THE USE OF SECOND DERIVATIVES IN APPLIED NUMERICAL OPTIMISATION

P. R. DIMMER

Thesis submitted for the degree of Doctor of Philosophy in the University of Leicester.

February 1979

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

ACKNOWLEDGMENTS

I wish to acknowlege the advice given by my supervisor, Dr. O. P. D. Cutteridge, the encouragement given by my colleague, Mrs. M. Dowson, and the understanding shown by my wife, all of which have contributed by no mean amount to the completion of this thesis.

PRD

STATEMENT OF ORIGINALITY AND THESIS AVAILABILITY

The accompanying thesis submitted for the degree of Doctor of Philosophy entitled "The use of second derivatives in applied numerical optimisation" is based on work conducted by the author in the Department of Engineering of the University of Leicester mainly during the period between October 1972 and October 1978.

All the work recorded in this thesis is original unless otherwise acknowledged in the text or by references. None of the work has been submitted for another degree in this or any other university.

This thesis may be made available for consultation, photo-copying and for use through libraries either directly or via the British Lending Library.

Signed ..... Date 13th February 1979

GLOSSARY OF SYMBOLS

All vectors are underlined; matrices are denoted by capital Roman letters; a superscript of τ indicates transpose.

An underlined symbol without a superscript of τ is a column vector e.g. $\underline{x}$ is a column vector and $\underline{x}^{\tau}$ is a row vector.

The i th component of $\underline{x}$ is denoted x_i ; the component of A in the i th row and j th column is denoted A_{ij} .

Superscripted parentheses are used to denote values at an iteration e.g. $\underline{x}^{(o)}$ is the initial value assigned to $\underline{x}$ and $\underline{x}^{(p)}$ is the value of $\underline{x}$ after the p th iteration.

Superscripted square brackets (e.g. $\underline{x}^{[q]}$) are used to denote evaluation within a main iteration.

A superscript of * means a solution value e.g. $\underline{x}_1^*$ means the value of $\underline{x}$ at the first solution; this is denoted $\underline{x}^*$ if there is only one solution.

Two norms are used:

$$\|\underline{x}\|_\infty = \max_{1\le i\le n} |x_i| \quad \text{and} \quad \|\underline{x}\|_2 = \left\{ \sum_{i=1}^{n} |x_i|^2 \right\}^{\frac{1}{2}}$$

where n is the number of components of $\underline{x}$.

The following commonly-used variables have the meanings given below.

B an approximation to the Hessian matrix;

d initial displacement for the transistor model problem;

f the problem objective function;

$\underline{g}$ the $n \times 1$ gradient vector of f : $\left(\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \ldots \frac{\partial f}{\partial x_n}\right)^{\tau}$;

H the $n \times n$ Hessian matrix of f defined by $H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$;

I the $n \times n$ unit matrix;

i,j,k,r general integer variables;

J the $m \times n$ Jacobian matrix of $\underline{s}$ defined by $J_{ij} = \frac{\partial s_i}{\partial x_j}$;

m the number of simultaneous equations or component functions in a sum of squares problem;

n the number of unknown variables;

p the iteration number except for Chapter V where it is the complex frequency variable;

q normally an evaluation number within an iteration;

S an approximation to the inverse Hessian matrix;

$\underline{s}$ the $m \times 1$ vector of component functions in a sum of squares problem;

$\underline{x}$ the $n \times 1$ vector of optimisation variables;

$\underline{\delta}$ an $n \times 1$ vector displacement of $\underline{x}$;

θ the objective function of the sub-problem in the second derivative sum of squares algorithms;

λ,μ scalar search parameters;

$\underline{\sigma}$ the $m \times 1$ vector of component functions in the sum of squares sub-problem of the second derivative algorithms;

$\underline{0}$ the zero vector (either $m \times 1$ or $n \times 1$).

CONTENTS

CHAPTER I

INTRODUCTION

One of the consequences of the development of electronic computers has been the growth of numerical optimisation theory and its application to a wide variety of problems in many disciplines. The objective in all such problems is to determine the optimal values of a set of variables by minimising (or maximising) a function of those variables. The function to be optimised, which is termed the objective function, may have some direct significance. For example, in an engineering environment the objective function might represent the weight, the strength or the cost of some component or structure. Alternatively the objective function might be artificially constructed to reflect an optimal state at its maximum or minimum value. If it is desired, for example, to fit a mathematical model to numerical data, a suitable choice might be the sum of the squares of the differences between the numerical data and the corresponding values predicted by the model. In his book on non-linear parameter estimation, Bard (1) lists a number of application areas, from nuclear physics to the design of artificial limbs, in which examples of this type occur. The data may be obtained from experimental observations, or derived from consideration of the desirable properties of the device being modelled.

The book edited by Avriel, Rijckaert and Wilde (2) contains optimisation problems from various branches of engineering: chemical, electrical, civil and mechanical. One application included (Cutteridge (3)) is the use of optimisation in electrical network design. Under the direction of Dr. O.P.D. Cutteridge, electrical network synthesis has been a research area at the University of Leicester for a number of years, early work being based on networks of fixed topology (e.g. Cutteridge (4)), and more recent developments allowing the evolution of a network from a basic structure (Cutteridge (5), Cutteridge and Di Mambro (6)).

This application is of the model-fitting variety where it is desired to match parameters to values obtained from design considerations. The methods developed rely heavily on optimisation techniques to attempt to solve, at various stages, a set of highly non-linear simultaneous equations. Experience has shown that the optimisation techniques which are most successful are combinations of two basic algorithms (Cutteridge (7),(8)), chosen for their complementary properties. The one normally used for obtaining final convergence is based on the Gauss-Newton method, which uses first partial derivatives of the individual functions forming the set of simultaneous equations. A second algorithm is necessary because the Gauss-Newton method will not converge if the initial estimate of the solution is too inaccurate.

It was against this background that the present research was undertaken. The main question to be considered was whether any benefit could be obtained from the use of second partial derivatives of the component functions. Although there exist algorithms which make use of the second partial derivatives of the objective function, to the author's knowledge there has been no published algorithm for the solution of non-linear simultaneous equations which makes use of the second derivatives of the component functions. Although the former type of method can be applied to such problems, there is evidence to suggest they are not as effective as the custom-designed methods.

Based on consideration of the next term in the Taylor series expansion beyond those terms used by the Gauss-Newton method, several new algorithms were developed and assessed on trial problems. The most difficult problems were taken from the electrical network design application, although several more simple problems from various sources were used during the research to verify hypotheses and to check programming. The algorithms were not written in any way which would bind them to a particular application but could be applied to any set of simultaneous equations, or

indeed to the more general minimisation of a sum of squares, provided that the second derivatives of the component functions are available. Thus the scope of the work is multi-disciplinary.

1.1. Thesis layout

Chapter II gives the derivation of the basic equations for a selection of optimisation methods. A complete survey of all methods was beyond the scope of this thesis so the content has been restricted to those methods which make use of first or higher order derivatives or which approximate derivatives by a numerical method using function values only. Even with this restriction the content of Chapter II does not pretend to be complete; however all algorithms used in this research and those closely related, are included. Chapter II also contains a description of some factors which influence the choice of an algorithm, and presents some conflicting views on these topics.

Chapter III presents some basic numerical results using three of the algorithms previously described. It aims to clarify some of the points of the previous chapter and illustrates the effect of using the same basic algorithm with different implementation details. The results also demonstrate the superiority of a sum of squares algorithm over a general function minimisation algorithm in the type of problem considered.

Chapter IV describes the development of second derivative sum of squares methods and gives the results of various trials which were conducted during that development on a transistor modelling problem. In Chapter V, the final methods were further tested on problems taken from the field of circuit design. The results of these two chapters show that the second derivative method is more powerful than its first derivative counterpart in that fewer iterations were required when convergence to the same solution occurred and that the former method was able to converge to a solution from less accurate estimates. In addition an example is given

to demonstrate that the extra range of convergence observed cannot necessarily be obtained by use of a general function minimisation method.

Chapter VI examines some possible extensions to the new algorithms including the use of alternative basic equations containing higher derivatives.

1.2. Computing facilities used

During the period of research (1972-1978), several computers have been used. Initial investigations were carried out on the University of Leicester's ICL-Elliott 4130 but by far the major part of the work was done on the Rutherford Laboratory ICL 1906A using the George IV operating system. Main access to the machine was via a MOP teletype terminal used for on-line editing and job submission, and a GEC 2050 remote job entry (RJE) station which was used to receive line printer output. The teletype and RJE station were sited at the University of Leicester and were connected to the mainframe via a Post Office Tariff 'T' line. Following the closure of the ICL 1906A at the Rutherford Laboratory and during periods when access via the above facility was not available, use was made of the ICL 1906A at the University of Nottingham.

All programs containing optimisation algorithms were written in Algol 60 to run in fully automatic batch mode. Use was made of the Numerical Algorithms Group (NAG) library of procedures for software for matrix inversion and for the provision of a variable metric algorithm.

Figures 4.3 to 4.5 were produced on the Rutherford Laboratory III FR80 microfilm recorder. Flow charts were produced on the computer aided design research group's configuration at Leicester, using the interactive graphical program FLOW (9).

1.3 Notation

The author would like to draw the reader's attention to the glossary of symbols situated at the beginning of this thesis and to add some remarks. A superscript consisting of an integer expression enclosed by parentheses indicates evaluation of the variable at that iteration, e.g. $\underline{x}^{(p+1)}$ means the value of $\underline{x}$ at the $(p+1)$th iteration. These superscripts may be attached to scalars, vectors or matrices but are sometimes omitted when no confusion can occur. Similarly, the independent variables of certain functions may also be omitted. Thus $\underline{g}(\underline{x}^{(p)})$ may be written $\underline{g}^{(p)}$ or merely $\underline{g}$.

CHAPTER II

DERIVATIVE-RELATED OPTIMISATION METHODS

The objective of this chapter is to present some of the methods which are applied to optimisation problems and to discuss some of the factors which influence the choice and implementation of an algorithm. In particular, methods for local unconstrained optimisation are considered.

Following some basic definitions and a short section to show the relevance of the subject matter, the basic equations and their derivation for a number of optimisation methods are given. The methods are categorised according to whether they were designed for general function optimisation or for sum of squares optimisation. They are further subdivided according to the order of analytical derivatives required.

There is an enormous number of such methods and a comprehensive survey was beyond the scope of this thesis. The methods selected for mention are those that have some bearing on the main research theme, i.e. those methods which make use of analytical derivatives or which attempt to approximate derivatives using function values only. Even with this restriction the content is incomplete. However all methods which were used in this research plus those closely related are included. Various publications can be consulted for more extensive surveys, e.g. Kowalik and Osborne (10), Powell (11),(12), Murray (13) and Lill (14).

It is apparent, and perhaps unavoidable considering the amount of literature on the subject, that some algorithms are adorned with multiple names and worse still, occasionally the same name is used for several different algorithms. This is especially true of methods associated with Newton. In naming these methods here, where the definitive reference has been unobtainable, the author has attempted to be consistent with the majority of workers in the field.

2.1. Basic definitions

The general, *global* optimisation problem may be regarded as the problem of finding the minimum value of a real function, the *penalty* or *objective function*, defined by one or more variables. There is no loss of generality here since the problem of finding the maximum value of a function is merely the equivalent of minimising its negative. It is assumed that the variables are real and continuous over the region of interest. This is in contrast to the discrete optimisation problem of integral variables.

Suppose there are n variables represented by the vector

$$\underline{x} = (x_1 \; x_2 \ldots x_n)^T$$

and f is the objective function defined for all $\underline{x}$ within the region of interest R.

A global minimum is defined to be any vector $\underline{x}^*$ within R which satisfies

$$f(\underline{x}^*) \leq f(\underline{x}) \quad \text{for all } \underline{x} \text{ within } R .$$

It is therefore possible to have more than one global minimum.
A *local* minimum is defined to be any vector $\underline{x}^*$ within R which satisfies

$$f(\underline{x}^*) \leq f(\underline{x}) \quad \text{for all } \underline{x} \text{ within } R^* \tag{2.1}$$

where R^* is a neighbourhood of $\underline{x}^*$ within R. If there is strict inequality in Equation (2.1), the minimum is sometimes called a *proper* local minimum (Kowalik and Osborne (10)) or a *strong* local minimum (Murray (15)). It is possible to have several local minima unless the objective function is convex over R, in which case there is only one local minimum which is also a global minimum.

If the components of $\underline{x}$ are permitted to take any real value, i.e. if R is equivalent to n-dimensional real space E^n, the problem is one of *unconstrained* optimisation. In practical problems, there is often a restriction on the possible values of component x's, in which case the problem is *constrained*. In their review of global optimisation methods, Dixon, Gomulka and Szegö (16) point out that use of a digital computer automatically places upper and lower bounds on the variable values and their definition of an unconstrained problem is modifed accordingly. However, this author would like to distinguish between the problem and the method of solution, believing that the definition of Dixon et al. is appropriate to the latter.

2.2. Relevance of local unconstrained optimisation

Since the advent of the digital computer much attention has been devoted to methods for solving the local unconstrained minimisation problem. The techniques developed have application outside their apparently limited area and continue to have a major role to play in both constrained and global optimisation studies.

In 1966, Box (17) gave a number of transformations which could be applied to minimisation variables thus giving the means of respecifying certain constrained problems so that they may be treated as unconstrained. Furthermore, certain more complex constraints can be effectively imposed by suitable definition of the objective function to which an optimisation method for unconstrained problems is applied. Lootsma (18) has reviewed such approaches.

Several algorithms for finding global minima require local optimisation methods, including the class of methods known as the multistart algorithm which is the most widely used global optimisation technique at the moment. This algorithm has been specified by Dixon (19) as follows.

Step 1: select $\underline{x}$ at random;

Step 2: start a local minimisation algorithm from $\underline{x}$ with a gradient termination criterion;

Step 3: test whether the final point is probably the global minimum and if so stop, otherwise return to Step 1.

Some examples of other global optimisation techniques which are not multistart algorithms but nevertheless either require, or are able to make use of, a local minimisation procedure are the method of Treccani, Trabattoni and Szegö, Evtushenko's method and Törn's clustering algorithm. The first two methods are described by Dixon et al. (16) and some experience with the use of Törn's method has been described by Gomulka (20).

2.3. Some methods for minimising a general function

Local minimisation of a function of unconstrained variables, where no special assumptions of the form of the function are made, is attempted by applying an algorithm which requires the evaluation of the function for prescribed values of the variables. In addition, the algorithm may require the first or second partial derivatives of the function with respect to the variables. The majority of such methods require a single initial estimate $\underline{x}^{(0)}$ of a minimum and from the supplied information construct a sequence of points (in n-dimensional space) $\underline{x}^{(1)}$, $\underline{x}^{(2)}$ ... which hopefully converge to a minimum $\underline{x}^*$. A notable exception to this general approach is Nelder and Mead's simplex algorithm (21) which requires $(n+1)$ initial points.

The simplest function that can have an unconstrained minimum is a positive definite quadratic function:

$$u(\underline{x}) = a + \underline{b}^T\underline{x} + \frac{1}{2}\underline{x}^T A\underline{x}$$

where A is a positive definitive symmetric matrix.

An iterative process for which there exists an integer r such that from any point $\underline{x}^{(o)}$ it is known that the minimum of $u(\underline{x})$ will be obtained in at most r steps is said to possess *quadratic termination.*

An algorithm is said to possess r *step second order convergence* if there exists a p_o and α such that

$$\| \underline{x}^{(p+r)} - \underline{x}^* \|_2 < \alpha \{ \| \underline{x}^{(p)} - \underline{x}^* \|_2 \}^2 \quad \text{for all} \quad p > p_o .$$

Most commonly used algorithms possess quadratic termination, while it is necessary to restrict the objective function in order to prove second order convergence for any algorithm (Dixon (22)).

2.3.1. Algorithms using second derivatives

It is appropriate to describe first one of the oldest optimisation techniques, the Newton-Raphson method, which had its foundations in the seventeenth century. It still has relevance today because it exhibits good terminal convergence properties. The method requires the evaluation of second derivatives of the objective function and is based upon the Taylor series expansion:

$$f(\underline{x} + \underline{\delta}) = f(\underline{x}) + \underline{\delta}^T \underline{g}(x) + \ldots$$

where $\underline{g}$ is the gradient vector of f.

Ignoring higher order terms, differentiating, and equating the first derivatives to zero, a necessary condition for a local minimum:

$$\underline{0} = \underline{g} + H\underline{\delta} \tag{2.2}$$

where H is the Hessian matrix of second partial derivatives of f.

Providing H is non-singular, the Newton-Raphson iteration is defined by:

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} + \underline{\delta}^{(p)} = \underline{x}^{(p)} - H^{(p)-1} \underline{g}^{(p)} . \tag{2.3}$$

Clearly this algorithm minimises the quadratic function $u(\underline{x})$ in one iteration. However, on a general function the algorithm may not converge and as is apparent from observing that Equation (2.2) is derived from the condition for a stationary point only, the sequence defined by Equation (2.3) may converge to a non-minimum point.

For these reasons, it is common practice to treat $-H^{-1}\underline{g}$ as a search direction only, so that Equation (2.3) is redefined:

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} - \lambda^{(p)} H^{(p)-1} \underline{g}^{(p)} \qquad (2.4)$$

where the scalar $\lambda^{(p)}$ may be chosen merely to ensure a decrease in the objective function or to locate the minimum of $\phi(\lambda) = f(\underline{x}^{(p)} - \lambda H^{(p)-1}\underline{g}^{(p)})$ to a certain accuracy.

Equations (2.3) and (2.4) cannot be used if $H^{(p)}$ is singular. One possible remedy, described by Murray (23), is a modification of Equation (2.3) on the lines of the Marquardt-Levenberg method which was originally intended for the minimisation of a sum of squares and is described in Section 2.4.1. When applied to a general function, the method becomes:

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} - (H^{(p)} + \lambda^{(p)} Q^{(p)})^{-1} \underline{g}^{(p)} \qquad (2.5)$$

where $Q^{(p)}$ is some specified matrix and $\lambda^{(p)}$ is a scalar chosen so that $(H^{(p)} + \lambda^{(p)} Q^{(p)})$ is positive definite and $f(\underline{x}^{(p+1)}) < f(\underline{x}^{(p)})$. One possible choice for $Q^{(p)}$ is the unit matrix I.

This approach, including a similar choice for $Q^{(p)}$, has been used by Cutteridge (8) who derives his basic equation from extension of the steepest descent method (see next section) to include second derivatives. From the equation:

$$\delta_j = -\mu \left\{ \frac{\partial f}{\partial x_j} + \sum_{i=1}^{n} \delta_i \frac{\partial^2 f}{\partial x_i \partial x_j} \right\} \qquad (j=1,2,\ldots,n) \qquad (2.6)$$

which in vector-matrix form becomes

$$\underline{\delta} = - \mu(\underline{g} + H\underline{\delta})$$

we have

$$\underline{\delta} = - (H - \lambda I)^{-1} \underline{g}$$

where $$\lambda = - \frac{1}{\mu} \quad .$$

In the form of Equation (2.5) this becomes:

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} - (H^{(p)} - \lambda^{(p)} I)^{-1} \underline{g}^{(p)} \qquad (2.7)$$

which is merely Equation (2.5) with $Q^{(p)} = - I$.

Equation (2.7) forms the basis of Cutteridge's gradient-descent method, further details of which are given later.

Other alternatives to Equation (2.3), which still retain the use of second derivatives, have been proposed by Greenstadt (24), and by Fiacco and McCormick (25).

2.3.2. Algorithms using first derivatives

Mention has already been made of the steepest descent method which is another old method having been devised by Cauchy (26) in 1847. It requires the gradient vector $\underline{g}$ of first derivatives of the objective function and defines a sequence of points by

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} - \lambda^{(p)} \underline{g}^{(p)}$$

where $\lambda^{(p)}$ is chosen to minimise the function of one variable $\phi(\lambda) = f(\underline{x}^{(p)} - \lambda \underline{g}^{(p)})$. The algorithm does not possess quadratic termination and it is generally accepted that in its basic form the method is unsuitable for practical problems as convergence is often intolerably slow.

Of more use in a practical situation is the class of first derivative methods comprising the conjugate gradient algorithms, where the sequence of points is defined by:

$$\begin{aligned} \underline{x}^{(p+1)} &= \underline{x}^{(p)} - \lambda^{(p)}\underline{y}^{(p+1)} \\ \underline{y}^{(p+1)} &= \underline{g}^{(p)} - \alpha^{(p)}\underline{y}^{(p)} \end{aligned} \qquad (2.8)$$

where $\lambda^{(p)}$ is the univariate search parameter and $\alpha^{(p)}$ generates different members of the class.

On a quadratic function, the first n directions $\underline{y}$ are mutually conjugate with respect to the Hessian matrix, and therefore if $\lambda^{(p)}$ is chosen to minimise $\phi(\lambda) = f(\underline{x}^{(p)} - \lambda\underline{y}^{(p+1)})$, the minimum of the quadratic function will be located in at most n iterations.

The most well known algorithm of this class is by Fletcher and Reeves (27), who in 1964 defined the parameter of Equations (2.8) as follows:

$$\alpha^{(p)} = \begin{cases} 0 & \text{for } p = 0,\ n+1,\ 2(n+1),\ \ldots \\ \dfrac{\underline{g}^{(p)\tau}\underline{g}^{(p)}}{\underline{g}^{(p-1)\tau}\underline{g}^{(p-1)}} & \text{otherwise.} \end{cases}$$

If the substitution $\alpha^{(p)} = -\dfrac{\nu^{(p)}\lambda^{(p-1)}}{\lambda^{(p)}}$ is made in Equations (2.8) then we have:

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} - \lambda^{(p)}\underline{g}^{(p)} + \nu^{(p)}\underline{\delta}^{(p-1)}$$

where $$\underline{\delta}^{(p)} = \underline{x}^{(p+1)} - \underline{x}^{(p)} .$$

Defining the values of $\lambda^{(p)}$, $\nu^{(p)}$ to be those that minimise $\phi(\lambda,\nu) = f(\underline{x}^{(p)} - \lambda\underline{g}^{(p)} + \nu\underline{\delta}^{(p-1)})$ yields the memory gradient method of Miele and Cantrell (28). The introduction of a two-dimensional search

represented a departure from the normal trend in algorithm design which was further extended by Cragg and Levy (29), who included a k-dimensional search with $k \leq n$ in their supermemory gradient method. An iteration of this method is defined by:

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} - \lambda^{(p)}\underline{g}^{(p)} + \sum_{i=1}^{\min(p,k)} \nu^{(p,i)}\underline{\delta}^{(p-i)} .$$

A further important class of methods and one which has attracted a great deal of attention in recent years, is formed by those algorithms which attempt to approximate Equation (2.4) by the use of first derivatives only. These algorithms are characterised by the following formulae:

$$\begin{aligned} \underline{x}^{(p+1)} &= \underline{x}^{(p)} - \lambda^{(p)} S^{(p)} \underline{g}^{(p)} \\ S^{(p+1)} &= S^{(p)} + F(S^{(p)}, \underline{\delta}^{(p)}, \underline{h}^{(p)}) \end{aligned} \qquad (2.9)$$

where

$$\begin{aligned} \underline{\delta}^{(p)} &= \underline{x}^{(p+1)} - \underline{x}^{(p)} \\ \underline{h}^{(p)} &= \underline{g}^{(p+1)} - \underline{g}^{(p)} \end{aligned}$$

and F is a matrix function for updating S, an approximation to the inverse Hessian.

An initial value $S^{(o)}$ of S must be provided in addition to $\underline{x}^{(o)}$. The general term for such algorithms is variable metric, but if $S^{(o)}$ is a positive definite matrix, and the condition:

$$S^{(p+1)}\underline{\delta}^{(p)} = \underline{h}^{(p)}$$

is observed, the algorithm may be classified as quasi-Newton. The derivation of this condition is from observing that for a quadratic function:

$$\underline{\delta}^{(p)} = H\underline{h}^{(p)} .$$

The first version of Equation (2.9) was proposed in 1959 by Davidon (30), who defined:

$$F(S,\underline{\delta},\underline{h}) = \frac{\underline{\delta}\,\underline{\delta}^{\tau}}{\underline{\delta}^{\tau}\underline{h}} - \frac{S\underline{h}\,\underline{h}^{\tau}S}{\underline{h}^{\tau}S\underline{h}} \quad . \tag{2.10}$$

Four years later, Davidon's algorithm was revised by Fletcher and Powell (31) and in this form proved to be one of the best first derivative algorithms of the decade. In 1967, Broyden (32) suggested a family of formulae:

$$F(S,\underline{\delta},\underline{h}) = \frac{\underline{\delta}\,\underline{\delta}^{\tau}}{\underline{\delta}^{\tau}\underline{h}} - \frac{S\underline{h}\,\underline{h}^{\tau}S}{\underline{h}^{\tau}S\underline{h}} + \alpha\,\underline{y}\,\underline{y}^{\tau} \tag{2.11}$$

where

$$\underline{y}(S,\underline{\delta},\underline{h}) = \frac{S\,\underline{h}}{\underline{h}^{\tau}S\underline{h}} - \frac{\underline{\delta}}{\underline{\delta}^{\tau}\underline{h}}$$

and α is a scalar which generates the family.
Clearly the Davidon formula corresponds to $\alpha = 0$. Dixon (33) has shown that for a general function the sequence of points generated by Equations (2.9) and (2.11) is independent of α given perfect univariate minimisation of $\phi(\lambda) = f(\underline{x}^{(p)} - \lambda S^{(p)}\underline{g}^{(p)})$ at each stage.

A whole host of formulae have been proposed, but in 1970 Huang (34) showed that almost all of them could be generated from:

$$F(S,\underline{\delta},\underline{h}) = \frac{\rho\,\underline{\delta}\,\underline{z}^{\tau}}{\underline{z}^{\tau}\underline{h}} - \frac{S\underline{h}\,\underline{y}^{\tau}}{\underline{y}^{\tau}\underline{h}}$$

where $$\underline{y}(S,\underline{\delta},\underline{h}) = \alpha\underline{\delta} + \beta S\underline{h}$$

and $$\underline{z}(S,\underline{\delta},\underline{h}) = \zeta\underline{\delta} + \eta S\underline{h} \quad .$$

Here there are three independent parameters, ρ and the ratios $\alpha:\beta$, $\zeta:\eta$.

In addition, S may be asymmetric. This family includes Broyden's class, which is obtained by setting ρ to 1 and restricting other constants to keep S symmetric, leaving one free parameter.

The inclusion of a k-dimensional search from consideration of directions given by previous iterations to form supermemory variants of quasi-Newton algorithms, and indeed other methods, has been examined by Wolfe and Viazminsky (35).

Of the various quasi-Newton formulae proposed, one that is gaining acceptance is the complementary Davidon-Fletcher-Powell formula, obtained by writing:

$$\alpha = \underline{h}^{\tau} S \underline{h}$$

in Equation (2.11). The name is derived from the fact that this matrix updating formula is equivalent to:

$$B^{(p+1)} = B^{(p)} + \frac{\underline{h}^{(p)} \underline{h}^{(p)\tau}}{\underline{h}^{(p)\tau} \underline{\delta}^{(p)}} - \frac{B^{(p)} \underline{\delta}^{(p)} \underline{\delta}^{(p)\tau} B^{(p)}}{\underline{\delta}^{(p)\tau} B^{(p)} \underline{\delta}^{(p)}} \qquad (2.12)$$

where B is an approximation to the Hessian itself. Note that the formula given by Equation (2.12) is equivalent to the Davidon formula with S, $\underline{h}$ and $\underline{\delta}$ replaced by B, $\underline{\delta}$ and $\underline{h}$ respectively. This formula was proposed by Fletcher (36) and in his computer algorithm was applied in the H-update form.

However, in 1972 Gill and Murray (37) proposed that quasi-Newton methods should be implemented in the form of Equation (2.12) by using a triangular factorisation of an approximation to the Hessian, thereby reducing the adverse effects of rounding error and enabling advantage to be taken of a sparse Hessian matrix. Details of computer programs and results obtained using this method appear in a report by Gill, Murray and Pitfield (38). Algorithms based on those written by Gill et al. using Equation (2.12) have now replaced the Davidon-Fletcher-Powell algorithms in the NAG library.

2.3.3. Algorithms using function values only

The design of a minimisation algorithm which does not require derivative information explicitly has attracted the attention of many researchers and consequently there are a number of algorithms of this type. These will not be reviewed here, but it is appropriate to mention two algorithms based on a similar approach. Given that an expression for the evaluation of the objective function is available, it is always possible to calculate derivatives numerically and apply one of the algorithms of the previous section. A more formal procedure was established by Stewart (39), who based his method on the Davidon variable metric algorithm (Equations (2.9) and (2.10)) and prescribed when the calculation of derivatives should use the central difference formula as opposed to the quicker but less accurate forward difference formula. Gill and Murray (37), whilst having reservations about Stewart's choice of step length, similarly describe a strategy for incorporating numerical derivatives in their variable metric procedure.

2.4. Some methods for minimising a sum of squares

Some methods designed specifically for the local minimisation of a function comprising a sum of squares of component functions of unconstrained variables will now be considered. Assuming there are m component functions, the objective function may be defined by :

$$f(\underline{x}) = \sum_{i=1}^{m} \{s_i(\underline{x})\}^2 . \qquad (2.13)$$

In addition to the values of the objective function and its derivatives, there are now available values of the component functions $s_i(\underline{x})$ and their derivatives.

The minimisation of $f(\underline{x})$ in Equation (2.13) is closely related to the solution of a set of a simultaneous equations:

$$\underline{s}(\underline{x}) = (s_1(\underline{x}) \; s_2(\underline{x}) \; \ldots \; s_n(\underline{x}))^T = \underline{0}$$

where the value of the objective function at any stage is equal to the sum of the squares of the residuals of the simultaneous equations. However, whereas there is a solution $\underline{x}^*$ to the simultaneous equations if and only if $f(\underline{x}^*) = 0$, this is not a necessary condition for $\underline{x}^*$ to be the solution of the general sum of squares minimisation problem. The methods to be considered are most suitable for the simultaneous equations problem, though they may be applied to the minimisation problem even when the solution does not yield a zero objective function. It is a requirement that $m \geq n$.

The following two sections discuss some methods requiring and not requiring first derivatives of the component functions respectively. As far as the author is aware, there is no published algorithm which is specifically designed for sum of squares minimisation and makes use of second (or higher) derivatives of the component functions.

2.4.1. Algorithms using first derivatives

Consider the Taylor series expansion of the component functions:

$$s_i(\underline{x}+\underline{\delta}) = s_i(\underline{x}) + \sum_{j=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j} \delta_j + \ldots \qquad (i = 1,2,\ldots,m) \quad (2.14)$$

If higher order terms are ignored, this may be expressed in vector-matrix form:

$$\underline{s}(\underline{x}+\underline{\delta}) = \underline{s}(\underline{x}) + J\underline{\delta}$$

where J is the Jacobian matrix of first derivatives.

Equating the right-hand side to zero, we have:

$$\underline{0} = \underline{s}(\underline{x}) + J\underline{\delta} \ . \tag{2.15}$$

If $m = n$, and providing J is non-singular, we may write

$$\underline{\delta} = - J^{-1}\underline{s}(\underline{x}) \ . \tag{2.16}$$

If however $m > n$, Equation (2.16) is not appropriate. The approach then is to attempt to find the least-squares solution of Equation (2.15) i.e. a value of $\underline{\delta}$ which will minimise the residuals of Equation (2.15). This is obtained by writing

$$\frac{\partial}{\partial \delta_k}\left[\sum_{i=1}^{m}\left(s_i(\underline{x}) + \sum_{j=1}^{n}\frac{\partial s_i(\underline{x})}{\partial x_j}\delta_j\right)^2\right] = 0 \qquad k = 1,2,\ldots,n$$

whence

$$\sum_{i=1}^{m}\left(s_i(\underline{x}) + \sum_{j=1}^{n}\frac{\partial s_i(\underline{x})}{\partial x_j}\delta_j\right)\frac{\partial s_i(\underline{x})}{\partial x_k} = 0 \qquad k = 1,2,\ldots,n$$

which in vector-matrix form gives

$$J^{\tau}\underline{s} + J^{\tau}J\,\underline{\delta} = \underline{0} \ . \tag{2.17}$$

Thus if $J^{\tau}J$ is non-singular, we have

$$\underline{\delta} = - (J^{\tau}J)^{-1}J^{\tau}\underline{s} \ . \tag{2.18}$$

If $m = n$ and J is non-singular then clearly this reduces to Equation (2.16).

