ON THE USE OF FUNCTION VALUES TO IMPROVE QUASI-NEWTON METHODS

By:

R.A. Ghandhari

A Thesis presented to

The School of Mathematical and Computer Sciences

(Department of Computer Science)

University of Essex

Colchester

1989

In fulfillment of the Requirements for

the Degree of Doctor of Philosophy

To my children, for whom I live

CONTENTS

Acknowledgments

Preface

Symbols, Notation and Abbreviations

Chapter 1 Introduction.....1

1-1 Definitions and Terminology.....1

1-2 The Statement of the Problem.....3

1-3 Brief Summary of Existing Methods.....6

1-4 General Algorithm for Gradient Methods.....10

1-5 Basic Gradient Methods.....12

1-5-1 Method of Steepest Descent.....12

1-5-2 Newton's Method.....13

1-5-2-1 Modified Newton Methods.....15

1-6 Conjugate Direction Methods.....17

1-6-1 The Conjugate Gradient Method.....22

1-6-1-1 Properties of Conjugate Gradient Methods...34

Chapter 2 Quasi-Newton Methods.....36

2-1 Derivation of the Secant Equation.....41

2-2 Generation of Formulas.....43

2-2-1 Rank-1 Methods.....45

2-2-2 Rank-2 Methods.....50

2-2-2-1 The Broyden Family.....50

2-2-2-2 Powell-Symmetric-Broyden (PSB) Update.....60
2-3 Other Formulas.....63
2-3-1 Huang Family.....63
2-3-2 Variants of Quasi-Newton Methods.....64
2-3-2-1 Biggs's Method.....64
2-3-2-2 Davidon and Sorensen Methods.....65
2-3-2-3 The Methods of Ford and Saadallah.....67
2-4 Variable Metric Methods and Scaling.....71
2-4-1 Variable Metric Methods.....71
2-4-2 Motivation for Scaling.....73
2-4-3 Scaling of Variable Metric Methods.....75
2-5 Combination of Conjugate Gradient and Quasi-Newton Methods.....79

Chapter 3 Implementing the Newton and Quasi-Newton Methods..86
3-1 Line-Search Techniques.....89
3-2 Trust-Region Techniques.....97
3-2-1 Adjusting the Trust-radius.....100
3-2-2 Characterizing s.....102
3-3 Convergence Properties.....109
3-4 Termination Conditions.....115

Chapter 4 Nonlinear Models for the Gradient.....117
4-1 Design of a Nonlinear Parametric Model.....120
4-2 Determination of the Parameter(s).....123

4-2-1 Function Values.....123
4-2-2 Use of Curvature Estimates.....124
4-2-3 Specialized Models.....125
4-3 Derivation of w.....127
4-4 Properties of Class II.....128
4-5 Estimation of Curvature at x and x^*.....139
4-6 Nomenclature for Algorithms.....144

Chapter 5 Exponential Models.....147
5-1 The Model A.....148
5-2 The General Class II with Curvature Estimate at x^*
(Algorithm J).....156
5-3 The Model B.....158
5-4 The Model C.....165
5-5 The Model D.....168
5-6 Computational Considerations and Lemmas.....171

Chapter 6 Alternative Models.....180
6-1 The Model E.....182
6-2 The Model F.....188
6-3 The Model G (Algorithm Gfc0).....191
6-4 The Model H (Algorithm Hfc0).....194

Chapter 7 Numerical Results and Conclusions.....198
7-1 The Test Problems.....198
7-2 Program Implementation.....214

7-3 Numerical Results and Discussion.....215
7-3-1 Criteria for Comparison.....216
7-3-2 Statistics and Diagrams for All Algorithms.....217
7-3-3 Selecting the Best Algorithms.....255
7-3-4 The Best Algorithms Overall.....265
7-3-5 Numerical Study of Convergence.....277
7-4 Summary and Conclusions.....283
7-5 Suggestions for Further Research.....285

References.....288

ACKNOWLEDGMENTS

I would like to express my deepest gratitude and thanks to my supervisor, Dr. John A. Ford, for his sound guidance, valuable assistance, cooperation and constant encouragement through this work. Working with him was an invaluable experience. He has been a source of instruction in computational science and academic discipline.

My discussions with Mr. W.E. Hart and Professor J. Oliver were very useful and effective. I would like to thank them sincerely.

Finally, I would like to state my thanks to my friends Mr. Moodi, Mr. Djalali and Mr. Gholami for their encouragement, support and help.

ON THE USE OF FUNCTION VALUES TO IMPROVE QUASI-NEWTON METHODS

R.A. Ghandhari

Abstract

This thesis presents new techniques for quasi-Newton methods in unconstrained optimization. We investigate the use of nonlinear models for the gradient of the objective function, in contrast to most existing methods, which are (essentially) founded upon a linear model.

The key to the methods consists of the determination of one or two parameters in the nonlinear models, by suitable means, and then using this information to construct an approximation (similar in form to the 'secant' equation) to the so-called 'Newton' equation. We achieve this by using (a) values of the objective function and/or (b) estimates of the curvature of the objective function. For some models, the parameter is obtained by solving a simple nonlinear equation in one variable, while, for others, we do not need to solve any equation and the parameter is obtained explicitly.

In many of the new techniques, the condition number of the approximate Hessian matrix can be controlled in relation to that of the matrices generated by standard quasi-Newton techniques. We can also guarantee to maintain the positive-definiteness of the matrices in every iteration. {This, in turn, ensures that the search-direction is downhill (in the case of using a line-search) or that the quadratic model possesses a minimum (in the case of utilizing a trust-region technique).}

The numerical performance of the new techniques has been evaluated using a standard quasi-Newton method (the 'BFGS' algorithm) as a basis for comparison. Using this assessment, the new methods may be viewed as acceleration techniques for quasi-Newton methods.

Preface

The subject of optimization (maximization or minimization) is a combination of theory and experiment, and has applications in almost every branch of technology and industry. The function may have to be optimized under certain constraints or the problem may be unconstrained. Methods of optimization may be classified as derivative (or gradient) methods and non-derivative methods.

We assume that the n-variable function to be optimized and its first partial derivatives are all continuous. In this work, we intend to examine derivative methods for unconstrained problems and will seek to improve the numerical performance of one well-known and widely-used class of methods, namely, the "quasi-Newton" algorithms.

The material of this thesis is organized into seven chapters. The first three chapters are devoted to a survey of existing methods for the unconstrained problem while, in the remainder of the thesis, we investigate new techniques for improving quasi-Newton methods.

In **chapter 1**, after giving some basic definitions, we discuss the derivation and properties of fundamental algorithms in this area, such as Steepest Descent, Newton's method and the Conjugate Direction methods.

In **chapter 2**, besides the derivation of the 'secant' equation (which constitutes the basis of quasi-Newton methods), we discuss a number of well-known update formulas. It is shown that the

'secant' equation is based on a linear model for the gradient.

To find a better estimate for the solution, every quasi-Newton method usually requires the utilization of either a line-search or a trust-region technique at every iteration. **Chapter 3** surveys these techniques, the convergence properties of quasi-Newton methods and termination conditions appropriate to a gradient method.

In **chapter 4**, we investigate the design of nonlinear parametric models for the gradient and different approaches to the determination of the parameter(s) in the models. This leads to a number of new algorithms which may be utilized in place of the standard quasi-Newton methods.

The new algorithms are classified into two groups (exponential and non-exponential models) and are examined in **chapters 5** and **6**, respectively. We show that all of the new algorithms may be expected to tend to the standard quasi-Newton algorithms as the minimum is approached. The condition number of matrices of the new algorithms are controlled in relation to that of a standard quasi-Newton algorithm. We also ensure the maintenance of positive-definiteness of the Hessian approximation, which property plays a very important role in the convergence of the algorithms.

In **chapter 7**, we present numerical results for the new algorithms described in chapter 5 and 6. In order to assess the results, we have compared the new algorithms with two existing

algorithms (one of which is the 'BFGS' algorithm (one of the best known standard quasi-Newton methods to date)). We conclude with a discussion of the numerical results and some suggestions for further research.

Symbols, Notation and Abbreviations

Superscript "T" : Transposition

z : The solution of the minimization problem

x_k : k-th estimate of z

$\|.\|_2$: The Euclidean norm of a vector, that is

$$\|x\|_2 = \left\{\sum_{i=1}^{n} x_i^2\right\}^{1/2}$$

F(x) or F : Objective function

subscript 'k'
for any symbol : The symbol at iteration k or at the current iteration

subscript 'k+1'
or superscript '*'
for any symbol : The symbol at iteration k+1

g(x) : The gradient of F(x), that is

$$[g(x)]^T = [\partial F/\partial x_1,\ \partial F/\partial x_2,\ \ldots,\ \partial F/\partial x_n]$$

s or s_k : $x_{k+1} - x_k$

y or y_k : $g_{k+1} - g_k$

G(x) or G : The n by n Hessian matrix of F(x), whose (i,j)-th

element is given by $\partial^2 F(x)/\partial x_i \partial x_j$

B, B_k, B^*	: Approximations to the Hessian matrix
H, H_k, H^*	: Approximations to the inverse Hessian matrix
R	: The set of real numbers
p or p_k	: The search direction on iteration k
$t_k^\#$ or $t^\#$	: The step length on iteration k
pd	: Positive-definite
psd	: Positive-semi-definite
CG	: Conjugate gradient
QN	: Quasi-Newton
$\{x_k\}$	: The sequence $x_0, x_1, x_2, \ldots$

CHAPTER 1

INTRODUCTION

1-1 Definitions and Terminology

Before stating the problem we are going to consider, we need some definitions.

Definition: "Open Neighbourhood"

$x^$ is in the open neighbourhood of x with radius r (> 0) if and only if $\|x^ - x\|_2 < r$. We denote this open neighbourhood by $N(x,r)$.

Definition: "Local Minimum"

A point $z \in R^n$ is said to be a local minimum of F if there exists an open neighbourhood of z such that, for any x in this open neighbourhood, $F(z) \leq F(x)$ holds.

Definition: "Global Minimum"

A point $z \in R^n$ is said to be a global minimum of F if, for any $x \in R^n$, $F(z) \leq F(x)$ holds.

A point $z \in R^n$ is said to be a strict local or strict global minimum of F, if strict inequality in the last definitions holds, as appropriate, whenever $x \neq z$.

Definition: "Lipschitz Continuous"

Let $G \in R^{n\times n}$, $x \in R^n$ and $\|.\|$ be a norm on R^n. G is said to be Lipschitz continuous at x if there exists an open set $D \subset R^n$ containing x and a constant L such that, for all $y \in D$,

(1-1) $\|G(x) - G(y)\| \leq L\|x - y\|$.

Usually $\|.\|$ is the Euclidean norm although other norms may be used. The constant L is called a Lipschitz constant for G at x. For any specific D containing x for which (1-1) holds, G is said to be Lipschitz continuous at x in the set D. If (1-1) holds for every $x \in D$, then we write $G \in Lip_L(D)$.

Rates of Convergence (Ortega and Rheinboldt(1970))

A sequence of points $\{x_k\}$ is said to converge to z if

$$\lim_{k\to\infty} \|x_k - z\| = 0.$$

The rate of convergence essentially conveys information about how quickly $\|x_k - z\|$ decreases. The rates for different iterative processes provide a precise means for comparing them and hence a criterion for determining which of them is "faster". Suppose $\{x_k\}$ is a sequence that converges to z. If constants $p\ (>0)$ and $c\ (>0)$ exist such that

(1-2) $$\lim_{k\to\infty} \{\|x_{k+i} - z\|/\|x_k - z\|^p\} = c,$$

then $\{x_k\}$ is said to converge to z with (i-step) q-order p and asymptotic constant c. The constant c affects the speed of convergence, but it is not as important as the order. In the case

of i = 1, the convergence is called q-linear, q-quadratic and q-cubic for p = 1, 2 and 3, respectively. For general i, the convergence is called i-step q-linear, i-step q-quadratic and i-step q-cubic for p = 1, 2 and 3, respectively. For p = 1, c must be less than one to have convergence. If

$$(1\text{-}3) \qquad \lim_{k \to \infty} \{\|x_{k+i} - z\| / \|x_k - z\|\} \leq c_k$$

where $c_k \to 0$, the convergence is called i-step q-superlinear (and q-superlinear, if i = 1).

1-2 The Statement of the Problem

The problem we discuss is the determination of a local minimum of a nonlinear continuous objective function F(x). We may write the problem in the following mathematical form:

$$(1\text{-}4) \qquad \min F(x),$$

where $F : R^n \to R$.

We shall focus on the problem of locating a local minimum z, but if F is convex, it is known that a local minimum is also the global minimum as well (see, for instance, Luenberger(1984)). In general, finding the global minimum of a nonlinear function with more than one local minimum is very difficult. However, the possibility of finding a global minimum has been considered by many authors: for example, Dixon, Gomulka and Szego(1975) and Dixon and Patel(1981). From this point on, we will use the term

"minimum" in place of "local minimum" when there is no danger of ambiguity.

We propose to confine our discussion to the case when there are no restrictions on the choice of value of x (so-called unconstrained problems). However, in many problems, practical considerations place constraints or limitations on the range of permissible values for x. Methods for unconstrained problems cannot cope directly with these limitations, but there are a number of reformulations which may, in certain cases, be used to convert a constrained problem into an unconstrained one (see, for instance, Fletcher(1987)). The structure of most constrained optimization problems is essentially contained in the following:

$$\min F(x)$$

$$\text{subject to} \quad c_i(x) = 0 \quad i \in E,$$

$$c_j(x) \geq 0 \quad j \in I,$$

where E and I are the index sets of equality constraints and inequality constraints, respectively. The above formulation does not cover problems with discrete values. A problem with discrete variables may be solved by integer programming techniques (for example, branch and bound methods). The subject of constrained optimization splits into two parts, linearly-constrained programming and nonlinear programming. In linearly-constrained programming, each constraint is a linear function. The simplest cases are when the objective function is either linear or quadratic (linear programming or quadratic programming), for such problems can be solved in a finite number of steps. The most

difficult type of constrained minimization problem is nonlinear programming with nonlinear constraint functions.

If the constraints can be reformulated so that they can be eliminated directly, then this should be done, but this is not usually possible. A different approach is to convert the problem into a single unconstrained problem or a sequence of unconstrained problems. Basically (in this category), there are two approaches. The first is called the Penalty Function method, in which a penalty term is added to the objective function for any violation of the constraints and by this means the problem is converted to a (number of) unconstrained problem(s). In other words the Penalty function method seeks the unconstrained minimum of a new objective function defined by the following problem

$$\min \quad U(x) \equiv F(x) + r.P(x)$$

where $F(x)$ is the main objective function and r is a monotonically increasing scale factor which controls the actual effect of the constraint violations. $P(x)$ should be (i) a continuous function on R^n, (ii) non-negative for all $x \in R^n$ and (iii) zero if and only if x is in the constrained region. Essentially, the method consists of performing unconstrained minimizations of U for a sequence of increasing values of r.

The second method is called the Barrier Function method, in which a barrier term that prevents the points generated from leaving the feasible region is added to the objective function.

Conditions for a Minimum

Definition: "Positive-definite (pd) Matrix

A square matrix A is said to be positive-definite (pd) if A is symmetric and $x^T Ax > 0 \ \forall \ x \neq 0$.

Definition: "Positive-semi-definite (psd) Matrix

A square matrix A is said to be positive-semi-definite (psd) if A is symmetric and $x^T Ax \geq 0 \ \forall \ x$.

The conditions for the point z to be a minimum of the function F(x) are (Fletcher(1987)):

(a) Necessary conditions :

g(z) = 0;

G(z) psd;

(b) Sufficient conditions :

g(z) = 0;

G(z) pd.

1-3 Brief Summary of Existing Methods

To determine the minimum of a nonlinear function that is differentiable and for which the partial derivatives may readily be computed, there are two major classes of algorithms:

(a) First derivative methods, which employ the first partial derivatives (only): the Steepest Descent, Conjugate-direction and

quasi-Newton methods are in this class;

(b) Second derivative methods, which utilize the first and second partial derivatives: Newton's method and the modified Newton methods are in this class.

Together, these two classes comprise what are termed "gradient methods". To obtain a minimum of a function that is continuous but where either the function is not differentiable or its derivatives are not readily available, there are two choices, as follows:

(a) to estimate first partial derivatives by finite-difference approximations and then apply one of the first derivative methods. Gill, Murray and Wright(1981), for example, have used this technique in a quasi-Newton method;

(b) to employ methods which are based on comparison of function values only. These methods are known as "Direct search methods" or "Function comparison methods". These methods should be used only when there is no other suitable alternative method available, because the required number of function evaluations tends to increase exponentially with the problem dimension.

An example of a direct search method is the Simplex method (see Spendley et al(1962), Nelder and Mead(1965) and Bunday and Garside(1987)). A set of n+1 equidistant points $(x_0, x_1, \ldots, x_n)$ in n-dimensional space is known as a regular simplex. Thus in two dimensions the simplex is an equilateral triangle and in three dimensions it is a regular tetrahedron. The idea of the method is to compare the values of the function at the n+1 vertices of the

simplex and move the simplex towards the optimum point during the iterative process. The original simplex method (by Spendley et al) maintained a regular simplex at each stage. Nelder and Mead proposed several modifications to the method which allow the simplices to become nonregular. Bunday and Garside claim that the Nelder and Mead algorithm is a very robust direct search method provided that the number of variables does not exceed five or six. In the following discussion we need some notation: we write F_i for the function value at x_i, and define h and l as the suffix such that

$$F_h = \max_i (F_i), \quad F_l = \min_i (F_i)$$

related to x_h and x_l, respectively. Further we define x^ as the centroid of the current points (excluding x_h).

At each stage in the process x_h is replaced by a new point; three operations are used- reflection, contraction and expansion. These are defined as follows: The result of the reflection of x_h is denoted by x_r (with F_r as its function value) and its coordinates are defined by the relation

$$x_r = (1 + \alpha)x^ - \alpha x_h,$$

where α is a positive constant, the reflection coefficient. If F_r lies between F_h and F_l, then x_h/F_h is replaced by x_r/F_r and we start again with the new simplex. If $F_r < F_l$, this means that the reflection has produced a new minimum, so we expand x_r to x_e (with function value F_e) by the relation

$$x_e = \gamma x_r + (1 - \gamma)x^,$$

where γ (> 1) is the expansion coefficient. If $F_e < F_l$ we replace

x_h by x_e and restart the process; but if $F_e > F_l$ then we have a failed expansion and we replace x_h by x_r before restarting. After reflection, we find that (i) $F_r > F_h$ or (ii) $F_r > F_i$ for all $i \neq h$. In the case of (i) we proceed directly to the contraction step, that is we find a new point x_c by the relation

$$x_c = \beta x_h + (1 - \beta) x\hat{},$$

where $\beta \in (0,1)$ is the contraction coefficient. In the case of (ii) we replace x_h by x_r (and F_h by F_r) and then find x_c using the last relation. We then accept x_c for x_h and restart, unless $F_c > \min \{F_h, F_r\}$, that is the contracted point is worse than the better of x_h and x_r. For such a failed contraction, we replace all the x_i's by $(x_i + x_l)/2$ (this operation may be called 'Shrinkage') and restart the process.

For termination of the search, the 'standard error' of the F_i's in the form

$$\sum_{i=0}^{n} (F_i - F_a)^2$$

where

$$F_a = \sum_{i=0}^{n} F_i \; /(n+1)$$

should fall below some predefined limit.

1-4 General Algorithm for Gradient Methods

Gradient algorithms for solving problem (1-4) are iterative, starting from an initial guess x_0 and generating a sequence of points $x_1, x_2, \ldots, x_k, \ldots$, where x_k is the k-th approximation to the minimum, and stopping when x_k is judged to be sufficiently close to a minimum z. The two commonly-used techniques for proceeding from x_k to x_{k+1} are known as "line-search" and "trust-region".

Definition: "Quadratic Termination"

An iterative method that minimizes a quadratic function in a finite number of iterations (N, say) is said to possess "quadratic termination".

Definition: "Descent Direction"

p is a descent direction from x for a given function F if the directional derivative of the function at x in this direction is negative; that is, $p^T g(x) < 0$.

When we want to move from x_k to x_{k+1} (for k = 0, 1, ...) by means of a line-search, we need two components: direction and step length. Each method has its own means of determining the direction, but we are interested in those methods which evaluate the direction (p_k) as follows:

(1-5) $p_k = -M_k g_k$,

where M_k is a matrix. (The particular way of defining the matrix essentially identifies the method.) Then, to locate a new point x_{k+1} we set (for some scalar $t_k^{\#}$)

$$x_{k+1} = x_k + t_k^{\#} p_k, \tag{1-6}$$

where x_{k+1} is required to satisfy certain conditions, of which

$$F(x_{k+1}) < F(x_k)$$

should be one. (Here and later in this text we assume that a monotonic line-search is applied (for the concept of a nonmonotonic line-search we refer to Grippo et al(1986).)) In theory, $t_k^{\#}$ is often chosen to minimize $F(x_k + tp_k)$ over all $t > 0$ (or, in the case of multiple local minima, to be that one with the smallest (positive) value). Since an infinite process will generally be needed to calculate such a value for $t_k^{\#}$, in practice $t_k^{\#}$ is selected according to less demanding criteria in order that a suitable value can be determined from the calculation of $F(x_k + tp_k)$ for only a few different values of t.

When we employ a trust-region technique at iteration k, we construct a quadratic model (with a unique minimizer) for $F(x_k + s)$. We then minimize this model to find a vector s that may be called the correction vector. The correction vector s will be added to x_k to arrive at x_{k+1}.

We will consider the line-search and trust-region techniques in more detail in chapter 3.

1-5 Basic Gradient Methods

The basic methods for generating the sequence x_1, x_2, ..., x_k, ... from an initial point x_0 are the Steepest Descent method and Newton's method. Both methods stem from the use of a Taylor series to approximate F(x) about the current point x_k (see, for example, Gill, Murray and Wright(1981)).

1-5-1 Method of Steepest Descent

Consider the linear approximation to F(x) based on the Taylor series expansion about x_k:

$$L(x) \equiv F(x_k) + g_k^T(x - x_k).$$

Assuming that a step of unity will be taken along p_k (that is, $p_k = s_k = x_{k+1} - x_k$ in this case), it would appear that a good way to achieve a large reduction in F is to choose p_k so that

$$g_k^T(x_{k+1} - x_k) = g_k^T p_k \qquad \text{(the reduction in L(x))}$$

is as large and negative as possible. Using the Cauchy-Schwarz inequality (see, for example, Johnson and Riess(1982)), we may conclude that the direction $p_k = -g_k$ causes F to decrease most rapidly as we move away from x_k. Thus $-g_k$ is the "steepest descent" direction and the method that utilizes this direction is known as the Steepest Descent method. It can be shown the method converges to a minimum or saddle-point of F (see Goldstein(1967)), but with only a linear rate of convergence (with a constant c_k in (1-3) often near to unity

(Luenberger(1984))). However, this algorithm may sometimes be used to find a sufficiently accurate starting approximation for Newton's method or others.

