every time; a parametric approach as used by the hybrid algorithm could well prove beneficial.

Recently, Anderson and Osborne [77] have given an algorithm which generalises Madsen's method to other norms. This new algorithm could be used with the l_1 norm, like the hybrid algorithm of the author; however unlike the author's algorithm it does not include the facility to limit individual components of the corrections and it does not employ the parametric programming technique.

The author believes that similar techniques to those described in this Chapter could usefully be applied to the method of approximation programming for constrained problems. As described by Griffith and Stewart [78] this method solves a series of linear programming problems, each of which approximates the nonlinear programming problem within a small range, and which it is hoped will eventually lead to the solution of the nonlinear problem.

4. CASE STUDIES OF THE APPLICATION OF OPTIMIZATION TECHNIQUES

4.1 Introduction

This chapter consists of a number of case studies illustrating the varied ways in which the author has applied the techniques of optimization to problems of practical relevance. It is not claimed that the cases are a representative cross-section of the many different areas in which optimization techniques are currently being used. The main source from which they are drawn is the field of engineering, as also is the case with the conference proceedings edited by Dixon [79] ; other sources are the fields of biochemistry, theoretical physics and statistics. In spite of this somewhat limited range, the author believes that many of the experiences described have relevance to other areas of application.

In choosing an optimization algorithm, the major considerations must be reliability and efficiency; the relative importance of these two criteria will depend upon the particular problem. There are no hard and fast rules which will determine the choice. Statistical measures of the relative merits of differing algorithms are not feasible since to obtain such measures the algorithms in question would have to be tested on a sample of problems drawn at random from the total population of all problems. In practice, comparisons given in the literature are usually on the basis of a small sample of standard test cases; further there is usually a bias in these test cases in that they are often of an artificial nature chosen to illustrate some particular difficulty. For example, the banana-like valley of Rosenbrock [19] is a good test of an algorithm's ability to cope with an objective function with contours forming a steep-sided valley, but the problem itself would, if met by a researcher, be solved by inspection. Although more recently there has been a tendency to include larger numbers of test cases in published work, the problem of sample bias will always remain.

The results of the comparison of two algorithms on the same problem can themselves be suspect. For example, Dixon [64] showed that differences in behaviour of a class of updating formulae for rank-one methods were due solely to inaccuracies in the line searches. Nash [43] found that Meyer and Roth [42] had unjustly compared Marquardt's algorithm unfavourably with an algorithm of their own. Nash found that with a slightly different implementation of Marquardt's algorithm it converged where previously it had failed on Meyer and Roth's test problems; furthermore Meyer and Roth only obtained convergence with their algorithm in all cases by variation of parameters in their program.

Much attention is quite rightly devoted to the efficiency of an algorithm, usually in terms of computational effort but often also in terms of the computer storage. When an algorithm is widely-used then obviously small increases in efficiency are multiplied many times. However, there are many other sources of inefficiency, such as human error, poor formulation of the problem and failure of the chosen algorithm to locate a solution. All of these causes can result in waste of computer resources on a scale far in excess of that due to choosing an algorithm which is less efficient than another.

A pragmatic choice of algorithm is thus often made on the basis of other considerations, such as whether the objective function is a sum of squares function, the availability of derivative information, the relative cost of processor time and memory and the degree of accuracy required in the solution. The author believes that too much emphasis is placed on the choice of method and too little on the savings that can be made in formulating the problem in the best possible way. Problems are often a lot simpler than they seem at first sight, and this is illustrated by some of the case studies where the choice of algorithm for the optimization is often of secondary importance.

4.2 Problems with one variable

Optimization problems involving a nonlinear function of a single variable are of special interest for several reasons. First, certain methods have been developed specifically for such problems. Second, line searches to find a minimum of a function are an important feature of many algorithms for the optimization of problems with several variables. Third, as will be shown for example in sections 4.3.3 and 4.4.3, it is often possible to reduce problems of more than one variable to a form involving a search on a single variable.

The most well-known methods of locating the minimum of a single-variable function are the Golden Section and Fibonacci searches, described by Kowalik and Osborne [9]. These methods use function values only; more sophisticated techniques often requiring derivative information have been developed using quadratic [13] or cubic [14] interpolating functions, usually with safeguards to ensure convergence. The related problem of searching for a zero of a function can be done using the well-known Newton method, the method of false position or a dichotomous search. The solution of a single nonlinear equation would normally be done in one of these ways, whilst a set of equations is often solved by minimizing a sum of squares objective function.

The first three case studies which follow are essentially the solution of single nonlinear equations; the fourth example is of a complicated line search used in an optimization algorithm.

4.2.1 Radiative heat transfer in dielectrics

This case study arises from work Maxwell [80] undertook on a chemical process engineering problem involving heat transfer through layers of glass. Maxwell sought the author's assistance with the solution of a set of equations involving $2n + 1$ unknowns , which governed the transfer of the energy of electromagnetic radiation in dielectric media. The equations can be written as

$$(W + \gamma M)\ \underline{X} = 0 \qquad (4.1)$$

where the unknowns are the scalar γ and the vector $\underline{X}^T = (X_1, \ldots X_{2n})$. The square matrices W and M are of order 2n and whose elements are given, for any row i, by the expressions

$$W_{ij} = -\frac{w_j}{2} + \delta_{ij} \qquad j = 1, \ldots n$$

$$W_{ij} = -\frac{w_{j-n}}{2} + \delta_{ij} \qquad j = n+1, \ldots 2n$$

$$M_{ij} = 0 \qquad j \neq i$$

$$M_{ij} = \mu_i \qquad j = i \leqslant n$$

$$M_{ij} = -\mu_{i-n} \qquad j = i > n$$

where δ_{ij} is the Kronecker delta symbol and the constants w_i and μ_i satisfy

$$\sum_{i=1}^{n} w_i = 1 \qquad (4.2a)$$

$$w_i > 0 \qquad i = 1, \ldots n \qquad (4.2b)$$

$$\mu_i > \mu_{i-1} > 0 \qquad i = 2, \ldots n \qquad (4.2c)$$

Non-zero solutions for $\underline{X}$ can only occur at values of γ for which $W + \gamma M$ is singular. To find these values of γ, we can first premultiply equation (4.1) by the inverse of M to give the equivalent set of equations

$(M^{-1}W + \gamma I)\underline{X} = 0$ where I is the unit diagonal matrix. The eigenvalues λ_i and the corresponding eigenvectors $\underline{v}_i$ of the unsymmetric matrix $M^{-1}W$ can then be found by any suitable numerical procedure, to give the required solutions to equation (4.1) in the form $\gamma = -\lambda_i$ and $\underline{X} = \alpha\underline{v}_i$, where α is an arbitrary constant. Such an approach using an eigenvalue/eigenvector analysis could be costly in terms of computer resources and also susceptible to rounding-error problems. The author felt that a further examination of the equations might lead to a simpler method of solution.

Referring to Figure 4.1(a) which shows the equations in a fuller manner, if we start with the last equation and work backwards to the second equation, at each stage subtracting the preceding equation from the current equation, we find that the equations reduce to the simpler form shown in Figure 4.1(b). Non zero solutions for $\underline{X}$ will, as stated before, occur at values of γ for which the matrix of coefficients is singular, corresponding to a zero determinant. By multiplying out, the determinant Δ is given by

$$\Delta = (1-\gamma^2\mu_1^2)\ (1-\gamma^2\mu_2^2)\ \ldots\ (1-\gamma^2\mu_n^2)\left(1-\sum_{i=1}^{n}\frac{w_i}{1-\gamma^2\mu_i^2}\right) \qquad (4.3)$$

Further examination of Figure 4.1(b) shows that the second and subsequent equations yield the relationships

$$(1 + \gamma\mu_i)X_i = (1 + \gamma\mu_{i-1})X_{i-1} \qquad i = 2, \ldots n \qquad (4.4a)$$

$$(1 - \gamma\mu_1)X_{n+1} = (1 + \gamma\mu_n)X_n \qquad (4.4b)$$

$$(1 - \gamma\mu_i)X_{n+i} = (1 - \gamma\mu_{i-1})X_{i-1} \qquad i = 2, \ldots n \qquad (4.4c)$$

Given a value $\gamma = \gamma_i$ for which $\Delta = 0$, then the first equation in Figure 4.1(b) will be a linear combination of the others since these form an independent set as seen by equations (4.4a) - (4.4b). The solution $\underline{X}_i$

$$
\begin{bmatrix}
1+\gamma\mu_1-\frac{w_1}{2} & -\frac{w_2}{2} & \cdots & -\frac{w_n}{2} & -\frac{w_1}{2} & \cdots & -\frac{w_{n-1}}{2} & -\frac{w_n}{2} \\
-\frac{w_1}{2} & 1+\gamma\mu_2-\frac{w_2}{2} & \cdots & \frac{w_n}{2} & -\frac{w_1}{2} & \cdots & -\frac{w_{n-1}}{2} & -\frac{w_n}{2} \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\
-\frac{w_1}{2} & -\frac{w_2}{2} & \cdots & 1+\gamma\mu_n-\frac{w_n}{2} & -\frac{w_1}{2} & \cdots & -\frac{w_{n-1}}{2} & -\frac{w_n}{2} \\
-\frac{w_1}{2} & -\frac{w_2}{2} & \cdots & -\frac{w_n}{2} & 1-\gamma\mu_1-\frac{w_1}{2} & \cdots & -\frac{w_{n-1}}{2} & -\frac{w_n}{2} \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\
-\frac{w_1}{2} & -\frac{w_2}{2} & \cdots & -\frac{w_n}{2} & -\frac{w_1}{2} & \cdots & 1-\gamma\mu_{n-1}-\frac{w_{n-1}}{2} & -\frac{w_n}{2} \\
-\frac{w_1}{2} & -\frac{w_2}{2} & \cdots & -\frac{w_n}{2} & -\frac{w_1}{2} & \cdots & -\frac{w_{n-1}}{2} & 1-\gamma\mu_n-\frac{w_n}{2}
\end{bmatrix}
\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \\ X_{n+1} \\ \vdots \\ X_{2n-1} \\ X_{2n} \end{bmatrix}
= 0
$$

Figure 4.1(a) Energy transfer equations before row subtractions

$$
\begin{bmatrix}
1+\gamma\mu_1 - \frac{w_1}{2} & - \frac{w_2}{2} & \cdots & - \frac{w_n}{2} & - \frac{w_1}{2} & \cdots & - \frac{w_{n-1}}{2} & - \frac{w_n}{2} \\
-(1+\gamma\mu_1) & 1+\gamma\mu_2 & \cdots & 0 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\
0 & 0 & \cdots & 1+\gamma\mu_n & 0 & \cdots & 0 & 0 \\
0 & 0 & \cdots & -(1+\gamma\mu_n) & (1-\gamma\mu_1) & \cdots & 0 & 0 \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & \cdots & 1-\gamma\mu_{n-1} & 0 \\
0 & 0 & \cdots & 0 & 0 & \cdots & -(1-\gamma\mu_{n-1}) & (1-\gamma\mu_n)
\end{bmatrix}
\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \\ X_{n+1} \\ \vdots \\ X_{2n-1} \\ X_{2n} \end{bmatrix}
= 0
$$

FIGURE 4.1(b) Energy transfer equations after row subtractions

corresponding to γ_i can then be expressed, by taking X_1 as an unknown, in the form

$$\underline{X}_i^T = X_1(1, \frac{1+\gamma_i\mu_1}{1+\gamma_i\mu_2}, \ldots \frac{1+\gamma_i\mu_1}{1+\gamma_i\mu_n}, \frac{1+\gamma_i\mu_1}{1-\gamma_i\mu_1}, \ldots \frac{1+\gamma_i\mu_1}{1-\gamma_i\mu_n}$$

or in the simpler form

$$\underline{X}_i^T = \alpha\ (\frac{1}{1+\gamma_i\mu_1}, \frac{1}{1+\gamma_i\mu_2}, \frac{1}{1+\gamma_i\mu_n}, \frac{1}{1-\gamma_i\mu_1}, \ldots \frac{1}{1-\gamma_i\mu_n}) \quad (4.5)$$

where α is some arbitrary constant. All that now remains is to find some means of computing the roots of the polynomial equation $\Delta = 0$.

If we take $Z = \gamma^2$, then Δ is a polynomial of degree n in Z; each solution Z_i to $\Delta = 0$ will give two values of $\gamma = \pm Z_i^{\frac{1}{2}}$. If we put $Z = 0$ in equation (4.3) then it will be seen that $\Delta = 1 - \sum_{i=1}^{n} w_i$; by equation (4.2a) this is zero and hence $\gamma = 0$ is a repeated root. Consider now the sign of the determinant in the range $\frac{1}{\mu^2_{k+1}} < Z < \frac{1}{\mu^2_k}$. From equation (4.3) it will be seen that

$$\Delta = -w_k(1-z\mu_1^2) \ldots (1-z\mu_{k-1}^2)(1-z\mu_{k+1}^2)\ldots(1-z^2\mu_n^2) \text{ at } Z = \frac{1}{\mu_k^2}$$

and $$\Delta = -w_{k+1}(1-z\mu_1^2) \ldots (1-z\mu_k^2)(1-z\mu_{k+2}^2)\ldots(1-z^2\mu_n^2) \text{at } Z = \frac{1}{\mu_{k+1}^2}$$

Therefore, by virtue of equations (4.2b) and (4.2c) it follows that there will be a change in sign of the determinant as Z increases from $\frac{1}{\mu^2_{k+1}}$ to $\frac{1}{\mu^2_k}$. Since the determinant is a continuous function, there will therefore be a root in this range; since there are n-1 such ranges, these account for the remaining roots. A simple dichotomous search for the roots can be performed using the following algorithm for each range.

Step 1 Set $Z_1 = \frac{1}{\mu^2_{k+1}}$ $Z_2 = \frac{1}{\mu^2_k}$

Step 2 Set $Z = (Z_1 + Z_2)/2$

Step 3 If $Z_2-Z_1 < \varepsilon$ then take Z as the root and terminate this search; otherwise ...

Step 4 Compute $f = 1 - \sum_{i=1}^{n} w_i/(1-Z\mu_i^2)$

Step 5 If $f < 0$ then set $Z_1 = Z$ else set $Z_2 = Z$, and continue from Step 2.

It will be seen from equation (4.3) that within any range the sign of the determinant changes as the sign of the quantity f changes. Further, as Z tends to $\frac{1}{\mu^2_{k+1}}$ from above then f tends to minus infinity and as Z tends to $\frac{1}{\mu^2_k}$ from below then f tends to plus infinity. Consequently the test at Step 5 is a valid one for determining which of the two values Z_i and Z_2 currently bracketing the solution is to be replaced by Z. The major effect of rounding error is in the calculation of f; since it is only necessary to determine the sign of f then this should cause no problems even for small values of ε, the accuracy required in the solution. The algorithm will always terminate in a finite number of steps and so there is no need for a loop count. It will be seen that this algorithm has considerable advantages over the original eigenvalue/eigenvector approach; the only storage necessary is for the data vectors $\underline{w}$ and $\underline{\mu}$ plus the solutions γ_i and $\underline{X}_i$ themselves.

4.2.2 Creep Rupture of a Cylindrical Stucture

Materials subjected to stress exhibit the phenomenon of creep, whereby internal damage to the material increases with time, produces a redistribution of the internal stresses and ultimately results in rupture. An accurate estimate of the time to rupture under a given history of loading is essential for the safe design of structures. One technique is to use a finite element [81] model to predict accurately the creep behaviour; this can be expensive in terms of computer time and storage even for simple structures and so less costly methods are sought for use in the early stages of design. One such approach is to derive expressions for the upper and lower bounds of rupture life; these expressions involve integral terms which still require computer evaluation but at significantly less cost in terms of computer resources than the finite element method. Wojewódski, who has reported earlier work with Leckie [82], sought the author's assistance with the computational aspects of applying bounding methods to a cyclically-loaded structure, in which the solution of a nonlinear equation was required.

The relevant features of the model are shown in Figure 4.2. The structure in question was a thick, hollow cylinder of internal radius a and external radius b subjected to an internal pressure p(t) which varied cyclically as a two-level step function of time t. The temperature θ in the cylinder was constant with time, but varied linearly with radius r from θ_a at the inner surface to θ_b at the outer surface. Assuming axial symmetry, the damage ψ due to creep varies with radius and time only. Given the damage distribution at time t, the corresponding distribution of the radial stress component σ_r is given by integration of the differential equation

$$\frac{d\sigma_r}{dr} = \left(\frac{C}{K}\right)^{\frac{1}{n}} \frac{\psi}{r^{\frac{2}{n}+1}} \qquad (4.6)$$

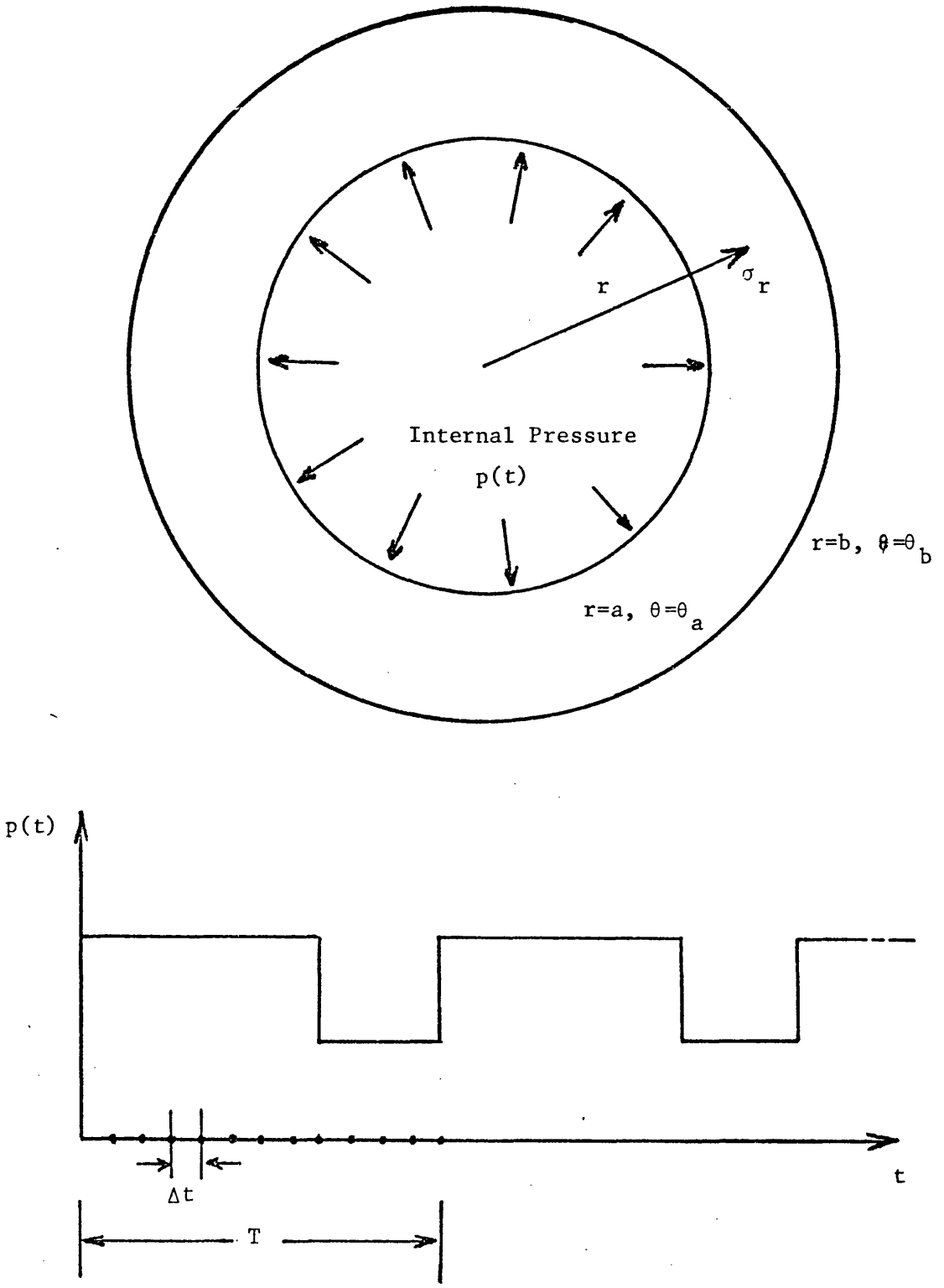


Figure 4.2 Thick Cylinder Subject to Cyclic Loading

subject to the boundary conditions $\sigma_r(a) = -p(t)$ and $\sigma_r(b) = 0$. The material properties K and n vary nonlinearly with θ and the constant C must be determined so as to satisfy the boundary conditions. Once the stress distribution is known, the rate of change of ψ with respect to t can be computed and ψ integrated numerically over a time-step Δt and the whole process can be repeated. The present interest is in the evaluation of C.