Equation (2.18) forms the basis of the Gauss-Newton method, which defines:

$$\begin{aligned} \underline{\delta}^{(p)} &= - \left(J^{(p)\tau}J^{(p)}\right)^{-1}J^{(p)\tau}\underline{s}^{(p)} \\ \underline{x}^{(p+1)} &= \underline{x}^{(p)} + \underline{\delta}^{(p)} \end{aligned} \tag{2.19}$$

where $J^{(p)}$ and $\underline{s}^{(p)}$ are functions of $\underline{x}^{(p)}$.

As in the Newton-Raphson iteration, it is common to introduce a scalar λ to Equations (2.17) so that

$$\lambda J^{\tau}\underline{s} + J^{\tau}J\underline{\delta} = \underline{0}$$

whence

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} - \lambda^{(p)}(J^{(p)\tau}J^{(p)})^{-1} J^{(p)\tau}\underline{s}^{(p)} \quad . \qquad (2.20)$$

Provided a point at which $J^{(p)\tau}\underline{s}^{(p)} = \underline{0}$ has not been reached and $J^{(p)\tau}J^{(p)}$ is non-singular, a value of $\lambda > 0$ which will reduce the objective function can be found (see Appendix I); in this way divergence may be prevented. However, there is still the problem of singularity of $J^{(p)\tau}J^{(p)}$. This can be avoided by the use of generalised inverses (Fletcher (40)) which are applicable whatever the rank of the Jacobian. For appropriate ranks, this method yields Equations (2.16) or (2.19). Fletcher also proposed a similar modification to Equation (2.3) where the Hessian may be singular.

An alternative method for avoiding the singularity problem is by suitably modifying the matrix to be inverted. The earliest method for the Gauss-Newton procedure was proposed by Levenberg (41) in 1944, and later, Marquardt (42) independently proposed a similar method. The Marquardt-Levenberg method is characterised by the equation:

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} - (J^{(p)\tau}J^{(p)} + \lambda^{(p)}Q^{(p)})^{-1} J^{(p)\tau}\underline{s}^{(p)} \qquad (2.21)$$

where $Q^{(p)}$ is a positive definite matrix and $\lambda^{(p)}$ is a scalar. A common choice of $Q^{(p)}$ is the unit matrix I so that as $\lambda^{(p)} \to \infty$ the direction given is $-J^{(p)\tau}\underline{s}^{(p)}$ which, as can be seen from Equation (2.13), is in the direction of steepest descent.

It is appropriate to mention two variations on this theme, both of which interpolate between the direction of steepest descent and the direction given by the Gauss-Newton point when it exists. One method is due to Powell and is described in the next section, the other is due to Jones (43).

Jones proposed that the locus of possible points $\underline{x}^{(p+1)}(\lambda)$ in the Marquardt - Levenberg method be replaced by a spiral passing through the Gauss-Newton point, through $\underline{x}^{(p)}$, and tangential to the steepest descent direction at $\underline{x}^{(p)}$. In this way a search of the possible new points could be conducted without the necessity for re-solving Equation (2.21). In the event of singularity of $J^{(p)T}J^{(p)}$, a Marquardt-Levenberg modification is used to define a point to replace the Gauss-Newton point.

2.4.2. Algorithms using function values only

Approaches similar to those used for the minimisation of a general function are used in sum of squares problems. In 1965, Broyden (44) proposed formulae for updating the inverse Jacobian matrix of Equation (2.16) for the solution of non-linear simultaneous equations. This approach is the counterpart of the variable metric method, but because in this case a matrix of first derivatives is replaced the method uses function values only. Broyden saw his method as being an alternative to other techniques for reducing the amount of computation which had been suggested earlier, such as using a constant Jacobian for a certain number of steps.

The hybrid method due to Powell (45), (46) uses a Jacobian update formula which maintains non-singularity. By use of the Jacobian approximation the method obtains an approximation to the Gauss-Newton point which is used directly to define $\underline{x}^{(p+1)}$ providing the norm of the correction vector is sufficiently small. If it is not, the new point $\underline{x}^{(p+1)}$ is taken from the line of steepest descent or the line joining the approximate Gauss-Newton point and the predicted minimum along the line of steepest descent. The actual point selected is dependent on the required norm of the correction vector.

Finally, the direct use of difference formulae for first derivatives in methods for sum of squares minimisation has been tackled by Brown and Dennis (47). They found the exclusive use of the forward difference

representation to be adequate.

2.5. Relationship between Newton-Raphson and Gauss-Newton

Although Newton-Raphson and Gauss-Newton were described above under different headings, the former being a general function minimisation method and the latter a method for a sum of squares, they are quite closely related. For example, it is quite possible to apply a sum of squares method to a general function minimisation by equating expressions for the first derivatives of the function to zero, thereby creating a set of simultaneous equations. If the Gauss-Newton method, in the form of Equation (2.16), was applied to these derivative equations, the method would be precisely the same as applying Newton-Raphson to the general function in the first place.

It is also possible to apply a general function minimisation method to a sum of squares function thereby ignoring the special form of the function. If Newton-Raphson is applied thus, the resulting equations are not the same as the Gauss-Newton equations.

Writing Equation (2.17) in summation form gives:

$$\sum_{i=1}^{m} s_i \frac{\partial s_i}{\partial x_k} + \sum_{j=1}^{n} \delta_j \sum_{i=1}^{m} \frac{\partial s_i}{\partial x_j} \frac{\partial s_i}{\partial x_k} = 0 \qquad (k = 1,2,\ldots,n) \; . \qquad (2.22)$$

From Equation (2.13) we have:

$$\frac{\partial f}{\partial x_k} = 2 \sum_{i=1}^{m} s_i \frac{\partial s_i}{\partial x_k} \quad \text{and} \quad \frac{\partial^2 f}{\partial x_j \partial x_k} = 2 \sum_{i=1}^{m} \left(\frac{\partial s_i}{\partial x_j} \frac{\partial s_i}{\partial x_k} + s_i \frac{\partial^2 s_i}{\partial x_j \partial x_k} \right) . \qquad (2.23)$$

Substituting the above in Equation (2.2) gives

$$\sum_{i=1}^{m} s_i \frac{\partial s_i}{\partial x_j} + \sum_{j=1}^{n} \delta_j \sum_{i=1}^{m} \left(\frac{\partial s_i}{\partial x_j} \frac{\partial s_i}{\partial x_k} + s_i \frac{\partial^2 s_i}{\partial x_j \partial x_k} \right) = 0 \quad (k = 1,2,\ldots,n). \qquad (2.24)$$

Equations (2.22) and (2.24) differ by the term involving second derivatives and therefore it is often said that the Gauss-Newton method approximates the second derivatives of the objective function by:

$$\frac{\partial^2 f}{\partial x_j \partial x_k} \doteq 2 \sum_{i=1}^{m} \frac{\partial s_i}{\partial x_j} \frac{\partial s_i}{\partial x_k} \quad . \tag{2.25}$$

However, the inference that Gauss-Newton is an approximation of Newton-Raphson is, in the author's view, incorrect. An approximate method is generally taken to mean one that gives inferior results to those given by a more precise method. The results of the next chapter show that on certain problems this is not the case with Gauss-Newton and Newton-Raphson, and therefore the two methods should be considered to be distinct.

Finally it is appropriate to mention that since the Marquardt-Levenberg methods are based on the two Newton methods, the relationship (2.25) also exists between the general function version and the sum of squares version (cf. Equations (2.5) and (2.21)).

2.6. Desirable properties of an optimisation method

Before assessing the suitability of an optimisation method for a particular problem or range of problems, it is necessary to consider the desirable properties of the method in a practical rather than mathematical sense. This question has been considered by Himmelblau (48), who has presented one of the most extensive comparisons of optimisation algorithms for minimising a general function; 15 problems were used to compare a total of 26 implementations from 15 basic algorithms. Himmelblau states that it is generally accepted that an algorithm's robustness, i.e. its ability to solve a range of problems, is the primary criterion to be used in the assessment of a general purpose algorithm.

This should be the case, yet it would appear from published material that a comparison of successful performance on relatively simple problems is given far more emphasis. Furthermore, such comparisons often do not give the computer time taken to termination, though in defence of this it must be said that the actual central processor time used by a program cannot be readily calculated on many machines, something that was recog-

nised by Lootsma (49) in 1972 and is still apparent today (see Section 4.7).

In practical situations the concern is only that type of problem currently being posed. It is therefore the ability of the algorithm to solve a restricted range of problems that is of primary importance rather than its suitability to the general purpose role. Unfortunately practical engineering problems, such as those described later, are generally more difficult than those problems used in published assessments, which are therefore of limited use.

However, assuming suitable algorithms have been identified, their efficiency in terms of computation time for the given problem becomes relevant. This can only be calculated from previous assessments if they included computation times, not only for the whole run, but for individual steps of the problem solution such as function evaluation time, derivative evaluation time etc. The number of function evaluations or iterations is also necessary, but without additional information is inadequate. In order to complete the calculation of estimated computation time on the user's machine, it is also necessary to know the relative computer performances, for which the report by Verstege and Wichmann (50) is useful.

But the algorithm's success rate is of primary importance and its efficiency is of secondary importance. The development of faster computers has caused a shift in the relative importance of these two characteristics from the latter to the former and will continue to do so.

One further point which is often overlooked should be mentioned here, namely performance of an algorithm on problems which it is unable to solve. If it is possible to recognise that the algorithm is failing then the adoption of an alternative strategy is enabled without further useless calculations. This ability may be regarded to be of tertiary importance.

From the mathematical viewpoint, Wolfe (77), (78) derived conditions for a general algorithm of the form:

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} + \lambda^{(p)}\underline{\delta}^{(p)}$$

either to converge to a stationary point or to yield $f(x) \to -\infty$, when applied to certain (loosely restricted) functions. These conditions have been restated by Dixon (79) in a more simple form:

> on a well behaved function, the above algorithm will satisfy the ternination criterion $||\underline{g}(x)|| < \varepsilon_o$ providing that on a regular subsequence of iterations the following three conditions hold:
>
> (i) the direction of search $\underline{\delta}$ has a significant component in the negative gradient direction;
>
> (ii) the step taken is bounded away from zero;
>
> (iii) the reduction in the objective function is bounded away from zero;
>
> and providing that on other iterations the objective function value does not increase.

However, Wolfe's conditions do not guarantee that the limit point of the algorithm is a solution of the problem under consideration. In a minimisation problem the point reached may not be a minimum and if the problem is the solution of a set of simultaneous equations, even a limit point which is a minimum need not solve the equations. This may be seen from Equation (2.23) where

$$\underline{g} = 2J^T\underline{s} \quad .$$

Thus $\underline{g} = \underline{0}$ gives m-rank(J) non-trivial independent solutions for $\underline{s}$. Nevertheless, it would be expected that algorithms which always satisfy Wolfe's conditions have a head start in attaining the property of robustness though it is possible that a degradation of efficiency could result.

2.7. Factors influencing algorithm robustness and efficiency

The identification of the desirable properties of an optimisation algorithm raises the question as to which factors influence the attainment of such properties. The next sections discuss three factors which effect robustness and efficiency, namely:

(i) the use of derivative information;

(ii) general function minimisation or sum of squares minimisation;

(iii) the univariate search type.

While the first two topics effect the choice of a method, the third effects its implementation. The second assumes of course that the objective function can be expressed as a sum of squares.

All three topics have been subject to conflicting views in the past and some of these will be given. Wright and Cutteridge (51) have stated opposing views on all three topics and on four others in addition: variable transformations, multimodality problems, sampling methods and error function definitions.

2.7.1. The use of derivative information

The case for using first and higher order derivatives, as stated by Wright and Cutteridge, is that the more information there is available at any stage of an optimisation procedure, the better can decisions be made. Two points are made in support of the case against. First, derivative information remote from the solution is far too local and second, in highly non-linear space the Taylor series approximations, on which many derivative methods are based, have little mathematical meaning.

Apart from the possibility that the use of derivative information is ineffective, two other reasons for avoiding its use have been given. The first is the amount of storage required (Ramsay (52)) and the second is

that the calculation of analytical derivatives is sometimes a laborious task (Broyden (44)). However, the first objection should become less of a problem since the reducing cost of computer hardware has meant that larger core storage is feasible. Butlin (53), (54) has reported the production of random access stores as large as 10^{15} bits. If the second objection is really valid, the calculation of numerical derivatives is always a possibility. However, this leads to further controversy. Ramsay is wary of the stability and accuracy of numerical derivatives, an opinion emphasised by Wozny (55) who states the view that no slope information is better than erroneous slope information. On the other hand Jones (43) would even prefer the use of numerical derivatives to analytical derivatives unless the latter are markedly easier to compute than the original function.

The question on the value of the use of derivative information can therefore be posed in two parts:

(i) to what order should analytical derivatives be used?

(ii) should a method be selected which approximates derivatives, either numerically or by use of an updating formula, to an order of one or more higher than those provided analytically?

These questions require answers from the point of view of robustness and speed.

An examination of Himmelblau's table of results, which presents an ordering of the algorithms tested for the two main characteristics discussed in Section 2.6 reveals that the top five algorithms used analytical derivatives, though the use of analytical derivatives did not guarantee a higher placing. Stewart's method, the only algorithm tested which used numerical derivatives, was placed midway.

The construction of a hierarchy of methods is open to question on such points as problem weighting and failure interpretation, but nevertheless, Himmelblau's comparison shows that in practical problems, whilst the use of derivative information shows merit, its mere inclusion is insufficient to guarantee either improved robustness or improved speed.

Unfortunately Himmelblau's survey did not include second derivative methods. However, there is evidence elsewhere (e.g. Fiacco and McCormick (25)) to suggest that they can be extremely beneficial. Indeed, Murray (23) recommends their use unless the computation time for the second derivatives is extremely large compared with that for the function and gradient.

2.7.2. General function minimisation or sum of squares minimisation

In 1966, Box (17) compared several algorithms by applying them to some problems which could be expressed as sums of squares with zero residuals at the solution. One of the conclusions reached was that when solving simultaneous non-linear equations it was better to apply an algorithm designed for this type of problem in preference to doing an ordinary function minimisation. The derivation of the Gauss-Newton method given in Section 2.4.1 would tend to support this hypothesis. However, the major factor which casts doubt on the extension of this hypothesis for problems with non-zero residuals is the "second derivative approximation" approach which was described in Section 2.5.

Some recent work by McKeown (56) has clarified the situation. From Equation (2.23), the Hessian matrix for a sum of squares problem may be written in the form:

$$H = 2(J^T J + W)$$

where W is a matrix whose components are defined by:

$$W_{jk} = \sum_{i=1}^{m} s_i \frac{\partial^2 s_i}{\partial x_j \partial x_k} .$$

McKeown presents numerical evidence with theoretical basis that the rate of convergence of algorithms based on Equation (2.21) is dependent on the largest absolute eigenvalue of $(J^T J)^{-1} W$. When this quantity is large, general function minimisation should be preferred.

Clearly W is the zero matrix if all the s_i are zero, such as at the solution of a set of simultaneous equations, where McKeown's theory suggests that the use of Gauss-Newton type methods is preferable. However, generally the quantity required to judge which type of method to apply is unknown.

Some work by Betts (57) also has relevance here. His method builds up an estimate E of W in much the same way as the quasi-Newton approximations. The search direction of the algorithm is either the Gauss-Newton direction or the approximation to the Newton-Raphson direction obtained by the replacement of W by E. The latter is used when it is suspected that the algorithm has reached the neighbourhood of the solution provided that sufficient iterations have passed to accumulate a realistic estimate of W. Although this strategy is not in full alignment with McKeown's theory, the results quoted for Betts' hybrid method are good for both zero and non-zero residual problems.

2.7.3. The univariate search type

Many of the algorithms presented earlier require or are improved by a univariate search which aims to reduce the objective function. There are several questions involved here, the main one being the accuracy to which the search should be conducted. The opposing views for a linear search expressed by Wright and Cutteridge are:

(i) remote from the solution, where search directions are of questionable meaning, the computation of accurate linear searches is inefficient;

(ii) accurate linear searches reduce instability and can be used to indicate multimodality.

While it seems reasonable to assume that the use of an over-accurate univariate search would involve needless calculation, a more surprising result appears in the survey by Box who attributed certain instances of failure to attempts to locate a univariate minimum too precisely. The implication is that without proper safeguards, over-accurate univariate minimisation can effect the robustness of an algorithm.

An alternative to univariate minimisation is the acceptance of the first reduction in the objective function encountered. This for example is the method adopted by Jones in his spiral sum of squares algorithm and by Betts in his algorithm.

The form of the linear search in variable metric algorithms has created a great deal of interest, and comparative studies have been done by Dixon (58) and by Sargent and Sebastian (59), who also considered the search accuracy in the Fletcher-Reeves conjugate gradient algorithm. Dixon's investigations included termination of the univariate search once a minimum had been bracketed as well as termination on the satisfaction of more complex tests. Sargent and Sebastian considered a termination criterion of the form:

$$|\lambda^{[q]} - \lambda^{[q-1]}| < \varepsilon |\lambda^{[q-1]}|$$

where $\lambda^{[q-1]}, \lambda^{[q]}$ are successive values of the search parameter within an iteration and ε is a suitable small constant.

The balance of evidence from both papers was that accurate univariate searches were undesirable. However, it must be emphasised that these conclusions were reached from consideration of specific algorithms only and apparently from the point of view of computational efficiency rather than robustness. Univariate minimisation is still highly regarded in some quarters.

If minimisation is to be attempted to any accuracy, the question of non-unimodality must also be considered. The most common attitude is to apply a search which assumes unimodality thus avoiding a local minimum selection procedure unless non-unimodality is discovered by chance. In contrast to this method, the gradient-descent method of Cutteridge (Equation (2.7)) attempts to find most local minima in its specialised univariate search. In terms of the general correction vector $\underline{\delta}$, Cutteridge's basic equation is:

$$\underline{\delta} = - (H - \lambda I)^{-1} \underline{g} \ . \qquad (2.26)$$

The entire range given by real values of λ is considered, thus including as possible directions for $\underline{\delta}$, the Newton-Raphson direction and in the limits as $\lambda \to \pm \infty$, the directions of steepest ascent and steepest descent. Clearly Equation (2.26) will give large components of $\underline{\delta}$ near values of λ which are eigenvalues of H. In practice, the components of $\underline{\delta}$ are limited so that their absolute values do not exceed a prescribed amount, thus defining $\underline{\delta}$ for all real values of λ. Hence, the objective function is expressed as a multimodal function of λ, the global minimum of which is sought. Local minima may also be used in the complete method for solving simultaneous equations (see Section 4.1).

It is interesting to note that with H of Equation (2.26) replaced by $J^T J$, assuming the problem is one of minimising a sum of squares, the above gradient-descent method becomes Levenberg's method, even to the point of following Levenberg's original recommendation of finding the minimum of $\phi(\lambda) = f(\underline{x} - (H - \lambda I)^{-1} \underline{g})$. Such a version has also been investigated by Cutteridge.

Although correction component limiting is necessary in the gradient-descent method, other research done at Leicester (e.g. Cutteridge and Dowson (60)) has indicated that it is a worthwhile aid to convergence of the main algorithm when applied at each univariate search. For a given

(positive) component limiting factor δ_{max}, two interpretations have been used:

$$\text{(i)}\quad y_i = \begin{cases} \delta_i(\lambda) & \text{if} \quad \| \underline{\delta}(\lambda) \|_\infty \leq \delta_{max} \\ \dfrac{\delta_{max}\delta_i(\lambda)}{\| \underline{\delta} \|_\infty} & \text{if} \quad \| \underline{\delta}(\lambda) \|_\infty > \delta_{max} \end{cases}$$

$$\text{(ii)}\quad y_i = \begin{cases} \delta_i(\lambda) & \text{if} \quad |\delta_i(\lambda)| \leq \delta_{max} \\ \dfrac{\delta_{max}\delta_i(\lambda)}{|\delta_i(\lambda)|} & \text{if} \quad |\delta_i(\lambda)| > \delta_{max} \end{cases} \qquad (2.27)$$

where y_i is the limited correction component. The limiting can be applied during the univariate search or having completed the search although the former would intuitively appear to be preferable. The introduction of derivative discontinuities by the application of limiting techniques has been demonstrated and discussed by Dowson (61) who devised improvements to the univariate search of the Cutteridge gradient-descent algorithm.

2.8. Combining optimisation methods

It is probably unrealistic to imagine that any one optimisation method can fulfil all requirements on a particular range of problems and therefore, in an attempt to improve performance, the effect of combining two or more methods into one algorithm has been investigated by several researchers. In addition to those methods of this category previously mentioned, Phillips (62) has considered a combination of non-derivative methods for general minimisation. Performance was assessed using the criterion of the number of function evaluations, computing time not being available, on problems in which all the individual methods converged to the solution. The conclusion was that the combination was generally slower than its component methods, due mainly to the calculations performed to select the most appropriate method to adopt.

More extensive and more successful work has been done by Cutteridge (3),(7),(8) and by Cutteridge and Dowson (60) whose two-part combination methods have been applied to the simultaneous equations problem and have been largely based on the Gauss-Newton* method. From a point sufficiently close to the solution Gauss-Newton exhibits rapid convergence but on difficult problems is unable to converge from a general starting point. The idea conceived by Cutteridge was to use another method to attempt to find, from a general starting point, a point from which Gauss-Newton can successfully take over. Several methods have been tried for this role, including Marquardt-Levenberg variations, conjugate gradients and quasi-Newton methods. One of the most successful of these appears to be the Marquardt-Levenberg based method of Section 2.3.1. This method, as well as most of the others considered for the first part of the algorithm, is a general function minimisation method.

2.9. Discussion

The first part of this chapter has presented factual information on some methods for local unconstrained optimisation while the latter part has attempted to demonstrate that there are different viewpoints on aspects of the subject which still exist today. In general function minimisation the use of quasi-Newton methods is extremely popular; it is suspected that this is so even when second derivatives are available. Yet the advice of Murray and the results of Cutteridge and Fiacco and McCormick indicate that the use of second derivatives can be beneficial in many instances. Furthermore, the amount of computer storage required to retain second derivative values is of reducing significance.

---

*In the references cited, the method is often referred to as "Newton-Raphson", but according to the definitions given here the method is in fact Gauss-Newton.

In sum of squares minimisation there are some similarities between Betts' algorithm and Cutteridge's two-part algorithms, but whereas Betts prefers the use of a sum of squares method remote from the solution and the use of a general function minimisation method to give final convergence, Cutteridge prefers the reverse. These two authors also appear to have different attitudes towards the desirability of accurate univariate minimisation in their algorithms. The overall trend would appear to be to eliminate sub-problem minimisation, though the memory methods, the latest developments of which are quite recent, also oppose this tendency.

The trend towards combination optimisation methods would appear to be justifiable since there exist certain difficult problems on which no one basic method is able to exhibit rapid or even satisfactory convergence at all stages. The question now posed is whether the combination of two methods is sufficient.

CHAPTER III

BASIC NUMERICAL RESULTS

In order to clarify some of the aspects covered in Chapter II basic numerical exercises were carried out on certain test problems which could be defined by a set of non-linear simultaneous equations. The various Newton methods were of particular interest, with special regard to the effect that the use of correction limiting and different search types had on the robustness of the algorithms and the respective convergence ranges of the general function and sum of squares minimisation methods when applied to the aforementioned problems. The author felt that insufficient evidence had been presented, either internally or in published papers, to give a clear picture on these topics. The results produced also provided a basis for comparison of the performances of the developed algorithms which are described in the next chapter.

The two main algorithms used for the present exercise were Newton-Raphson (NR) and Gauss-Newton (GN), though at a later stage it was thought that it would be useful to compare the performance of a quasi-Newton algorithm with these, to which end the Gill-Murray-Pitfield (GMP) method was used on selected problems. Also at this stage the conventional steepest descent algorithm was run on some problems which confirmed its poor convergence properties; these results are not presented.

3.1. The example problems

The examples selected were chosen to give a range of problems of various degrees of difficulty. Each set of simultaneous equations had at least one known solution. Thus it was possible to tackle all the problems as the minimisation of a sum of squares with zero residuals at the solution, or as a general function minimisation.

The problems are presented by giving the components $\underline{s}(\underline{x})$, where it is desired to solve the set of simultaneous equations:

$$\underline{s}(\underline{x}) = \underline{0} \ .$$

The objective function in the minimisations was always defined as follows:

$$f(\underline{x}) = \sum_{i=1}^{m} \{s_i(\underline{x})\}^2 \quad .$$

3.1.1. Rosenbrock's function

Rosenbrock's function is a problem widely quoted in optimisation studies and is given by the objective function defined above (with $m = 2$) where:

$$s_1 = 10(x_2 - x_1^2)$$
$$s_2 = 1 - x_1 \quad .$$

The single solution is $\underline{x}_1^* = (1,1)^\tau$.

3.1.2. Modified Rosenbrock's function

The second example was a modification of Rosenbrock's function designed to introduce further solutions whilst not increasing the degree of the components of $\underline{s}$ beyond 2. The component functions were defined by:

$$s_1 = 10(x_2^2 - x_1^2)$$
$$s_2 = 1 - x_1^2 \ .$$

The four solutions are:

$$\underline{x}_1^* = (1,1)^\tau \ , \quad \underline{x}_2^* = (1,-1)^\tau \ , \ \underline{x}_3^* = (-1,1)^\tau \quad \text{and} \quad \underline{x}_4^* = (-1,-1)^\tau \ .$$

3.1.3. Higher degree single-solution equation pair (HDS)

The third example, abbreviated HDS, was designed to give a single solution from a pair of simultaneous equations of higher degree:

$$s_1 = 2x_1^3 x_2 - x_2^3$$
$$s_2 = x_1 x_2 - 8 \quad .$$

The solution is: $\underline{x}^* = (2,4)^\tau$.

3.1.4. Higher degree multi-solution equation pair (HDM)

The next example (HDM) was one that had been used by a colleague in some earlier work:

$$s_1 = 2x_1^3 x_2 - x_2^3$$
$$s_2 = 6x_1 - x_2^2 + x_2 \quad .$$

The three solutions of this set of equations are:

$\underline{x}_1^* = (2,4)^\tau$, $\underline{x}_2^* = (0,0)^\tau$ and $\underline{x}_3^* = (1.465,-2.507)^\tau$ approx.

The Jacobian is singular at $\underline{x}_2^*$.

3.1.5. Miele's function

Miele's function, which is given by Cragg and Levy (29), is defined as the objective function comprising the sum of squares of the components of $\underline{s}$ where

$$s_1 = (\exp(x_1) - x_2)^2$$
$$s_2 = 10(x_2 - x_3)^3$$
$$s_3 = \tan^2(x_3 - x_4)$$
$$s_4 = x_1^4 \quad .$$

Solutions to this problem are given by $(0, 1, 1, 1 + k\pi)^\tau$ where k is an integer. The Jacobian is singular at these points.

3.1.6. Transistor model problem

The first engineering test problem used in this research was originally supplied by the Marconi Company (63) and concerned the d.c. modelling of transistors based on the extended Ebers-Moll model (64). This describes the transistor in terms of a network which attempts to reproduce the forward and inverse current gains. The functions to be considered are effectively the net junction currents evaluated at sets of measurement points $(Y_{1i}, Y_{2i}, Y_{3i}, Y_{4i}, Y_{5i})$, with $i = 1, 2, 3, 4$ in the example presented below.

The problem was to solve the set of simultaneous equations in eight unknowns, given by:

$$s_i = x_3(1-x_1x_2)\left[\exp\{x_4(Y_{1i}-Y_{3i}x_6.10^{-3}-Y_{5i}x_7.10^{-3})\}-1\right] - Y_{5i} + Y_{4i}x_2$$

$$s_{i+4} = \frac{x_1x_3}{x_2}(1-x_1x_2)\left[\exp\{x_5(Y_{1i}-Y_{2i}-Y_{3i}x_6.10^{-3}+Y_{4i}x_8.10^{-3})\}-1\right] - Y_{5i}x_1 + Y_{4i}$$

for $i = 1,2,3,4$,

where

$$Y_{11} = 0.485,\ Y_{21} = 0.369,\ Y_{31} = 5.2095,\ Y_{41} = 23.3037$$
$$Y_{12} = 0.752,\ Y_{22} = 1.254,\ Y_{32} = 10.0677,\ Y_{42} = 101.779$$
$$Y_{13} = 0.869,\ Y_{23} = 0.703,\ Y_{33} = 22.9274,\ Y_{43} = 111.461$$
$$Y_{14} = 0.982,\ Y_{24} = 1.455,\ Y_{34} = 20.2153,\ Y_{44} = 191.267$$

and

$$Y_{5i} = Y_{3i} + Y_{4i} \quad \text{for} \quad i = 1,2,3,4 .$$

One solution to the above is given by:

$$\underline{x}_1^* = (0.9,\ 0.45,\ 1.0,\ 8.0,\ 8.0,\ 5.0,\ 1.0,\ 2.0)^T .$$

In the context of electrical network design, where the parameter values must be positive, $\underline{x}_1^*$ is the only known solution. However, the eight equations presented above do have at least one other solution, given by:

$$\underline{x}_2^* \doteq (0.8985,\ 0.9740,\ 11.65,\ 3.251,\ 6.711,\ -8.764,\ 1.251,\ -0.5251)^T \ .$$

3.2. Algorithm implementation details

The basic equations for GMP, NR and GN were given in Chapter II. An Algol procedure for the GMP method was accessed from the NAG library. Where there was a choice of parameter values for the procedure, the recommendations of NAG (65) were followed where applicable. Successful convergence was indicated by a test on $\| \underline{g} \|_2$, the value of which was required to be less than a prescribed amount. A value of 10^{-6} was generally used. The other termination criterion encountered was the completion of a prescribed number of function evaluations. 500 was specified, although the NAG recommendation was many fewer, viz. $20\,n$.

The NR and GN methods were programmed by the author. A refinement was made to the basic NR method to ensure that the downhill direction at the current point was chosen to define the new point. Thus Equation (2.4) was modified as follows:

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} - \alpha^{(p)} \lambda^{(p)} H^{(p)-1} \underline{g}^{(p)} \qquad (3.1)$$

where $\alpha^{(p)} = \text{sign}\ (\underline{g}^{(p)T} H^{(p)-1} \underline{g}^{(p)})$.

Hence only positive values of λ needed to be considered in the region of $\lambda = 0$.

Equation (2.20) formed the basis of the GN implementation, though because in all the problems considered the Jacobians were square matrices, the non-generalised form was also used, i.e.

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} - \lambda^{(p)} J^{(p)-1} \underline{s}^{(p)} \quad . \qquad (3.2)$$

When $m = n$, Equations (2.20) and (3.2) are analytically equivalent, but rounding errors may give different computed results.

3.2.1. Correction vector adjustment in NR and GN

Equations (3.1) and (3.2) include the search parameter λ which, as was discussed in Section 2.7.3, provides a means of scaling the full correction vector. Four methods of scaling were examined in these trials:

(i) no adjustment

(ii) bisection

(iii) bracketing

(iv) minimisation.

The second means of adjusting the correction vector, by limiting the size of the correction components, was investigated in conjunction with the last scaling method only.

Method (i), use of the full correction vector, simply required $\lambda^{(p)}$ to be set to 1 in Equations (3.1) and (3.2). No reduction in the objective function was enforced.

The other three methods did enforce this restriction and therefore required to examine $\phi(\lambda) = f(\underline{x}^{(p)} + \underline{\delta}^{(p)}(\lambda))$ for various values of λ where

$$\text{for NR :} \quad \underline{\delta}^{(p)}(\lambda) = -\alpha^{(p)}\lambda H^{(p)-1}\underline{g}^{(p)}$$

$$\text{and for GN :} \quad \underline{\delta}^{(p)}(\lambda) = -\lambda J^{(p)-1}\underline{s}^{(p)}.$$

To denote values of λ calculated within an iteration of the main optimisation algorithm, a superscript within square brackets is used.

The bisection method consisted of successively halving λ until a value $\lambda^{[j]}$ was found such that

$$\phi(\lambda^{[j]}) < \phi(0) \ .$$

The first value tried was $\lambda^{[o]} = 1$.

In method (iii), a search designed to bracket a univariate minimum was used. It was based on the Fibonacci sequence and was a modified version of one used by other research workers at Leicester. The requirement was to find values $\lambda^{[i]}$, $\lambda^{[j]}$, $\lambda^{[k]}$ of λ such that

$$\lambda^{[i]} < \lambda^{[j]} < \lambda^{[k]}$$

$$\phi(\lambda^{[j]}) < \phi(\lambda^{[i]}) \quad \text{and} \quad \phi(\lambda^{[j]}) < \phi(\lambda^{[k]}) \ .$$

$\lambda^{[j]}$ would then have been used to define the new point.