1-5-2 Newton's Method

The derivation of Newton's method is based on the assumption that the objective function being minimized can be approximated locally by a quadratic function: the method proceeds by minimizing this approximate function precisely. The quadratic approximation is obtained by taking the first three terms of the Taylor series about x_k of $F(x)$, that is

$$(1\text{-}7) \quad q(x) \cong F(x_k) + g_k^T(x - x_k) + 1/2\ (x - x_k)^T G_k (x - x_k),$$

where $q(x)$ is the resulting quadratic approximation and is minimized (if G_k is pd) at

$$(1\text{-}8) \quad x_{k+1} = x_k - (G_k)^{-1} g_k.$$

This is the fundamental relation defining Newton's method. Comparing (1-8) and (1-6) gives what is usually called the "Newton direction", that is

$$(1\text{-}9) \quad p_k = -G_k^{-1} g_k,$$

and $t_k^* = 1$ for Newton's method. To move "towards" the minimum, we need to know that the direction p_k defined by (1-9) is a descent direction: that is,

$$g_k^T p_k \ (= -g_k G_k^{-1} g_k) < 0$$

should hold. Positive definiteness of G_k^{-1} (or, equivalently, G_k) is sufficient to guarantee that this holds. Newton's method has one important advantage: if it converges to the solution, it usually has very quick local convergence (namely, q-quadratic) and it also may easily be seen to have the property of quadratic termination (with N = 1). Another advantage of Newton's method is that x_{k+1} only depends upon F_k and x_k, so that bad effects from previous iterations are not carried along. This is an advantage of Newton's method which is not shared by quasi-Newton methods (to be considered in the next chapter).

The following theorem (whose proof may be found in, for example, Fletcher(1987)) states the important property of Newton's method referred to above (namely, convergence of q-order two in certain circumstances).

Theorem 1-1

Assume that F(x) is twice continuously differentiable in R^n and that the Hessian matrix is Lipschitz continuous, that is,

$G \in Lip_L(D)$,

where D is a neighbourhood of a local minimizer z. If x_k is sufficiently close to z for some k, and if G(z) is positive definite, then Newton's method is well defined for all k, and converges with q-order two.

In contrast to these desirable properties, the method also

suffers from the following disadvantages:

1 - It requires the evaluation of the Hessian matrix G_k at each iteration, which can be very expensive (or it may be unavailable).

2 - At each step, the method requires the solution of a linear system

(1-10) $G_k p_k = -g_k$

to obtain p_k.

3 - When the initial estimate x_0 is poor, G_k may be non-positive definite. In this case, the direction p_k may then be an ascent direction. If the matrix G_k is singular or nearly singular, accurate solution of the linear system may be very difficult or impossible.

1-5-2-1 Modified Newton Methods

The very important advantage of Newton's method (second order convergence) has caused researchers to try to eliminate the disadvantages of the method while retaining its advantages as far as possible. For each of the disadvantages, there exist proposed remedies which we now briefly discuss. Such versions are called modified Newton methods.

Evaluation of the Hessian matrix at every iteration (especially for a complicated objective function) can be very expensive, as observed above. To deal with this disadvantage, there are different approaches, for example:

(a) to evaluate the matrix once and for all. This approach converges to the solution q-linearly (see Goldstein(1967)) and is not generally efficient, because the matrix is fixed for all iterations;

(b) to evaluate the matrix once every m iterations (say). This approach ultimately converges to the solution with q-order $(m+1)^{1/m}$ (see Fletcher(1987)). This approach can be effective for functions which have a slowly-changing Hessian, but deciding on a suitable value for m is difficult;

(c) to obtain the matrix by a finite-difference formula employing first derivatives. This approach is expensive, time-consuming and there is no guarantee that it will yield a positive definite matrix.

The solution of a linear system of equations to obtain the direction p_k at each iteration is another disadvantage. Solving the system in an inexact manner may remove this disadvantage to some extent. Dembo et al(1983) have utilized the conjugate gradient ("CG") method to solve the linear system only approximately. A rate of convergence between q-linear and q-quadratic has been shown for this algorithm, which has been called the "Truncated Newton algorithm" or "TNCG algorithm".

To maintain convergence, p_k must be a descent direction. For p_k to be a descent direction, $p_k^T G_k p_k$ must be positive. If $p_k^T G_k p_k$ is not positive, we have one of the following possibilities:

(a) $p_k^T G_k p_k$ is negative: in this case, to obtain a descent direction, we may reverse the direction p_k;

(b) $p_k^T G_k p_k$ is zero because of singularity of G_k or orthogonality of p_k and $G_k p_k$ (= $-g_k$). In this event we modify the Newton search direction (1-9) by giving it a bias towards the steepest descent direction $-g_k$. This may be done by adding a multiple of the unit matrix to G_k and solving (instead of (1-10)) the system

$$(G_k + \lambda I)p_k = -g_k,$$

where $\lambda \in R$. The idea of modifying matrices by the addition of a multiple of the unit matrix was first suggested by Levenberg(1944) and Marquardt(1963) in the context of nonlinear least-squares problems and by Goldfeld et al(1966) in the context of minimization. The trust-region technique (to be considered in chapter 3) is closely linked to this idea.

When G_k is pd but F_{k+1} is not (sufficiently) smaller than F_k, equation (1-8) may be modified in the following manner, which is called the "Relaxed Newton" method:

$$x_{k+1} = x_k - t_k^{\#} G_k^{-1} g_k, \qquad (1\text{-}11)$$

where $t_k^{\#}$ is chosen to ensure that F_{k+1} is (sufficiently) smaller than F_k.

1-6 Conjugate Direction Methods

The conjugate direction methods were originally developed by Hestenes and Stiefel(1952) for linear systems of equations

$$Gx = b, \text{ say,} \qquad (1\text{-}12)$$

where G is usually a pd matrix. Since the problem

(1-13) $\min F(x) \equiv 1/2\ x^T G x - b^T x + c,$

where the Hessian, G, is positive definite, reduces to the problem of solving (1-12), then Hestenes and Stiefel's approach may be viewed as a minimization algorithm for quadratic functions under the stated conditions. These methods are based on the concept of mutual conjugacy of search directions.

Definition

Given a symmetric matrix G of order n, a set of n-th order non-null vectors $p_0, p_1, \ldots, p_{n-1}$ is said to be "conjugate with respect to G" or "G-conjugate" if

(1-14) $p_i^T G p_j = 0$ for i, j = 0, 1, ..., n-1 and $i \neq j$.

Proposition:

If a set of vectors $\{p_i\}_{i=0}^{n-1}$ is G-conjugate and G is a pd matrix, then the set $\{p_i\}_{i=0}^{n-1}$ is linearly independent.

Proof : see, for example, Luenberger(1984).

In our study of these methods, we concentrate, to begin with, on the minimization of quadratic functions (of the form (1-13), say). Let G be an n by n pd matrix and $\{p_i\}_{i=0}^{n-1}$ be G-conjugate vectors. Then the set is linearly independent, which implies that the solution of (1-12) or problem (1-13) (z,say) may be expanded in terms of the p_i's:

$$(1\text{-}15) \qquad z = \sum_{i=0}^{n-1} t_i p_i,$$

where the scalars t_i may be obtained (by the property of conjugacy) from

$$(1\text{-}16) \qquad t_i = p_i^T b / p_i^T G p_i, \quad i = 0, 1, \ldots, n-1.$$

The basic conjugate direction method (assuming that a set of conjugate directions is available) is defined by taking an arbitrary initial point and obtaining t_i, at the i-th step, for $i = 0, 1, 2, \ldots, n-1$, by (1-16). The set of p_i's could be constructed from an arbitrary set of linearly independent directions $\{q_i\}$, say, by a Gram-Schmidt process (see, for example, Strang(1980)) as follows:

$$\text{put } p_0 = q_0;$$

$$(1\text{-}17) \qquad p_i = q_i + \sum_{j=0}^{i-1} c_{ij} p_j \quad \text{for } i = 1, 2, \ldots, n-1,$$

where the coefficients c_{ij}'s are chosen to make $p_i^T G p_j = 0$ for $i, j = 0, 1, 2, \ldots, n-1$ and $i \neq j$. In practice, such a process is not used because it needs explicit knowledge of G, while other methods that will be described below do not need G.

Theorem 1-2 (Luenberger(1984))

Let $\{p_i\}_{i=0}^{n-1}$ be a set of G-orthogonal vectors. For any $x_0 \in R^n$, the sequence $\{x_k\}$ generated according to

(1-18a) $$x_{k+1} = x_k + t_k p_k, \quad k = 0, 1, \ldots,$$

with

(1-18b) $$t_k = -g_k^T p_k / p_k^T G p_k$$

and

(1-18c) $$g_k = Gx_k - b,$$

converges to the unique solution z of $Gx = b$ after n steps (that is, $x_n = z$.)

Proof: Since the p_k's are linearly independent, for some set of β_k's we may write

$$z - x_0 = \sum_{k=0}^{n-1} \beta_k p_k .$$

Multiplying by G and taking the scalar product with p_k gives

(1-19) $$\beta_k = (p_k^T G(z - x_0)) / (p_k^T G p_k)$$
$$= (p_k^T G z - p_k^T G x_0) / (p_k^T G p_k).$$

Now following the iterative process from x_0 up to x_k gives

(1-20) $$x_k - x_0 = t_0 p_0 + t_1 p_1 + \ldots + t_{k-1} p_{k-1},$$

and, hence, multiplying by G and taking the scalar product with p_k gives

$$p_k^T G(x_k - x_0) = t_0 p_k^T G p_0 + \ldots + t_{k-1} p_k^T G p_{k-1} = 0,$$

so that

(1-21) $p_k^T G x_k = p_k^T G x_0$.

Substituting (1-21) into (1-19) gives

$$\beta_k = (p_k^T G(z - x_k))/(p_k^T G p_k).$$

On the other hand,

$$g_k = Gx_k - b = G(x_k - z)$$

which gives

$$\beta_k = -(p_k^T g_k)/(p_k^T G p_k) = t_k,$$

from (1-18b).■

In the following theorem it will be shown that, as the method of conjugate directions progresses, each x_k from (1-18a) minimizes the objective function over the k-dimensional linear variety

$$E_k = x_0 + D_k ,$$

where D_k is the subspace of R^k spanned by $\{p_0, p_1, \ldots, p_{k-1}\}$.

Theorem 1-3

Let $\{p_k\}_{k=0}^{n-1}$ be a sequence of conjugate vectors in R^n. Then, for any $x_0 \in R^n$, the sequence $\{x_k\}$ generated according to (1-18a) with exact line searches has the property that x_k minimizes F over the linear variety E_k.

Proof: Since G is a pd matrix, F is a strictly convex function and the result therefore follows if we show that g_k is orthogonal to D_k. The proof is by induction. For k = 0, the

result is true because D_0 is an empty set. Assuming that it is true for k, to prove that g_{k+1} is orthogonal to D_{k+1}, it is sufficient to show that

$$p_r^T g_{k+1} = 0 \quad \text{for } r = 0, 1, \ldots, k.$$

Differentiating (1-13) gives

(1-22) $\quad g(x) \equiv Gx - b.$

Writing (1-22) for x_k and x_{k+1}, subtracting them and then using (1-18a) yields

(1-23) $\quad g_{k+1} = g_k + t_k G p_k.$

Taking the scalar product with p_r (for r = 0, 1, ..., k-1) leads to

(1-24) $\quad p_r^T g_{k+1} = p_r^T g_k + t_k p_r^T G p_k \quad \text{for } r = 0, 1, \ldots, k-1.$

In this expression $p_r^T g_k$ vanishes because of the inductive hypothesis and $p_r^T G p_k$ vanishes because of the conjugacy of p_k's. Then

$$p_r^T g_{k+1} = 0 \text{ for } r = 0, 1, \ldots, k-1.$$

It remains to prove that $p_k^T g_{k+1}$ is zero. This is so in the direction p_k because of the exact line search.■

1-6-1 The Conjugate Gradient Method

The directions in the conjugate direction method that we have studied so far had to be specified beforehand, but this is not necessary for the conjugate gradient method. Instead, in the conjugate gradient method, the directions are determined at each

step of the iteration as the method progresses by employing the previous direction(s), the gradient at previous point(s) and the current point x_k. The following algorithm (due to Fletcher and Reeves(1964)) may be taken as the prototype for conjugate gradient methods.

(1-25a) $p_0 = -g_0$;

for k = 0, 1, 2,... do

(1-25b) $x_{k+1} = x_k + t_k p_k$,

where t_k is chosen to minimize $F(x_k + tp_k)$ along p_k.

(1-25c) $p_{k+1} = -g_{k+1} + b_k p_k$,

where

(1-25d) $b_k = g_{k+1}^T g_{k+1} / g_k^T g_k$.

Theorem 1-4

The Fletcher-Reeves method, when applied to the quadratic function in (1-13) with exact line-searches, terminates after $m \leq n$ iterations, and the following hold for all i satisfying $0 \leq i < m$:-

(1-26a) $p_i^T G p_j = 0$,

and

(1-26b) $g_i^T g_j = 0$,

both holding for j = 0, 1, 2, ..., i-1.

(These are referred to as the "conjugate direction" and

"orthogonal gradient" properties, respectively.)

Proof: If $g_0 = 0$, then m = 0, and (1-26) is vacuously true. If $g_0 \neq 0$, we prove the theorem by induction. For i = 0, (1-26) is again vacuously true. Let (1-26) hold for some value of i; we must prove the following:

(1-27a) $p_{i+1}^T G p_j = 0$

and

(1-27b) $g_{i+1}^T g_j = 0,$

(in both cases, for j = 0, 1, 2, ..., i)

First, we derive some expressions which we will need. Setting k = i in (1-23) gives

(1-28) $g_{i+1} = g_i + t_i G p_i .$

Multiplying (1-28) by p_i and using

(1-29) $g_{i+1}^T p_i = 0$

(which follows from the assumption of exact line-searches (see chapter 3)) leads to

(1-30) $t_i = -g_i^T p_i / p_i^T G p_i .$

From (1-25c) (with k = i-1), we have

$$p_i = -g_i + b_{i-1} p_{i-1},$$

which yields (on multiplying by g_i and using (1-29) with i replaced by i-1)

$$g_i^T p_i = -g_i^T g_i .$$

Substituting this value in (1-30) then gives

(1-31) $t_i = g_i^T g_i / (p_i^T G p_i) .$

To prove (1-27b), equation (1-25c) (with k = j-1) becomes

$$(1\text{-}32) \qquad g_j = -p_j + b_{j-1} p_{j-1}.$$

If we transpose (1-23) (with k = i) and then multiply it by g_j, we obtain

$$(1\text{-}33) \qquad g_{i+1}^T g_j = g_i^T g_j + t_i p_i^T G g_j,$$

which, by substituting for g_j from (1-32) in the second term of the right-hand-side, becomes

$$(1\text{-}34) \qquad g_{i+1}^T g_j = g_i^T g_j - t_i p_i^T G p_j + t_i b_{j-1} p_i^T G p_{j-1}.$$

When j = i, the first two terms of the right-hand-side cancel each other (from (1-31)) and the last term vanishes, by (1-26a). For j < i, the right-hand-side of (1-34) is zero because the first term is zero using (1-26b) and the remainder vanishes by (1-26a).

To prove (1-27a) we write (1-25c) with k = i and postmultiply by Gp_j. We thus obtain

$$\begin{aligned} p_{i+1}^T G p_j &= (-g_{i+1} + b_i p_i)^T G p_j \\ &= -g_{i+1}^T G p_j + b_i p_i^T G p_j. \end{aligned} \qquad (1\text{-}35)$$

Substituting for Gp_j from (1-28) (with i replaced by j) in the first term of the right-hand-side of (1-35) gives

$$\begin{aligned} (1\text{-}36) \qquad p_{i+1}^T G p_j &= -g_{i+1}^T (g_{j+1} - g_j)/t_j + b_i p_i^T G p_j \\ &= -g_{i+1}^T g_{j+1}/t_j + g_{i+1}^T g_j/t_j + b_i p_i^T G p_j. \end{aligned}$$

For j = i, the right-hand-side is seen to be zero, by using (1-26b) for the second term and substituting for the values of t_j

and b_i from (1-31) and (1-25d), respectively. For $j < i$, the right-hand-side is zero, by using (1-26b) for the first two terms and by employing (1-26a) for the last term.

Finally, we prove that, after at most $m \leq n$ iterations, the method terminates at the minimum: writing (1-25b), with $k = j+1$ (1) $n-1$ and adding, leads to

$$x_n = x_{j+1} + \sum_{i=j+1}^{n-1} t_i p_i . \tag{1-37}$$

Using (1-22) for x_n and x_{j+1} and substituting gives

$$Gx_n - Gx_{j+1} = g_n - g_{j+1} . \tag{1-38}$$

Multiplying (1-37) by G and employing (1-38) leads to

$$g_n - g_{j+1} = \sum_{i=j+1}^{n-1} t_i G p_i . \tag{1-39}$$

Multiplying (1-39) by p_j gives

$$g_n^T p_j = g_{j+1}^T p_j + \sum_{i=j+1}^{n} t_i p_i^T G p_j . \tag{1-40}$$

The right-hand-side of (1-40) vanishes because of the exact line-search and by conjugacy. Then

$$g_n^T p_j = 0,$$

which is true for $j = 0,1,\ldots,n-1$. Since $\{p_j\}_{j=0}^{n-1}$ are G-conjugate, consequently, they are linearly independent (see the proposition above). Therefore $g_n = 0$. Thus, if the method does not terminate earlier, it must terminate after the n-th iteration. ■

The above theorem indicates that we can minimize a quadratic function by means of an algorithm which does not require explicit knowledge of the matrix G. However, it does require exact line-searches.

It is important to consider how the conjugate gradient method defined by (1-25) can be applied to the minimization of non-quadratic functions. It is obvious that when we apply conjugate gradient methods to non-quadratic functions, they will not usually terminate within n steps. We may, therefore, simply continue to apply the algorithm and finish only when some termination criteria are met. This may lead to non-descent directions p_{k+1} because, if the formula (1-25c) is premultiplied by g_{k+1}, we find that

$$g_{k+1}^T p_{k+1} = -g_{k+1}^T g_{k+1} + b_k g_{k+1}^T p_k .$$

If an exact line-search was performed on the previous iterations, the direction p_{k+1} must be a descent direction, since $g_{k+1}^T p_k$ vanishes and hence $g_{k+1}^T p_{k+1}$ is necessarily negative. However if an inexact line-search was made, the quantity $b_k g_{k+1}^T p_k$ may be positive and larger than $-g_{k+1}^T g_{k+1}$, in which case p_k will not be a descent direction. A typical remedy for such an eventuality is the possibility of resetting p_k periodically to the steepest descent direction. For instance, p_k could be reset to $-g_k$ every n iterations (approximately), in which case a cyclic type of method is obtained. This strategy is known as "restarting" or "resetting". The conjugate gradient method that restarts with the

steepest descent direction after every n iterations is sometimes known as the "traditional conjugate gradient method" (see, for example, Gill et al(1981)).

When we minimize a non-quadratic function by the basic conjugate gradient method (1-25) we may justify restarting with the negative gradient by means of the following reasoning: in this event, the algorithm may be considered as consisting of several "major iterations", each containing (about) n "minor iterations". The last point of one major iteration is used as the first point of the next approximation. In each major iteration, we find the minimum of a local quadratic. The convergence of the algorithm for non-quadratic function depends on the convergence of the algorithm for the local quadratic approximation in each of the major iterations. The convergence of the basic conjugate gradient method (1-25) for each local quadratic function is based on the conjugacy of the directions which, in turn, is established by assuming $p_0 = -g_0$ (see above).

Consequently, the convergence of the algorithm for a non-quadratic function depends on the search direction being the negative gradient at the beginning of each major iteration. The algorithm (1-25) using such a 'restart' procedure becomes:

(1-41a) 1 $k = 0;\ p_0 = -g_0;$

(1-41b) 2 $x_{k+1} = x_k + t_k^{\#} p_k$

where $t_k^{\#}$ minimizes $F(x_k + tp_k)$;

3 If $k < n-1$

THEN

begin

(1-41c) $p_{k+1} = -g_{k+1} + b_k p_k$

(where $b_k = g_{k+1}^T g_{k+1} / g_k^T g_k$);

$k = k + 1;$

go to 2;

end

ELSE

begin

$x_0 = x_n;$

go to 1;

end.

In fact, for some problems with certain types of symmetry, it might be appropriate to restart the above algorithm more frequently (say, after every m (« n) iterations) instead of after every n iterations. Fletcher(1987) has analyzed the quadratic case with the Hessian G. If the eigenvalues and vectors of G are $\{\lambda_i\}$ and $\{u_i\}$, respectively, then g_0 and g_k can be expanded in terms of these eigenvectors as

$$g_0 = \sum_{j=1}^{n} \rho_j u_j; \quad g_k = \sum_{j=1}^{n} \rho_j h(\lambda_j) u_j,$$

where $h(\lambda)$ is a polynomial of degree k:

$$h(\lambda) \equiv 1 + \alpha_1 \lambda + \alpha_2 \lambda^2 + \ldots + \alpha_k \lambda^k,$$

for some constant α_i's. Thus, g_m can be close to zero if either

(i) x_0 is such that $\rho_j \approx 0$ for many values of j, or

(ii) the eigenvalues of G fall into multiple or close groups so that a good approximation to zero can be obtained by a low-order polynomial $h_m(\lambda)$ over the points $\{\lambda_j\}$.

Unfortunately, it is not usually possible to test these criteria a priori, and the easiest way of investigating them is to try the method and see if it works well.

The above discussion does suggest that on large problems for which the CG method is suitable, resetting p to the steepest descent direction every n iterations is an irrelevant consideration.

Another important method of this type is the Polak-Ribiere method (Polak(1971)), where b_k is obtained from the following expression:

$$(1\text{-}42) \qquad b_k^{PR} = (g_{k+1} - g_k)^T g_{k+1} / g_k^T g_k.$$

This leads to a value identical to the value of b_k used in the Fletcher-Reeves formula for the quadratic case (by (1-26b)). Another value for b_k has been suggested by Hestenes and Stiefel(1952) and has been used by Shanno(1978) and Nazareth and Nocedal(1982). This value is

(1-43) $b_k^{HS} = y_k^T g_k / y_{k-1}^T p_{k-1}$

where

$y_k = g_{k+1} - g_k$.

