THE EFFECT OF DATA STRUCTURES ON THE BEHAVIOUR OF CERTAIN INTERPOLATION METHODS FOR UNCONSTRAINED FUNCTION MINIMIZATION

by

M. JUDELMAN

A thesis submitted to the Faculty of Science of the University of the Witwatersrand in fulfillment of the requirements of the degree of Master of Science.

Johannesburg, 1975

DECLARATION OF CANDIDATE

I hereby declare that this thesis is my own work, and that the material forming the basis of this work has not been incorporated in any thesis submitted by me to any other university for degree purposes.

[illegible]

M. JUDELMAN

ACKNOWLEDGEMENTS

I wish to thank Professor D.H. Jacobson, Supervisor of this thesis, for his help and his constant interest and encouragement in the research project.

I would also like to express my deep gratitude to Harry Botsaris, Alan Wassyng and Colin Wright for their invaluable help in preparing this thesis.

TO

Hanna, Mom, Dad and Judy

CONTENTS

C H A P T E R 1

I N T R O D U C T I O N

CHAPTER 1 :

INTRODUCTION

In order to gain a better understanding of practical systems and processes it is necessary to describe them with the aid of mathematics. Quite often this modelling theory is very complex but even when it is relatively simple, it may contain parameters which change with time, or vary in a random manner. It is often necessary to predict the optimum operating conditions of a system such that some performance criterion is extremised and therefore optimization methods are used to explore the local region of operation in order to determine appropriate system-parameter adjustments. The performance criterion could be, in an industrial process, the cost of running the process, or, in mathematics, the squared difference between a specified function and an approximation to it. It is also frequently the case that restrictions are imposed on the permissible values that the parameters or independent variables may take. These restrictions, or constraints, vary according to the process and can be simple ones on the range of the variables or complicated functions of the variables.

The optimization problem, therefore, is to maximize or minimize a scalar quantity or function, called the objective function, subject to certain constraints. Linear functions subject to linear constraints give rise to what are termed linear programming problems, while non-linear

CHAPTER 1 :

INTRODUCTION

In order to gain a better understanding of practical systems and processes it is necessary to describe them with the aid of mathematics. Quite often this modelling theory is very complex but even when it is relatively simple, it may contain parameters which change with time, or vary in a random manner. It is often necessary to predict the optimum operating conditions of a system such that some performance criterion is extremised and therefore optimization methods are used to explore the local region of operation in order to determine appropriate system-parameter adjustments. The performance criterion could be, in an industrial process, the cost of running the process, or, in mathematics, the squared difference between a specified function and an approximation to it. It is also frequently the case that restrictions are imposed on the permissible values that the parameters or independent variables may take. These strictions, or constraints, vary according to the process and can be simple ones on the range of the variables or complicated functions of the variables.

The optimization problem, therefore, is to maximize or minimize a scalar quantity or function, called the objective function, subject to certain constraints. Linear functions subject to linear constraints give rise to what are termed linear programming problems, while non-linear functions subject to linear or non-linear constraints are termed non-linear programming problems. Although linear programming has no meaning without constraints, non-linear problems can be constrained or unconstrained, and it is this latter case with which this thesis deals.

Unconstrained non-linear programming methods are usually divided into direct search methods, which use function evaluations only, and gradient methods, which use additional information in the form of the first derivative vector and sometimes the second derivative, or Hessian, matrix. However, there are different ways of classifying these methods, and one of them is the following:
Nearly all optimization methods approximate the given function by a well known function which is easily analysed, but they differ in the way in which the approximation is done. Some methods, which we have called Interpolation Methods, fit the approximating well known function, or model, to calculated values of the objective function at certain points, while others assume a model, but do not use it directly; instead, they use a derivation of the model to get a difference equation which when solved leads to an estimate of the minimum of the objective function.

For many years the most popular model was the quadratic function, although in recent years Jacobson and Oksman [1] have suggested the homogeneous model and Davison and Wong [2] have suggested a model using L-functions. In Interpolation methods, whatever the model may be, there are a certain number of independent coefficients which have to be determined in order to fit the model to the given function. In this thesis we investigate the effect of the amount of data used to find the above mentioned coefficients. A number of existing interpolation methods are modified to facilitate the checking of the effect of different data structures on them; and a new model, based on a variation of the usual quadratic used, is developed. This model, called the Quadratic Gradient Model (Q.G.M.), is compared to other interpolation methods and to a few standard optimization techniques; and is also tested using a number of the data structures.

The investigation outlined above is presented in the following way: In Chapter 2 we formulate the unconstrained optimization problem and discuss the most widely used existing Interpolation and Non-interpolation methods. Chapter 3 presents the newly developed Q.G.M., while Chapter 4 presents the methods used when investigating the effect of the size of data structures, and explains the necessary modifications to certain existing Interpolation methods. Chapter 5 gives a proof of convergence for the Q.G.M. and in Chapter 6 we have the numerical results concerning the comparison of different methods and the influence of the different data structures on these.

In Chapter 7 a conclusion is reached as to the optimal amount of data which should be used in Interpolation methods and some ideas for further research are suggested.

C H A P T E R 2

F U N C T I O N M I N I M I Z A T I O N

CHAPTER 2 :

FUNCTION MINIMIZATION

The first section of this Chapter formulates the unconstrained optimization problem, the conditions for its solution and the classical method of solving it. The second section describes a selected number of well known optimization techniques in general and then relevant Interpolation methods are discussed in some detail in the last section.

2.1 The Problem

Let us first of all introduce some notation:

R^n	- The Euclidean space of ordered n-tuples of real numbers.
$f:A \rightarrow B$	- A mapping of the function f from its domain A to its range B.
$g(x)$	- Given a function $f:R^n \rightarrow R^1$, this is the so-called gradient of f at x, or the column vector of first partial derivatives of f.
$H(x)$	- Given a function $f:R^n \rightarrow R^1$, this is the $n \times n$ Hessian matrix of second partial derivatives of f.
$B(x,\epsilon(x))$	- The ball or neighbourhood of x defined as the set: $\{x' \| x-x' \| \leq \epsilon(x)\}$
e_i	- Vector whose i th component is one and whose other components are all zero.

Using the above notation we can now formulate the unconstrained optimization problem as follows:

$$\text{minimize } f(x),\ x \in R^n \qquad (2.1.1)$$

where the objective function $f:R^n \rightarrow R^1$ is a continuously differentiable function of x.

Although we have formulated the problem here as a minimization problem we note that a maximum problem can be solved by a minimization technique since

$$\text{maximum } f(x) = \text{minimum } [-f(x)] \qquad (2.1.2)$$

Therefore the words optimize and minimize may be regarded as synonomous for our purposes.

Before we state the necessary and sufficient conditions for solving the above problem, it is necessary to define different kinds of minima:

(i) The point x^* is called a local minimum of f if there is a region R containing x^* so that

$$f(x) \geq f(x^*), \quad x \in R \qquad (2.1.3)$$

(ii) If the point x^* in the region R is such that

$$f(x) > f(x^*), \quad x \neq x^*, \quad x \in R \qquad (2.1.4)$$

then x^* is a local proper minimum.

(iii) If x^* is a point such that

$$f(x) \geq f(x^*), \text{ for all } x \in R^n \qquad (2.1.5)$$

then it is a global minimizer or minimum.

We note that a particular function may have several local minima with one of them the global minimum. The distinction between local and global minima is not essential for our purposes because a global minimum of f on a set R may be a local minimum on a set S where R is a subset of S. However, the distinction is important when the results of optimization methods have to be interpreted because it is usually impossible to determine if a local minimum is also the global minimum unless all minima are found and evaluated. It is very important to stress that existing optimization techniques can only find local proper minima. Indeed, it has been shown numerically that most of the standard convergent numerical processes implemented on a computer converge to the same point from a given initial value x_o. Because of this we will always refer to the local proper minimum as the minimum without loss of generality.

The classical methods of calculus use the concept of critical point to solve problem (2.1.1). A point x^* is a critical point of $f:R^n \to R^1$ if $g(x) = 0$ and if $H(x^*)$ is defined. Clearly, if x^* is a local minimum and $g(x^*)$ exists, we have $g(x^*) = 0$ (2.1.6)
and (2.1.6) is a necessary condition for x^* to be the minimum. However, a critical point need not be a minimum. In order to ensure a minimum we need a theorem from calculus which states that if x^* is a critical point of f and $H(x^*)$ is positive definite, then x^* is a proper local minimum of f. This implies that satisfaction of the conditions

$$g(x^*) = 0, \ H(x^*) > 0 \qquad (2.1.7)$$

is sufficient for x^* to be a proper local minimum.

A straightforward method of solving (2.1.1) would be to solve the set of non-linear equations (2.1.6), which is a problem of considerable difficulty. Note that a solution of (2.1.6) only gives us a critical point which could be a minimum, maximum or saddle point; the number of critical points cannot be determined by inspection; and the method is not readily applicable to functions with discontinuous derivatives, although such functions frequently have well defined minima.

Modern optimization methods use iterative techniques (except for tabulation and random search methods) which require an initial point x_o to be specified and then proceed to generate a sequence of points x_i, $i = 1,2,3,\ldots$ which converges to the minimum. These iterative techniques can be conveniently represented by the equation

$$x_{i+1} = x_i - t_i\, p_i \qquad (2.1.8)$$

where p_i is an n dimensional direction vector, and t_i is the positive steplength or distance moved along it. Another feature which all current minimization techniques have in common is that of descent, i.e.

$$f(x_{i+1}) < f(x_i), \text{ if } \|g(x_i)\| > 0 \qquad (2.1.9)$$

This feature does not ensure convergence, but at least gives improved approximations to the solution. Most minimization methods have the properties (2.1.8) and (2.1.9), but differ as to the way in which t_i and p_i are chosen.

2.2 Existing Non-interpolation Methods

In order to simplify the picture as much as possible we divide these methods into groups and subgroups under rough headings and then describe each subgroup briefly.

The classification may be done as follows:

(i) Direct Search Methods :

- (a) Tabulation Methods
- (b) Sequential Methods
- (c) Linear Methods
- (d) Derivative Estimation Methods

(ii) Gradient Methods :

- (a) First Order Methods
- (b) Second Order Methods
- (c) Quasi-Newton Methods
- (d) Conjugate Gradient Methods

The methods of group (i) use values of the objective function only, although in some methods they are used to obtain a numerical approximation to the derivatives of the objective function.

In group (i)a it is assumed that the minimum x^* lies within the region R defined by

$$X_i \leq x_i \leq X_i + d_i, \; i = 1,2,\ldots,n \tag{2.2.1}$$

where X_i and d_i are known. The function is then evaluated at a certain number of points in R and the smallest function value is taken as the minimum. The points in R can be chosen as nodes of a grid, randomly, or using a multivariate Fibonacci Search - See Sugie [3]. It can be shown that the number of function evaluations for the Fibonacci Search is proportional to the product of the logarithms of the required interval reduction factors, whereas, in the other two techniques it is proportional to the product of the factors themselves. This makes the Fibonacci Search much more effective, but even so it is much worse than the methods in groups (i)b and (i)c.

Group (i)b methods probe the objective function by performing function evaluations at the vertices of some geometric configuration in the space of the independent variables. When a better point is found a new geometric configuration is formed around the new point, and so on. These methods include Evolutionary Operation, proposed by Box [4], and the well known Simplex method, proposed by Spendley, Hext and Himsworth [5].

The difference between the methods in group (i)c and the ones already mentioned is that this group is the only one which uses a set of direction vectors throughout the search. In general the methods which belong to this group and which adaptively change the set of direction vectors with each iteration are better than those which use the initial set throughout. The most popular methods in this group are: Hooke and Jeeves Method [6], which attempts to use the principal axis of the objective function as a search direction; Rosenbrock's method [7], which uses n mutually orthonormal direction vectors, one of which is in the direction of recent best progress; the method of Davies, Swann and Campey, described by Swann [8], which is based on Rosenbrock's method but differs from it in that a one dimensional linear search is made along each direction in turn; and Powell's method [9], which uses the concept of conjugate directions.

The methods of group (i)d are essentially gradient methods, and therefore could have been included in group (ii). However, since the gradient is not calculated analytically but estimated using function values only, we have included these methods in group (i). A typical such method is that of Stewart [10] who modifies Davidon's [11] algorithm by using difference approximations to estimate the gradient. The difference approximations can be of the form:

$$\frac{\Delta f(x)}{\Delta x_i} \approx \frac{f(x + h_i\ e_i) - f(x)}{h_i} \qquad (2.2.2)$$

where h_i is a suitable steplength;
or of the central difference form:

$$\frac{\Delta f(x)}{\Delta x_i} \approx \frac{f(x + h_i\, e_i) - f(x - h_i\, e_i)}{2h_i} \qquad (2.2.3)$$

From the performance point of view the methods of group (i)d and Powell's method of group (i)c are the best, while the tabulation methods of group (i)a are usually the least effective.

Since the above methods do not use derivatives, the check for descent is done by calculating the objective function value at the new point and seeing whether it is less than the function value at the current point. The gradient methods of group (ii), however, differ from those of group (i), not only because they use the gradient, but also in the way in which descent is ensured.

The condition used for descent is based on the Taylor expansion of a function:

(2.2.4)

$$f(x_{k+1}) = f(x_k) + g^T(x_k)\,\Delta x + \tfrac{1}{2}\Delta x\, H(\xi x_k + (1-\xi)x_{k+1})\Delta x$$

where: $\Delta x = x_{k+1} - x_k$, $0 < \xi < 1$

Using the equation (2.1.8) for an iterative process we have:

$$\Delta x = -t_k p_k \qquad (2.2.5)$$

and substituting this into (2.2.4) we get:

$$\Delta f = -t_k\, g^T(x_k)\, p_k + \tfrac{1}{2}t_k^{\,2}\, p_k^T\, H(x_k - t_k p_k(1-\xi))\, p_k \qquad (2.2.6)$$

where: $\Delta f = f(x_{k+1}) - f(x_k)$

We can easily choose t_k to make the first term in the expansion the dominant one and since for descent we want $\Delta f < 0$

this gives us:

$$-g^T(x_k)\ p_k < 0 \qquad (2.2.7)$$

Since gradient information is available in the methods of group (ii), criterion (2.2.7) is the one usually used in this group to ensure descent.