The first value of λ was chosen thus:

$$\lambda^{[o]} = \min(1/3, \lambda^{(p)} , \lambda_c^{(p)}/3)$$

where $\lambda^{(p)}$ was the selected value of λ on the previous iteration (defaulting to $1/3$ if there was no previous iteration) and $\lambda_c^{(p)}$ was the value of λ for which component correction limiting (see below) would first occur. If necessary, further values of $\phi(\lambda)$ would be evaluated by reducing λ until a value $\lambda^{[i]}$ was found such that $\phi(\lambda^{[i]})$ was less then $\phi(0)$. Thereafter, values of λ were generated. according to the Fibonacci sequence until a minimum had been bracketed.

Method (iv) took this process a stage further by attempting to locate a minimum to a prescribed accuracy by use of a quadratic fit algorithm devised by Cutteridge and Henderson (see Dowson (61)). This algorithm assumes a unimodal function and attempts to prevent slow convergence by restricting the quadratic interpolation points to lie within the central 80% of the region forming the bracket. The line minimisation was terminated when

$$|\lambda^{[j]} - \lambda^{[i]}| < \varepsilon|\lambda^{[j]}| \quad \text{and} \quad |\lambda^{[k]} - \lambda^{[j]}| < \varepsilon|\lambda^{[j]}|$$

where ε is the accuracy constant (0.01 was the value used).

Where correction vector component limiting was imposed it was according to Equation (2.27). Limiting was imposed within the search, so that the search function was redefined:

$$\phi(\lambda) = f(\underline{x}^{(p)} + \underline{y}(\lambda)) \ .$$

where
$$y_i(\lambda) = \begin{cases} \delta_i(\lambda) & \text{if } |\delta_i(\lambda)| \leq \delta_{max} \\ \dfrac{\delta_{max}\delta_i(\lambda)}{|\delta_i(\lambda)|} & \text{if } |\delta_i(\lambda)| > \delta_{max} \end{cases}$$

for $i = 1,2,\ldots,n$ and δ_{max} is the correction limit.

For bracketing purposes, $\lambda_c^{(p)}$ was defined:

$$\lambda_c^{(p)} = \frac{\delta_{max}}{\| \underline{\delta}^{(p)}(1) \|_\infty} \ .$$

3.2.2. Termination criteria for NR and GN

The criteria used for termination of the NR and GN algorithms were largely based on those developed by Cutteridge et al. A total of six were used, though some were never evoked. In the following the notation $\Delta^{(p)} = \| \underline{\delta}^{(p)}(1) \|_\infty$ is used.

The six termination criteria were as follows.

T1: maximum modulus correction component too small,

i.e. $\Delta^{(p)} < \varepsilon_1$

where ε_1 is a constant (10^{-6} was the value used) .

This condition tests for convergence. Whereas for GN with $m = n$ such convergence, if it occurs, must be to a solution of the problem expressed as a set of simultaneous equations, this is not the case for NR where the above condition relates to the size of $\| \underline{g}^{(p)} \|_\infty$. It

is quite possible to have $\underline{g} = \underline{0}$ and $\underline{s} \neq \underline{0}$. At such a point the Jacobian must be singular because of the identity $\underline{g} = 2J^T\underline{s}$.

T2: scaling parameter too small,

i.e. $\lambda^{(p)} < \dfrac{\varepsilon_2}{\Delta^{(p)}}$

where ε_2 is a constant (10^{-10} was the value used).

This is an additional convergence criterion included as a safeguard. Clearly the algorithms are unable to continue when λ is very small.

T3: maximum modulus correction component too large,

i.e. $\varepsilon_3\Delta^{(p)} > \Delta^{(o)}$

where ε_3 is a constant (0.01 was the value used).

This criterion tests for divergence.

T4: too many successive increases in the rate of increase of the (same) maximum modulus component,

i.e. $\Delta^{(p-i)} = |\delta_k{}^{(p-i)}|$

for $i = 0,1,\ldots,(r+2)$ and any (constant) k in the range $k = 1,2,\ldots,n$ and

$\Delta^{(p-j)} - \Delta^{(p-j-1)} > \Delta^{(p-j-1)} - \Delta^{(p-j-2)} > 0$

for $j = 0,1,\ldots,r$ where r is an integer constant (2 was the value used).

Again, this is a test for divergence.

T5: failure of matrix inversion.

In these optimisation algorithms no attempt was made to cater for singular matrices and so the computation was terminated in the event of failure of the matrix inversion procedure, which was selected from the NAG library.

T6: too many iterations.

A limit of the number of iterations was always specified as a precaution to prevent the excessive use of computer time.

Termination criteria T2, T3 and T4 are predictions of singularity of the matrix to be inverted and have been extensively tested by Henderson (80) who showed that considerable computer time could be saved by terminating the algorithm when one of these criteria were satisfied rather than waiting for the inversion procedure to fail, assuming that no attempt to continue from a point yielding a singular matrix was to be made. This was the case in the trials presented here. It could be argued that by modifying the NR and GN algorithms at such points or by using a more robust method based on NR and GN, such as those described in Chapter II, termination due to matrix singularity could have been avoided. This was not done because the context in which GN has been used for electrical engineering problems at the University of Leicester makes the basic form described here more appropriate.

The GN algorithm is generally used as one component of a two-part algorithm because of the good terminal convergence properties. If GN fails to reach a solution then control is returned to the method forming the other component (see Section 4.1). A more robust GN-based method could not maintain rapid convergence along paths distant from the solution and therefore if used it would merely be doing the job of the second

component. Thus a basic form of GN is used. If this method approaches a point where J^TJ is singular, the unscaled optimisation variable corrections become large indicating that the process is attempting to move outside the region of solution feasibility or desirability. Rather than attempt to continue the process from such a point it is preferred to return control to the other method at its point of exit.

It should therefore be borne in mind that the methods being considered for the terminal convergence role which are described in this and later chapters are basic in that they do not attempt to overcome the singularity problem and hence could be made more robust. The presented comparisons of algorithms of this type are fair in that the handicap is common to all.

3.3. Results obtained

For each of the six problems of Section 3.1, the optimisation methods used and results obtained are presented. The abbreviations N , Bi , Br and M are used to indicate the four correction vector adjustment methods: no adjustment, bisection, bracketing and minimisation respectively. Thus GN/M means Gauss-Newton with univariate minimisation. Correction limiting was only used on the transistor model problem.

In the tables that follow, one function evaluation is the evaluation of the set of $\underline{s}$ components plus the corresponding objective function. The number of derivative evaluations is not explicitly stated but may be deduced from the algorithm type and the number of iterations required. Termination criteria for GMP are denoted by 'S', meaning successful termination effected by the gradient test, and by 'F' indicating failure because the prescribed number of function evaluations was reached. Termination criteria codes for NR and GN refer to those of Section 3.2.2.

3.3.1. Rosenbrock's function

Seven methods, namely NR/N , NR/Bi , NR/M , GMP , GN/N , GN/Bi and GN/M were used on Rosenbrock's function from the standard starting point $\underline{x}^{(o)} = (-1.2, 1.0)^T$. All methods were successful. The results are given in Table 3.1. Of the three types of scaling method that were applied to NR and GN , use of the full correction vector proved to be by far the quickest.

Method	Termination criterion satisfied	Final objective function	No. of iterations	No. of function evaluations
NR/N	1	2.1×10^{-20}	7	7
NR/Bi	1	8.7×10^{-21}	22	29
NR/M	1	1.9×10^{-21}	14	125
GMP	S	2.3×10^{-13}	32	102
GN/N	1	0.0	3	3
GN/Bi	1	0.0	11	33
GN/M	1	2.1×10^{-16}	16	111

Table 3.1 Rosenbrock's function

3.3.2. Modified Rosenbrock's function

The same seven methods were applied to the modified Rosenbrock's function from the starting point $\underline{x}^{(o)} = (-30,5)^T$. The results are given in Table 3.2. Although all of the Newton methods eventually satisfied termination criteria T1 , it was found that two of them converged to $(-\sqrt{1/101},0)^T$ which is not a solution of the simultaneous equations but which yields $\underline{g} = \underline{0}$. Fewer iterations, though not necessarily less time, were needed for the univariate minimisation methods.

Method	Termination criterion satisfied	Approx. final point (x_1, x_2)	Final objective function	No. of iterations	No. of function evaluations
NR/N	1	$(-\sqrt{1/101},0)$	9.9×10^{-1}	19	19
NR/Bi	1	$(-\sqrt{1/101},0)$	9.9×10^{-1}	19	19
NR/M	1	(-1,-1)	1.8×10^{-14}	8	81
GMP	S	(-1,1)	2.8×10^{-9}	15	66
GN/N	1	(-1,1)	2.4×10^{-12}	9	9
GN/Bi	1	(-1,1)	2.4×10^{-12}	9	9
GN/M	1	(1,1)	1.3×10^{-17}	7	68

Table 3.2 Modified Rosenbrock's function

3.3.3. HDS problem

Three methods, GN/N , GN/Bi and GN/M , were applied to the higher degree single-solution equation pair from the starting points $(5,5)^T$, $(50,50)^T$ and $(500,500)^T$. The results, which are presented in Table 3.3, illustrate the dangers of univariate minimisation in this problem. From two of the starting points the univariate minimisation quickly reduced x_2 to zero, yielding an objective function value of 64 and a singular Jacobian. In one trial the bisection method also converged to a point where the Jacobian was nearly singular.

Starting point $\underline{x}^{(o)T} = (x_1, x_2)$	Method	Termination criterion satisfied	Final objective function	No. of iterations	No. of function evaluations
(5,5)	GN/N	1	8.7×10^{-17}	7	7
(5,5)	GN/Bi	1	8.7×10^{-17}	7	7
(5,5)	GN/M	1	0.0	19	172
(50,50)	GN/N	1	1.4×10^{-11}	12	12
(50,50)	GN/Bi	1	0.0	13	15
(50,50)	GN/M	5	6.4×10^{1}	3	59
(500,500)	GN/N	1	1.5×10^{-11}	22	22
(500,500)	GN/Bi	2	5.6×10^{1}	22	265
(500,500)	GN/M	5	6.4×10^{1}	3	61

Table 3.3 HDS problem

3.3.4. HDM problem

Four methods were used on the HDM problem: NR/N , NR/M , GN/N and GN/M. The results from two starting points, $(5,5)^T$ and $(500,500)^T$, are given in Table 3.4. Although NR/M terminated due to singularity in both cases, it has to all intents and purposes converged, giving a lower objective function that in some cases where termination criterion T1 was satisfied. For all methods ultimate convergence to $(0,0)^T$, when it occurred, was very slow due to singularity of the Jacobian and Hessian at this point.

3.3.5. Miele's function

Five methods were used on Miele's function: NR/Bi , NR/M , GMP , GN/Bi and GN/M. The starting point used by Cragg and Levy was $(1,2,2,2)^T$. However, both the Jacobian and Hessian are singular at this point so for NR and GN two steepest descent iterations were used to move away from the starting point and the standard optimisation algorithms used thereafter. Near the solution the matrices again became singular and a further steepest descent iteration was used. Thus here NR and GN were variations of the methods used on the other problems. The termination criterion was effectively T1 though applied to the steepest descent correction vector. GMP , of course, was unaffected by these singularity problems. The results of the trials are given in Table 3.5.

3.3.6. Transistor model

Eight methods, namely NR/N , NR/Bi, NR/M , GMP , GN/N , GN/Bi , GN/Br and GN/M , were tried on the transistor model problem which is substantially more difficult than any of the preceding problems. Apart from its extreme non-linearity, the component functions can produce very large values even when parameter values are restricted to lie in the

Starting point $\underline{x}^{(o)\tau} = (x_1, x_2)$	Method	Termination criterion satisfied	Approx. final point	Final objective function	No. of iterations	No. of function evaluations
(5,5)	NR/N	1	(0,0)	3.4×10^{-13}	71	71
(5,5)	NR/M	5	(0,0)	1.1×10^{-18}	80	983
(5,5)	GN/N	1	(2,4)	0.0	8	8
(5,5)	GN/M	1	(2,4)	6.2×10^{-11}	5	50
(500,500)	NR/N	1	(0,0)	3.5×10^{-13}	104	104
(500,500)	NR/M	5	(0,0)	9.8×10^{-19}	85	1051
(500,500)	GN/N	1	(2,4)	7.0×10^{-10}	23	23
(500,500)	GN/M	1	(0,0)	5.1×10^{-24}	31	345

Table 3.4 HDM problem

Method	Termination criterion satisfied	Final objective function	No. of iterations	No. of function evaluations
NR/Bi	1	1.4×10^{-25}	46	51
NR/M	1	3.6×10^{-32}	14	170
GMP	S	3.0×10^{-21}	64	300
GN/Bi	1	1.6×10^{-31}	32	37
GN/M	1	3.0×10^{-33}	12	148

Table 3.5 Miele's function

region of interest. Unless precautions are taken, an optimisation algorithm is liable to attempt to calculate numbers outside the range which can be held by the computer, thus giving floating point overflow errors which normally terminate the program. In the computing environment used, certain system software switches could be set to enable continuation when some, but not all, overflow errors occurred. To have prevented all overflow errors would have required additional programming which would have effectively modified the optimisation algorithms described in this chapter. It was decided neither to use the overflow switches nor to do any specific programming to prevent overflow errors in this set of trials. Thus for this problem only, an extra termination criterion is introduced, that of an overflow error occurring, denoted by '0' .

It was stated earlier that for this problem the parameter values should remain positive if an engineering design is to be realised. To maintain this restriction, a variable transformation was used so that the optimisation variables were the logarithms of the model parameters. The initial model parameters were defined as specified displacements (d) from the solution $\underline{x}_1^*$, so that

$$x_i^{(o)} = \max(x_{1i}^* + d,\ 0.1) \quad \text{for} \quad i = 1,2,\ldots,8$$

where $$\underline{x}_1^* = (x_{11}\ x_{12}\ \ldots\ x_{18})^T = (0.9,0.45,1.0,8.0,8.0,5.0,1.0,2.0)^T.$$

The results of this set of trials is shown in Table 3.6. Where convergence was obtained the number of iterations is given; failure is indicated by the letter 'F' followed by the code of the termination criterion satisfied.

Overflow was a problem for method NR/N even for small values of d and hence the number of trials for this method was restricted. Where convergence was obtained, termination criterion T1 or S was evoked except for GMP with d = - 0.1 . Here the NAG algorithm terminated

with a message to the effect that the process did not seem to be converging, even though the value of the objective function was low (1.2×10^{-17}) , the objective function was decreasing and the rate of decrease was increasing.

If the number of successful runs for each method is used to assess performance, the methods in increasing order of merit for this example are: NR/N , GMP , NR/Bi , NR/M , GN/N , GN/Bi , GN/Br , GN/M .

Starting displacement d	NR/N	NR/Bi	NR/M	GMP	GN/N	GN/Bi	GN/Br	GN/M
0.4	*	FO	F3	FO	F5	F5	F3	F5
0.3	*	FO	F3	FF	F5	F5	8	8
0.2	FO	F3	F3	FO	6	6	6	6
0.1	FO	21	18	FO	5	5	5	5
-0.1	FO	40	14	39+	4	4	4	5
-0.2	FO	F2	F2	45	5	5	5	5
-0.3	*	17	17	FO	5	5	5	6
-0.4	*	19	F2	FO	7	7	6	6
-0.5	*	FO	41	70	F5	6	7	7
-0.6	*	FO	26	FO	F5	7	F5	7
-0.7	*	FO	FO	FO	F5	F5	8	9
-0.8	*	FO	FO	FO	FO	FO	F3	F3

Table 3.6 Transistor model

* not attempted

\+ algorithm terminated although convergence was imminent

In all computer runs described so far, no correction limiting was applied. The transistor model problem was used to examine the effect of applying various limits δ_{max} in the GN/M method. The results are given in Table 3.7 using the same conventions as for the previous table. Based on these results, Table 3.8 shows, for each correction component limit value, the number of runs terminating successfully and the displacement range about zero throughout which convergence to the solution

d \ δ_{max}	0.1	0.2	0.3	0.4	0.5	0.6	1.0	10^{50}
1.8	F4	F4	F3	F3	F3	F3	F3	F3
1.7	F4	F3	F3	F3	F3	F3	F3	F3
1.6	F4	F4	F3	F3	F3	F3	F3	F3
1.5	F4	F4	F3	F3	F3	F3	F3	F3
1.4	F4	F3	F3	F3	F3	F3	F3	F3
1.3	F4	F4	F3	F3	F3	F3	F3	F3
1.2	F4	F4	F3	F3	F3	F3	F3	F3
1.1	F4	F3	F3	F3	F3	F3	F3	F3
1.0	F3	20	F3	F3	F3	F3	F3	F3
0.9	F4	F4	13	F3	10	F3	F3	F3
0.8	F3	F3	F3	F3	10	13	F3	F3
0.7	30	16	F3	F3	10	11	F3	F3
0.6	21	22	8	10	9	9	9	9
0.5	14	10	13	14	F3	F3	F3	F3
0.4	11	10	10	13	F3	F3	F3	F5
0.3	8	7	7	7	7	7	7	8
0.2	7	6	6	6	6	6	6	6
0.1	5	5	5	5	5	5	5	5
-0.1	6	5	5	5	5	5	5	5
-0.2	9	6	5	5	5	5	5	5
-0.3	13	9	7	7	6	6	6	6
-0.4	18	11	8	8	7	6	6	6
-0.5	18	11	8	8	7	6	7	7
-0.6	18	11	8	7	7	7	6	7
-0.7	18	11	9	8	7	7	7	9
-0.8	25	14	10	10	9	8	F3	F3
-0.9	30	19	14	11	10	11	14	F3
-1.0	29	18	15	11	10	10	F3	F3
-1.1	28	18	14	11	10	11	F3	F3
-1.2	28	17	13	11	10	10	F3	F5
-1.3	27	17	13	11	10	9	F3	F3
-1.4	27	17	13	11	10	11	F3	F3
-1.5	27	17	13	11	10	F3	9	F5
-1.6	27	17	14	13	10	F4	11	F5
-1.7	27	20	17	13	11	11	F4	10
-1.8	27	19	F4	20	18	F4	F3	F5
-1.9	33	18	34	F4	F3	F4	12	F5
-2.0	33	18	18	F4	15	F4	12	F5
-2.1	33	18	F3	F4	15	F4	10	F5
-2.2	33	18	F4	F4	F4	F4	13	F5
-2.3	33	18	F4	F4	F4	F4	10	F5
-2.4	33	18	16	17	F4	F4	12	9
-2.5	33	19	17	F4	16	F4	14	13
-2.6	F4	F4	F3	F4	F3	F4	15	F5
-2.7	F4	F3	F3	F3	F3	24	F3	F5
-2.8	F4	F3	F3	F3	F3	F3	F3	F5
-2.9	F4	F3	F3	F4	F3	F3	F3	F5
-3.0	F4	F3	F3	F3	F4	F3	10	F5

Table 3.7 Correction limiting on the transistor model problem

was observed. The results clearly indicate that in general the smaller the correction limit the greater is the probability of success. The same pattern was apparent in similar though much less extensive trials on NR and GMP . However, the penalty for choosing too small a limiting value

is that more iterations than are strictly necessary are used.

Although the general trend in the relationship between the correction limit and the rate of convergence is clear from Table 3.8, there were some results which did not conform and therefore cast doubts on the validity of some of the termination criteria. These trials were rerun with the suspect criteria removed; in no case was convergence to the solution subsequently achieved.

δ_{max}	Number of successful runs	d range of convergence
0.1	32	3.2
0.2	33	3.2
0.3	28	2.3
0.4	25	2.4
0.5	28	2.1
0.6	22	1.7
1.0	23	1.0
10^{50}	13	1.0

Table 3.8 Summary of effect of correction limiting

3.4. Discussion

For the type of problem considered, i.e. optimisation of a function which can be expressed as a sum of squares which is zero at the solution, the evidence that GN is better than NR is substantial. For similar correction scaling methods there was no case where NR reached a solution and GN failed, although many examples of the converse were found. When both methods were successful, GN almost always required fewer iterations, the only exception was to be found in the trials on Rosenbrock's function. Although no run timings are given, from the form of the two algorithms it is clear that this would generally imply that GN is quicker. Thus to describe GN as an approximation to NR is misleading.

On the other hand, the performance of GMP , one of the most highly regarded quasi-Newton methods, is consistent with it being an approximation to NR , although it has certain improved features which were described in Chapter II. These features are undoubtedly a benefit in a great number of cases, especially from the point of view of speed of computation, although it should be remarked that GMP generally required more iterations than NR/M. However, when the use of more than one optimisation algorithm is possible the first consideration should be which algorithms are likely to converge to a solution. The results of the transistor model problem suggest that a suitable form of NR is more likely to do so than the standard GMP method.

Thus the results show that the method using second derivatives (NR) did better than the method which uses first derivatives to approximate second derivatives (GMP) . The method which uses first derivatives and makes no attempt to approximate second derivatives (GN) achieved a better performance still. This does not necessarily mean that the use of second derivatives is a retrograde step; it is essential to bear in mind the type of problems that the algorithms were designed to solve. Whilst this exercise demonstrates that merely using an algorithm which employs higher derivatives is no guarantee that an improved performance will be obtained, and indeed it shows that a considerably worse performance can be encountered, an investigation into the use of second derivatives in the type of problem described requires GN to be compared with a second derivative algorithm for solving simultaneous equations.

On the method of correction scaling, the results appear to indicate that while the N and Bi methods are superior for relatively simple problems, some form of univariate minimisation is desirable on more difficult problems. Apart from the evidence of the HDS problem, where

examples of GN/M failing when GN/N succeeded were noted, a minimisation method appears to be the safer policy to adopt. And yet accepting the first point which reduces the objective function is highly favoured by many algorithms designers; one possible explanation is that their trial problems are not sufficiently complex.

Assuming that univariate minimisation is to be adopted the question to be answered is to what accuracy should the search be taken. On the few runs of bracketing only that were tried, the performance of the univariate minimisation was not matched. This admittedly flimsy evidence suggests that a higher accuracy should be sought.

The advantage and drawback of limiting corrections has been demonstrated, the problem being to choose the highest value of correction limit which would produce convergence. However, no method of selecting this optimum value for a given problem other than by user experience through trial and error has been suggested. This difficulty would not exist in a method which was inherently more stable and thus able to limit correction values automatically.

CHAPTER IV

DEVELOPMENT OF SECOND DERIVATIVE SUM OF SQUARES METHODS

Although there exist several algorithms for minimising a general unconstrained function which make use of second derivatives, to the author's knowledge there has been no published work on the development of a second derivative method designed specifically for the minimisation of a sum of squares function or for the solution of a set of simultaneous equations. It has been shown in Chapter III that on certain problems the sum of squares method performs distinctly better than the general function method. There are also indications from work done by others that second derivatives are of benefit in general function minimisation. The logical question was therefore whether a second derivative sum of squares method could be developed which would improve further on the performance given by existing methods applied to the aforementioned problems. Could the convergence range of a second derivative method eliminate the need for a two-part algorithm, or would the evaluation of second derivatives prove to be an unproductive complication?

The development of algorithms in order to answer these questions formed the major part of this research. It involved a continuous process of modification based on trials conducted mainly on the transistor model problem which was chosen since it represented an engineering problem where there was clearly scope for improvement in the type of optimisation algorithm under consideration. Later, a second engineering problem was used for further trials.

Section 2.4.1 gave the derivation of the GN method from consideration of the first two terms of the Taylor Series expansion. The methods to be described here are based on inclusion of the second derivative term, i.e.

$$s_i(\underline{x} + \underline{\delta}) \doteq s_i(\underline{x}) + \sum_{j=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j} \delta_j + \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^2 s_i(\underline{x})}{\partial x_j \partial x_k} \delta_j \delta_k (i=1,2,\ldots,m).$$

As with GN , the right-hand side is then equated to zero:

$$0 = s_i(\underline{x}) + \sum_{j=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j} \delta_j + \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^2 s_i(\underline{x})}{\partial x_j \partial x_k} \delta_j \delta_k \quad (i=1,2,\ldots,m). \quad (4.1)$$

Whereas a solution or least-squares solution to the equivalent set of simultaneous equations for GN is obtainable by matrix inversion, assuming the matrix is non-singular, there is no analytical method for solving Equations (4.1) in the general case. It was therefore necessary to devise an iterative method for this purpose.

This approach was justified by the reasoning that Equations (4.1) should be easier to solve than the exact simultaneous equations for functions with non-trivial third or higher derivatives, provided only that a solution existed. Clearly for simultaneous equations with lower order derivatives, such as those given by Rosenbrock's function and the modified form of Rosenbrock's function, Equations (4.1) are an exact representation. Therefore, defining an iteration to be the process to get from $\underline{x}^{(p)}$ to $\underline{x}^{(p+1)} = \underline{x}^{(p)} + \underline{y}^{(p)}$, such problems would yield a solution in one iteration from any starting point with $\underline{y}^{(o)}$ defined as $\underline{\delta}$.

4.1. Initial attempts

The first attempts at solving Equations (4.1) can be summarised as follows:

(i) from the current point $\underline{x}^{(p)}$, use the (unscaled) GN correction vector to give an estimate $\underline{\delta}^{[o]}$ of $\underline{\delta}$;

(ii) substitute the current estimate $\underline{\delta}^{[q]}$ of $\underline{\delta}$ in a modified version of Equations (4.1) to obtain an improved estimate $\underline{\delta}^{[q+1]}$;

(iii) do a univariate minimisation on the objective function $\phi(\lambda) = f(\underline{x}^{(p)} + \lambda\underline{\delta}^{[q+1]})$ and define $\underline{x}^{(p+1)}$ to be the point giving the minimum.

Two modified versions of Equations (4.1) were used to update the estimate of $\underline{\delta}$, both of which gave simultaneous linear equations in the unknown vector $\underline{\delta}^{[q+1]}$:

$$(a)\quad 0 = s_i(\underline{x}) + \sum_{j=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j} \delta_j^{[q+1]} + \psi_i^{[q]} \qquad (i=1,2,\ldots,m)$$

$$\text{where}\quad \psi_i^{[q]} = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j \partial x_k} \delta_j^{[q]} \delta_k^{[q]} \;;$$

$$(b)\quad 0 = s_i(\underline{x}) + \sum_{j=1}^{n} \left\{ \frac{\partial s_i(\underline{x})}{\partial x_j} + \frac{1}{2}\sum_{k=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j \partial x_k} \delta_k^{[q]} \right\} \delta_j^{[q+1]}$$

$$(i=1,2,\ldots,m).$$

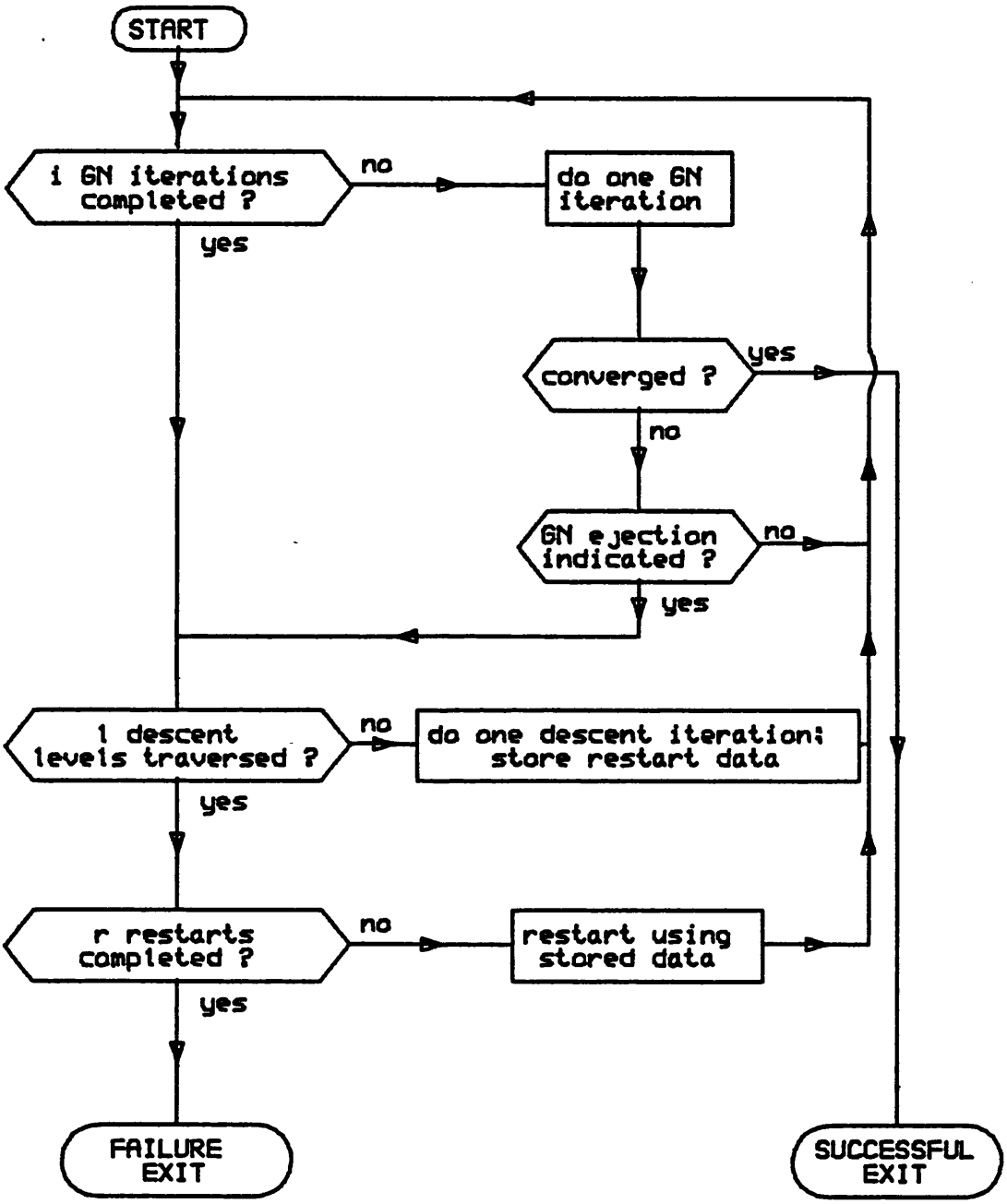
Both versions were tried on Rosenbrock's function and both converged in two iterations. This was an encouraging performance, bearing in mind that Equations (4.1) were only approximated, which compares favourably with all versions of GN (Table 3.1). On the transistor model problem, however, success was not forthcoming. An initial displacement of $d = 0.5$ from the positive solution was used which in hindsight seems to be too large a step to have taken since GN/M only converges from this point with fairly severe correction limiting imposed. Nevertheless, with the introduction of further univariate searches in the scheme described and with correction limiting imposed convergence to the solution was obtained, though not consistently.

Using an approximation to Equations (4.1) at an early stage was questionable since the results of Chapter III, some of which were produced subsequently, have shown that using convergence as a criterion the introduction of approximations is liable to make the algorithm perform less well. Thus it appeared to be more logical to determine what could

be achieved with exact corrections before attempting to approximate, however beneficial such approximations may be by reducing computing time. The limited success in the transistor model problem led to an investigation of the use of exact corrections.

The first "exact" method adopted was to apply GN to Equations (4.1) to solve for $\underline{\delta}$ to a reasonably high accuracy and then to minimise $\phi(\lambda) = f(\underline{x} + \lambda\underline{\delta})$ to give a new point. However, when this process was used on the transistor model problem GN experienced difficulty in obtaining $\underline{\delta}$. It was not known whether this was because there were no real solutions or whether GN was an inadequate method to use on the sub-problem of finding $\underline{\delta}$. At this stage of preliminary investigation it was decided to make use of a more powerful optimisation method, namely Cutteridge's two-part algorithm consisting of the gradient-descent and GN methods. It was not envisaged that this algorithm would form a part of any second derivative sum of squares algorithm that might be developed, but that it would provide a powerful research tool to aid such development. The flexibility that was enabled by allowing the user to specify algorithm parameters if their default values were unsuitable was particularly attractive. For example, one parameter specifies the maximum number of descent "levels" to be traversed. By setting this parameter to zero, the method reduces to a GN/M type algorithm. At the other extreme, a number of "restarts" from local minima found during the univariate search, described in Section 2.7.3, can be prescribed in the event that the method is unable to converge otherwise.

Figure 4.1 gives a simplified flowchart of the method; some of the program path possibilities and much of the detail have been omitted for clarity. However, in general the program enters the GN phase at the beginning of the run and re-enters after each descent iteration or after restarting from a previously stored local minimum. One descent



i = max. no. of GN iterations
(at each GN attempt)
l = max. no. of descent levels
r = max. no. of restarts

Figure 4.1 Cutteridge's two-part optimisation method

level is traversed on each descent iteration, restart data from the higher levels being used first. If GN is unsuccessful, the next descent iteration proceeds from the point given by the last descent iteration so the GN path is discarded. The GN termination condition on the number of iterations refers to the number since the last GN starting point and not the total number executed during the entire process.