The integration of equation (4.6) was carried out using the method of Euler with a constant step-size $\Delta r = (b-a)/m$, where m is the number of steps selected. The radial distribution of σ_r is then given at the discrete values $r = a + k\,\Delta r$ $(k = 0, .. m)$ where

$$\sigma_r(r_k) = \sigma_r(a) + \sum_{i=0}^{k-1} C^{\frac{1}{n_i}} \gamma_i \, \Delta r$$

and

$$\gamma_i = \psi_i / (K_i^{\frac{1}{n_i}} r_i^{\frac{2}{n_i}+1})$$

The suffix i denotes evaluation at $r = a + i\Delta r$. It therefore follows from the boundary conditions that C must satisfy the nonlinear equation

$$-p(t) + \sum_{i=0}^{m-1} C^{\frac{1}{n_i}} \gamma_i \, \Delta r = 0 \qquad (4.7)$$

The time-step Δt was taken as T/12 where T is the period of the cyclic loading; since several thousand cycles could be required to cause rupture, then a fast, but reliable, method for solution of equation (4.7) was essential. Fortunately, the fact that Δt was small compared with the rupture life meant that a good initial estimate of C was always available from the value last calculated at the appropriate stress-level. For the first evaluations of C in the first cycle, initial estimates of C had to be specified. Since in the range of interest n varied monotonically with θ,

then it followed that C must always lie between the two values obtained by assuming a constant temperature of θ_a or θ_b. Taking the average of these two gives

$$C \simeq \tfrac{1}{2}\left[\left(p(t) \Big/ \sum_{i=0}^{m-1} \gamma_i \Delta r \right)^{n(\theta_a)} + \left(p(t) \Big/ \sum_{i=0}^{m-1} \gamma_i \Delta r \right)^{n(\theta_b)} \right]$$

Given that the estimates are good, then Newton's method for nonlinear equations can be used with confidence, and will result in fast convergence. The following algorithm was used for the search on C.

Step 1 Set $C = C_o$ where C_o is the initial estimate as described above; set $k = 1$

Step 2 Compute $\sigma_r(r_m) = -p(t) + \sum_{i=0}^{m-1} C^{\frac{1}{n_i}} \gamma_i \Delta r$

Step 3 Compute $\beta = -\sigma_r(r_m) \Big/ \sum_{i=0}^{m-1} \frac{C^{\frac{1}{n_i}} \gamma_i \Delta r}{n_i}$

where βC is the correction to C predicted by Newton's method to reduce $\sigma_r(r_m)$ to zero.

Step 4 If $|\beta| < 10^{-8}$ then accept the current estimate of C and terminate the search.

Step 5 If $k = 30$ then terminate the search with an appropriate error indicator set.

Step 6 If $\beta < -\frac{1}{2}$ then set $\beta = -\frac{1}{2}$ else
if $\beta > 1$ then set $\beta = 1$

Step 7 Set $C = C + \beta C$; $k = k + 1$
Return to Step 2

For the cases studied, the error exit at Step 5 and the limitation to the corrections at Step 6 were never invoked. Apart from the first evaluation of C at each stress-level, thereafter the algorithm invariably converged in one iteration.

4.2.3 A model of void growth in metals

A mathematical model of the creep rupture by the growth of voids at grain boundaries in polycrystalline metals subject to stress was proposed by Kelly [83]. This case study is taken from the work done by the author on the numerical and computational aspects involved in the computer implementation of the model.

Kelly took a simple two-dimensional representation of the metal, with an assumed regular octagonal grain structure, and he only considered systems of biaxial loading. These assumptions introduce a measure of symmetry to the model and consequently only three different sets of conditions can exist on the separate faces of the grains. On face i of three chosen representative faces, the voids are assumed to be spheres of radius r_i (at time t) and to be regularly-spaced at a distance l_i between centres. The effect of the voids on a face is measured by a damage parameter $\omega_i = \pi r_i^2/4l_i^2$; the model always ensures that $0 < \omega_i < 1$. The normal stress on face i is given by S_i; the effective stress, taking note of damage, is then $S_i/(1-\omega_i)$. Given the values of ω_i, and making assumptions about the effect of shear stress, the values of S_i can be found by solving three linear simultaneous equations, the first two of which are obtained from equilibrium of forces and the third from compatibility of displacements. The growth rate of the voids is then given by the differential equation

$$\frac{d\omega_i}{dt} = A_i \left(\frac{S_i}{1-\omega_i} - \frac{B_i}{\omega_i^{\frac{1}{2}}} \right) \qquad (4.8)$$

where A_i and B_i are independent of time but are functions of the void spacing l_i and material properties. Given the void spacing and the initial damage at time t=0, the model integrates the three equations (4.8) to obtain the variation of damage and stress with time. The integration normally terminates when two faces have ruptured, due to the damage or

effective stress becoming too high, when a cleavage path through the material will exist.

The integration, which has to be done numerically, is done using a fourth-order Runge-Kutta method due to Merson and described by Fox and Mayers [62]. The major difficulty lies in handling discontinuities in the differential equations introduced either by rupture of a face or by equation (4.8) breaking down as ω_i tends to zero. In the first case, rupture of a face is assumed when either $S_i/(1-\omega_i) > \sigma_R$ or $\omega_i > \omega_R$; σ_R is a material property while ω_R is set to a value close to unity, say 0.99. In the second case, if $\omega_i < \omega_L$ then it is assumed that $d\omega_i/dt = 0$; ω_L is taken as 0.1% of the initial damage on the appropriate face. To maintain accuracy in the numerical integration, it is necessary to precisely locate the value of t at which these discontinuities occur. This problem was considered by O'Regan [84]; his method of solution varies according to the Runge-Kutta method being used. In the current instance, a simple line search was used to locate the position of the discontinuities using the well-known method of false position. At the end of any time-step, when an integration will have been performed from t to $t + \Delta t$ say, the damage and stress values are inspected to determine whether a discontinuity occurs within the range of integration. If so, then an objective function f is defined by $\omega_R-\omega_i$, $\omega_i- \omega_L$ or $\sigma_R - S_i/(1-\omega_i)$ as appropriate; note that f changes from a positive to negative value when a time-step crosses a discontinuity. The following algorithm was used to successfully locate the value t_d such that the discontinuity occurs at $t+t_d$.

Step 1 Set $t_l=0$; $t_u=\Delta t$; $f_l=f(t+t_l)$; $f_u=f(t+t_u)$. Set the count k=0.

Step 2 If $f_l-f_u < \varepsilon$ set $t_d=t_u$ and exit; the value of ε is chosen according to the accuracy required.

Step 3 If k = 50 set $t_d=t_u$ and exit; this is a fail-safe limit on the number of iterations.

Step 4 Set $\Delta t=f_1(t_u-t_1)/(f_1-f_u)$ which is given by a linear interpolation between t_1 and t_u to predict $f(t+t_1+\Delta t) = 0$

Step 5 If $\Delta t < 0.1\ (t_u-t_1)$ then set $\Delta t=0.1\ (t_u-t_1)$
If $\Delta t > 0.9\ (t_u-t_1)$ then set $\Delta t=0.9\ (t_u-t_1)$
These tests ensure that a guaranteed minimum rate of convergence is obtained.

Step 6 Evaluate $f=f(t+t_1+\Delta t)$ by integrating from t to $t+t_1+\Delta t$. Adjust the bounds on the discontinuity by setting $t_1=t_1+\Delta t$ and $f_1=f$ if $f > 0$ or by setting $t_u=t_1+\Delta t$ and $f_u=f$ if $f \leq 0$. Return to Step 2.

4.2.4 Discontinuities in the slope of a multimodal function

This case study is taken from work the author carried out on the two-part algorithm discussed in Chapter 2. It is included here, rather than in the earlier chapter, because the length of description required outweighs its importance with regard to the two-part algorithm and it is of more interest as an example of what can be achieved by using a complicated special-purpose search on a particular nonlinear function of a single variable.

It was described in section 2.2.2 how the descent part of the two-part algorithm sets $\underline{x}^{k+1} = \underline{x}^{k} + s\Delta\underline{x}$ where

$$\Delta\underline{x} = Z\,\underline{w} \tag{4.9a}$$

$$\underline{w}^{T} = \left(\frac{1}{\phi_1+\lambda}, \ldots \frac{1}{\phi_n+\lambda}\right) \tag{4.9b}$$

$$s = \min\left(1, \frac{p}{|\Delta x_1|}, \ldots \frac{p}{|\Delta x_n|}\right) \tag{4.9c}$$

The matrix Z depends upon the gradient vector and Hessian matrix of the sum of squares objective function $F(\underline{x})$ evaluated at $\underline{x} = \underline{x}^{k}$; the values ϕ_i ($i = 1, .. n$) are the eigenvalues of the Hessian matrix ordered such that $\phi_i > \phi_{i-1}$; p is a constant which specifies the maximum permitted change $|\Delta x_i^{k+1} - \Delta x_i^{k}|$ in the variables at an iteration and λ is a parameter which is chosen at each iteration to give the global minimum of the function $F(\lambda) = F(\underline{x}^{k} + s\Delta\underline{x})$. It was shown that $F(\lambda)$ is a multimodal function with finite discontinuities at each value $\lambda = -\phi_i$. In addition, there are discontinuities in slope due to the manner in which s is calculated. It will be seen from above that, for a given λ, s adopts the minimum of n+1 discrete values; as λ is varied, the position of this minimum value can shift when a discontinuity in $ds/d\lambda$ will result. If we define

$|\Delta x_{imax}| = \max(\Delta x_1, .. \Delta x_n)$ then such shifts will occur at values of λ where $|\Delta x_{imax}| = p$ or where imax changes when $|\Delta x_{imax}| > p$.

In the original algorithm developed by the author, the search for the global minimum of $F(\lambda)$ was carried out by considering in turn the regions of λ in which $F(\lambda)$ is continuous; thus the first region is $-\infty < \lambda \leq -\phi_n$, the second $-\phi_n \leq \lambda \leq -\phi_{n-1}$ and so on. A grid search was employed to bracket minima within each region and then the position of any minimum was located more accurately by an interpolation. Dowson [66] improved the reliability of the grid searches by doing a further preliminary subdivision into regions in which both $F(\lambda)$ and $dF(\lambda)/d\lambda$ are continuous. This entailed using a numerical search to locate values of λ corresponding to "changes of state" of the corrections; Dowson used this expression to describe the situations producing a discontinuity in $ds/d\lambda$ just described. The state of the corrections can be quantified by the integer ψ where $\psi = 1$ if $|\Delta x_{imax}| < p$ and $\psi = imax + 1$ otherwise; note that ψ then corresponds to the position of the minimum in the right-hand side of equation (4.9c). Dowson's search for the positions of the changes of state can then be explained, as follows, for all regions $-\phi_i \leq \lambda \leq -\phi_{i-1}$ (i=2, .. n)

Step 1 Set $\Delta\lambda = (\phi_i - \phi_{i-1})/4$; $\lambda_1 = -\phi_i$; $\lambda_2 = \lambda_1 + \Delta\lambda$

Step 2 Set ψ_1 equal to ψ at $\lambda = \lambda_1$

Step 3 Set ψ_2 equal to ψ at $\lambda = \lambda_2$

Step 4 If $\psi_2 = \psi_1$ then continue from step 8

Step 5 Using a dichotomous search, find to within a prescribed tolerance the values λ_c and $\psi_c = \psi$ at $\lambda = \lambda_c$ such that $\lambda_1 < \lambda_c < \lambda_2$; $\psi = \psi_1$ for $\lambda < \lambda_c$; and $\psi_c \neq \psi_1$

Step 6 If $\psi_c = \psi_2$ then continue from step 8, having recorded the position of a discontinuity at $\lambda = \lambda_c$. Otherwise ...

Step 7 Set $\lambda_1 = \lambda_c$; $\psi_1 = \psi_c$. Return to Step 4.

Step 8 If $\lambda_2 < -\phi_{i-1}$ then set $\lambda_1 = \lambda_2$; $\psi_1 = \psi_2$; $\lambda_2 = \min(-\phi_{i-1}, \lambda_2 + \Delta\lambda)$ and return to Step 3. Otherwise, the search in the current region for discontinuities has been completed.

A similar scheme is used in the two end-regions where $\lambda \leq -\phi_n$ and $\lambda \geq -\phi_1$. Dowson acknowledged that this search does not give a guarantee of finding all the slope discontinuities. Consequently, the author decided to try to develop a method with this guarantee to find what improvements (if any) could then be obtained in the grid search. It will be seen from equations (4.9a) and (4.9b) that each correction Δx_i can be expressed as a rational polynomial in λ of the form

$$\Delta x_i = \frac{\sum_{k=1}^{n} A_{ik} \lambda^{k-1}}{\sum_{k=1}^{n+1} B_k \lambda^{k-1}} \qquad (4.10)$$

The coefficients B_k are the same for each correction and are obtained from the product $(\phi_1 + \lambda)(\phi_2 + \lambda) \ldots (\phi_n + \lambda)$; the coefficients A_{ik} vary between corrections and are obtained from the summations $\sum_{j=1}^{n} Z_{ij} (\phi_1 + \lambda)(\phi_2 + \lambda) \ldots$ $\ldots(\phi_{j-1} + \lambda)(\phi_{j+1} + \lambda) \ldots (\phi_n + \lambda)$. The new method was based on this rational polynomial form of the corrections; it was able to guarantee that all discontinuities in $ds/d\lambda$ were located by using the fact that the number of roots of a general polynomial equation is known to equal the degree of the polynomial. It will be noted that, in general, the values of the roots can

only be found numerically.

In the first stage of the calculation, the range of real λ is divided into N+1 adjacent regions such that within each region the index imax of $|\Delta x_{imax}|$ remains constant; it will be apparent that N is initially unknown. Let the kth such region be given by $\alpha_k < \lambda < \alpha_{k+1}$ in which imax = I_k; assume that $\alpha_o = -\infty$ and $\alpha_{N+1} = +\infty$. It will be noted that if two or more corrections are equal over a finite range, then the columns of Z corresponding to each correction must necessarily be equal. The method which is to be described could be modified to account for equal corrections by testing for these at the outset and thereafter only considering one representative correction from each group of equal corrections. The values of α_k and I_k are found by comparing the relative magnitudes of all possible pairs of corrections in the range of real λ as follows.

Consider any two corrections Δx_i and Δx_j. Since the denominators of the appropriate rational polynomials given by equation (4.10) will be the same, only the numerators need be considered when comparing the magnitudes of Δx_i and Δx_j; the fact that the denominator becomes zero at $\lambda = -\phi_i$ does not invalidate this argument. For the end-regions, when λ tends to plus or minus infinity, the greater of the two corrections will correspond to the greater of the two values $|A_{in}|$ and $|A_{jn}|$, since these are the coefficients of the dominant terms in the numerator. If $A_{in} = A_{jn} = 0$ then lower-order terms in the numerator need to be considered. At any values of λ for which $|\Delta x_i| = |\Delta x_j|$ then the relative magnitudes of the two corrections will change; in this event the range of real λ will divide up into a number (always odd) of regions such that throughout each region one correction remains greater than the other. Such values of λ are found from the complete solution of

$$\sum_{k=1}^{n} (A_{ik} - A_{jk}) \lambda^{k-1} = 0 \qquad (4.11a)$$

$$\sum_{k=1}^{n} (A_{ik} + A_{jk}) \lambda^{k-1} = 0 \qquad (4.11b)$$

Each of the two equations has n-1 roots; complex roots must be discarded. If n is even, there is a guarantee of at least one real root to each equation.

Assume that in total there are M real and distinct roots $r_l (l = 1, .. M)$ to the equations; assume also that $r_l > r_{l-1}$. Further, define $r_o = -\infty$ and $r_{M+1} = +\infty$, noting that equation (4.10) shows that both corrections are zero at $\lambda = r_o$ and $\lambda = r_{M+1}$. We then have M + 1 regions of the form $r_l < \lambda < r_{l+1}$ $(l = 0, .. M)$; within each region the continuity of the numerators ensures that whichever is the greater correction (whose index is denoted by J_1) at the start of the region remains the greater throughout the region. It has just been shown how J_1 can be evaluated at r_o and r_{M+1}. For inner regions $r_l < \lambda < r_{l+1}$ $(l = 1, .. M-1)$ J_1 can be calculated by considering the relative magnitudes of $d|\Delta x_i|/d\lambda$ and $d|\Delta x_j|/d\lambda$ at $\lambda = r_1$. Note that $d|\Delta x_i|/d\lambda = -d\Delta x_i/d\lambda$ when $\Delta x_i < 0$; $d|\Delta x_i|/d\lambda = |d\Delta x_i/d\lambda|$ when $\Delta x_i = 0$; and $d|\Delta x_i|/d\lambda = d\Delta x_i/d\lambda$ when $\Delta x_i > 0$. Also, for comparison purposes, the denominator of equation (4.10) can be taken as unity giving $d\Delta x_i/d\lambda = \sum_{k=2}^{n} (k-1) A_{ik} r_1^{k-2}$ at $\lambda = r_1$. Then J_1 is set to i if $d|\Delta x_i|/d\lambda > d|\Delta x_j|/d\lambda$; otherwise J_1 is set to j. An alternative, and much simpler method of determining J_1 (which occurred to the author at a later date) would be to compare the relative magnitudes of the corrections evaluated at an interior point of the region, say at $\lambda = (r_l + r_{l+1})/2$.

The accumulation of the results of all the comparisons between pairs of corrections is simple in principle, albeit complicated in practice. The steps can be summarised as follows.

Step 1 Set i = 1

Step 2 Set j = i + 1

Step 3 Compare $|\Delta x_i|$ and $|\Delta x_j|$ to obtain M, r_1 and J_1 (1=0, .. M)

as already discussed

Step 4 If $i = 1$ and $j = 2$ then initialize cumulative information by setting $N=M$, $\alpha_1 = r_1$ and $I_1=J_1$ $(1 = 0, .. M)$ and continue from step 6. Otherwise ...

Step 5 add results of comparison to cumulative total. For each 1 for which $J_1=j$ check whether the region $r_1 < \lambda < r_{1+1}$ overlaps with any region $\alpha_k < \lambda < \alpha_{k+1}$ in which $I_k=i$. In the event of an overlap, since it is known that $|\Delta x_j| > |\Delta x_i|$ in region 1 then in the overlap the α_k, I_k information must be amended in one of the four ways shown in Figure 4.3.

Step 6 If $j < n$ then set $j = j + 1$ and continue from step 3

Step 7 If $i < n-1$ then set $i = i+1$ and continue from step 2; otherwise the process is complete.