If we neglect the final term in equation (2.2.2), assuming that the second derivative is small enough to be neglected, the equation becomes

$$\Delta f = g^T\ \Delta x \qquad (2.2.8)$$

and methods based upon (2.2.8) are called first order methods - group (ii)a.

The fundamental first order method is the method of <u>steepest descent</u>, which is credited to Cauchy (1847). It is based on the fact that the direction of the gradient is the one which gives the greatest local change in the function. If, therefore, we choose a specified steplength t_i and make p_i equal to the gradient vector we can obtain a function decrease by applying equation (2.1.8). This procedure is then repeated until some stop criterion is satisfied. Other methods in this group do not use a fixed steplength, but minimize the function along the direction of the gradient. In both cases, however, these methods behave poorly because the gradient direction gives the greatest local decrease, but might not be at all in the direction of the minimum, and therefore a very large number of iterations can become necessary.

Methods which do use a search direction which coincides with the direction to the minimum are considered in group (ii)b. An approximate direction to the minimum can be found as follows:

If, in equation (2.2.4), we substitute:

$$x^{\star} \text{ (the minimum)} = x_{k+1}$$

$$H(x_k) \approx H(\xi \; x_k + (1-\xi)x_{k+1}) \tag{2.2.9}$$

and use the cond [illegible] that at the minimum $g(x^{\star}) = 0$ it can easily be shown [illegible]

$$g(x_k) + H(x_k)\Delta x = 0 \tag{2.2.10}$$

Solving for x we get:

$$\Delta x = -H^{-1}(x_k)g(x_k) \tag{2.2.11}$$

or

$$x^{\star} = x_k - H^{-1}(x_k)g(x_k) \tag{2.2.12}$$

The use of equation (2.2.9) actually approximates the general function by a quadratic and, therefore, if f is a quadratic the minimum can be reached from x_k in a single step. As a result of the above we see that if we choose

$$p_i = H^{-1}(x_k)g(x_k) \tag{2.2.13}$$

it will be in the direction of a stationary or critical point (assuming that the objective function is roughly quadratic). If we wish this direction to point toward a minimum we must ensure that $H(x_k)$ is positive definite. Equation (2.2.13) is the basis of all <u>second order</u> methods, or those using quadratic approximations.

If equation (2.2.13) is used directly with a fixed steplength t_i we have the fundamental Newton's method. This method is of little use practically for two main reasons:

1. H is not always positive definite.
2. At each step H must be calculated and inverted.

One way, which is used in this group, to overcome difficulty 1) is suggested by Greenstadt [12]. This is to find the eigenvalues (λ) and normalized eigenvectors (U) of H_k and then set

$$H_k = \sum_{i=1}^{n} |\lambda_i| U_i U_i^T \qquad (2.2.14)$$

Another way, used by Goldfield [13], is to replace H_k by

$$A_k = H_k + \lambda_k I \qquad (2.2.15)$$

and use λ_k to ensure $A_k > 0$.

The group of methods (group (ii)c) which try to overcome problem 2) are called Quasi-Newton methods because they approximate $H^{-1}(x_k)$ by another matrix, but still use the basic Newton direction. If $H^{-1}(x_k)$ is approximated by, say $E(x_k)$, we arrive at the basic equation of most Quasi-Newton or variable metric methods as :

$$x_{k+1} = x_k - t_k\, E(x_k) g(x_k) \qquad (2.2.16)$$

$E(x_{k+1})$ is usually calculated iteratively from $E(x_k)$ and it can be shown (see Himmelblau [14]) that the general form of this is :

$$E(x_{k+1}) = E(x_k) + \frac{\Delta x\, y^T}{y^T\, \Delta g} - \frac{E(x_k)\Delta g \quad z^T}{z^T\, \Delta g} \qquad (2.2.17)$$

where: $\Delta x = x_{k+1} - x_k$ and $\Delta g^T = g^T(x_{k+1}) - g^T(x_k)$

If we choose

$$y = z = \Delta x - E(x_k)\Delta g \qquad (2.2.18)$$

we get Broyden's method [15] , whereas if we choose

$$y = \Delta x,\ z = E(x_k)\Delta g \qquad (2.2.19)$$

Davidon's method [11] , as modified by Fletcher and Powell [16] , results. The latter method is one of the most popular optimization methods and is often used as a standard against which other algorithms are compared. This method also uses conjugate directions and can therefore also be included in group (ii)d.

The methods of group (ii)d and Powell's direct search method [9] use the concept of conjugate directions. This concept is defined as follows: A set of n vectors d_1, d_2, ..., d_n in R^n is said to be conjugate with respect to a positive definite matrix Q if

$$d_i^T\, Q\, d_j = 0,\ i \neq j \qquad (2.2.20)$$

The results of using this concept are:

1) The d_i vectors are linearly independent and can therefore be used as a basis for n dimensional space.
2) If f is a quadratic and is minimized along n Q-conjugate directions, then the minimum will be reached in at most n steps.

The best known methods in this group are Fletcher-Reeves' method [17] , Zoutendijk's method [18] and the Partan methods [14] . Zoutendijk's method has the disadvantage of requiring a matrix inversion, but the other two mentioned also have the disadvantage of requiring a univariate minimization along the search direction at each step.

It is difficult to compare different optimization techniques using different strategies, but on the whole the gradient methods are usually better than the direct search methods, and of the former the best are the Quasi-Newton and conjugate gradient methods.

method [17] , Zoutendijk's method [18] and the Partan methods [14] . Zoutendijk's method has the disadvantage of requiring a matrix inversion, but the other two mentioned also have the disadvantage of requiring a univariate minimization along the search direction at each step.

It is difficult to compare different optimization techniques using different strategies, but on the whole the gradient methods are usually better than the direct search methods, and of the former the best are the Quasi-Newton and conjugate gradient methods.

2.3 Interpolation Methods

Since this thesis deals with these methods, and because, in Chapter 4, we modify some of these, we will give a full description of the well known methods of this type (for the equations of some Interpolation methods see Appendix B).

The common characteristic of all Interpolation methods is that they fit a model to the given objective function at certain points, but they differ in the model used and even if the same model is used the method of minimization might be entirely different. Fiacco and Mc Cormick [19] and Winfield [20] both use a quadratic model, for example, but their methods are not at all alike. The quadratic model they use is

$$f(x) = \tfrac{1}{2}x^T A x + b^T x + d \qquad (2.3.1)$$

where A is an n x n symmetric matrix, b is an n vector and d is a scalar. A has $\frac{1}{2}n(n+1)$ independent elements, b has n components and d has one, which gives us a total of $\frac{1}{2}(n+1)(n+2)$ independent coefficients which have to be determined.

Fiacco and Mc Cormick [19] minimize the given objective function along each of the coordinate vectors in turn and use these points to obtain the diagonal elements of A. The remaining $\frac{1}{2}n(n-1)$ searches are made along vectors which have two components equal to one and the remaining components equal to zero. The least values of the objective function along these directions are used to find the off-diagonal elements of A. Once A has been found, the search direction

$$p_i = A^{-1} g(x) \qquad (2.3.2)$$

is used to locate the minimum of f(x), which is an estimate of the minimum of the given objective function. This procedure is possible because g(x) can be calculated using the values of A. However, this method is useful only when A is positive definite because if it is not, the located point will not be a minimum. In the event, the authors do not have any suggestions for this situation.

Winfield [20], on the other hand, chooses an initial grid of $N = \frac{1}{2}(n+1)(n+2)$ points and calculates the components of A, b and d by solving N simultaneous equations with N unknowns. The point, in the grid, which has the lowest function value is defined as the <u>basepoint</u> x_b and the coordinates of the other points are defined relative to this point. The quadratic model then becomes

$$q(y) = \tfrac{1}{2}y^T A\, y + b^T y + d \qquad (2.3.3)$$

where $y = x - x_b$.
Once A, b and d have been found, the following constrained problem is solved:

$$\min q(y), \text{ subject to } y^T y - r^2 = 0 \qquad (2.3.4)$$

where r is the radius of a sphere defining a region of validity R and is taken as $0.99\,\|y_n\|$, where y_n is the furthest distance from the basepoint. If the solution, y^*, to equation (2.3.4) gives a better function value than that of x_b, then y^* is taken as the new basepoint and the whole procedure is repeated. If not, x_b is retained as basepoint, the volume of R is reduced by a constant factor and equation (2.3.4) is solved again. In the first case, when y^* is chosen as the new basepoint, the N nearest points to it are included in the grid, while in the second case, the choice of $r = 0.99\|y_n\|$ ensures that y_n will not be included in the new grid and that y^* will. This means that, at each iteration, a point leaves the grid and a new one enters it. The points leaving the grid are stored in a data table and are sometimes re-used by the algorithm if the search moves past them again. Winfield reports that the best size for the data table is a little less than 2N.

From the above, we can see that this method uses the function evaluations efficiently, because, as successfull trial points are located, the radius of R increases and the search takes larger steps, while when unsuccessfull points are located, the radius of R decreases until a new direction is found. The volume reduction factor controls the tendency of the method to explore, versus the tendency to make small sure gains based on experience and Winfield has chosen the factor $(\frac{3}{4}).2^n$ by experimenting with a set of test problems. Other advantages of this method are that no derivatives are required and that A need not be positive definite. However, the main disadvantage of the algorithm is the large amount of computational work required to locate a new trial point. Since it is of the order of n^6 the method is best used for low dimensional functions which are expensive to evaluate.

Although the quadratic is the most popular model, recently other models have been suggested. The homogeneous model, developed by Jacobson and Oksman [1], is based on a derivation of a different form of the quadratic used in equation (2.3.1). The different form of the quadratic is

$$f(x) = \tfrac{1}{2}(x-\beta)^T Q(x-\beta) + \bar{\omega} \tag{2.3.5}$$

where Q is an n x n positive definite symmetric matrix, β the location of the minimum of f(x), and $\bar{\omega}$ the minimum value. The derivative of this form is

$$g(x) = Q(x-\beta) \tag{2.3.6}$$

and if we substitute (2.3.6) into (2.3.5) we get

$$f(x) = \tfrac{1}{2}(x-\beta)^T g(x) + \bar{\omega} \tag{2.3.7}$$

If equation (2.3.7) is given a more generalised form by changing the coefficient ½ to $^1/_\gamma$, we arrive at the homogeneous model

$$f(x) = \frac{1}{\gamma}(x-\beta)^T g(x) + \bar{\omega} \tag{2.3.8}$$

where γ is called the degree of homogeneity.

When a homogeneous model is fitted to the given objective function, there are n+2 coefficients which have to be determined: Multiplying (2.3.8) by γ, defining $\omega = \gamma\bar{\omega}$ and rearranging terms we have

$$\beta^T g(x) + \gamma\ f(x) + \omega = x^T g(x) \tag{2.3.9}$$

Furthermore, defining $v \triangleq x^T g(x)$ (2.3.10) are expensive to evaluate.

Although the quadratic is the most popular model, recently other models have been suggested. The homogeneous model, developed by Jacobson and Oksman [1], is based on a derivation of a different form of the quadratic used in equation (2.3.1). The different form of the quadratic is

$$f(x) = \tfrac{1}{2}(x-\beta)^T Q(x-\beta) + \bar{\omega} \qquad (2.3.5)$$

where Q is an n x n positive definite symmetric matrix, β the location of the minimum of f(x), and $\bar{\omega}$ the minimum value. The derivative of this form is

$$g(x) = Q(x-\beta) \qquad (2.3.6)$$

and if we substitute (2.3.6) into (2.3.5) we get

$$f(x) = \tfrac{1}{2}(x-\beta)^T g(x) + \bar{\omega} \qquad (2.3.7)$$

If equation (2.3.7) is given a more generalised form by changing the coefficient $\tfrac{1}{2}$ to $1/\gamma$, we arrive at the homogeneous model

$$f(x) = \frac{1}{\gamma}(x-\beta)^T g(x) + \bar{\omega} \qquad (2.3.8)$$

where γ is called the degree of homogeneity.

When a homogeneous model is fitted to the given objective function, there are n+2 coefficients which have to be determined: Multiplying (2.3.8) by γ, defining $\omega = \gamma\bar{\omega}$ and rearranging terms we have

$$\beta^T g(x) + \gamma\, f(x) + \omega = x^T g(x) \qquad (2.3.9)$$

Furthermore, defining $v \overset{\Delta}{=} x^T g(x)$ (2.3.10)

$$y \triangleq \begin{bmatrix} g(x) \\ f(x) \\ -1 \end{bmatrix} \qquad (2.3.11)$$

$$\alpha \triangleq \begin{bmatrix} \beta \\ \gamma \\ \omega \end{bmatrix} \qquad (2.3.12)$$

equation (2.3.9) can be written in matrix form as

$$y^T \alpha = v \qquad (2.3.13)$$

The components of α are β, which is an n vector, and γ and ω, which are scalars; thus making α an n+2 vector. In order to find the n+2 coefficients the appropriate number of points x_i, i = 1, ..., n+2 and the associated values $f(x_i)$, $g(x_i)$, i = 1, ..., n+2 are needed. Then, the following relation holds:

$$Y\alpha = V \qquad (2.3.14)$$

where

$$Y \triangleq \begin{bmatrix} y_1^T \\ \vdots \\ y_{n+2}^T \end{bmatrix}, V \triangleq \begin{bmatrix} v_1 \\ \vdots \\ v_{n+2} \end{bmatrix} \qquad (2.3.15)$$

and

$$y_i = \begin{bmatrix} g(x_i) \\ f(x_i) \\ -1 \end{bmatrix}, v_i = x_i^T g(x_i), \ i = 1, \ldots, n+2 \qquad (2.3.16)$$

If the objective function is homogeneous, the solution of equation (2.3.14) is

$$\alpha = Y^{-1} V \qquad (2.3.17)$$

On the other hand, for general functions, a only provides an approximation to the minimum. Therefore, if a supplies a better function value than the current trial point, it is included in the grid of n+2 points and the procedure is repeated. If not, cubic interpolation (presented by Fletcher and Powell [16]) is used to achieve descent. In order to present a proof of convergence, Jacobson and Pels [21] modified the original algorithm to use Armijo's Rule [22] instead of cubic interpolation, with results nearly as good as those of the original algorithm.