4.2. Basic theory of methods

It was clear from the early work that Equations (4.1) were unsatisfactory in that there was not necessarily a solution vector of real components. This problem was solved by the introduction of a scalar in a similar way to the scalar introduction in GN . Although the scalar is applied directly to the correction vector, if it is included in the simultaneous equations from which GN is derived, we have:

$$0 = \lambda s_i(\underline{x}) + \sum_{j=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j} \delta_j \qquad (i=1,2,\ldots,m) \qquad (4.2)$$

or $$(1-\lambda)s_i(\underline{x}) = s_i(x) + \sum_{j=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j} \delta_j \qquad (i=1,2,\ldots,m)$$

Thus Equations (4.2) are derived from the Taylor series expansion with the left-hand side replaced by $(1-\lambda)s_i(\underline{x})$ instead of zero. Clearly λ would be expected to lie in the range $0 < \lambda \leq 1$ so that a reduction in the values of s_i is more realistically anticipated as opposed to the rather grand expectation of solving the problem in one iteration. The error in the approximation introduced by truncation of the Taylor series increases as λ increases from zero and the approximation is clearly invalid when $\frac{\partial f}{\partial \lambda} = 0$, there being no justification for choosing higher values of λ .

Applying similar theory to Equations (4.1), we obtain:

$$0 = \lambda s_i(\underline{x}) + \sum_{j=i}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j}\delta_j + \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j \partial x_k}\delta_j\delta_k \quad (i=1,2,\ldots,m) \quad (4.3)$$

Higher order terms could be included similarly. Once again it is reasonable to choose a value of λ greater than zero but not greater than the first positive value to yield $\frac{\partial f}{\partial \lambda} = 0$.

The basic idea of the new method was to use Equations (4.3) to define $\underline{\delta}$ for various values of λ , choosing a suitable correction vector by examination of the objective function $f(\underline{x} + \underline{\delta}(\lambda))$. To substantiate this idea, an existence theorem for implicit functions is quoted.

Let the set of simultaneous equations (A) be defined by:

$$\omega_i(x_1,x_2,\ldots,x_k,x_{k+1},\ldots,x_{k+r}) = 0 \qquad (i=1,2,\ldots,k)$$

and let (A) be satisfied by $\underline{x} = \underline{x}^*$.

Suppose that:

(i) the k functions ω_i are continuous in the neighbourhood of $\underline{x}^*$;

(ii) the functions ω_i possess continuous first partial derivatives in the neighbourhood of $\underline{x}^*$;

(iii) the (k x k) Jacobian formed by the partial derivatives of ω_i with respect to $x_1, x_2,\ldots,x_k$ is non-singular at $\underline{x}^*$;

then there exists a unique set of continuous functions:

$$x_i = \Psi(x_{k+1},x_{k+2},\ldots,x_{k+r}) \qquad (i=1,2,\ldots,k)$$

which satisfy (A) and reduce to x_i^* $(i=1,2,\ldots,k)$ when $x_{k+j} = x_{k+j}^*$ $(j=1,2,\ldots,r)$. An outline proof of this theorem by induction can be found in Khinchin (66).

To apply the theorem to Equations (4.3) it is necessary to enforce $m = n$. The right-hand sides of the equations are used to define the functions $\omega_i(\delta_1,\delta_2,\ldots,\delta_n,\lambda)$ $(i=1,2,\ldots,n)$ which are zero for $\lambda = 0$ and $\underline{\delta} = \underline{0}$. At this point $\frac{\partial \omega_i}{\partial \delta_j} = \frac{\partial s_i}{\partial x_j}$ and $\frac{\partial \omega_i}{\partial \lambda} = s_i$. Thus the conditions of the existence theorem are satisfied provided the functions s_i $(i=1,2,\ldots,n)$ and their first and second partial derivatives with respect to x_j $(j=1,2,\ldots,n)$ are continuous in the neighbourhood of the current point $\underline{x}$ and the $(n \times n)$ Jacobian given by $J_{ij} = \frac{\partial s_i}{\partial x_j}$ is non-singular at this point. Given these conditions, there exists a value Λ such that Equations (4.3) can be solved for $0 \leqslant \lambda < \Lambda$. Furthermore, a reduction in the objective function can be obtained by a value of λ in this range, unless the solution has been reached. The proof is similar to the equivalent GN proof and is given in Appendix I.

To extend the idea to the case $m > n$ it is necessary to replace Equations (4.3) by their generalised least-square forms which are derived in the same way as the generalised GN method (Section 2.4.1.). The resulting equations are:

$$\sum_{i=1}^{m}\left\{(\lambda s_i + \sum_{j=1}^{n}\frac{\partial s_i}{\partial x_j}\delta_j + \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial^2 s_i}{\partial x_j \partial x_k}\delta_j\delta_k)(\frac{\partial s_i}{\partial x_r} + \sum_{j=1}^{n}\frac{\partial^2 s_i}{\partial x_j \partial x_r}\delta_j)\right\} = 0$$

$$(r=1,2,\ldots,n). \qquad (4.4)$$

Defining the left-hand sides of Equations (4.4) to be $\omega_r(\delta_1,\delta_2,\ldots,\delta_n,\lambda)$ for $r=1,2,\ldots,n$ we note that at $\lambda = 0$ and $\underline{\delta} = \underline{0}$:

$$\frac{\partial \omega_r}{\partial \lambda} = \sum_{i=1}^{m} s_i \frac{\partial s_i}{\partial x_r} \quad \text{and} \quad \frac{\partial \omega_r}{\partial x_j} = \sum_{i=1}^{m}\frac{\partial s_i}{\partial x_j}\frac{\partial s_i}{\partial x_r}$$

for $r=1,2,\ldots,n$ and $j=1,2,\ldots,n$.

Thus the implicit function existence theorem requires continuity of s_i $(i=1,2,\ldots,m)$, their first and second derivatives and non-singularity

of J^TJ at the current point for a unique solution giving $\underline{\delta} = \underline{0}$ at $\lambda = 0$. Once again, only positive values of λ need be considered in order to reduce the objective function (Appendix I).

The generalised equations were used in the numerical trials of Chapter V by applying a least-squares technique to Equations (4.3), which is analytically equivalent to solving Equations (4.4).

4.3. Implementation of computer algorithms

The solution of Equations (4.3) or their variants formed the heart of the computer algorithms, and is much more difficult than the solution of the GN equations because an analytical method is unavailable and $\underline{\delta}$ is not generally a linear function of λ , although such a form was investigated later.

It was required to solve Equations (4.3) for various values of λ as determined by the univariate search procedure seeking to reduce $\phi(\lambda) = f(\underline{x} + \underline{\delta}(\lambda))$. At this stage $\underline{s}$ and its first and second derivatives with respect to $\underline{x}$ are constant. This sub-problem of solving for the unknown $\underline{\delta}$ was tackled by applying an optimisation method, the objective function for which was:

$$\theta(\underline{\delta}) = \sum_{i=1}^{m} \{\sigma_i(\underline{\delta})\}^2 \qquad (4.5)$$

where $$\sigma_i(\underline{\delta}) = \lambda s_i(\underline{x}) + \sum_{j=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j} \delta_j + \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j \partial x_k} \delta_j \delta_k \quad (i=1,2,\ldots,m).$$

Although in the early stages the full power of the Cutteridge method was brought to bear on these equations, the desirability of a more simple method of solution was a prime consideration. During development of the second derivative methods, the Cutteridge method was gradually reduced in power until GN alone was used to attempt to solve the equations. This was a significant step because what was in effect a second derivative GN

method required nothing more complex than the repeated application of GN .

The results of Chapter III demonstrated a widely-held view that GN exhibits good convergence properties provided its initial point is sufficiently close to the solution. In the method of solution of Equations (4.3), by judiciously varying λ the problem was tailored to fit the starting point. By making λ small enough the solution to Equations (4.3) can always be found from an initial estimate of $\underline{0}$, or better still the GN correction vector at $\underline{x}$ multiplied by λ . Based on the $\underline{\delta}(\lambda)$ so obtained, suitable extrapolation was applied to attempt to solve for a higher value of λ and the process repeated as necessary. The aim of this univariate search was to bracket the minimum of $f(\underline{x} + \underline{\delta}(\lambda))$, whence interpolation could be applied to find the minimum more precisely, using the same termination criterion as method (iv) of Section 3.2.1. The extrapolation had to balance two conflicting factors: too many evaluations of $\underline{\delta}$ for various λ is clearly inefficient, too few give inaccurate estimates which can have the same effect. However, inaccurate estimates can have a more disastrous effect. Equations (4.3) can clearly have more than one solution. Figure 4.2 shows plots of log. $f(\underline{x} + \underline{\delta}(\lambda))$ against λ for eight distinct $\underline{\delta}$-paths which are solutions of Equations (4.3) for the transistor model problem with the components of $\underline{x}$ set to 1.0 above the components of the solution $\underline{x}_1^*$. The only solution of interest to this algorithm is the one passing through $\underline{\delta} = \underline{0}$. An inaccurate estimate could cause a "solution jump" during the extrapolation process with a resultant breakdown of the univariate minimisation procedure.

A further difficulty encountered by this method is that it is not always possible to bracket the minimum. Figures 4.3 to 4.5, also taken from the transistor model problem, show the three possibilities. In Figure 4.3 the paths of the eight solution components are reasonably well-

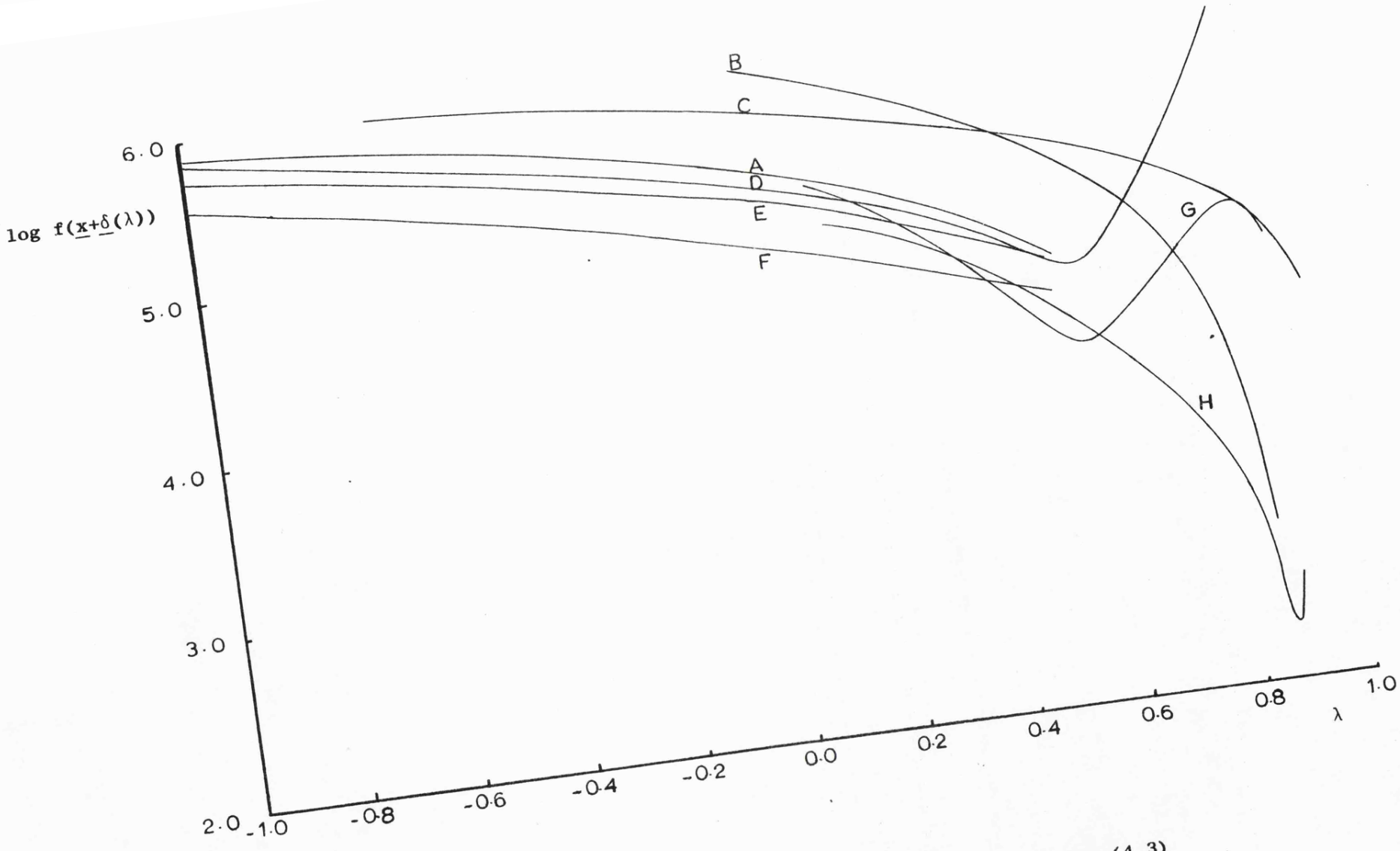


Figure 4.2 The objective function for various solutions to Equations (4.3)

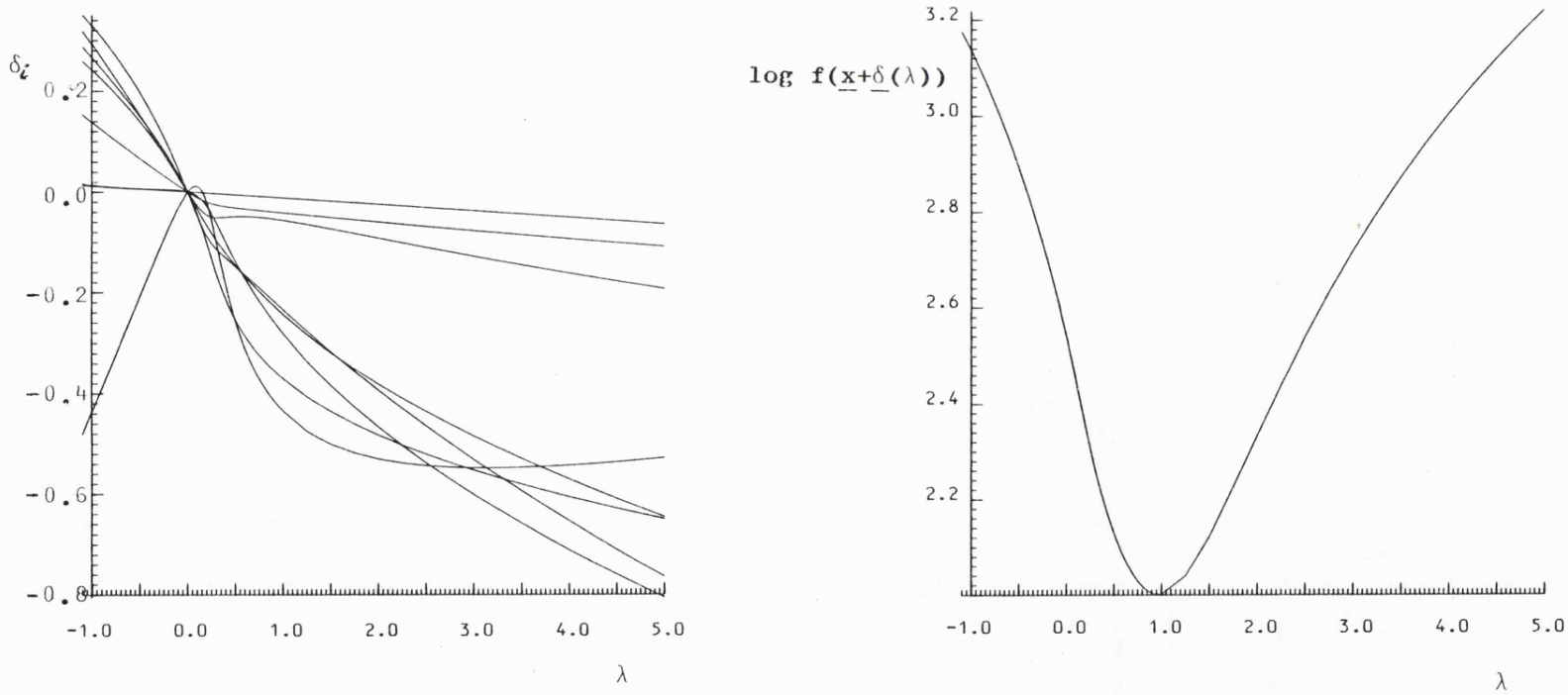


Figure 4.3 Components of solution of Equations (4.3) producing unimodal objective function

behaved and the univariate minimum can be bracketed, though the question of to what accuracy is relevant. In Figure 4.4 the solution components are again well-behaved though the resulting objective function is non-unimodal. For reasons already stated the aim would be to locate the minimum of lowest positive λ . In Figure 4.5, although the objective function is steadily reducing as λ increases, some of the solution components appear to become complex at about $\lambda = 0.52$. The univariate search would have to be terminated at or before such a point without spending excessive time looking for non-existent solutions. It was also necessary to consider how close to the limiting point the search should be taken.

The essential steps of the developed second derivative methods applied to Equations (4.3) are shown in flowchart form in Figure 4.6. A singular Jacobian necessitates termination of the method though in practice continuation is possible after the application of another method to move away from the singular point. Termination conditions were based on those described in Chapter III using the GN corrections evaluated at the beginning of each iteration. Conditions T1, T5 and T6 were implemented as described, whereas in many runs conditions T3 and T4 were merely flagged and the procedure allowed to continue in order to check their validity. In only one run was convergence obtained after one of these conditions (T3) was satisfied. Termination of the algorithm after an unsuccessful solution attempt for $\lambda = \varepsilon_2$, a small value, corresponds to condition T2 . If the solution attempt for small λ was successful, $\underline{\delta}(\lambda^{[o]})$ is sought where $\lambda^{[o]}$ is a higher value based on information from the previous iteration. Assuming success at this stage, bracketing is attempted followed by univariate minimisation if bracketing is accomplished. In all cases the method of attempting to solve Equations (4.3) was by use of GN , initial estimates of $\underline{\delta}$ being based on information accumulated during the iteration. It was quite possible for GN to fail to find

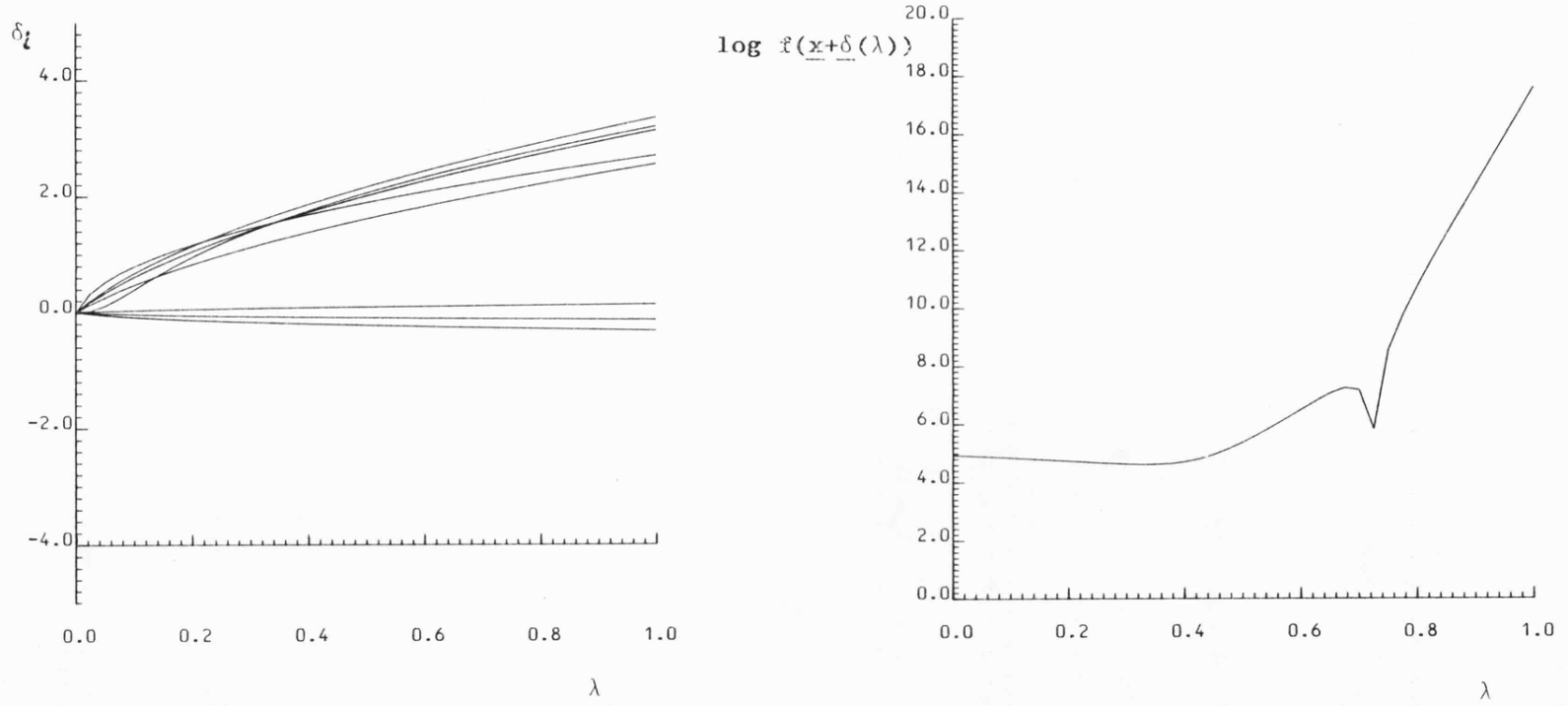


Figure 4.4 Components of solution of Equations (4.3) producing non-unimodal objective function

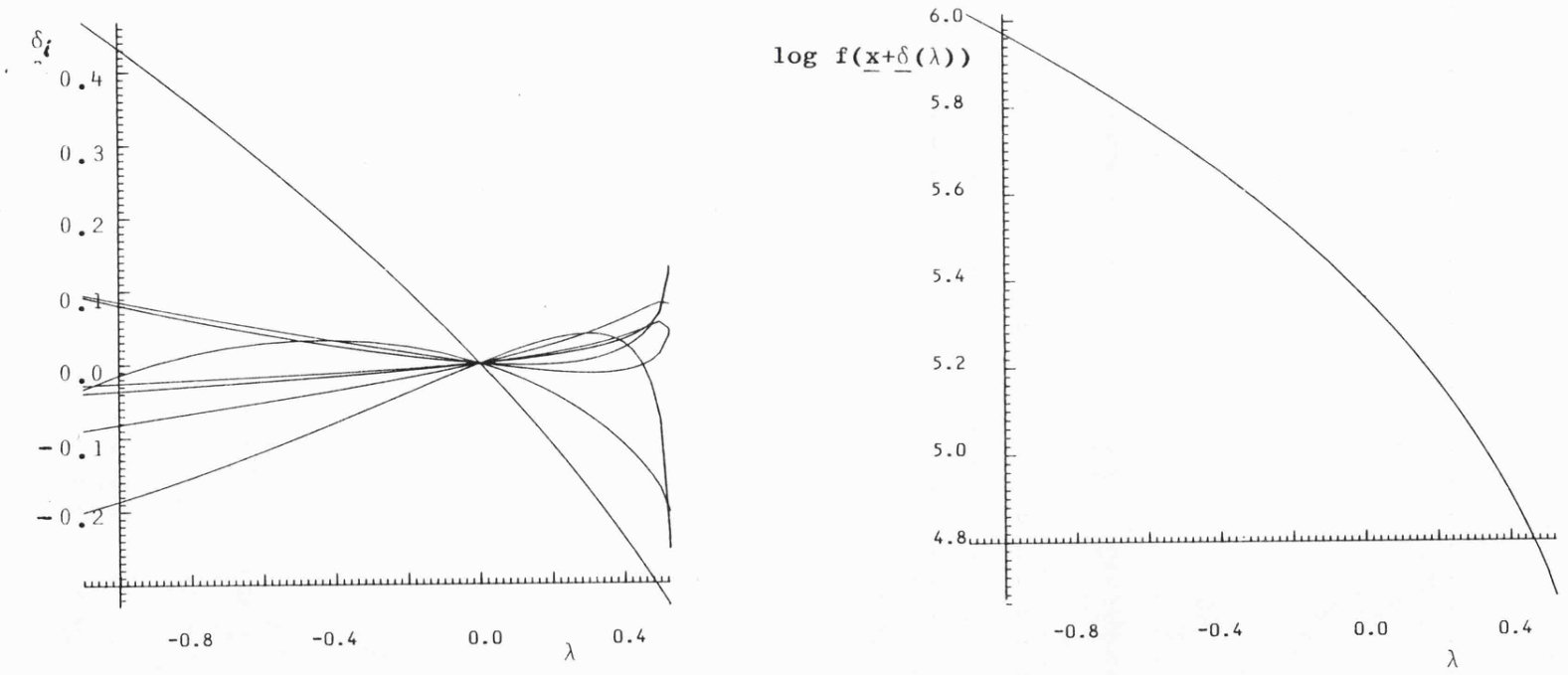


Figure 4.5 Components of solution of Equations (4.3) becoming complex

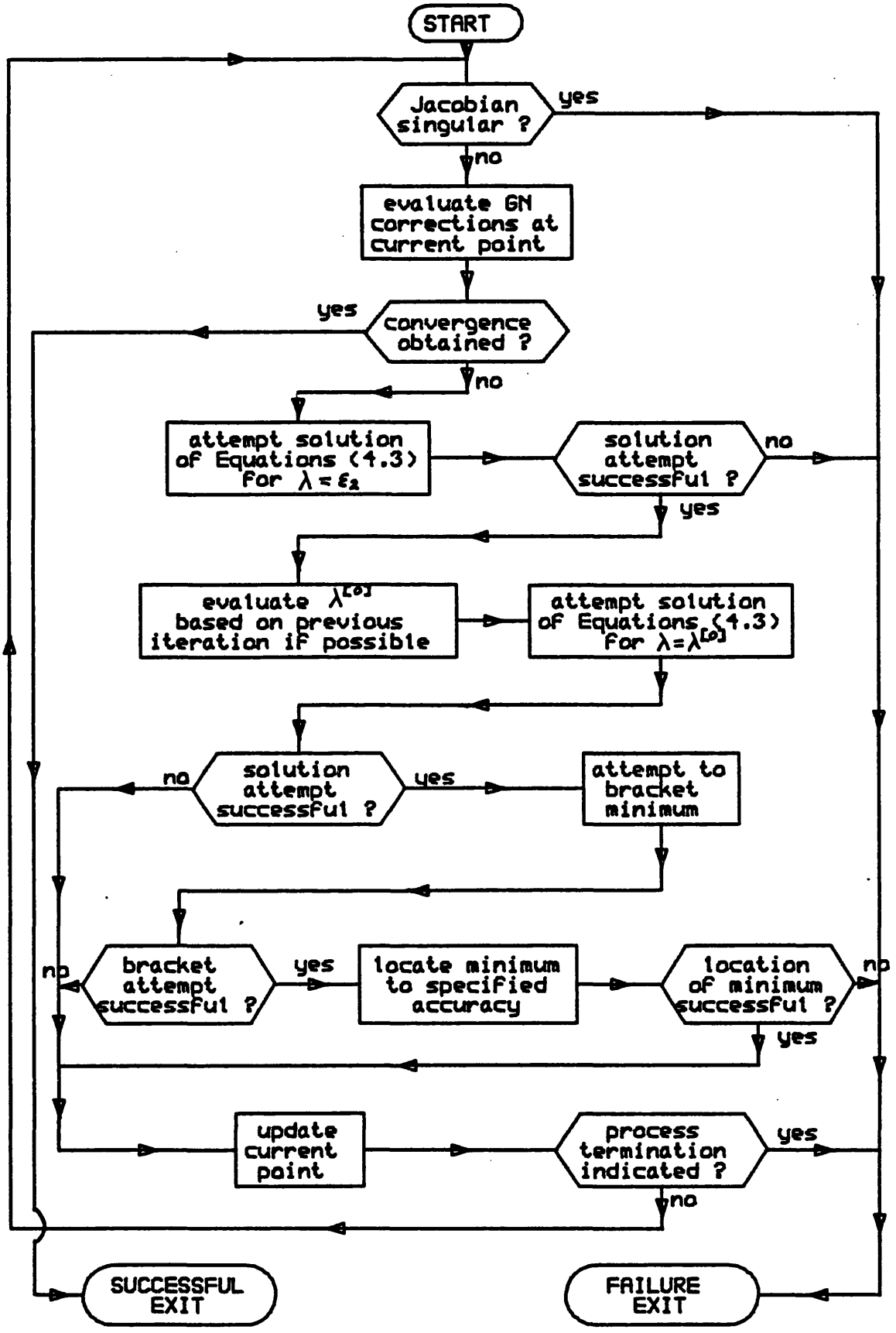


Figure 4.6 Flowchart of second derivative method

$\underline{\delta}(\lambda)$ for a particular value of λ, in which case a smaller value of λ, for which the estimates of $\underline{\delta}$ would be more accurate, would be tried. If this failed, a still smaller value would be tried. The process of repeating solution attempts could be terminated as soon as Equations (4.3) had been solved for any value of λ, not necessarily the one originally specified. This was useful in the bracketing and $\lambda^{[o]}$ stages, when the value of the objective function gave useful information provided it was for a value of λ greater than those for which Equations (4.3) had already been solved. This was not the case at the minimum location stage. Here information had been gathered from points around the minimum and therefore $\underline{\delta}$ should have been obtained relatively easily. If this was not so then a solution jump was indicated and the algorithm was terminated so that remedial action could be taken. In practice the algorithm is able to continue from here.

4.3.1. Variations on the basic theme

The first implementation of the scheme just described met with some success when tried on the transistor model problem. A number of starting points were found from which the second derivative algorithm converged whereas GN failed. However, there were also several points from which GN converged and the new algorithm failed. Clearly further attention to the details of the method were required. Also, research at this stage was hampered by the amount of time taken by the many GN iterations used in the attempts to solve for $\underline{\delta}$; it was therefore necessary to pay attention to computing efficiency before the potential of the algorithm had been assessed.

Nevertheless the algorithm showed promise and two variations were investigated. The first was the incorporation of a linear search

immediately before updating the current point (see Figure 4.6). Suppose that for the pth iteration the outcome of the earlier minimisation, or the bracketing attempt if it was unsuccessful, was a vector $\underline{\delta}^{(p)}$. The basic method, subsequently referred to as SDA, updated the problem unknowns by the relationship:

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} + \underline{\delta}^{(p)} .$$

The modified method, which will be called SDB , used the relationship:

$$\underline{x}^{(p+1)} = \underline{x}^{(p)} + \mu^{(p)}\underline{\delta}^{(p)}$$

where $\mu^{(p)}$ is the value of μ which minimises $\phi(\mu) = f(\underline{x}^{(p)} + \mu\,\underline{\delta}^{(p)})$. The second variation on SDA was to conduct a similar linear search each time any $\underline{\delta}(\lambda)$ was found. This method will be called SDC. SDC therefore contained a two-dimensional search, whereas SDB may be regarded as approximating such a search.

4.3.2. Correction limiting

The benefits of correction limiting, which were demonstrated in Chapter III, demanded that consideration be given to the application of this technique on the second derivative methods. In the λ-search, for each $\underline{\delta}$ found, correction limiting was imposed by setting to the limit those components of $\underline{\delta}$ exceeding the limit, with bracketing and minimisation attempted on the objective function values given by the modified $\underline{\delta}$s . This corresponds to the method used on GN and NR , but since the search was no longer linear, it was not possible to determine the exact point at which component limiting would occur, although this could be estimated using GN corrections. Previous bracketing algorithms developed at Leicester made use of the component limiting points for each component, but in the second derivative method the estimate of only the first limiting point was used (to define $\lambda^{[o]}$) .

The second search would then be applied to the final limited $\underline{\delta}$.

4.3.3. Control of GN entry

If an attempt to solve for $\underline{\delta}$ at a particular value of λ was unsuccessful, a new attempt at a lower value of λ would normally be initiated. The value selected would be based on the value of λ for which Equations (4.3) had already been solved.

Suppose that λ_a is the lowest unsuccessful value of λ

λ_b is the highest successful value of λ

λ_c is the value of λ for the new attempt

and λ_d is the lowest value of λ for which the objective function is of use.

λ_c was defined by:

$$\lambda_c = \lambda_b + \Lambda$$

where $$\Lambda = 0.1\ (\lambda_a - \lambda_b)$$

unless this caused λ_c to cross from above λ_d to below λ_d , in which case λ_c would be set to λ_d .