There are a total of $n(n-1)/2$ combinations of two corrections; the loop structure of the above algorithm is such that each combination is considered once only. At step 5, it will be noticed that only those regions 1 for which $J_1 = j$ (i.e. $|\Delta x_j| > |\Delta x_i|$) are considered for incorporation into the running total. On completion of step 5, for each region k it will be seen that

$$|\Delta x_{I_k}| \geq |\Delta x_1| \qquad (1=1, \ldots i) \quad \text{if } I_k \leq i$$

$$|\Delta x_{I_k}| \geq |\Delta x_1| \qquad (1=1, \ldots j) \quad \text{if } I_k > i$$

On completion of the last execution of step 5, when we have $i = n-1$ and $j=n$ then it will follow that for all the regions

$$|\Delta x_{I_k}| \geq |\Delta x_1| \qquad (1=1, \ldots n)$$

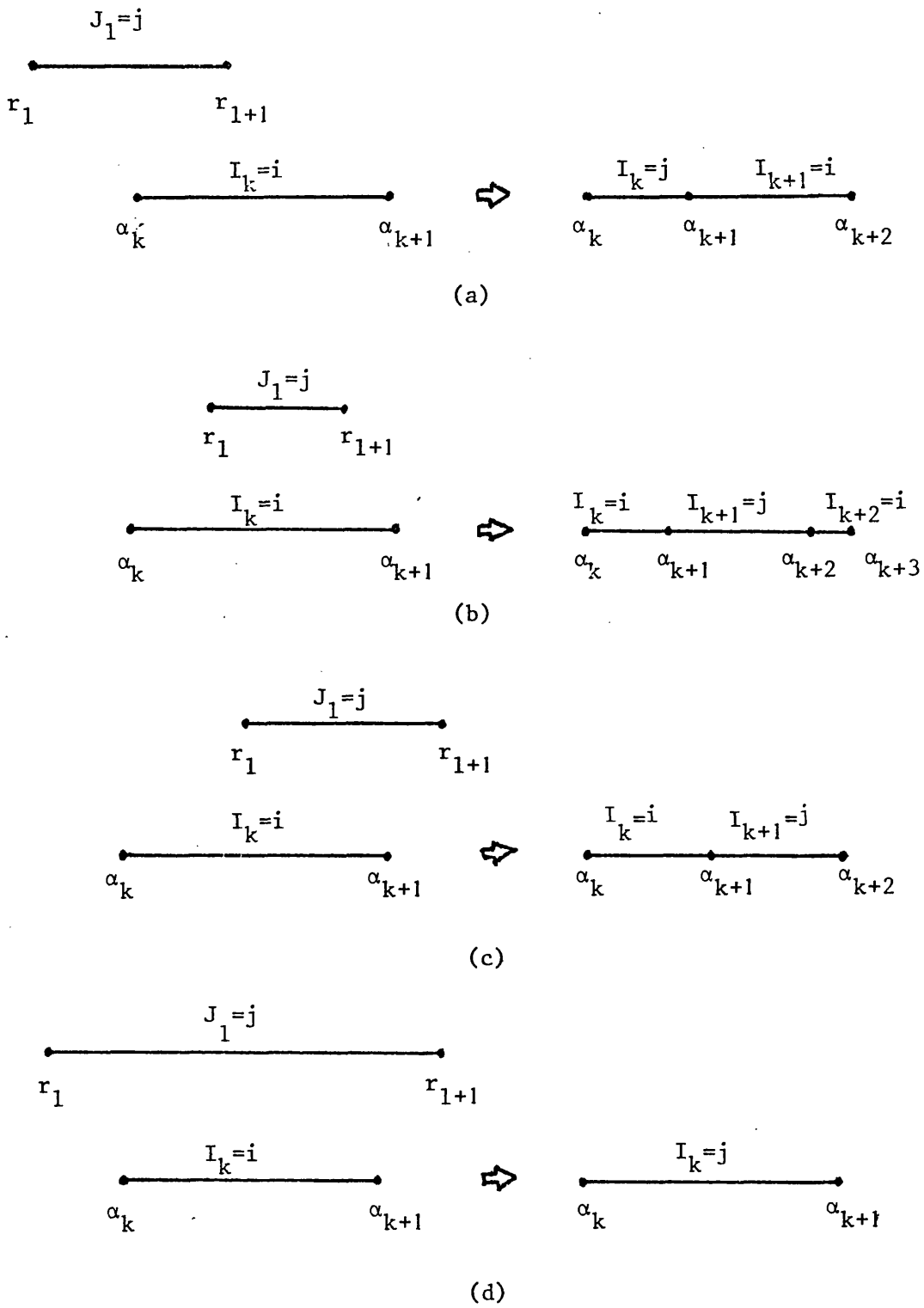


Figure 4.3 Updating cumulative information on maximum magnitude corrections

The way in which this accumulation operates is illustrated in Figure 4.4. for a hypothetical case when n=4. There are twelve distinct values of λ at which two corrections are equal in magnitude; these values are numbered across the top of the figure and, for convenience, are shown equally-spaced. Of the six comparisons, only three (steps 2, 4 and 8 in the figure) affect the accumulated information. Note how at step 8 the range $\alpha_2 < \lambda < \alpha_3$ is extended.

In the second stage of the method, information is added to that obtained in the first stage specifying the regions in which $|\Delta x_{I_k}| < p$ and $|\Delta x_{I_k}| \geq p$. This, in general, will require a further subdivision and a corresponding increase in N. It is only necessary to consider those corrections which, for at least part of the range of real λ, are greatest in magnitude; thus in the example of Figure 4.4 correction 4 would not be considered. The values of λ at which $|\Delta x_i| = p$ are given as the roots of the equations

$$\sum_{k=1}^{n} A_{ik} \lambda^{k-1} = p \sum_{k=1}^{n+1} B_k \lambda^{k-1} \qquad (4.12a)$$

$$\sum_{k=1}^{n} A_{ik} \lambda^{k-1} = -p \sum_{k=1}^{n+1} B_k \lambda^{k-1} \qquad (4.12b)$$

Each equation has n roots; complex roots must be discarded. If n is odd then there is at least one real root to each equation. Assume that in total there are M real distinct roots r_l (l=1, .. M) and ordered such that $r_l > r_{l-1}$. In the region $r_l < \lambda < r_{l+1}$ (l=0, .. M) it follows that $|\Delta x_i|$ will either be above or below p throughout the whole of the region; it is assumed that $r_o = -\infty$ and $r_{M+1} = +\infty$. The test for whether a correction is above or below p between consecutive roots is based on considerations of the slope of $|\Delta x_i|$ at the lower root; it could equally-well be done by evaluating $|\Delta x_i|$ at some

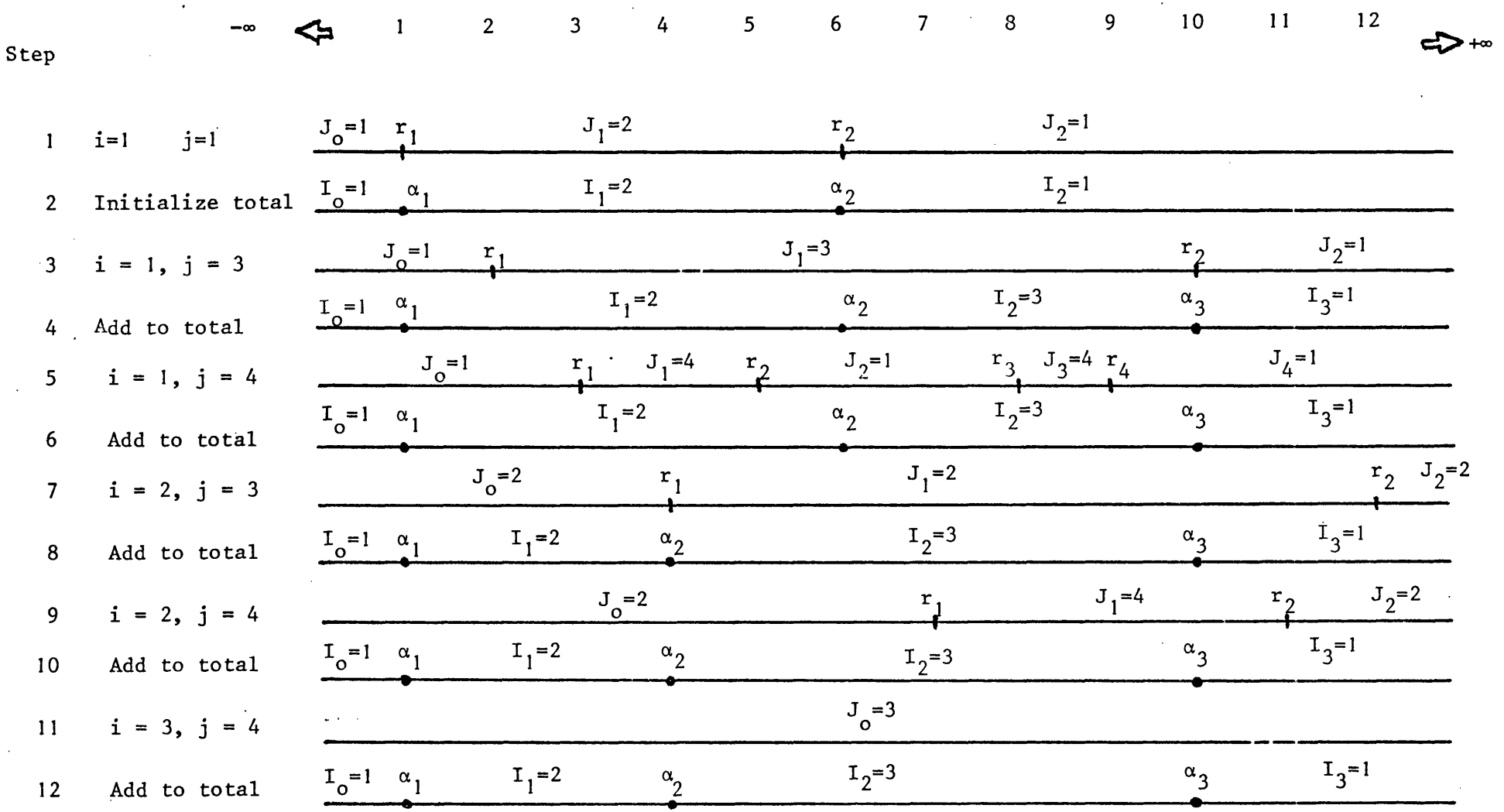


Figure 4.4 Accumulation of total information on relative magnitudes of the corrections

point between the roots. Using this information, the requisite amendments can be made to the regions $\alpha_k < \lambda < \alpha_{k+1}$ in which $I_k=i$; an additional integer value π_k is stored which is set to 1 if $|\Delta x_i| < p$ and to 2 if $|\Delta x_i| \geq p$. Note that a region can be subdivided into two or more regions with different values of π_k.

The third and final stage is the addition of the points $\lambda = -\phi_i$ (i=1, .. n) at which discontinuities in $F(\lambda)$ occur. Such points will correspond to singularities in the corrections and must always be in regions for which $\pi_k = 2$; the information is incorporated by introducing a further n values of α_k with $\pi_k = i + 2$. The final regions are then summarised by the integer N and the set of values α_k, I_k, π_k (k=0, .. N) where, for each k, we have a region $\alpha_k < \lambda < \alpha_{k+1}$ in which the correction of maximum magnitude is $|\Delta x_{I_k}|$ and

$$|\Delta x_{I_k}| < p \qquad \text{if } \pi_k = 1$$

$$|\Delta x_{I_k}| \geq p \qquad \text{if } \pi_k \geq 2$$

and there is a singularity in $F(\lambda)$ at $\lambda = \alpha_k$ if $\pi_k > 2$.

The method was successfully implemented as a computer program. The major programming difficulty was the complex data structure caused by the need to insert new regions or merge existing regions. This was done using a system of pointers using FORTRAN; other languages would be more convenient, for example ALGOL W which has a suitable data structure. With the pointer system, the values of α_k, I_k and π_k were stored in arrays while a fourth array held pointers giving the order in which the α_k, I_k and π_k values were stored in their arrays, being not necessarily in ascending order. The solution of the polynomial equations was carried out using a standard NAG [70] Library routine using the method of Grant and Hitchins [85]. A major difficulty was allowing for rounding error when distinguishing

real from imaginary roots; the NAG routine returns roots in the form (a,b) where a and b are the real and imaginary parts. A root was taken as real if $|b| < 10^{-5} (1 + |a|)$. To prevent overflows in the evaluation of the coefficients A_{ik} and B_k the matrix Z and the eigenvalues ϕ_i were both divided by the value of the maximum magnitude eigenvalue. Since rounding errors could be introduced both in the evaluation of the coefficients and in the solution of the equations, a check was made: when values of λ were calculated for which $|\Delta x_i| = |\Delta x_j|$, the corrections were evaluated by using the original equation (4.9a); if they did not agree to within a given relative tolerance of 10^{-5} then Newton's method was invoked to adjust λ suitably. A similar process was applied when finding λ such that $|\Delta x_i| = p$.

Tests were carried out using data for Rosenbrock's function and the eight exponential equations shown in the Appendix. A third test was done on a problem of six equations with randomly-generated Z and ϕ_i values. In all cases, it was never found necessary to use Newton's method to refine the roots; this indicated that rounding-errors were not a problem (double-precision arithmetic was being used). Further, the results were compared with those obtained using Dowson's method. Apart from minor differences in values due to rounding-errors, the results obtained by both methods were identical. In addition to confirming that the new method had been programmed correctly, this showed that, for the test cases and most probably for other problems with a moderate number of variables, Dowson's search is extremely reliable in locating the discontinuities in slope. Since it requires an order of magnitude less processor time, and considerably-less storage for code, then there was considered to be no advantage in using the new method.

4.3 Problems with two variables

Optimization problems involving two variables are a special case in so far as it is possible, on a two-dimensional graph, to plot the contours of the objective function. Many computer installations have packages available to do this. Consequently, if an attempt at optimizing a function fails, then all that is usually necessary is to plot the contours of the function in the region of interest. If there are any constraints, these too can be shown on the plot. An inspection of the contours will then yield a sufficiently-close estimate of the position of the minimum for the optimization method to converge when started from this estimate.

Thus in most instances, the main concern with two-dimensional problems will be with the efficiency rather than the reliability of the method used for their solution. It will often be possible to develop fast *ad hoc* methods of solution. The following three case studies illustrate these points. The first and last cases are of problems requiring solution many times, with different sets of data, in a single computer run. The second case describes an algorithm for finding to a high resolution the region of stability of an electrical machine.

4.3.1 A modification to a finite-element method

The following nonlinear least squares problem is taken from work Hayhurst and Henderson [86] carried out in which the finite element method was used to predict the effect of stress redistribution due to creep in a notched cylindrical bar. In order that the nature of the problem can be explained it is necessary to use some of the concepts involved in the finite element method; these are introduced without further explanation, a good text on the subject being that by Zienkiewicz [81].

The model is shown in Figure 4.5; axial symmetry could be assumed because the bar was of circular cross-section, the notch was cut circumferentially and the load was applied axially. Further, since the two halves of the bar obtained by cutting through the notch were symmetric, the analysis was restricted to one half only. A triangular finite element, described in Chapter 4 of Zienkiewicz's book [81] was used and the half bar subdivided by a mesh made up of these elements. Note that each element represents a solid of revolution about the axis.

Considering a single element, the nodes are numbered i, j and m in anti-clockwise order. For node i, the radial and axial coordinates are denoted by r_i and z_i respectively; the applied nodal forces are given by X_i and Y_i; and the resulting elastic displacements are u_i and v_i. Similar terms for the other nodes are obtained by changing the subscripts. The nodal displacements are related to the applied forces by the equation

$$\underline{F} = k\,\underline{\delta} \qquad (4.13)$$

where $\underline{F}^T = (X_i, Y_i, X_j, Y_j, X_m, Y_m)$, $\underline{\delta}^T = (u_i, v_i, u_j, v_j, u_m, v_m)$ and k is a 6 x 6 symmetric matrix known as the element stiffness matrix. The values for k are obtained from the double-integral expression

$$k = 2\pi \iint B^T D\, B\, r\, dr\, dz \qquad (4.14)$$

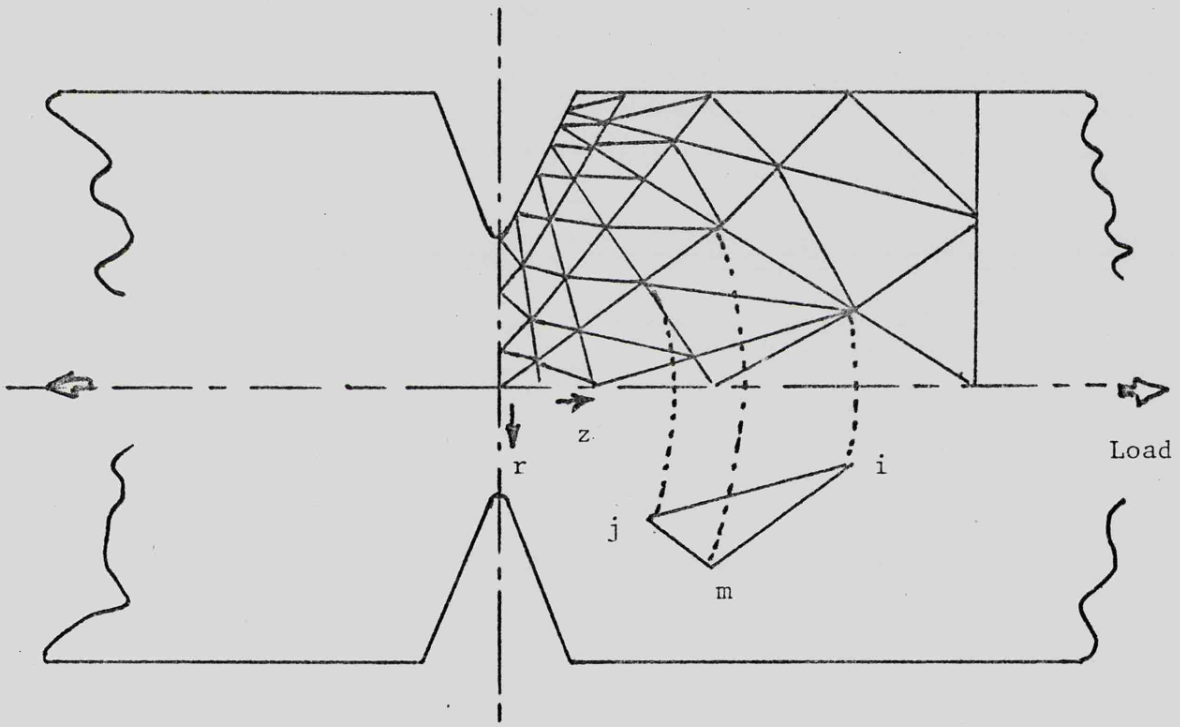


Figure 4.5 Axisymmetric Finite Element and Notched Bar

The 4 x 4 matrix D is the elasticity matrix and depends upon the material properties; the 4 x 6 matrix B depends solely upon the element geometry. If B is partitioned into three submatrices such that $B = \left[B_i, B_j, B_m \right]$ then we can write

$$B_i = \begin{bmatrix} 0 & c_i \\ b_i & 0 \\ \frac{a_i}{r} + b_i + c_i \frac{z}{r} & 0 \\ c_i & b_i \end{bmatrix} \qquad (4.15)$$

with $a_i = r_j z_m - r_m z_j$, $b_i = z_j - z_m$ and $c_i = r_m - r_j$. Similar expression for B_j and B_m can be derived by cyclic rotation of the subscripts.

When all the equations (4.13) for each element are aggregated over the mesh, then a single set of equations is obtained relating the externally-applied nodal forces to the nodal displacements (the internal nodal forces sum to zero). For each node, either the applied force or the node displacement is given by boundary conditions and so the equations can be solved to give the unknown forces and displacements. Once the displacements are known for all the nodes of an element, the elastic strains and stresses can be derived. The rate of increase of strain due to creep is then derived from a supplied constitutive relationship between strain rate and stress. By numerically integrating the strain over a time-step, the effect of creep on the elastic strain distribution, and hence stress, can be found. The integration is then repeated until rupture of the bar occurs.