Powell(1977) makes the following observation:

(i): Suppose that such a method (either Fletcher-Reeves or Polak-Ribiere) makes little progress, so that $g_{k+1} \approx g_k$. Then the Polak-Ribiere formula (1-42) leads to $b_k \approx 0$ and hence $p_{k+1} = -g_{k+1}$, so that in these circumstances the algorithm tends to reset p_{k+1} automatically to the steepest descent direction and consequently the problem of finding m (see above) is "automatically" handled. For the Fletcher-Reeves method, $b_k \approx 1$, so this observation does not hold;

(ii) if the angle between the search direction and the steepest descent direction remains approximately constant and close to $\pi/2$ (so that the descent property $g^T s < 0$ is nearly violated) over a number of iterations, an algorithm becomes highly inefficient. For the Polak-Ribiere method, this inefficient behaviour does not occur because of the above argument (that p_{k+1} becomes nearly equal to $-g_{k+1}$ automatically), while, for the Fletcher-Reeves method, one should correct this inefficient behaviour by a steepest descent direction restart.

Finally, because of the reasons given below, Powell(1986) has claimed that the value

$b_k = \max \{0, (g_{k+1} - g_k)^T g_{k+1} / g_k^T g_k\}$

may be more useful than those used in the Fletcher-Reeves or Polak-Ribiere algorithms. If the term $b_k p_k$ (in equation (1-41c)) dominates the term $-g_{k+1}$ then, in the Fletcher-Reeves algorithm, the directions of p_{k+1} and p_k are nearly the same, but these directions can be almost opposite in the Polak-Ribiere method due to $b_k^{PR} < 0$. {b_k^{FR} is always positive, except at the solution, where it is zero; while b_k^{PR} may take negative values.}

Powell(1984) has shown global convergence for the Fletcher-Reeves algorithm when an exact line-search is used. Al-Baali(1985) has shown that this algorithm with an inexact line-search is globally convergent under certain standard conditions for terminating an inexact line-search.

There are some variants of the conjugate gradient method in which p_0 is not the steepest descent direction: for example, Beale's method (Beale(1972)). In this method, the initial search direction may be any non-zero vector. Other search directions (except the second one) then satisfy a recurrence of the form:

$$p_1 = -g_1 + c_{2,1} p_0;$$

$$p_i = -g_i + c_{i,1} p_0 + c_{i,i-1} p_{i-1} \quad \text{for } i > 1,$$

where

$$c_{i,j} = g_{i-1}^T (g_j - g_{j-1}) / p_{j-1}^T (g_j - g_{j-1}).$$

There are some methods which relax the requirement that line-searches be exact (see, for example, Nazareth and

Nocedal(1978)). Another (three-term) recurrence method due to Nazareth and Nocedal(1982) utilizes an arbitrary direction for p_0 and an inexact line-search procedure.

We now turn our attention to restarting strategies. Powell(1977) has shown that it is more efficient if p_0 is reset to directions other than the steepest descent direction because, by experience, the immediate reduction in the objective function is usually less than it would be if restarted with other directions. He suggests a technique that is based on the use of a nonsingular matrix for transforming the variables and which leads to a steepest descent direction in the transformed space. (Besides this, we may also utilize a matrix to accelerate conjugate gradient methods: see chapter 2.)

Powell(1977) has argued that the frequency of restarts should depend on the objective function. He points out that Beale's method (see above) may not generate a descent direction when F(x) is a general function and in this event another restart may be required. He quotes a result (due to McGuire and Wolfe(1973)) that, if Beale's algorithm is applied to a general function and if no restarts are made, then the calculated sequence of points converges to a limit which is not the required minimum of F. Powell states that a suitable time to restart is when the condition

$$(1\text{-}44) \qquad (|g_k^T g_{k+1}| / \|g_{k+1}\|_2^2) \geq c_1,$$

where $c_1 \in (0.1, 0.9)$ holds. By this means one decides whether

enough orthogonality between g_k and g_{k+1} has been lost to warrant a restart. Powell claims that the test (1-44) is more than a device to prevent convergence to a wrong limit: it has a positive advantage, because the test for restarting is suited to the particular form of F(x) through involving g_k and g_{k+1} in the test.

1-6-1-1 Properties of Conjugate Gradient Methods

These methods (with exact line-search and exact arithmetic) generally have quadratic termination, which is often regarded as a condition for a method to be considered efficient. In comparison with Newton's method, the number of required iterations is increased but, by way of compensation, we avoid the evaluation and storage of the Hessian matrix and the solution of linear systems of equations. Therefore these methods may be more useful and applicable for large problems for which storage for the Hessian matrix is not available. In comparison with the steepest descent method, these methods have only slightly more complicated formulas, but they do accelerate the slow convergence that is generally obtained from such methods. The traditional conjugate gradient methods are convergent for non-quadratic functions because the steepest descent direction is used regularly (Luenberger(1984)) and they exhibit n-step q-superlinear convergence:

$$\|x_{k+n} - z\| = o(\|x_k - z\|) \text{ for } k = 0, n, 2n, 3n, \ldots$$

(McCormick and Pearson(1969)). Fletcher states that these results are not really relevant to the practical solution of many large problems. Gill et al(1981) state that algorithms which do not contain a restarting strategy almost always converge linearly. Luenberger(1984) quotes a result (due to Cohen(1970)) that, since these methods find the minimum of a quadratic function in (about) n iterations and since Newton's method does so in one step (under certain conditions), it may be expected that (for general non-quadratic problems) such methods will exhibit n-step second-order convergence:

$$\|x_{r+n} - z\| \leq c\|x_r - z\|^2 \text{ for some } c \geq 0 \text{ and } r = 0, n, 2n, \ldots.$$

CHAPTER 2

QUASI-NEWTON METHODS

The rationale and motivation underlying the development of quasi-Newton ("QN") methods lie in the attempt to eliminate the defects of Newton's method while retaining its fast convergence (as far as possible). We remind the reader that the approaches described under the title "modified Newton methods" in chapter 1 are neither cheap to implement nor able to remove all of the drawbacks of the Newton's method simultaneously. The first papers on QN methods were written by Davidon(1959) [for minimization problem] and by Broyden(1965) [which was devoted to the more general problem of systems of nonlinear equations]. The Davidon method was developed and refined by Fletcher and Powell(1963) and is now known as the DFP method. (Note that, from now on, we will follow the notational convention of omitting the subscript "k" and replacing the subscript "k+1" with the superscript "*" when there is no danger of ambiguity.)

In the following, we describe how some of the defects of Newton's method are remedied by QN methods while, at the same time, we explain some of the specifications and properties of these methods.

(i) QN methods do not evaluate the Hessian matrix; instead, they build approximations to the Hessian (or its inverse) by use of an "updating" (or correcting) process. We denote matrices that

approximate the Hessian by B and those which approximate the inverse Hessian by H. The corresponding updating formulas are known as the "B-version" and "H-version", respectively.

(ii) Many of the update formulas produce pd matrices, provided that the initial matrix is pd and that other simple conditions (for example, $s^T y > 0$) hold . This property is important because (with exact arithmetic) it ensures that the next search direction is a descent direction.

(iii) If one of the B-version formulas is utilized, it is necessary (by analogy with the Newton's method) to solve the following linear system of equations to find p:-

(2-1) $Bp = -g$.

In the general case, solving (2-1) requires $O(n^3)$ operations and this defect can be eliminated by either of following:

(a) we employ one of the H-version formulas which does not require the solution of a linear system at every iteration, instead allowing the determination of the search direction p by a (computationally much less expensive) matrix-vector multiplication, as follows:

(2-2) $p = -Hg$.

(b) We update the Cholesky factors of the matrix, rather than an explicit representation of the elements of the matrix. If the Cholesky factors LDL^T of B are available, the system (2-1) can be solved in $O(n^2)$ operations. Furthermore, the factors L^* and D^* of the updated matrix B^* can be determined for a low-rank QN update in a number of operations comparable to the number required to

compute H^* (see Gill et al(1981)). When the Cholesky factorization of the approximate Hessian is available, we can obtain a convenient estimate of the condition number of B. It can be shown that if d_{max} and d_{min} are the largest and the smallest diagonal elements of D, then

$$K(B) \geq \frac{d_{max}}{d_{min}},$$

where K(.) denotes the condition number (with respect to the 2-norm). On those occasions when this lower bound indicates that the matrix B is badly-conditioned, the procedures for computing and updating the Cholesky factors can be modified so that K(B) does not exceed a fixed upper bound. The use of Cholesky factorization allows one to avoid a very serious problem that might otherwise arise in QN methods: the loss (through rounding error) of positive-definiteness in the Hessian (or inverse Hessian) approximation. This aspect of QN updating is important, especially for more difficult problems. In theory, many QN updating formulas guarantee to maintain positive-definiteness under some simple conditions (see below), while in practice it is not uncommon for rounding errors to cause the updated matrix to become singular or indefinite. However, in spite of these observations, Fletcher(1987) quotes a result (due to Grandinetti(1979)) that, even on ill-conditioned problems, no practical advantage in using factorization can be detected.

(iv) QN methods do not possess the second-order convergence which is such an attractive property of Newton's method. This is so

because, in QN methods, we do not evaluate the Hessian matrix explicitly. However, many QN methods in current use do possess q-superlinear convergence, under certain assumptions. Despite this slower rate of convergence, however, these methods often effect an order of magnitude reduction in the arithmetic operations required. Thus the value of these methods lies in being fast relative to the small amount of computation they require. To discuss the convergence properties of QN methods, we need to refer frequently to issues involving line-searches and trust-region techniques (to be considered in chapter 3). Thus we will study the convergence properties of these methods in more detail in Chapter 3 after examining these issues.

(v) In contrast to the steepest descent and Newton methods, which utilize information from the current iteration only, these methods collect and employ information from the current and all previous iterations.

(vi) There is a common advantage over conjugate gradient ('CG') methods which is shared by the Newton and QN methods. This is that, as we enter the neighbourhood of the solution, we may expect to be able to eliminate the line-search entirely, because a step-length of one will almost always be acceptable. In QN methods, as the iterations proceed, a good approximation to the Hessian (or its inverse) is usually created, producing Newton-like behaviour. This is not true for CG methods because (Shanno(1978)) Polak-Ribiere method is precisely the BFGS method (see below) where the approximation to the inverse Hessian is

reset to the identity matrix at every step, as no storage is used to develop a better approximation to the inverse Hessian. Therefore, no information concerning a proper step-length has been accumulated.

The central issue in QN methods is the construction of approximations to the Hessian (or its inverse). In section 1, we derive an equation which plays a major role in the construction of such matrices. We organize the remainder of this chapter as follows: in section 2, we discuss how to generate the update formulas and consider some known rank-1 and rank-2 formulas. Section 3 is devoted to other formulas (such as the Huang family and variants of QN methods), while, in section 4, variable metric methods and scaling will be examined. Finally, we end the chapter by looking at methods which combine CG and QN methods.

2-1 Derivation of the Secant Equation

Let s be the step taken from x, and consider expanding the gradient about $x + s = x^*$ in a Taylor series:

(2-3) $g(x) = g(x^*) - G^*s + \ldots$.

The higher terms on the right-hand-side of (2-3) are zero for a quadratic function and it is convenient to neglect them in the general case:

$$g(x + s) - g \approx G^*s$$

or

(2-4a) $y = B^*s,$

where B^* is an estimate of G^* and y is defined by

$$y = g^* - g.$$

The equation (2-4a) may also be written as follows:

(2-4b) $s = H^*y,$

where $H^* = (B^*)^{-1}$. Equation (2-4a) [or (2-4b)] has been called "the secant equation" by some researchers (for example, Dennis(1972)) and "the quasi-Newton equation" by Broyden(1965), among others. The first of these names for equation (2-4) was recently emphasized by Ford and Saadallah(1986) for the following reasons: define the line L that passes from x in the direction s:

(2-5) $L \equiv \{x(\tau) \equiv x + \tau s: \tau \geq 0\}.$

We note that $\tau = 0$ and $\tau = 1$ give the points x and x^*, respectively. Using

$$dx/d\tau = s$$

for all τ, applying the chain rule

$$dg/d\tau = dg/dx \; . \; dx/d\tau$$

or

$$Gs = dg/d\tau, \tag{2-6}$$

and substituting $\tau = 1$ (corresponding to x^*) gives

$$G^*s = dg/d\tau\big|_{\tau=1}, \tag{2-7}$$

which was called the "Newton equation" by Ford and Saadallah(1986). Along the line L, the gradient may be regarded as a function of τ and so we write it (with a minor abuse of notation) in the form $g(\tau)$. An approximation to the right-hand-side of (2-7) along the line L is obtained by using backward differences, yielding

$$dg/d\tau\big|_{\tau=1} \approx g^* - g = y. \tag{2-8}$$

This derivative estimate, when substituted into (2-7), indicates that (2-4a) is to be regarded as an approximation to (2-7). Ford and Saadallah argue that a "QN equation" is, then, any equation which constitutes a reasonable approximation to the Newton equation. They point out that, under this definition, the secant equation is a special case of a QN equation. Because of the relationship and similarity between this present work and that of Ford and Saadallah, we adopt their point of view and use the name "secant equation" for (2-4).

To derive an updating formula, we may utilize either (2-4a) or (2-4b), which will lead to formulas in terms of B or H,

respectively. Equation (2-4a) has exactly the same form as (2-4b) except that s and y are interchanged and H is replaced by B. This implies that any update formula for B derived to satisfy (2-4a) can be transformed into a corresponding update formula for H which satisfies (2-4b). Specifically, given any formula for B, a corresponding formula may be generated by interchanging the roles of B and H, and of s and y. Formulas related in this way are said to be "dual" or "complementary".

2-2 Generation of Formulas

Having obtained the secant equation, it is appropriate now to describe how one may generate an updating formula satisfying this relationship. We restrict our attention, for the time being, to the B-version, for the sake of brevity. As previously stated, we update (or correct) the matrix B; more precisely, at every iteration, the new matrix B^* is obtained by adding a correction C to previous matrix B:

(2-9) $B^* = B + C.$

To find C, we use the fact that B^* should satisfy (2-4a), that is

(2-10) $(B + C)s = y.$

This relation constitutes a set of n linear equations for the n^2 unknown components of the matrix C. Clearly, other conditions have to be added (for $n > 1$) to define C uniquely. Typical conditions by which we may further specify C are:

(i) C is a rank-1 matrix;

(ii) C is a rank-2 matrix;

(iii) C is a matrix of minimum-norm.

Many minimum-norm corrections turn out to be either rank-1 or rank-2 (see Broyden(1972)).

The minimum-norm approach is very attractive and important, because, first, the accumulated information (about the objective function) from the previous iterations contained in the approximate Hessian is changed as little as possible by the updating of the matrix with the information from the current iteration (namely, s and y) and, second, by this technique one may obtain most of the known update formulas, which were previously obtained by means of other derivations. Greenstadt(1970) was the first to suggest this approach. He considered the following:

$$\min\ (\|C\|_W) = \mathrm{Tr}(WCWC^T),$$

$$\text{subject to } C = C^T$$

and

$$Cs = y - Bs.$$

(In this problem, W is a pd weighting matrix and "Tr" denotes the trace.) We will describe how one may obtain some update formulas by a similar minimum-norm technique in the following sections.

When one of the B- or H-version formulas is available and yields a nonsingular matrix, we may obtain the other version by employing the Sherman-Morrison formula m times [see, for instance, Luenberger(1984)] for a correction matrix of rank m.

2-2-1 Rank-1 Methods

First, we turn our attention to Broyden's methods (Broyden((1965)), which were devised for the solution of general systems of non-linear equations $f(x)$: $R^n \rightarrow R^n$ (say), with Jacobian J. We study these methods here because they underlie and are closely connected with commonly-used minimum-norm update formulas (as we will see below). (In this section we modify the definition of y to be

$$y = f(x^*) - f(x).)$$

Broyden states that (2-4a) relates the change (y) in the vector function f to the change in x (namely, s). He argues that, because no information is available about the change in f when x is changed in a direction other than s, it is not possible to obtain a fresh estimate of rates of change of f(x) in such directions. Therefore, he reasons that there is really no justification for having B^* different from B in any direction q which is orthogonal to s. This can be expressed as the requirement

$$B^*q = Bq \quad \text{if } q^Ts = 0.$$

These conditions and (2-4a) uniquely determine B^* from B:

$$(2\text{-}11a) \qquad B^* = B + \frac{(y - Bs)s^T}{s^Ts}.$$

If B is nonsingular, it follows from the Sherman-Morrison formula that, on inversion, equation (2-11a) becomes

$$(2\text{-}11b) \qquad H^* = H + \frac{(s - Hy)s^TH}{s^THy}.$$

A second such method may be derived by a similar argument. This time, one utilizes (2-4b) and the following requirements:

$H^*v = Hv$ if $v^Ty = 0$.

This (with (2-4b)) uniquely determines H^* from H:

$$(2\text{-}12a) \qquad H^* = H + \frac{(s - Hy)y^T}{y^Ty}.$$

Again if H is invertible, using the Sherman-Morrison formula gives the "B-version" of (2-12a):

$$(2\text{-}12b) \qquad B^* = B + \frac{(y - Bs)y^TB}{y^TBs}.$$

An alternative derivation of the formulas (2-11a) and (2-12b) arises by assuming that the correction matrix C in (2-9) is rank-1, that is

$$(2\text{-}13) \qquad B^* = B + uv^T,$$

for some vectors u and v. From the secant equation (2-4a),

$B^*s = (B + uv^T)s = y$

or

$u(v^Ts) = y - Bs,$

and, therefore, u must be in the direction (y - Bs). For any vector v such that v^Ts is non-zero, the vector u is given by $(y - Bs)/v^Ts$ and B^* is defined by

$$(2\text{-}14) \qquad B^* = B + \frac{(y - Bs)v^T}{v^Ts}.$$

By substituting v = s and v = By, respectively, (2-11a) and (2-12b) are obtained.

Broyden points out that the formula (2-11a) shows good computational performance while the numerical results obtained by using the formula (2-12a) are not satisfactory. Because of these numerical properties, some authors (for example, Dennis and Schnabel(1983)) have called the formulas (2-11) and (2-12) Broyden's "good" method and Broyden's "bad" method, respectively.

We note that the formulas (2-11a) and (2-11b) are the dual of (2-12a) and (2-12b), respectively, according to the definition given in section (2-1).

Dennis and More(1977) have shown that the Broyden "good" update formula may be obtained by the minimum-norm technique. First, we prove the following Lemma.

Lemma 2-1:

Let $\alpha \in R$, $v \in R^n$, $v \neq 0$. Then the unique solution to

$$\min_{x \in R^n} \quad \|x\|_2$$

subject to $v^T x = \alpha$ is

$$x = \alpha v / v^T v.$$

Proof: If $\alpha = 0$, the Lemma is trivially true because $\|x\|_2 \geq 0$. If $\alpha \neq 0$, then we may write x in terms of a vector w say, in the following form:

$$x = \alpha w / v^T w,$$

under the assumption that $v^T w \neq 0$. Thus

$$\|x\|_2^2 = \alpha^2 \|w\|^2 / (v^T w)^2 \geq \alpha^2 / v^T v,$$

by the Cauchy-Schwarz inequality, with equality only if w is equal to a nonzero multiple of v. Therefore $\|x\|_2$ is minimized when w is a non-zero multiple of v, which implies that

$x = \alpha v / v^T v$. ■

Theorem 2-1

Given $B \in R^{n \times n}$, $y \in R^n$ and some non-null $s \in R^n$, define B^* by equation (2-11a). Then B^* is the unique solution to the problem

(2-15) $\min \{\|B^* - B\|_F\}$

subject to

$B^* s = y$ (the secant equation),

where $\|.\|_F$ denotes the Frobenius norm.

Proof: Define $C = B^* - B$. Postmultiplying by s and using the secant equation leads to:

$Cs = y - Bs$.

Now the problem can be rewritten in the form

$$(2\text{-}16) \quad \min_{c_i \in R^n} \sum_{i=1}^{n} \|c_i\|_2^2,$$

subject to $s^T c_i = (y - Bs)_i$, $i = 1, 2, \ldots, n$;

where the subscript i indicates row i of the corresponding matrix. The problem (2-16) can be broken up into n disjoint problems:

$\min \|c_i\|_2$

subject to $s^T c_i = (y - Bs)_i$,

for $i = 1, 2, \ldots, n$.

Using the above Lemma, the solution to each such problem is

$$c_i = (y - Bs)_i^T s / s^T s.$$

Thus the solution to (2-15) is (2-11a). In the other words, the Broyden "good" update formula yields the nearest matrix to B (in the Frobenius norm) among all the matrices that satisfy the secant equation. ■

Since the Broyden "good" update and "bad" update are duals of each other, therefore one may obtain the "bad" update (in its H-version) through the same minimum-norm technique.

In the case of optimization, since Hessian matrices are generally symmetric, it is reasonable to require that each approximate Hessian matrix B be symmetric as well. In other words, we should seek update formulas which possess the property of hereditary symmetry (that is, B^* is symmetric if B is). For a rank-1 update, the requirement of symmetry inheritance uniquely determines the update: in order that the general rank-1 update formula (2-13) (for some vectors u and v) should maintain the symmetry of B, v must be a multiple of u. The update formula then becomes

$$B^* = B + \frac{(y - Bs)(y - Bs)^T}{(y - Bs)^T s}. \qquad (2\text{-}17a)$$

This update is termed the symmetric rank-1 (SR1) update and has been discussed by Broyden(1967), Davidon(1968), Fiacco and McCormick(1968) and Murtagh and Sargent(1969), among others. The

H-version of the SR1 update formula is

$$(2\text{-}17b) \qquad H^* = H + \frac{(s - Hy)(s - Hy)^T}{(s - Hy)^T y}.$$

(We note that this formula is, therefore, self-dual.)

In spite of the advantages of QN methods which were stated at the beginning of this chapter, this particular method suffers from two serious disadvantages; it does not, in general, maintain positive-definiteness in the matrices {B} and, also, the denominator in (2-17a) or (2-17b) may be zero, so that the algorithm is no longer defined. Moreover, numerical difficulties arise when the denominator (of (2-17a) or of (2-17b)) is small. Therefore, a practical algorithm requires ad hoc safeguards to circumvent these difficulties, which is not entirely satisfactory.