The process was repeated in the event of failure until the following limiting condition was satisfied:

$$\lambda_a - \lambda_b < \varepsilon_4\ \lambda_b \quad (\lambda_b > 0)$$

$$\lambda_a < \varepsilon_5 \quad (\lambda_b = 0)$$

where ε_4 and ε_5 are specified constants. The latter condition was an algorithm termination criterion comparable with T2 (Section 3.2.2).

If a solution attempt was successful for a value greater than or equal to λ_d , the process would terminate. If the value was less than λ_d a new attempt would be initiated for $\lambda_c = \lambda_b + \Lambda$ and Λ would be

doubled in preparation for the next attempt. Once again λ_c would be reset if λ_d was traversed.

This entry into the GN algorithm was controlled by an Algol real procedure named PENFUN , the flowchart for which is shown in Figures 4.7(a) and 4.7(b). The following Algol variables are used:

DR - the λ increment Λ ;

IFLAG - to indicate whether λ_c is greater than, equal to or less than λ_d ;

OK - boolean variable to indicate on exit whether the objective function for RREQD was found ;

R - the current λ value, λ_c ;

RFOUND - on exit, the highest value of λ for which the objective function was found if not for RREQD ;

RMIN - the minimum value of λ for which the objective function is of use, λ_d ;

RREQD - the originally specified value of λ for which the objective function was required ;

RSUCC - the highest successful value of λ , λ_b .

Upon exit, the appropriate objective function value was returned in PENFUN . The two successful exits, in the terminology of Figure 4.6, are EXIT A where the equations were solved for R = RREQD , and EXIT B , where the equations were solved for $R \geq$ RMIN . EXIT C occurred when the limiting condition was satisfied.

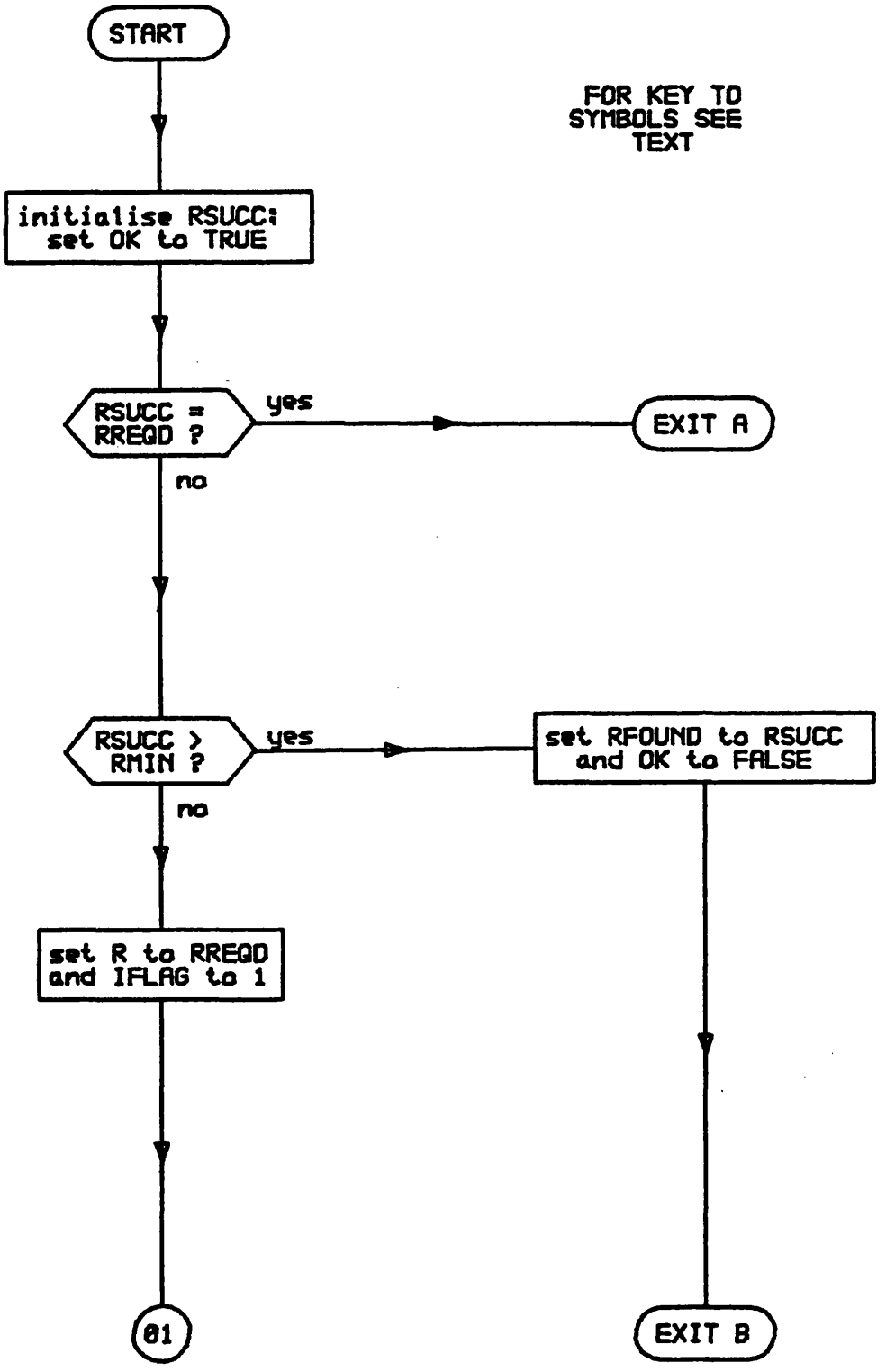


Figure 4.7(a) Control of GN entry by procedure PENFUN (part one)

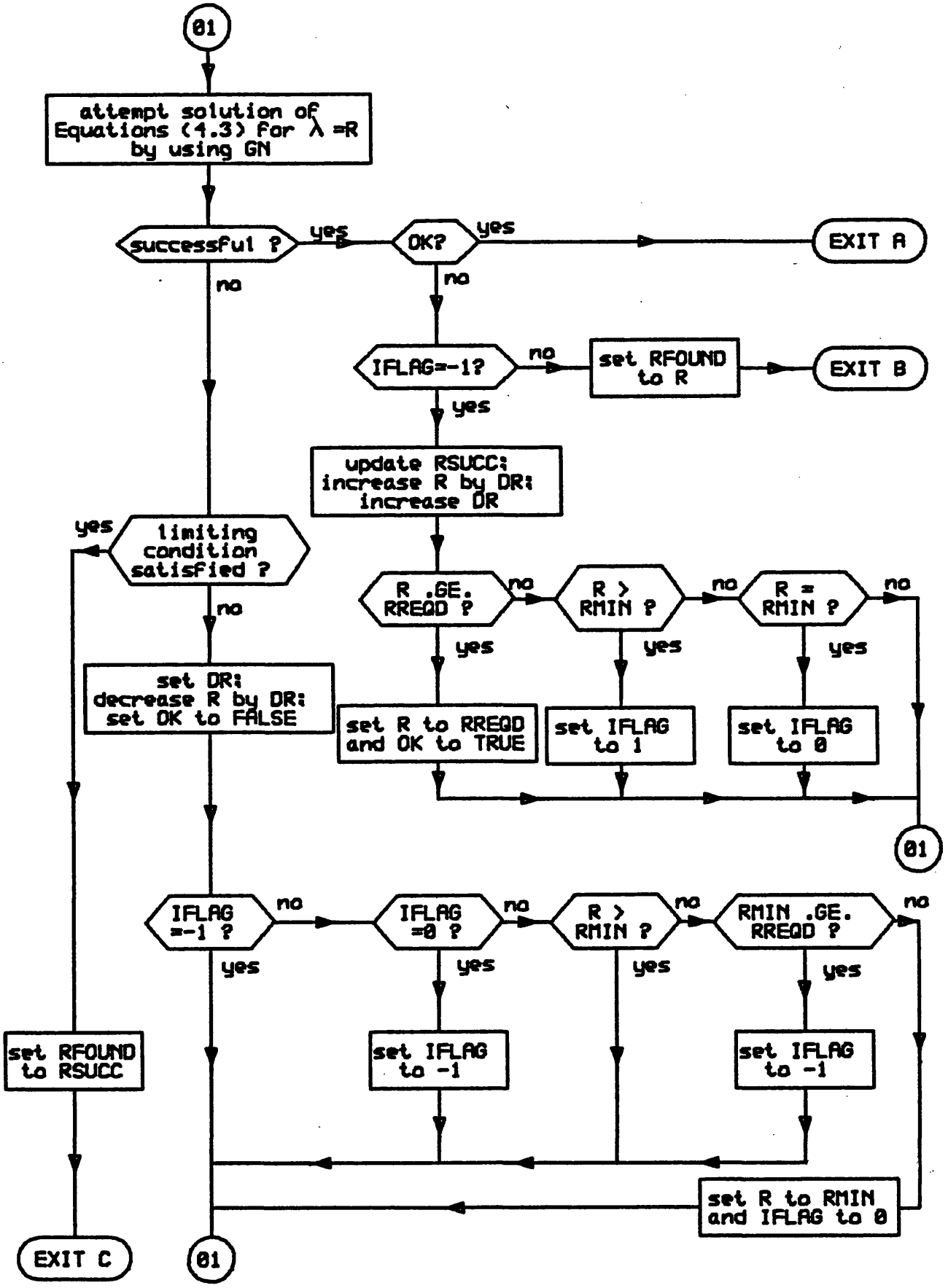


Figure 4.7(b) Control of GN entry by procedure PENFUN (part two)

4.3.4. Univariate search method

PENFUN was called whenever it was desired to solve for $\underline{\delta}$ and so existing procedures for bracketing and locating a univariate minimum had to be modified since they assumed that the objective function value would always be calculated whatever value of the variable was presented. The bracketing process used by Cutteridge et al. was for the range $\lambda \geq 0$, requiring the value of the objective function at $\lambda = 0$, i.e. $\phi(0)$. The objective function is evaluated at another value of λ, $\lambda^{[o]}$ and if this is not less than $\phi(0)$ further values of λ are generated by the formula:

$$\lambda^{[q+1]} = 0.1\ \lambda^{[q]} \qquad q = 0,1,...$$

until one which gives an objective function less than $\phi(0)$ is found or until λ reaches a prescribed small value and the bracketing attempt fails (cf. NR and GN termination condition T2). Assuming this search does not fail, a value $\lambda^{[j]}$ has been found such that $\phi(\lambda^{[j]}) < \phi(0)$. A second search using increasing values of λ is then initiated until an increase in the objective function is found. A pair of values enclosing at least one minimum can thus be determined. The values of λ used in this part were generated by the Fibonacci sequence:

$$\lambda^{[j+1]} = 2\lambda^{[j]}$$

$$\lambda^{[j+q+1]} = \lambda^{[j+q]} + \lambda^{[j+q-1]} \qquad q = 1,2,... \qquad (4.6)$$

The Fibonacci sequence in this form was unsuitable for the second derivative method because in the event of PENFUN failing for a particular value of λ, the next value generated would be much too large. For example, with a starting value of 0.2 the normal sequence is:

0.2, 0.4, 0.6, 1.0, 1.6, ...

but in the event of objective function evaluation failure at $\lambda = 1.0$, $\lambda = 0.64$ would be attempted by PENFUN . If this were successful, clearly the next value generated by the series, whether it be interpreted as 1.6 or 1.24 , is too high. Instead, the Fibonacci sequence was based on the distance from a point α , so that Equations (4.6) become:

$$\lambda^{[j+1]} = 2\lambda^{[j]} - \alpha$$

$$\lambda^{[j+q+1]} = \lambda^{[j+q]} + \lambda^{[j+q+1]} - \alpha \qquad q = 1,2,... \qquad (4.7)$$

where α is a constant.

α was set to zero initially. If the evaluation of $\phi(\lambda^{[j+q+1]})$ was unsuccessful but PENFUN returned with a value of λ greater than $\lambda^{[j+q]}$, the sequence was restarted with α set to $\lambda^{[j+q]}$ using the value found as the first point. In the above example this would yield:

0.2, 0.4, 0.6, 0.64, 0.68, 0.72, 0.8, 0.92, 1.12, ...

However it was found beneficial to ensure that $\lambda = 1.0$ was considered, so the final bracketing algorithm would have produced:

0.2, 0.4, 0.6, 0.64, 0.68, 0.72, 0.92, 1.0, 1.32, 1.72, ...

A simplified flowchart of the bracketing procedure is shown in Figure 4.8. It was programmed so that upper and lower bounds could be prescribed. The notation used is as follows. It is desired to find three values, X1, X2 and X3 , of the search parameter which give objective function values F1 , F2 and F3 respectively such that

$$X1 < X2 < X3 , \quad F2 < F1 \quad \text{and} \quad F2 < F3 .$$

On input, X1 is set to the lower bound; X2 lies within the specified range, F1 and F2 are given. XU is the upper bound and EXL is the effective lower bound of X2 , calculated from prescribed accuracy

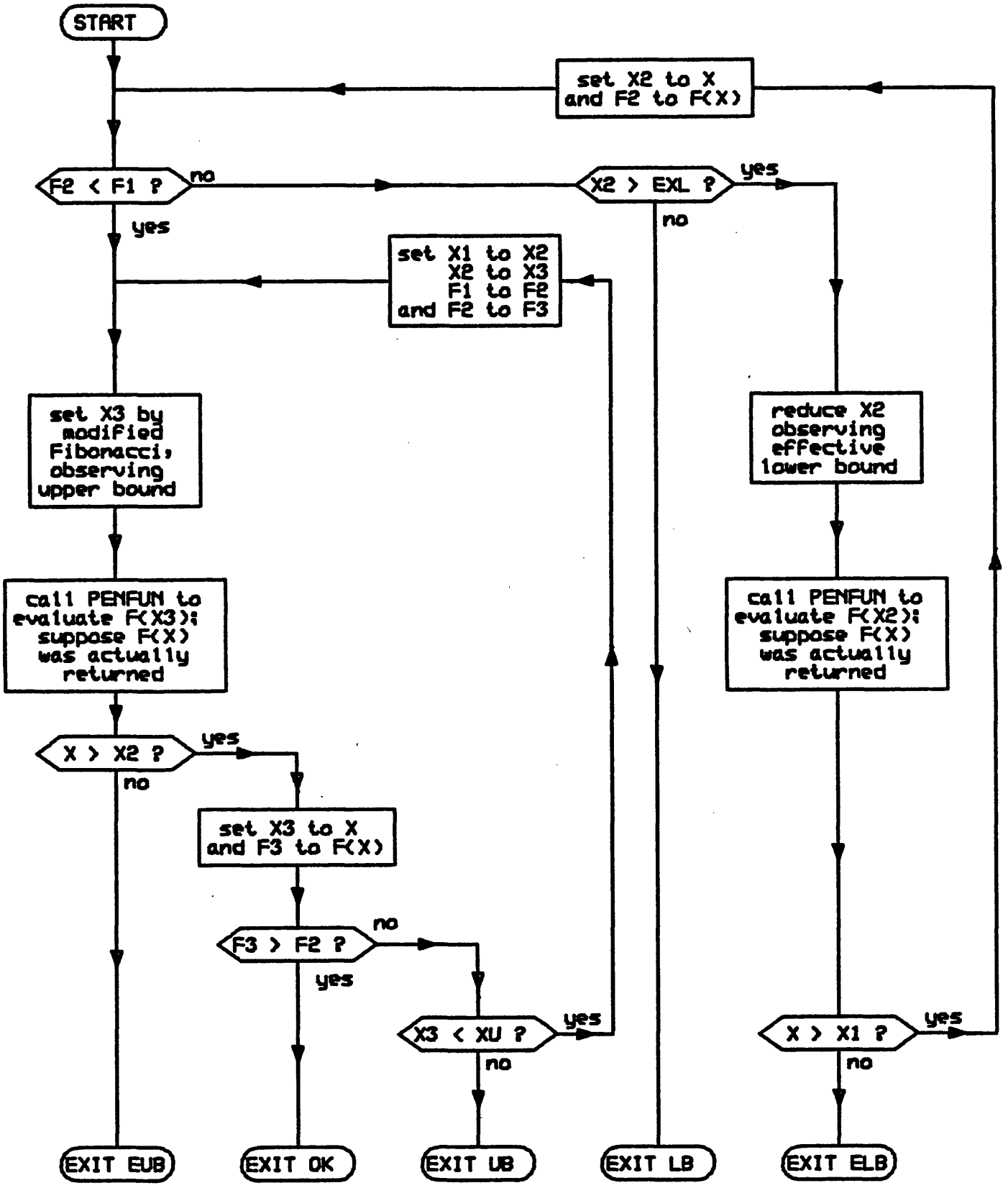


Figure 4.8 The bracketing attempt procedure

conditions, and is greater than X1 .

There are five possible exits as follows:

EUB - effective upper bound caused by objective function evaluation failure;

OK - bracketing successful;

UB - objective function still decreasing at upper bound;

LB - value of objective function at lower bound not reduced;

ELB - effective lower bound caused by objective function evaluation failure.

Exit condition ELB should not normally have occurred but was included to flag possible errors. In the main algorithm, univariate minimisation was attempted only if the second exit occurred.

It is not necessary to describe the minimisation procedure as it was based on the one mentioned in Section 3.2.1 and is described elsewhere. Modifications were of course necessary to accommodate PENFUN and to flag any failure of PENFUN .

4.3.5. Storing past solutions

For any given value of λ , an estimate of the solution to Equations (4.3) had to be supplied to GN for use as a starting point. This estimate was based on previous solutions or if none existed, the GN corrections for the overall problem at the current point. The simplest estimate to use was merely $\underline{\delta}$ for the previous value of λ i.e. a zero order interpolation/extrapolation method. However, this was soon rejected in favour of the linear based estimate of $\underline{\delta}(\lambda^{[q+1]})$: $\frac{\lambda^{[q+1]}}{\lambda^{[q]}}\ \underline{\delta}\ (\lambda^{[q]})$ (assuming $\lambda^{[q]} \neq 0$) . More sophisticated estimates,

which were expected to improve efficiency, required further sets of past values of λ and $\underline{\delta}$. Furthermore, the imposition of an upper limit on the number of sets to be retained required a decision as to which set to discard when the limit had been reached.

The initial methods of generating $\underline{\delta}$ estimates were based on the collocating polynomial in λ. This can be expressed as follows:

$$\underline{\Delta}(\lambda) = \sum_{i=1}^{k} \left\{ \prod_{\substack{j=1 \\ j\neq i}}^{k} \left(\frac{\lambda-\lambda_j}{\lambda_i-\lambda_j} \right) \underline{\delta}_i \right\} \tag{4.8}$$

where the degree of the polynomial is $k-1$, λ_i $(i = 1,2,\ldots,k)$ are distinct values of λ and $\underline{\delta}_i = \underline{\delta}(\lambda_i)$ $(i = 1,2,\ldots,k)$.

$\underline{\Delta}(\lambda)$ gives an estimate of $\underline{\delta}(\lambda)$ which is straightforward to evaluate.

Until k sets of values had been stored Equation (4.8) was used with k reduced appropriately, the first estimate being based on the GN corrections. Once the requisite number of sets had been retained, their number was held constant by discarding old data in favour of newly acquired data. Suppose we have the full quota of k sets of data corresponding to λ values $\lambda_1, \lambda_2, \ldots, \lambda_k$ where $\lambda_1 < \lambda_2 < \ldots < \lambda_k$. On acquisition of further data the set discarded was either that corresponding to λ_1 or to λ_k. The criterion used was to try to maintain a balance between the number of points above and below λ_p, where λ_p was the value giving the lowest objective function, ϕ_p.

The necessary steps are shown in Figure 4.9 where the new data is for $\lambda = \lambda_i$ at which the objective function is ϕ_i. First it was assumed that λ_i was placed in the set and the revised lowest objective function ϕ_r and corresponding λ value λ_r were determined. If the number of points below λ_r (r_b, say) exceeded the number above (r_a), the λ_1 set was discarded. Conversely, the λ_k set was discarded. If $r_b = r_a$, the set corresponding to the λ value furthest from λ_r was discarded.

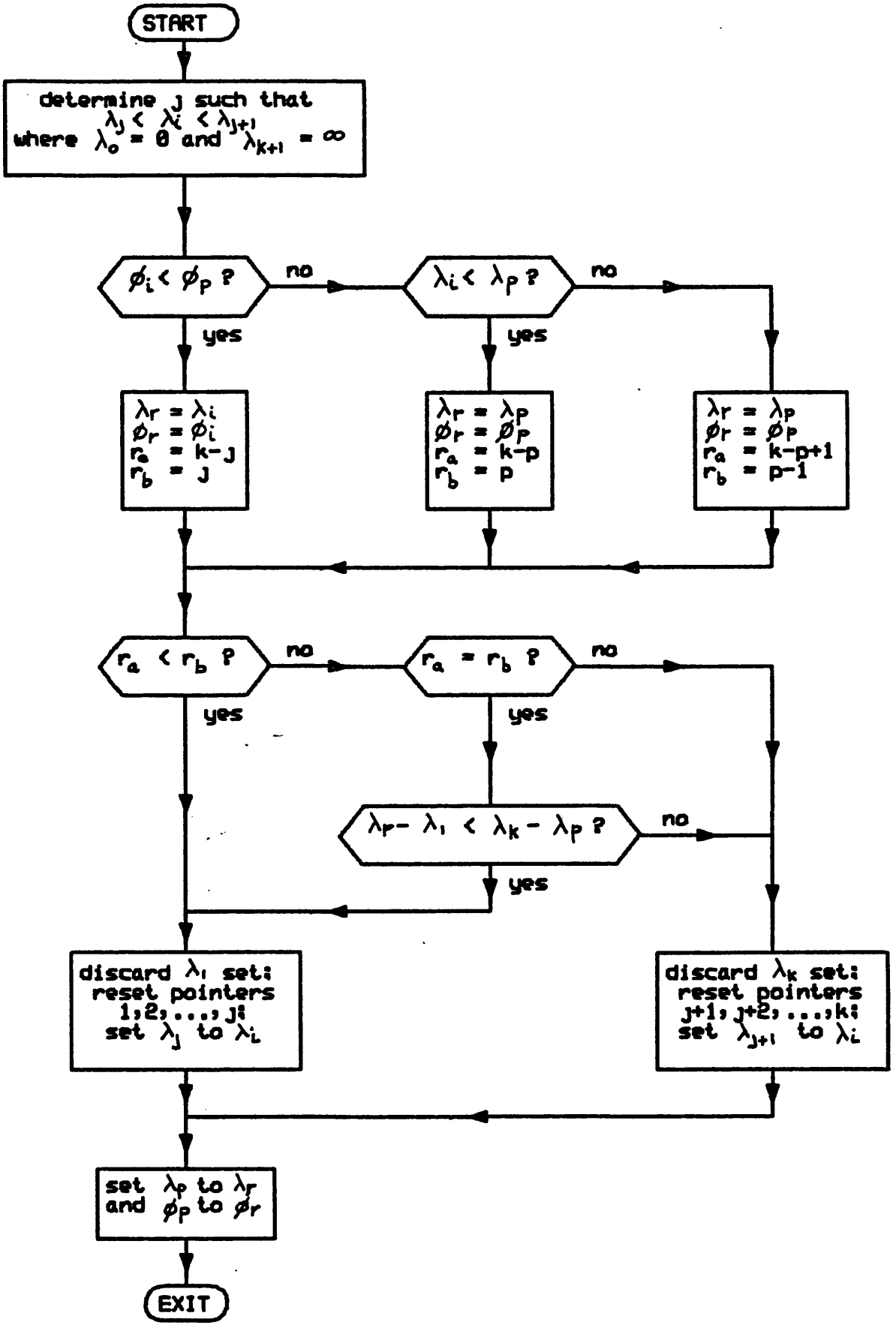


Figure 4.9 Method of discarding data sets

Finally the λ order and value of p had to be reset.

With each λ,ϕ pair, $\underline{\delta}$ is stored. Rather than shuffle this data around when the re-ordering was required, a system of pointers to fixed array positions was used. Two short procedures were required to administer the stored data.

4.4. Results obtained using the transistor model problem

Following the implementation of the first versions of the procedures described in the preceding sections, the transistor model problem was attempted using method SDB with a correction limit of 0.5. The linear $\underline{\delta}$-estimation technique obtained by setting k to 2 in Equation (4.8) was used. The results are shown in Table 4.1. which for each successful run gives the number of complete iterations of the second derivative method and in addition, the total number of GN iterations that the method required for the subsidiary problem of solving for $\underline{\delta}$. Note that termination conditions T3 and T4 were not used in these tests. Comparison with Table 3.7 shows that for the same correction limit, there were four starting points from which SDB converged and GN failed, namely for displacements $d = 1.2, 0.4, -2.2$ and -2.8. It was also pleasing to note that there were no starting points from which GN was successful and SDB failed. The large number of GN iterations which were required was attributed to the use of the linear interpolation technique; further tests using a better method were required.

More extensive trials were conducted using the quadratic $\underline{\delta}$-estimation technique obtained by setting k to 3 in Equation (4.8). First SDA was used with and without correction limiting from positive starting displacements. The results are shown in Table 4.2. At this stage termination conditions T3 and T4 were flagged but otherwise disregarded; such an occurrence is denoted by parentheses.

Initial displacement d	Termination criterion satisfied	No. of SDB iterations	No. of GN iterations
1.6	2		
1.4	2		
1.2	1	11	286
1.0	2		
0.8	1	5	114
0.6	1	5	115
0.4	1	5	165
0.2	1	3	30
-0.2	1	3	27
-0.4	1	5	188
-0.6	1	4	100
-0.8	1	6	100
-1.0	1	7	164
-1.2	1	8	271
-1.4	1	7	251
-1.6	1	8	284
-1.8	1	8	355
-2.0	1	10	547
-2.2	1	10	544
-2.4	2		
-2.6	2		
-2.8	1	9	252
-3.0	2		
-3.2	2		

Table 4.1 Linear SDB method with correction limit of 0.5 on transistor model problem

Initial displacement d	Termination criterion satisfied	No. of SDA iterations	No. of GN iterations
1.2	(3)2		
1.0	(3)2		
0.8	(3)2		
0.6	1	6	81
0.4	1	6	98
0.2	1	3	21

(a) No correction limiting

Initial displacement d	Termination criterion satisfied	No. of SDA iterations	No. of GN iterations
1.2	(3)2		
1.0	(3)2		
0.8	(3)(4)2		
0.6	1	5	78
0.4	1	5	103
0.2	1	3	21

(b) Correction limit of 0.5

Table 4.2 Quadratic SDA method on transistor model problem

Both methods showed one improvement over GN , from d = 0.4, but the method with correction limiting was unsuccessful from d = 0.8 , a point from which GN with the same correction limit reached the solution (even though it failed with more severe limiting). It was noted that linear SDB was successful from this point and later trials showed that SDB was more successful generally. Results for the normal quadratic SDB and SDC methods are shown in Tables 4.3 (no correction limiting) and 4.4 (correction limit of 0.5). Comparison with the equivalent GN trials reveals that in the unlimited versions, SDB had five successes above those of GN . These were at d = 1,2, 1.0, 0.8, 0.4 and -1.0. The success obtained from d = 1.2 was particularly noteworthy as GN had failed with all correction limits used. Furthermore, from d = 1.0 and d = 0.8 GN was successful only with one and two particular correction limits respectively. The failure of SDB from d = 0.8 was suspect and an examination of the results revealed that during one iteration a solution jump had taken place; it was expected that if this had been avoided the solution would have been reached.

SDC (unlimited) also revealed a solution jump, in this case in the d = -1.0 run. The interpolation procedure was then unable to minimise to the prescribed accuracy and the computer run time limit was reached. In both this and the SDB example, the solution jump occurred at the second value of λ generated by the Fibonacci sequence, i.e. double the first value. A higher initial value of α (Equation 4.7) would probably have overcome this. SDC (unlimited) had three successes not obtained with GN (unlimited): d = 0.4, -0.8 and 1.2.

Using the correction limit of 0.5 in the second derivative methods improved the range of d for which the solution was reached but the number of extra successes above those obtained with the equivalent GN method did not increase. There were three extra which were from the same starting points for both SDB and SDC, i.e. d = 0.4, -2.2 and -2.8.

Initial displacement d	Method SDB			Method SDC		
	Termination criterion satisfied	No. of main iterations	No. of GN iterations	Termination criterion satisfied	No. of main iterations	No. of GN iterations
1.4	(3)2			(3)2		
1.2	(3)1	6	209	(3)2		
1.0	1	6	88	(3)2		
0.8	1	5	111	(3)2		
0.6	1	6	69	1	5	40
0.4	1	6	85	1	5	68
0.2	1	3	19	1	3	20
-0.2	1	3	14	1	3	18
-0.4	1	3	27	1	3	40
-0.6	1	4	29	1	4	27
-0.8	(3)2			1	4	76
-1.0	1	6	134	time limit		
-1.2	2			1	6	151
-1.4	2			(3)2		
-1.6	2			(3)2		

Table 4.3 Quadratic SDB and SDC with no correction limiting on transistor model problem

Initial displacement d	Method SDB			Method SDC		
	Termination criterion satisfied	No. of main iterations	No. of GN iterations	Termination criterion satisfied	No. of main iterations	No. of GN iterations
1.0	(3)2			(3)2		
0.8	1	6	116	1	6	102
0.6	1	5	74	1	6	46
0.4	1	5	86	1	6	81
0.2	1	3	19	1	3	20
-0.2	1	3	14	1	3	18
-0.4	1	5	59	1	4	51
-0.6	1	4	53	1	5	52
-0.8	1	6	87	1	6	62
-1.0	1	7	100	1	7	138
-1.2	1	7	109	1	7	114
-1.4	1	7	119	1	7	117
-1.6	1	7	137	1	7	140
-1.8	1	7	211	1	8	178
-2.0	1	13	627	1	8	251
-2.2	1	11	330	1	9	248
-2.4	(4)2			2		
-2.6	(3)2			(3)2		
-2.8	1	9	204	1	9	173

Table 4.4 Quadratic SDB and SDC with correction limit of 0.5 on transistor model problem

It was noted that the successful starting points for unlimited SDB was not a subset of those for the limited version, so in an attempt to find a method which would combine these successful starting points a number of versions having different limiting factors (not less than 0.5) and different limiting methods on the two univariate searches, the λ-search and μ-search, were investigated at this time and later. The λ-search being non-linear gave scope for such interesting possibilities as limiting $\underline{\delta}$ components if they changed sign. However, these investigations were unsuccessful and the aim was not realised.

Unlimited SDB registered twice as many successes as unlimited GN for the set of (SDB) starting points investigated. It was therefore thought to be worthwhile to consider ways of making the second derivative algorithm more computationally efficient.

4.5. Factors affecting algorithm efficiency

The main area where improvements could be expected to be made was in the calculation of $\underline{\delta}(\lambda)$. The algorithms already described for determination of which values of λ to attempt to find $\underline{\delta}$ using GN were thought to be reasonably sound; apart from possible variations there four factors relating to the univariate search which influenced the efficiency of the algorithms were identified. These will be discussed in the following sections.

4.5.1. GN initial point estimation

A comparison of Tables 4.1 and 4.4 reveals that use of the higher order collocating polynomial substantially reduced the number of GN iterations required to solve the sub-problem of finding $\underline{\delta}$. The estimation of $\underline{\delta}$ presented to GN was clearly an important factor in the efficiency of the algorithm. By raising the value of k in

Equation (4.8), cubic and quartic estimation methods were examined to investigate whether further reductions in the number of GN iterations could be obtained.

Further methods investigated were based on the collocating quadratic in $\underline{\delta}$:

$$\Lambda_k(\delta_k) = \sum_{i=1}^{3} \left\{ \prod_{\substack{j=1 \\ j\neq i}}^{3} \left(\frac{\delta_k - \delta_{kj}}{\delta_{ki} - \delta_{kj}}\right) \lambda_i \right\} \qquad (k = 1,2,\ldots,n) \qquad (4.9)$$

where δ_{ki} (i=1,2,3) are 3 distinct values of the kth component of $\underline{\delta}$ and λ_i (i=1,2,3) are the corresponding values of λ .

The evaluation of $\underline{\delta}$ using this method is a little more complicated than for Equation (4.8). Expanding the right-hand sides of Equations (4.9) gives a quadratic expression for each of the δ_k . Substituting a fourth value λ_4 of λ for Λ_k , it was a simple matter to determine, for each $\underline{\delta}$ component, whether the appropriate quadratic had real solutions and to solve for them if it did. This gave two estimates of $\delta_k(\lambda_4)$; the one chosen was the one closer to the known value of $\delta_k(\lambda_i)$ where λ_i is the closest retained value to λ_4 . If the quadratic had no real roots, $\delta_k(\lambda_i)$ was used as the component estimate.