Initial computer runs showed that as the integration progressed the original equilibrium between applied load and internal stresses was not maintained. It was thought that one cause for this might be the use of a representative stress for each element in the strain rate calculation. The four stress components $\underline{\sigma}$ in an element are related to the nodal displacements by the relation $\underline{\sigma} = DB\underline{\delta}$. It will be seen from equation (4.15) that B, and

hence $\underline{\sigma}$, varies with r and z. The representative stress used was that of the centroid of the element $(\bar{r}, \bar{z})$. Hayhurst suggested that a better choice might be the point (r*, z*) chosen so that if B is evaluated at (r*, z*) to give B*, then the value k* for k, obtained by replacing B by B* in the double-integral of equation (4.14), is not far removed from the true value of k. A least squares objective function can then be defined by

$$f(r^*, z^*) = \sum_{i=1}^{6} \sum_{j=i}^{6} (k^*_{ij} - k_{ij})^2 \qquad (4.16)$$

Since k is symmetric, the summation is only taken over the upper triangle; strictly speaking a weight factor of two could be applied to the off-diagonal terms. Although there are 21 residuals, six values of k_{ij} are independent of the (r,z) terms in the B matrix expression given in equation (4.15); consequently the problem reduces to one of 15 residuals and 2 independent variables. It then remains to find optimum values for (r*, z*).

The Gauss-Newton method was used for the optimization, including a line search similar to that used in the Gauss-Newton section of the two-part algorithm described in Chapter 2. The initial guess for the variables was taken as $(\bar{r}, \bar{z})$, the centroid. At each iteration, the maximum permitted changes in the variables were $\pm$ 0.5r* and $\pm$ 0.5z*. Note that since only two variables were present, the evaluation of the Gauss-Newton corrections, which requires the solution of linear simultaneous equations, was readily done algebraically. The search was programmed in FORTRAN IV on a DEC PDP-11/20 minicomputer and, when tested, transferred to an IBM 370/195 for incorporation in the main finite element program.

The effect on k* of optimizing (r*, z*) is shown in Table 4.1. The actual values of k_{ij} are shown together with the k^*_{ij} values obtained using the centroid and the optimum points. The six values of k_{ij} which are independent of the terms in equation (4.15) for B involving r and z are

i	j	k_{ij} (exact)	k^*_{ij} (centroid)	k^*_{ij} (optimum)
1	1	44.3	23.2	27.2
1	2	0.00	0.00	0.00
1	3	-21.2	-21.2	-12.8
1	4	6.04	6.04	2.26
1	5	1.01	1.01	5.30
1	6	-6.04	-6.04	-2.26
2	3	8.06	8.06	8.06
2	5	-8.06	-8.06	-8.06
3	3	57.7	55.4	55.9
3	4	-26.2	-26.2	-26.3
3	5	2.69	5.04	4.90
3	6	18.1	18.1	18.3
4	5	2.01	2.01	2.14
5	5	17.5	15.1	14.8
5	6	6.04	6.04	5.91

N.B. Nodal coordinates of element are (0,0), (10,0) and (10,10)

At centroid, (6.667, 3.333), objective function = 464

At optimum (5.514, 2.699), objective function = 424

Table 4.1 Optimization of (r*, z*) for one finite element

omitted. It will be seen that a slight improvement in the objective function of equation (4.16) is obtained. However, this is unusually large; most of the elements in the finite element meshes used show a less marked reduction on optimization. Consequently, no significant changes in the results of the creep integration were found when the optimum, rather than centroid, points were used to calculate representative stresses. Eventually, the numerical difficulty was surmounted by using an improved mesh, with finer elements in the vicinity of the notch.

4.3.2. Stability of an electrical machine

This two-dimensional search is part of a contouring program developed by the author. The program produces, on a digital plotter, a graph showing the region of stability of a reluctance-synchronous machine over a range of operating conditions. The mathematical basis of the program is due to Lipo and Krause [87] who used small displacement theory to apply Nyquist's criterion to small perturbations about a steady-state operating point. Only the results of the analysis done by Lipo and Krause which affect the program are given here, and then without further explanation.

The design of a particular machine is specified by the values of a set of machine parameters. Given the voltage and frequency, a steady-state operating-point can be specified uniquely by two values - the slip f_R and the torque T. The search strategy assumes that the region of stability in the f_R-T plane is bounded by a closed, convex curve such as that shown in Figure 4.6. The dotted lines in the figure are the maximum and minimum steady-state torques which, for any given value of f_R, are obtained by a straight forward calculation. Using Nyquist's criterion, it can be shown that the operating point (f_R, T) will lie on the bounding curve of the region of stability if the locus of the complex function $G(j\nu)$ passes through (-1, 0). This locus is obtained by varying the scalar ν, where j denotes the complex operator $j^2 = -1$. For any given ν, $G(j\nu)$ is a function of f_R and T, as well as the fixed machine parameters.

If we write $G(j\nu) = a(\nu) + jb(\nu)$ then the analysis shows that $a(-\nu) = a(\nu)$ and $b(-\nu) = -b(\nu)$. Consequently, in any search on ν for given f_R and T to locate a point (if any) for which $G(j\nu) = (-1, 0)$ it is only necessary to consider positive values of ν. If ν^* is the value of ν for which $a(\nu^*) = -1$, then the value $b(\nu^*)$ indicates how close to the bounding curve the values of f_R and T are. The program approximates the bounding curve by locating a finite number of points (f_R, T) around the curve, at

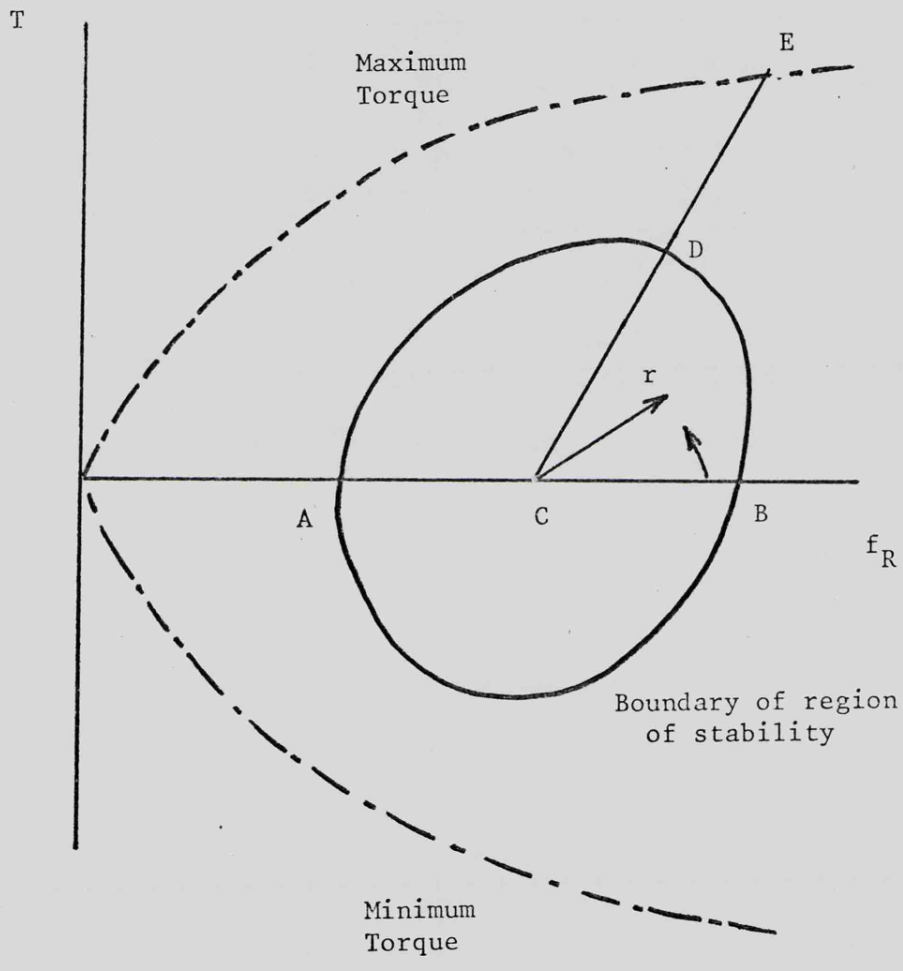


Figure 4.6 Search for the boundary of the region of stability

roughly-equal spacings. This is done by searching along lines (such as CE in Figure 4.6) whose origin is at a point near to the middle of the region of stability and finding their points of intersection with the bounding curve. If r is the distance along such a line, the intersection will be at $r = r^*$ for which the corresponding value $b(\nu^*)$ is zero. Thus the search is two-dimensional with variables ν and r. Two nested one-dimensional searches were used since, in addition to being reliable, they could readily cope with the physical constraints on ν and r.

The inner search is on ν to locate ν^*, given f_R and T. It is assumed that as ν increases from zero $a(\nu)$ increases from values below -1 to values above -1; this assumption has not been invalidated by any of the cases so far studied (if it was, the search would terminate with an error message). A starting value for ν is taken as the value for ν^* last found (or unity if this is the very first search on ν). If $a(\nu)$ is greater than -1, ν is multiplied by 0.1 repeatedly until $a(\nu) < -1$. Then ν is doubled repeatedly until $a(\nu) \geq -1$; it is then known that $\nu/2 < \nu^* \leq \nu$. This bracket is reduced using a dichotomous search until at the midpoint of the bracket $|1+a(\nu)| < \varepsilon_1$ is satisfied; ε_1 is the required accuracy of value 10^{-4} in the present case. A fail-safe exit is taken if the bracket is reduced to a very small amount without the accuracy criterion being satisfied; this would happen if rounding-error effects are larger than the chosen value of ε_1. Similar fail-safe criteria are built into the other searches described later. On exit from the search, ν^* is taken as the midpoint of the range; $b(\nu^*)$ is evaluated for use in the outer search.

A preliminary calculation is carried out to set up a system of lines whose intersections with the bounding curve are located by the outer search. First, the two intersections of the curve with the f_R-axis, A and B in Figure 4.6, are found. Starting with f_R=0.01, the value of f_R is incremented in steps of 0.01 and $b(\nu^*)$ calculated corresponding to f_R with

T=0. If the signs of $b(\nu^*)$ on two successive f_R values are opposite, then a bracket has been obtained enclosing one of the points A and B. This bracket is reduced by a dichotomous search until $|b(\nu^*)| < \varepsilon_2$, where the required accuracy ε_2 was taken as 10^{-5} in the present case. A polar coordinate system (r, θ) is then set up with its origin at C, the midpoint of the line AB. Lines are generated by taking values of θ at intervals of one degree in the range 0 to 360 degrees. For each value of θ, the corresponding line is given by the following parametric functions of r:

$$f_R = \tfrac{1}{2}\left[(f_R^B + f_R^A) + (f_R^B - f_R^A)\, r \cos\theta\right]. \qquad (4.17a)$$

$$T = \tfrac{1}{4}\,(TMAX - TMIN)\, r \sin\theta \qquad (4.17b)$$

In these expressions f_R^A and f_R^B are the values of f_R at A and B, while TMAX and TMIN are the maximum and minimum torques for the value of slip f_R at C. This choice of scale is made so that equal intervals of θ will correspond, roughly, to equal distances along the arc of the curve between successive intersections. It also ensures that the values of r* are in the order of unity.

Full details of the search for r* will not be given here since the process is straight forward. An initial estimate of r* is taken as 0.8 times the value of r* found for the previous θ; note that r*=1 when θ = 0 or 180. This estimate is multiplied by 1.5 repeatedly until either the sign of $b(\nu^*)$ changes, when the intersection D with the bounding curve will be bracketed, or else the upper limit of r is reached at E (which is itself located by a further search) without a change of sign. In the latter event, the search returns to the original estimate of r* and halves it repeatedly until $b(\nu^*)$ changes sign. A dichotomous search, identical to that used to locate A and B, is employed to locate the intersection accurately.

The program was written in FORTRAN IV for a CDC Cyber 72; the digital plotting was done with the aid of a standard package. Figure 4.7 shows the typical output of the program. Rough estimates show that if a standard method of contouring were used, in which values of a function are obtained by interpolation between values at specified points on a grid, then considerably more computation would be necessary. A similar technique for producing contours, using an (r, θ) coordinate system, was provided by the author for Dinibütün and Corbett [88] who used it successfully to calculate contours for two control variables corresponding to small, specified departures from an optimum operating point. In this case however, the plotting was done by hand using the coordinates of the contours as output from the program used by Dinibütün and Corbett.

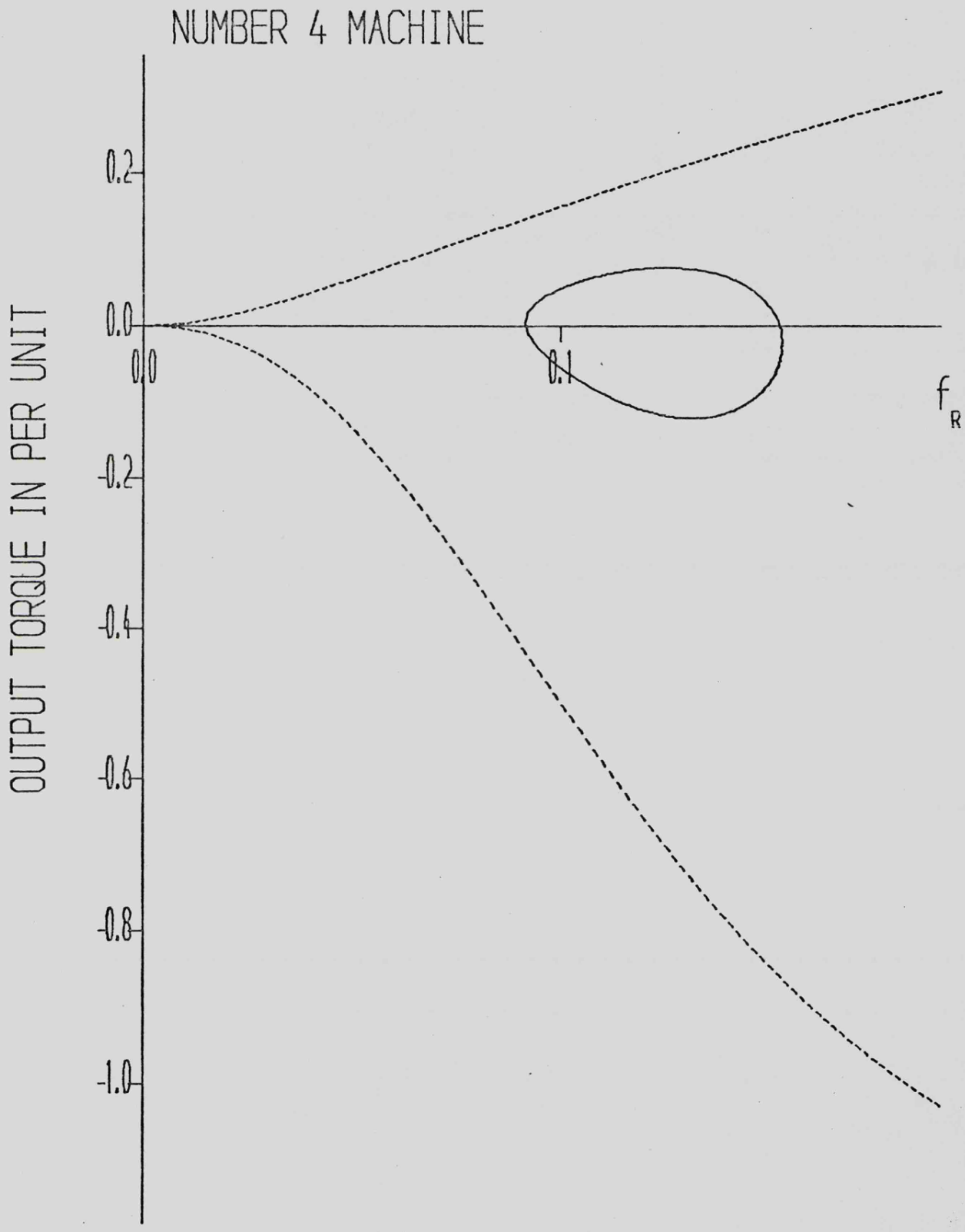


Figure 4.7 Sample output from the contouring program

4.3.3 The Michaelis-Menten Equation

In biochemistry, the velocity y of a chemical reaction catalyzed by an enzyme, present in concentration x, is given by the Michaelis-Menten equation which states that [89]

$$y = \frac{Vx}{K+x} \qquad (4.18)$$

The constant V is the maximum velocity, which y will approach as x tends to infinity; K is the Michaelis constant and is equal to that concentration giving half the maximum velocity. To determine the values of V and K for a specified reaction, experiments are first carried out in which the velocities y_i are measured at different values of concentration x_i (i=1,..m). The best estimate of V and K is then taken as that which gives the closest agreement, in a defined manner, between the measured values of y_i and those predicted by equation (4.18) at the m experimental points. A least squares measure requires that the minimum be found of the function

$$S_1 (V,K) = \sum_{i=1}^{m} \left(y_i - \frac{Vx_i}{K+x_i}\right)^2 \qquad (4.19)$$

The standard statistical technique of linear regression unfortunately cannot be used since the residuals in equation (4.19) are nonlinear functions of K. To avoid the need to use a nonlinear iterative minimization, other measures of fit have been proposed. The most widely-used is the Lineweaver-Burk plot which fits a straight line to the points $(1/x_i, 1/y_i)$. If equation (4.18) is re-arranged we can write

$$\frac{1}{y} = a + \frac{b}{x} \qquad \text{where } a = \frac{1}{V}, \; b = \frac{K}{V} \qquad (4.20)$$

The values of a and b which minimize

$$S_2(a,b) = \sum_{i=1}^{m} \left(\frac{1}{y_i} - a - \frac{b}{x_i}\right)^2 \qquad (4.21)$$

are found, in the standard manner for linear regression, by solving the two linear equations corresponding to $\partial S_2/\partial a = 0$ and $\partial S_2/\partial b = 0$. The

corresponding values for V and K at the minimum are then found directly from $V = 1/a$ and $K = b/a$.

If the observations fit the Michaelis-Menten equation exactly, then the nonlinear minimization of equation (4.19) will yield the same values for V and K as that those using the linear formulation of equation (4.21). In practice, there will be experimental errors in the measurements of y_i and x_i (and possibly inaccuracies due to the model itself). The effect of these errors will not be the same for the two methods and so different values of V and K will be obtained. Cormack and Lamb [90] wished to investigate the effect of these experimental errors for the y_i values; comparisons were required of the results produced by the nonlinear least squares fit, the Lineweaver-Burk plot, a further three linear methods and a second nonlinear least squares method using the measure

$$S_3(V,K) = \sum_{i=1}^{m} (\log_e y_i - \log_e \frac{Vx_i}{K+x_i})^2 \qquad (4.22)$$

The approach of Colquoun [91] was followed in which a set of experiments was simulated using a computer. For each experiment, it was assumed that the true values were V=30 and K=6; the choice of values is unimportant since they are effectively scaling-factors with a linear effect on all the results obtained. In each experiment m values of x_i are specified and experimental values of y_i are simulated by taking $y_i=Vx_i/(K+x_i)+\varepsilon_i$ where ε_i is an error in y_i chosen at random from a given population. Three experiments were considered, with m=3, 5 and 6; six different possible error distributions were used. For each of the eighteen possible combinations of experiment and error distribution, 500 sets of values for y_i were to be generated and the corresponding estimates of V and K obtained. The author carried out the computer simulation; that part of the work concerned with the nonlinear minimization of the functions S_1 and S_3 in equations (4.19) and (4.22) will now be described.

It was evident that a robust and efficient minimization method was needed since S_1 and S_3 had each to be minimized 9000 times. Colquoun used a pattern search based on that of Bell and Pike [92] ; he found that this converged within 220 function evaluations for most cases, although he does not state with what accuracy. The author felt that a more efficient search using derivatives was possible since the evaluation of the derivatives was straight forward. A Gauss-Newton algorithm, embodying a line search on the corrections, was used with success and converged within ten iterations to a relative accuracy of 10^{-5} in V and K. From physical considerations, V and K cannot be negative and so the search was carried out using the transformed variables $\log_e V$ and $\log_e K$. Two difficulties were occasionally encountered, caused by the generation of experimental data unlikely to be met in practice; either K tended to zero (in the absence of constraints on K it would go negative) or else both V and K became large. Further examination of the expressions for S_1 and S_3 showed that a simple univariate search on K could be employed and which would cope with these difficulties.