2-2-2 Rank-2 Methods

2-2-2-1 The Broyden Family

The Broyden Family (Broyden(1967)) may be represented in terms of the scalar parameter ϕ_B by

$$(2\text{-}18) \qquad B^*_{\phi_B} = B + \frac{yy^T}{s^T y} - \frac{Bss^T B}{s^T Bs} + \phi_B vv^T,$$

where

$$v = (s^T Bs)^{1/2}\left(\frac{y}{s^T y} - \frac{Bs}{s^T Bs}\right).$$

When $\phi_B = 1$, we obtain the DFP update

$$(2\text{-}19) \qquad B^*_{DFP} = B + \left(\frac{1 + s^T Bs}{s^T y}\right)\frac{yy^T}{s^T y} - \frac{ys^T B + Bsy^T}{s^T y},$$

which was the first of these methods and was devised by Davidon(1959) and refined by Fletcher and Powell(1963).

Another important formula generated by the family is the BFGS update (corresponding to $\phi_B = 0$), which was independently suggested by Broyden(1970), Fletcher(1970), Goldfarb(1970) and Shanno(1970). The BFGS formula may be conveniently written in the form

$$(2\text{-}20) \qquad B^*_{BFGS} = B + \frac{yy^T}{s^T y} - \frac{Bss^T B}{s^T Bs}.$$

The SR1 formula is also a (degenerate) member of this family, corresponding to the value

$$(2\text{-}21) \qquad \phi_B = \frac{s^T y}{s^T y - s^T Bs}.$$

The Broyden family does not preserve positive-definiteness for all values of ϕ_B. Since the BFGS update (for $\phi_B = 0$) does preserve positive-definiteness when

$$(2\text{-}22) \qquad s^T y > 0$$

holds (see, for example, Fletcher(1987)) and the sum of a pd matrix and a psd matrix is a pd matrix, then (if $\phi_B \geq 0$ and (2-22) holds) the corresponding member of the family will preserve

positive-definiteness. More precisely, (2-18) preserves positive-definiteness for $\phi_B > \phi_s$, where ϕ_s is a value which causes B^* to be singular. The following expression gives ϕ_s (see, for example, Fletcher(1987)):

$$(2\text{-}23) \qquad \phi_s = \frac{(s^T y)^2}{(s^T y)^2 - (y^T H y)(s^T B s)},$$

which is negative (by the Cauchy-Schwarz inequality) if B (and, consequently, H) are pd. Therefore, ϕ_s is a threshold in the sense that, if ϕ_B is less than ϕ_s, the formula generates an indefinite matrix while, for $\phi_B > \phi_s$, the formula produces a pd matrix (provided that (2-22) holds.) Before studying the properties of this family, we consider whether the imposition of the condition (2-22) is realistic. Defining

$$s = x^* - x$$

and using (1-6) gives

$$s = t^{\#} p.$$

Employing (1-23) leads to

$$y = Gs,$$

in the case of a quadratic function with pd Hessian G. Taking the scalar product with s leads to

$$(2\text{-}24) \qquad s^T y = s^T G s > 0.$$

For a *general* function,

$$s^T y = s^T (g^* - g) = s^T g^* - s^T g,$$

and we may consider two distinct cases. The first case (when an exact line-search is utilized) implies that $s^T g^* = 0$, while $s^T g < 0$ holds (provided that $g \neq 0$) by the descent property. The

second case (when an inexact line-search is employed) is dealt with by imposing the condition $s^T g^* > s^T g$ (or a similar, stronger condition) in the line-search (this will be studied in more detail in chapter 3). Thus, in either case, we are able to ensure that (2-22) holds. Use of members of the family for $\phi_B > \phi_s$ (with $s^T y > 0$ satisfied) ensures (in theory, at least) that all Hessian approximations remain pd. However, in practice, it is not uncommon for rounding errors to cause the updated matrix to become singular or indefinite (see, for example, the comments at the beginning of this chapter and Gill et al(1981)).

We now commence our discussion of the properties of specific members of the family. The common properties of the DFP and the BFGS update are (for more detail and proofs, see Luenberger(1984) and Fletcher(1987)):

(1) for quadratic functions (with exact line-searches):

(i) they terminate in, at most, n iterations;

(ii) they generate conjugate directions when $B_0 = I$;

(iii) if n iterations are taken (with $B_0 = I$), the final matrix is the Hessian (that is, $B_n = G$).

(2) For general functions:

(iv) in exact arithmetic, they preserve positive-definiteness for matrices if (2-22) holds and $\phi_B > \phi_s$ (see above); hence, the descent property holds;

(v) they have a superlinear rate of convergence (see section 2-5);

(vi) they have global convergence for strictly convex

functions (with exact line-searches).

Early implementations of such methods attempted to carry out fairly accurate line-searches. The development of low accuracy line-searches showed that the DFP formula is less satisfactory than the BFGS update, as far as the number of function and gradient evaluations is concerned and, currently, the BFGS method is preferred in practice. Another pointer to the use of the BFGS formula is given by Powell(1985) who shows that the worst-case behaviour of the DFP method on certain simple quadratic functions is significantly inferior. Powell(1976) proves global convergence for the BFGS formula when an inexact line-search is used, a result which has not been obtained for the DFP formula so far. Numerical results given in Fletcher(1987) confirm the superiority of the BFGS update over other formulas.

Using the Sherman-Morrison formula, the inversion of (2-19) and (2-20) yields:

$$(2\text{-}25) \qquad H^*_{DFP} = H + \frac{ss^T}{s^Ty} - \frac{Hyy^TH}{y^THy};$$

$$(2\text{-}26) \qquad H^*_{BFGS} = H + \left(1 + \frac{y^THy}{s^Ty}\right)\frac{ss^T}{s^Ty} - \frac{sy^TH + Hys^T}{s^Ty}.$$

Comparing the formulas (2-19) and (2-26) (or (2-20) and (2-25)) shows that the DFP and BFGS update formulas are dual (or complementary) to each other, according to the definition given in section 2-1.

A *Broyden* method (Luenberger(1984)) is defined as a QN method in which, at each iteration, a member of the Broyden family is

used as the updating formula. The parameter ϕ_B may, in general, be allowed to vary from one iteration to another. Hence, a particular Broyden method is defined by a sequence ϕ_{1_B}, ϕ_{2_B}, ..., of parameter values. A *pure* Broyden method is one which uses a constant ϕ_B.

A special subset of the Broyden family is that given by values of $\phi_B \in [0,1]$, and was introduced by Fletcher(1970). This subset is often termed "the convex class" of formulas. The discussion above shows that, for ϕ_B in this interval, the updated matrices are guaranteed to be pd. The extreme values of this interval obviously yield the BFGS and the DFP updates, respectively. We may write "the H-version" of the Broyden family as follows:

$$(2\text{-}27) \qquad H^*_{\phi_H} = H^*_{DFP} + \phi_H v v^T,$$

where

$$v = (y^T H y)^{1/2}\left(\frac{s}{s^T y} - \frac{Hy}{y^T H y}\right).$$

Fletcher(1987) has given the following relation between ϕ_H and ϕ_B:

$$\phi_B = (\phi_H - 1)/(\phi_H - 1 - \phi_H \mu),$$

where

$$\mu = (y^T H y)(s^T B s)/(s^T y)^2.$$

The expression (2-27) gives the DFP and the BFGS formulas for $\phi_H = 0$ and $\phi_H = 1$, respectively. This expression indicates that, for $\phi_H > 0$, the eigenvalues of $H^*_{\phi_H}$ are greater than the corresponding eigenvalues of H^*_{DFP} (by the "Interlocking

Eigenvalues Theorem" (Fletcher(1970))). In particular, this result holds for the BFGS formula (corresponding to $\phi_H = 1$).

The SR1 update (2-17b) is a member of a Broyden family for

$$\hat{\phi_H} = \frac{s^T y}{s^T y - y^T H y}.$$

$\hat{\phi_H}$ cannot lie in the range [0,1] when H is pd and $s^T y > 0$. Thus the SR1 update cannot be a member of the convex class, under these conditions. In fact, if $s^T y > y^T H y$, then $\hat{\phi_H} > 1$, while, if $s^T y < y^T H y$, then $\hat{\phi_H} < 0$. The value of $\hat{\phi_H}$ (that is, whether $\hat{\phi_H} < 0$ or $\hat{\phi_H} > 1$) could therefore be taken as an indication of which member of the convex class to use. That is, if $\hat{\phi_H} > 1$, then use of H^*_{BFGS} (corresponding to $\phi_H = 1$) is indicated while, if $\hat{\phi_H} < 0$ then H^*_{DFP} (corresponding to $\phi_H = 0$) is (arguably) the formula to use. If equality holds, then no indication is given of which formula to use; in this case, Fletcher preferred to use H^*_{BFGS} on account of the importance of avoiding near-singularity in H^*. This way of selecting a formula is successful in practice and was suggested by Fletcher(1970). It is commonly called the "Fletcher Switch method". When the function is quadratic with pd Hessian G, Fletcher gives the following interpretation: on replacing s by $G^{-1}y$, then the relation $s^T y > y^T H y$ (for example) becomes

$$y^T G^{-1} y > y^T H y.$$

If this is true, the indication is that H is 'smaller' (in some sense) than G^{-1}, so that the 'larger' formula H^*_{BFGS} is used. If, however, $y^T G^{-1} y < y^T H y$, then H is 'larger' than G^{-1} and so the 'smaller' formula H^*_{DFP} is employed.

In addition to the methods of obtaining the well-known members of the Broyden family given by their originators, Goldfarb(1970) followed Greenstadt's approach (see above) and devised a family of formulas by minimum-norm techniques. He has stated that this family is equivalent to the Broyden family.

Dennis and Schnabel(1981) have introduced new derivations, based on utilizing the minimum-norm technique to obtain pd QN updates. They obtain the BFGS and the DFP update formulas by employing the Broyden "good" and "bad" update formulas, respectively. Here, for the sake of example, we restrict our attention to the BFGS formula. Before proving the required result, we need to deal with some preliminaries.

Lemma 2-2:

Let $y,s \in R^n$, s non-null, and let

$$Q(y,s) = \{A \in R^{n\times n} : As = y\}.$$

Then Q(y,s) contains a pd matrix if and only if, for some non-null $v \in R^n$ and nonsingular $J \in R^{n\times n}$, $y = Jv$ and $v = J^Ts$.

Proof:

Sufficiency: if v and J exist, then (clearly) $y = Jv = JJ^Ts$ and JJ^T is the pd matrix we seek.

Necessity: suppose A is a pd matrix with $y = As$ and let LL^T be the Cholesky factorization of A. Then $y = LL^Ts$. Setting $J = L$ and $v = L^Ts$ completes this part of the proof.■

Now assume that we have a pd approximate Hessian B and we wish to obtain B^*, which inherits these properties as well as the property of incorporating data from the current iteration by being in Q(y,s). There exists a Cholesky factorization of B:

(2-28) $B = LL^T$

and, from Lemma (2-2), B^* exists if and only if one can find a vector v and a matrix J^* such that $y = J^*v$ and $v = J^{*T}s$, whence we can take

(2-29) $B^* = J^*J^{*T}$.

Comparing (2-28) and (2-29), it seems that we should try to obtain J^* from L.

This observation motivates choosing J^* by the following procedure:

1. assume one knows $v \in R^n$; find the $J^* \in R^{n\times n}$ which is nearest L in the Frobenius norm and satisfies $J^*v = y$.
2. Solve for v so that $J^{*T}s = v$.

The proof of the theorem below shows that the matrix B^* obtained by this argument is the BFGS update.

Theorem 2-2

Let $L \in R^{n\times n}$ be nonsingular, $B = LL^T$, $y,s \in R^n$, s be non-null. There is a pd matrix $B^* \in Q(y,s)$ if and only if $s^Ty > 0$. If there is such a matrix, then the BFGS update $B^* = J^*J^{*T}$ is one such, where

(2-30) $J^* = L + (y - \alpha Bs)(L^Ts)^T/\alpha s^TBs$

and

$$\alpha = \mp(y^T s/s^T Bs)^{1/2}.$$

Proof: Recall first, from Lemma 2-2, that a necessary condition for the update to exist is that there exist a non-null $v \in R^n$ and a nonsingular J^* such that $J^* v = y$ and $J^{*T} s = v$. Therefore

$$v^T v = (J^{*T} s)^T (J^{*-1} y) = s^T y,$$

which shows that $s^T y > 0$ is necessary. Now we derive the BFGS via the above procedure. If we know v, then the nearest matrix to L that sends v to y is just the "good" Broyden update (Theorem(2-1)). In this setting,

$$J^* = L + (y - Lv)v^T/v^T v. \tag{2-31}$$

Thus, the problem of determining the n^2 elements of J^* has been reduced to that of finding the n components of v. Using the condition that

$$v = J^{*T} s = L^T s + v\,[(y^T s - v^T L^T s)/v^T v)] \tag{2-32}$$

implies that $v = \beta L^T s$ for some scalar β. Hence, the problem of determining the n components of v has now been reduced to that of finding the scalar β. Substituting $v = \beta L^T s$ in (2-32) leads to

$$\beta^2 = y^T s/s^T Bs.$$

Therefore, if $s^T y > 0$, we have defined a pd update in Q(y,s). Substituting $v = \beta L^T s$ with either value of β in (2-32) yields (2-30). The last result to prove is that B^* (as defined by (2-20)) is identical to $J^* J^{*T}$ (where J^* is given by (2-30)), but this is easily verified.■

Davidon(1975) introduced an algorithm that employs the Broyden family formula (in the "H-version"), but he selects the free parameter ϕ_H according to a different criterion. His choice for ϕ_H is obtained by minimizing the ratio of the largest to the smallest eigenvalue of $H^{-1}H^*$ among updates that give H^* pd. He called his choice of ϕ_H "optimally conditioned".

The optimally conditioned value ϕ_H obtained by Davidon may be within or outside the convex class ($\phi_H \in [0,1]$) . Schnabel(1976) shows that a similar improvement can be achieved when ϕ_H is restricted to the interval [0,1] (which gives the convex class of updates bounded by the DFP and BFGS updates (see above)). Schnabel sets ϕ_H to zero or unity when $\phi_H < 0$ or $\phi_H > 1$, respectively.

2-2-2-2 Powell-Symmetric-Broyden (PSB) Update

This update formula is a special case of a general formula which is obtained by the following process on the Broyden formula (2-14). (2-14) satisfies the secant equation but it is not symmetric. To symmetrize it, we form

$$B_2^* = (B^* + B^{*T})/2 \qquad (2\text{-}33)$$

B_2^* is symmetric but is not, in general, consistent with the secant equation. To find a matrix that simultaneously satisfies the secant equation and is symmetric, we apply the cycle of (2-14), followed by (2-33), repeatedly. To make the process

clear, we state it in pseudo-programming language, as follows:

Q_0 := B;
i := -2;
REPEAT
    i := i + 2;
    Q_{i+1} := Q_i + $(y - Q_i s)v^T/v^T s$;
symmetrize Q_{i+1},
to give Q_{i+2} := $(Q_{i+1} + Q_{i+1}^T)/2$;
UNTIL Q_{i+2} satisfies the secant equation.

{We have used a different notation for the matrices, in this case, to simplify the description.}

The limit of this process (Powell(1970c)) may be shown to be a symmetric update matrix which satisfies the secant equation:

$$(2\text{-}34) \qquad B^* = B + \frac{rv^T + vr^T}{v^T s} - \frac{(r^T s)(vv^T)}{(v^T s)^2},$$

where r = y - Bs. This update matrix is defined for any v that is not orthogonal to s. Substituting v = s in (2-34) gives the following formula:

$$(2\text{-}35) \qquad B^* = B + \frac{rs^T + sr^T}{s^T s} - \frac{(r^T s)ss^T}{(s^T s)^2},$$

which was suggested by Powell(1970c) and is (naturally) termed the Powell-Symmetric-Broyden (PSB) update.

Setting v = y - Bs and v = y in (2-34) gives the SR1 and DFP update formulas, respectively (Gill et al(1981)). It has also

recently been observed that one may also obtain the BFGS update (B-version) formula by substituting

$$v = y - \alpha Bs,$$

in (2-34) where α is given in (2-30) (see, for example, Dennis, Martinez and Tapia(1989)). (Of course, by "duality", the BFGS formula in the H-version can be obtained just as the DFP B-version was obtained above.)

An alternative derivation of the PSB update formula is given by Dennis and Schnabel, by means of the minimum-norm technique:

Theorem 2-3: (Dennis and Schnabel(1983))

Let $B \in R^{n\times n}$ be symmetric, $s,y \in R^n$, $s \neq 0$. Then the unique solution to

$$\min \|B^* - B\|_F$$

subject to the secant equation and $B^* - B$ symmetric

is the PSB update formula.

2-3 Other Formulas

2-3-1 Huang Family

Huang(1970) has devised a five-parameter family of updating formulas as follows:

$$(2\text{-}36) \qquad H^* = H + \rho \frac{su^T}{u^T y} - \frac{Hyv^T}{v^T y}$$

where

$$u = \alpha_1 s + \alpha_2 Hy$$

and

$$v = \beta_1 s + \beta_2 Hy.$$

Fletcher(1987) points out that there are, essentially, only three parameters, because the numerators and denominators of the second and third terms of the right-hand-side may each be divided by one of the corresponding constants. In the general case, this family is not symmetric and the updated matrix satisfies

$$(2\text{-}37) \qquad H^* y = \rho s \qquad (\text{or } B^* s = y/\rho),$$

instead of the secant equation $H^* y = s$. To obtain a symmetric update formula for this family, the number of parameters is reduced to two. A special case of the symmetric Huang family is the Broyden family (for which $\rho = 1$) for all iterations, while another special case (for which $\rho \neq 1$, in general) gives the methods of Biggs(1971 and 1973) (see below). One of the nonlinear gradient models to be introduced in chapter 4 leads to updates belonging to the symmetric Huang family with ρ positive

but (in general) not equal to unity.

2-3-2 Variants of Quasi-Newton Methods

2-3-2-1 Biggs's Method

An estimation of the curvature at x^* along direction s may be obtained by

$$s^T B^* s = s^T y,$$

in place of the true value $s^T G^* s$. The estimate $s^T y$ is precise for a quadratic objective function. Otherwise, the estimate may not be exact because of non-quadratic behaviour, and therefore may be in need of improvement. This was the motivation of Biggs's(1973) proposal for interpolating the objective function by means of a cubic polynomial at the current points x and x^* and finding the curvature estimate along s from this cubic. The interpolation uses function values and the directional derivatives at x and x^*. The curvature estimate based on this cubic interpolation is

$$\delta = 4s^T g^* + 2s^T g - 6(F^* - F) \qquad (2\text{-}38)$$

(for the derivation of this equation, see chapter 4).

Biggs defines η^* by

$$\eta^* = s^T y/\delta, \qquad (2\text{-}39)$$

so that

$$\eta^* = \rho \qquad (2\text{-}40)$$

in (2-37). This obviously gives $\eta^* = 1$ when $\delta = s^T y$ or the

function behaves quadratically. Using the suggestion of Biggs, the Broyden family may be modified as follows:

$$B_{\phi}^{*} = B + (1/\eta^{*})\frac{yy^{T}}{s^{T}y} - \frac{Bss^{T}B}{s^{T}Bs} + \phi_{B}vv^{T},$$

where v is defined in (2-18).

2-3-2-2 Davidon and Sorensen Methods

Most minimization methods update quadratic approximations to the objective function F, and use the minimizers or "reducers" of the successive approximations to estimate the minimizer for F. (Since the new iterate at each step is usually obtained by an inexact line-search (see chapter 3) and is therefore not a precise minimizer of the approximation, we call it a "reducer".) To build these approximations, usually the gradient(s) of F are utilized and the updating process is rarely influenced by function values (Biggs(1971 and 1973), Ford and Saadallah(1986), Saadallah(1988) and our methods in this thesis are exceptions.) Davidon(1980) suggested a generalization from quadratic to "conic approximations", defined as ratios of quadratics whose denominators are squares $(a + b^{T}x)^{2}$ where a is a real number and b, $x \in R^{n}$. Equivalently, he introduced the "collinear scaling"

$$S(x) = x_{0} + \frac{Jx}{1 + h^{T}x}, \text{ say,}$$

with $J \in R^{n\times n}$, h, $x_{0} \in R^{n}$, which is a generalization of the affine scaling

$S(x) = x_0 + Jx.$

In the Davidon method employing either conic approximations or collinear scaling, the function values are utilized, in addition to gradient values. Davidon has shown that a particular member of his class of methods has q-superlinear rate of convergence under some assumptions on the objective function. He has pointed out that methods based on conic approximation and collinear scaling are equivalent, in the absence of rounding error.

Sorensen(1980) has investigated the second Davidon method (based on collinear scaling) and has introduced a class of algorithms. The derivation of this class of algorithms is based upon a certain set of equations. Manipulation of these equations and setting the corresponding variables to a special set of values leads to the secant equation. Moreover, this close relationship to existing QN methods makes it possible to identify particular members of the class which are likely to have a fast local rate of convergence.

In fact, Sorensen demonstrates that a particular member of the class has a q-superlinear rate of convergence. To implement these methods, it appears that two $n \times n$ matrices (one for the Hessian and another for the matrix J of the collinear scaling) need to be stored. Sorensen has shown that it is not necessary to hold J and H separately. Instead, he has shown how maintaining and updating one specific matrix is sufficient to be able to implement his algorithm.

2-3-2-3 The Methods of Ford and Saadallah

Newton's method iteratively solves the nonlinear system of equations

$$f(x): R^n \rightarrow R^n$$

by

$$x^* = x - J^{-1}f,$$

where J is the Jacobian matrix (evaluated at x). As previously indicated, one of the most important defects of Newton's method is the requirement to evaluate the Jacobian (or, equivalently, the Hessian matrix in the case of minimization problems). Regarding $f(x(t))$, temporarily, as a function of t only on the line

$$L \equiv \{x(t) \equiv x + tp\},$$

using

$$dx/dt = p$$

for all t, and applying the chain rule leads to

(2-41) $$df/dt = Jp.$$

The main aim in QN methods is to obtain an approximation to the Jacobian matrix. If an accurate estimate of df/dt is available, it may be used, via equation (2-41), to establish a condition that any approximation to the Jacobian matrix should satisfy. Assuming that the vector function f is sufficiently complicated, we shall have to be content with an estimate of df/dt. Assuming that

$$x^* = x + t^*p,$$

where $t^{\#}$ is some "acceptable" value of t (a process for obtaining $t^{\#}$ is to be described in chapter 3), substituting an approximation to J, say B, in (2-41) and evaluating the expression at the point x^* gives

$$B^*p = df/dt|_{t=t^{\#}},$$

which is equivalent to

$$(2\text{-}42) \quad B^*s = t^{\#}df/dt|_{t=t^{\#}},$$

using $s = t^{\#}p$.