Whereas Equation (4.8) gives the same polynomial expression for each $\underline{\delta}$ component, this was not the case when using Equations (4.9). The reason for this more complicated method was that it could predict when a $\underline{\delta}$ component was going complex. It was thought that since it was able to do this, it might be better at approximating the $\underline{\delta}$ curves and that it might be possible to use the predictions of complex components to determine when to terminate the bracketing procedure. This estimation method is subsequently referred to as "quadratic reversed".

The final method investigated, "simplified quadratic reversed", had similar aims and was based on Equations (4.9) but with $\lambda_3 = \delta_{k3} = 0$ (k = 1,2,...,n), i.e. a quadratic interpolation through the origin.

Thus this method required only two sets of values to be stored.

4.5.2. Termination of GN

The five criteria used to indicate GN failure, T2 to T6, were described in Section 3.2.2. Failure of GN for any particular value of λ was not a disaster and was to be expected on some occasions. Even when the solution for $\underline{\delta}$ could have been reached eventually, it may have been more beneficial to cease the attempt and use a lower value of λ, for which, in the bracketing stage, the estimates of $\underline{\delta}$ were likely to be more accurate. It was therefore undesirable to allow GN to continue indefinitely and the most convenient means of preventing this was to set an upper limit on the number of iterations (termination condition T6). The default number in the Cutteridge algorithm was 200, too many for this application. During development of the second derivative methods this figure was gradually decreased. The final limit chosen was 10 iterations; there were indications that a still lower figure might be better but extensive tests were not undertaken.

The criterion used to indicate a successful termination of GN in the above trials was T1, i.e. a test on the size of the correction components. This required at least one matrix inversion to find the correction vector. It was decided to investigate termination of GN if the objective function of the sub-problem was sufficiently small, i.e.

$$\text{if } \theta(\underline{\delta}) \lesssim \varepsilon_6$$

for suitable values of ε_6. This effectively reduced the restrictions on the accuracy of $\underline{\delta}(\lambda)$. If the original estimate of $\underline{\delta}(\lambda)$ satisfied the condition, no GN iterations were used.

4.5.3. Termination of bracketing attempt

The condition for an upper limit on the bracketing attempt through failure of GN , which would occur if components of $\underline{\delta}$ became complex, was given in Section 4.3.3

i.e. $$\lambda_a - \lambda_b < \varepsilon_4 \lambda_b$$

where λ_a and λ_b are the lowest unsuccessful and highest successful values of λ respectively. The higher the value assigned to ε_4 , the more likely it was that termination of the bracketing attempt would occur.

4.5.4. Termination of univariate minimisation

Assuming the bracketing attempt was successful, the univariate minimisation came into play. The condition for termination of this was described in Section 3.2.1 ,

i.e. $$|\lambda^{[j]} - \lambda^{[i]}| < \varepsilon|\lambda^{[j]}| \quad \text{and} \quad |\lambda^{[k]} - \lambda^{[j]}| < \varepsilon|\lambda^{[j]}|$$

where $\lambda^{[i]}$ and $\lambda^{[k]}$ form the current bracket and $\lambda^{[j]}$ is a point between the two. Raising the value of ε reduces the accuracy to which the minimum is required.

4.6. Results of efficiency study

A number of timed trials using SDB without correction limiting were conducted for the various estimation procedures and various values of ε , ε_4 and ε_6 . Three starting points of the transistor model problem from which GN failed were chosen for these trials:
d = 0.8, 0.4 and -1.0.

The results obtained using the six estimation methods described are shown in Table 4.5. Algorithm efficiency using the estimation

methods based on Equation (4.8) improved as the order of the estimation algorithm increased until a collocating polynomial of degree four was used. The time taken by the three trials using the quartic method was the same as that taken by the linear method. The two methods based on Equations (4.9) took the longest time and examination of the results revealed that they were not particularly good at predicting non-real values of $\underline{\delta}$ components.

Estimation method	SDB iterations (GN iterations)			Total time taken (secs)
	d = 0.8	d = 0.4	d = -1.0	
Linear	5(114)	6(95)	6(163)	143
Quadratic	5(111)	6(85)	6(134)	134
Cubic	5(110)	6(84)	6(132)	132
Quartic	5(115)	6(88)	6(154)	143
Quadratic reversed	5(120)	6(97)	6(146)	145
Simplified quadratic reversed	6(163)	6(171)	6(188)	198

Table 4.5 Results obtained using different estimation methods

Although the cubic estimation method was marginally better than the quadratic, it was decided to continue to use the latter because it required one fewer set of past values to be stored and because of its compatability with the univariate minimisation procedure. The retained points were those used to define the new point and were therefore the minimum number necessary to ensure the new point was surrounded by retained data.

Table 4.6 shows the results for various values of ε_6, the GN termination constant. The total time taken steadily reduced as ε_6 was increased until eventually two of the test runs failed to reach the solution. This trial demonstrated that highly accurate values of $\underline{\delta}$ were not required and that a reduction in computer time of 50% was obtainable by accepting less accuracy.

ε_6	SDB iterations (GN iterations)			Total time
	d = 0.8	d = 0.4	d = -1.0	taken (secs.)
0.0	5(111)	6(85)	6(134)	134
10^{-10}	5(109)	6(72)	6(142)	122
10^{-8}	5(105)	6(67)	6(133)	117
10^{-6}	6(92)	6(54)	6(120)	104
10^{-4}	6(81)	6(43)	6(98)	93
10^{-2}	6(54)	7(35)	6(80)	75
10^{0}	6(41)	6(24)	7(49)	66
10^{2}	not attempted	failed	failed	-

Table 4.6 Results for various values of ε_6

Table 4.7 shows the results for various values of the bracketing termination constant ε_4 , the best result being obtained with a value of 5.0 . This implied that using the λ generation scheme of Section 4.3.4, once GN had failed during the bracketing stage it was not worth attempting to refine the previous value of δ . Because of the λ generation scheme used any value of ε_4 above 1.0 would have produced exactly the same results.

ε_4	SDB iterations (GN iterations)			Total time
	d = 0.8	d = 0.4	d = -1.0	taken (secs.)
0.05	5(111)	6(85)	6(134)	134
0.1	5(103)	6(85)	6(123)	130
0.2	5(103)	6(85)	6(121)	128
0.5	6(74)	6(85)	6(93)	111
5.0	6(57)	6(85)	6(88)	102

Table 4.7 Results for various values of ε_4

Table 4.8 shows the results for various values of the univariate minimisation accuracy constant ε , and indicates once again that strict accuracy was not required. The best time recorded was for the run which did no univariate minimisation, i.e. it attempted bracketing only.

ε	SDB iterations (GN iterations)			Total time
	d = 0.8	d = 0.4	d = -1.0	taken (secs.)
0.01	5(111)	6(85)	6(134)	134
0.1	5(111)	6(80)	6(133)	129
0.5	5(92)	6(72)	5(127)	113
10.0	6(90)	7(65)	6(121)	107

Table 4.8 Results for various values of ε

Having obtained these results, the accumulative effect of accepting lower accuracy was examined. ε_6 was set to 10^{-2}, ε_4 to 5.0 and ε to 10.0. With the exception of ε_6, these were the values producing the quickest execution times; ε_6 was set higher because of the observed danger of failure. In fact this combination failed for $d = -1.0$ so ε_6 was set to 10^{-4} and the trial repeated (Table 4.9). All three runs were successful and the execution time recorded was the quickest to date, 60% less than for the original method.

ε_6	ε_4	ε	SDB iterations (GN iterations)			Total time taken (secs)
			$d = 0.8$	$d = 0.4$	$d = -1.0$	
10^{-2}	5.0	10.0	7(26)	6(25)	failed	-
10^{-4}	5.0	10.0	7(32)	7(32)	8(63)	53

Table 4.9 Accumulative effect of easing accuracy requirements

The revised method was then tried from a range of starting points first using no correction limiting and then using a correction limit of 0.5. The results are given in Table 4.10 together with the equivalent results of the earlier method (taken from Tables 4.3 and 4.4). With one exception the number of GN iterations for solving Equations (4.3) was always fewer. The exception was for $d = -2.2$ with a correction limit of 0.5 for which the new method was unsuccessful. This reinforces the view that reduced accuracy should be accepted with caution when convergence to a solution is in doubt. However, the solution jump problem which occurred for $d = -0.8$ in the earlier version of SDB (unlimited) was no longer present. Although extensive trials were not conducted on a revised SDC method, some tests were done and it was noted that the solution jump for the $d = 1.0$ run was overcome as well.

Also included in Table 4.10 are the results of the revised SDB method for a correction limit of 0.2 which had not been tried previously.

As expected the range of successful starting points increased though there were only three improvements over the equivalent GN , at d = 1.4, 1.2 and 0.8. However, for all trials of the revised method there was no occasion where SDB failed and the equivalent GN method was successful. Furthermore, there were only three runs, all with a correction limit of 0.2 , where the number of GN iterations to the solution was less than the number of SDB iterations to the solution.

Initial displacement d	SDB iterations (GN iterations)				
	No correction limiting		Correction limit of 0.5		Correction limit of 0.2
	Previous result	New result	Previous result	New result	
1.6	*	*	*	*	failed
1.4	failed	failed	*	*	15(89)
1.2	6(209)	8(57)	*	*	13(69)
1.0	6(88)	7(34)	failed	failed	11(64)
0.8	5(111)	7(32)	6(116)	6(44)	11(80)
0.6	6(69)	6(40)	5(74)	6(40)	13(91)
0.4	6(85)	7(32)	5(86)	5(35)	6(44)
0.2	3(19)	3(10)	3(19)	3(10)	3(13)
-0.2	3(14)	3(7)	3(14)	3(6)	5(11)
-0.4	3(27)	4(11)	5(59)	5(19)	9(49)
-0.6	4(29)	4(16)	4(53)	5(20)	9(43)
-0.8	failed	6(57)	6(87)	7(36)	14(71)
-1.0	6(134)	8(63)	7(100)	8(62)	15(104)
-1.2	failed	failed	7(109)	8(60)	15(95)
-1.4	failed	failed	7(119)	7(68)	15(97)
-1.6	failed	failed	7(137)	8(71)	13(137)
-1.8	*	*	7(211)	12(200)	18(298)
-2.0	*	*	13(627)	15(278)	19(373)
-2.2	*	*	11(330)	failed	19(409)
-2.4	*	*	failed	failed	23(435)
-2.6	*	*	failed	failed	failed
-2.8	*	*	9(204)	9(99)	*

*not attempted

Table 4.10 Results of the revised SDB method and comparison with earlier results

4.7. Computing time comparison of second derivative methods and GN

The amount of work done in an iteration of any of the second derivative (SD) methods clearly exceeds that done in a GN iteration. A comparison of computing times for the methods was required but the research programs as they stood were not suitable for this because as they contained the

remains of many discontinued ideas their efficiency was impaired. It was decided instead to isolate the main components of the algorithms and to time these individually in order to construct a composite picture. The facilities offered by the operating system to do this were limited; it was necessary to repeat the required operation many times in order to get a reasonably accurate estimate of the time taken for one such operation. Using this method the following times were recorded for the transistor model problem;

evaluation of the objective function $f(\underline{x})$: $t_f = 0.0017$ seconds;

evaluation of the Jacobian : $t_j = 0.0052$ seconds;

evaluation of all second derivatives : $t_D = 0.041$ seconds;

given the above, extra time required for evaluation of sub-problem objective function $\theta(\underline{\delta})$: $t_F = 0.011$ seconds;

given the above, extra time required for evaluation of sub-problem Jacobian : $t_J = 0.012$ seconds;

solution of simultaneous linear equations using Crout's factorisation method : $t_i = 0.042$ seconds.

These figures clearly indicate that the SD methods will take longer than GN , assuming convergence in both cases, unless the number of GN iterations on the sub-problem is fewer than those required for the full problem. This cannot be expected in general and in fact never occurred on the transistor model problem. Using typical values for the number of function evaluations per iteration it was calculated that the ratio of time taken on an SD iteration to that for a GN iteration on the

full problem was approximately $2j + 1$ for SDA and SDB rising to $2j + 1.6$ for SDC, where j is the number of GN iterations of the sub-problem per SD iteration. This has been noted to vary from 2 to 20 which means that the SD method would normally take several times as long as GN if both converged to the same solution even allowing for the reduced number of SD iterations required. However, it should be remarked that for the same size problem, t_F and t_J are constant. Therefore the ratio becomes more favourable for the SD method if the complexity of the objective function increases, even though this could mean proportional increases in t_j and t_D.

Nevertheless, it was to be expected that the main use of the SD algorithms would be when GN failed. The question arose as to whether it was possible to determine when to change from the SD method to GN during the optimisation process, i.e. to determine when a point had been reached from which GN would converge to the solution. It was thought that one possible method would be to compare $\underline{\delta}(\lambda^{(p)})$ with the GN correction vector for the same value of λ. However, even at a point from which GN would converge to the solution it was found that the two sets of corrections could be quite different; different orders of magnitude, even corresponding components having opposite sign were noted. Thus it was decided to adopt the method used in the Cutteridge algorithm, i.e. to try GN from the initial point and from the points generated by each iteration of the SD method, terminating GN according to the criteria of Section 3.2.2. The results using this method in conjunction with SDB (unlimited) are given in Table 4.11. Execution times were reduced in all cases.

Initial displacement d	Total no. of SDB iterations	No. of GN iterations on sub-problem	Total no. of GN iterations on main problem
1.2	3	31	40
1.0	2	18	30
0.8	2	16	25
0.6	0	0	9
0.4	2	16	15
0.2	0	0	6
-0.2	0	0	5
-0.4	0	0	6
-0.6	0	0	7
-0.8	2	39	19
-1.0	1	18	13

Table 4.11 Results using GN prior to each SDB iteration

4.8. Results obtained using basic problems

During the development of the SD methods all the other problems described in Chapter III were used and the results obtained compared with those of GN/N (for SDA) and GN/M (for SDB and SDC). These results are not fully presented here because a consistent set, that is one with all the various parameters of the methods set to the same values for each run, was never produced. However some general results may be stated.

It was found that whenever the GN methods converged to a solution the equivalent SD method converged to a solution, and when the two solutions were the same, the SD method took fewer iterations. Normally, the total number of GN iterations required by the SD method in solving Equations (4.3) exceeded those required by GN on the full problem. An exception to this rule was observed for Rosenbrock's function where both SDA and SDB reached the solution in one main iteration consisting of 3 GN iterations. GN/N and GN/M took 3 and 16 iterations respectively on this problem.

It was necessary to enforce the original accuracy requirements to ensure that SDA and SDB converged to a solution of the modified

Rosenbrock function in one iteration; with relaxed accuracy requirements two iterations were required. SDC required two iterations on this problem even with the original specification, and on Rosenbrock's function it took five iterations. This was caused by non-unimodality in the λ-search thought to be due to inaccuracies in the points given by the μ-search to which SDC is susceptible.

On the HDS problem, SDA converged from the three starting points but SDB and SDC failed from $(50,50)^T$ and $(500,500)^T$, experiencing the same difficulty as GN/M from these points. From $(5,5)^T$, SDC was again appreciably worse than the other two SD methods.

4.9. Discussion

Comparisons of the three SD methods with their first derivative counterparts, have shown that in general the SD methods have a greater range of convergence and when both types of method converge to the same solution, the SD methods require fewer iterations. However, the use of correction component limiting has been noted to cause some deviation from the latter observation. SDB and SDC, which include an extra univariate search, have greater convergence ranges than SDA, which further demonstrates the usefulness of this technique. The accuracy problems that SDC encountered with the λ-search on simple examples suggests that SDB is the best of the three algorithms.

The convergence ranges of the SD methods were extended by the use of correction component limiting, but not as significantly as for GN. This result is not surprising because as the correction component limit is decreased, the directions given by the SD methods and by GN become more similar. Thus there is little point in using an SD method if severe correction limiting is to be imposed. However the remarks made earlier indicate that this may not be a disadvantage. With no

correction component limiting the convergence range of SDB for the transistor model problem was noted to be between two and three times that of GN .

The dangers of inaccuracy have again been demonstrated. Although a substantial reduction in computing time resulted from a relaxation of the accuracy conditions, the robustness of the SD algorithms eventually became affected though this may not have been due to the reduced accuracy of the univariate search parameter. However, even with this reduction in computing time, for most problems the SD algorithms were unable to compete with the efficiency of their GN counterparts when both converged. This will probably remain so unless either Equations (4.3) can be solved more quickly or unless further approximations are introduced. On the transistor model problem the former is thought unlikely because an evaluation of $\theta(\underline{\delta})$ took seven times as long as an evaluation of $f(\underline{x})$ even after the calculation of $\underline{s}$ and its first and second derivatives. The introduction of further approximations was avoided for the reasons already given. Thus it is thought that the main benefit of the SD methods is to be derived when GN fails. Certainly the complexity of the SD methods eliminates any thought that they might replace the need for a two-part optimisation method.

CHAPTER V

APPLICATION OF METHODS TO RC CIRCUIT DESIGN

Further engineering test examples were taken from the field of circuit design, an application area in which there has been a research interest at the University of Leicester for a number of years. One of the products of this research has been the development of a program for the synthesis of 3-terminal lumped linear networks containing two types of elements, resistors and capacitors, using the technique of coefficient matching (Calahan (67)). Development of the program, which is known as the Automated Network Design Program (ANDP), is still continuing and consequently several versions exist. One version, written in Algol, was suitable for the incorporation of the second derivative methods to facilitate further investigations.

The method of coefficient matching involves the use of an optimisation algorithm and in the particular version of ANDP used this was a two-part algorithm comprising the Fletcher-Reeves method of conjugate gradients (CG) and GN , i.e. a general function method and a sum of squares method.

The results of Chapter IV indicated that the range of convergence of the SD methods was greater than that of GN . As well as providing further examples for testing this premise, the network design application enabled the comparison of a general function method with the SD methods. If the extra convergence range of the SD methods could be obtained with a few CG iterations, little or no benefit would be derived from their use since they are far more complex. Alternatively, the results could give an indication as to whether the use of a three-part optimisation method would be beneficial.

5.1. The Automated Network Design Program

The main features of ANDP have been described by Cutteridge and

Krzeczkowski (68) although the program given in the appendix of their report does not correspond exactly to the version used by the author.

The performance of any 3-terminal RC network is defined completely by a triplet of short-circuit admittance functions which are ratios of polynomials in the complex frequency. From an initial network, ANDP attempts to find a suitable network structure complete with network element values such that polynomials specified by the user are realised to within a multiplicative constant by a prescribed accuracy. The initial network, which is also specified by the designer, must yield polynomials of the requisite degrees.

Given the values of the network elements, the corresponding polynomial coefficients can be evaluated. Thus the problem is reduced to one of solving a set of m simultaneous equations in n unknowns where m is the number of coefficients to be matched and n is the number of elements plus the number of common factors (see later). However, a suitable solution to the equations, i.e. one with positive element values, may not always exist for the particular arrangement of RC elements under consideration. For this reason, ANDP has an automatic facility for adding or removing an element when it appears that the specified polynomials will not be realised with the current topology. Thus the program relies heavily on the optimisation algorithm used to attempt to solve the simultaneous equations and for this application, the two-part CG-GN algorithm is the best of those tried so far (Krzeczkowski (69)).

The basic procedure is as follows. When an RC network topology has been defined a number of CG iterations are evaluated followed by a number of GN iterations. The criterion for changing from the former method to the latter is based on the value of the objective function and its rate of decrease over a number of iterations. If GN succeeds in reducing the objective function below a certain small value (1.0×10^{-13}) the network coefficients have been matched to the desirable accuracy and

and the program terminates. Otherwise one of the several alternative GN termination criteria must be satisfied eventually indicating that the polynomial coefficients are unlikely to be realised with the current topology. The removal of an element is effected if over a number of iterations its value decreases at an increasing rate. Since the optimisation variables are the logarithms of the network elements, element values must remain positive though they can become very small. If neither element removal nor successful GN termination is indicated, ANDP will attempt to find a suitable element to add to the network. This is done by examining the effect of adding a single element to the existing network between each pair of nodes in turn. Both types of element, resistor and capacitor, are considered unless the current topology already has an element of that type between the pair of nodes under examination. The value assigned to the additional element is the one which minimises the objective function when other variables are held constant at their current values; the necessary value of the new element can be calculated easily. For each of the new networks thus defined, a single GN iteration is calculated. Those networks in which the new element decreases in value are considered no further. Also discarded are those networks which have too many variables whose GN corrections exceed a certain amount, this indicating instability. From the remaining networks, the one which has the lowest objective function is chosen and the process restarted by applying CG iterations to the new network. This network modification stage is known as the virtual element analysis. Should all possible networks be discarded, the program terminates having failed in its objective. ANDP will also terminate if one particular element shows a tendency to increase at an increasing rate. The latest developments of ANDP include a facility for changing the number of nodes of the network in the event

of virtual element analysis failure (Savage (70)).

5.2. The trial problems

The actual example on which this investigation was based was taken from the report by Cutteridge and Krzeczkowski. The desired admittance functions are given by:

$$Y_{11} = Y_{22} = \frac{\Delta_{22}}{\Delta_{1122}} = \frac{\Delta_{11}}{\Delta_{1122}} \; ; \; -Y_{12} = \frac{\Delta_{12}}{\Delta_{1122}}$$

where $\Delta_{11} = \Delta_{22} = (p + \alpha)\ (1197p^3+56613.14p^2+28368.584p+191.184)$

$\Delta_{12} = (p + \alpha)\ (3p^3-1.14p^2+197.176p+77.616)$

$\Delta_{1122} = (p + \alpha)\ (800000p^2+408000p+3840)$

and p is the complex frequency variable.

The inclusion of the common factor $(p + \alpha)$ is necessary to eliminate the negative coefficient that would otherwise occur in the polynomial expression for Δ_{12} ; network theory demands non-negative polynomial coefficients for an RC realisation. Thus Δ_{11} , Δ_{22} and Δ_{12} are quartics and Δ_{1122} is a cubic which gives a total of 19 polynomial coefficients. An attempt to match these coefficients therefore yields 19 equations.

The number of variables is equal to the number of network elements plus one (for α). This quantity varies during a run of ANDP unless a solution is found without the addition or removal of elements of the original network topology. If the network topology is altered the effect of changes to the program is increased so that even minor changes can radically modify the program path.

This was illustrated by the different solutions given by the version of ANDP reported by Cutteridge and Krzeczkowski, which will be called

ANDP1 , and the version of ANDP made available to the author (ANDP2) when both were used on the test problem with equivalent data. The initial network in both cases consisted of 4 resistors, 4 capacitors and 5 nodes plus the reference node 0 . Initial values of the variables were set to 1.0 . The network modifications effected by ANDP2 are shown in Figure 5.1. The first modification was the addition of a resistor between nodes 0 and 4 ; the second modification was the addition of a capacitor between nodes 0 and 3 ; the third and final modification was the addition of a resistor between nodes 1 and 2 . ANDP1 is reported to have made the same first and second modifications, but a different third modification. Three subsequent topology modifications were required before the solution network (Solution A) shown in Figure 5.2 was obtained. The solution network given by ANDP2 (Solution B) is shown in Figure 5.3.

Since it would have been difficult to draw any conclusions from a comparison of different versions of ANDP when topological modification was necessary, it was decided to use initial network structures from which a solution could be obtained by merely changing element and common factor values. The two topologies used were those of Solution A and Solution B , and will be denoted Topology A and Topology B respectively. Thus prior to topological modification the former having 12 elements yielded 19 simultaneous equations in 13 unknowns and the latter having 11 elements gave 19 simultaneous equations in 12 unknowns.

5.3. The modified version of ANDP2

In order to assess the use of the SD algorithms on these problems ANDP2 was modified to produce a third version, ANDP3 , to enable a prescribed number of iterations of SDB or SDC to take place at the

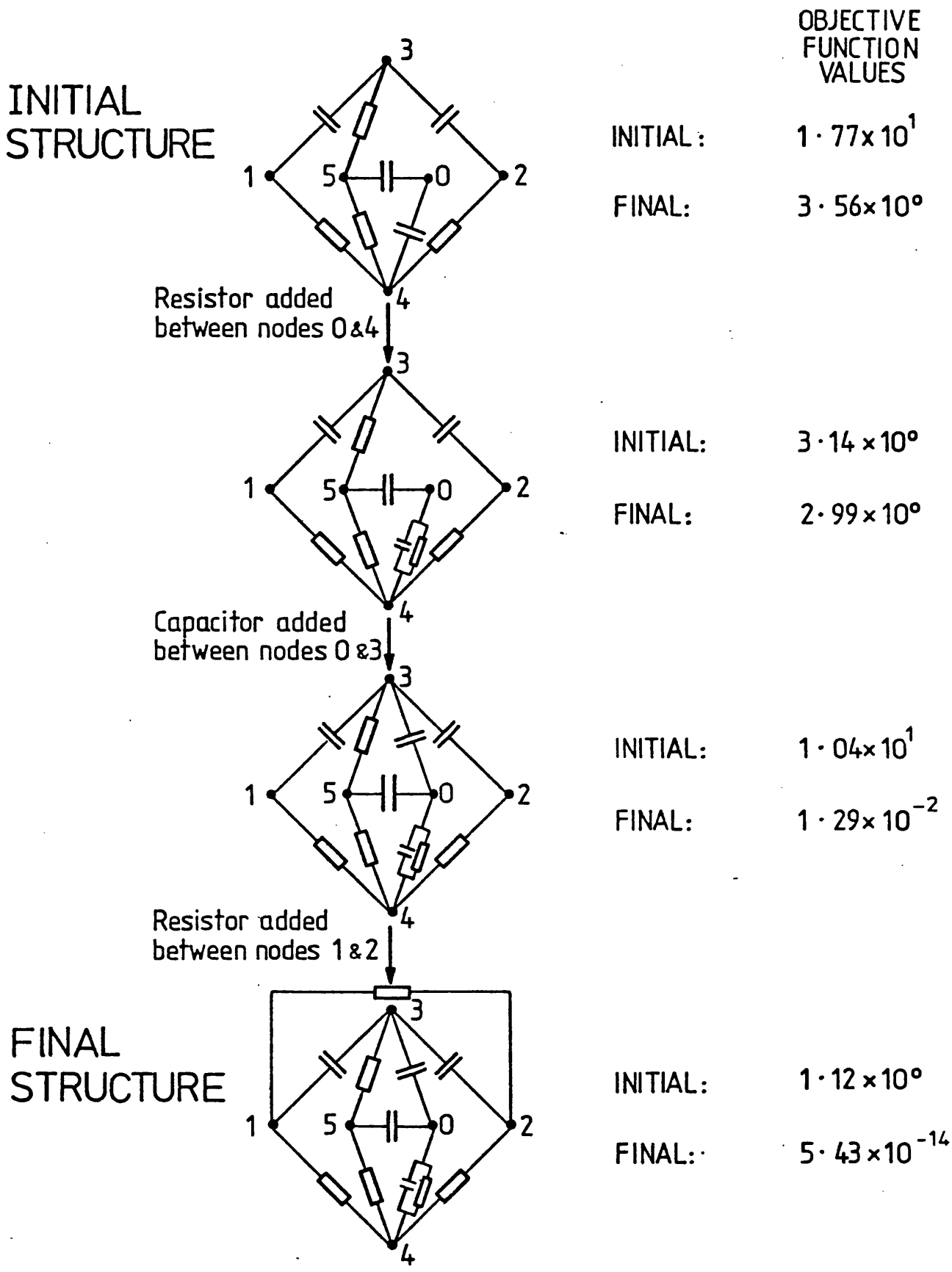
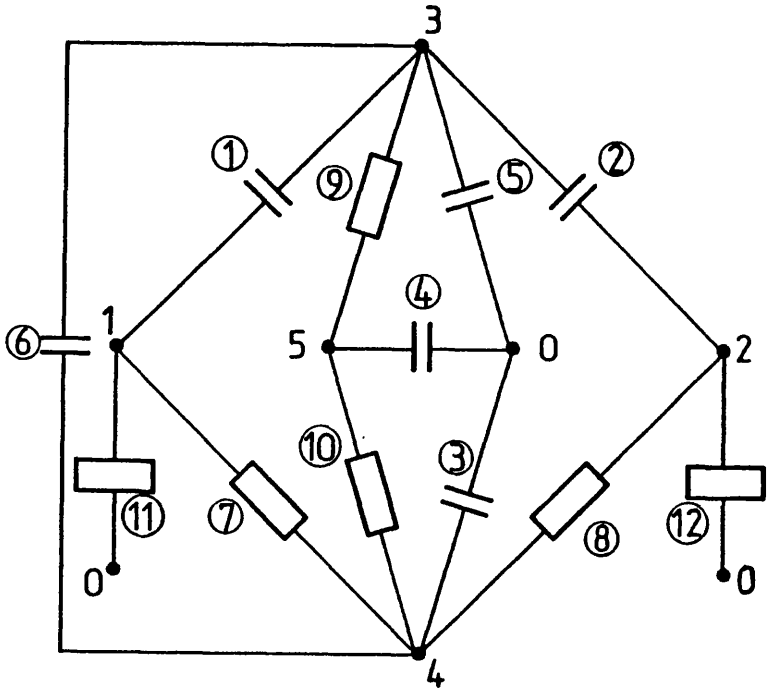


Figure 5.1 Network modifications given by ANDP2



ELEMENT AND COMMON FACTOR SOLUTION VALUES

$$x_1^* = 1.500 \times 10^{-3}$$
$$x_2^* = 1.500 \times 10^{-3}$$
$$x_3^* = 6.716 \times 10^{0}$$
$$x_4^* = 1.086 \times 10^{0}$$
$$x_5^* = 5.864 \times 10^{-1}$$
$$x_6^* = 1.060 \times 10^{-2}$$
$$x_7^* = 4.043 \times 10^{-2}$$
$$x_8^* = 4.043 \times 10^{-2}$$
$$x_9^* = 5.850 \times 10^{-1}$$
$$x_{10}^* = 8.616 \times 10^{-1}$$
$$x_{11}^* = 2.958 \times 10^{-2}$$
$$x_{12}^* = 2.958 \times 10^{-2}$$
$$\alpha^* = x_{13}^* = 1.937 \times 10^{0}$$

Figure 5.2 Solution A (as given by ANDP1)

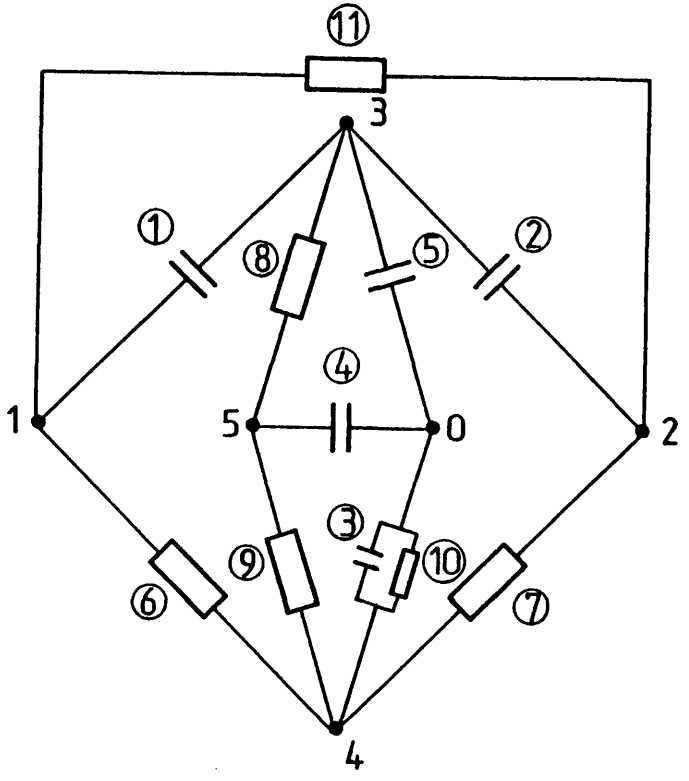


Figure 5.3 Solution B (as given by ANDP2)

ELEMENT AND COMMON FACTOR SOLUTION VALUES

$$x_1^* = 1.500 \times 10^{-3}$$
$$x_2^* = 1.500 \times 10^{-3}$$
$$x_3^* = 2.006 \times 10^{1}$$
$$x_4^* = 4.492 \times 10^{0}$$
$$x_5^* = 5.970 \times 10^{-1}$$
$$x_6^* = 7.000 \times 10^{-2}$$
$$x_7^* = 7.000 \times 10^{-2}$$
$$x_8^* = 5.746 \times 10^{-1}$$
$$x_9^* = 2.224 \times 10^{0}$$
$$x_{10}^* = 1.024 \times 10^{-1}$$
$$x_{11}^* = 2.541 \times 10^{-7}$$
$$\alpha^* = x_{12}^* = 1.194 \times 10^{0}$$

point when the program would normally have changed optimisation methods from CG to GN . After the prescribed number of SD iterations the program would enter the GN phase and proceed as normal. Thus the optimisation algorithm became a three-part process comprising CG - SDB (or SDC) - GN . The second derivatives required were calculated numerically. Since there were more simultaneous equations than unknowns, generalised Gauss-Newton was applied to Equations (4.3) in the SD methods.