Even though Jacobson and Oksman's algorithm uses a grid of n+2 points, it does not solve the n+2 simultaneous equations at each iteration, but inverts Y recursively using Householder's formula. This is possible for the reason that only one row of Y is changed at each iteration, and it avoids the large amount of computational work that would have been necessary to invert Y at each iteration. Winfield [20] also changes one row only of a matrix at each iteration, but solves a constrained problem instead of simply inverting the matrix, and therefore cannot do this recursively. Householder's recursive formula saves much computational work, but is sometimes unstable, and in order to overcome this problem, another modification to the original algorithm was suggested by Kowalik and Ramakrishnan [23]. In this modification Householder's formula is replaced by a semi-triangular factorization which is numerically stable provided that a pivoting strategy is used in the process of updating these factors.

Their results are an improvement on the original algorithm, in that their method requires less function evaluations, is numerically more stable and also has the advantage of implementing special storage schemes for large-scale problems.

In order to differentiate between the two ways in which the systems of equations are solved, we have divided Interpolation Methods into two forms:

(i) Grid-to-grid methods, e.g. Winfield [20], which solve the full set of N x N equations at each iteration.

(ii) Point-to-point methods, e.g. Jacobson and Oksman [1], which solve the equations by inverting a matrix recursively.

Although Winfield's method can only be used in grid-to-grid form, as mentioned above; most other Interpolation Methods can be used in either form.

A different way of using Interpolated models has been suggested by Botsaris [24]. Instead of using an Interpolation model to approximate the Hessian matrix and minimizing directly using equation (2.3.2), as Fiacco and McCormick [19] do, Botsaris uses an Interpolation model to approximate the Hessian matrix but then uses this approximation in his Differential Descent methods. It is interesting to note that numerical results show that for this purpose the Interpolated Model is more stable than a difference model which was also tested.

In general, Interpolation methods compare favourably to Non-interpolation methods. They use a lesser, or in some cases the same, number of function evaluations for most test functions than the best Non-interpolation methods, such as that of Fletcher and Powell [16], and tend to require less stringent restrictions.

C H A P T E R 3

QUADRATIC GRADIENT MODEL (Q.G.M.)

CHAPTER 3 :

QUADRATIC GRADIENT MODEL (Q.G.M.)

In this chapter we present the Q.G.M. method for solving unconstrained optimization problems. The method is based on a quadratic function and uses an Interpolation model to approximate the inverse of the Hessian matrix. The algorithm is presented in a point-to-point conceptual form, using a method for ensuring descent, which facilitates the proof for convergence, but might not necessarily give the best practical results.

3.1 Basis for Model

If we take equation (2.3.7), which is

$$f(x) = \tfrac{1}{2}(x-\beta)^T g(x) + \omega \qquad (3.1.1)$$

and find an expression for $(x-\beta)^T$ from equation (2.3.6), which is

$$g(x) = Q(x-\beta) \qquad (3.1.2)$$

we arrive at the Q.G.M.

From (3.1.2)

$$x-\beta = Q^{-1} g(x) \qquad (3.1.3)$$

and its transpose will be

$$(x-\beta)^T = g^T(x)(Q^{-1})^T \qquad (3.1.4)$$

Since Q is the Hessian matrix of f(x) it is symmetric. This means that

$$(x-\beta)^T = g^T(x)\ Q^{-1} \qquad (3.1.5)$$

Substituting this into equation (3.1.1) we get

$$f(x) = \tfrac{1}{2}g^T(x)\ Q^{-1}\ g(x) + \omega \qquad (3.1.6)$$

The actual Q.G.M. is defined using equation (3.1.6) as a basis and is of the form

$$f(x) = \tfrac{1}{2}g^T(x)\ S\ g(x) + \omega \qquad (3.1.7)$$

where S is an n x n positive definite symmetric matrix. If the objective function is a quadratic or can be written in the form of equation (3.1.7), then the minimum β can be found directly from equation (3.1.3) simply by rearranging terms and substituting S for Q^{-1} :

$$\beta = x - S\ g(x) \qquad (3.1.8)$$

where x can be any initial trial point.

However, in the case of more general functions, the location of the minimum is not given by equation (3.1.8) since our model is only an approximation to the actual function. Therefore, at each trial point, using the function value and first derivative at that point, the Q.G.M. is formed and minimized directly in n space by equation (3.1.8) to yield a search direction. This equation may be used if S is singular, but we prefer restarting the algorithm with a gradient step. Although the point-to-point form presented here gives better results, the grid-to-grid form is better for testing the influence of data structures on the method.

Equation (3.1.7) is solved for S and ω in the following way :

Let

$$S \triangleq (s_{ij})$$

$$a \triangleq \begin{bmatrix} s_{11} \\ \vdots \\ s_{nn} \\ s_{12} \\ \vdots \\ s_{1n} \\ s_{23} \\ \vdots \\ s_{2n} \\ \vdots \\ s_{n-1,n} \\ \omega \end{bmatrix}, \quad y(x) \triangleq \begin{bmatrix} \frac{1}{2}g_1^{\,2}(x) \\ \vdots \\ \frac{1}{2}g_n^{\,2}(x) \\ g_1(x)g_2(x) \\ \vdots \\ g_1(x)g_n(x) \\ g_2(x)g_3(x) \\ \vdots \\ g_2(x)g_n(x) \\ \vdots \\ g_{n-1}(x)g_n(x) \\ 1 \end{bmatrix}$$

Then equation (3.1.7) can be written in the following way:

$$f(x) = y^T(x)\ \alpha \tag{3.1.10}$$

Note that both $y(x)$ and α are $N \times 1$ vectors, where $N = \frac{n}{2}(n+1)+1$. If $f(x)$ and $y(x)$ are evaluated at N distinct points x_i, $i = 1, \ldots, N$ so that the $y(x_i)$'s are linearly independent then the system of equations can be written in matrix form as :

$$F = Y\ \alpha \tag{3.1.11}$$

where

$$F = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}, \quad Y = \begin{bmatrix} y^T(x_1) \\ \vdots \\ y^T(x_n) \end{bmatrix} \tag{3.1.12}$$

Since the $y(x_i)$'s are linearly independent the sclution of (3.1.11) is

$$\alpha = Y^{-1}\ F \tag{3.1.13}$$

where S can be constructed easily from the first N-1 components of α and the Nth component of α is the minimum function value ω. In order to save computational work, the inversion of Y is carried out recursively by defining

$$Y_o = I,\ \alpha_o^T = [\underbrace{1, \ldots, 1}_{n \text{ times}}, 0, \ldots, 0]$$

and then replacing, for each new trial point, corresponding rows and elements of Y_k and F_k with the values of $y(x_k)$ and $f(x_k)$ in the following way:

$$Y_{k+1} = Y_k + e_{k+1} (y^T(x_{k+1}) - e_{k+1}^T Y_k) \qquad (3.1.14)$$

$$F_{k+1} = F_k + e_{k+1} (f(x_{k+1}) - e_{k+1}^T F_k) \qquad (3.1.15)$$

In order to obtain the inverse of Y recursively, we use Sherman-Morrison's formula, which states that if $A \epsilon R^{n \times n}$ is invertible and $u,v \epsilon R^n$, then

$$(A + u v^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1+v^T A^{-1} u} \qquad (3.1.16)$$

provided that $1 + v^T A^{-1} u \neq 0$

Substituting : $Y_k = A$

$u = e_{k+1}$

$v^T = y^T(x_{k+1}) - e_{k+1}^T Y_k$

into equation (3.1.16) we arrive at the necessary updating formula

$$Y_{k+1}^{-1} = Y_k^{-1} - \frac{Y_{k+1}^{-1} e_{k+1} (y^T(x_{k+1}) Y_k^{-1} - e_{k+1}^T)}{y^T(x_{k+1}) Y_k^{-1} e_{k+1}} \qquad (3.1.17)$$

and $y^T(x_{k+1}) Y_k^{-1} e_{k+1} \neq 0$ (3.1.18)

From equations (3.1.15) and (3.1.17) and from the fact that

$\alpha_{k+1} = Y_{k+1}^{-1} F_{k+1}$

we obtain the recursive solution

$$\alpha_{k+1} = \alpha_k + \frac{Y_k^{-1} e_{k+1} (f(x_{k+1}) - y^T(x_{k+1}) \alpha_k)}{y^T(x_{k+1}) Y_k^{-1} e_{k+1}} \qquad (3.1.19)$$

provided, again, that (3.1.18) holds.

3.2 The Algorithm

Step 1. Assume X_o, η_1, η_2, L given

2. Set $i = 0$, $j = 1$, $N = \frac{n}{2}(n+1) + 1$, $Y_o = I$,

$$a_o^T = [\underbrace{1, \ldots\ldots, 1}_{n \text{ times}}, 0, \ldots\ldots 0]$$

3. If $||g(x_i)|| = 0$ stop; else go to 4.

4. If largest element of a_k is greater than L, set $p_i = -g(x_i)$ and use Armijo's Subprocedure to calculate x_{i+1}, set $x_o = x_{i+1}$ and go to 1; else go to 5.

5. Construct S_i from the elements of a_i

6. Set $p_i = \sigma S_i\ g(x_i)$, where $|\sigma| = 1$ and its sign is chosen so that

$p_i^T\ g(x_i) < 0$

7. If $|p_i^T\ g(x_i)| \geq \eta_1$, use Armijo's Subprocedure to calculate x_{i+1} and go to 8; else set $x_o = x_i$ and go to 1.

8. Calculate $y(x_{i+1})$ and $f(x_{i+1})$ as defined in equation (3.1.9).

9. If $|y^T(x_{i+1})\ Y_i^{-1}\ e_j| < \eta_2$ set $x_o = x_{i+1}$ and go to 1; else calculate Y_{i+1}^{-1} and a_{i+1} from:

$$Y_{i+1}^{-1} = Y_i^{-1} - \frac{Y_i^{-1}\ e_j\ (y^T(x_{i+1})\ Y_i^{-1} - e_j^T)}{y^T(x_{i+1})\ Y_i^{-1}\ e_j}$$

$$\alpha_{i+1} = \alpha_i + \frac{Y_i^{-1} e_j \left(f(x_{i+1}) - y^T(x_{i+1})\, \alpha_i\right)}{y^T(x_{i+1})\, Y_i^{-1} e_j}$$

and go to 12.

10. Set $i = i + 1$
11. If $j = N$, reset $j = 1$ and go to 3; else set $j = j + 1$ and go to 3.

<u>Armijo Subprocedure</u>

1. Assume α, $\sigma \in (0,1)$ are given.
2. Set $k = 0$
3. Set $x_{i+1} = x_i + \sigma^k p_i$
4. If $f(x_{i+1}) - f(x_i) + \alpha\, \sigma^k p_i^T g(x_i) \leq 0$ return; else set $k = k + 1$ and go to 1.

Certain operations in the algorithm are now more fully described.

In order to choose a search direction which is one of descent, step 6 calculates a direction with the help of equation (2.2.7), which is

$$p_i^T g(x_i) < 0$$

In other words, if S is negative definite, the model will have a maximum, not a minimum, and, as a result, a direction opposite to that of the maximum is chosen. However, if $|p_i^T g(x_i)|$ is very small, the direction p_i will most likely not yield a respectable reduction in f(x) because it is very nearly orthogonal to the gradient. In this case, then, the algorithm is restarted by step 7.

To guard against the possibility that equation (3.1.18) does not hold, the expression

$$s = |y^T(x_{i+1})\ Y_i^{-1}\ e_j|$$

is checked at each iteration and if it is less than a prescribed small number the algorithm is restarted by step 9. Also, as explained before, the algorithm will not generally converge in N steps for general functions and, therefore, at the Nth step, when the last row of Y_o has been replaced by $y^T(x_N)$, the index j must be reset to unity, as in step 11, s[illegible] t the replacement of rows starts over again from th[illegible] irst row.

Armijo's Subprocedure, based on Armijo's Rule [22], guarantees, not only a function decrease, but also convergence. Although the original proof presented by Armijo is based on a Steepest Descent algorithm, it can easily be modified for more general algorithms. Other methods for ensuring descent or convergence are those of Curry [25], who requires the minimization of a function of one variable at each step; Goldstein [26], who requires the assumptions that $f(x) \in C^2$ on $S(x_o) = \{x : f(x) \leq f(x_o)\}$, that $S(x_o)$ be bounded and that a bound for the norm of the Hessian matrix is known; and Fletcher and Powell [16], who use cubic interpolation. The hypotheses of Armijo's convergence theorem are more restrictive than those imposed by Curry, but less restrictive than those imposed by Goldstein, and, therefore, provide a Rule which is both practical case, then,the algorithm is restarted by step 7.

To guard against the possibility that equation (3.1.18) does not hold, the expression

$$s = |y^T(x_{i+1})\ Y_i^{-1}\ e_j|$$

is checked at each iteration and if it is less than a prescribed small number the algorithm is restarted by step 9. Also, as explained before, the algorithm will not generally converge in N steps for general functions and, therefore, at the Nth step, when the last row of Y_o has been replaced by $y^T(x_N)$, the index j must be reset to unity, as in step 11, so that the replacement of rows starts over again from the first row.

Armijo's Subprocedure, based on Armijo's Rule [22], guarantees, not only a function decrease, but also convergence. Although the original proof presented by Armijo is based on a Steepest Descent algorithm, it can easily be modified for more general algorithms. Other methods for ensuring descent or convergence are those of Curry [25], who requires the minimization of a function of one variable at each step; Goldstein [26], who requires the assumptions that $f(x) \in C^2$ on $S(x_o) = \{x : f(x) \leq f(x_o)\}$, that $S(x_o)$ be bounded and that a bound for the norm of the Hessian matrix is known; and Fletcher and Powell [16], who use cubic interpolation. The hypotheses of Armijo's convergence theorem are more restrictive than those imposed by Curry, but less restrictive than those imposed by Goldstein, and, therefore, provide a Rule which is both practical and ensures convergence.