When we consider minimization problems, we may assume that, for the objective function F, B is an approximation to the Hessian matrix and f (which is then denoted by g) is the gradient vector of F. Therefore (2-42) (in the case of minimization problems) where we regard g(x(t)) as a function of t only on the line L, becomes

$$(2\text{-}43) \quad B^*s = t^{\#}dg/dt|_{t=t^{\#}}.$$

Broyden(1965) estimated df/dt in equation (2-42) by means of backward differences and obtained, therefore,

$$(2\text{-}44) \quad B^*s = f^* - f = y.$$

Hence, J^*s is the scaled directional derivative of f(x) (in the case of nonlinear equations) or the scaled directional derivative of g(x) (in the case of minimization problems), so that y in equation (2-44) or (2-4a) is to be interpreted as an estimate of this derivative.

Ford(1986) pointed out that, unless $\|s\|$ is small, this estimate is likely to be rather crude. Therefore he suggested

using the polynomial of degree m which interpolates $g(x(t))$ at some points t_i, $i = 0, 1, \ldots, m$, where $t_0 = 0$ (corresponding to x) and $t_1, t_2, \ldots, t_m$ are obtained by some means (for example, during a line-search), in order to obtain a better estimate for the directional derivative. To implement such a method, we need at least three values of the gradient along the given search direction to gain any advantage. However, it is current practice to use line-searches with low-accuracy (see section 3-1), which implies that three (or more) gradient values from points on a single search direction will only rarely be available.

Along the line L, as previously indicated, the gradient may be regarded as a function of t and so we may write it (with a minor abuse of notation) in the form $g(t)$ $\{\equiv g(x + tp)\}$. To estimate dg/dt in (2-43), Ford and Saadallah(1987) suggested a new model for $g(t)$ (for the motivation of the new model, the reader is referred to the beginning of chapter 4), which is defined by only two gradient values, namely g and g^*. The model is

$$g(t) \equiv (a + b\tau)\exp(\theta f(\tau)),$$

where $f(\tau)$ is an arbitrary function of $\tau = t/t^{\#}$ and θ is a parameter to be determined. Their reason for writing the scalar multiplier factor as an exponential function was that such a function is never zero, so that the gradient model yields no "spurious" stationary points. Using τ in place of t as the independent variable causes a natural scaling for t that may be useful due to slight simplification in the algebraic operations.

To obtain θ, Ford and Saadallah use information contained in

the current Hessian approximation B. On the one hand, the curvature of F in the direction p is $p^T dg/dt$, which may be determined from the model and, specifically, in terms of θ. On the other hand, the curvature of F in the direction p is given by $p^T Gp$, which may be approximated by $p^T Bp = -p^T g$ (see chapter 4). Hence, on equating the two expressions for the curvature estimates, we obtain an equation defining θ. The root of this equation is used to evaluate

$$w = dg/d\tau\big|_{\tau=1},$$

which is (hopefully) a better approximation than y to the directional derivative. Near the minimum, the gradient may be expected to exhibit linear-behaviour, so that $\theta \approx 0$ and $w \approx y$ would be anticipated. Saadallah(1988) has utilized a similar approach with the following models:

$$g(\tau) \equiv (a + b\tau)/(1 + \theta f(\tau)),$$

$$g(\tau) \equiv a\tau + b \exp(\theta\tau),$$

(which have one parameter each) and the two-parameter model

$$g(\tau) \equiv a \exp(\theta_1 f(\tau)) + b\tau \exp(\theta_2 f(\tau)).$$

2-4 Variable Metric Methods and Scaling

The aim of this section is to describe a few methods which seek to compensate for poor scaling of the variables and of the objective function. We organize the section under these titles:

(i) variable metric methods;

(ii) motivation for scaling;

(iii) scaling of variable metric matrices.

2-4-1 Variable Metric Methods

As previously indicated, the first QN method for optimization was suggested by Davidon(1959), who called it a 'variable metric method' (or 'VMM') under the assumption that the matrix H is a non-negative definite matrix. Greenstadt(1970) defines variable metric methods as a class of the gradient methods in which the direction p is computed from (1-5) {p = -Mg} where M is a (preferably) pd matrix. Oren(1973) and Fletcher(1987) restrict variable metric methods to those QN methods which maintain positive-definiteness in the matrix H. These methods may then be regarded as based on a definition of a 'norm' in terms of a matrix M, say (which is usually the Hessian or its approximation), as follows:

(2-45) $$\|v\|_M = (v^T M v)^{1/2},$$

where v is any vector in R^n and M is a pd matrix.

Let us suppose that the distance d from x to u is defined by

$$d(x,u) = [(x - u)^T M(x - u)]^{1/2} = \|x - u\|_M .$$

In terms of this metric, it may be shown that the Newton and QN methods can be viewed as steepest descent procedures. The difference between the two types of method lies in the choice of the matrix M and, hence, the norm or metric. Following Crockett and Chernoff(1955), one possible way of defining the "steepest descent" direction under the stated metric is to let $x_{k+1}(h)$ minimize F(x) on the ellipsoid

$$(x - x_k)^T M(x - x_k) = h^2,$$

with centre x_k and radius h. The direction of steepest descent from x_k (in the ellipsoid with centre x_k) may be defined as the direction from x_k to $x_{k+1}(h)$, as $h \to 0$. In the limit, the direction $x_{k+1}(h) - x_k$ approaches a limit which is the direction of the vector

$$p = -M^{-1} g_k .$$

Using (2-45) to measure distances (instead of the normal Euclidean definition) with respect to varying M at each iteration is the reason for calling these methods "variable metric methods". Davidon(1959) states that, if M^{-1} is constant, then $0.5 g^T M^{-1} g$ would be the amount by which F(x) would exceed its minimum value and, therefore, he considers M^{-1} (or H) as a specifying matrix when one refers to the lengths of vectors.

2-4-2 Motivation for Scaling

The Broyden family may be written as

(2-46a) $H^{*}_{\phi_H} = E + C$

where

(2-46b) $E = H - A + \phi_H vv^T$, $A = Hyy^TH/(y^THy)$, $C = ss^T/(s^Ty)$,

and v is as in (2-27). If one rescales x by a factor ω, then

$x^ = \omega x$, $s^ = \omega s$, $g^ = g/\omega$, $y^ = y/\omega$, $C^ = \omega^2 C$, $v^ = v$.

Thus the magnitude of the elements of $H^{*}_{\phi_H}$ depends on the scale chosen for the variable x. In particular if

$\|y\| \gg \|s\|$ or $\|y\| \ll \|s\|$,

the elements of C will be very small or very large compared to E in (2-46) and consequently C may dominate E (or vice versa). Bard(1968) has found that if C_0 is very large or very small, the DFP matrix may become nearly singular and this disadvantageous feature may not easily be recovered from in subsequent iterations.

We note that under the scaling

$F^ = \lambda F$,

we have

$g^ = \lambda g$, $y^ = \lambda y$, $s^ = s$, $C^ = C/\lambda$, $A^ = A$, $v^ = v$, $H^ = H$.

This shows that C depends on λ, while E is not affected. This implies that the sequence of points generated for F^ will in general be different from the sequence of points generated for F.

We may also observe that when one multiplies H by a constant, all terms in (2-46) are equally magnified except C, which is invariant.

The above observations show that the Broyden family is not invariant to linear scaling of variables, function or H. This fact may be a cause of instabilities and has motivated researchers to find remedies. These remedies are

(i) scaling H in each iteration,

(ii) scaling H_0 before evaluating H_1 only (see Bard(1968) for further explanation).

If the initial matrix is unscaled or scaled without knowledge of the actual magnitude of the elements of the true inverse, severe loss of accuracy due to rounding-errors can occur. An arbitrary initial scaling can cause large loss of accuracy in terms arising from C if the scale of H is too large, or in terms arising from E if the scale of H is too small. Both losses are significant, as loss of accuracy in C prevents convergence to an approximate Hessian (see Fletcher and Powell(1963), in relation to the DFP update), while poor accuracy in E (such that C dominates E) prevents the determination of a suitable search direction in the next iteration because, in this event, the next search direction may be obtained approximately as

$$-Cg^* = -(s^T g^*/s^T y)s,$$

which is in the direction of the last search.

2-4-3 Scaling of Variable Metric Methods

Following on from the above discussion, Oren(1973,1974) suggested methods which utilize a formula from a two-parameter (ϕ_H, γ) family given by

(2-47a) $H^* = \gamma E + C,$

where

(2-47b) $\gamma = \eta\, g^T s/(g^T Hy) + (1 - \eta) s^T y/(y^T Hy),$

E and C are as in (2-46b) and both η and ϕ_H lie in [0,1]. For $\gamma = 1$, (2-47a) is identical to the convex class of the Broyden family. Oren states that, for an arbitrary γ, (2-47) has the effect of scaling F by γ while using a member of the convex class of the Broyden family. Furthermore, these formulas are invariant under scaling of the objective function or of the variables.

Oren and Spedicato(1976) consider the problem of the condition number of the matrix $H^{-1}H^*$, and from this derive the value

(2-48a) $\phi_H = b(c - b\gamma)/(\gamma\,(ac - b^2)),$

where

(2-48b) $a = y^T Hy,\ b = s^T y,\ c = s^T Bs.$

Shanno and Phua(1978) state that the principal drawback to the self-scaling variable metric updates is that, for a quadratic function, the sequence (2-47) can fail to converge to the inverse Hessian matrix. McCormick and Ritter(1972) have shown that this convergence is desirable on a non-quadratic problem, and it is a property which one cannot easily ignore. Spedicato(1975) has

considered the problem of initializing H_0, suggesting that H_0 be set to a diagonal matrix whose elements are the reciprocals of the true diagonal elements of the Hessian evaluated at the initial approximation x_0. There is a major drawback to this, namely that it is often expensive (or even impossible) to obtain these derivatives. A second objection (due to Shanno and Phua(1978)) is that computational efficiency is not always improved.

Shanno and Phua(1978) have considered two much simpler initial scalings of H_0 which require no additional information about the objective function other than that routinely required by variable-metric algorithms. One of the initial scalings may be used together with any member of the Broyden family, while the other one is for use with the BFGS update only. In both of these scalings $H_0 = I$ may be used initially. Nothing can be determined about an approximate scaling until one iteration has been completed, but it then becomes possible to scale at the end of the first iteration. For their first method of initial scaling, Shanno and Phua utilize $t_0^{\#}$, which is chosen according to linear search conditions (see chapter 3, section 3-1) to ensure a sufficient reduction in F. Once x_1 has been chosen, but before H_1 is calculated, H_0 is scaled by

$$\hat{H_0} = t_0^{\#} H_0; \tag{2-49}$$

then we compute H_1, by updating $\hat{H_0}$ rather than H_0. Shanno and Phua's justification for utilizing the above scaling is as

follows: assuming only that the criterion for choosing $t_0^{\#}$ is sensitive to multiplication of the objective function by a constant, an assumption satisfied by virtually all commonly-used step determination methods, the sequence of iterates arising from any member of the Broyden family then becomes self-scaling, without use of the parameter γ. In fact, (2-47a) may be viewed as deriving from a composite process of the form

(2-50a) $\quad \hat{H} = \gamma H,$

and

(2-50b) $\quad H^{*} = \hat{E} + C = \gamma E + C,$

because

(2-50c) $$\hat{E} = \hat{H} - \frac{\hat{H} y y^{T} \hat{H}}{y^{T} \hat{H} y} + \phi_H \hat{v} \hat{v} = \gamma E.$$

This means that (2-47a) is equivalent to scaling H by a factor γ and then applying a Broyden family update. Thus a motivation for the self-scaling variable metric method update is that, if H is a good approximation to the true inverse Hessian, $t_0^{\#}$ will be 1. If $t_0^{\#} \neq 1$, H is first scaled by multiplication by $t_0^{\#}$, then the Broyden family update is applied. Another initial scaling suggested by Shanno and Phua is specially for the BFGS update. Recalling the Oren-Spedicato formula (2-48) for the relationship between ϕ_H and γ which will optimally condition $H^{-1}H^{*}$, Shanno and Phua use the fact that, for the BFGS method, $\phi_H = 1$ and derive the optimal γ as

(2-51a) $\quad \gamma = b/a,$

yielding the initial scaling

(2-51b) $\hat{H}_0 = (b/a)H_0$,

where a and b are as in (2-48). This gives

(2-51c) $\hat{H}_0 = (s^T y)/(y^T H_0 y)H_0$.

The initial scalings have these properties:

(i) they leave self-scaling VMMs unchanged, while making the Broyden family self-scaling.

(ii) The level of improvement tends to increase monotonically as the number of variables increases.

(iii) In addition to providing the desired invariance, they both greatly enhance the numerical stability of members of the Broyden family.

(iv) After the initial iteration, the approximate Hessian is never rescaled, while the self-scaling VMMs need to scale H by γ at each iteration.

(v) The computational evidence documented in the paper by Shanno and Phua shows that both initial scalings are remarkably similar on small problems, but limited computational experience on a large problem, where effort does increase with growth in problem size, indicates that the scaling (2-51) may be preferable.

(vi) The scaled BFGS shows marked computational superiority (except on a highly specialized class of functions) over the unscaled BFGS.

2-5 Combination of Conjugate Gradient and Quasi-Newton Methods

In this section, we study methods which have been proposed as combinations of CG and QN methods. The main objective of the proposed combination is to allow the use of a variable amount of storage according to availability in such a way that more storage will (hopefully) mean improved performance. Briefly, we wish to discuss the following issues in this section:

(a) motivation for the combination;

(b) what should be employed in the combination?

(c) matrices and CG;

(d) different possibilities for the combination;

(e) properties of the combined methods.

(a) Motivation for Combination

Both CG and QN methods are successful in minimizing nonlinear functions of several variables, and each have their properties and advantages. When the number of variables is high, a CG method requires much less storage to implement than a QN method, particularly since the CG method does not require storage of the matrix, as does a QN method. Precisely speaking, QN methods use $O(n^2)$ storage locations while CG methods require some small multiple of n locations of storage for their implementation. Therefore CG methods may find application when storage limitations occur. On the other hand, CG methods are generally slower than QN methods, that is, they may be expected to require

more iterations than a QN method on the same problem. McCormick and Ritter(1972) have compared these two methods and showed that under certain conditions CG methods exhibit an n (or (n-1)) step superlinear rate of convergence while QN methods exhibit, at every step, superlinear convergence. Therefore, researchers have attempted to combine CG and QN methods in such a way as to retain the advantages of both methods.

(b) What Should Be Employed in the Combination?

The major difference between the methods of CG and QN is the existence of the approximate Hessian matrix (or its inverse) in QN methods. This matrix may be interpreted as an accumulator of data from previous iterations, by which the convergence is accelerated. The matrix that is obtained by one of the QN update formulas may be employed in an appropriate manner (which depends on available storage) for improving CG methods. Since the main philosophy behind these combined methods is the use of them in a limited storage environment, the matrix should have a special form such that it can be utilized in these circumstances. QN update formulas consist of a sequence of rank-1 or rank-2 corrections to a matrix which normally begins as the identity matrix. On the other hand, in practice, the matrix is not itself needed and only certain products with some vectors are required. Thus, the matrix need not be stored explicitly; instead, the required vectors should be stored. Therefore, if one limits the number of updates allowed and stores the vectors defining the

rank-1 corrections explicitly, then far fewer than n(n+1)/2 locations (which are needed for a symmetric n by n matrix) are needed to define and record the update QN matrix. Limiting the number of rank-1 corrections implies that we will discard some of the old rank-1 corrections. Since saving more rank-1 corrections leads to a more precise matrix there is, as a consequence, a tradeoff between the number of vectors saved and the speed of convergence.

(c) Matrices and Conjugate Gradient Methods

Allwright(1973) introduced a CG algorithm that employs a matrix Q, say, and the normal CG for Q = I. Powell(1977) utilized a nonsingular matrix for transforming variables in a CG method so that one may choose an arbitrary downhill restarting direction. This leads to quadratic termination for a general function. Nazareth(1979) showed that the BFGS method can be interpreted as a CG method for which the metric (or preconditioner) is changed at each step. Perry(1976) notes that the CG direction can be written in the form of the product of a matrix and gradient vector (as it is for QN methods). By adding a new term to his matrix, he tried to find a similar (but not identical) formula to the QN equation for CG methods. By symmetrization of Perry's matrix and forcing the matrix to satisfy the secant equation, Shanno(1978) introduced a new formula for the matrix that is identical to the BFGS method for $H_k = I$ for any k. Shanno states that Polak-Ribiere method is the BFGS method where the

approximation to the inverse Hessian is restarted as the identity matrix at every step, so that no storage is used to develop a better approximation to the inverse Hessian. The CG algorithm implied by Shanno's matrix does not actually require a matrix at all and no additional information beyond that used by normal CG methods is required. By this means, Shanno claims that it is possible to generate a CG algorithm corresponding to a memoryless QN method for any value of ϕ_u in the Broyden family.

Others (for example, Buckley(1978a,1978b), Nazareth(1977,1979) and Nocedal(1980)) have utilized a matrix for the purpose of improving CG methods. These methods have been named "preconditioned conjugate gradient" (or "PCG") methods (because the variables are preconditioned by the matrix) or CG methods with transformed variables. We will follow the first nomenclature. Most PCG methods have the following general form:

$p_0 = -H_0 g;$

for $k = 1, 2, \ldots$

$p_k = -H_k g_k + b_k p_{k-1}$,

where b_k may be chosen by means of (1-25d), (1-42) or (1-43).

$x_{k+1} = x_k + t_k^{\#} p_k$.

For some PCG methods, all the matrices H_i ($i = 0, 1, \ldots$) are set to one fixed H for a specified period. Naturally, the PCG methods

reduce to the normal CG algorithm when H = I. Choosing different update formulas for the matrix (for example, the single rank-1 or the BFGS update formula) and different values for b_k lead to different PCG algorithms. A PCG algorithm can be interpreted as the following process:

perform the transformation $u_k = H_k^{-1/2} x_k$ with the pd matrix H_k;
apply the CG method in the transformed coordinates;
transform back into the x coordinates.

When we apply the CG algorithm in the transformed coordinates u, we obtain a sequence of points and search directions satisfying the standard properties of the CG algorithm in the new coordinates. Naturally, this process terminates in at most n steps for quadratic function.

(d) Different Possibilities for the Combinations

There are two general classes of approach for accelerating CG methods with QN matrices. The first class was proposed by Nazareth(1979) and consists of applying the CG with a variable preconditioner, which will be updated using a QN formula. At a given iteration, the preconditioner is modified or remains the same, depending on the amount of storage available. The second class of approach was proposed by Buckley(1978a,1978b) and can be understood as interleaving QN and CG methods. In this class, if, at a given iteration, it is decided that the preconditioner is to

be changed, then the matrix is updated and a QN step is performed. On the other hand, if the matrix is not updated, a preconditioned CG step is defined. Both classes employ the PCG with this difference: that the first utilizes the PCG for all iterations, whereas the second periodically employs a QN step and, for the rest of the iterations, the PCG method is applied. The periodic QN step plays the role of restarting and thus guarantees convergence. Nazareth and Nocedal(1982) have considered the convergence properties of these two classes and called them "variable metric conjugate gradient" (or "VMCG") and "interleaved QN-CG" methods, respectively. They have proved that both classes (with exact line-searches) have quadratic termination, provided that the preconditioning matrix for VMCG is one of the Broyden family update formulas and, for the interleaver QN-CG, that the preconditioning matrices are obtained by the BFGS update formula. Nocedal(1980) introduced an update formula by employing the BFGS update formula to generate matrices using information from the last m iterations, where m is any number, depending on available storage. The matrix (called a special BFGS update) is updated at every iteration by dropping the oldest information and replacing it with the newest. Nocedal(1980) introduced an algorithm that is in the first class of approaches and employs the special BFGS update formula. He shows that this algorithm is very efficient and its performance improves consistently as the storage increased. It also produces pd matrices and preserves the quadratic termination. For other

work that has been carried out to improve CG methods, see Buckley and Lenir(1983) and Gill and Murray(1979).

(e) Properties of the Combined Methods

As previously indicated, these methods run without storing any matrix, but they will use more storage than ordinary CG methods and may be applied for cases when this small, extra amount of storage locations is available but the $O(n^2)$ locations required for QN methods is not available. In terms of the number of iterations required by these algorithms, generally fewer iterations are used than in a CG method, but a few more are required than in a QN method. Computational experiments with these methods are described in Buckley(1978b), Nazareth(1977) and Nocedal(1980).

CHAPTER 3

IMPLEMENTING THE NEWTON AND QUASI-NEWTON METHODS

In section 1-4, we briefly pointed out how a minimization problem may be solved by a gradient method. Here, we restrict our attention to some of the issues involved in implementing such an algorithm which utilizes the exact Hessian matrix (in the case of Newton's method) or an approximate Hessian generated by a QN method. This is a general framework for such algorithms:

Algorithm 3-1:

(0) Let $k = 0$, and choose x_0;

(1) If the method is "QN", then

if $k = 0$ choose a pd matrix as the approximate Hessian (see the comment below), else update the Hessian;

If the method is "Newton", then evaluate the Hessian.

(2) If "convergence" then stop;

(3) Scale the matrix if necessary;

(4) To find a better estimate of the solution apply either

(4a) a line-search technique; or

(4b) a trust-region technique;

(5) Update x_k, g_k and k; Go to step (1).

Comment: The initial matrix for a QN method may be any pd matrix. Unless a better approximation to the Hessian at x_0 is available,

we often utilize the unit matrix. When one utilizes a line-search technique, using the unit matrix on the first iteration yields a step in the steepest descent direction, which can be effective and advantageous.

(From now on we will use the term "(quasi-) Newton method" to mean either "Newton method" or "quasi-Newton method", as appropriate.)

In the preceding chapters, we have considered steps (1) and (3) in detail. Two other important steps (namely, (2) and (4)) and the convergence properties of QN methods will be studied in this chapter to complete our description of the implementation of the algorithm. Before stating our organization of this chapter, we describe the motivation for step (4). First, we recall the following points from preceding chapters which will be useful in this chapter.

Newton's method and QN methods are based on constructing a quadratic model with a matrix M, say (either the Hessian in Newton's method, or an approximation to the Hessian in QN methods). In either case, a (hopefully better) estimate x_{k+1} for the solution may be obtained by finding the minimum of the quadratic model. We may define x_{k+1} by the following expression:

(3-1) $\quad x_{k+1} = x_k + p_k,$

where

(3-2) $p_k = -M_k^{-1} g_k$ [$\equiv s_N$, the full (quasi-) Newton step].