Partial differentiation of equation (4.19) gives

$$\frac{\partial S_1}{\partial V}(V,K) = -2 \sum_{i=1}^{m} \frac{x_i}{K+x_i} \left(y_i - \frac{Vx_i}{K+x_i}\right) \qquad (4.23)$$

For a given value of K, a stationary value of S_1 will be at $V = V^*$ such that $\partial S_1(V^*,K)/\partial V=0$; from equation (4.23) it follows that

$$V^* = \sum_{i=1}^{m} \frac{y_i x_i}{K+x_i} \Big/ \sum_{i=1}^{m} \frac{x_i^2}{(K+x_i)^2} \qquad (4.24)$$

This stationary value is a minimum with respect to V since further differentiatio of equation (4.23) shows that

$$(V,K)\ \frac{\partial^2 S_1}{\partial V^2}\ (V,K) = 2 \sum_{i=1}^{m} \left(\frac{x_i}{K+x_i}\right)^2 \geq 0 \qquad \text{for all V}$$

The function $S_1(V^*,K)$ is a function of the single variable K since V^* is a

function of K. A direct search method to locate the minimum of $S_1(V^*,K)$ could be used; however since the derivatives are available at little extra cost of computation, they were used in the search. If $S_1(V^*,K)$ is partially-differentiated with respect to K we obtain

$$\frac{\partial S_1}{\partial K}(V^*,K) = 2\sum_{i=1}^{m}\left(y_i - \frac{V^*x_i}{K+x_i}\right)\left(-\frac{\partial V^*}{\partial K}\frac{x_i}{K+x_i} + \frac{V^*x_i}{(K+x_i)^2}\right) \qquad (4.25)$$

It is found by straight-forward algebraic manipulation that those terms in equation (4.25) involving $\partial V^*/dK$ sum to zero thus giving

$$\frac{\partial S_1}{\partial K}(V^*,K) = 2V^*\sum_{i=1}^{m}\frac{x_i}{(K+x_i)^2}\left(y_i - \frac{V^*x_i}{K+x_i}\right) \qquad (4.26)$$

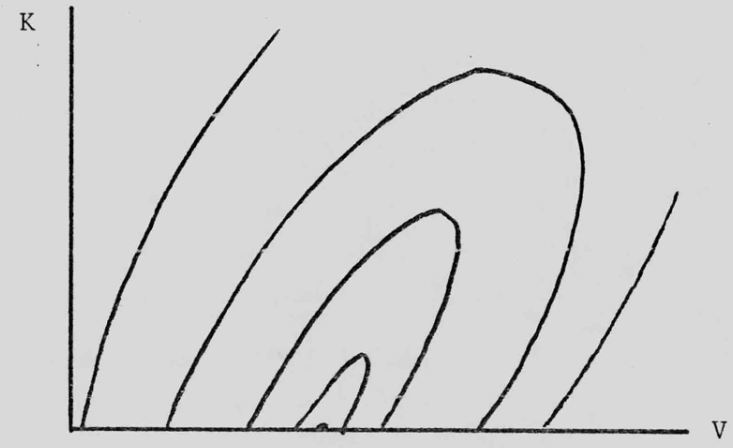
A simple bracketing procedure, followed by a dichotomous search, was used to locate a zero of $\partial S_1(V^*,K)/\partial K$. It was assumed that $S_1(V^*,K)$ is unimodal for the range of positive K; this was borne out by experience. The steps in the search are summarised as follows:

Step 1 — Set K=0; Compute V^*, $\partial S_1(V^*,K)/\partial K$ using equations (4.24) and (4.26)

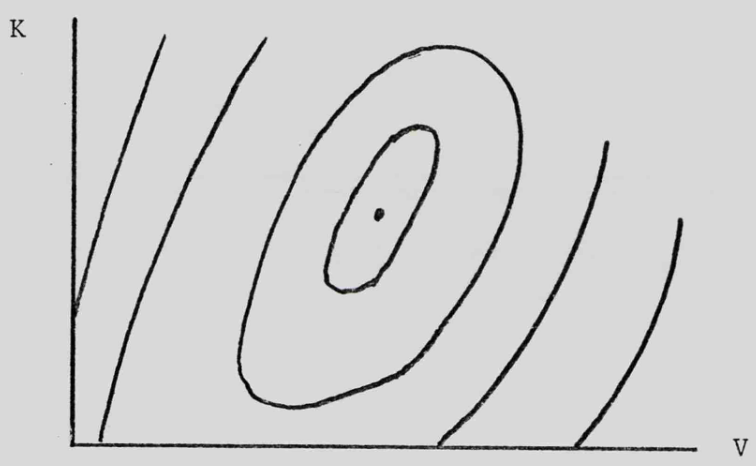
Step 2 — If $\partial S_1(V^*,K)/\partial K > 0$ then K, if unconstrained, would tend to negative values as shown in Figure 4.8a; in this case set K=0, V=V* as the best estimate and terminate the search. Otherwise, set KS=K, K=1, and L=1; continue from step 3.

Step 3 — Compute V^*, $\partial S_1(V^*,K)/\partial K$

Step 4 — If $\partial S_1(V^*,K)/\partial K > 0$ then a zero of $\partial S_1(V^*,K)/\partial K$, and hence a minimum for S_1, lies in the range KS to K as shown in Figure 4.8b; use a dichotomous search to locate the position of the zero to a relative accuracy ε on K and terminate the search. Otherwise .....



(a) Constrained minimum



(b) Unconstrained minimum

(c) Minimum when K,V tend to infinity

Figure 4.8 Possible contours of sum of squares when fitting Michaelis-Menten hyperbola

Step 5 If L=10 then it is assumed that V and K are tending to large values as shown in Figure 4.8c; terminate the search with a suitable error indicator set. Otherwise .....

Step 6 Set KS=K, K=2xK, L=L+1 and continue from step 3.

A similar analysis was applied to S_3 and the same search algorithm was used. Without going into any details, it can be shown that for S_3

$$V^* = \exp\left[\frac{1}{m}\sum_{i=1}^{m} \log_e\left\{ y_i \frac{(K+x_i)}{x_i} \right\}\right]$$

$$\frac{\partial S_3}{\partial K}(V^*,K) = 2\sum_{i=1}^{m} \log_e\left\{y_i \frac{(K+x_i)}{x_i}\right\}\left[\frac{1}{K+x_i} - \frac{1}{m}\sum_{j=1}^{m}\frac{1}{K+x_j}\right]$$

The algorithm was programmed in FORTRAN IV on an IBM 360/44. With $\varepsilon = 10^{-6}$ as the required accuracy in the search on K, most searches terminated within 30 iterations, comparing favourably with the method of Bell and Pike even when allowing for the additional computation in the present method. Faster algorithms could be obtained by better utilization of the derivatives; for example, Newton's method. Since the method was robust and acceptably fast (typically 3500 complete searches in 5 minutes processor time) a further investigation was not necessary. It is not appropriate here to describe the results of the simulation other than to state that the estimates of V and K produced by the Lineweaver-Burke plot were very poor in comparison with those produced by the nonlinear fits.

4.4 Problems with more than two variables

The following three case studies are optimization problems, involving more than two variables, which the author has solved using the techniques discussed previously. In all three instances, examination of the formulation of the problems led to a simpler method of solution. This is not always possible, especially in the design situation where there can be a large number of variables and no underlying simpler form (the author successfully applied the two-part algorithm of Chapter 2 to a least squares problem arising in the design of a zoom-lens; in this case he was supplied [93] with a "black-box" set of FORTRAN subroutines which calculated the functions and first derivatives).

The first of the three case studies describes the development of a global optimization method for a least squares problem with many local minima arising in cluster analysis. The second concerns the solution of nonlinear equations arising from the quantum theory of phase transitions; the problem was greatly simplified by reformulation. The third case is from the design of waveguides for microwaves; the problem involved five variables and was in fact reduced to a single variable problem.

4.4.1 An algorithm for cluster analysis

The process of classification [94] is of importance in many diverse areas of study such as archaeology, botany and linguistics. A classification of a given set of objects places each object into one of several initially-undefined classes; this can be contrasted with the related process of assignment in which a single object is placed in one of a number of previously-defined classes. An important statistical approach to classification is that of cluster analysis in which a quantitative measure of similarity between objects is defined and then the set of objects is partitioned into groups, in some optional fashion, such that objects in the same group are similar. Many algorithms exist [95] for cluster analysis; that to be described here uses a Euclidean measure of similarity as follows.

Suppose that there are n objects and it is required to partition them into g groups; note that the most suitable choice of g will usually be found by experimenting with different values. Each object can be described in terms of certain of its measurable properties; if the investigator chooses p such properties then an object can be represented by the point in p-dimensional Euclidean space given by the measurements on that object. For any given partition of the objects into disjoint groups, the variability of the objects within a group is defined as the sum of the squared distances of each point (object) from the centroidal value of all the points making up the group. The optimum partition is taken as that which minimizes the total of these within-group sum of squared distances. The following notation is used to express this mathematically; the n x p matrix X contains in X_{ij} the jth measurement for object i; the g x p matrix Z holds in Z_{kj} the centroidal value of the jth measurement for group k; and the n x g matrix Y is used to define the partition by having $Y_{ik} = 1$ if object i belongs to group k and $Y_{ik} = 0$ if otherwise. The total within-group sum of

squared distances is represented by S where

$$S = \sum_{k=1}^{g} \sum_{i=1}^{n} Y_{ik} \sum_{j=1}^{p} (X_{ij} - Z_{kj})^2 \qquad (4.27)$$

$$Z_{kj} = \frac{\sum_{i=1}^{n} Y_{ik} X_{ij}}{\sum_{i=1}^{n} Y_{ik}} \qquad j=1,\ldots p;\ k=1,\ldots g \qquad (4.28)$$

It is readily-shown [94, p.334] that a complete ennumeration of all possible partitions is not computationally-feasible for other than small values of n. As a result, various approximating algorithms have been developed which use iterative schemes to compute what is hoped to be a good estimate to the optimum partition. These methods fall into the following three classes, for each of which one typical algorithm is instanced. First, agglomerative algorithms [96] start with an initial partition of the objects into n groups so that each group contains a single object. At each iteration, two groups are merged so that after n-g iterations there are just g groups; the choice of which groups to merge is made so as to minimize the increase in S for that iteration. Second, divisive algorithms [97] commence with a single group containing all n objects. At each iteration one group is divided into two smaller groups so that after g-1 iterations there are g groups; this time the choice of group and the manner of its division are chosen to maximise the reduction in S at that iteration. Third, relocation algorithms [98] start from an initial partition into g groups, usually chosen in some random fashion. At each iteration, a set of possible movements of one or more objects between groups is considered, according to pre-defined rules; the move giving the greatest reduction in S is chosen. The iterations are continued until none of the allowable moves will produce a further reduction in S.

Fisher [99] proposed that the ennumeration problem could be reformulated as the following nonlinear programming problem:

$$\text{Minimize } S$$

$$\text{Subject to } \sum_{k=1}^{g} Y_{ik} = 1 \qquad i = 1,.. \ n$$

$$Y_{ik} \geq 0 \qquad i = 1,.. \ n; \ k = 1,.. \ g$$

The objective function S is defined as previously; however this time the values of the elements Y_{ik} are treated as continuous independent variables within the range 0 to 1. Each value Y_{ik} now represents the proportion of object i that is allocated to group k. At a minimum of S, the value of Y_{ik} could be interpreted as the probability that object i belongs to group k; this information could be more useful than the disjoint partition produced by the algorithms described earlier. Fisher did not suggest a method for solving his formulation of the problem; Gordon [100] was interested in developing a suitable algorithm and gave this task to the author.

An examination of the constrained optimization problem revealed that, as will be demonstrated, a local minimum of S can only occur at points where the values of the elements of Y are 0/1 as before. Taking partial derivatives of S and Z_{kj} with respect to any element Y_{lm} gives

$$\frac{\partial S}{\partial Y_{lm}} = \sum_{j=1}^{p} (X_{lj} - Z_{mj})^2 - 2 \sum_{k=1}^{g} \sum_{i=1}^{n} Y_{ik} \sum_{j=1}^{p} (X_{ij} - Z_{kj}) \frac{\partial Z_{kj}}{\partial Y_{lm}} \qquad (4.29a)$$

$$\frac{\partial Z_{kj}}{\partial Y_{lm}} = 0 \qquad \text{if } k \neq m \qquad (4.29b)$$

$$\frac{\partial Z_{kj}}{\partial Y_{lm}} = \frac{X_{lj} - Z_{mj}}{\sum_{i=1}^{n} Y_{im}} \qquad \text{if } k = m \qquad (4.29c)$$

In the expression for $\partial S/\partial Y_{lm}$ it will be seen that, in the summation over values of k, the only non-zero value of $\partial Z_{kj}/\partial Y_{lm}$ is for k = m; thus by substituting the corresponding expression for $\partial Z_{mj}/\partial Y_{lm}$ and (without altering the result) reversing the order of summation we obtain

$$\frac{\partial S}{\partial Y_{lm}} = \sum_{j=1}^{p} (X_{lj} - Z_{mj})^2 - 2 \sum_{j=1}^{p} \frac{(X_{lj} - Z_{mj}) \sum_{i=1}^{n} Y_{im} (X_{ij} - Z_{mj})}{\sum_{i=1}^{n} Y_{im}}$$

By the definition of Z_{mj} it follows that $\sum_{i=1}^{n} Y_{im} (X_{ij} - Z_{mj}) = 0$; hence the second term in the preceding expression for $\partial S/\partial Y_{lm}$ is zero and so

$$\frac{\partial S}{\partial Y_{lm}} = \sum_{j=1}^{p} (X_{lj} - Z_{mj})^2 \qquad (4.30)$$

Now suppose that a local minimum can exist with at least one row i of Y not made up of 0/1 values. Consider two elements Y_{lq} and Y_{lr}, both greater than zero and less than unity. If Y_{lq} is increased by δY and Y_{lr} reduced by the same amount, then the corresponding change δS in S is given by

$$\delta S = \delta Y(\partial S/\partial Y_{lq} - \partial S/\partial Y_{lr}) \qquad (4.31)$$

If q and r are chosen such that $\partial S/\partial Y_{lq} < \partial S/\partial Y_{lr}$, a reduction in S can always be obtained (except for pathological cases in which the derivatives are equal). Gordon added the observation that, from the previous analysis, $\partial S/\partial Y_{lm}$ equals the distance of object 1 from the mth group centroid. Consequently if, for given Y, one object is moved to its nearest group centroid say q, it will at the same time get further away from the other group centroids; consequently $\partial S/\partial Y_{lq}$ will remain the least of all the derivatives for that object and S will continuously decrease right up to $Y_{lq} = 1$.

Although a solution of the constrained optimization problem would still yield a disjoint partition, it was believed that an effective algorithm could be developed based on methods of constrained optimization. The current problem is characterised by three main features. First, it has a sum of squares objective function; second the number of variables ng and the number of constraints (n+1)g are large for typical cases of interest (for example, n=37 and g=5 in the example cited later); third, and most unusually, the global minimum of S which is sought belongs to the set of already-known disjoint partitions. The most common methods for least squares problems use the Gauss approximation to the Hessian; for example Marquardt [39], Hartley [37]; the computer resources for the storage and manipulation of matrices would be costly. Similarily, methods for handling the constraintse.g. Rosen [101]would also be costly in matrix storage and manipulation. Methods for global optimization, such as the trajectory approach of Branin [51] are primarily intended for problems with a moderate number of stationary points, where the problem lies in locating the local minima, rather than ennumerating them as in the present case. The simplest and, according to Dixon, Gomulka and Hersom [48], most-used method of global optimization was adopted, namely the multistart algorithm. This consists of carrying out local minimizations from a set of different initial values of the variables; the best minimum thereby found is then accepted as a good estimate to the global minimum. Obviously, the more minimizations that can be carried out, the greater the reliability of this method. Consequently the simple method of steepest descent [23] was chosen for the local minimization; this requires a modest amount of storage, no matrix manipulation and was found in practice to locate a minimum in a few iterations. The constraints were removed by a transformation of the variables Y_{ij} so that unconstrained minimization could be performed on the transformed variables.

First, the equality constaints were removed by using variables w_{ij} such that

$$Y_{ij} = w_{ij} / \sum_{k=1}^{g} w_{ik} \qquad (4.32)$$

Although it would be possible to reduce storage by treating one value, w_{ij}, say, as fixed and allowing the other g-1 values for that object to vary, this scheme was not implemented since it would imply an unequal treatment of the variables (Y_{ij}'could not become zero in its own right, only by values W_{ij} ($j \neq j'$) becoming large). Second, the non-negativity constraints on Y_{ij} were preserved by a further transformation of w_{ij} to v_{ij}; of various possible methods [9 p.82] a logarithmic transformation was taken to give

$$Y_{ij} = \exp(v_{ij}) / \sum_{k=1}^{g} \exp(v_{ik}) \qquad (4.33)$$

It can be easily verified that

$$\frac{\partial S}{\partial v_{ij}} = Y_{ij} \left(\frac{\partial S}{\partial Y_{ij}} - \sum_{k=1}^{g} Y_{ik} \frac{\partial S}{\partial Y_{ik}}\right) \qquad (4.34)$$

and it will be noted that the derivatives $\partial S/\partial Y_{ij}$ are obtained as a by-product during the computation of S. This expression for $\partial S/\partial v_{ij}$ is therefore the most practical to use since, for any values of v_{ij}, the corresponding Y_{ij}'s must be computed before obtaining the value of S.

At each iteration of the algorithm, the steepest descent corrections Δv_{ij} are computed by

$$\Delta v_{ij} = -\lambda \, \partial S/\partial v_{ij} \qquad (4.35)$$

where the scalar parameter $\lambda > 0$ is chosen to give a reduction in S and for which purpose the following simple univariate search on λ was employed. First an initial trial value of λ is taken, and reduced if necessary, to give a decrease in S; then this value is doubled until S increases when a minimum of S will be bracketed by the last three trial points. A simple Golden Section is then used to locate the position of the minimum more accurately. It was soon found by experience that it was desirable to impose

an upper bound λ_m on λ such that $|\Delta v_{ij}| < \Delta v_L$ where Δv_L is a pre-set value which determines λ_m for any iteration. When $\Delta v_L < 1$, then the search usually progresses slowly to a local minimum close to the starting-point; intuitively the curve of steepest descent is then being followed fairly closely to a nearby minimum. Further, when $\Delta v_L > 5$ then the search terminates rapidly in a few iterations but again at a local minimum close to the starting-point; probably this is because the search has insufficient opportunity to change direction advantageously. Values of $\Delta v_L \simeq 3$ were found to give a suitable balance and produce the best minima. It will be noted that the recommended value $\Delta v_L = 3$ does not depend in any way upon the data matrix X and is therefore universally-applicable to all problems. It was observed that, almost invariably, the chosen value of λ at each iteration equalled its maximum value λ_m for that iteration. The minimization was thus greatly-accelerated by taking $\lambda = \lambda_m$ as the initial trial value in the univariate search. If it produces a reduction in S then it is accepted, tacitly assuming that a minimum of S does not exist at $\lambda < \lambda_m$; otherwise the search on λ is continued as previously described by trying smaller values of λ. Note that, at the cost of some further computation, the assumption could be verified by computing $\partial S/\partial\lambda$ at $\lambda = \lambda_m$ since the derivatives $\partial S/\partial Y_{ij}$ will be available from the calculation for S at $\lambda = \lambda_m$; the possibility of the assumption being in error did not warrant this additional effort. The iterations are terminated when, for each object i there is some k such that $Y_{ik} \geq 0.999$ and the sign of the steepest descent corrections is such that Y_{ik} would increase at the next iteration. Occasionally rounding errors produce a failure in the search on λ to reduce S before this criterion is satisfied, but a local minimum of S will still usually have been effectively attained. A fail-safe limit of 100 iterations was imposed; this has not yet been invoked for the cases studied and usually the search terminates in 10 - 20 iterations.