As mentioned before, a property of the Q.G.M. algorithm is that, under certain assumptions, for a quadratic, the recursive procedure used to invert Y_i will produce Y^{-1} after N steps. We will now prove this:

THEOREM 3.1

If f(x) is a quadratic and $y^T(x_{i+1})\ Y_i^{-1}\ e_{i+1} \neq 0$ then $\alpha_N = \alpha$ and $Y_N^{-1} = Y^{-1}$

PROOF:

$$\alpha_{i+1}{}^T\, y(x_{i+1}) = y^T(x_{i+1})\ \alpha_{i+1}$$

$$y^T(x_{i+1})\ \alpha_{i+1} = y^T(x_{i+1})\ \alpha_i + \frac{y^T(x_i) Y_i^{-1}\ e_{i+1}(f(x_{i+1}) - y^T(x_{i+1})\alpha_i)}{y^T(x_{i+1})\ Y_i^{-1}\ e_{i+1}}$$

Since $y^T(x_{i+1})\ Y_i^{-1}\ e_{i+1} \neq 0$, it follows that

$$\alpha_{i+1}{}^T\ y(x_{i+1}) = f(x_{i+1})$$

Also,

$$\alpha_{i+1}{}^T\ y(x_i) = y^T(x_i)\ \alpha_{i+1}$$

$$y^T(x_i)\alpha_{i+1} = y(x_i)\alpha_i + \frac{y^T(x_i) Y_i^{-1}\ e_{i+1}\ (f(x_{i+1}) - y^T(x_{i+1})\alpha_i)}{y^T(x_{i+1})\ Y_i^{-1}\ e_{i+1}}$$

Now,

$e_i{}^T\ Y_i = y^T(x_i)$ implies that

$y^T(x_i)\ Y_i^{-1} = e_i{}^T$

so that

$$y^T(x_i)Y_i^{-1}\, e_{i+1} = e_i^T\, e_{i+1} = 0$$

Therefore

$$a_{i+1}^T\, y(x_i) = a_i^T\, y(x_i) = f(x_i)$$

Proceeding in a similar manner

$$a_{i+1}^T\, y(x_k) = a_i^T\, y(x_k) = \ldots\ldots = a_k^T\, y(x_k) = f(x_k)$$

Since

$$a_N^T\, y(x_k) = f(x_k),\ k = 1, \ldots\ldots, N$$

it follows that $a_N = a$.

Also, since

$$Y_N = \begin{bmatrix} y^T(x_1) \\ \vdots \\ \vdots \\ y^T(x_N) \end{bmatrix}, \quad F_N = \begin{bmatrix} f(x_1) \\ \vdots \\ \vdots \\ f(x_N) \end{bmatrix}$$

and $Y_N\, a = F_N$

it follows that $Y_N^{-1} = Y^{-1}$.

Q.E.D.

It will be shown in Chapter 6 that the above conceptual algorithm can be modified slightly to produce an implementable algorithm which reduces the use of the costly Armijo Subprocedure.

C H A P T E R 4

EFFECT OF DATA STRUCTURES

CHAPTER 4 :

EFFECT OF DATA STRUCTURES

Interpolation methods are based on fitting a model to the given objective function. Each model has a certain number, say N, of independent coefficients which have to be determined in order to fit the model to the objective function. This is done by evaluating the objective function at, say M, points of a grid and then solving a set of M simultaneous equations with N unknowns. The effect of the data structures on these methods is tested by enlarging or decreasing the size of the grid (M) to include more or less than N points.

This is possible when the optimization methods are in grid-to-grid form and, therefore, in the first section we present a generalised grid-to-grid algorithm. Thereafter, it is applied to certain Interpolation methods, some of which are modified from point-to-point form to grid-to-grid form. The next two sections discuss the methods used to solve the abovementioned set of simultaneous equations when, firstly, M, the number of points in the grid, is less than the number of unknowns, N, and, secondly, when M is greater than N.

4.1 General Grid-to-Grid Algorithm

The general grid-to-grid algorithm for solving optimization problems with the aid of Interpolation methods is presented in a conceptual form which facilitates the proof of convergence. Before this is done some notation and terminology is in order:

M -	The number of points in the grid
N -	The number of unknowns
M = N -	The data structure is said to be <u>exact</u>
M > N -	The data structure is called <u>overdetermined</u>
M < N -	The data structure is called <u>underdetermined</u>

Whatever the model, the equations used to solve for the necessary coefficients may be expressed as follows:

$$y^T(x)a = v(x) \qquad (4.1.1)$$

where a is the vector of unknown independent coefficients of the chosen model, $y(x)$ is the vector of coefficients of a and $v(x)$ is a known scalar. Both $v(x)$ and the elements of $y(x)$ depend on the model used and are functions of x, $f(x)$ and $g(x)$ of the given objective function. If a has N components, then $y(x_i)$ and $v(x_i)$ are evaluated at the points x_i, $i = 1,2,\ldots,$ M and the resultant set of equations is written in matrix form as

$$Y\,a = V \qquad (4.1.2)$$

where

$$Y \triangleq \begin{bmatrix} y^T(x_1) \\ y^T(x_2) \\ \vdots \\ y^T(x_M) \end{bmatrix}, \quad V \triangleq \begin{bmatrix} v(x_1) \\ v(x_2) \\ \vdots \\ v(x_M) \end{bmatrix} \tag{4.1.3}$$

Using the above notation the general algorithm may be stated as follows:

Step 1. Assume x_o, η, N, L, M given

2. Evaluate $f(x_i)$ at an initial grid of points x_i, $i = 1,2,\ldots, M$.

3. Choose as the basepoint x_b the point of the grid at which $|f(x_i)|$ has the smallest value.

4. Order the points by increasing magnitude of the absolute value of $f(x_i)$, i.e.
$|f(x_i)| < |f(x_{i+1})|$, $i = 2,3, \ldots, (M-1)$.

5. If $||g(x_b)|| = 0$, stop; else go to 6.

6. Calculate $\alpha = B Q^{-1} R V$, where B,Q,R and V are matrices defined by the model and data structure used. If Q is singular, set $p = -g(x_b)$ and use Armijo's Subprocedure to generate a better point x_b, set $x_b = x_b$ and go to 4; else go to 7.

7. If the largest element of α is greater than L, set $p = -g(x_b)$ and use Armijo's Subprocedure to produce a better point x_ℓ, set $x_b = x_\ell$ and go to 4; else go to 8.

8. Calculate p, which is the search direction defined by the model being used.

9. If $|p^m g(x_b)| > \eta$, use Armijo's Subprocedure to

calculate x_ℓ, set $x_b = x_\ell$ and go to 4; else set $p = -g(x_b)$, use Armijo's Subprocedure to calculate x_ℓ, set $x_b = x_\ell$ and go to 4.

<u>Armijo's Subprocedure</u>

```
Step 1.  $k = 0$, $\alpha, \sigma \in (0,1)$
      .  $x_1 = x_b + \sigma^k p$
      3. If $f(x_1) - f(x_b) - \alpha\,\sigma^k\,p^T g(x_b) \leq 0$, return;
         else set $k = k + 1$ and go to 1.
```

Certain operations in the algorithm will now be more fully described before applying the algorithm to specific Interpolation Methods.

In Step 2 of the algorithm the points of the initial grid may be chosen in many different ways. They may be any set of points for which x, f(x) data is available, provided that their location uniquely defines the model, or they may be chosen in a methodical manner so as to represent the region around the starting point as well as possible. One possibility would be to have the grid include the initial point, x_o, and M-1 other points chosen from the set $x_o + e_j + e_k$, $j,k, = 1,2,\ldots.n$, $j \neq k$, where the e_j, e_k can either be zero or vectors along the co-ordinate axes. Winfield [20] suggests that the initial grid be spread over the largest region in which the modelling is effective, but since an estimate of this region is not readily available, the initial grid may only include points which are very close to the initial point.

In Step 4 the points of the grid are kept in a table in the order of increasing absolute function value. By ordering the points in this way at each iteration the new trial point is included as the basepoint in the first place and the point in the last place (having the largest absolute function value) is excluded. This results in a data table having a constant number of M entries only. Winfield [20] suggests a different way of using the data table. The co-ordinates of the points of the grid are defined relative to the basepoint, the points are ordered by increasing Euclidean distance from the basepoint and the data table has nearly 2M entries. This method uses more memory than the method we have presented, but has the advantage of being able to re-use points which are not among the M-1 closest points, but have not been discarded. This might occur, when, after a series of consecutive trial points have failed to reduce f(x), new successfull trial points are located again, the basepoint moves and the growing sphere of validity will enclose some of the former failure points, which now are active again, i.e. amongst the M-1 closest points, and serve to ward the search away from the previous unsuccessfull area.

Since it is very costly to restart a grid-to-grid algorithm (an entire new grid of points has to be chosen), in Step 9, if $|p^T g(x_b)| \leq \eta$, which means that the search direction p will not give a reasonable reduction in function value, we use Armijo's Subprocedure with $p = -g(x_b)$ instead of the p previousl chosen. As mentioned in Chapter 3, Armijo's Rule is very useful to ensure descent and convergence, and is therefore used in this conceptual algorithm.

The two major differences between point-to-point algorithms and grid-to-grid algorithms are as follows:

(i) In the grid-to-grid form, the matrix Q is not inverted recursively. This means that, at each iteration, a new set of equations is solved, leading to much computational work.

(ii) In the grid-to-grid form, the algorithm is started by an initial full grid, not one point only, and therefore, if the model is the same function as the objective function, the method will converge in 1 step instead of N steps for the point-to-point form. If M = N, the total number of function evaluations will, however, be the same for both forms.

It is important to note that the above general algorithm cannot be adapted to Winfield's method [20] for the reason that it has no search direction, but solves a constrained minimization problem instead. Most other Interpolation methods, however, do fit into the general algorithm and the forms of p, B, Q, R and V depend on the actual method and data structure used.

Following is a list of some Interpolation methods for the exact data structure case where M = N and

$B = I$

$Q = Y$

$R = I$.

Note that in this case Q is an N x N square matrix.

(i) Jacobson and Pels [21] - In this algorithm modified to grid-to-grid form, we have

$$Y \triangleq \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix}, \quad V \triangleq \begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix} \tag{4.1.4}$$

where : N = n + 2

$$y \triangleq \begin{bmatrix} g(x) \\ f(x) \\ -1 \end{bmatrix}, \quad v \triangleq x^T g(x) \tag{4.1.5}$$

and

$$a^T \triangleq [\beta^T, \gamma, \omega] \tag{4.1.6}$$

$$p \triangleq \sigma(x_b - \beta) \tag{4.1.7}$$

where β is an n vector of the location of the minimum, γ the degree of homogeneity, ω the scaled value ($\gamma\,\bar{\omega}$) of the minimum, $\bar{\omega}$ the actual value of the minimum, and σ is a coefficient such that $|\sigma| = 1$ and its sign is chosen so that $\sigma(x_b - \beta)^T g(x_b) < 0$.

(ii) Q.G.M. (See Chapter 3) - Since the Q.G.M. in Chapter 3 was presented in point-to-point form it also must be modified slightly to fit the general algorithm.

The definitions of Y, V and p in this case are :

$$Y \triangleq \begin{bmatrix} y^T(x_1) \\ \vdots \\ \vdots \\ y^T(x_N) \end{bmatrix}, \quad V \triangleq \begin{bmatrix} f(x_1) \\ \vdots \\ \vdots \\ f(x_N) \end{bmatrix} \tag{4.1.8}$$

where

$$N = \frac{n}{2}(n+1) + 1$$

$$y(x) \triangleq \begin{bmatrix} \tfrac{1}{2}g_1^{\,2}(x) \\ \vdots \\ \tfrac{1}{2}g_n^{\,2}(x) \\ g_1(x)g_2(x) \\ \vdots \\ g_1(x)g_n(x) \\ g_2(x)g_3(x) \\ \vdots \\ g_{n-1}(x)g_n(x) \\ 1 \end{bmatrix} \tag{4.1.9}$$

and

$$a \triangleq \begin{bmatrix} s_{11} \\ \vdots \\ s_{nn} \\ s_{12} \\ \vdots \\ s_{1n} \\ s_{23} \\ \vdots \\ s_{2n} \\ \vdots \\ s_{n-1,n} \\ \omega \end{bmatrix} \tag{4.1.10}$$

The definitions of Y, V and p in this case are :

$$Y \triangleq \begin{bmatrix} y^T(x_1) \\ \vdots \\ y^T(x_N) \end{bmatrix}, \quad V \triangleq \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix} \tag{4.1.8}$$

where

$$N = \frac{n}{2}(n+1) + 1$$

$$y(x) \triangleq \begin{bmatrix} \frac{1}{2}g_1^{\,2}(x) \\ \vdots \\ \frac{1}{2}g_n^{\,2}(x) \\ g_1(x)g_2(x) \\ \vdots \\ g_1(x)g_n(x) \\ g_2(x)g_3(x) \\ \vdots \\ g_{n-1}(x)g_n(x) \\ 1 \end{bmatrix} \tag{4.1.9}$$

and

$$a \triangleq \begin{bmatrix} s_{11} \\ \vdots \\ s_{nn} \\ s_{12} \\ \vdots \\ s_{1n} \\ s_{23} \\ \vdots \\ s_{2n} \\ \vdots \\ s_{n-1,n} \\ \omega \end{bmatrix} \tag{4.1.10}$$

$$p \triangleq \sigma \; S \; g(x_b) \tag{4.1.11}$$

where

$$S \triangleq (s_{ij})$$

is the minimum of the function and σ is a coefficient such that $|\sigma| = 1$ and its sign is chosen so that $p_i^T g(x_b) < 0$.