It is possible that x_{k+1} may not satisfy

(3-3) $F_{k+1} < F_k$,

which is an obvious condition for convergence to the solution. If M_k is not pd, x_{k+1} may not be a minimum of the quadratic model and (3-3) may not hold, as a result of this fact. Even when M_k is pd (which implies a strictly convex function for the quadratic model), one may obtain x_{k+1} by (3-1) such that (3-3) does not hold or, if it holds, F_{k+1} is not sufficiently less than F_k in order to guarantee convergence to the solution. These difficulties may be considered as the main reasons why we use either line-search or trust-region techniques. These techniques are utilized to obtain a new point such that the function value at the new point is less than (in fact, sufficiently less than) F_k to guarantee convergence to the solution when one uses a (quasi-) Newton method.

We study line-search techniques in section 1. It is worth pointing out that line-search techniques may often be utilized in other methods. Section 2 will be devoted to discussion of trust-region techniques. We consider convergence properties of QN methods when line-search techniques or trust-region techniques are utilized in section 3.

Finally, termination conditions (step (2)) for algorithm 3-1 will be examined in section 4.

3-1 Line-search Techniques

Following on from the motivation given above, one may consider p_k as a search direction and x_{k+1} as a point which has been obtained on a line starting from x_k and in the direction of p_k. To find an appropriate new estimate for the solution, the expression (3-1) may be parameterized by means of a single variable (say, t (> 0)). Thus, it becomes

(3-4) $x(t) \equiv x_k + tp_k$.

In this way, if p_k is assumed to be a descent direction (which is so if the inequality

(3-5) $p_k^T g_k < 0$

holds), one may **search** along the half **line** (which starts from x_k and proceeds in the direction of p_k) and find a point such that (3-3) (and other conditions which will be considered below) hold. Hence, the term "line-search" is used to denote such an operation. To the knowledge of the author, the first researcher to utilize the parameter t in this way was Curry(1944), who used it for the steepest descent method. The condition (3-5) is automatically satisfied when M_k is pd, and, otherwise, by reversing p_k (for instance) when it is not a descent direction. One may consider (3-4) as a device for (temporarily) converting the n-variable function F(x) to the one-variable function defined by

(3-6) $f(t) \equiv F(x_k + tp_k)$,

say. Therefore, obtaining a new estimate (for the minimum of F) from x_k in direction p_k is equivalent to finding a minimum of f(t), under the condition (3-5) (or, equivalently, $f'(0) < 0$). In some early methods, $t_k^{\#}$ (known as the "step-length") was defined to be the (or a) value of t that exactly minimizes the function (3-6), subject to the condition $t \geq 0$. This type of line-search is known as an "exact" line-search. In the following notes we describe why, in current practice, exact line-searches are avoided and, instead, low-accuracy line-searches are used.

(i) In practical applications, the time spent evaluating F(x) and g(x) may well dominate the time for the whole minimization process (for QN methods). Therefore, calculating an exact $t_k^{\#}$ (which may require an infinite number of F and g evaluations) cannot be contemplated as a practical reality; even if the computation of $t_k^{\#}$ is a finite process, it is not worthwhile for points remote from the solution, where finding an accurate $t_k^{\#}$ is not needed (see Fletcher and Reeves(1964) and Dembo et al(1983)).

(ii) In practice, it is not always possible to perform an exact line-search because of rounding errors.

(iii) The main computational costs incurred in a method employing a line-search technique are the F and g evaluations. Suppose that, in one iteration of an algorithm which employs an exact line-search, one evaluates F and g m times. Employing a low-accuracy line-search, we will perform r (> 1) iterations with the same number (m) of F and g evaluations. In the case of

the exact line-search, the matrix is updated only once, whereas, in the case of the inexact line-search, the matrix is updated r times. The matrix obtained from these repeated updates may be expected to be closer to G than the matrix which has been obtained from one update in the exact line-search method. This, in turn, leads us to anticipate more rapid progress towards the minimum for the low-accuracy line-search method. Therefore, to implement a QN method (in which the evaluation of F and g dominates other costs), employing a low-accuracy line-search will be more efficient and, for a fixed number of F and g evaluations, a better approximation to G (and more rapid convergence to the solution) will be obtained in comparison with a QN method which utilizes exact line-searches.

(iv) Even though the first theories of convergence for QN methods assumed that exact line-searches were performed, the advent of low-accuracy line-searches has brought about the development of new results on convergence which take into account that such line-searches have been implemented. For example, in section 3-3, we will discuss some of these results.

For these reasons, practical implementations now impose rather weak conditions on $t_k^{\#}$, so that a suitable value can be found by calculating the function f(t) (or $F(x_k + tp_k)$) for only a few values of t. A procedure by which one obtains a value of $t_k^{\#}$ satisfying weak conditions is known as an 'inexact' line-search. Suitable conditions on $t_k^{\#}$ have been studied by many authors, including Armijo(1966), Goldstein and Price(1967), Wolfe(1969),

Wolfe(1971), Powell(1976), and Fletcher(1987), with a view to preventing oscillatory behaviour and to ensure, at every iteration, a sufficient reduction in function value. Dennis and Schnabel(1983) state two examples to show that overall convergence to the local minimum of F is not always possible, even when (3-3) is satisfied, because of oscillatory behaviour as a result of not ensuring a sufficient reduction in function value and the selection of a small step-length. Hence, if convergence is to be assured, $t_k^{\#}$ (as chosen in the line-search) must generate a sufficient decrease in the objective function, at each iteration. This requirement suggests that there must exist a close relationship between the function value, the search direction p_k along which the function is decremented and the corresponding value of $t_k^{\#}$. Hence the condition (3-3), after strengthening, becomes

(3-7) $$F_{k+1} \leq F_k + \alpha t_k^{\#} p_k^T g_k,$$

and to ensure that $t_k^{\#}$ is not too small, one imposes the condition

(3-8) $$p_k^T g_{k+1} > \beta p_k^T g_k,$$

where α and β are constants which satisfy the restrictions

(3-9) $$\alpha \in (0, 0.5]; \ \beta \in (\alpha, 1].$$

(From this point on, we refer to conditions (3-7) and (3-8) as the α-condition and β-condition, respectively.) For example, $\alpha = 0.0001$ and $\beta = 0.9$ are often stated to be suitable values (Fletcher(1987) and Gill et al(1981)). In fact, the α- and β-conditions define an interval for values of t in which any

point is "acceptable" and decreases the function value "sufficiently". The bounds of this interval on the left and right are defined by the β- and α-condition, respectively. The condition $\alpha < \beta$ (from (3-9)) guarantees the existence of an acceptable interval (or, at least, an acceptable point) for t_k^* (see the result of Dennis and Schnabel(1983) quoted below). Selection of a smaller value for β and/or a greater value for α gives a smaller interval for t_k^* and, consequently, a line-search with higher-accuracy is obtained. However, utilizing the β-condition when $\beta \to 0$ (or $\beta \to \alpha$ and $\alpha \to 0$) does not lead to an exact line-search, since any gradient vector g_{k+1} for which

$$p_k^T g_{k+1} > 0$$

will always satisfy such a β-condition. Fletcher(1987) suggests the following criterion in place of (3-8), if we wish to obtain a line-search which becomes exact as $\beta \to 0$. The criterion is:

$$|g_{k+1}^T p_k| \leq -\beta g_k^T p_k .$$

This may be expected to increase the number of F and g evaluations per iteration and (possibly) to decrease the number of iterations required for convergence.

Dennis and More(1977) prove that an iterative method is superlinearly convergent if and only if its steps asymptotically approach the Newton step in both length and direction. Since in QN methods the matrix by which we determine the search direction is expected to approximate the true Hessian (or its inverse) in the final iterations, therefore the search direction becomes

(nearly) the Newton direction. Therefore, for the step to approximate the Newton step in both length and direction in the final iterations, it is necessary to select a step-length of one, where possible. On these grounds, a common procedure in many line-search algorithms is (specifically with QN methods) to try, first, a step-length equal to 1. Computational experience has emphasized the importance of taking such a step. Failure to do so leads to forfeiture of the advantages of Newton's method near the solution (Dennis and Schnabel(1983)). If $t_k^{\#} = 1$ fails to satisfy the β-condition, we first either

(i) scale up the previous step-length by a constant factor (say, 2); or

(ii) extrapolate by means of a quadratic or cubic polynomial based on the available information at the two rightmost points (see Fletcher(1987), for instance).

The step (i) [or (ii)] is repeated until the β-condition holds. Then, if the α-condition holds in addition, we have located an acceptable point, otherwise, we may then determine a new estimate of the minimum by interpolating with a quadratic or cubic polynomial between the last two points (which constitute a bracket for the minimum). (If, for the first estimate of the step-length, the β-condition only holds, the lower bound is set to zero and the bracket is then [0,1].) If the step-length obtained fails to satisfy the α- and β-condition, the interpolation is repeated until we locate an acceptable point, by using the result of the last interpolation and one or other of

the lower bound or the upper bound (as appropriate) to constitute a new interval. Since the length of each bracket is (in theory) strictly smaller than that of its predecessor, this process must converge to a point. In practice, because of rounding errors, the new estimate of the minimum (we call it t_e) may be very close to one of the bounds, from which an infinite cycle may arise. To prevent this phenomenon, Gill et al(1981) choose t_e according to

$$t_e = t_L + (t_U - t_L)\delta, \text{ or}$$

$$t_e = t_U - (t_U - t_L)\delta,$$

when the current estimate of step-length is too close to t_L or t_U (the lower and upper bounds of the current bracket), respectively. Here δ is a constant equal, say, to 0.1. In this way, the line-search procedure is said to be "safeguarded".

Dennis and Schnabel(1983) have proved that, if $g_k^T p_k < 0$, $\{F(x_k + tp_k): t > 0\}$ is bounded below and $0 < \alpha < \beta < 1$ holds, then there exist positive values t_L and t_U such that $t_U > t_L$ and any point

$$x_{k+1} = x_k + tp_k$$

satisfies the α- and β-conditions if $t \in (t_L, t_U)$. The importance of the α- and β-conditions is illustrated by the following result: Fletcher(1987) has proved that, for a descent method (that is, in each iteration, $g_k^T p_k < 0$ holds) with an inexact line-search based on the α- and β-conditions, if $g(x)$ exists and is uniformly continuous on the level set $\{x: F(x) < F(x_0)\}$, and

if the angle between p_k and g_k is bounded away from orthogonality, then either $g_k = 0$ for some k, or $F_k \to -\infty$, or $g_k \to 0$.

When the matrix M_k (or M_k^{-1}) is pd, and its condition number K is bounded, θ_k (the angle between g_k and p_k) is bounded away from $\pi/2$ because of the following argument: we may write

$$(3\text{-}10) \quad \cos\theta_k = p_k^T g_k / \|p_k\|_2 \|g_k\|_2 = -g_k^T M_k^{-1} g_k / \left(\|M_k^{-1} g_k\|_2 \|g_k\|_2 \right).$$

Defining λ_{max} and λ_{min} as the largest and smallest eigenvalues of M_k, respectively, we may write

$$(3\text{-}11) \quad g_k^T M_k^{-1} g_k \geq \lambda_{max}^{-1} \|g_k\|_2^2$$

and

$$(3\text{-}12) \quad \|M_k^{-1} g_k\|_2 \|g_k\|_2 \leq \lambda_{min}^{-1} \|g_k\|_2^2.$$

Using (3-11) and (3-12) in (3-10), we obtain

$$(3\text{-}13) \quad \cos\theta_k \leq -\frac{\lambda_{min}}{\lambda_{max}} = -\frac{1}{K}.$$

Therefore θ_k is bounded away from $\pi/2$ as long as K is bounded above.

Finally in this section, we derive a few inequalities which will be useful in subsequent chapters. From (3-4),

$$(3\text{-}14) \quad s_k = x_{k+1} - x_k = t_k^{\#} p_k.$$

If one defines the quantities

$$(3\text{-}15) \quad \sigma_0 = s_k^T g_k \ (= t_k^{\#} p_k^T g_k); \ \sigma_1 = s_k^T g_{k+1}; \ \gamma = \sigma_1/\sigma_0;$$

$$\lambda = (F_{k+1} - F_k)/\sigma_0;$$

it follows from the α- and β-conditions and (3-9) that, if p_k (or, equivalently, s_k) is a descent direction for F at the point x_k (that is, $\sigma_0 = t_k^{\#} p_k^T g_k < 0$), then

(3-16) $\gamma < \beta \leq 1;$

(3-17) $\lambda \geq \alpha > 0.$

3-2 Trust Region Techniques

As previously indicated, in some iterations the full (quasi-) Newton step s_N is unsatisfactory. This indicates that the quadratic model does not adequately model the objective function F in a region containing x_k and $x_k + s$. Using a line-search technique, we would then retain the same step direction (or the search direction), but choose a shorter step length. This new length is determined by building a new one-dimensional quadratic or cubic model, based only on function and gradient information in the (quasi-) Newton direction. While line-search techniques are successful, they have the disadvantage that they make no further use of the n-dimensional quadratic model. In trust-region techniques, when s_N is unsatisfactory and we need to take a shorter step, we first choose a shorter step *length*, and then use the full n-dimensional quadratic model to choose the step *direction*.

We (temporarily) return to a consideration of line-search techniques to explain the motivation for trust-region techniques.

Maintaining the positive-definiteness of matrices in QN methods (see chapter 2) has the distinct advantage that it guarantees that the search direction p is a descent direction. However, it also has some drawbacks. Although the true Hessian is usually pd at the solution, this may not be the case at points far from the solution. Thus, there may be an inconsistency and incompatibility between the true Hessian and its approximation.

The development of trust-region techniques has allowed the design of algorithms which are capable of handling indefinite matrices (either the true (possibly, sparse) Hessian or a Hessian approximation). When one employs a QN method, sparsity cannot usually be preserved together with positive-definiteness, while in the context of large-scale problems, preserving the sparsity structure is of paramount importance (Conn et al(1988)). On the other hand, Newton's method may not be expensive when the Hessian matrix is sparse and one may well prefer to utilize Newton's method rather than a QN method which preserves positive-definiteness while failing to exploit the sparsity.

We recall that the quadratic model (which is the basis of (quasi-) Newton methods) is obtained by taking the first three terms of a Taylor series. However, the minimum of this model may be a point for which the objective function has not decreased sufficiently with respect to its value at the initial point of the iteration. This motivates us to consider taking a step (or correction vector) s, by minimizing a local quadratic model over a restricted ellipsoid concentrated about x_k, such that $F(x_k + s)$

is sufficiently less than $F(x_k)$. In the other words, to obtain a better estimate for the solution, one should take a *step* from x_k which is *restricted* to lie in a *region* over which we may *trust* the the quadratic model. Hence the names "restricted step" method (due to Fletcher(1987)) and "trust-region" method (due to Dennis and Schnabel(1983)), among others, have been applied to such algorithms.

The radius of this region (or the *trust-radius*) h (> 0) may be expanded and contracted until we obtain a value based upon how well the local model predicts the behaviour of the objective function. Mathematically, trust-region techniques lead to the problem of obtaining a constrained minimum of a quadratic function in each iteration; that is,

$$\min_s q_k(s) \equiv F_k + s^T g_k + 1/2\ s^T M s, \qquad (3\text{-}18)$$

$$\text{subject to} \quad \|s\| \le h,$$

where the matrix M (which may be indefinite) is the Hessian or an approximation. We restrict our attention here to the use of $\|.\|_2$. If the full (quasi-) Newton step s_N satisfies $\|s\|_2 \le h$, then we regard the step as acceptable, subject to certain conditions (see below). When s_N is not acceptable, the problem (3-18) can be written

$$\min_s q_k(s) \qquad (3\text{-}19)$$

$$\text{subject to} \quad \|s\|_2 = h$$

(Dennis and Schnabel(1983)). In an iteration, we may try

different values of h, to find a step s, such that the function is sufficiently decreased. The process by which one obtains s for a fixed value of h may be considered as a 'minor' iteration. Two aspects of the problem (3-18) are unspecified and will be considered in the following subsections. They are:

(i) how the trust-radius h is adjusted;

(ii) how s may be obtained for a fixed h, or how to characterize the vector s.

3-2-1 Adjusting the Trust-radius

If a small value is selected for h, one may

(i) make only slow progress towards the minimum;

(ii) increase the possibility of computational error due to rounding errors.

To prevent these occurrences, h should be as large as possible, subject to a certain measure of agreement existing between $q_k(s)$ and $F(x_k + s)$. This can be quantified by defining the following ratio which measures the accuracy with which $q_k(s)$ approximates $F(x_k + s)$, in the sense that the closer the ratio is to unity, the better the agreement. The ratio is defined by

(3-20) $$r_k = \Delta F / \Delta q,$$

in which

$$\Delta F = F_k - F(x_k + s)$$

and

$$\Delta q = q_k(0) - q_k(s) = F_k - q_k(s)$$

are the actual reduction in F and the corresponding predicted reduction, respectively.

The following algorithm (Sorensen(1982)) describes how one may adjust the value of h_k and predict a value for h_{k+1}.

Algorithm 3-2

(0) Let k = 0, and let $0 < \eta_1 < \eta_2 < 1$, $0 < \gamma_1 < 1 < \gamma_2$ be specified constants. Let x_0, M_0 and h_0 be given;

(1) If "convergence" then stop;

(2) Compute F_k, q_k, M_k;

(3) Compute a solution s to the problem (3-18));

(4) Evaluate $F(x_k + s)$, $q_k(s)$ and r_k using (3-20);

(5a) If $r_k < \eta_1$ then **begin** $h_k = \gamma_1 h_k$; go to 3; **end;**

else $\{r_k \geq \eta_1\}$

begin

(5b) $x_{k+1} = x_k + s$;

(5c) if $r_k > \eta_2$ then $h_k = \gamma_2 h_k$;

(5d) $h_{k+1} = h_k$; $k = k + 1$;

end

(6) Go to 1.

There are ways to update h_k at steps (5a) and (5c) which make

better use of the information available at the current point x_k. For example, Sorensen(1982) suggests that the cubic polynomial which fits $\Phi(\alpha) \equiv q_k(\alpha s)$ by interpolating $\Phi(0)$, $\Phi'(0)$, $\Phi''(0)$ and $\Phi(1)$ will have a minimum ($\alpha\hat{}$) in (0,1) when the test at step (5a) is passed. The region is contracted by setting $\gamma_1 = \alpha\hat{}$ if $\alpha\hat{}$ is not "too close" to 0 or 1. Fletcher(1987) suggests a similar approach and states that, in practice, a more sophisticated value for h_{k+1} may be chosen in an interval $(0.1\|s\|_2 , 0.5\|s\|_2)$ on the basis of a polynomial interpolation, if a similar test to (5a) is passed.

3-2-2 Characterizing s

Most trust-region techniques utilize the 2-norm for the norm in (3-19). Introducing a Lagrange multiplier ν, it is shown (see, for instance, Fletcher(1987)) that (3-18) (or (3-19)) may be converted to the problem of solving a system of linear equations in order to determine s:

(3-21) $(M + \nu I)s = -g \quad (\nu \geq 0)$.

Such methods were first suggested by Levenberg(1944) and by Marquardt(1963) in the context of nonlinear least squares. The application to general minimization is given by Goldfeld et al (1966).

In the system (3-21), the condition $\nu = 0$ will be active only if the full (quasi-) Newton step s_N is smaller (in norm) than h_k;

otherwise, it can be assumed that the constraint $\|s\|_2 = h$ is active and, in this event, the problem (3-18) can be written as (3-19).

The difficulty with solving (3-21) to obtain s in a minimization algorithm is that there is no finite method of determining ν, when $\|s_N\|_2 > h$. Therefore, in practice, other strategies for approximately solving (3-18) are used.

Powell(1970a) introduced a strategy which can be viewed as a way to change over from a step in the steepest descent direction to a step in the (quasi-) Newton direction as the method approaches the solution. This is known as the 'dogleg' strategy. Dennis and Mei(1979) modified Powell's dogleg strategy and called it the 'double dogleg'.

Before we begin to consider these strategies, it will be useful to make a few preliminary definition. A Cauchy point is defined by the minimum of $q_k(s)$ in the steepest descent direction (or by a value of t which minimizes $q_k(-tg_k)$). This is therefore given by

$$x_k - t_c g_k,$$

where

$$t_c = g_k^T g_k / g_k^T M g_k.$$

We denote the Cauchy step by s_c $(= -t_c g_k)$. The dogleg strategy may be described in a pseudo-programming language as follows:

Compute the Cauchy step s_c.

If $s_c > h$ then $x_{k+1} = x_k + c_1 s_c$

where c_1 should satisfy $\|c_1 . t_c . g_k\| = h$;

else **begin**

calculate s_N;

If $s_N < h$ then $x_{k+1} = x_k + s_N$

else $x_{k+1} = x_k + s$,

where

$$s = (1 - \theta) s_c + \theta s_N$$

(to compute θ see below)

end.

The value of θ should satisfy

$$\|s\|_2 = h = \|(1 - \theta) s_c + \theta s_N\|_2 ,$$

according to Powell(1970b). Dennis and Mei(1979) restrict θ to lie in (0,1).

Near the solution, the linear approximation of the objective function (which may be considered as the basis of the steepest descent method and, consequently, the Cauchy step) may be a fairly precise approximation. Thus, when we are fairly close to the solution, a dogleg step may have too much bias toward the gradient direction. This is the motivation for modifying Powell's dogleg step and including the "double dogleg" strategy (Dennis

and Mei(1979)), so that more bias toward the (quasi-) Newton step is obtained. The 'double dogleg' strategy can be described in the pseudo-programming language as follows:

```
Compute s_N and ||s_N||_2 .
If ||s_N||_2 ≤ h then s = s_N
   else  begin
           compute  λ (see below) and ||λs_N||;
                if ||λs_N|| ≤ h then s = h.s_N/||s_N||;
                     else
                             Compute s_c;
                             if ||s_c|| < h then compute θ (see below)
(3-22)                            s = (1 - θ)s_c + θλs_N
                             else s = -hg/||g||.
         end.
```

Now we describe the determination of λ and θ. The parameter λ defines a step length in the (quasi-) Newton direction at which the reduction of the quadratic approximation to F resulting from taking this step is the same as that from taking the Cauchy step. That is,

$$q_k(\lambda s_N) = q_k(s_c),$$

which yields

$$\lambda = 1 \mp (1 - c_1)^{1/2},$$

where

$$c_1 = \|g\|^4 / (g^T M g)(g^T M^{-1} g).$$

Dennis and Mei show that $c_1 \leq 1$ and, therefore, that λ is well-defined. To improve the strategy, Dennis and Mei suggest

$$\lambda = 1 - c_2(1 - c_1)^{1/2}, \tag{3-23}$$

where c_2 (> 0) is obtained in such a way that q_k is monotonically decreasing along the line that connects the end of vectors s_c and s_N. The parameter θ is chosen such that $0 < \theta < 1$ and $\|s\|_2 = h$.