The new algorithm was compared by Gordon and Henderson [102] against the agglomerative and divisive algorithms available in the CLUSTAN [103] package and a specially-written relocation algorithm. The test data used was a set of 37 surface pollen samples taken by Birks [104] who wished to ascertain whether the samples fell naturally into groups according to the proportions of pollen grains in each sample belonging to each of 48 species; thus in this case n=37 and p=48. Values of g=3, 4 and 5 were used in the runs cited. The new algorithm, named EUCLID, was also tested in combination with the relocation algorithm (RELOC) so that the final groups produced by EUCLID are used as starting values for RELOC; this combined algorithm is referred to as HYBRID. By their nature, the agglomerative and divisive algorithms can only be tried once on the same data since their starting partition is fixed; however, the EUCLID, RELOC and HYBRID algorithms start from a random partition and were each run 20 times for comparison purposes. The initial values of Y for EUCLID and HYBRID were generated from the set of numbers uniformly distributed in the range 1 to 3 and scaled so that $\sum_{k=1}^{g} Y_{ik} = 1$ for each object i; this ensured that all values were reasonably close to 1/g. The results of the comparison are shown in Tables 4.2 and 4.3; there is no result shown for the divisive algorithm when g=5 since this required an excessive amount of computation. Table 4.2 shows the ranking of the runs for each value of g; all runs enclosed in brackets produced the same final partition. Thus, for g=3, the best partition was produced by 18 EUCLID (E) runs, 19 HYBRID (H) runs and 20 RELOC (R) runs; the next best partitions in order were produced by the agglomerative (A) algorithm, the divisive algorithm (D) and EUCLID, and finally EUCLID and HYBRID. Table 4.3 shows the partitions obtained when g=5, together with the sum of squares S. It is believed by Gordon and Henderson that the results, together with the other unpublished test cases, demonstrate that the algorithm is a useful practical tool; the algorithm is

Number of Groups g	Algorithms ranked in order of increasing sum of squares of the final partition
3	(18E, 19H, 20R), A, (D, E), (E, H)
4	(E, 9H), E, A, 2E, (D, E, 6H, 13R), 6E, (3H, 4R), 5E, 3R, 2E, 2H, 2E
5	(2E, 12H, 2R), E, A, 2E, (3H, R), 2E, H, 2E, (3H, 8R), 2E, (H, 7R), 7E, 2R, 2E

Key

A = Agglomerative

D = Divisive

E = EUCLID

H = HYBRID

R = Relocation

Table 4.2 Comparison of five clustering algorithms

Sum of Squares	Algorithm	Partition
1.101	(2E, 12H, 2R)	(1,2,4-6), (3,7,9,10), (8,11,15-26) (12,13), (14,27-37)
1.145	E	(1,2,4,5), (3,6,7,9,10), (8,11,15-26), (12,13), (14,27-37)
1.147	A	(1-6), (7,9,10), (8,11,15-25), (12,13), (14,26-37)
1.186	E	(1,2,4,5), (3,6-10), (11,15-26) (12,13), (14,27-37)
1.229	E	(1,5), (2-4,6-10), (11,15-26), (12,13), (14,27-37)
1.254	(3H,R)	(1,2,4-6), (3,7,9,10), (8,11-13,17-19), (14,27-37), (15,16,20-26)
1.320	E	(1-6), (7,9,10), (8,11,19), (12,15-18,20-26), (13,14,27-37)
1.329	E	(1,2,4-6), (3,7,9,10), (8,11,17-19), (12,13,15,16,20-26), (14,27-37)
1.335	H	(1,2,4-6), (3,7,9,10), (8,11,12,15-26 (13,27-29,32), (14,30,31,33-37)
1.373	E	(1-7,9,10), (8,11), (12,13), (14,27-37), (15-26)
1.377	E	(1-5), (6-11), (12,13,17-19), (14,27-37), (15,16,20-26)
1.384	(3H,8R)	(1-7,9,10), (8,11,17-19,24), (12,13) (14,27-37), (15,16,20-23,25,26)
>1.384	11E, H, 9R	

Key A = Agglomerative
E = EUCLID
H = HYBRID
R = Relocation

Table 4.3 Partitions of Birks' samples into five groups

both fast (the 60 runs summarised in Table 4.2 took only 41 seconds of CPU time on an IBM 370/165) and compact, since the storage requirement is proportional to n for any given p and g. The algorithm is now incorporated in CLUSTAN and it is hoped to receive reports on its efffectiveness from other researchers.

Following this work, further consideration has been given recently to the application of a standard method for constrained optimization. It has been found that, by taking advantage of the special nature of the constraints which permit some algebraic analysis to be used in place of numerical procedures, it is possible to modify Rosen's method of gradient projection in order that it can be used, at acceptable computational cost, on the cluster analysis problem Presumably similar conclusions could be obtained with other existing methods such as those described by Zoutendijk [105]. Rosen's method must first be outlined; this will be done as it applies to problems with linear constraints of the type

$$\begin{aligned} &\text{Minimize } f(\underline{x}) \\ &\text{Subject to } B\underline{x} = \underline{b} \\ &\qquad\qquad C\underline{x} \geqslant \underline{c} \end{aligned}$$

Here the r x n matrix B and corresponding vector $\underline{b}$ define r equality constraints and the s x n matrix C and vector $\underline{c}$ define s inequality constraints on the independent variables $\underline{x}^T = (x_1, \ldots x_n)$. Rosen's method starts with an initial feasible vector $\underline{x}^o$ and generates a sequence of further feasible vectors converging to a local minimum of $f(\underline{x})$; the following steps are performed at each iteration:

Step 1 The direction of steepest descent for unconstrained minimization is computed as

$$\underline{d}^T = -(\partial f/\partial x_1, \ldots \partial f/\partial x_n)$$

where the derivatives are evaluated at the current vector $\underline{x}^k$.

Step 2 The matrix A is composed of the rows of B and C corresponding to active constraints; the equality constraints are always active while the inequality constraints are only active for rows i for which $\sum_{j=1}^{n} C_{ij} x_j = 0$. This matrix A is used to define a linear manifold in which any correction $\Delta\underline{x}$ to $\underline{x}^k$ must continue to satisfy the current active constraints by ensuring that $A\,\Delta\underline{x} = 0$. The search direction $\underline{d}$ is orthogonally projected onto this manifold to give a new search direction $\Delta\underline{x}$ such that

$$\Delta\underline{x} = P\underline{d}$$

$$P = \left[I - A^T(AA^T)^{-1}A\right]$$

Step 3 If $\Delta\underline{x} \neq 0$ then a line search is used to find $\lambda = \lambda_m$ such that $f(\underline{x}^k + \lambda\,\Delta\underline{x})$ is a minimum in the range $0 \leqslant \lambda \leqslant \lambda^*$. The upper bound λ^* is determined by the first of the non-active inequality constraints to be satisfied as an equality. Then $\underline{x}^{k+1}$ is set to $\underline{x}^k + \lambda_m\,\Delta\,\underline{x}$ and the process continued from Step 1; if $\lambda_m = \lambda^*$ then a further constraint will be added to the active set.

Step 4 If $\Delta\underline{x} = 0$, then the vector $\underline{u} = -(AA^T)^{-1}\,A\,\underline{d}$ is calculated. If all $u_i \geqslant 0$, then the Lagrangian conditions [8, p19] are satisfied, a local minimum has been reached and the search is terminated. Otherwise if some $u_i < 0$ then further progress can be made if the corresponding active constraint is released by deleting the ith row of A and returning to Step 1;

usually the row corresponding to the most negative u_i would be chosen.

Originally, Rosen's method was discarded because the computation and storage of P would require excessive computer resources. However examination of the constraint set shows that it may be divided into independent sets within which only those values Y_{ik} for a given object i appear. Thus it is admissible to consider the constraints on each row of Y in turn and compute the corresponding projection matrix P_i. The matrix A_i for object i will be typically given by

$$A_i = \begin{bmatrix} 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\ 1 & 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & & & & & & 0 \end{bmatrix} \qquad (4.36)$$

The first row corresponds to the equality constraint $\sum_{k=1}^{g} Y_{ik}=1$; the second and subsequent rows correspond to m_i active inequality constraints for which $Y_{ik}=0$. For simplicity, the active inequalities are shown as applying to variables Y_{ik} (k=1, .. m_i) although the argument which follows is not affected by this assumption other than in the group numbering. It is found, by some straightforward but tedious matrix algebra that

$$P_i = \begin{bmatrix} 0 & . & . & 0 & 0 & 0 & . & . & 0 \\ . & . & . & . & . & . & . & . & . \\ 0 & . & . & 0 & 0 & 0 & . & . & 0 \\ 0 & . & . & 0 & 1-\frac{1}{g-m_i} & \frac{-1}{g-m_i} & . & . & \frac{-1}{g-m_i} \\ . & . & . & . & . & . & . & . & . \\ 0 & . & . & 0 & \frac{-1}{g-m_i} & \frac{-1}{g-m_i} & . & . & 1\frac{-1}{g-m_i} \end{bmatrix} \qquad (4.37)$$

in which the first m_i rows contain zeroes and the last $g-m_i$ rows contain non-zero elements in columns m_{i+1} through to column g. At the same time, it is found also that

$$(A_i A_i^T)^{-1} A_i = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & \frac{1}{g-m_i} & \cdots & \frac{1}{g-m_i} \\ 1 & 0 & 0 & \cdots & 0 & 0 & \frac{-1}{g-m_i} & \cdots & \frac{-1}{g-m_i} \\ 0 & 1 & 0 & \cdots & 0 & 0 & \frac{-1}{g-m_i} & \cdots & \frac{-1}{g-m_i} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & \frac{-1}{g-m_i} & \cdots & \frac{-1}{g-m_i} \end{bmatrix} \qquad (4.38)$$

Using these algebraic results, the method of Rosen resolves to the following simple form:

Step 1 Compute $\underline{d}^T = -(\partial S/\partial Y_{11}, \partial S/\partial Y_{12}, \ldots \partial S/\partial Y_{ng})$

Step 2 Set

$\Delta Y_{ik}=0$ if $Y_{ik} = 0$

$\Delta Y_{ik} = -(\frac{\partial S}{\partial Y_{ik}} - \frac{1}{g-m_i} \sum_{k=1}^{m_i} \frac{\partial S}{\partial Y_{ik}})$ if $Y_{ik} > 0$

where the summation over k is taken only for those values of k for which $Y_{ik} > 0$.

Step 3 If one, or more, $\Delta Y_{ik} \neq 0$ then find $\lambda = \lambda m$ to minimize S as for the standard method of Rosen;

Set $Y_{ik} = Y_{ik} + \lambda_m \Delta Y_{ik}$ $(i=1,..n;k=1,..g)$ and return to step 1.

Step 4 If all $\Delta Y_{ik}=0$ then it follows that each row of Y has g-1 zero elements and one element of unity. For row i, the values of $\underline{u}$ are given by

$u_1 = \partial S/ \partial Y_{ik}'$ where $Y_{ik}' = 1$

$u_{k+1} = \partial S/\partial Y_{ik} - \partial S/\partial Y_{ik}'$ for $Y_{ik} = 0$

It was shown earlier that $\partial S/\partial Y_{ik} \geq 0$ and so the Lagrangian condition is satisfied if, for each object i,

we have $\partial S/\partial Y_{ik'} < \partial S/\partial Y_{ik}$ $(k \neq k')$ and a local minimum of S will be reached. Otherwise, the values i and k are found for which $\partial S/\partial Y_{ik} - \partial S/\partial Y_{ik'}$ is most negative. By increasing Y_{ik} to unity and reducing $Y_{ik'}$ to zero, then a guaranteed reduction in S will be obtained since this corresponds to relocating object i from group k' to group k. This relocation procedure is then repeated until the Lagrangian condition is satisfied.

It is interesting to note that a hybrid algorithm results from Rosen's method. However, the reposition movements permitted at Step 4 are only a subset of those allowed by the RELOC part of HYBRID; since RELOC considers all possible repositionments of all objects then it can progress from a local minimum of S while Rosen's method cannot. The algorithm was implemented as a computer program; an immediate difficulty encountered was in catering for round-off errors when discriminating between 0/1 values of Y and values close to these limits. This problem was circumvented by using tests of the form $|Y_{ik}| < 10^{-4}$ and $|1-Y_{ik}| < 10^{-4}$ at the expense of additional computation. The major problem met was that it is known initially there are no active inequality constraints and at a solution there are n(g-1) active equality constraints. Since the method only permits one additional constraint to become active at each iteration, then a minimum of n(g-1) iterations are required. In trials with the data of Birks exactly n(g-1) iterations were taken (148 for g=5). Although processor time per iteration of Rosen's method was comparable with that of EUCLID, the fact that each iteration was restricted to taking small steps in Y meant that, as with EUCLID for small Δv_L, minima close to the starting point were found. It would be possible to modify the new method such that a step was taken with maximum component Δv_L at each iteration; this would require the re-evaluation of the correction vector as λ is increased whenever a new inequality constraint became active. The additional programming complication does not at this time seem justified, in view of the success of EUCLID.

4.4.2 Quantum theory of phase transitions

This problem is by MacLeod [106] who wished to solve a set of fifteen nonlinear equations which he had derived when studying the quantum theory of phase transitions. The following description is of the equations, as they were given to the author by Macleod, and the method adopted by the author to solve them. No explanation of the physical significance of the equations can be given since this is the substance of MacLeod's thesis which has not yet been completed.

The fifteen independent variables comprise the thirteen variables K_1 through to K_{13} and the two variables p_1 and p_2. The variables K_i are subjected to a transformation, involving p_1 and p_2, to give new variables K_i'; it is required to find a fixed-point of the transformation such that $K_i' = K_i$, thereby generating thirteen equations. Two additional equations, specifying required conditions on p_1 and p_2, make up the fifteen. The steps in the transformation are as follows:

Step 1 For i=1, .. 13 set

$$u_i = 2 \cosh (p_1 + p_2 X_{i2}) \exp \left(3 \sum_{j=1}^{13} X_{ij} K_j\right)$$

Step 2 For i=1, .. 4 set

$$a_i = \sum_{j=1}^{13} A_{ij} \sum_{k=1}^{13} X'_{ij} u_k \qquad \text{where } X' = X^{-1}$$

Step 3 Set

$$w_1 = \frac{1}{8} \log_e (a_1 a_2^3 a_3^3 a_4) + 6 \log_e 2$$

$$w_2 = \frac{1}{8} \log_e (a_1 a_2 / a_3 a_4) + p_1/3$$

$$w_3 = \frac{1}{8} \log_e (a_1 a_4 / a_2 a_3)$$

$$w_4 = \frac{1}{8} \log_e (a_1 a_3^3 / a_2^3 a_4)$$

Step 4 For i=1, ...13 set

$$x_i = \tfrac{1}{2} \log_e \cosh (3w_2 + w_3 X_{i2} + w_4 y_i) + \tfrac{1}{2} \log_e \cosh (3w_2 + w_3 X_{i2} + w_4 z_i)$$

Step 5 For i=1, ... 13 set

$$K'_i = \sum_{j=1}^{13} X'_{ij} x_j + \delta_{i1}(3w_1 + \log_e 2) + \delta_{i2} w_2 + \delta_{i3} w_3 / 2$$

where $\delta_{ii} = 1$ and $\delta_{ij} = 0$, $i \neq j$

As shown in Figure 4.9 , the elements of the matrix X and the vectors $\underline{y}$ and $\underline{z}$ are constant while the elements of the matrix A depend upon p_1. A Jacobian matrix J is defined so that $J_{ij} = \partial K'_i / \partial K_j$; from both physical and mathematical considerations it can be shown that its transpose J^T always has an eigenvalue of 9. The conditions required on p_1 and p_2 are that, at a fixed-point of the transformation, the eigenvector $\underline{v}$ of J^T corresponding to the eigenvalue of 9 must satisfy the equations

$$\sum_{i=1}^{13} \frac{\partial K'_i}{\partial p_1} v_i = 0 \qquad (4.39a)$$

$$\sum_{i=1}^{13} \frac{\partial K'_i}{\partial p_2} v_i = 0 \qquad (4.39b)$$

At first sight, the problem lends itself to a nested search in which fixed values of p_1 and p_2 could be taken and values of K_i satisfying the equations $K'_i - K_i = 0$ found; the p_1 and p_2 values could then be varied to satisfy equations (4.39a) and (4.39b). An immediate difficulty was apparent in that, at step 3 of the transformation, the arguments in the logarithmic terms could become negative. To prevent this occurring, constraints could be placed on the K_i values but it is evident that an explicit form for these constraints is not readily available. However, consideration of this difficulty led to a far simpler formulation of the original problem.

The original search for a fixed-point using thirteen independent variables K_i was replaced by an equivalent search using four variables b_1 to b_4 which are not subject to any constraints. The values of b_i are sought to give zero values of the four residual terms f_i obtained by the following steps.

$$X = \begin{bmatrix}
1 & 6 & 6 & 6 & 3 & 6 & 2 & 12 & 6 & 6 & 3 & 6 & 1 \\
1 & -6 & 6 & 6 & 3 & -6 & -2 & -12 & 6 & 6 & 3 & -6 & 1 \\
1 & 0 & -6 & 6 & -3 & 0 & 0 & 0 & 6 & -6 & 3 & 0 & -1 \\
1 & 2 & -2 & -2 & 3 & -6 & -2 & 4 & -2 & -2 & 3 & 2 & 1 \\
1 & -2 & -2 & -2 & 3 & 6 & 2 & -4 & -2 & -2 & 3 & -2 & 1 \\
1 & 4 & 2 & 2 & 1 & 0 & 0 & 0 & -2 & -2 & -1 & -4 & -1 \\
1 & 2 & 2 & -2 & -1 & 2 & -2 & -4 & 2 & -2 & -1 & 2 & 1 \\
1 & 2 & -2 & 2 & -1 & -2 & 2 & -4 & -2 & 2 & -1 & 2 & 1 \\
1 & 0 & 2 & -2 & -3 & 0 & 0 & 0 & -2 & 2 & 3 & 0 & -1 \\
1 & -2 & 2 & -2 & -1 & -2 & 2 & 4 & 2 & -2 & -1 & -2 & 1 \\
1 & -2 & -2 & 2 & -1 & 2 & -2 & 4 & -2 & 2 & -1 & -2 & 1 \\
1 & -4 & 2 & 2 & 1 & 0 & 0 & 0 & -2 & -2 & -1 & 4 & -1 \\
1 & 0 & -2 & -2 & 1 & 0 & 0 & 0 & 2 & 2 & -1 & 0 & -1
\end{bmatrix}$$

$$\underline{y} = \begin{bmatrix} 3 \\ 3 \\ -3 \\ -1 \\ -1 \\ 1 \\ 3 \\ -1 \\ 1 \\ 3 \\ -1 \\ 1 \\ 1 \end{bmatrix} \qquad \underline{z} = \begin{bmatrix} 3 \\ 3 \\ -3 \\ -1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -3 \end{bmatrix}$$

$$A = \begin{array}{c}
\begin{array}{cccccccccccc} 1 & t & t^2 & t^2 & t^2 & t^3 & t^3 & t^3 & t^4 & t^4 & t^4 & t^5 & t^6 \end{array} \\
\begin{bmatrix}
1 & 6 & 6 & 3 & 3 & 6 & 2 & 12 & 6 & 3 & 3 & 6 & 1 \\
1 & 2 & 2 & -1 & -1 & 2 & -2 & -4 & 2 & -1 & -1 & 2 & 1 \\
1 & -6 & 6 & 3 & 3 & -6 & -2 & -12 & 6 & 3 & 3 & -6 & 1 \\
1 & -2 & 2 & -1 & -1 & -2 & 2 & 4 & 2 & -1 & -1 & -2 & 1
\end{bmatrix}
\end{array}$$

$$t = \tanh(p_1)$$

Figure 4.9 Data for phase transition equations

Step 1 Replace the logarithmic terms in step 3 of the transformation by the b_i values such that

$$w_1 = b_1/8 + 6\log_e 2$$
$$w_2 = b_2/8 + p_1/3$$
$$w_3 = b_3/8$$
$$w_4 = b_4/8$$

Step 2 Compute K'_i, i=1, .. 13 using steps 4 and 5 of the transformation

Step 3 Set $K_i = K'_i$, i=1, .. 13

Step 4 Compute a_i, i=1, .. 4 using steps 1 and 2 of the transformation

Step 5 Compute the residuals

$$f_1 = b_1 - \log_e(a_1 a_2^3 a_3^3 a_4)$$
$$f_2 = b_2 - \log_e(a_1 a_2 / a_3 a_4)$$
$$f_3 = b_3 - \log_e(a_1 a_4 / a_2 a_3)$$
$$f_4 = b_4 - \log_e(a_1 a_3^3 / a_2^3 a_4)$$

It will be noted that when the residuals are zero then a fixed-point of the transformation will have been found and step 3, which assumes that $K_i=K_i'$, will be valid. The fact that it is invalid for other values of b_i not giving a fixed-point is of no consequence. The residuals as defined by step 5 above could still give problems with the logarithmic terms and so an alternative formulation was tried so that

$$f_1 = a_1 a_2^3 a_3^3 a_4 - \exp(b_1)$$
$$f_2 = a_1 a_2 - a_3 a_4 \exp(b_2)$$
$$f_3 = a_1 a_4 - a_2 a_3 \exp(b_3)$$
$$f_4 = a_1 a_3^3 - a_2^3 a_4 \exp(b_4)$$

Practical experience showed that this form could lead to apparent, but erroneous, solutions in which the b_i values tended to large, negative values

with correspondingly small a_i values. In this case although the residuals became small in absolute magnitude, the two terms comprising the residual were still unequal and tests for equality were susceptible to rounding error. To ensure that the residuals were composed of two terms of order unity they were then reformulated as

$$f_1 = \exp(-b_1)\, a_1 a_2^3 a_3^3 a_4 - 1$$

$$f_2 = \exp(-b_2)\, (a_1 a_2 / a_3 a_4) - 1$$

$$f_3 = \exp(-b_3)\, (a_1 a_4 / a_2 a_3) - 1$$

$$f_4 = \exp(-b_4)\, (a_1 a_3^3 / a_2^3 a_4) - 1$$

Using this form, a fixed point for any given values of p_1 and p_2 was found readily using Newton's method for nonlinear equations, with a line search to ensure that the sum of squares of the residuals was reduced at each iteration.