(iii) A special case of Jacobson and Pels [21] - This model is a special case of Jacobson and Pels' method, where γ, the degree of homogeneity, is taken as 2. This turns their model into equation (2.3.7), which is derived from a quadratic model. For this special case:

$$Y \triangleq \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix}, \quad V \triangleq \begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix} \tag{4.1.12}$$

where

$$N = n + 1$$

$$y \triangleq \begin{bmatrix} g(x) \\ -2 \end{bmatrix}, \quad v \triangleq x^T g(x) - 2f(x) \tag{4.1.13}$$

and

$$a^T \triangleq [\beta^T, \bar{\omega}] \tag{4.1.14}$$

$$p \triangleq \sigma(x_b - \beta) \tag{4.1.15}$$

where β is an estimated vector of the location of the minimum, $\bar{\omega}$ the minimum value of the function and σ as in section (i).

$$p \triangleq \sigma \; S \; g(x_b) \qquad (4.1.11)$$

where

$$S \triangleq (s_{ij})$$

is the minimum of the function and σ is a coefficient such that $|\sigma| = 1$ and its sign is chosen so that $p_i^T g(x_b) < 0$.

(iii) A special case of Jacobson and Pels [21] - This model is a special case of Jacobson and Pels' method, where γ, the degree of homogeneity, is taken as 2. This turns their model into equation (2.3.7), which is derived from a quadratic model. For this special case:

$$Y \triangleq \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix}, \; V \triangleq \begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix} \qquad (4.1.12)$$

where

$$N = n + 1$$

$$y \triangleq \begin{bmatrix} g(x) \\ -2 \end{bmatrix}, \quad v \triangleq x^T g(x) - 2f(x) \qquad (4.1.13)$$

and

$$a^T \triangleq [\beta^T, \bar{\omega}] \qquad (4.1.14)$$

$$p \triangleq \sigma(x_b - \beta) \qquad (4.1.15)$$

where β is an n vector of the location of the minimum, $\bar{\omega}$ the minimum value of the function and σ as in section (i).

4.2 Underdetermined Data Structure

In this case $M < N$ and the equation

$$Y a = V \tag{4.2.1}$$

has an infinite number of solutions. Two different approaches have been used to obtain a unique solution for a:

(i) To use the minimum norm solution (See Appendix A).

If Rank $(Y) = M$, then

$$a = Y^T (Y Y^T)^{-1} V \tag{4.2.2}$$

and in Step 6 of the algorithm of Chapter 4.1 we will have

$$B = Y^T \tag{4.2.3}$$

$$Q = Y Y^T \tag{4.2.4}$$

$$R = I \tag{4.2.5}$$

If Rank $(Y) < M$, in the general algorithm, we use Armijo's Subprocedure although another possibility is to use the pseudoinverse as follows:

$$a = Y^P V \tag{4.2.6}$$

For methods to calculate the pseudoinverse see Penrose [28] and Golub and Kahan [29].

One of Penrose's methods is based on the fact that any matrix can be partitioned in the form:

$$Y = \begin{bmatrix} A & B \\ C & C A^{-1} B \end{bmatrix} \tag{4.2.7}$$

where A is a non singular sub-matrix whose rank is equal to that of the whole matrix. Using this partitioning it is easily verified that

$$Y^P = \begin{bmatrix} A^T P A^T & A^T P C^T \\ B^T P A^T & B^T P C^T \end{bmatrix} \qquad (4.2.8)$$

where

$$P = (A A^T + B B^T)^{-1} A(A^T A + C^T C)^{-1} \qquad (4.2.9)$$

Golub and Kahan's idea is to use the singular value decomposition of a matrix, which is

$$Y = U \Sigma V^T \qquad (4.2.10)$$

and U and V are unitary matrices and Σ is a rectangular diagonal matrix of the same size as Y with non-negative real diagonal entries which are called the singular values of Y. Using this decomposition it can be shown that

$$Y^P = V \Sigma^I U^T \qquad (4.2.11)$$

where Σ^I is obtained from Σ by replacing each positive diagonal entry by its reciprocal.

(ii) The second approach is to use the fact that if $M < N$ then there are (N-M) unknowns which may be chosen arbitrarily. Once these have been chosen, we are left with a set of M equations with M unknowns, which is solved easily. In other words (N-M) components out of the N components of α are chosen arbitrarily. For example if $M = N - 1$, we need to choose one component and give it an arbitrary value. The most reasonable choice for this in the three Interpolation Methods mentioned in Chapter 4.1 is ω or $\bar{\omega}$. The minimum value or scaled minimum value is of no large

minimum value or scaled minimum value is of no large

significance in choosing the search direction p and can therefore be taken arbitrarily as zero, for example. If $M < N - 1$, this approach is not of much practical use for Jacobson and Pels' method because it essentially boils down to a random choice method. However, for the Q.G.M. method, the components of a are the elements of the matrix S and ω, so it is possible to choose the components of a so that S becomes a diagonal matrix. This method is possible, of course, only when $M = n$ or $M = n+1$ depending on whether ω is left as an unknown or not. If $M < N - 1$ and $M \neq n$ or $M \neq n+1$ then, again, it is very difficult to find criteria for the choice of elements of a.

The above approaches may also be applied to Winfield's method (See Appendix B.2). This method is already in grid-to-grid form and the only modifications necessary are to Steps 1 and 4 which become :

Step 1 : Evaluate $f(x_i)$ at an initial grid of M points

Step 4 : Compute A,b,d (altogether $N = \frac{1}{2}(n+1)(n+2)$ unknowns) so that $\frac{1}{2}y_j^T A\, y_i + b^T y_j + d = f(x_j)$, $j = 1, \ldots, M$.

significance in choosing the search direction p and can therefore be taken arbitrarily as zero, for example. If $M < N - 1$, this approach is not of much practical use for Jacobson and Pels' method because it essentially boils down to a random choice method. However, for the Q.G.M. method, the components of a are the elements of the matrix S and ω, so it is possible to choose the components of a so that S becomes a diagonal matrix. This method is possible, of course, only when $M = n$ or $M = n+1$ depending on whether ω is left as an unknown or not. If $M < N - 1$ and $M \neq n$ or $M \neq n+1$ then, again, it is very difficult to find criteria for the choice of elements of a.

The above approaches may also be applied to Winfield's method (See Appendix B.2). This method is already in grid-to-grid form and the only modifications necessary are to Steps 1 and 4 which become :

Step 1 : Evaluate $f(x_i)$ at an initial grid of M poin

Step 4 : Compute A,b,d (altogether $N = \frac{1}{2}(n+1)(n+2)$ unknowns) so that $\frac{1}{2}y_j^T A y_i + b^T y_j + d = f(x_j)$, $j = 1, \ldots, M$.

4.3 Overdetermined Data Structure

The overdetermined data structure case is defined by having $M > N$, which gives the equation

$$Y \alpha = V \tag{4.3.1}$$

a solution only if V is in the range of Y. If V is not in the range of Y, which is usually the case, we shall seek a least squares solution (See Appendix A).

The least squares solution is unique and can be found in one of two ways :

(i) If Rank (Y) = N, then

$$\alpha = (Y^T Y)^{-1} Y^T V \tag{4.3.2}$$

and we substitute in Step 6 of the general algorithm of Chapter 4.1

$$B = I \tag{4.3.3}$$

$$Q = Y^T Y \tag{4.3.4}$$

$$R = Y^T \tag{4.3.5}$$

If Rank (Y) < N, we use Armijo's Subprocedure to generate a better point in the general algorithm.

(ii) The second method is to use the pseudoinverse to find α, as is described in Chapter 4.2.

A totally different approach is also suggested for the overdetermined data structure case. In order to fit a model exactly to the objective function, if α has N components, we need N points. Therefore, if $M > N$, we can choose a certain number, say K, of grids each having N points. We then obtain K models and solve K sets of N equations with N unknowns. This gives us K different solutions for α and we can either choose the solution which gives the best new trial point or we can try a linear combination (possibly weighted) of the different minima supplied by each model.

If we choose the best minimum supplied by the K solutions of α, the general algorithm of Chapter 4.1 has to be modified from Step 6 onwards as follows:

Step 6 : Calculate $\alpha_i = B_i Q_i^{-1} R_i V_i$, $i = 1, \ldots, K$ where K is the number of grids and B_i, Q_i, R_i and V_i are matrices defined by the model and data structure used. If any of the Q_i's are singular, disregard that grid. If all Q_i's, $i = 1, \ldots, K$ are singular, set $p = -g(x_b)$ and use Armijo's Subprocedure to generate a better point x_ℓ, set $x_b = x_\ell$ and go to 4; else go to 7.

7 : If the largest element of all α_i's is greater than L, set $p = -g(x_b)$ and use Armijo's Subprocedure to produce a better point x_ℓ, set $x_b = x_\ell$ and go to 4; else go to 8.

8 : Calculate p_i, $i = 1, \ldots, K$ where p_i is the search direction for each grid.

9 : If any $|p_i^T g(x_b)| \leq \eta$, disregard that grid. If all $|p_i^T g(x_b)| \leq \eta$, $i = 1, \ldots, K$ set $p = -g(x_b)$, use Armijo's Subprocedure to calculate x_ℓ, set $x_b = x_\ell$ and go to 4; else use Armijo's Subprocedure K times to generate x_{ℓ_i}, $i = 1, \ldots, K$ and go to 10.

10 : Calculate $f(x_{\ell_i})$, $i = 1, \ldots, K$, find the x_ℓ, say x_{ℓ_j}, for which $|f(x_{\ell_j})| < |f(x_{\ell_i})|$, $i = 1, \ldots, K$, $j \neq i$, set $x_b = x_{\ell_j}$ and go to 4.

An even better point than the x_{ℓ_j} of Step 10 can usually be obtained by choosing x_{ℓ_j} as

$$x_{\ell_j} = \sum_{k=1}^{K} Z_k \, x_{\ell_k} \tag{4.3.6}$$

where the Z_k's are weights such that $\sum_{k=1}^{K} Z_k = 1$. The weights may be chosen so as to give points closer to x_b more influence or larger weights may be given to points having smaller function values.

Whichever way this method is used, it becomes very unwieldy, (especially if $(M - N) > 1$) if we choose

$$K = C_M^N \tag{4.3.7}$$

where C_M^N is the total number of combinations of choosing groups of N numbers out of a total of M numbers. Therefore, we suggest choosing not more than four or five different grids. They may be chosen arbitrarily or an attempt can be made to have each model represent a different region in the n dimensional space surrounding the initial point.

C H A P T E R 5

CONVERGENCE OF THE ALGORITHMS

CHAPTER 5 :

CONVERGENCE OF THE ALGORITHMS

It was proved in Chapter 3 that the Q.G.M. algorithm possesses the property of quadratic convergence, i.e. on a quadratic function it convergences in a finite number of steps, N, where $N = \frac{n}{2}(n+1) + 1$. Also, in Chapter 4, it was shown that the three algorithms presented in grid-to-grid form will converge in one step if the model is the same as the objective function.

In this Chapter we discuss the conditions and models used for the convergence of these algorithms for general functions and, in the second section, supply a proof of convergence of the Q.G.M. algorithm.

5.1 Algorithm Models and Convergence Conditions

Except for Winfield's method [20], all other algorithms in Chapters 3 and 4 use the iterative formula

$$x_{i+1} = x_i - \lambda_i p_i \qquad (5.1.1)$$

where λ_i is the step size or steplength, p_i is the search direction and its sign is chosen so as to ensure descent. Different optimization methods using (5.1.1) will need different conditions stipulated on the objective function f, on λ and on p in order to prove convergence. The requirements on f may be that it is continuously differentiable or even twice continuously differentiable. The conditions on p_i and λ_i may be those which choose λ_i to minimize $f(x_i + \lambda p_i)$ or may use Armijo's Rule [22], which chooses $\lambda_i = \sigma^{k_i}$ where k_i is the smallest integer $k \geq 0$ that satisfies

$$f(x_i + \sigma^k p_i) - f(x_i) - \sigma^k \alpha p_i^T g(x_i) \leq 0 \qquad (5.1.2)$$

for some fixed $\alpha, \sigma \in (0,1)$. These are just a few examples of conditions which may be imposed on f, λ_i and p_i. However, a number of these conditions are common to most algorithms and it would be wise to provide a systematic approach to the study of convergence properties of algorithms.

This is done by Polak [27] and others, who make use of models for algorithms. A whole class of algorithms is represented by a generalised model which is proved to be convergent under certain assumptions. The advantage of this idea is that if an algorithm is found to fit a certain model, it need only fulfill the conditions of that model to be proved convergent and, therefore, a separate proof of convergence is not necessary for each algorithm.

The algorithms we are concerned with fit into one of Polak's models and it is this one which we will present to solve the abstract problem:
Given a closed subset T of a Banach space B, construct points in T having property P.

Algorithm Model

Let A be a mapping from T to 2^T, the set of non-empty subsets of T and c be the stop rule, a mapping from T to R^1.

```
Step 0 : Compute $x_o \in T$
     1 : Set $i = 0$
     2 : Compute a point  $y \in A(x_i)$
     3 : Set $x_{i+1} = y$
     4 : If $c(x_{i+1}) \geq c(x_i)$, stop;  else set $i = i + 1$
         and go to 2.
```

This model is presented in a very generalised form. The stop rule c, for example, might be the objective function or the norm of the gradient. Points having property P are usually called desirable points, which is more general than stationary point and could include a saddle point, a root of a system of equations or a stability point of a differential equation.

The convergence theorem for this model is as follows:

Theorem 5.1

Suppose that

(i) $c(x)$ is either continuous at all non-desirable points $x \in T$, or else $c(x)$ is bounded from below for $x \in T$.

(ii) for every $x \in T$ which is not desirable, there exists an $\epsilon(x) > 0$ and a $\sigma(x) < 0$ such that

$$c(x'') - c(x') \leq \sigma(x) < 0$$

for all $x' \in T$ such that $\|x' - x\| \leq \epsilon(x)$ and for all $x'' \in A(x')$.

Then either the sequence $\{x_i\}$ constructed by the algorithm model is finite and its next to last element is desirable, or else it is infinite and every accumulation point of $\{x_i\}$ is desirable.

Proof See Polak [27]

In order to prove convergence of a particular algorithm, the mappings c and A must be determined, the property P must be decided upon and the existence of accumulation points must be guaranteed. A proof of convergence for their method was supplied by Jacobson and Pels [21], and since the proof applies to both the grid-to-grid and point-to-point forms, we will restrict ourselves to a proof of convergence for the Q.G.M.