A further modification to ANDP2 was the addition of a facility to enable a prescribed number of extra CG iterations to be calculated at the point when the optimisation method would normally have been changed in favour of GN . This facility was used to compare CG with SDB and SDC .

5.4. Results obtained

In the first set of trials ANDP2 and ANDP3 were supplied with an initial network consisting of Topology A with element and common factor values set to 1.0 , this being the initial value recommended in Cutteridge and Krzeczkowski's report should no other information be available. ANDP3 was tried with one and two iterations of each of the two SD algorithms, a total of four runs. Neither ANDP2 nor ANDP3 reached any solution. ANDP3 with SDC failed with overflow problems while the other three runs all failed to find a suitable new network at the first virtual element analysis.

Consequently the initial values applied to Topology A were moved closer to the solution, by either halving or doubling as appropriate, provided this did not overshoot the solution value. The new initial vector of variables was therefore:

$$\underline{x}^{(o)} = (0.5,0.5,2.0,1.0,1.0,0.5,0.5,0.5,1.0,1.0,0.5,0.5,1.0)^{T} .$$

Once again ANDP2 failed to reach Solution A but ANDP3 with one SDB iteration prescribed was successful. The criterion for changing from CG was satisfied after 37 iterations; following the single SDB iteration 14 GN iterations were required to reach the solution. Use of SDC within ANDP3 again failed.

A number of trials using additional CG iterations followed by the normal process through GN were then conducted. The additional iterations were used on the initial topology only. The results are shown in Table 5.1. The standard version of ANDP2 entered a virtual element analysis (VEA) after 3 GN iterations. The program eventually failed at a later, unsuccessful virtual element analysis. One or two extra CG iterations on the initial topology were sufficient to direct the process to a solution, but not that of the initial topology. The solution given is shown in Figure 5.4. Four or more extra CG iterations always gave Solution A , though it is interesting to note that the number of GN iterations required did not decrease monotonically as the number of CG iterations increased.

Using Topology B with the recommended initial values ANDP3 converged to Solution B with either one SDB iteration or one SDC iteration. 49 CG iterations were required before the CG termination criterion was satisfied, thereafter convergence to the solution was obtained with one iteration of either SD method followed by 17 GN iterations. The results for ANDP2 are shown in Table 5.2. Even with extra CG iterations, Solution B was never obtained. The basic version failed to reach any solution; at the first virtual element analysis a resistor was added and the program eventually terminated at a later virtual element analysis. Two of the runs reached a fourth solution, Solution D , shown in Figure 5.5. Two other runs terminated because the run time limit was reached although a generous amount of time

On initial network topology				Result of first virtual element analysis	Reason for termination
Extra CG iterations	Total CG iterations	Total GN iterations	Solution A found?		
0	37	3	no	capacitor added	VEA failure
1	38	11	no	capacitor removed	Solution C found
2	39	11	no	capacitor removed	Solution C found
4	41	21	yes	not entered	Solution A found
8	45	21	yes	not entered	Solution A found
16	53	12	yes	not entered	Solution A found
32	69	26	yes	not entered	Solution A found
64	101	39	yes	not entered	Solution A found

Table 5.1 Results using extra CG iterations in ANDP2 on Topology A

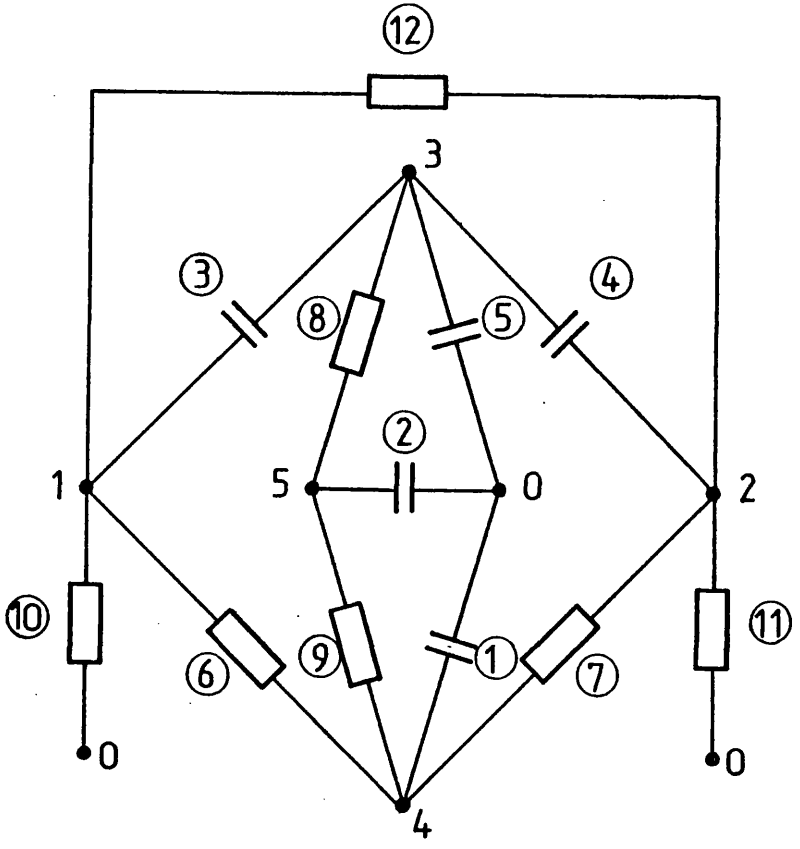


Figure 5.4 Solution C

ELEMENT AND COMMON FACTOR SOLUTION VALUES

$$x_1^* = 6.732 \times 10^{0}$$
$$x_2^* = 1.059 \times 10^{0}$$
$$x_3^* = 1.500 \times 10^{-3}$$
$$x_4^* = 1.500 \times 10^{-3}$$
$$x_5^* = 5.970 \times 10^{-1}$$
$$x_6^* = 4.042 \times 10^{-2}$$
$$x_7^* = 4.042 \times 10^{-2}$$
$$x_8^* = 5.760 \times 10^{-1}$$
$$x_9^* = 8.558 \times 10^{-1}$$
$$x_{10}^* = 2.958 \times 10^{-2}$$
$$x_{11}^* = 2.958 \times 10^{-2}$$
$$x_{12}^* = 2.628 \times 10^{-7}$$
$$\alpha^* = x_{13}^* = 1.942 \times 10^{0}$$

On initial network topology				Result of first virtual element analysis	Reason for termination
Extra CG iterations	Total CG iterations	Total GN iterations	Solution B found?		
0	49	6	no	resistor added	VEA failure
1	50	5	no	resistor added	Solution D found
2	51	5	no	capacitor added	large element value
4	53	5	no	resistor added	Solution D found
8	57	5	no	VEA failure	VEA failure
16	65	5	no	VEA failure	VEA failure
32	81	7	no	resistor removed	run time limit
64	113	8	no	resistor removed	run time limit

Table 5.2 Results using extra CG iterations in ANDP2 on Topology B

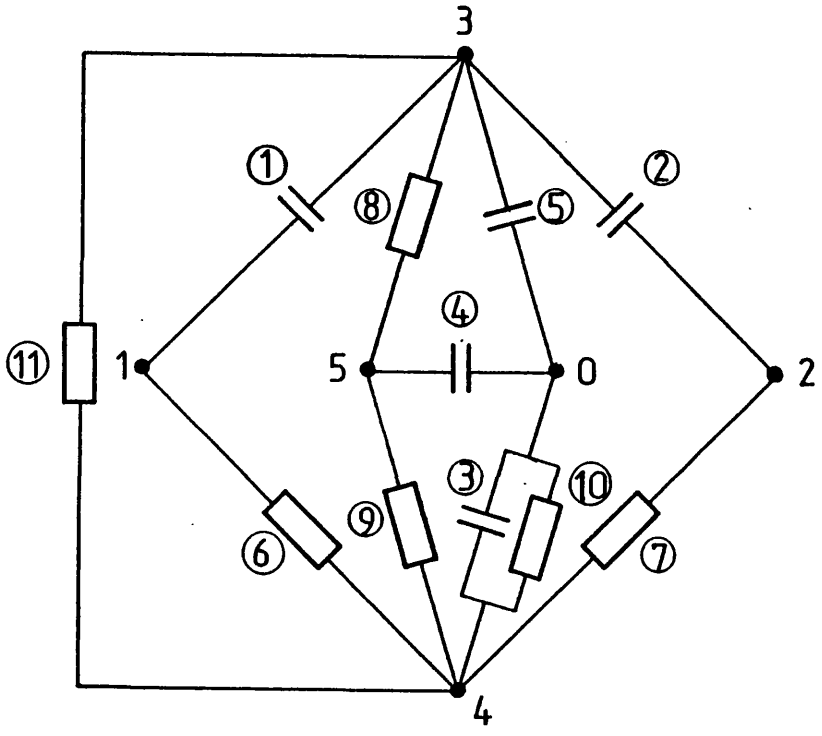


Figure 5.5 Solution D

ELEMENT AND COMMON FACTOR SOLUTION VALUES

$$x_1^* = 1.500 \times 10^{-3}$$
$$x_2^* = 1.500 \times 10^{-3}$$
$$x_3^* = 2.009 \times 10^{1}$$
$$x_4^* = 4.464 \times 10^{0}$$
$$x_5^* = 5.970 \times 10^{-1}$$
$$x_6^* = 7.000 \times 10^{-2}$$
$$x_7^* = 7.000 \times 10^{-2}$$
$$x_8^* = 4.800 \times 10^{-1}$$
$$x_9^* = 2.101 \times 10^{0}$$
$$x_{10}^* = 1.024 \times 10^{-1}$$
$$x_{11}^* = 5.396 \times 10^{-2}$$
$$\alpha^* = x_{12}^* = 1.078 \times 10^{0}$$

was allowed enabling many topological modifications before the limit became effective. Two further runs terminated at the first virtual element analysis and one run failed because of a large element value.

5.5. Discussion

The application of the SD sum of squares optimisation methods to RC network design has further demonstrated the greater range of convergence to a solution that is obtainable by use of these methods in preference to GN . This was the case on both of the trial initial networks. On the first example use of CG iterations overcame the convergence deficiency of GN . This was not so on the second example where the CG-GN optimisation method failed to give the required solution, which in some senses was the solution closest to the initial estimate, despite the calculation of CG iterations additional to those normally used. In this case the new SD optimisation methods were undoubtedly of benefit. It is possible that in the second example an alternative general function minimisation method would yield a point from which GN would converge to the required solution, but after several years of research on the network design program by various workers, CG is still the favoured method for this application.

Of the two SD methods tried, SDB again appears to be better. Although it would probably be possible to overcome the overflow problems encountered by SDC , the implication is that SDB is more stable and therefore more reliable.

CHAPTER VI

FURTHER EXTENSIONS TO SECOND DERIVATIVE ALGORITHMS

The aim of this chapter is to discuss two extensions to the second derivative algorithms which are more radical than those of Chapter IV;

(i) the use of solutions to Equations (4.3) other than the one given by $\underline{\delta} = \underline{0}$ at $\lambda = 0$,

(ii) the use of equations other than Equations (4.3) or (4.4).

To simplify explanation, correction vectors $\underline{\delta}$ derived from the vector function $\underline{\delta}(\lambda)$ satisfying the condition expressed in (i) are denoted by $\underline{\delta}_o$; others are simply referred to as "alternative correction vectors" which are derived from "alternative solutions" of Equations (4.3). Thus "$\underline{\delta}_o$ - method" is a general term which can be applied to methods SDA , SDB and SDC .

6.1. Use of alternative correction vectors

Of the many possible real solutions of Equations (4.3) only one has been considered so far. The question that arose was whether the alternative solutions could be used beneficially. Two possible uses were conceived: the provision of further solutions to a multi-solution problem and the provision of an alternative strategy when a problem solution was not reached by the particular $\underline{\delta}_o$ - method in use. The first of these can be illustrated by considering a multi-solution simultaneous quadratic equation problem where $m = n$, e.g. the modified version of Rosenbrock's problem, which is precisely represented by Equations (4.3). The alternative correction vectors therefore yield alternative problem solutions.

The provision of an alternative strategy when the $\underline{\delta}_o$ - method failed was seen as analagous to the restart facility of the Cutteridge algorithm

with the minor difference that the latter was based on univariate multimodality. It should be remarked that in the SD algorithms multimodality in the univariate searches (cf. Figure 4.4) provided a further possibility for an alternative strategy but lack of time prevented investigation.

6.1.1. Generation of alternative correction vectors

Chapter IV described the method by which determination of $\underline{\delta}_o$ for a particular value of λ was attempted based on the knowledge of the solution to Equations (4.3) at $\lambda = 0$. Minor modifications to this method enabled it to be applied to alternative solutions once $\underline{\delta}(\lambda)$ had been determined for some λ, wherein lay the difficulty; no value of λ existed for which $\underline{\delta}$ could be immediately evaluated.

Initial attempts at finding alternative solutions used the Cutteridge algorithm suitably modified. Instead of terminating the algorithm when convergence was indicated (cf. Figure 4.1) the process was directed to the next restart point. The solution search was terminated when all restart points had been used or when the prescribed computer run time had expired. In order that some restart points were generated it was necessary to force the first iteration to be of the descent type.

Trials on the use of alternative correction vectors were required but although the above method was moderately successful at generating alternative solutions, it was very expensive in terms of both descent and GN iterations. It was decided to attempt to generate solutions by means of GN only and incorporate this method in the previously developed algorithms. The occurrence of solution jumps in earlier work suggested that this was feasible and indeed some success was obtained by merely applying GN from a starting point derived from changing the signs of the starting point which yielded the $\underline{\delta}_o$ - solution.

The version which was incorporated into the SD algorithms was as follows. For ease of implementation ε_2, the value to which λ was set at the beginning of each SD iteration (see Section 4.3), was chosen for the value of λ at which solutions to Equations (4.3) would be sought. The normal procedure was used to determine $\underline{\delta}_o(\varepsilon_2)$; if this failed further solutions were not generated. If $\underline{\delta}_o(\varepsilon_2)$ was found it was defined as an origin from which orthogonal directions were generated using the Gram-Schmidt orthogonalisation process (see Birkhoff and MacLane (71)), the initial direction being defined by the original estimate of $\underline{\delta}_o(\varepsilon_2)$ that had been given to GN. Along the orthogonal directions, on both sides of the origin, points were generated by the Fibonacci sequence and at these the value of θ, obtained by substituting for $\underline{\delta}$ in Equation (4.5), was examined. If a decrease in θ was discovered, provided the previous point had given an increase in θ, GN was initiated. This procedure is illustrated in Figure 6.1. The first step length was calculated from the range of interest and the number of points required, two parameters to the algorithm. When GN converged to a new solution it was stored together with the GN starting point to be used later to generate a further set of orthogonal directions if required. Once the supply of new solutions had been exhausted the algorithm would terminate.

The use of derivative information at the θ evaluation stage was also considered. For example, in Figure 6.1, a negative gradient at the sixth point (counting from the left) would indicate the presence of a minimum which could not be deduced from an examination of θ values alone at the points shown. However, in practice derivative evaluations involved a lot of work for very little reward so their usage was dropped.

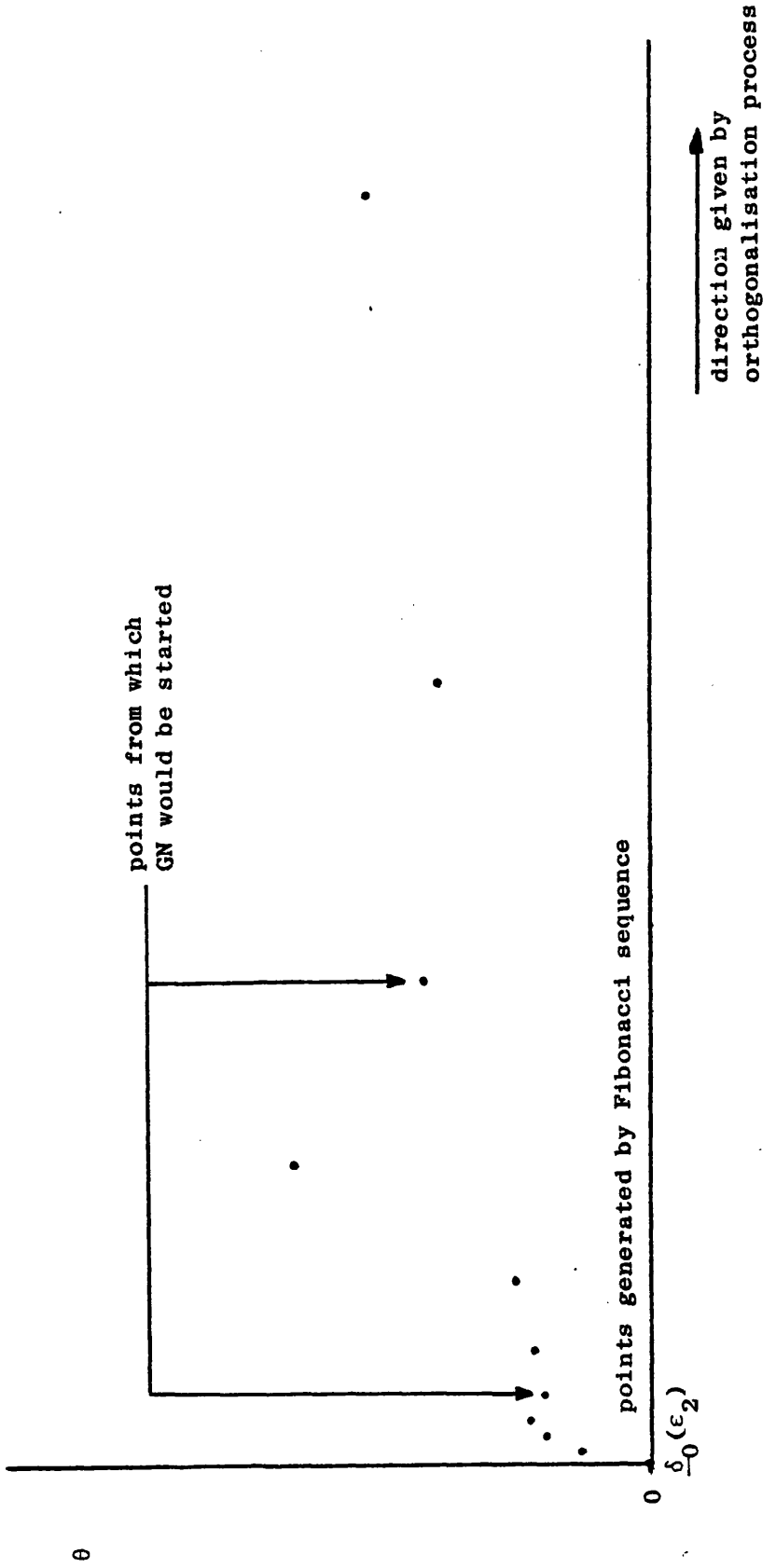


Figure 6.1 Multi-solution search procedure

6.1.2. Verification of generation method

In order to verify that the method of generating multi-solutions was reasonably effective it was applied to two problems which were known to have several solutions. The first one used was Brown and Gearhart's four-cluster problem (72), which is defined by:

$$s_1 = (x_1 - x_2^2)(x_1 - \sin x_2)$$

$$s_2 = (\cos x_2 - x_1)(x_2 - \cos x_1) \ .$$

This pair of equations has an infinite number of solutions, four of which are close together in the first quadrant. The approximate values of these are:

$$\underline{x}_1^* \doteq (0.68, 0.82)^T \ , \quad \underline{x}_2^* \doteq (0.64, 0.80)^T$$

$$\underline{x}_3^* \doteq (0.71, 0.79)^T \ , \quad \underline{x}_4^* \doteq (0.69, 0.77)^T \ .$$

Brown and Gearhart conducted extensive tests on this problem using several deflation techniques from eleven starting points, only two of which were reported. In no run were more than two of the four solutions of the first quadrant found; this was attributed to the tendency of the application of deflation at a point to direct the optimisation algorithm away from its vicinity.

The orthogonalisation method with GN included was applied from one of the initial points given, namely $(0.9, 1.0)^T$. GN was initiated from this point to give a solution to define as an origin after which the objective function was evaluated at 15 points in each orthogonal direction on each side of the origin. GN was again used where indicated and further sets of orthogonal directions were generated as solutions were found.

When the search was restricted to the first quadrant, only two

solutions were found. When the search was expanded to four quadrants, seven distinct solutions were found, including the four in the first quadrant.

The second problem used was the modified Rosenbrock problem to which the same procedure was applied from the initial point $(-30,5)^{\tau}$. All four solutions were found in a total of 17 GN iterations.

The results of the two trial problems indicated that the method was adequate at finding multiple solutions.

6.1.3. Results obtained

Initial trials on the use of alternative correction vectors were conducted on the transistor model problem with the problem parameters constrained. Figure 4.2 showed plots of $\log.f(\underline{x} + \underline{\delta}(\lambda))$ against λ for various $\underline{\delta}$-paths found when the components of $\underline{x}$ were set to 1.0 above the components of $\underline{x}_1^*$ for the transistor model problem. The path labelled "A" corresponds to the $\underline{\delta}_o$ - solution. Of the remaining $\underline{\delta}$ - paths, five were traced back through $\lambda = 0$. These are labelled "B", "C", "D", "E" and "F". From the initial point defined above, five runs were conducted using one of the alternative correction vectors on the first iteration and $\underline{\delta}_o$ - solutions on subsequent iterations. Apart from this modification, the optimisation method used corresponded to SDB with a correction limit of 0.5. Normal SDB using only $\underline{\delta}_o$ solutions failed to reach a problem solution from this point (see Table 4.10). The method using the alternative correction vectors fared no better.

A similar procedure was adopted using two alternative $\underline{\delta}$ - solutions from $d = -2.0$, this time with no correction limiting. Whereas the normal SDB method failed, use of the alternative $\underline{\delta}$ - solutions in the first iteration eventually led to the solution in each case. During

further trials on both SDB and SDC a few more successes were recorded from points from which the sequence of $\underline{\delta}_o$ - solutions failed.

In order to examine this method further, and also to determine the effect of using alternative $\underline{\delta}$ - solutions to find multiple solutions of an application problem, the transistor model problem was used without constraining the variables. In Section 3.1.6 two problem solutions were given, one of which had negative components. Although this second solution is invalid in the engineering sense, it is perfectly valid when used as an optimisation test case. It was also required to know how the use of alternative $\underline{\delta}$ - solutions compared with simply generating different initial points for the SD methods. Although GN was used in the orthogonalisation procedure described in Section 6.1.1, there was no reason why this could not be substituted by SDB and the orthogonalisation used to generate starting points for the original problem. Although ideally the origin should be a solution, if one had not been found previously an arbitrary point and direction could be used to start the orthogonalisation. This technique was also tested on the unconstrained transistor model problem using SDB .

Table 6.1 shows the results obtained using GN and SDB on the transistor model problem without imposing the constraint of positive parameter values. The table shows which, if either, of the two problem solutions was given on a number of trials for various values of d , where as before the initial variable values were given by:

$$x_i^{(o)} = \max(x_{1i}^* + d,\ 0.1) \qquad \text{for} \qquad i = 1,2,\ldots,8 \ .$$

It is interesting to note that for each method the frequency of convergence to $\underline{x}_1^*$ is not less than that achieved when the variables were transformed, a result that might appear to be contrary to the findings or opinion of other researchers (e.g. Cutteridge and Dowson (60)). In this

d	Solution given by GN	Solution given by SDB
1.5	*	F
1.4	*	F
1.3	*	F
1.2	*	F
1.1	*	2
1.0	F	1
0.9	F	1
0.8	F	1
0.7	1	1
0.6	1	1
0.5	1	1
0.4	F	1
0.3	1	1
0.2	1	1
0.1	1	1
-0.1	1	1
-0.2	1	1
-0.3	1	1
-0.4	1	1
-0.5	1	1

d	Solution given by GN	Solution given by SDB
-0.6	1	1
-0.7	F	F
-0.8	F	1
-0.9	F	1
-1.0	F	1
-1.1	F	1
-1.2	F	1
-1.3	1	1
-1.4	F	1
-1.5	F	1
-1.6	F	2
-1.7	F	2
-1.8	F	2
-1.9	F	1
-2.0	F	2
-2.1	F	2
-2.2	F	2
-2.3	1	2
-2.4	F	2
-2.5	F	F

* = not attempted

F = failed

Solution 1 $\underline{x}_1^* = (0.9, 0.45, 1.0, 8.0, 8.0, 5.0, 1.0, 2.0)^\tau$

Solution 2 $\underline{x}_2^* \doteq (0.8985, 0.9740, 11.65, 3.251, 6.711, -8.764, 1.251, -0.5251)^\tau$

Table 6.1 Transistor model problem unconstrained

case the single known unrealisable solution did not prove to be a major distraction, as might have been the case if there had been more such solutions.

The use of alternative correction vectors and multiple starting points was examined on all those trials of Table 6.1 which either failed to produce a problem solution or converged to $\underline{x}_2^*$. The two methods are denoted by I and II respectively.

A flowchart of Method I is shown in Figure 6.2. Basically the method followed the normal SDB method except that at each iteration the

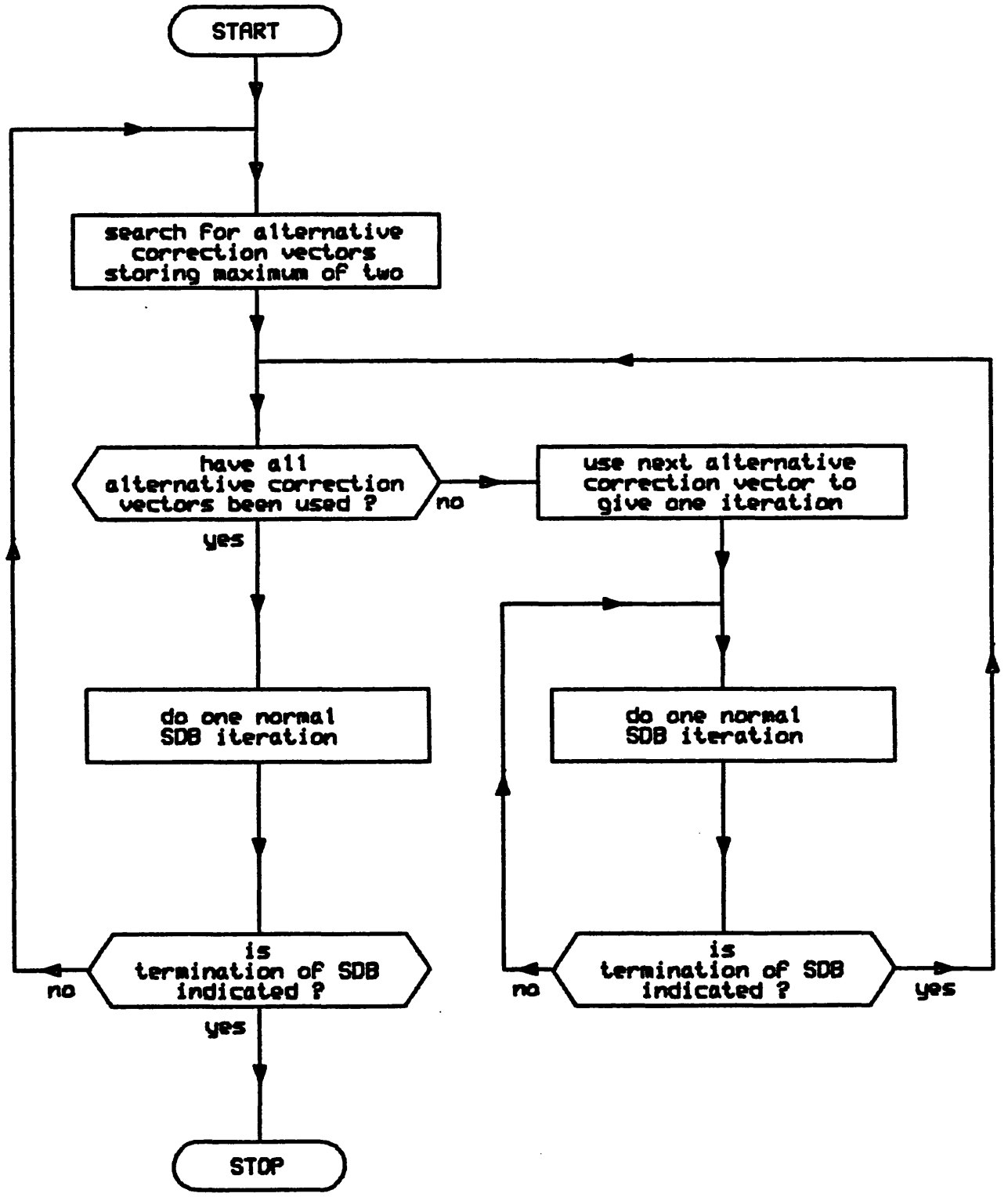


Figure 6.2 Flowchart of Method I

orthogonalisation was used to search for alternative correction vectors, storing a maximum of two. If any were found these were used in turn to define one modified iteration followed by use of the normal SDB method until termination was indicated. After the alternative correction vectors had been used, the original SDB was continued. The results for this method and for Method II are shown in Table 6.2. For Method I, Table 6.2 shows the number of alternative correction vectors found on the first five iterations and the result of pursuing the strategy described. In no case were any alternative correction vectors found after the third SDB iteration. On the six trial points from which normal SDB failed completely, only one yielded $\underline{x}_1^*$ when Method I was applied. On four of the six runs no alternative correction vectors were found. Of the nine trial points from which SDB converged to $\underline{x}_2^*$, three gave $\underline{x}_1^*$ when Method I was used.

Method II consisted of using the orthogonalisation procedure on the transistor model problem itself, but using $\underline{x}_2^*$ as the origin only if it had been located by SDB. For those initial displacements from which SDB failed to converge, implying that in a practical case $\underline{x}_2^*$ would not have been known, the orthogonalisation was generated from the point given by the initial displacement and $\underline{0}$ was used to define the first direction. Table 6.2 shows the number of starting points so generated and the result of using normal SDB from each. $\underline{x}_1^*$ was found in seven cases out of the nine where the orthogonalisation was started from $\underline{x}_2^*$ and in one case out of the six which required the arbitrary initial direction.

The orthogonalisation procedure in both Methods I and II used 10 function evaluations in each direction and the maximum absolute value of the variables, either δ_i or x_i $(i=1,2,\ldots,8)$ as the case may be, was 100.0. Smaller steps in the orthogonalisation, either by reducing the range or increasing the number of function evaluations, did not produce

displacement d	Problem solution given by normal SDB method	I Using alternative correction vectors (c.v.'s)		II Using multiple starting points (s.p.'s)	
		No. of alternative c.v.'s found on first five iterations	Problem solutions given by each alternative c.v. followed by SDB	No. of alternative s.p.'s generated by orthogonalisation	Problem solutions given by alternative s.p.'s .
1.5	F	0,0,0,0,0	-	8+	F,F,F,F,2,F,F,F
1.4	F	0,0,0,0,0	-	8+	F,F,F,F,2,F,F,F
1.3	F	0,0,0,0,0	-	6+	F,F,F,F,2,F
1.2	F	0,0,0,0,0	-	9	2,F,2,F,F,F,F,F,F
1.1	2	0,0,0,0,0	-	4	F,F,F,F
-0.7	F	0,1,0,0,0	F	4+	F,F,F,F
-1.6	2	1,2,1,0,0	1,1,2,2	2	F,F
-1.7	2	1,0,1,0,0	1,2	3	1,F,F
-1.8	2	1,2,1,0,0	1,1,2,2	2	1,F
-2.0	2	1,0,1,0,0	F,2	3	1,F,F
-2.1	2	1,0,1,0,0	F,2	3	1,F,F
-2.2	2	1,1,1,0,0	F,2,2	3	1,F,F
-2.3	2	1,1,1,0,0	F,2,2	3	1,F,F
-2.4	2	1,1,1,0,0	F,F,2	6	1,F,F,F,F,F
-2.5	F	1,0,0,0,0	1	7+	1,F,1,F,F,1,2

F = failed, + indicates that the run time limit was reached so more starting points might have been generated.