It was decided, for reasons of efficiency, that it would be better to search for p_1 and p_2 simultaneously with the search for a fixed-point, rather than in an outer loop as first proposed. The problem thus became one of six equations in the six unknowns b_1, b_2, b_3, b_4, p_1 and p_2. The Jacobian J was evaluated on the assumption that for the given variable values, a fixed-point had been reached; that this assumption only holds true at a solution of the equations does not affect the results. Thus to compute the Jacobian, the values of K_i are first set to the K'_i values derived from the current variables (steps 1 and 2 of the revised formulation of the problem). Then steps 1 to 5 of the original transformation are followed, only this time computing the partial derivatives $\partial u_i/\partial K_j$, $\partial a_i/\partial K_j$, $\partial w_i/\partial K_j$, $\partial x_i/\partial K_j$ and ultimately $\partial K'_i/\partial K_j$. At the same time the values $\partial K'_i/\partial p_1$ and $\partial K'_i/\partial p_2$ required by equations (4.39 a) and (4.39 b) are found. It should be noted that there is no problem with the logarithmic terms when evaluating the derivatives $\partial w_i/\partial K_j$.

A major saving in computational effort was made by avoiding the need for an eigenvalue/eigenvector analysis. Assuming that J^T has an eigenvalue of 9, then the eigenvector $\underline{V}$ must satisfy $(J^T - 9I)\,\underline{V} = 0$. This gives 13 equations but it is known that (J^T-9I) is of rank 12 and there are 12 independent equations. Taking V_1 as unity and considering equations 2 through to 13 gives

$$\begin{bmatrix} J_{2,2}-9 & J_{3,2} & \cdots & J_{13,2} \\ J_{2,3} & J_{3,3}-9 & \cdots & J_{13,3} \\ \vdots & \vdots & \ddots & \vdots \\ J_{2,13} & J_{3,13} & \cdots & J_{13,13}-9 \end{bmatrix} \begin{bmatrix} V_2 \\ V_3 \\ \vdots \\ V_{13} \end{bmatrix} = \begin{bmatrix} J_{1,2} \\ J_{1,3} \\ \vdots \\ J_{1,13} \end{bmatrix}$$

Any standard library procedure for solving sets of linear equations with a real, unsymmetric matrix of coefficients can be used to calculate the values of V_2 to V_{13}.

The problem was programmed on an IBM 360/44 in FORTRAN IV and a solution successfully obtained for which p_1=2.6413; p_2=0; K_1=-2.7790; K_2=0.44021 and the remaining values of K_i are zero. Unfortunately, this solution was of pathological interest to MacLeod. Trials with the program starting from a wide range of initial estimates of the variables always produced this same solution, tending to support the belief that it was the only solution. Subsequent work by MacLeod, and similar experiences by other workers in the same area of quantum theory research, have led him to reconsider the theoretical basis of the equations.

4.4.3 Waveguide design

This case study comes from work by Croydon[107] on the design of waveguides for the transmission of microwaves, following earlier work by Baden Fuller[108]. First, the problem as presented to the author will be described; then the manner in which the author applied the techniques of optimization to solve the problem will be discussed.

Croydon had written a computer program which solved a set of eight ordinary, simultaneous differential equations by numerical integration. The dependent variables will be denoted by Y_1 to Y_8 and the variable of integration by r. Croydon sought values of four parameters $\underline{K}^T=(K_1,K_2,K_3,K_4)$, which appeared in the differential equations, such that at $\underline{K}=\underline{K}^*$ the four equations $Y_i=0$ (i=5, .. 8) were all satisfied at some point r=r* during the integration. He used a coarse pattern search whereby each K_i was varied over a set of ten discrete values; integrations were carried out for all 10^4 possible combinations of values for $\underline{K}$. The output from these integrations was inspected to see if good estimates of $\underline{K}^*$ and r* could be found. A further complication was that, having located a solution, Croydon then wished to vary a fifth parameter β, also appearing in the equations, to find a value $\beta=\beta^*$ such that r*=b, where b is a specified constant. Obviously Croydon's scheme was costly in terms of both human and computer resources; a method of automating the search for $\underline{K}^*$ and β^* was developed as follows by the author.

First, further information was obtained about Croydon's program. The equations were derived from Maxwell's classical equations for electromagnetic radiation. The variable r denotes the radial distance from the longitudinal axis of a cylindrical waveguide, consisting of a central air gap at r<a and an annular ferrite core, from r=a to r=b, encased in a copper sleeve. The dependent variables are the magnetic field strengths H_θ and H_z and the electric

field strength E_θ and E_z; the suffices θ and z refer to the tangential and axial components respectively. These quantities are represented by the complex values

$$H_\theta = Y_1 + j\,Y_2 \qquad E_\theta = Y_5 + j\,Y_6$$

$$H_z = Y_3 + j\,Y_4 \qquad E_z = Y_7 + j\,Y_8$$

where $j^2=-1$.

The differential equations can be expressed in the form

$$\frac{d}{dr}\underline{Y} = A\,\underline{Y}$$

in which the 8 x 8 matrix A depends upon β and r; it is the dependence upon r which makes a numerical integration necessary. The values of $\underline{K}$ specify eight boundary conditions

$$\underline{Y} = B\,\underline{K} \qquad \text{at } r=a$$

in which the 8 x 4 matrix B is a function of β. A further four boundary conditions are given by the requirement that at r=b the electric field strength be zero i.e. $Y_i=0$ for i=5 to 8. The search on $\underline{K}$ and β is carried out to satisfy these last four conditions.

Since the differential equations are linear in $\underline{Y}$, it follows that the solution for $\underline{Y}$ is of the form

$$\underline{Y} = C(r,\beta)\underline{K}$$

where, as indicated, the elements of the 8 x 4 matrix depend upon r and β. Although C cannot be expressed analytically, the value of C at given r and for specified β can be found by carrying out integrations for four linearly-independent trial vectors for $\underline{K}$. If for the 1th trial, the vector is $\underline{K}^1$ and the values of $\underline{Y}$ at the required radius are $\underline{Y}^1$ then for each row i of C we have

$$\sum_{j=1}^{4} K_j^1\, C_{ij} = Y_i^1 \qquad (1=1, \ldots 4)$$

These equations can be solved simultaneously to give the values of C on each row. Note that while this is an exact relationship, the values of Y_i^1 are subject to errors incurred by the numerical integration process. If the last four rows of C are represented by the 4 x 4 matrix $D(r,\beta)$ then the problem can be stated as find β^* and $\underline{K}^*$ such that

$$D(b,\beta^*)\ \underline{K}^* = 0$$

The trivial solution $\underline{K}^* = 0$ is of no interest since this would imply zero electric and magnetic fields throughout the waveguide. Nontrivial solutions can exist only if D is singluar, which could be achieved by varying β. In this event, $\underline{K}^* = \alpha\underline{V}$ where $\underline{V}$ is the eigenvector corresponding to a zero eigenvalue of D and α is an arbitrary constant. An equivalent, but easier-to-program, approach was taken in preference to this eigenvalue-eigenvector method.

Without loss of generality we can fix one component of $\underline{K}$; K_4 was chosen and taken as unity. We then define residuals

$$f_i = \sum_{j=1}^{3} D_{ij}\ (b,\beta)K_j + D_{i4}(b,\beta) \qquad (i=1, \ldots 4)$$

and search for β, K_1, K_2 and K_3 to reduce these residuals to zero. Since the residuals are linear in the K_j's, then a standard linear least squares solution gives the values of K_1, K_2 and K_3 which, for the current β, correspond to the minimum sum of squares of the residuals, which will be denoted by $f(\beta)$. A simple line search can then be used to locate values of β^* (if any) for which $f(\beta^*) = 0$.

A FORTRAN IV program to do the search was written and successfully run. The evaluation, for given β, of the elements of C (and hence of D) was done in a different way from that described. To reduce the effect of errors in the integration, the partial derivatives $\partial Y_i/\partial K_j$ were integrated in parallel with the integration of the Y_i values. It will be observed

that $C_{ij}=\partial Y_i/\partial K_j$. The relevant differential equations are, letting $Z_{ij} = \partial Y_i/\partial K_j$

$$\frac{dZ_{ij}}{dr} = \sum_{k=1}^{8} A_{ik}Z_{kj} \qquad (i=1,.8;\ j=1,2,3)$$

$$Z_{ij} = B_{ij} \quad \text{at} \quad r=a$$

A simple grid search was used in which $f(\beta)$ was evaluated at a series of equally-spaced values of β in the range of interest. If a minimum of $f(\beta)$ was bracketed by three successive β values, then the position of this minimum was located accurately by a Golden Section search. The typical variation of $f(\beta)$ with β is shown in Figure 4.10, where it will be observed that there are two solutions for β^*. The discontinuity at $\beta=1$ is due to the presence of terms involving division by β^*-1 in the matrices A and B.

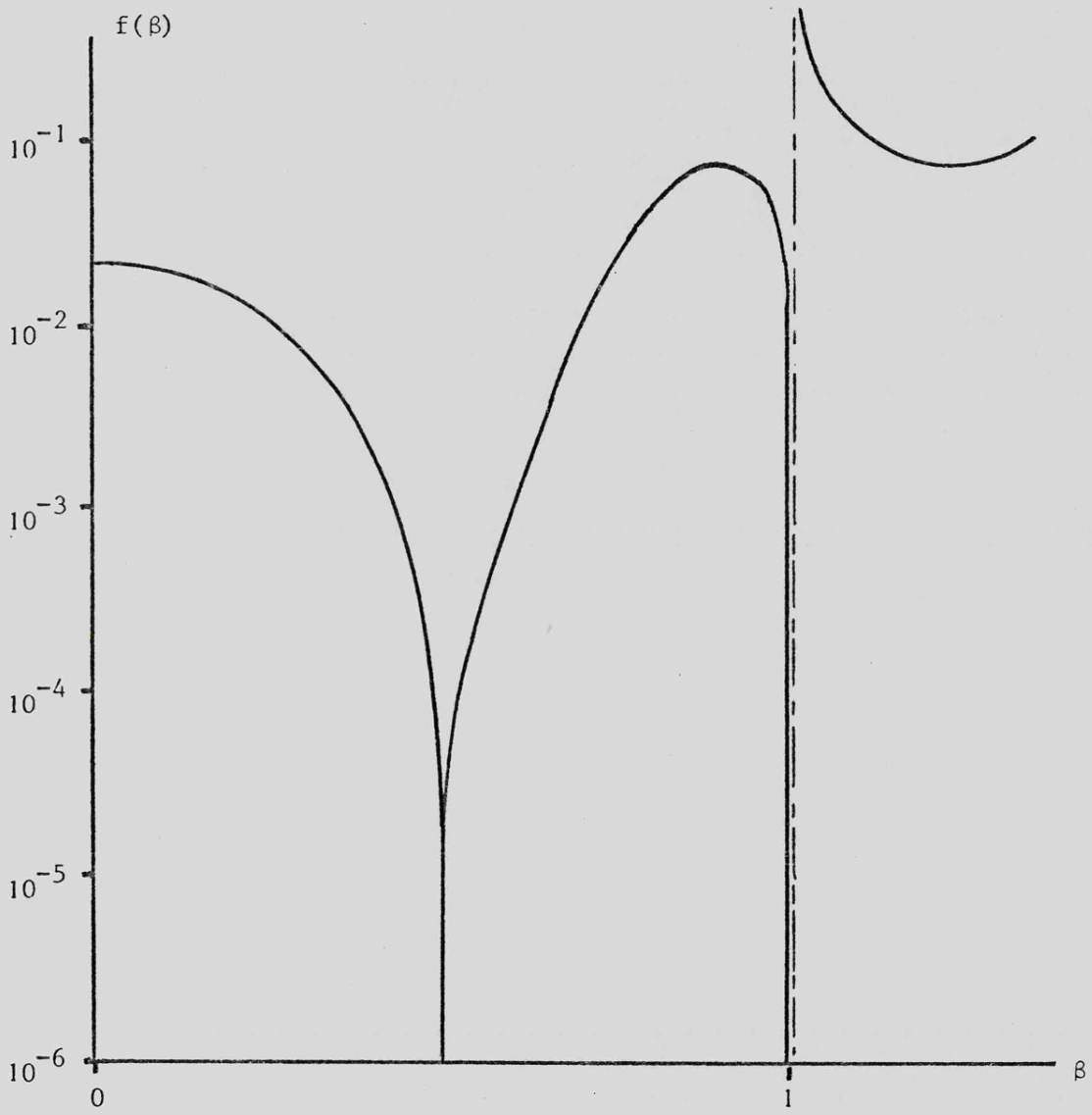


Figure 4.10 Typical form of f(β)

5. ASSESSMENT OF THE RESEARCH

The author believes that the research has proved worthwhile in several areas. First, the two-part algorithm described in Chapter 2 is a new and powerful tool for solving difficult nonlinear least squares problems and systems of nonlinear equations. Second, the hybrid algorithm discussed in Chapter 3 embodies some novel features, including the use of parametric linear programming methods. It is a reliable and accurate method for solving sets of nonlinear equations of the type and difficulty used by most researchers to test new algorithms, although it was not so successful as the two-part algorithm on the extremely difficult Problem 1 of the Appendix. Third, the case studies highlight a number of aspects of the application of optimization techniques to practical problems; the author believes that the main observation from this part of the research is that it is beneficial to consider alternative formulations of a problem before carrying out the optimization. Fourth, a new and extremely-useful algorithm for cluster analysis was developed.

APPENDIX TEST PROBLEMS

The following nine test problems have the objective function $F(\underline{x}) = \sum_{i=1}^{m} f_i^2(\underline{x})$ where $m \geq n$. The starting point is denoted by $\underline{x}^o$ and the solution by $\underline{x}^*$.

Problem 1 Skwirzynski [60]

$m=8;\ n=8$

$$f_i(\underline{x}) = x_3(1-x_1x_2)(\exp[\alpha_i]-1) - Y_{5i} + Y_{4i}\,x_2$$

$$f_{i+4}(\underline{x}) = \frac{x_1x_3}{x_2}(1-x_1x_2)(\exp[\beta_i]-1) - Y_{5i}x_1 + Y_{4i}$$

$$\alpha_i = x_4(Y_{1i} - Y_{3i}\,x_6 \times 10^{-3} - Y_{5i}x_7 \times 10^{-3})$$

$$\beta_i = x_5(Y_{1i} - Y_{2i} - Y_{3i}\,x_6 \times 10^{-3} + Y_{4i}\,x_8 \times 10^{-3})$$

$$i = 1, 2, 3, 4$$

The constants Y_{ij} $(i=1, .. 5;\ j=1, .. 4)$ are given by the table below, where element Y_{ij} is in row i and column j.

0.485	0.752	0.869	0.982
0.369	1.254	0.703	1.455
5.2095	10.0677	22.9274	20.2153
23.3037	101.779	111.461	191.267
28.5132	111.8467	134.3884	211.4823

$$\underline{x}^o = (a, a, a, a, a, a, a, a)^T \qquad a = 0.1\ (0.2)\ 0.9,\ 1(1)\ 10$$

$$\underline{x}^* = (0.9000,\ 0.4500,\ 1.000,\ 8.000,\ 8.000,\ 5.000,\ 1.000,\ 2.000)^T$$

$F(\underline{x}^o)$ is shown for each of the fifteen starting points in Tables 2.1 - 2.4

$F(\underline{x}^*) = 0$

A solution with negative values of $\underline{x}$ exists at $\underline{x}^* = (0.8985, 0.9740, 11.65, 3.251, 6.711, -8.763, 1.251, -0.5251)^T$

This alternative solution is of no interest since it has no physical realisation. A logarithmic transformation of the variables was used in the optimization to ensure that only positive solution values could be found. This transformation introduces the local minimum of $F(\underline{x}) = 0.0548$ at

$$\underline{x} = (0.9014, 0.8910, 3.882, 5.3240, 10.65, 0.0, 1.089, 0.7033)^T$$

Problem 2 Meyer and Roth [42]

m=5; n=3

$$f_i(\underline{x}) = a_i x_1 x_3/(1 + a_i x_1 + b_i x_2) - y_i$$

$$\underline{a} = (1.0, 2.0, 1.0, 2.0, 0.1)^T$$

$$\underline{b} = (1.0, 1.0, 2.0, 2.0, 0.0)^T$$

$$\underline{y} = (0.126, 0.219, 0.076, 0.126, 0.186)^T$$

$\underline{x}^o = (10.39, 48.83, 0.74)^T$; $F(\underline{x}^o) = 0.0365$

$\underline{x}^* = (3.13, 15.16, 0.78)^T$; $F(\underline{x}^*) = 4.0_{10}^{-5}$

Problem 3 Rosenbrock [19]

m=2 n=2

$f_1(\underline{x}) = 10(x_2 - x_1^2)$

$f_2(\underline{x}) = 1 - x_1$

$\underline{x}^o = (-1.2, 1.0)^T \quad F(\underline{x}^o) = 24.2$

$\underline{x}^* = (1.0, 1.0)^T \quad F(\underline{x}^*) = 0.0$

Problem 4 Rosenbrock [19]

Identical to Problem 3 except that

$$\underline{x}^o = (-0.86, 1.14)^T; \quad F(\underline{x}^o) = 19.5$$

Problem 5 Meyer and Roth [42]

m=23; n=3

$f_i(\underline{x}) = x_3(\exp[-a_i x_1] + \exp[-b_i x_2]) - y_i$

$\underline{a} = (0, 0.6, 0.6, 1.4, 2.6, 3.2, 0.8, 1.6, 2.6, 4.0, 1.2, 2.0, 4.6,$
$3.2, 1.6, 4.2, 2.0, 3.2, 2.8, 4.2, 5.4, 5.6, 3.2)^T$

$\underline{b} = (0, 0.4, 1.0, 1.4, 1.4, 1.6, 2.0, 2.2, 2.2, 2.2, 2.6, 2.6, 2.8,$
$3.0, 3.2, 3.4, 3.8, 3.8, 4.2, 4.2, 4.4, 4.8, 5.0)^T$

$y_i = x_3^* (\exp[-a_i x_1^*] + \exp[-b_i x_2^*])$

$\underline{x}^o = (12.0, 1.0, 25.0)^T \quad ; \quad F(\underline{x}^o) = 216.0$

$\underline{x}^* = (14.3, 1.5, 20.1)^T \quad ; \quad F(\underline{x}^*) = 0.0$

Note that the values of y_i are set to give a minimum sum of squares of zero.