5.2 Proof of Convergence for the Q.G.M.

In order to apply Theorem 5.1 and the model mentioned in the previous section to the Q.G.M. method, we make the following definitions and assumptions:

(i) x_i is desirable (has property P) if $\|g(x_i)\| = 0$

(ii) Let f(x) correspond to c(x) in the model

(iii) Let f(x) be cont. diff. in R^n

(iv) $x_o \in R^n$ is chosen so that $V = \{x | f(x) \leq f(x_o)\}$ is compact

(v) $W > \sup_x \|g(x)\|, x \in V$

Theorem 5.2

Let $\{x_i\}$ be the sequence in R^n generated by the Q.G.M. algorithm presented in Chapter 3. Then either the sequence is finite and terminates at a desirable point or else it is infinite and every accumulation point x^* of $\{x_i\}$ is desirable.

Proof

If the sequence is finite, the test for $\|g(x)\| = 0$ ensures that the last point is desirable. In the case of an infinite sequence we need to prove that conditions (i) and (ii) of Theorem 5.1 are satisfied.

Clearly, (i) is satisfied by the assumption that f(x) is continuous.

To prove condition (ii) satisfied, we note that either $p_i = -g(x_i)$ or $p_i = \sigma_i S g(x_i)$ where $|\sigma_i| = 1$ and its sign is chosen so that $p_i^T g(x_i) < 0$.

Clearly, in either case

$$-p_i^T g(x_i) > 0$$

and since the algorithm ensures that

$$|p_i^T g(x_i)| \geq \eta$$

we have

$$-p_i^T g(x_i) \geq \eta$$

and because of assumption (v) we can choose an $\epsilon > 0$ such that

$$\epsilon \| g(x_i) \|^2 \leq \eta$$

herefore $-p_i^T g(x_i) \geq \epsilon \| g(x_i) \|^2$

Note also that

$$\| p_i \| = \| S g(x_i) \|$$

and from assumption (v) and the check in the algorithm that the largest element of α or S is not greater than L, we see that there exists an L1 > 0 so that

$$\| p_i \| \leq L1$$

Define

$$A(x) \triangleq \{ y = x + \delta[\hat{k}(x,p)] p \mid p \in D(x) \}$$

where $\delta[\hat{k}(x,p)]$ is the largest δ, $0 < \delta \leq 1$ generated by the Armijo Subprocedure, to satisfy

$$f(x + \delta[k(x,p)] p) - f(x) - \delta[k(x,p)] \alpha \, p^T g(x) \leq 0$$

and where

$$D(x) \overset{\Delta}{=} \{ p | \; \|p\| \geq L1 \text{ and } -p^T g(x) > \epsilon \|g(x)\|^2 \}$$

For x non-desirable we define

$$\Delta[(x,p)] \overset{\Delta}{=} f(x+\lambda(x,p)p) - f(x) - \alpha \; \lambda(x,p)p^T g(x)$$

Using the mean value theorem

$$\Delta[\lambda(x,p)] = -[p^T g(x) - p^T g(\xi) - (1-\alpha)p^T g(x)]\lambda(x,p)$$

$$\leq -[p^T g(x) - p^T g(\xi) + (1-\alpha)\epsilon \|g(x)\|^2]\lambda(x,p)$$

for $\xi \in [x, x+\lambda(x,p)p]$

and for all $p \in D(x)$.

Consider

$$\tilde{\Delta}[\lambda] = -\lambda(x,\tilde{p})[\tilde{p}^T g(x) - \tilde{p}^T g(\xi) + (1-\alpha)\epsilon\|g(x)\|^2]$$

where $\tilde{p} \in \tilde{D} \overset{\Delta}{=} \{p | \; \|p\| \leq L1\}$

and $\xi \in [x, x+\lambda(x,\tilde{p})\tilde{p}]$, $x \in R^n$

Since $\|p\|$ is bounded and $g(x)$ is continuous, there exists a $\bar{\lambda}(x) > 0$ such that

$$\tilde{\Delta}[\bar{\lambda}(x)] \leq \delta(x) < 0 \text{ for all } \tilde{p} \in \tilde{D}$$

Since $D(x)$ is a subset of $\tilde{D}$

$$\Delta[\bar{\lambda}(x)] \leq \delta(x) < 0 \text{ for all } p \in D(x)$$

By continuity of $g(x)$

$$-\bar{\lambda}(x)[\tilde{p}^T g(x') - \tilde{p}^T g(\xi') + (1-\alpha)\epsilon\|g(x')\|^2] \leq \frac{\delta(x)}{2}$$

Therefore

$$-\lambda(x)[\tilde{p}^T g(x') - \tilde{p}^T g(\xi') + (1-\alpha) \; \|g(x')\|^2] \leq \frac{\delta(x)}{2}$$

for all $\tilde{p} \in \tilde{D}$ and for all $x' \in B(x,\epsilon(x))$

This implies that

$$f(x'+\bar{\lambda}(x)p) - f(x') - \bar{\lambda}(x)\alpha p^T g(x') \leq \frac{\delta(x)}{2}$$

for all $x' \in B(x,\epsilon(x))$ and for all $p \in D(x')$

From our definition of $A(x)$ we have that

$\delta[\hat{k}(x',p)] \geq \bar{\lambda}(x)$ where

$$f(x'+\delta[\hat{k}(x',p)]p) - f(x') - \delta[\hat{k}(x',p)]\alpha\, p^T g(x') \leq 0$$

Therefore

$$\begin{aligned} f(x'+\delta[\hat{k}(x',p)]p) - f(x') &\leq \delta[\hat{k}(x',p)]\alpha\, p^T g(x') < 0 \\ &\leq \bar{\lambda}(x)\ \alpha\ p^T g(x') \\ &\leq -\ \bar{\lambda}(x)\ \alpha \| g(x') \|^2 \\ &\leq \frac{\alpha\ \bar{\lambda}(x)\ \alpha\ \| g(x) \|^2}{2} \end{aligned}$$

for all $x' \in B(x,\epsilon(x))$ and for all $p \in D(x')$

thus satisfying the second condition of Theorem 5.1.

C H A P T E R 6

COMPUTATIONAL RESULTS

CHAPTER 6 :

COMPUTATIONAL RESULTS

The easiest, and perhaps most fruitful, way to test the effectiveness of an algorithm is to use it to solve test problems and then compare it with other algorithms. For this comparison to have any meaning, some criteria for evaluation must be established. This is done in the first section of this Chapter and following this is a list, in the second section, of "classical" test functions, which, because of their properties, are used to test the algorithms.

In the third section we present the results of using our algorithms to solve the test problems and compare these algorithms amongst themselves, for a number of different data structures, and to other standard minimization techniques. We end the Chapter by presenting some conclusions based on the numerical results.

6.1 Test Criteria

The points of interest when trying to measure the performance of an algorithm are the following:

(i) Robustness - success in obtaining an optimal solution, to within a certain precision, for a wide range of problems.

(ii) Number of function evaluations - including evaluations of the gradient vector and Hessian matrix.

(iii) Computer time to termination to within the desired degree of precision.

(iv) Simplicity of use - Time required to introduce data and functions into the computer program.

Not only are these properties difficult to measure, but the problem is complicated further by the fact that some of these properties depend to quite a large extent on how the algorithms are programmed for the computer. Different techniques of solving simultaneous equations, tests of matrices for singularity, reset conditions and the like can influence the performance of an algorithm greatly. Since the details of programming affect mainly properties (iii) and (iv), it is difficult to use them as practical criteria for evaluating algorithms.

Computer time to termination, property (iii), could be an excellent criteria for evaluation if one could ensure that the type of computer, the input/output routines, the method of time-sharing, and the method of coding the algorithms are always the same for different algorithms. Then, if an algorithm has fewer function evaluations but more computational work solving equations the total time taken would be a good measure of its effectiveness compared to another algorithm which has many more function evaluations but no inversion of matrices.

Since it is virtually impossible to use all algorithms in the same way and under exactly the same conditions and because this kind of information is usually missing from reports in the literature, the commonest criteria used are properties (i) and (ii). Criterion (i) is easily tested by solving as many difficult test problems as possible. If the test problems are chosen to have especially flat plateaus or steep valleys one can hope to predict the general effectiveness of an algorithm in solving other problems by its performance in solving these test problems. Criterion (ii), which is used in this Chapter, is also easily tested but has a number of disadvantages which should be noted.

First of all, when evaluating the number of function evaluations, the evaluations of the gradient vector and even the Hessian matrix must be included, and a decision as to how these are to be weighted relative to the evaluation of the objective function itself must be made. Secondly, the number of function evaluations may be reduced by different methods such as matrix operations, heuristic operations and so forth, so that in general a comparison based solely on function evaluations can easily be misleading.

In our case, however, all algorithms except that of Winfield [20] use similiar methods of solution and, therefore, a comparison based on function evaluations is a reasonably good indication of their performance. Since our algorithms do not evaluate the Hessian matrix we need only consider the weighting of the gradient vector evaluation. If f(x) is a function of n variables, then the gradient is an n vector, and, therefore, each gradient vector evaluation is taken as n function evaluations. For the remainder of this Chapter the term "function evaluations" will refer to the sum total of objective function and gradient vector evaluations.

An additional factor which must be common to all algorithms in order to make the comparison meaningfull is the termination criteria used to stop execution. Although this is not a measure of performance, it depends on the required degree of precision, which is associated with the concept of robustness. Algorithms may be designed to terminate on achieving a given small value for one of the following:

a) A fractional change in f(x)

b) A fractional change in x

c) The norm of the gradient

Each of the above, if used alone, has its disadvantages. a) could terminate on a flat plateau, b) on a steep slope and c) at a saddle point.

function evaluations can easily be misleading.

In our case, however, all algorithms except that of Winfield [20] use similiar methods of solution and, therefore, a comparison based on function evaluations is a reasonably good indication of their performance. Since our algorithms do not evaluate the Hessian matrix we need only consider the weighting of the gradient vector evaluation. If f(x) is a function of n variables, then the gradient is an n vector, and, therefore, each gradient vector evaluation is taken as n function evaluations. For the remainder of this Chapter the term "function evaluations" will refer to the sum total of objective function and gradient vector evaluations.

An additional factor which must be common to all algorithms in order to make the comparison meaningfull is the termination criteria used to stop execution. Although this is not a measure of performance, it depends on the required degree of precision, which is associated with the concept of robustness. Algorithms may be designed to terminate on achieving a given small value for one of the following:

a) A fractional change in f(x)

b) A fractional change in x

c) The norm of the gradient

Each of the above, if used alone, has its disadvantages. a) could terminate on a flat plateau, b) on a steep slope and c) at a saddle point.

Therefore, the uniform termination criteria we use, includes both a) and c) and the algorithm terminates when both criteria are fulfilled.

6.2 Test Functions

The following test functions have been chosen for the reason that they are among the most common problems used in the literature to [illegible] the performance of algorithms. Although numerical comparisons are of limited value when applied to problems using a single initial point, most problems mentioned in the literature have "classical" starting points and it is these we will use mostly. In the following list the starting point will be denoted by x_o, the minimum by x_m and the minimum function value by $f(x_m)$.

1) Rosenbrock's Function (Fletcher-Powell, 1963)

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

$$x_o^T = (-1.2,\ 1.0)$$

$$x_m^T = (1.0,\ 1.0), \quad f(x_m) = 0$$

This function has a steep curved valley along the curve $x_2 = x_1^2$

2) Beale (1958)

$$f(x) = [1.5-x_1(1-x_2)]^2 + [2.25-x_1(1-x_2^2)]^2 + [2.625-x_1(1-x_2^3)]^2$$

$$x_o^T = (0.1,\ 0.1)$$

$$x_m^T = (3.0,\ 0.5), \ f(x_m) = 0$$

This function has a narrow curving valley approaching the line $x_2 = 1$

3) Helical Valley (Fletcher-Powell, 1963)

$$f(x) = 100[(x_3-100)^2 + (r-1)^2] + x_3^2$$

where

$$2\pi\theta = \begin{cases} \tan^{-1}(\frac{x_2}{x_1}) & , x_1 < 0 \\ \pi + \tan^{-1}(\frac{x_2}{x_1}) & , x_1 < 0 \end{cases}$$

$$r = (x_1^2 + x_2^2)^{\frac{1}{2}}$$

$$x_o^T = (-1,0,0)$$

$$x_m^T = (1,0,0),\ f(x_m) = 0$$

4) Quartic with Singular Hessian (Fletcher-Powell, 1963)

$$f(x) = (x_1+10x_2)^2 + 5(x_3-x_4)^2 + (x_2-2x_3)^4 + 10(x_1-x_4)^2$$

$$x_o^T = (3, -1, 0, 1$$

$$x_m^T = (0,0,0,0),\ f(x_m) = 0$$

This function has a flat minimum.

5) Four Dimensional Banana (Coleville, 1968)

$$f(x) = 100(x_1^2-x_2)^2 + (1-x_1)^2 + 90(x_3^2-x_4) + (1-x_3)^2 +$$
$$10.1[(x_2-1)^2 + (x_4-1)^2]^2 + 19.9(x_2-1)(x_4-1)$$

$$x_o^T = (-3, -1, -3, -1)$$

$$x_m^T = (1,1,1,1),\quad f(x_m) = 0$$

This function has a banana shaped ridge and is a four dimensional version of Rosenbrock's function.

6.3 Results and Comparisons

In order to simplify the presentation of our results it is necessary first to introduce some parameters and notation:

M	- Number of points in grid
N	- Number of unknowns necessary for fitting certain model.
C_i	- The i th unknown in list of N unknowns.
Δx	- Parameter indicating grid size. It is the approximate distance in n space of points of grid from the starting point x_o.
δ_1, δ_2	Stop criteria.

METHOD 1 -	The Q.G.M. in point-to-point form (See algorithm in Chapter 3.2).
METHOD 2 -	General grid-to-grid algorithm (See Chapter 4.1).
METHOD 2A -	Method 2 applied to the Q.G.M. (See Chapter 4 - Pages 44-46).
METHOD 2B	Method 2 applied to Jacobson and Pels' algorithm (See Chapter 4 - Page 44).
METHOD 2C -	Method 2 applied to the special case of Jacobson and Pels' algorithm where γ, the degree of homogeneity, is set equal to 2 (See Chapter 4 - Page 46).
METHOD 2D -	Winfield's SQM method (See Appendix B.2).