Problem solution 1 $\underline{x}_1^* = (0.9, 0.45, 1.0, 8.0, 8.0, 5.0, 1.0, 2.0)^\tau$

Problem solution 2 $\underline{x}_2^* = (0.8985, 0.9740, 11.65, 3.251, 6.711, -8.764, 1.251, -0.5251)^\tau$

Table 6.2 Transistor model results using alternative correction vectors and multiple starting points

more alternative correction vectors for Method I at points from which normal SDB failed. Because of the small number found, it is difficult to draw conclusions on the use of alternative correction vectors in this case. There does, however, seem to be a correlation between the performance of the $\underline{\delta}_o$ - method and the ease of finding alternatives. In Method II, use of the arbitrary initial orthogonalisation direction yielded more starting points than by using $\underline{x}_2^*$ as the origin, but the success rate on these was no better than that of Method I unless finding $\underline{x}_2^*$ is counted as success.

The suitability of the use of alternative correction vectors for finding multiple solutions of a problem is therefore open to doubt. In the cases where SDB found $\underline{x}_2^*$ similar numbers of starting points were generated by Methods I and II; but whereas those for Method I were more likely to converge to a solution, 15 out of 21 as opposed to 6 out of 23 , Method II produced the known alternative problem solution from a greater number of trial displacements d . It would appear that the application of only one alternative correction vector is insufficient to ensure that the path of the method is directed away from that given by pure $\underline{\delta}_o$ moves. When SDB failed to converge, Method I was inefficient at finding alternative correction vectors while Method II generated many unproductive starting points, the overall success rates being identical.

6.2. Use of alternative equations

Two variations on Equations (4.3) were considered. The first was intended to reduce the amount of computation in an iteration of an SD method while hopefully retaining improved convergence properties over GN . This amounted to a simplication of the equations to restore linearity with respect to the correction vector. The second variation was the inclusion of higher derivatives in Equations (4.3) with the

intention of increasing the solution convergence range for a given problem.

6.2.1. Linear correction vector

Section 4.1 mentioned that in the early stages of research GN was applied to Equations (4.1) and if successful a linear search to minimise $\phi(\lambda) = f(\underline{x} + \lambda\underline{\delta})$ was used to define the new point. This is equivalent to applying Method SDA to equations of the form:

$$0 = \lambda^2 s_i(\underline{x}) + \lambda \sum_{j=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j} \delta_j + \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^2 s_i(\underline{x})}{\partial x_j \partial x_k} \delta_j \delta_k$$

$$(i=1,2,\ldots,m). \quad (6.1)$$

This makes $\underline{\delta}$ a linear function of λ so the equations need be solved only once per iteration. In addition it was remarked earlier that solutions to Equations (4.1), and hence also (6.1), do not always exist. In the trials done subsequently, if a solution to Equations (6.1) was not forthcoming, an alternative method was used for that iteration. The two auxiliary methods considered were GN and SDB . The results obtained on the transistor model problem with positively constrained parameters and no correction limiting are given in Table 6.3, which shows, for those cases which converged to $\underline{x}_1^*$, the number of iterations based on Equations (6.1), the number of iterations requiring the auxiliary method and the total number of GN iterations required in the attempts to solve Equations (6.1).

Let the methods using Equations (6.1) be denoted by SD(GN) and SD(SDB) . Comparison with earlier trials for equivalent initial displacements reveals that SD(GN) had one more success than GN (cf. Table 3.7) and SD(SDB) had five more than GN but one less than SDB (cf. Table 4.10) . The success achieved by SD(SDB) from $d = 1.4$ had not been previously recorded by any method except with

Initial displacement d	Using GN as auxiliary method			Using SDB as auxiliary method			
	No. of iterations using Eqs. (6.1)	No. of GN iterations required on main problem	No. of GN iterations to solve Eqs. (6.1)	No. of iterations using Eqs. (6.1)	No. of SDB iterations required on main problem	No. of GN iterations to solve Eqs. (6.1)	No. of GN iterations to solve Eqs. (4.3)
2.0	*			failed			
1.8	*			failed			
1.6	*			failed			
1.4	*			6	2	53	22
1.2	*			6	1	32	8
1.0	failed			5	1	28	9
0.8	failed			5	1	41	10
0.6	4	1	54	5	1	56	19
0.4	5	0	54	5	0	54	0
0.2	3	0	9	3	0	9	0
-0.2	3	0	8	3	0	8	0
-0.4	3	0	12	3	0	12	0
-0.6	4	0	15	4	0	15	0
-0.8	failed			failed			
-1.0	failed			failed			

* not attempted

Table 6.3 Results using linear correction vector when possible

severe correction limiting imposed. An examination of the results revealed that in certain cases an excessive number of GN iterations had been used in unsuccessful attempts to solve Equations (6.1). Nevertheless, even taking this factor into account, it is interesting to note that in most cases more GN iterations were required to solve the subsidiary simultaneous equations in method SD(SDB) than in method SDB . Bearing in mind that Equations (6.1) had to be solved once per iteration whereas Equations (4.3) required solving several times per iteration, this result bears testimony to the difficulty of solving Equations (6.1) and the efficiency of the procedures developed for SDB . It is probable therefore that the aim of reducing computation time in the second derivative method was not realised; the operating system facilities did not permit easy verification of this conclusion.

6.2.2. Use of higher derivatives

An obvious extension to Equations (4.3) is to include the third derivative term or even higher derivative terms. Certainly the result of Section 4.2 regarding a reduction in the objective function for a suitable choice of λ is still valid provided that the higher derivatives are continuous in the neighbourhood of the current point. This means that the computation procedure described in Chapter IV is applicable to the revised equations.

Some preliminary trials using Equations (4.3) modified to include the third derivative term were conducted on the HDS and transistor model problems. It was expected that the third derivative method would require fewer iterations than the equivalent SD method when both converged to the same solution, and that the third derivative method would extend the range of solution convergence in the transistor model problem. However, the former result was not obtained consistently and the latter

result was not obtained at all. The reasons for this are not known; the possibility of a program error or inaccuracy cannot be excluded.

The use of higher derivatives can also be applied to the general function minimisation method. Equations (2.6), which form the basis of Cutteridge's descent method, can be extended to include a third derivative term thus:

$$\delta_i = -\mu\left\{\frac{\partial f}{\partial x_i} + \sum_{j=1}^{n} \delta_j \frac{\partial^2 f}{\partial x_j \partial x_i} + \frac{1}{2}\sum_{k=1}^{n}\sum_{j=1}^{n} \delta_j \delta_k \frac{\partial^3 f}{\partial x_k \partial x_j \partial x_i}\right\}$$

$$(i=1,2,\ldots,n). \qquad (6.2)$$

Since this is solved by $\underline{\delta} = \underline{0}$ at $\mu = 0$, the methods of Chapter IV are again applicable. Cutteridge et al. have noted considerable improvement over the steepest descent method by the inclusion of the second derivative term. The major difference between the Cutteridge approach and that of the methods of Chapter IV is whereas in the former approach all values of μ are considered, the latter approach is unable to cross the first discontinuity of the objective function. Nevertheless, there is sufficient evidence to suggest that Equations (6.2) merit investigation.

6.3. Discussion

Although the method of generating alternative solutions worked well enough on the four-cluster and modified Rosenbrock problems, much more difficulty was experienced on the transistor model problem with unconstrained variables, particularly on runs where SDB failed to give a solution. Even when SDB reached $\underline{x}_2^*$, the use of multiple starting points has much to commend it since it is the simpler method. These trials have therefore not produced any evidence that the use of alternative correction vectors is beneficial.

The replacement of Equations (4.3) by ones which yield a linear correction vector was expected to give a more efficient, if less robust method. That it did not was surprising but the conclusion drawn was that the procedures developed for the normal SD methods were reasonably efficient.

The few numerical trials that were done on equations which included higher derivatives were disappointing and suggest that further research into these is required. Lack of time prevented investigation into the use of higher derivatives in general function minimisation, but extrapolating from a comparison of second derivative methods with first derivative methods suggests that it would be worthwhile.

CHAPTER VII

RÉSUMÉ AND CONCLUSIONS

Chapter II presented a number of methods for local unconstrained optimisation which were categorised according to whether they were applicable to a general objective function or to one which could be expressed as a sum of squares. The research undertaken by the author has concentrated to a large extent on the use of second derivatives in algorithms for the sum of squares problem. The reasons for this were three-fold:

(i) several authors have suggested that the use of second derivatives in general function minimisation can be of benefit in many instances;

(ii) a good terminal performance is generally achieved by first derivative sum of squares algorithms when applied to certain problems in electrical circuit design;

(iii) there was a lack of second derivative sum of squares algorithms.

The first and third points were mentioned in Chapter II while the second point was illustrated in Chapter III, which reported on a series of trials conducted on a transistor model problem and also on a number of more simple problems. Each problem could be formulated as the optimisation of an objective function expressable as a sum of squares with zero residuals at the solution. The three main algorithms used were Gauss-Newton (GN) , Newton-Raphson (NR) and a quasi-Newton method due to Gill, Murray and Pitfield (GMP) . Several means of adjusting the full correction vectors given by the first two methods were also examined since, as was explained in Chapter II, this was a subject on which conflicting views were evident.

A comparison of similar implementations showed that in general, where convergence to the same solution was observed, GN required fewer iterations than NR , implying that the former was more efficient on this type of problem. GMP generally required the largest number of iterations but as the form of the algorithm is different from the other two and as computer run times were not obtainable to a suitable accuracy no conclusion could be drawn regarding the relative efficiency of GMP . However, on the transistor model problem, where convergence to the solution was obtainable only from points quite near the solution, it was observed that GN was better than all three forms of NR used and GMP was worse than two of the NR implementations. Furthermore, on the more difficult problems it was observed to be beneficial to seek univariate minimisation at each iteration rather than to adopt the other, less accurate correction scaling methods. It was concluded that reduced accuracy should be accepted with caution even though faster run times might be observed on simpler problems. The benefits of correction component limiting were also illustrated, but there are difficulties in selecting a suitable limit to apply.

Thus research was directed towards the development of a second derivative sum of squares method. Chapter IV described the investigation of several possibilities and gave the results of the most successful versions. The methods developed were all based on equations derived from the Taylor series in a similar way to the derivation of the basic equations of the GN method. The solution of the new equations, which is unobtainable analytically, was attempted by applying an auxiliary optimisation method. In the final versions of the second derivative (SD) algorithm, the optimisation method used was GN . Thus the SD method essentially consisted of the repeated application of GN to a number of sub-problems.

The results of Chapter IV demonstrated that the new SD methods required fewer iterations than GN when both converged to the same solution and that the SD methods extended the range of solution convergence on difficult problems. The former result is only of academic significance since the SD methods require far more calculations to be performed and so are unlikely to compete with a successful performance of their first derivative counterparts. However, the latter result could mean the difference between finding and not finding a solution. On the transistor model problem, it was observed that one SD method had a solution convergence range of between two and three times that of GN when no correction limiting was applied in either case. Hence it is envisaged that the new methods would only be used in the event of failure of the more simple methods.

In Chapter V two of the new methods were applied to problems in the design of electrical networks comprising resistors and capacitors only. The results demonstrated that the improved solution convergence range, which was again observed, could not necessarily be obtained by merely doing a few iterations of a general function minimisation method. In one of the two trial networks investigated the problem solution was reached using a three-part algorithm containing an SD method whereas it was not obtained by use of the original two-part conjugate gradient/GN method even though extra conjugate gradient iterations were calculated in an attempt to obtain convergence. Thus there is evidence that optimisation algorithms consisting of more than two methods may be worthwhile.

Chapter VI discussed some extensions to the SD algorithms using alternative correction vectors and alternative basic equations. Numerical trials showed that use of alternative correction vectors was unlikely to be of benefit because of difficulties encountered in their evaluation and the lack of success with those found. Results obtained by restarting the

basic algorithm from suitable points generated by an orthogonalised search process proved to be more successful as well as requiring less computation.

Among the alternative basic equations considered was one which produced a linear correction vector, thereby eliminating the need for applying the subsidiary optimisation algorithm more than once per iteration. However, it was found that on the transistor model problem the number of iterations of the subsidiary method was not reduced and the range of convergence was impaired. Thus use of these equations was abandoned.

Chapter VI also included a short discussion on the use of third derivatives in the basic equations, both for sum of squares minimisation and for general function minimisation. Unfortunately, lack of time prevented adequate investigation. This is considered to be an area for further research.

Thus the case for use of any one of the present SD algorithms rests with the improved convergence range it could give. It is possible that a general function method in a two-part algorithm could give a similar improvement in which case the preferred method would be the one generally requiring less computer time. The result of Chapter V has demonstrated that it cannot be assumed that the improvement will be obtained by a general function method.

A wider usage of the SD algorithms requires more computational efficiency within an iteration; there are two methods of achieving this. The first is to solve the basic equations of the SD methods more efficiently. An indication of the viability of thereby producing a method which would compete with the computational efficiency of a first derivative method for a given problem may be determined by an examination of the times taken to compute the problem and sub-problem objective functions. In the case of the transistor model problem such an

approach is thought to be impracticable.

The second method of achieving further computational efficiency is by the introduction of approximations. Such an approach is liable to adversely effect the convergence ranges of the algorithms but on certain problems this may not be important.

It is clear that the second derivative sum of squares algorithms suffer because no closed-form solution of their basic equations exists. This is not the case with second derivative general function minimisation algorithms and therefore the effectiveness of their use can be more easily assessed. As has been shown elsewhere, there is undoubtedly a great number of instances where the use of second derivatives in general function minimisation is beneficial.

Finally, one point needs further emphasis. The GN method used here could have been extended to continue from points of singularity which were not stationary thereby creating an algorithm to fulfil the conditions of Wolfe's convergence theorem. Although this was not required for the reasons stated in Section 3.2.2, such a modification might have extended the range throughout which convergence to a solution was obtained. However, similar modifications could be applied to the SD methods. It therefore remains an open question whether the observed difference in solution convergence ranges would still apply to the modified algorithms.

APPENDIX I

TO SHOW THAT A POSITIVE VALUE OF THE SEARCH PARAMETER IS NORMALLY SUITABLE IN THE GAUSS-NEWTON AND SECOND DERIVATIVE METHODS

For there to exist at the current point a value of $\lambda > 0$ which will reduce the objective function in the Gauss-Newton method or in the second derivative method we must have at that point:

$$\left[\frac{\partial}{\partial\lambda} f(\underline{x} + \underline{\delta}(\lambda))\right]_{\lambda=o} < 0 \quad .$$

We note that

$$\left[\frac{\partial}{\partial\lambda} f(\underline{x} + \underline{\delta}(\lambda))\right]_{\lambda=o} = \left[2 \sum_{j=1}^{n} \frac{\partial\delta_j}{\partial\lambda} \sum_{i=1}^{m} s_i(\underline{y}) \frac{\partial s_i}{\partial y_j}(\underline{y})\right]_{\lambda=o} \qquad (I.1)$$

where $\underline{y} = \underline{x} + \underline{\delta}(\lambda)$.

(i) Gauss-Newton (m=n): if the component functions are differentiable and the solution has not been reached, a suitable value of λ exists.

The Gauss-Newton equation for $m=n$ is

$$\lambda \underline{s}(\underline{x}) + J^T \underline{\delta} = \underline{0}$$

i.e. $$\lambda s_i(\underline{x}) + \sum_{j=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j} \delta_j = 0 \quad (i=1,2,\ldots,m) \ .$$

$$\therefore \quad s_i(\underline{x}) + \sum_{j=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j} \frac{\partial\delta_j}{\partial\lambda} = 0 \quad (i=1,2,\ldots,m) \ .$$

Ensuring that $\underline{\delta}(o) = \underline{0}$ and substituting for the derivatives in Equation (I.1):

$$\left[\frac{\partial}{\partial\lambda} f(\underline{x} + \underline{\delta}(\lambda))\right]_{\lambda=o} = -2 \sum_{i=1}^{m} \{s_i(\underline{x})\}^2 \leq 0$$

with equality at the solution. Hence result.

(ii) Gauss-Newton ($m \neq n$): if the component functions are differentiable and the gradient of the objective function is non-zero, a suitable value of λ exists.

From Equation (2.20):

$$\lambda \sum_{i=1}^{m} s_i(\underline{x}) \frac{\partial s_i(\underline{x})}{\partial x_k} + \sum_{j=1}^{n} \delta_j \sum_{i=1}^{m} \frac{\partial s_i(\underline{x})}{\partial x_j} \frac{\partial s_i(\underline{x})}{\partial x_k} = 0 \quad (k=1,2,\ldots,n) .$$

$$\therefore \quad \sum_{i=1}^{m} s_i(\underline{x}) \frac{\partial s_i}{\partial x_k} + \sum_{j=1}^{n} \frac{\partial \delta_j}{\partial \lambda} \sum_{i=1}^{m} \frac{\partial s_i(\underline{x})}{\partial x_j} \frac{\partial s_i(\underline{x})}{\partial x_k} = 0 \quad (k=1,2,\ldots,n) . \tag{I.2}$$

Ensuring that $\underline{\delta}(o) = \underline{0}$, making the appropriate substitution we have:

$$\left[\frac{\partial}{\partial \lambda} f(\underline{x} + \underline{\delta}(\lambda))\right]_{\lambda=o} = -2 \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \frac{\partial \delta_j}{\partial \lambda} \frac{\partial s_i(\underline{x})}{\partial x_j} \right)^2 \leq 0$$

with equality when $\sum_{i=1}^{m} s_i \frac{\partial s_i}{\partial x_k} = 0 \quad (k=1,2,\ldots,n)$.

Hence result.

(iii) Second derivative method ($m=n$): if the component functions are twice differentiable and a solution has not been reached, a suitable value of λ exists.

Equations (4.3) define the correction vector

$$\lambda s_i(\underline{x}) + \sum_{j=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j} \delta_j + \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^2 s_i(\underline{x})}{\partial x_j \partial x_k} \delta_j \delta_k = 0 \quad (i=1,2,\ldots,m).$$

$$\therefore \quad s_i(\underline{x}) + \sum_{j=1}^{n} \frac{\partial s_i(\underline{x})}{\partial x_j} \frac{\partial \delta_j}{\partial \lambda} + \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^2 s_i(\underline{x})}{\partial x_j \partial x_k} \frac{\partial \delta_j}{\partial \lambda} \delta_k = 0 \quad (i=1,2,\ldots,m).$$

Again ensuring that $\underline{\delta}(o) = \underline{0}$, we have

$$\left[\frac{\partial}{\partial\lambda} f(\underline{x} + \underline{\delta}(\lambda))\right]_{\lambda=o} = -2 \sum_{i=1}^{m} \{s_i(\underline{x})\}^2 \leqslant 0$$

with equality at the solution.

Hence result.

(iv) Second derivative method ($m \neq n$): if the component functions are twice differentiable and the gradient of the objective function is non-zero, a suitable value of λ exists.

Equations (4.4) define the correction vector:

$$\sum_{i=1}^{m} \left\{\left(\lambda s_i + \sum_{j=1}^{n} \frac{\partial s_i}{\partial x_j} \delta_j + \frac{1}{2} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^2 s_i}{\partial x_j \partial x_k} \delta_j \delta_k\right) \left(\frac{\partial s_i}{\partial x_r} + \sum_{j=1}^{n} \frac{\partial^2 s_i}{\partial x_j \partial x_r} \delta_j\right)\right\} = 0$$

$$(r=1,2,\ldots,n) .$$

At the point $\lambda = 0$, choosing $\underline{\delta} = \underline{0}$ we have

$$\sum_{i=1}^{m} \left\{\left(s_i + \sum_{j=1}^{n} \frac{\partial s_i}{\partial x_j} \frac{\partial \delta_j}{\partial \lambda}\right) \frac{\partial s_i}{\partial x_k}\right\} = 0 \qquad (k=1,2,\ldots,n)$$

which is equivalent to Equation (I.2). Hence result from (ii) above.

REFERENCES

1. Bard, Y. (1974): "Nonlinear parameter estimation". Academic Press.
2. Avriel, M., Rijckaert, M.J. and Wilde, D.J. (eds.) (1973): "Optimisation and design". Prentice Hall, New Jersey.
3. Cutteridge, O.P.D. (1973): "Electrical network and general system synthesis using optimisation techniques". Ref. (2), pp. 76-80.
4. Cutteridge, O.P.D. (1962): "Synthesis of linear n-terminal networks". Proc. I.E.E., Vol. 109, Part C, No. 16, pp. 640-645.
5. Cutteridge, O.P.D. (1973): "Computer synthesis of lumped linear networks of arbitrary structure". Ref. (73), pp. 105-111.
6. Cutteridge, O.P.D. and Di Mambro, P.H. (1974): "Some examples demonstrating feasibility of evolutionary approach to linear-network synthesis". Elect. Lett., Vol.10, No.3, pp.30-31.
7. Cutteridge, O.P.D. (1972): "Numerical experience with some two-part programmes for the solution of non-linear simultaneous equations". Leicester University Engineering Department Report No. 72-20.
8. Cutteridge, O.P.D. (1974): "Powerful 2-part program for solution of nonlinear simultaneous equations". Elect. Lett., Vol. 10, No. 10, pp. 182-184.
9. Barson, R.J., Bradly, D.P., Dimmer, P.R. and Young, A.G. (1977): "Engineering applications". Course notes for: "Interactive computer graphics for engineers", pp. H-1 to H-20. University of Leicester.
10. Kowalik, J. and Osborne, M.R. (1968): "Methods for unconstrained optimization problems". Elsevier, New York.
11. Powell, M.J.D. (1971): "Recent advances in unconstrained optimization". Math. Prog., Vol. 1, pp. 26-57.
12. Powell, M.J.D. (1976): "A view of unconstrained optimization". Ref. (74), pp. 117-152.

13. Murray, W. (ed.) (1972): "Numerical methods for unconstrained optimization". Academic Press.

14. Lill, S.A. (1976): "A survey of methods for minimising sums of squares of nonlinear functions". Ref. (74), pp.1-26.

15. Murray, W. (1972): "Fundamentals". Ref. (13), pp. 1-12.

16. Dixon, L.C.W., Gomulka, J. and Szegö, G.P. (1975): "Towards a global optimisation technique". Ref. (75), pp. 29-54.

17. Box, M.J. (1966): "A comparison of several current optimization methods and the use of transformations in constrained problems". Computer J., Vol. 9, pp. 67-77.

18. Lootsma, F.A. (1972): "A survey of methods for solving constrained minimization problems via unconstrained minimization". Ref. (49), pp. 313-348.

19. Dixon, L.C.W. (1977): "Global optima without convexity". Numerical Optimisation Centre Technical Report No. 85.

20. Gomulka, J. (1977): "A user's experience with Törn's clustering algorithm". Numerical Optimisation Centre Technical Report No. 90.

21. Nelder, J.A. and Mead, R. (1965): "A simplex method for function minimisation". Computer J., Vol. 7, pp. 308-313.

22. Dixon, L.C.W. (1975): "On quadratic termination and second order convergence - two properties of unconstrained optimisation algorithms". Ref. (75), pp. 211-228.

23. Murray, W. (1972): "Second derivative methods". Ref. (13), pp. 57-77.

24. Greenstadt, J. (1967): "On the relative efficiences of gradient methods". Math. Comp., Vol. 21, pp. 360-367.

25. Fiacco, A.V. and McCormick, G.P. (1968): "Nonlinear programming - sequential unconstrained minimization techniques". John Wiley and Sons Inc.

26. Cauchy, A. (1847): "Methode générale pour la résolution des systemes d'équations simultanées. Compte. rende. hebd. Seanc., Acad. Sci. Paris, Vol. 25, pp. 536-538.

27. Fletcher, R. and Reeves, C.M. (1964): "Function minimisation by conjugate gradients". Computer J., Vol. 7, pp. 149-154.

28. Miele, A. and Cantrell, J.W. (1969): "Study on a memory gradient method for the minimization of functions". J. Optim. Theory Appl., Vol. 3, No. 6, pp. 459-470.

29. Cragg, E.E. and Levy, A.V. (1969): "Study on a supermemory gradient method for the minimization of functions". J. Optim. Theory Appl., Vol. 4, No. 3, pp. 191-205.

30. Davidon, W.C. (1959): "Variable metric method for minimization". A.E.C. Research and Development Report No. AN-5990.

31. Fletcher, R. and Powell, M.J.D. (1963): "A rapidly convergent descent method for minimization". Computer J., Vol. 6, pp. 163-168.

32. Broyden, C.G. (1967): "Quasi-Newton methods and their application to function minimisation". Math. Comp., Vol. 21, pp. 368-381.

33. Dixon, L.C.W. (1972): "Quasi-Newton algorithms generate identical points". Math. Prog., Vol. 2, pp. 383-387.

34. Huang, H.Y. (1970): "Unified approach to quadratically convergent algorithms for function minimisation". J. Optim. Theory Appl., Vol. 5, pp. 405-423.

35. Wolfe, M.A. and Viazminsky, C. (1976): "Supermemory descent methods for unconstrained minimization". J. Optim. Theory Appl., Vol. 18, No. 4, pp. 455-468.

36. Fletcher, R. (1970): "A new approach to variable metric algorithms". Computer J., Vol. 13, pp. 317-322.

37. Gill, P.E. and Murray, W. (1972): "Quasi-Newton methods for unconstrained optimization". J.I.M.A., Vol. 9, pp.91-108.

38. Gill, P.E., Murray, W. and Pitfield, R.A. (1972): "The implementation of two revised quasi-Newton algorithms for unconstrained optimization". N.P.L. Report NAC 11.

39. Stewart III, G.W. (1967): "A modification of Davidon's minimization method to accept difference approximations of derivatives". J.A.C.M., Vol.14, pp. 72-83.

40. Fletcher, R. (1970): "Generalized inverses for nonlinear equations and optimization". Ref. (76), pp. 75-85.

41. Levenberg, K. (1944): "A method for the solution of certain non-linear problems in least squares". Quart. Appl. Math., Vol. 2, pp. 164-168.

42. Marquardt, D.W. (1963): "An algorithm for least squares estimation of non-linear parameters". S.I.A.M.J., Vol. 11, pp. 431-441.

43. Jones, A. (1970): "Spiral - a new algorithm for non-linear parameter estimation using least squares". Computer J., Vol. 13, pp. 301-308.

44. Broyden, C.G. (1965): "A class of methods for solving non-linear simultaneous equations". Math. Comp., Vol. 19, pp. 577-593.

45. Powell, M.J.D. (1970): "A hybrid method for nonlinear equations". Ref. (76), pp.87-114.

46. Powell, M.J.D. (1968): "A Fortran subroutine for solving systems of non-linear algebraic equations". A.E.R.E. Report No. 5947. Also, Ref. (76), pp. 115-161.

47. Brown, K.M. and Dennis Jr., J.E. (1972): "Derivative free analogues of the Levenberg-Marquardt and Gauss algorithms for nonlinear least squares approximation". Num. Math., Vol. 18, pp. 289-297.

48. Himmelblau, D.M. (1972): "A uniform evaluation of unconstrained optimization techniques". Ref. (49), pp. 69-97.

49. Lootsma, F.A. (ed.) (1972): "Numerical methods for non-linear optimization". Academic Press.

50. Verstege, P. and Wichmann, B.A. (1975): "An experimental data base for computer performance information". N.P.L. Report No. NAC 62.

51. Wright, D.J. and Cutteridge, O.P.D. (1976): "Applied optimisation and circuit design". C.A.D., Vol.8, pp.70-76.

52. Ramsay, J.O. (1970): "A family of gradient methods for optimization". Computer J., Vol. 13, pp. 413-417.

53. Butlin, G.A. (1976): "U.S.A. visits 1975-6". Leicester University Engineering Department Report No. 76-6.

54. Butlin, G.A. (1976): "Trends in hardware and software for interactive computer graphics". Leicester University Engineering Department Report No. 76-5.

55. Wozny, M.J. (1971): "On random search strategies for parameter optimisation". Proc. 9th Allerton Conference on Circuit and System Theory, pp. 390-396.

56. McKeown, J.J. (1975): "On algorithms for sums of squares problems". Ref. (75), pp. 229-257.

57. Betts, J.T. (1976): "Solving the nonlinear least square problem - application of a general method". J. Optim. Theory Appl., Vol. 18, No. 4, pp. 469-483.

58. Dixon, L.C.W. (1972): "The choice of step length, a crucial factor in the performance of variable metric algorithms". Ref. (49), pp. 149-170.

59. Sargent, R.W.H. and Sebastian, D.J. (1972): "Numerical experience with algorithms for unconstrained minimization". Ref. (49), pp. 45-68.

60. Cutteridge, O.P.D. and Dowson, M. (1973): "Methods for the solution of non-linear simultaneous equations incorporating some variations of Levenberg's technique". Leicester University Engineering Department Report No. 73-13.

61. Dowson, M. (1976): "Multimodal univariate search techniques and their application to optimisation problems". M. Phil. Thesis, University of Leicester.

62. Phillips, D.A. (1974): "A preliminary investigation of function optimisation by a combination of methods". Computer J., Vol. 17, No. 1, pp. 75-79.

63. Richards, G., Skwirzynski, J.K. (1969,1970): Personal correspondence to Dr. O.P.D. Cutteridge.

64. Ebers, J.J. and Moll, J.L. (1954): "Large-signal behaviour of junction transistors". Proc. I.R.E., pp. 1761-1772.

65. Numerical Algorithms Group (1975): NAG Algol Library Manual Mark 5.

66. Khinchin, A. (1960): "A course of mathematical analysis". Hindustan Publishing Corp.

67. Calahan, D.A. (1968): "Computer-aided network design". McGraw-Hill, New York.

68. Cutteridge, O.P.D. and Krzeczkowski, A.J. (1976): "An automated network design program". Leicester University Engineering Department Report No. 76-4.

69. Krzeczkowski, A.J. (1976): "Theory and design of lumped linear three-terminal RC networks". Ph.D. Thesis, University of Leicester.

70. Savage, W.H. (in preparation): "Automated synthesis of lumped linear three-terminal networks". Ph.D. Thesis, University of Leicester.

71. Birkhoff, G. and MacLane, S. (1965): "A survey of modern algebra". The Macmillan Company.

72. Brown, K.M. and Gearhart, W.B. (1971): "Deflation techniques for the calculation of further solutions of a non-linear system". Num. Math., Vol. 16, pp. 334-342.

73. Skwirzynski, J.K. and Scanlon, J.O. (eds.) (1973): "Network and signal theory". Peter Peregrinus.

74. Dixon, L.C.W. (ed.) (1976): "Optimization in action". Academic Press.

75. Dixon, L.C.W. and Szegö, G.P. (eds.) (1975): "Towards global optimisation". North Holland/American Elsevier.

76. Rabinowitz, P. (ed.) (1970): "Numerical methods for nonlinear algebraic equations". Gordon and Breach.

77. Wolfe, P. (1969): "Convergence conditions for ascent methods". S.I.A.M. Review, Vol. 11, No. 2, pp. 226-235.

78. Wolfe, P. (1971): "Convergence conditions for ascent methods. II: some corrections". S.I.A.M. Review, Vol.13, No. 2, pp. 185-188.

79. Dixon, L.C.W. (1974): "Nonlinear optimisation: a survey of the state of the art". Ref. (81), pp. 193-216.

80. Henderson, J.T. (1978): "Optimization techniques and their application". Ph.D. Thesis, University of Leicester.

81. Evans, D.J. (ed.) (1974): "Software for numerical mathematics". Academic Press.

ABSTRACT

THE USE OF SECOND DERIVATIVES IN APPLIED NUMERICAL OPTIMISATION

P. R. DIMMER

Local unconstrained numerical optimisation techniques are applied to a vast number of problems from various branches of engineering. The available methods may be divided into two classes: those that assume no special form for the objective function and those that require an objective function in the form of a sum of squares. While there exist a number of methods of the first class that use second derivatives, until now there has been a lack of second derivative methods of the second class.

Although methods of the first class can be applied to an objective function in the form of a sum of squares, it is generally recognised that if the sum is zero at the solution methods of the second class exhibit better terminal convergence. This is demonstrated here using several examples, including a transistor model problem where the objective function is defined to be the sum of the squares of the residuals of a set of highly non-linear simultaneous equations. Problems of this type are prevalent in methods for the design of electrical circuits. The main objective of this research was to determine whether the use of second derivatives could be of benefit in the solution of these problems.

A number of second derivative sum of squares optimisation algorithms were devised, investigated and assessed using the transistor model problem as a standard test case. The most successful methods were then incorporated into a program for the synthesis of three-terminal lumped linear networks comprising resistors and capacitors. The development of the algorithms and their performance on these and various other trials is described; based on the results obtained some conclusions are drawn regarding the areas where the new algorithms are likely to be of benefit.