Problem 6 Meyer and Roth [42]

Identical to problem 5 except that the values of y_i are rounded off to 1 or 2 significant figures to give

$$\underline{y} = (4\ 0,\ 1\ 0,\ 5,\ 2.5,\ 2.5,\ 2.0,\ 1.0,\ 0.7,\ 0.8,\ 0.7,\ 0.4,\ 0.4,\ 0.3,$$
$$0.22,\ 0.2,\ 0.1,\ 0.05,\ 0.07,\ 0.03,\ 0.03,\ 0.03,\ 0.02,\ 0.01)^T$$

The solution then becomes

$$\underline{x}^* = (31.5,\ 1.51,\ 19.9)^T; \qquad F(\underline{x}^*) = 1.25$$

This solution is that quoted by Meyer and Roth. In fact, the value of $F(\underline{x}^*)$ is insensitive to changes in x_1^* and so this variable is not exactly determined by the optimization.

Problem 7 Meyer and Roth [42]

m=10; n=3

$$f_i(\underline{x}) = x_1 + x_2 \exp [a_i\ x_3] - y_i$$

$$\underline{a} = (1,\ 5,\ 10,\ 15,\ 20,\ 25,\ 30,\ 35,\ 40,\ 50)^T$$

$$y_i = x_1^* + x_2^* \exp [a_i\ x_3^*]$$

$$\underline{x}^o = (20.0,\ 2.0,\ 0.5)^T; \quad F(\underline{x}^o) = 2.1_{10}22$$

$$\underline{x}^* = (15.5,\ 1.2,\ 0.02); \quad F(\underline{x}^*) = 0.0$$

Note that the values of y_i are set to give a minimum sum of squares of zero.

Problem 8 Meyer and Roth [42]

Identical to Problem 7 except that the values of y_i are rounded off to 3 significant figures to give

$\underline{y}$ = (16.7, 16.8, 16.9, 17.1, 17.2, 17.4, 17.6, 17.9, 18.1, 18.7)^T

The solution then becomes

$\underline{x}^*$ = (15.67, 0.999, 0.022); $F(\underline{x}^*)$ = 0.006

Problem 9 Meyer and Roth [42]

m=16; n=3

$f_i(\underline{x}) = x_1 + \exp[x_2/(a_i + x_3)] - y_i$

$\underline{a}$ = (50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120, 125)^T

$\underline{y}$ = (34780, 28610, 23650, 19630, 16370, 13720, 11540, 9744, 8261, 7030, 6005, 5147, 4427, 3820, 3307, 2872)^T

$\underline{x}^o$ = (0.02, 4000, 250)^T; $F(\underline{x}^o) = 1.7_{10}9$

$\underline{x}^*$ = (0.0056, 6181.4, 345.2)^T; $F(\underline{x}^*)$ = 88

BIBLIOGRAPHY

[1] Euclid (-), Book IV.

[2] Widder, D.V. (1947) "Advanced Calculus", Prentice-Hall, New York, pp. 113-117.

[3] Huskey, D.H. (1976) "Chronology of Computing Devices", *IEEE Trans. Computers*, C-25, pp. 1190-1199

[4] Hitchcock, F.L. (1941) "The Distribution of a Product from Several Sources to Numerous Localities", *J. Maths. Physics,* 20, pp. 224-230.

[5] Dantzig, G.B. (1951) "Maximization of a Linear Function of Variables Subject to Linear Equalities", *in* "Activity Analysis of Production and Allocation," *ed.* Koopmans, T.C., Wiley, New York.

[6] Himmelblau, D.M. (1972), "Applied Nonlinear Programming", McGraw-Hill, New York.

[7] Kuhn, H.W. and Tucker, A.W. (1951) "Nonlinear Programming" *in* "Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability", *ed.* Neyman, J., University of California Press, Berkeley, pp. 481-493.

[8] Fiacco, A.V. and McCormick, G.P. (1968) "Nonlinear Programming : Sequential Unconstrained Minimization Techniques", Wiley, New York.

[9] Kowalik, J. and Osborne, M.R. (1968) "Methods for Unconstrained Optimization Problems", Elsevier, New York.

[10] Swann, W.H. (1972) "Direct Search Methods" *in* "Numerical Methods for Unconstrained Optimization", *ed.* Murray, W., Academic Press, London.

[11] Kiefer, J. (1957) "Optimal Sequential Search and Approximation Methods under Minimum Regularity Conditions", *SIAM J. Appl. Maths,* 5, pp. 105-136.

[12] Davies, D., Swann, W.H. and Campey (1964) "Report on the Development of a New Direct Search Method of Optimization", ICI Note 64/3.

[13] Powell, M.J.D. (1965) "A Method for Minimizing a Sum of Squares of Nonlinear Functions Without Calculating Derivatives", *Computer J.* 7, pp. 155-162.

[14] Fletcher, R., and Reeves., C.M. (1964) "Function Minimization by Conjugate Gradients", *Computer J.* 7, pp 149-154.

[15] Bard, Y. (1970) "Comparison of Gradient Methods for the Solution of Nonlinear Parameter Estimation Problems", *SIAM J. Numer Anal.*, 7, pp. 157-186.

[16] Murray, W., *ed.* (1972) "Numerical Methods for Unconstrained Optimization", Academic Press, London.

[17] Krolak, P. and Cooper, L. (1963) "An Extension of Fibonaccian Search to Several Variables," *CACM*, 6, pp. 639-641

[18] Hooke, R. and Jeeves, T.A. (1961) "Direct Search Solution of Numerical and Statistical Problems", *JACM*, 8, pp. 212-229.

[19] Rosenbrock, H.H. (1960), "An Automatic Method for Finding the Greatest or Least Value of a Function", *Computer J.*, 3, pp 175-184

[20] Bandler, J.W. and MacDonald (1969) "Optimization of Microwave Networks by Razor Search", *IEEE Trans. Microwave Theory and Techniques*, MTT-17, pp. 552-562.

[21] Nelder, J.A. and Mead, R. (1965) "A Simplex Method for Function Minimization", *Computer J*, 7, pp. 308-313.

[22] Fletcher, R. (1970) "A New Approach to Variable Metric Algorithms", *Computer J*, 13, pp. 317-322.

[23] Cauchy, M.A. (1847) "Méthode Générale pour la Résolution des Systèmes D'équations Simultanées", *Comptes Rendus de L'académie des Sciences*, 25, pp. 536-538.

[24] Greenstadt, J.L. (1970) "Variations of Variable Metric Methods", *Maths. Comp.*, 24, pp. 1-22.

[25] Murray, W. (1972) "Second Derivative Methods" *in* "Numerical Methods for Unconstrained Optimization", *ed.* Murray, W., Academic Press, London.

[26] Wilkinson, J.H. (1965) "The Algebraic Eigenvalue Problem", Oxford University Press, London.

[27] Gill, P.E., Murray, W. and Picken, S.M. (1972) "The Implementation of Two Modified Newton Algorithms for Unconstrained Optimization", NPL Report, NAC 24.

[28] Davidon , W.C. (1959) "Variable Metric Method for Minimization", AEC Research and Development Report, ANL-5990.

[29] Fletcher, R. and Powell, M.J.D. (1963) "A Rapidly Convergent Descent Method for Minimization", *Computer J.*, 6, pp. 163-168.

[30] Broyden, C.G. (1965), "A Class of Methods for Solving Non-Linear Simultaneous Equations", *Maths. Comput.*, 19, pp. 577-593.

[31] Gill, P.E. and Murray W. (1972) "Quasi-Newton Methods for Unconstrained Optimization", *JIMA*, 9, pp. 91-108.

[32] Barrodale, I. and Roberts F.D.K. (1974) "Solution of an Overdetermined System of Equations in the l_1 Norm", *CACM*, 17, pp. 319-320.

[33] Osborne, M.R. and Watson, G.A. (1969) "An Algorithm for Minimax Approximation in the Nonlinear Case", *Computer J.*, 12, pp. 63-68.

[34] Lill, S.A. (1976) "A Survey of Methods for Minimizing Sums of Squares of Nonlinear Functions", *in* "Optimization in Action", *ed.* Dixon, L.C.W., Academic Press, London.

[35] Dennis, Jr., J.E. (1973) "Some Computational Techniques for the Nonlinear Least Squares Problem", *in* "Numerical Solution of Systems of Nonlinear Algebraic Equations", *eds.* Byrne, G.D. and Hall, C.A., Academic Press, New York.

[36] Gauss, K.F. (1809) "*Theoria motus corporum coelestiam", Werke,* 7, pp. 240-254.

[37] Hartley, H.O. (1961) "The Modified Gauss-Newton Method for the Fitting of Nonlinear Regression Functions by Least Squares", *Technometrics*, 3, pp. 269-280.

[38] Levenberg, K. (1944), "A Method for the solution of Certain Non-Linear Problems in Least Squares", *Quart. Appl. Maths.*, 2, pp. 164-168.

[39] Marquardt, D.W. (1963), "An Algorithm for Least Squares Estimation of Nonlinear Parameters," *SIAM J. Numer. Anal.*, 11, pp. 431-441

[40] Goldfeld, S.M., Quandt, R.E. and Trotter H.F. (1966) "Maximisation by Quadratic Hill-Climbing", *Econometrica* 34, pp. 541-551

[41] Fletcher, R. (1971) "A Modified Marquardt Subroutine for Nonlinear Least Squares", UKAEA Report AERE-R6799, HMSO

[42] Meyer, R.R. and Roth, P.M. (1972) "Modified Damped Least Squares : An Algorithm for Non-Linear Estimation", *JIMA*, 9, pp. 218-233.

[43] Nash, J.C. (1977), "Minimizing a Non-Linear Sum of Squares Function on a Small Computer", *JIMA*, 19, pp. 231-237.

[44] Powell, M.J.D. (1968) "A FORTRAN subroutine for Solving Systems of Non-Linear Algebraic Equations", UKAEA Report AERE-R5947, HMSO

[45] Jones, A. (1970) "Spiral - A New Algorithm for Non-Linear Parameter Estimation Using Least Squares," *Computer J.*, 13, pp. 301-308.

[46] Wolfe, M.A. (1976) "Some Methods for Least Squares Estimation", *JIMA*, 18, pp. 219-236.

[47] McKeown, J.J. (1975) "Specialised Versus General-Purpose Algorithms for Minimising Functions That Are Sums of Squared Terms", *Math. Prog.*, 9, pp. 57-68.

[48] Dixon, L.C.W., Gomulka, J. and Hersom, S.E. (1976) "Reflections on the Global Optimization Problem", *in* "Optimization in Action", *ed.* Dixon, L.C.W., Academic Press, London.

[49] Price, W.L. (1977) "A Controlled Random Search Procedure for Global Optimisation", *Computer J.*, 20, pp. 367-370.

[50] Goldstein, A.A. and Price, J.F. (1971) "On Descent from Local Minima", *Maths. Comp.*, 25, pp. 569-574.

[51] Branin, Jr., F.H. (1972) "Widely Convergent Method for Finding Multiple Solutions of Simultaneous Nonlinear Equations", *IBM J. Res. Develop.*, 16, pp. 504-522.

[52] Cutteridge, O.P.D. (1973) "Electrical Network and General System Synthesis Using Optimisation Techniques", *in* "Optimisation and Design", *eds.* Avriel, M., Rijckaert, M. and Wilde, D.J., Prentice-Hall, London.

[53] Calahan, D.A. (1965) "Computer Design of Linear Frequency Selective Networks", *Proc. IEEE*, 53, pp. 1701-1706.

[54] Cutteridge, O.P.D. and Di Mambro, P.H. (1974) "Some Examples Demonstrating Feasibility of Evolutionary Approach to Linear Network Synthesis", *Electronics Letters*, 10, pp. 30-31.

[55] Cutteridge, O.P.D. and Krzeczkowski, A.J. (1975) "Improved Methods of Synthesizing Linear Networks by Coefficient Matching", *IEEE Trans. Circuits and Systems*, CAS-22, pp. 486-489.

[56] Cutteridge, O.P.D. (1971) "Numerical Experience with Some Two-part Programmes for the Solution of Non-linear Simultaneous Equations", University of Leicester Engineering Department Report 72-20.

[57] Phillips, D.A. (1974) "A Preliminary Investigation of Function Optimization by a Combination of Methods", *Computer J.*, 17, pp. 75-79

[58] Chien, H.H.Y (1972) "A Multiphase Algorithm for Single Variable Equation Solving", *JIMA*, 9, pp. 290-298.

[59] Cutteridge, O.P.D. and Dowson M. (1973) "Methods for the Solution of Nonlinear Simultaneious Equations Incorporating Some Variations of Levenberg's Technique", University of Leicester Engineering Department Report 73-13.

[60] Skwirzynski, J.K. (1970) Private communication to O.P.D. Cutteridge

[61] Ramsay, J.O. (1970) "A Family of Gradient Methods for Optimization", *Computer J.*, 13, pp. 413-417.

[62] Fox, L. and Mayers, D.F. (1968) "Computing Methods for Scientists and Engineers", Clarendon Press, Oxford.

[63] Braun, M. (1975) "Differential Equations and Their Applications", Springer-Verlag, New York.

[64] Dixon, L.C.W. (1972) "Quasi-Newton Algorithms Generate Identical Points", *Math. Prog.*, 2, pp. 383-387.

[65] Bowdler, H.J., Martin R.S., Peters, G. and Wilkinson, J.H. (1966) "Solution of Real and Complex Systems of Equations", *Num. Math.*, 8, pp. 217-234

[66] Dowson, M. (1976) "Multimodal Univariate Search Techniques and Their Application to Optimization Problems", M. Phil. Thesis, University of Leicester.

[67] Golub, G. (1965) "Numerical Methods for Solving Linear Least Squares Problems", *Num. Math.*, 7, pp. 206-216

[68] Ebers, J.J. and Moll J.L. (1954) "Large Signal Behaviour of Junction Transistors", *Proc. IRE*, pp. 1761-1772.

[69] Curtis, A.R., Powell, M.J.D. and Reid, J.K. (1974) "On the Estimation of Sparse Jacobian Matrices", *JIMA*, 13, pp. 117-119.

[70] Numerical Algorithms Group Program Library, IBM 360/370 Mark V Implementation, NAG Central Office, Oxford.

[71] Beale, E.M.L. (1968) "Mathematical Programming in Practice", Pitman, London.

[72] Gass, S.I. (1958) "Linear Programming", McGraw-Hill, New York.

[73] Hill, T.W. and Ravindran, A. (1975) "On Programming with Absolute-Value Functions", *JOTA*, 17, pp. 181-183.

[74] Spyropoulos, K., Kiountouzis, E. and Young, A. (1973) "Discrete Approximation in the L_1 Norm", *Computer J.*, 16, pp. 180-186.

[75] Beale, E.M.L. (1959) "On Quadratic Programming", *Naval Research Logistics Quarterly*, 6, pp. 227-243.

[76] Madsen, K. (1975) "An Algorithm for Minimax Solution of Overdetermined Systems of Non-linear Equations", *JIMA*, 16, pp. 321-328.

[77] Anderson, D.H. and Osborne, M.R. (1977) "Discrete Nonlinear Approximation Problems in Polyhedral Norms : A Levenberg-like Algorithm", *Num. Math.*, 28, pp. 157-170.

[78] Griffith, R.E. and Stewart, R.A. (1961) "A Non-Linear Programming Technique for the Optimization of Continuous Processing Systems", *Management Science*, 7, pp. 379-392.

[79] Dixon, L.C.W., *ed.* (1976) "Optimization in Action", Academic Press, London.

[80] Maxwell, R.W. (1974) Private communication to the author.

[81] Zienkiewicz, O.C. (1967) "The Finite Element Method", McGraw-Hill, London.

[82] Leckie, F.A. and Wojewódski, W. (1976) "Estimates of the Rupture Life of Structural Components Subjected to Proportional Cyclic Loading", *J. Mech. Phys. Solids*, 24, pp. 239-250.

[83] Kelly, D.A. (1976) "A Two-dimensional Model of Creep Cavitation Failure in Copper Subject to Biaxial Stress Systems at 250°C", *Metal Science*, 10, pp. 57-62.

[84] O'Regan, P.G. (1970) "Step Size Adjustment at Discontinuities for Fourth Order Runge-Kutta Methods", *Computer J*, 13, pp. 401-404

[85] Grant, J.A. and Hitchins, G.D. (1971) "An Always Convergent Minimization Technique for the Solution of Polynomial Equations", *JIMA*, 8, pp. 122-129

[86] Hayhurst, D.R. and Henderson J.T. (1977) "Creep Stress Redistribution in Notched Bars", *Int. J. mech. Sci.*, 19, pp. 133-146

[87] Lipo, T.A. and Krause, P.C. (1967) "Stability Analysis of a Reluctance-Synchronous Machine", *IEEE Trans. Power Apparatus and Systems*, PAS-86, pp. 825-834.

[88] Dinibütün, A.T. and Corbett, A.G. (1974) "Report of Work Done at Leicester University on a Collaborative Control Engineering Project with ICI", University of Leicester Engineering Department Report 74-17.

[89] Colquhoun, D. (1971) "Lectures on Biostatistics", Clarendon Press, Oxford.

[90] Cormack, R.M. and Lamb, J.F. (1977) Private communication to the author.

[91] Colquhoun, D. (1969) "A Comparison of Estimators for a Two-Parameter Hyperbola", *J. Royal Stat. Soc., Series C*, 18, pp. 130-140.

[92] Bell, M. and Pike, M.C. (1966) "Remark on Algorithm 178, Direct Search," *CACM*, 9, pp. 684-685

[93] Clayton, A.J. (1973). Private communication to the author.

[94] Cormack, R.M. (1971) "A Review of Classification", *J. Royal Stat. Soc, Series A*, 134, pp. 321-367.

[95] Hartigan, J.A. (1975) "Clustering Algorithms", Wiley, New York.

[96] Wishart, D (1969) "An Algorithm for Hierarchical Classifications", *Biometrics*, 25, pp. 165-170.

[97] Edwards, A.W.F. and Cavalli-Sforza, L.L. (1965) "A Method for Cluster Analysis", *Biometrics*, 21, pp. 362-375.

[98] Forgy, E.W. (1965) "Cluster Analysis of Multivariate Data : Efficiency Versus Interpretability of Classifications", *Biometrics*, 21, pp. 768-769.

[99] Fisher, W.D. (1958) "On Grouping for Maximum Homogeneity", *J. Amer. Stat. Ass.* 62, pp. 1159-1178.

[100] Gordon, A.D. (1975) Private communication to the author.

[101] Rosen, J.B. (1960) "The Gradient Projection Method for Nonlinear Programming, Part I, Linear Constraints", *J.SIAM*, 8, pp. 181-217

[102] Gordon, A.D. and Henderson J.T. (1977) "An Algorithm for Euclidean Sum of Squares Classification", *Biometrics*, 33, pp. 355-362

[103] CLUSTAN, 16 Kingsburgh Road, Murrayfield, Edinburgh EH12 6DZ

[104] Birks, H.J.B. (1973) "The Past and Present Vegetation of the Isle of Skye - a Palaeoecological Study", Cambridge University Press, London.

[105] Zoutendijk, G. (1966) "Nonlinear Programming : A Numerical Survey", *J. SIAM Control*, 4, pp. 194-210.

[106] MacLeod, A. (1976) Private communication to the author.

[107] Croydon, R. (1974) Private Communication to the author.

[108] Baden Fuller, A.J. (1961) "Waveguide Devices Containing Ferrites", AEI Report ER.4162.