FUNCTION i	-	The i th function in the list of Chapter 6.2.
L	-	Upper bound of elements of α in Methods 1 and 2.
η_1, η	-	Parameters used in Methods 1 and 2 respectively to ensure that the chosen search direction leads to a reasonable decrease in function value.
η_2	-	Parameter used in Method 1 to ensure that Y and α can be updated using the Sherman-Morrison formula.

The results presented in this section have been obtained from implementable algorithms, as opposed to the conceptual algorithms described in Chapters 3 and 4. The two major differences between the conceptual and implementable algorithms are:

(i) In the implementable algorithm the stop criterion is not $\|g(x_i)\| = 0$ and it is, as mentioned in section 6.1, a combination of

$$f(x_{i+1}) - f(x_i) \leq \delta_1 \tag{6.3.1}$$

and

$$\|g(x_{i+1})\| \leq \delta_2 \tag{6.3.2}$$

The algorithm terminates only if both (6.3.1) and (6.3.2) are satisfied.

(ii) In order to facilitate the proof of convergence the conceptual algorithms <u>always</u> use Armijo's Subprocedure to ensure descent. The implementable algorithms, however, calculate the minimum of the fitted model and if this point produces a lower function value, it is chosen as the new trial point x_{i+1}. Only if this is not the case, is Armijo's Subprocedure used to find x_{i+1} such that descent is ensured.

All computations were performed in double precision, using FORTRAN IV, on the IBM 360/50 computer of the University of the Witwatersrand and the same techniques of solving linear equations, matrix operations and so on, were used so as to make the comparison of results using function evaluations as meaningful as possible. The values of the parameters were chosen as follows: $\delta_1 = 10^{-8}$, $\delta_2 = 10^{-4}$, $L = 10^{40}$, $\eta = \eta_1 = 10^{-16}$, $\eta_2 = 10^{-24}$. All tables give the total number of function evaluations (i.e. function plus gradient evaluations) necessary to reach the minima of the test functions from the respective starting points mentioned in Chapter 6.2.

Table 3.1 presents the results of Method 1, and, for comparison, includes the results of Jacobson and Oksman's algorithm and the IBM System/360 Scientific Subroutine Package version of Fletcher and Powell's algorithm. The comparison is included in order to give a rough idea of the performance of Method 1 but, as mentioned previously, is of little practical value because of the nonuniformity of the different algorithms.

The IBM Fletcher Powell routine performs poorly for Test Function 1 because it uses a linear search which brackets the minimum before using cubic interpolation and another figure quoted in the literature is 240. Since Method 1 is essentially a derivation of Newton's method using a quadratic model its results are generally of the same magnitude as that of Fletcher and Powell but decidedly worse than these of Jacobson and Oksman, who use a homogeneous model.

METHODS / FUNCTIONS	METHOD 1	JACOBSON AND OKSMAN	FLETCHER AND POWELL
1	258	207	501
3	259	136	304
4	487	230	400
5	968	675	805

TABLE 6.3.1

Tables 6.3.2 to 6.3.5 show the influence of initial grid size (represented by Δx) on Methods 2A to 2D for the case where M = N. The different methods were run for more grid sizes than those presented in the Tables and it was found that it is very difficult to establish criteria for choosing the optimal grid size. Although Winfield suggests choosing a large initial grid we found that usually the smaller grids were better conditioned and more robust. Large grids tend to become unwieldy and complicated from the computational point of view and when using Method 2 on a new unknown Test Function it is hard to tell, without previous knowledge of the function, just how large the initial grid should be so as to include the minimum.

Another factor which affects the results of Method 2 is the choice of the points of the grid. Even though Δx represents the size of the grid ,the points themselves may be chosen in many different ways within the frame of a given Δx. Tables 6.3.2 - 6.3.5 all use the same initial grid for each Test Function and although Methods 2A-2D were tried with different initial points for the same Δx, the results did not differ greatly from those given in the Tables.

FUNCTIONS \ Δx	0.001	0.01	0.1	10	100	1000
1	276	363	272	350	299	278
2	197	185	143	176	191	179
3	408	315	425	145	329	147
4	594	823	672	642	709	877
5	1705	1587	1403	1693	1310	1651

TABLE 6.3.2 - METHOD 2A, M = N

FUNCTIONS \ Δx	0.001	0.01	0.1	10	100	1000
1	248	265	239	234	273	259
2	156	123	149	167	174	153
3	242	299	290	228	220	220
4	440	353	369	494	491	595
5	1364	1486	1198	1077	1098	1230

TABLE 6.3.3 - METHOD 2B, M = N

FUNCTIONS \ Δx	0.001	0.01	0.1	10	100	1000
1	214	228	197	243	279	253
2	114	125	133	159	149	141
3	332	274	299	293	348	203
4	532	983	511	745	790	960
5	1403	1239	1498	1321	1029	1386

TABLE 6.3.4 - METHOD 2C, M = N

λx / FUNCTIONS	0.001	0.01	0.1	10	100	1000
1	86	65	87	84	71	59
2	43	50	56	36	53	31
3	110	91	97	93	104	87
4	115	123	110	103	121	129
5	203	256	249	237	206	225

TABLE 6.3.5 - METHOD 2D, M = N

The effect of data structure size is presented in the following Tables. In Tables 6.3.6 - 6.3.9, if $M < N$ we use equations (4.2.2) to (4.2.5) for the minimum norm solution, and if $M > N$ we use equations (4.3.2) to (4.3.5) for the least squares solution. In both cases, when Q is singular Armijo's Subprocedure is used. Since small grids were found to be the best to work with, Tables 6.3.6 - 6.3.9 present the results when $\Delta x = 0.001$, and the results for different size grids can be found in Appendix C.

For all Methods, when the data structure size varies (i.e. M is greater than, equal to, or less than, N), the best results are usually achieved when $M = N$. This is also the case for different initial grid sizes (See Appendix C, Tables C.1 - C.4) and for equal grid sizes having different initial points. Better results than those achieved when $M = N$ occur more frequently for the cases of $M > N$ than for those of $M < N$, although there are no consistent guidelines to the optimum value of M which produces the best results. In addition it must be noted that even when the results are better than those of the case $M = N$, the difference in results is not an appreciable one.

The general trend is that the number of function evaluations increases as M increases from the value of N and as it decreases from the value of N. In the latter case the increase is much more rapid and the systems of equations less stable although Q was never singular in any of the examples.

A factor which does not appear in the Tables is the amount of computational work involved in solving the sets of simultaneous equations. When $M > N$ the sets of equations are always $N \times N$ and therefore as M increases there is no increase in the amount of computational work. However, when $M < N$, the sets of equations are $M \times M$ and as a result, if M decreases, so does the size of the set of equations and the computational work. In fact this compensates only slightly for the large increase in the number of function evaluations. This can be illustrated by taking the extreme case where $M = 1$ and there are no sets of equations to be solved. In this case the number of function evaluations may be from twice to ten times the number when $M = N$, and the total time taken to reach the minimum is also greater.

FUNCTIONS \ M	M = 1	M = N-2	M = N-1	M = N	M = N+1	M = N+2	M = 2N
1	608	943	750	276	283	314	329
2	653	834	581	197	192	204	199
3	1613	820	910	408	361	458	508
4	1108	698	415	594	581	671	856
5	2049	1856	1714	1705	1683	1765	1626

TABLE 6.3.6 - METHOD 2A, $\Delta x = 0.001$

FUNCTIONS \ M	M = 1	M = N-2	M = N-1	M = N	M = N+1	M = N+2	M = 2N
1	1187	1434	264	248	236	281	417
2	1824	553	282	156	148	131	221
3	921	785	543	242	198	324	397
4	1483	497	640	440	475	543	637
5	2971	1946	1328	1364	1664	1981	2425

TABLE 6.3.7 - METHOD 2B, $\Delta x = 0.001$

FUNCTIONS \ M	M = 1	M = N-2	M = N-1	M = N	M = N+1	M = N+2	M = 2N
1	941	941	449	214	230	247	302
2	865	865	379	114	158	139	188
3	2824	2951	414	332	326	426	508
4	25	153	541	532	593	660	812
5	2712	2643	1361	1403	1973	1994	2114

TABLE 6.3.8 - METHOD 2C, $\Delta x = 0.001$

FUNCTIONS \ M	M = 1	M = N-2	M = N-1	M = N	M = N+1	M = N+2	M = 2N
1	235	198	153	86	90	110	127
2	140	171	125	43	48	41	61
3	296	195	221	110	98	136	168
4	235	167	131	115	103	148	193
5	334	281	239	203	216	198	245

TABLE 6.3.9 - METHOD 2D, $\Delta x = 0.001$

Table 6.3.10 presents the results obtained when the last coefficient, C_N, in Method 2 was taken as zero for the case of M = N - 1 (This coefficient is d for Winfield's Quadratic Model and w for the Q.G.M. and Homogeneous Model). When comparing these results with the results of Tables 6.3.6 - 6.3.9 we see that for the same case of M = N - 1 this method is invariably better than the method presented in the abovementioned Tables but the performance is not enhanced when compared to the case of M = N even though the sets of equations in this case are (N - 1) x (N - 1) instead of N x N.

This method was also tried with different initial grid sizes with similiar results. For some of these different initial grid sizes see Appendix C - Table C.5.

Table 6.3.10 presents the results obtained when the last coefficient, C_N, in Method 2 was taken as zero for the case of $M = N - 1$ (This coefficient is d for Winfield's Quadratic Model and w for the Q.G.M. and Homogeneous Model). When comparing these results with the results of Tables 6.3.6 - 6.3.9 we see that for the same case of $M = N - 1$ this method is invariably better than the method presented in the abovementioned Tables but the performance is not enhanced when compared to the case of $M = N$ even though the sets of equations in this case are $(N - 1) \times (N - 1)$ instead of $N \times N$.

This method was also tried with different initial grid sizes with similiar results. For some of these different initial grid sizes see Appendix C - Table C.5.

FUNCTIONS \ METHOD	2A	2B	2C	2D
1	314	198	207	188
2	189	175	120	75
3	476	285	356	196
4	689	413	568	235
5	[illegible]14	1524	1492	447

TABLE 6.3.10 - $M = N - 1$, $C_N = 0, \Delta x = 0.001$

The method suggested in Chapter 4.3, Pages 50-51 of fitting several, say K, models at each iteration and choosing the one which produces the best minimum, was tried for values of K from 2 to 5. As K increases, the number of function evaluations increases because at each iteration K minima are tested for a decrease in function value. The computational work involved in solving the sets of equations also increases greatly for the reason that K sets of equations are being solved although the number of iterations in most cases decreases as the result of a better point being found at each iteration. Even when K = 2 the number of function evaluations and the amount of computational work are so great as to render the method quite impractical.

C H A P T E R   7

C O N C L U S I O N

CHAPTER 7 :

CONCLUSION

The results of the previous Chapter show that the Grid-to-Grid Methods usually attain their optimal data structure size when the grids contain the exact number of points necessary to fit a certain model to a given objective function.

For the overdetermined data structure, it was thought that an increase in data structure size would bring about a better approximation to the given objective function and eventually lead to a decrease in the number of function evaluations. In fact, as M increases, the least-squares model does give a better approximation to the given function and a better search direction. The problem, however, is that at each iteration we seek a least-squares solution using a data structure of the same size. Even when a better point is found and included in the grid, M - 1 points of the previous grid still remain in the new grid and as a result the new least-squares model will not differ much from the previous one. The stepsize therefore becomes smaller bringing about an increase in the total number of function evaluations.

In the underdetermined case, on the other hand, the search direction is rarely a good one and therefore the increase in function evaluations for this case is much more rapid than in the overdetermined case. When $M < N$ there are an infinite number of solutions and the criterion that was used to find a unique solution was that of minimum norm. This solution will, more often than not, provide a model which is not at all a good approximation to the given objective function and lead to a search direction which might be entirely erroneous. In fact, the very large number of function evaluations comes about mainly because of the many times Armijo's Rule is used as a result of a failure of the model to produce a good search direction.

In order to understand this phenomenon more fully, we shall discuss the different methods separately while noting that a minimum norm solution of Ax = b will produce the "smallest possible" x under the given constraints of the equations:

(i) In Method 2A the x's are the elements of the inverse of the Hessian matrix and if these are always small, the stepsize will be small and the search direction will not necessarily be a good one.

(ii) In Methods 2B and 2C the x's are the actual independent variables of the objective function. This means that these methods are actually drawing the search towards the origin of the axes instead of in the direction of the minimum. Indeed, the reason for the exceptionally good results of Method 2C on Test Function 4 (See Table 6.3.8) is that this function has its minimum at the origin.

(iii) In Method 2D the x's are the elements of the Hessian matrix. Since the search direction and stepsize depend on the increase of the Hessian matrix, a small Hessian matrix usually results in a large inverse, bringing about a bad search direction.

Even though it seems that the optimal data structure size occurs when M = N we do not think that research in this direction should stop at this point. While it must be admitted that the underdetermined data structure does not seem to be very promising unless a suitable criterion for a unique solution can be found, the overdetermined structure's results could perhaps be improved if a way could be found to make the data structure size more flexible. For example, if a good search direction and hence a better point is found, the data structure size should be decreased to allow the updated model to determine another good direction. It is in this area that we suggest that further research be directed.

REFERENCES

1. Jacobson, D.H. and Oksman, W. - An Algorithm that Minimizes Homogeneous Functions of N variables in N+2 iterations and rapidly Minimizes General Functions - Journal of Mathematical Analysis and Applications, 38 (1972), 535.

2. Davison, E.J. and Wong, P. - A robust algorithm that minimizes L-Functions in a finite number of steps and rapidly minimizes general functions, assuming the derivatives are not available - Control System Report No. 7313, University of Toronto, September 1973.

3. Sugie, N. - An extension of Fibonaccian searching to multidimensional cases - I.E.E.E. Transactions on Automatic Control, AC-9 (1964), 105.

4. Box, G.E.P. - Evolutionary Operation : a method for increasing industrial productivity - Applied Statistics, 2 (1957), 81.

5. Spendley, W., Hext, G.R. and Himsworth, F.R. - Sequential application of simplex designs in optimization and evolutionary operation - Technometrics, 4 (1962), 441.