Geometrically, when the Cauchy point is inside and the (quasi-) Newton point is outside the trust-region and $\theta \in (0,1)$, the dogleg strategy determines a point in the region through connecting the Cauchy point to the (quasi-) Newton point ((Q)N.P.), while the double dogleg strategy finds a point in the region through connecting the Cauchy point to the point at the end of the vector λs_N. In figure 3-1 (below), D.S. and D.D.S. indicate the dogleg and double dogleg step, respectively, and the point E indicates the end of the vector λs_N.

Connecting x_k, the Cauchy Point (C.P.), and the (Q)N.P. in Powell's strategy makes a shape similar to a dogleg while connecting x_k, the C.P., the point E and the Q(N).P. makes a shape similar to a double dogleg; hence the name for the respective strategies.

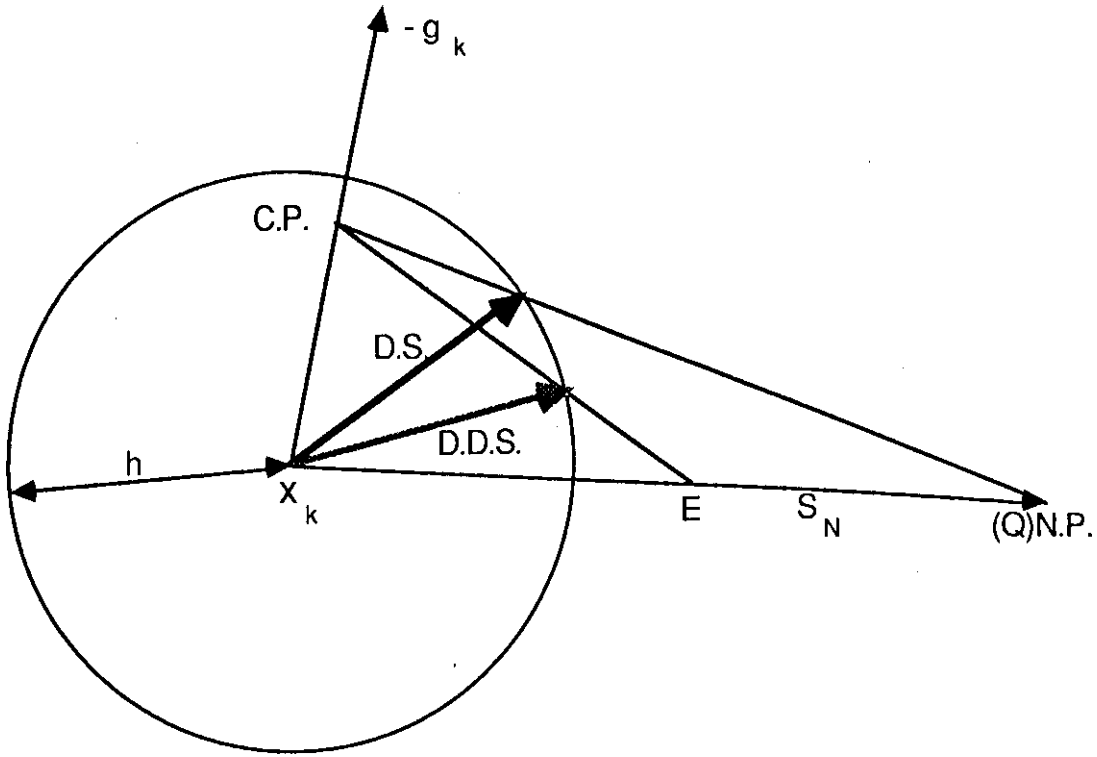


Figure 3-1

Briefly, both the dogleg and double dogleg strategies obtain s by a linear combination of the Cauchy step and the (quasi-) Newton step as follows:

$$s = (1 - \theta)s_c + \theta(\lambda s_N),$$

where $\lambda = 1$ gives the dogleg strategy and selecting λ by (3-23) (that is less than 1) gives the double dogleg strategy, to generate a step more biased towards the (quasi-) Newton step. {Since we first obtain λ (using (3-23)) and then θ by employing (3-22), therefore θ depends on λ. If we consider (λs_N) as a vector and s as another, since λ is assumed to be less than 1, the double dogleg strategy gives a step more biased towards the (quasi-) Newton step than the dogleg strategy (for which $\lambda = 1$).}

Another strategy for solving problem (3-18) is the locally constrained optimal (or "Hook") step introduced by Dennis and Schnabel(1983). The reader is referred to Sorensen(1981,1982), More(1982), Dennis and Schnabel(1983), Shultz et al(1985) and Byrd et al(1988) for more details on trust-region techniques.

3-3 Convergence Properties

After devising a method, it is necessary to consider its properties and, specifically, its local convergence behaviour. The value of a method depends upon the conditions under which it converges to the minimum and on its rate of convergence. A good method should be able to start from any initial point or, in other words, it should be globally convergent. We begin our discussion of results concerning convergence by introducing the concept of "bounded deterioration".

Definition: "Bounded Deterioration" (Dennis(1972))

A method of bounded deterioration is any iterative method

$$x_{k+1} = x_k - B_k^{-1} g_k, \quad i = 0, 1, \ldots,$$

such that, as long as x_k and G_k are defined,

$$\|B_k - G_k\| \leq c_1 + c_2 \sum_{j=1}^{k} \|x_{j-1} - x_j\|$$

for some constants c_1 and c_2.

Definition

F is bounded if the set $\{x: F(x) \leq F(x_0)\}$ is bounded.

There are a number of results concerning convergence properties for different methods. In the case of algorithms

implemented with a line-search, we restrict our attention to the Broyden family and, in particular, to the BFGS update which is generally regarded as currently the most efficient method in practice (see Fletcher(1987) for numerical results).

Powell(1972) showed that, if F is a convex and bounded function, G is continuous and $t^{\#}$ is obtained by an exact line-search, then the DFP method converges to the minimum. Powell(1971) showed that if G is pd at the minimum, satisfies a Lipschitz condition and $t^{\#}$ is obtained by an exact line-search, then the DFP method converges q-superlinearly. These results are extendable to the whole Broyden family by Dixon's theorem (Dixon(1972)), which states that, for general continuously differentiable objective function, the k-th search directions developed by any two members of Broyden family, say $p_k^{\phi_1}$ and $p_k^{\phi_2}$ are linearly dependent and, thus, that successive iterates generated by the two methods are identical. This result requires that the same starting point x_0 and initial approximation matrix H_0 are used and that line-searches are exact.

The results of Powell referred to above are idealized because exact line-searches cannot be achieved in practice for general functions, so it is important to consider what results may be obtained from members of the Broyden family when inexact line-searches are used.

For general functions, a proof of superlinear convergence for the DFP and BFGS methods has been given by Broyden, Dennis and More (1973), when the choice $t^{\#} = 1$ is made at each iteration.

This is a realistic assumption when we are near the minimum and utilize an inexact line-search technique. Broyden, Dennis and More state that, if G is nonsingular, Lipschitz continuous and pd at z, if x_0 and B_0 are sufficiently close to z and G, respectively, and if the DFP or BFGS update (both of which have bounded deterioration) is used, then $\{x_k\}$ converges to z superlinearly. Their analysis is based on the principle of bounded deterioration. They point out that any method whose Hessian approximation is of bounded deterioration is locally convergent to z at a superlinear rate.

Dennis and Walker(1981) have considered the convergence properties of different methods and have obtained generalizations of the results given by Broyden, Dennis and More. They have classified the different methods and consequently their convergence properties in terms of their derivations by minimum-norm techniques.

Powell(1976) studied the convergence properties of the BFGS update without exact line-searches. He has shown that if F is a convex function and if an inexact line-search is utilized under standard termination conditions, then $\{x_k\}$ converges to the solution. Furthermore if the Hessian is pd at z, then the rate of convergence is superlinear. According to Fletcher(1987), as yet this global convergence result has not been demonstrated for other formulas and, therefore, this result is another pointer to the superiority of the BFGS update.

So far we have been considering the convergence properties of

QN methods, when line-search techniques are applied. From now on we consider the convergence properties of QN methods (or Newton's method) when trust-region techniques are employed.

Recently Conn et al(1988) have employed the SR1 formula together with trust-region techniques, and point out that the traditional supremacy of the BFGS update is questioned by their numerical results. The numerical results documented in Conn et al(1988) show that the SR1 formula appears to be substantially more efficient in the trust-region framework than any other QN update tested.

Shultz, Schnabel and Byrd(1985) have gathered and systematized results concerning the convergence properties of QN or Newton methods which employ trust-region techniques, as given by Powell(1970a,1970b,1975), Sorensen(1982), Fletcher(1980), and others.

In the theorem below, Shultz et al discuss the convergence properties of a class of trust-region algorithms whose methods of computing M_k, s_k and h_k are unspecified. They have assumed that these quantities are computed in such a way that: (i) the algorithm is well-defined; (ii) under the conditions below, convergence is ensured. First, we quote the algorithm and the conditions.

Algorithm 3-3

(0) Choose γ_1, η_1, $\eta_2 \in (0,1)$, $x_0 \in R^n$, $h_{-1} > 0$ and let $k := 0$;

Repeat

(1) Compute a symmetric $B_k \in R^{n \times n}$;

(2) Find h_k and compute s_k satisfying

$\|s_k\| \leq h_k$ and

(a) $r_k \geq \eta_1$ (where r_k has been defined in (3-20))

and

(b) either $h_k \geq h_{k-1}$,

or $h_k \geq \|B_{k-1}^{-1} g_{k-1}\|$ with B_{k-1} pd,

or for some $h \leq h_k/\gamma_1$, $r_k < \eta_2$, or $r_{k-1} < \eta_2$;

(3) Let $x_{k+1} := x_k + s_k$ and $k := k + 1$;

Until FALSE

Before stating the conditions, we define

$\Delta = F - q(s) = -g^T s - 1/2\ s^T B s$.

Condition 1- There exist c_1 and $c_2 > 0$ such that, for all $g \in R^n$, for all symmetric $B \in R^{n \times n}$, and for all $h > 0$,

$\Delta \geq c_1 \|g\| \min\{h, c_2 \|g\|/\|B\|\}$.

Condition 2- There is a $c_3 > 0$ such that, for all $g \in R^n$, for all symmetric $B \in R^{n \times n}$, and for all $h > 0$, $\Delta \geq -c_3 \lambda_1 h^2$, where λ_1 is the smallest eigenvalue of the symmetric matrix B.

Condition 3- If B is pd and $\|-B^{-1}g\| < h$, then

$s = -B^{-1}g$.

Theorem (Shultz, Schnabel and Byrd(1985))

Let F(x) be twice continuously differentiable and bounded below, let G(x) satisfy $\|G(x)\| \leq \beta_1$ for all $x \in R^n$ where β_1 is a constant, and consider an algorithm of the type defined by Shultz, Schnabel and Byrd (above) which generates a sequence $\{x_k\}$. Then:

(i) if s satisfies condition 1 and $\|B_k\| \leq \beta_2$ for all k and a constant β_2, then g converges to zero (first order stationary point convergence);

(ii) if s satisfies condition 1 and 3, $B_k = G_k$ for all k, G(x) is Lipschitz continuous with constant L, and z is a limit point of $\{x_k\}$ with G(z) pd, then x_k converges q-quadratically to z;

(iii) if s satisfies conditions 1 and 2 , $B_k = G_k$ for all k, and x_k converges to z, then G(z) is psd (second order stationary point convergence with (i)).

Shultz et al state that the conditions which involve $\|.\|$ are not restricted to the Euclidean norm and the above theorem applies as well to rectangular trust-regions.

3-4 Termination Conditions

In this section we consider how to terminate a minimization algorithm which utilizes a (quasi-) Newton method. In general, we may stop an algorithm in one of the following cases:

(i) when time (or patience) runs out. In most iterative procedures, one quantifies available time by imposing an iteration limit;

(ii) when one arrives at the minimum. The factor that needs consideration is how to implement a suitable test in finite arithmetic and precision. A necessary condition for z to be an exact minimizer of F is $g(z) = 0$, but in an iterative and finite-precision algorithm, we may modify this condition to be $g(z) \approx 0$, that is,

(a) $\|g(z)\| \leq \varepsilon$,

together (possibly) with some or all of the following criteria:

(b) $F_k - F_{k+1} < \varepsilon$;

(c) $|x_{i,k} - x_{i,k+1}| \leq \varepsilon_i$ (for $i = 1, 2, \ldots, n$),

where $x_{i,k}$ denotes the i-th element of x_k;

(d) $\|x_{k+1} - x_k\| < \varepsilon$;

or

(e) $1/2\ g_k^T M g_k \leq \varepsilon$

(where $M = G^{-1}$ or $M = H$ in the Newton or QN method, respectively) which is a test on the predicted change in F.

The matrix M may become singular or nearly so, and consequently x_{k+1} may be very close to x_k. In this event, the

tests (b), (c), (d) and/or (e) may cause an algorithm to terminate prematurely, while the test (a) does not suffer from this disadvantage.

We may select different values for ε (or the ε_i's) for different problems. Although the criteria stated above can also occur at a maximum (except (b) which must be modified to

$$F_{k+1} - F_k < \varepsilon)$$

or saddle-point, when one utilizes a pd matrix in each iteration, it makes convergence to a maximum point impossible and to a saddle-point unlikely. In this way, each of the above criteria is considered as a necessary and sufficient condition for x to be (effectively) a local minimizer of F(x).

For more details on termination condition(s), the reader is referred to Fletcher(1987), Gill et al (1981) and Dennis et al (1983).

CHAPTER 4

NONLINEAR MODELS FOR THE GRADIENT

A fundamental characteristic of all quasi-Newton methods is the use of approximations $\{B_k\}$ to $\{G_k\}$ where G_k denotes the Hessian of F, evaluated at x_k. As indicated in chapter 2, the matrices $\{B_k\}$ are obtained via the so-called 'secant' equation:

(4-1) $B_{k+1}s_k = y_k$.

In chapter 2, we introduced the "Newton equation" (due to Ford and Saadallah(1986)) and showed that the secant equation is a particular approximation to the Newton equation, obtained by approximating the right-hand-side of (2-7) $[dg/d\tau|_{\tau=1}]$ with backward differences. Utilizing backward differences is equivalent to employing a linear model for the gradient as a function of τ:

(4-2) $g(x(\tau)) \equiv a + b\tau$.

The expansion of $g(x)$ about $x = x_{k+1}$ by a Taylor series gives

(4-3) $g(x) = g_{k+1} + G_{k+1}(x - x_{k+1}) + r$,

where

$$\|r\| = O(\|(x - x_{k+1}\|^2).$$

Substituting $x = x_k$ in (4-3) gives

(4-4) $y_k = G_{k+1}s_k - r$.

Then (4-1) and (4-4) lead to

$$r = G_{k+1}s_k - B_{k+1}s_k.$$

This relation shows that when the linear model (or a backward difference estimate for the derivative of g) is used for estimating $G_{k+1}s_k$, the closeness (or otherwise) of B_{k+1} to G_{k+1} (in the direction s_k) depends on r. When the behaviour of the function tends to be quadratic (which is the case, for example, near the solution), the behaviour of the gradient tends to be linear and, consequently, r is small and $G_{k+1}s_k$ is approximated well by $B_{k+1}s_k$. When an approximation to the minimum is poor, a quadratic model for the objective function may well not be precise and, consequently, a linear model for the gradient is unlikely to be accurate. This fact leads us to consider a nonlinear parametric model ($g(\theta,\tau)$, say) for the gradient restricted to the line L, where θ may be a single parameter or a vector. Such models were introduced by Ford and Saadallah(1986).

The parameter(s) may be held to represent the degree of "non-linearity" of the gradient or (equivalently) the degree of "non-quadraticity" of the objective function, at each iteration. By analogy with (2-8), differentiation of the nonlinear gradient model at $\tau = 1$, will yield a vector w (depending on the parameter(s) θ) which approximates the gradient derivative, as follows:

(4-5) $$w \approx dg/d\tau|_{\tau=1}.$$

This vector w will be used in place of y. Substituting w in place of y, the "secant equation " becomes

(4-6) $$B^*s = w,$$

and it is hoped that this will provide a better approximation to Newton's equation than the secant equation yields, for suitably chosen gradient models.

In order to determine the parameter(s) by which the model is defined, we derive certain equation(s) by utilizing the information that is available at each iteration. To obtain these equation(s), we may employ some or all of the following:

(a) function values;

(b) curvature estimate(s) at x_k;

(c) curvature estimate(s) at x_{k+1}.

The use of (b) has been investigated by Ford and Saadallah(1987) and Saadallah(1988) for some models and one particular curvature estimate. We shall consider other ways of finding the parameter(s) contained in the nonlinear gradient model by making use of (a), (b) and (c), either in isolation or in combination. Some of the ways in which we may combine (a), (b) and (c) are:

(d) Manipulation of the pair of equations arising from (b) and (c);

(e) Obtaining the least-squares "solution" of a pair of equations, according to the following pairings:

(e1) (a) and (b);

(e2) (a) and (c);

(e3) (b) and (c).

In the following sections, we will discuss these issues (among others):

- design of a nonlinear parametric model;

- determination of the parameter(s);
- derivation of w (the vector to be used in place of y);
- different approaches for estimating the curvature at x_k and x_{k+1};
- nomenclature for the algorithms.

For the most part, we shall discuss the algorithms we introduce in the context of using a line-search-type method. In most cases, the application in the case of trust-region-type methods is obvious and requires no further comment.

4-1 Design of a Nonlinear Parametric Model

To design a nonlinear parametric model for the gradient, we use the following criteria:

(1) It is convenient and desirable to define the model in such a way that when the parameter(s) tend to a certain value (we have chosen (somewhat arbitrarily) the value zero), the nonlinear model becomes linear. This is so because, near the solution, the objective function is likely to exhibit approximately quadratic behaviour (implying approximate linearity of the gradient), and we wish the model to be capable of representing such behaviour adequately.

(2) The model should have a functional form which is capable of analytic integration (when we wish to employ function values).

(3) It should be possible to show that the equation(s) that

result from the different approaches have solutions (under mild conditions on the data) and (preferably) a unique solution. From a computational point of view, it is highly desirable to work with equations which are easily solvable.

(4) The model should be such that no spurious stationary points or points of discontinuity are implied on the line between x and x^*; this may be achieved either by direct design of the model or by restricting the permitted values of the parameter(s): for example,

$$g(\theta,\tau) = \{a + b\tau\} \exp(\theta\tau)$$

is acceptable because $\exp(\theta\tau)$ is defined and different from zero for all finite values of θ and $\tau \in [0,1]$, while

$$g(\theta,\tau) = \{a + b\tau\}/(1 + \theta\tau)$$

is acceptable, provided that $\theta > -1$ because, for $\theta > -1$ and $\tau \in [0,1]$, $1/(1 + \theta\tau)$ is defined and different from zero. We recall that [0,1] is the interval of interest for τ on the line L (2-5), with x_k and x_{k+1} located at the two ends.

(5) If we are concerned to maintain positive definiteness in the matrices $\{B_k\}$, then it will be necessary that the condition

$$(4\text{-}7) \qquad s^T w > 0$$

is satisfied (by analogy with (2-22) in chapter 2). Again, in some cases, the model may satisfy this condition for all values of the parameter(s); in others, restrictions may be necessary.

In every iteration, we may have two or more points and their corresponding data (function-value and gradient). These points

are x_k and the point(s) which are obtained in the line-search. In iterations for which more than two points (or, equivalently, more than two gradients) are available, we may introduce a model with three vectors a, b, and c whose values may be obtained by the three values of the gradient. We have tried different models of this form, for example

$$g(\theta,\tau) = (a + b\tau + c\tau^2)\ \exp(\theta\tau);$$

in practice, we have not obtained any noticeable improvement from the use of such models, and we shall not consider them further. We therefore restrict our attention to models which are based on the two gradient vectors g_k and g_{k+1} (that is, g and g^*). Therefore our models may have two unknown vectors, a and b, to be determined by g and g^*. This is the general form of our models:

$$(4\text{-}8) \qquad g(\theta,\tau) \equiv \varphi(\theta,\tau)a + \omega(\theta,\tau)b,$$

where θ is a single parameter or a vector, and $\varphi(\theta,\tau)$ and $\omega(\theta,\tau)$ are scalar functions. Evidently, a and b are given by

$$(4\text{-}9a) \qquad a = (\omega_1 g - \omega_0 g^*)/(\varphi_0\omega_1 - \varphi_1\omega_0),$$

$$(4\text{-}9b) \qquad b = (\varphi_0 g^* - \varphi_1 g)/(\varphi_0\omega_1 - \varphi_1\omega_0),$$

where

$$(4\text{-}10a) \qquad \omega_0 = \omega(\theta,0),$$

$$(4\text{-}10b) \qquad \omega_1 = \omega(\theta,1),$$

$$(4\text{-}10c) \qquad \varphi_0 = \varphi(\theta,0),$$

$$(4\text{-}10d) \qquad \varphi_1 = \varphi(\theta,1).$$

[We assume that $\varphi_0\omega_1 \neq \varphi_1\omega_0$.]

4-2 Determination of the Parameter(s)

4-2-1 Function Values

Along the line (2-5), for $x \in R^n$, all n arguments of x (namely x_1, x_2, ..., and x_n) depend on the single variable τ. The derivative of $F(\tau)$ with respect to τ is obtained by employing the 'master' chain rule (due to Silverman(1985)), and is given by

$$(4\text{-}11) \quad F'(\tau) = dF/d\tau = \sum_{i=1}^{n} \{\partial F/\partial x_i \cdot dx_i/d\tau\}.$$

By substituting

$$\partial F/\partial x_i = g_i$$

and

$$dx_i/d\tau = s_i$$

(where g_i and s_i are, respectively, the i-th components of the vectors $g(\theta,\tau)$ and s) in (4-11), we obtain

$$F'(\tau) = \sum_{i=1}^{n} g_i s_i$$

or

$$(4\text{-}12) \quad F'(\tau) = s^T g(\theta,\tau).$$

Integrating (4-12) from $\tau = 0$ to $\tau = 1$ gives

$$\int_0^1 s^T g(\theta,\tau)\, d\tau = \int_0^1 F'(\tau)\, d\tau$$

or

$$(4\text{-}13) \qquad \int_0^1 s^T g(\theta,\tau)\,d\tau = F(1) - F(0).$$

This is the fundamental equation by which we determine (or partly determine) the parameter(s), if we are using function values. Substituting the values a and b from (4-9) in (4-8), and then substituting (4-8) in (4-13) leads to (on simplification)

$$(4\text{-}14) \qquad (\omega_1 - \gamma\omega_0)\int_0^1 \varphi(\theta,\tau)\,d\tau + (\varphi_0\gamma - \varphi_1)\int_0^1 \omega(\theta,\tau)\,d\tau$$

$$= \lambda(\varphi_0\omega_1 - \omega_0\varphi_1),$$

where ω_0, ω_1, φ_0 and φ_1 are as defined in (4-10) and σ_0, σ_1, γ and λ are as defined in (3-15).