6. Hooke, R. and Jeeves, T.A. - 'Direct Search' Solution of Numerical and Statistical Problems - Journal of A.C.M., 8 (1961), 212.

7. Rosenbrock, H.H. - An automatic method for finding the greatest or least value of a function - The Computer Journal, 3 (1960), 175.

8. Swann, W.H. - Report on the development of a New Direct Search Method of Optimization - Imperial Chemical Industries Ltd., Central Instrument Laboratory Rese ch Note 64/3 (1964).

9. Powell, M.J.D. - An efficient method of finding the minimum of a function of several variables without calculating derivatives - The Computer Journal, 7 (1964), 155.

10. Stewart, G.W. - A modification of Davidon's minimization method to accept difference approximations for derivatives - Journal of the A.C.M., 14 (1967), 72.

11. Davidon, W.C. - Variable metric method for minimization - A.E.C. Research and Development Report ANL - 5990 (Rev.), (1959).

12. Greenstadt, J. - On the relative efficiencies of Gradient Methods - Maths. of Computation, 21, (1967), 360.

13. Goldfield, S.M., Quandt, R.E. and Trotter, H.F. - Maximization by Quadratic Hill Climbing - Econometrica, 34 (1966), 541.

14. Himmelblau, D. - Applied Nonlinear Programming - Mc Graw-Hill, 1972.

15. Broyden, C.G. - Quasi-Newton methods and their applications to function minimization - Math. of Computation, 21 (1967), 368.

16. Fletcher, R. and Powell, M.J.D. - A rapidly convergent descent method for minimization - Computer J., 6 (1963), 163.

17. Fletcher, R. and Reeves, C.M. - Function minimization by conjugate gradients - Computer J., 7 (1964), 149.

18. Zoutendijk, G. - Methods of Feasible Directions - American Elsevier Co., 1960.

19. Fiacco, A.V. and McCormick, G.P. - Nonlinear Programming : Sequential Unconstrained Minimization Techniques - John Wiley and Sons, 1968.

20. Winfield, D. - Function Minimization by Interpolation in a Data Table - J. Inst. Maths. Applic.., 12 (1973), 339.

21. Jacobson, D.H. and Pels, L.M. - A Modified Homogeneous Algorithm for Function Minimization - J. Math. Anal. and Appl., 46 (1974), 533.

22. Armijo, L. - Minimization of functions having continuous partial derivatives - Pacific J. Math., 16 (1966), 1.

23. Kowalik, J.S. and Ramakrishnan, K.G. - A Numerically Stable Optimization Method based on a Homogeneous Function - Submitted to Math. Programming in May 1975.

24. Botsaris, C.A. - Differential Descent Methods for Function Minimization - Ph.D. Thesis submitted to University of the Witwatersrand, 1975.

25. Curry, H.B. - The method of steepest descent for nonlinear minimization problems - Quart. Appl. Math., 2 (1944), 258.

26. Goldstein, A.A. - Cauchy's method of minimization - Numer. Math., 4 (1962), 146.

27. Polak, E. - Computational Methods in Optimization - Academic Press, 1971.

28. Penrose, R. - On Best Approximate Solutions of Linear Matrix Equations - Proc. Cambridge Philos. Soc., 52 (1956), 17.

29. Golub, J. and Kahan, W. - Calculating the singular values and pseudoinverse of a matrix - SIAM J. Numer. Anal. Ser. B.2 (1965), 205.

30. Rao, C.R. and Mitra, S.K. - Generalised inverse of matrices and its Applications - John Wiley and Sons, 1971.

31. Penrose, R. - A generalized inverse for matrices - Proc. Cambridge Philos. Soc., 51 (1955), 406.

A P P E N D I C E S

APPENDIX A - Generalised Inverses

Let A be a rectangular m x n matrix in the following:

1. Definition - A generalised inverse of A is a matrix A^g of order n x m such that

$$A\,A^g\,A = A \tag{A.1}$$

The generalised inverse is not unique.

2. Definition - A minimum norm inverse of A is a matrix A^m of order n x m such that

$$A\,A^m\,A = A \tag{A.2}$$

and
$$(A^m\,A)^T = A^m\,A \tag{A.3}$$

The minimum norm inverse is not unique.

3. Definition - A pseudoinverse of A is a matrix A^p of order nxm such that

$$A\,A^p\,A = A \tag{A.4}$$

$$(A^p\,A)^T = A^p\,A \tag{A.5}$$

$$(A\,A^p)^T = A\,A^p \tag{A.6}$$

$$A^p\,A\,A^p = A^p \tag{A.7}$$

The pseudoinverse is unique.

The above definitions can be used to find solutions to the set of linear equations given by

$$A\,x = b,\ A \in R^{m \times n},\ x \in R^n,\ b \in R^m \tag{A.8}$$

where Rank (A) = k.

The different possibilities are as follows:

(i) If $m > n$ and the system of equation is inconsistent, there is no solution. Usually, however, in this case, a best approximate or least squares solution is used. This is defined by the point x^* such that

$$\| A x^* - b\| \leq \| A x - b\| \quad \text{for all } x \in R^n$$

Then the solution is

$$x^* = A^p b \qquad (A.9)$$

In the special case where $k = n$ this means that $A^T A$ is non singular and equation (A.9) reduces to

$$x^* = (A^T A)^{-1} A^T b \qquad (A.10)$$

(ii) If $m \leq n$ or the system of equations is consistent, then the general solution to (A.8) is

$$x^* = A^g b + (I - A^g A)y \qquad (A.11)$$

where y is an arbitrary vector in R^n.

This solution is not unique and unique solutions are obtained in the following ways:

a) If $k = n$ (this is only possible if $m = n$) the solution becomes

$$x^* = A^{-1} b \qquad (A.12)$$

b) If $m < n$ and $k = m$, the solution is

$$x^* = A^T (A A^T)^{-1} b \qquad (A.13)$$

because in this case $A A^T$ is non singular.

c) If $m < n$ and $k < m$ there are an infinite number of solutions and usually the minimum norm solution is chosen. This is unique and is defined by a point x^* such that

$$\|A x^* - b\| = \|A x - b\| \text{ and } \| x^*\| \leq \| x\| \text{ for all}$$
$$x^* \in R^n \qquad (A.14)$$

The solution to (A.14) is

$$x^* = A^m b \qquad (A.15)$$

Note that A^p is a special case of A^m and, therefore equation (A.9) is also a solution of (A.14). Also, if k = n as in section a), we have

$$A^p = A^m = A^{-1}$$

which makes equation (A.12) a particular case of (A.9) or (A.15).

For further details of this material see Rao and Mitra [30] and Penrose [28] and [31].

APPENDIX B - Existing Interpolation Methods

B.1 Jacobson and Pels [21]

Step 1. Assume x_o, η_1, η_2, N given

Set $\gamma_o = 2$, $\omega_o = 0$, $i = 0$

2. Compute $p_o \overset{\Delta}{=} -g(x_o)$ and use Armijo's Sub-procedure to calculate δ_o.

3. Set $x_1 = x_o + \delta_o p_o$

4. Set $\alpha_o^T = [x_1^T, \gamma_o, \omega_o]$, $P_o = I$, $j = 1$.

5. If $\| g(x_{i+1}) \| = 0$, stop; else go to 6.

6. Calculate:

$$y_{i+1} = \begin{bmatrix} g(x_{i+1}) \\ f(x_{i+1}) \\ -1 \end{bmatrix}$$

$$v_{i+1} = x_{i+1}^T g(x_{i+1})$$

If $|y_{i+1}^T P_i e_j| < \eta_1$, set $x_o = x_{i+1}$ and go to 1; else calculate:

$$P_{i+1} = P_i - \frac{P_i e_j (y_{i+1}^T P_i - e_j^T}{y_{i+1}^T P_i e_j}$$

$$\alpha_{i+1} = \alpha_i + \frac{P_i . e_j (v_{i+1} - y_{i+1}^T \alpha_i)}{y_{i+1}^T P_i e_j}$$

and go to 7.

7. Set $i = i+1$; if $j = n+2$ reset $j = 1$; else set $j = j+1$

8. If $|(x_i - \beta_i)^T g(x_i)| < \eta_2$, set $x_o = x_1$ and go to 1; else go to 9.

9. Set $p_i = \sigma_i(x_i + \beta_i)$ where
$\sigma_i = -\text{sign}\,[(x_i + \beta_i)^T g(x_i)]$

10. If $\|p_i\| + \|\gamma_i\| \leq N$, use Armijo's Subprocedure to calculate δ_i; else set $x_o = x_i$ and go to 1.

11. Set $x_{i+1} = x_i + \delta_i p_i$; go to 5.

In the above steps:

$$\alpha \triangleq [\beta, \gamma, \omega]$$

$$P_{i+1}^{-1} = P_i^{-1} + e_j (y_{i+1}^T - e_j P_i^{-1})$$

$$V_{i+1} = V_i + e_j (v_{i+1} - e_j^T V_i), \; V_o = \alpha_o$$

<u>Armijo Subprocedure</u>

```
Step 1.  Set $k(x_i) = 0$, $\delta_i(k(x_i)) = 1$
     2.  Calculate $\Delta f \triangleq f(x_i + \delta_i(k(x_i))p_i) - f(x_i)$
     3.  If $\Delta f + \frac{\delta_i(k(x_i))}{|\gamma_i| + 2} |p_i^T g(x_i)| \leq 0$
         set $\delta_i = \delta_i(k(x_i))$ and return;
         else set $k(x_i) = k(x_i) + 1$ and go to 4.
     4.  Set $\delta_i(k(x_i)) = \delta_i(k(x_i))/2k(x_i)$ and go to 2.
```

B.2 Winfield [20]

Step 1. Evaluate $f(x_i)$ at an initial grid of points x_i, $i = 1, \ldots\ldots, N$ where $N = \frac{1}{2}(n+1)(n+2)$

2. Let the basepoint x_b be the point of the initial grid at which $f(x_i)$ is lowest.

3. Define co-ordinates relative to the basepoint $y_j = x_y - x_b$ and order the points by increasing Euclidean distance from the basepoint. Let the subscript denote this ordering, with the origin $y_1 = 0$ being the basepoint and y_N being the point farthest from the basepoint.

4. Compute A, b, d so that
$\frac{1}{2}y_j^T A\, y_j + b^T y_j + d = f(x_j)$, $j = 1, \ldots, N$

5. Define a region of validity R of the quadratic model
$q(y) = \frac{1}{2}y^T A\, y + b^T y + d$
For the first model, and after every success in locating a new basepoint, let R be a sphere of radius $0.99\|y_N\|$ centred at the basepoint.

6. Choose the next trial point to be the y in R which minimizes $q(y)$.

7. At the minimizing y, compute
$x = x_b + y$
and evaluate $f(x)$.

8. If $f(x) \leq f(x_b)$, then x becomes a new basepoint. If $f(x) > f(x_b)$, then retain x_b as basepoint and reduce the volume of R by the factor $(\frac{1}{4}).2^n$.

9. The original data x_i, $f(x_i)$, $i = 1,2,\ldots, N$ plus the new point x and f(x) are kept in a data table having a capacity greater than the N data required to form a quadratic model. If x becomes a basepoint, all data in the table are re-ordered by Euclidean distance from this new basepoint. If x does not become a new basepoint, all data including the new x, f(x) are re-ordered by distance from the old basepoint. In the new ordering, x necessarily becomes one of the N-1 points closest to x_b. This is because x is in the R used in Step 6, and all points in that R are closer to x_b than the point, defined in Step 3, which played the role of x_N before the new ordering.

10. Let the basepoint and the N-1 points nearest it be designated "active points". Let the newly ordered data x_j, $f(x_j)$, $j = 1,2,\ldots N$, with x_1 the basepoint and x_N the most distant active point, be designated "active data". Then as a result of each evaluation of f(x), at least one member x_i, $f(x_i)$ of the set of active data is changed, and the most recent x, f(x) is included in the active data.

11. With a new set of active data, go to Step 3 and repeat. The computations of Step 3 through 11 constitute one SQM (Sequential Quadratic Models) cycle.

12. When f(x) or the radius of R is reduced below specified values, or some specified number of f(x) evaluations is exceeded, stop the program.

APPENDIX C - Additional Tables of Numerical Results

FUNCTIONS \ M	M = 1	M = N-2	M = N-1	M = N	M = N+1	M = N+2	M = 2N
1	655	853	773	299	312	324	352
2	650	791	588	191	184	215	195
3	1503	784	832	329	341	318	396
4	1312	613	755	709	684	714	755
5	2243	1891	1835	1810	1751	1856	1777

TABLE C.1 - METHOD 2A, $\Delta x = 100$

FUNCTIONS \ M	M = 1	M = N-2	M = N-1	M = N	M = N+1	M = N+2	M = 2N
1	951	834	294	273	265	306	397
2	1623	604	304	174	153	189	263
3	985	818	516	220	241	208	457
4	1625	934	687	491	531	556	643
5	2563	1037	1226	1098	1325	1724	2049

TABLE C.2 - METHOD 2B, $\Delta x = 100$

M / FUNCTIONS	M = 1	M = N-2	M = N-1	M = N	M = N+1	M = N+2	M = 2N
1	1124	865	435	279	291	304	346
2	883	794	461	149	197	208	243
3	2184	1121	624	348	324	398	446
4	43	200	885	790	859	756	905
5	2564	1987	976	1029	1423	1506	1704

TABLE C.3 - METHOD 2C, $\Delta x = 100$

M / FUNCTIONS	M = 1	M = N-2	M = N-1	M = N	M = N+1	M = N+2	M = 2N
1	231	173	127	71	82	97	123
2	198	126	173	58	66	73	89
3	325	233	264	143	162	125	188
4	267	151	128	106	94	144	189
5	397	324	256	193	212	184	237

TABLE C.4 - METHOD 2D, Δx = 100

METHODS / FUNCTIONS	2A	2B	2C	2D
1	237	245	193	187
2	134	212	108	79
3	415	338	407	185
4	586	437	603	267
5	2324	1846	1641	483

TABLE C.5 - $M = N-1$, $C_N = 0$, $\Delta x = 100$