4-2-2 Use of Curvature Estimates

In this case, we determine (by means of the gradient model) the curvature of the function in the direction s at either of the points x_{k+1} or x_k. This is set equal to a curvature estimate obtained from available function- and slope-values. Such an approach results in an equation of the form

$$(4\text{-}15) \qquad s^T dg(\theta,\tau)/d\tau = \text{curvature estimate},$$

which may be derived as follows: from (2-6), the curvature of the function in the direction s at any point on the line L is

$$s^T G s = s^T dg/d\tau,$$

where G and/or $dg/d\tau$ are evaluated at the appropriate points on L. In practice, we use one of our gradient models and a numerical

estimate for the curvature, so (4-15) is obtained. Differentiating $g(\theta,\tau)$ from (4-8) (using (4-9)) and then substituting in (4-15) gives the following simplified equations for the cases of curvature at x_k and x_{k+1}, respectively:-

(4-16) $$\left(\varphi_0'(\omega_1 - \gamma\omega_0) + \omega_0'(\varphi_0\gamma - \varphi_1)\right)/(\varphi_0\omega_1 - \varphi_1\omega_0) = c_0;$$

(4-17) $$\left(\varphi_1'(\omega_1 - \gamma\omega_0) + \omega_1'(\varphi_0\gamma - \varphi_1)\right)/(\varphi_0\omega_1 - \varphi_1\omega_0) = c_1;$$

where φ_0, φ_1, ω_0 and ω_1 are as in (4-10) and

(4-18a) $\varphi_0' = \varphi'(\theta,0)$,

(4-18b) $\varphi_1' = \varphi'(\theta,1)$,

(4-18c) $\omega_0' = \omega'(\theta,0)$,

(4-18d) $\omega_1' = \omega'(\theta,1)$,

(4-18e) $c_0 = (\text{curvature estimate at } x_k)/\sigma_0$,

(4-18f) $c_1 = (\text{curvature estimate at } x_{k+1})/\sigma_0$,

and where dashes denote differentiation with respect to τ.

4-2-3 Specialised Models

Most of the models we shall propose and study have one of the following forms, which are special cases of (4-8):

(i) $\omega(\theta,\tau) \equiv \tau\varphi(\theta,\tau)$,

leading to

(4-19) $g(\theta,\tau) \equiv (a + b\tau)\,\varphi(\theta,\tau)$.

or "the general class I";

(ii) $\omega(\theta,\tau) \equiv 1$,

yielding

(4-20) $g(\theta,\tau) = \varphi(\theta,\tau)a + b$

or "the general class II".

We now derive the versions of (4-14), (4-16) and (4-17), corresponding to these special cases of (4-8):

(i) The General Class I

From $\omega(\theta,\tau) \equiv \tau\varphi(\theta,\tau)$, we obtain

(4-21a) $\omega' \equiv \varphi(\theta,\tau) + \tau\varphi'(\theta,\tau)$,

(4-21b) $\omega_0' = \varphi_0$,

(4-21c) $\omega_1' = \varphi_1 + \varphi_1'$,

(4-21d) $\omega_1 = \varphi_1$,

(4-21e) $\omega_0 = 0$.

Therefore (4-14) becomes

$$(4\text{-}22) \quad \frac{1}{\varphi_0}\int_0^1 \varphi(\theta,\tau)\,d\tau + (\gamma/\varphi_1 - 1/\varphi_0)\int_0^1 \tau\varphi(\theta,\tau)\,d\tau = \gamma;$$

(4-16) yields

(4-23) $\varphi_0'/\varphi_0 + (\varphi_0/\varphi_1)\gamma - 1 = c_0$;

and (4-17) becomes

(4-24) $\gamma\varphi_1'/\varphi_1 - \varphi_1/\varphi_0 + \gamma = c_1$.

(ii) The General Class II

Equation (4-14), (4-16) and (4-17) become (when $\omega(\theta,\tau) \equiv 1$)

$$(4\text{-}25) \qquad \left\{\int_0^1 \varphi(\theta,\tau)\,d\tau - \varphi_0\right\}/(\varphi_1 - \varphi_0) = (\lambda - 1)/(\gamma - 1);$$

$$(4\text{-}26) \qquad \varphi_0'/(\varphi_1 - \varphi_0) = c_0/(\gamma - 1);$$

$$(4\text{-}27) \qquad \varphi_1'/(\varphi_1 - \varphi_0) = c_1/(\gamma - 1).$$

In addition to the above equations to determine the parameter(s) θ, we may employ one of the following equations, which are obtained by dividing (4-27) by (4-26) and by subtracting (4-26) from (4-27), respectively:-

$$(4\text{-}28) \qquad \varphi_1'/\varphi_0' = c_1/c_0,$$

$$(4\text{-}29) \qquad (\varphi_1' - \varphi_0')/(\varphi_1 - \varphi_0) = (c_1 - c_0)/(\gamma - 1).$$

[We assume that $\varphi_0' \neq 0$ and $\varphi_1 - \varphi_0 \neq 0$.]

4-3 Derivation of w

As indicated previously,

$$w = dg(\theta,\tau)/d\tau\big|_{\tau=1}$$

will be used instead of y to provide a better approximation to the Newton equation. For the general model (4-8), w is given by:

$$(4\text{-}30) \qquad w = \frac{1}{\varphi_0\omega_1 - \varphi_1\omega_0}\left\{(\varphi_0\omega_1' - \varphi_1'\omega_0)g^* - (\varphi_1\omega_1' - \varphi_1'\omega_1)g\right\}.$$

Now we obtain w for our two main models. For (4-19), w will be

obtained by substituting (4-21) in (4-30), resulting in:

(4-31) $w = (1 + \varphi_1'/\varphi_1)g^* - (\varphi_1/\varphi_0)g.$

For the model (4-20), w is obtained by substituting

$\omega_1 = \omega_0 = 1$

and

$\omega_0' = \omega_1' = 0$

in (4-30), yielding

(4-32) $w = \left(\varphi_1'/(\varphi_1 - \varphi_0)\right)(g^* - g) = \left(\varphi_1'/(\varphi_1 - \varphi_0)\right)y.$

Thus, this particular form of model yields a "quasi-Newton" equation which is the same as that satisfied by the Huang(1970) family:

$B^*s = \beta y,$

for some β.

4-4 Properties of the Class II

In comparing the two general classes I and II, the following observations may be made:

4-4-1:

For the class II, one may write w in terms of y as follows:

(4-33) $w = \rho(\theta).y \ [\equiv \rho.y],$

where

$$\rho = \frac{\varphi_1'}{\varphi_1 - \varphi_0},$$

from (4-32), while we are not able to write w for the class I in this form.

Let H_ϕ^{*y} and H_ϕ^{*w} be the inverse approximations generated from that member of the Broyden family (corresponding to the parameter ϕ [assumed to maintain positive-definiteness]) by means of the vectors y and w, respectively. Substituting w from (4-32) in place of y, we are able to control the ratio of $K(H_\phi^{*y})$ to $K(H_\phi^{*w})$ by using the relation obtained in the theorem below. Here, K(.) denotes the condition number with respect to the 2-norm. Because w and y are linearly dependent, the result has a particularly simple form. We first prove a Lemma.

Lemma 4-1

Let s and $z(\neq 0)$ be any two vectors in R^n, and let A be any (nxn) real, pd matrix. Then the following inequality holds:

$$\text{(4-34a)} \quad \frac{(z^T s)^2}{z^T A z} \leq s^T A^{-1} s,$$

or

$$\text{(4-34b)} \quad (z^T s)^2 \leq (z^T A z)(s^T A^{-1} s).$$

Equality holds if and only if

$$\text{(4-35)} \quad s = \beta . Az$$

for some scalar β.

Proof: Since A is pd, there exists a matrix B such that

$$\text{(4-36)} \quad A = BB^T.$$

Inequality (4-34b) is therefore equivalent to the inequality:

$$(4\text{-}37) \quad [(z^T s)^2 =]\ (z^T B B^{-1} s)^2 \leq (z^T B B^T z)(s^T B^{-T} B^{-1} s).$$

Defining

$$(4\text{-}38) \quad u = B^T z; \quad v = B^{-1} s,$$

equation (4-37) becomes

$$(4\text{-}39) \quad (u^T v)^2 \leq (u^T u)(v^T v),$$

which holds, by the Cauchy-Schwarz theorem (see, for instance, Johnson and Riess(1982)).

The fact that equality holds if $s = \beta Az$ is trivial to establish. To prove that equality holds only if $s = \beta Az$, we note that equality holds in (4-39) only if $v = \beta u$, by the Cauchy-Schwarz theorem. Therefore, using (4-38),

$$(4\text{-}40) \quad v = \beta u = \beta B^T z.$$

Premultiplying (4-40) by B, and using (4-38) and (4-36) leads to

$$Bv = \beta B B^T z,$$

or

$$BB^{-1} s = \beta Az \Rightarrow s = \beta Az. \blacksquare$$

Theorem 4-1:

Let the vector w [$\equiv \rho y$, where $\rho > 0$] be used (in place of y) to update a member of the Broyden family which maintains positive-definiteness under the condition $s^T y > 0$. Then the following inequalities hold:

$$(4\text{-}41) \quad \rho\, K(H_\phi^{*y}) \le K(H_\phi^{*w}) \le \frac{1}{\rho} K(H_\phi^{*y}) \text{ for } \rho < 1,$$

$$(4\text{-}42) \quad \frac{1}{\rho} K(H_\phi^{*y}) \le K(H_\phi^{*w}) \le \rho\, K(H_\phi^{*y}) \text{ for } \rho > 1.$$

[Remark: It is obvious that, for $\rho = 1$, $w = y$ and consequently $K(H_\phi^{*w}) \equiv K(H_\phi^{*y})$. The condition $\rho > 0$ is necessary and sufficient for H_ϕ^{*w} to be pd, if $s^T y > 0$.]

Proof: First we prove the right-hand inequalities. The Broyden family may be written as

$$(4\text{-}43) \quad H_\phi^{*y} = H_{DFP}^{*y} + \phi v v^T,$$

$$= H + \frac{ss^T}{s^T y} - \frac{Hyy^T H}{y^T H y} + \phi v v^T,$$

where v is as in (2-27). If we use the vector $w = \rho y$ in place of y, the corresponding H_ϕ^{*w} is obtained from

$$(4\text{-}44) \quad H_\phi^{*w} = H + \frac{ss^T}{s^T w} - \frac{Hww^T H}{w^T H w} + \phi v_w v_w^T,$$

where v_w is v in (4-43), except that w is used in place of y. It is easily verified that (since $w = \rho y$)

$$v_w = v,$$

and (4-44) becomes

$$(4\text{-}45) \quad H_\phi^{*w} = H + \frac{1}{\rho} \frac{ss^T}{s^T y} - \frac{Hyy^T H}{y^T H y} + \phi v v^T.$$

Substituting (4-45) from (4-43) gives

$$(4\text{-}46) \quad H_\phi^{*w} = H_\phi^{*y} + \left(\frac{1 - \rho}{\rho}\right) \frac{ss^T}{s^T y}.$$

The following formula (see, for instance, Isaacson and

Keller(1966)) may be employed to evaluate $K(H_\phi^{*w})$:

$$(4\text{-}47) \qquad K(H_\phi^{*w}) = \max_{\|z\|=1} z^T H_\phi^{*w} z \;/\; \min_{\|z\|=1} z^T H_\phi^{*w} z.$$

Using (4-46), we have

$$z^T H_\phi^{*w} z = z^T \left\{ H_\phi^{*y} + \left(\frac{1-\rho}{\rho} \right) \frac{ss^T}{s^T y} \right\} z$$

$$(4\text{-}48) \qquad = z^T H_\phi^{*y} z \left\{ 1 + \left(\frac{1-\rho}{\rho} \right) \frac{C(z)}{s^T y} \right\},$$

where

$$C(z) = \frac{(z^T s)^2}{z^T H_\phi^{*y} z}.$$

Therefore (4-47) becomes

$$(4\text{-}49) \qquad K(H_\phi^{*w}) = \max_{\|z\|=1} \left[z^T H_\phi^{*y} z \;.\; N(z) \right] / \min_{\|z\|=1} \left[z^T H_\phi^{*y} z \;.\; N(z) \right],$$

where

$$N(z) = 1 + \frac{1-\rho}{\rho} \frac{C(z)}{s^T y}.$$

Using (4-49), we may write

$$(4\text{-}50) \qquad K(H_\phi^{*w}) \leq \frac{\max_{\|z\|=1} z^T H_\phi^{*y} z}{\min_{\|z\|=1} z^T H_\phi^{*y} z} \; \frac{\max_{\|z\|=1} N(z)}{\min_{\|z\|=1} N(z)}$$

$$= K(H_\phi^{*y}) \; \frac{\max_{\|z\|=1} N(z)}{\min_{\|z\|=1} N(z)}.$$

Employing Lemma 4-1 with $A = H_\phi^{*y}$ gives

$$C(z) = \frac{(z^T s)^2}{z^T H_\phi^{*y} z} \leq s^T B_\phi^{*y} s = s^T y,$$

(using the secant equation), implying that C(z) is bounded above by $s^T y$.

Since $\rho > 0$, $\frac{1 - \rho}{\rho} > 0$ and $\frac{1 - \rho}{\rho} < 0$ are equivalent to $\rho < 1$ and $\rho > 1$, respectively. Assuming $s^T y > 0$, therefore, the maximum and minimum of N(z) depend on the sign of $\frac{1 - \rho}{\rho}$ (or, equivalently, on whether $\rho < 1$ or $\rho > 1$). To prove the right-hand inequalities in (4-41) and (4-42), we consider the maximum and minimum of N(z) in the following two cases:

(i) $\frac{1 - \rho}{\rho} > 0$ (or $\rho < 1$): in this case, the maximum and minimum of N(z) are given by

$$\max N(z) = 1 + \left(\frac{1 - \rho}{\rho}\right)\frac{s^T y}{s^T y} = \frac{1}{\rho}$$

and

$$\min N(z) = 1;$$

consequently, from (4-50),

$$K(H_\phi^{*w}) \leq \frac{1}{\rho} K(H_\phi^{*y}) \quad \text{for } \rho < 1.$$

which is the right-hand inequality in (4-41).

(ii) $\frac{1 - \rho}{\rho} < 0$ (or $\rho > 1$): in this case, the maximum and minimum of N(z) are given by

$$\max N(z) = 1,$$

and

$$\min N(z) = 1/\rho.$$

Therefore, from (4-50),

$$K(H_\phi^{*w}) \leq \rho K(H_\phi^{*y}) \quad \text{for } \rho > 1,$$

which is the right-hand inequality in (4-42).

We now prove the left-hand inequalities in (4-41) and (4-42). Using (4-46)

$$(4\text{-}46') \qquad H_{\phi}^{*y} = H_{\phi}^{*w} + \left(\frac{\rho - 1}{\rho}\right)\frac{ss^T}{s^Ty}.$$

The following formula (which is similar to (4-47)) may be employed to evaluate $K(H_{\phi}^{*w})$:

$$(4\text{-}47') \qquad K(H_{\phi}^{*y}) = \max_{\|z\|=1} z^TH_{\phi}^{*y}z \;/\; \min_{\|z\|=1} z^TH_{\phi}^{*y}z.$$

Using (4-46'), we have

$$z^TH_{\phi}^{*y}z = z^T\left\{H_{\phi}^{*w} + \left(\frac{\rho - 1}{\rho}\right)\frac{ss^T}{s^Ty}\right\}z$$

$$(4\text{-}48') \qquad = z^TH_{\phi}^{*w}z\left\{1 + \left(\frac{\rho - 1}{\rho}\right)\frac{C(z)}{s^Ty}\right\},$$

where

$$C(z) = \frac{(z^Ts)^2}{z^TH_{\phi}^{*w}z}.$$

Therefore (4-47') becomes

$$(4\text{-}49') \qquad K(H_{\phi}^{*y}) = \max_{\|z\|=1}\left[z^TH_{\phi}^{*w}z \;.\; N(z)\right] / \min_{\|z\|=1}\left[z^TH_{\phi}^{*w}z \;.\; N(z)\right],$$

where

$$N(z) = 1 + \frac{\rho - 1}{\rho}\,\frac{C(z)}{s^Ty}.$$

Using (4-49'), we may write

$$(4\text{-}50') \qquad K(H_{\phi}^{*y}) \le \frac{\max\limits_{\|z\|=1} z^TH_{\phi}^{*w}z}{\min\limits_{\|z\|=1} z^TH_{\phi}^{*w}z} \;\frac{\max\limits_{\|z\|=1} N(z)}{\min\limits_{\|z\|=1} N(z)}$$

$$= K(H_\phi^{*w}) \frac{\max_{\|z\|=1} N(z)}{\min_{\|z\|=1} N(z)}.$$

Employing Lemma 4-1 with $A = H_\phi^{*w}$ gives

$$C(z) = \frac{(z^T s)^2}{z^T H_\phi^{*w} z} \leq s^T B_\phi^{*w} s = s^T w = \rho s^T y,$$

(using (4-6) and (4-33)), implying that $C(z)$ is bounded above by $\rho s^T y$.

Since $\rho > 0$, $\frac{\rho - 1}{\rho} > 0$ and $\frac{\rho - 1}{\rho} < 0$ are equivalent to $\rho > 1$ and $\rho < 1$, respectively. Assuming $s^T y > 0$, therefore, the maximum and minimum of $N(z)$ depend on the sign of $\frac{\rho - 1}{\rho}$ (or, equivalently, on whether $\rho > 1$ or $\rho < 1$). To prove the left-hand inequalities in (4-41) and (4-42), we consider the maximum and minimum of $N(z)$ in the following two cases:

(i) $\frac{\rho - 1}{\rho} > 0$ (or $\rho > 1$): in this case, the maximum and minimum of $N(z)$ are given by

$$\max N(z) = 1 + \left(\frac{\rho - 1}{\rho}\right)\frac{\rho s^T y}{s^T y} = \rho$$

and

$$\min N(z) = 1;$$

consequently, from (4-50'),

$$\cdot \frac{1}{\rho} K(H_\phi^{*y}) \leq K(H_\phi^{*w}) \text{ for } \rho < 1.$$

which is the left-hand inequality in (4-41).

(ii) $\frac{\rho - 1}{\rho} < 0$ (or $\rho < 1$): in this case, the maximum and minimum of $N(z)$ are given by

$$\max N(z) = 1,$$

and

$$\min N(z) = \rho.$$

Therefore, from (4-50'),

$$\rho\, K(H_\phi^{*y}) \le K(H_\phi^{*w}) \quad \text{for } \rho > 1,$$

which is the left-hand inequality in (4-42).■

Comment: From numerical experience, we have found that it is desirable to restrict ρ to lie (say) in the interval [0.2,4.0]. When $\rho < 0.2$ and $\rho > 0$, the choice $\rho = 0.2$ seems to be indicated, while, if $\rho > 4.0$, ρ reset to the value 4.0 is, similarly, the obvious choice. In the case $\rho < 0$, we might consider setting ρ to either 1 (which gives the standard methods) or 0.2. Our experience has shown that the latter choice has a strong tendency to produce better results than the former, and we will therefore adopt this choice in future algorithms (where appropriate).

4-4-2:

It may be noted that equations (4-25) and (4-26) are easier, in some respects, to solve than equations (4-22) and (4-23), respectively. Let write (4-25) (or (4-26)) in the form

(4-51) $\quad u(\theta) = \delta_1,$

and (4-22) (or (4-23)) in the form

(4-52) $\quad v(\theta) = \delta_2.$

Since the function u does not depend on data from the problem

(unlike v, which depends on γ), we may observe the following:

(1) we may more easily analyse $u(\theta)$ to find out whether solutions exist or not;

(2) it could well be worthwhile to construct some kind of approximation (call it U) to the inverse function u^{-1} and use this to locate an approximation to the solution of (4-25) (or (4-26)) very rapidly:

$$\theta \approx U(\delta_1).$$

To solve a (one-variable) nonlinear equation which yields the parameter θ, we use an iterative method (say, Newton's method). To start the iterative method, one needs an initial point (say, θ_0). If θ_0 is selected as close to the root as possible, less computation may be expected and, secondly, the possibility of divergence is decreased.

One approach to selecting θ_0 is common for both equations (4-51) and (4-52): we obtain an approximation to the equation by means of a truncated Taylor series and solve the resulting equation for θ directly.

Because $u(\theta)$ is often either monotonically ascending or descending, another choice for θ_0 may be obtained as follows: we construct a "look-up" table from a number of values of θ (say, m) in an interval $[\theta_L, \theta_U]$ and the corresponding values of $u(\theta)$. Then, we test whether δ_1 (in (4-51)) does lie in $[u(\theta_L), u(\theta_U)]$ in the case of ascending u (or in $[u(\theta_U), u(\theta_L)]$, in the case of descending u). If it is in the interval, we find θ_0 (or the nearest θ corresponding to δ_1) in the table by means of a binary

search (using δ_1), which needs at most $\log_2 m$ searches, otherwise, we do not solve the equation, but set $w = y$. Two questions arise:

(i) how to select the number of values (m) of θ in the interval; and

(ii) how to select a suitable interval $[\theta_L, \theta_U]$ for θ.

To obtain an efficient binary search, m should be 2^r for some integer r. Selecting a large value of m implies more searches and a more precise estimate θ_0, while a small value of m leads to fewer searches and a less precise θ_0. Therefore there is a tradeoff between the precision obtained for θ_0 and the work involved in constructing the look-up table. We have used m = 128.

We now consider the issue of selecting an interval $[\theta_L, \theta_U]$ for θ. Considering (4-41) and (4-42), we may define what may be regarded as an acceptable range of values for $\rho(\theta)$ (for example,

$0.2 \leq \rho(\theta) \leq 4.0$),

which will restrict $k(H_\phi^{*w})$ in comparison with $K(H_\phi^{*y})$ by the theorem above. These restrictions on $\rho(\theta)$ give an interval for θ, as we now show. To obtain θ_L and θ_U, we substitute the corresponding expression for $\rho(\theta)$ in the above inequalities. If $\rho(\theta)$ is a simple expression, (for example, $\rho(\theta) \equiv \theta + 1$ in the model C or $\rho(\theta) \equiv \exp(\theta)$ in the model D), we obtain θ_L and θ_U in a straightforward manner. For example, in the latter case, we require

$0.2 \leq \exp(\theta) \leq 4.0$

which gives

$$\theta_L = \ln(0.2)$$

and

$$\theta_U = \ln(4.0).$$

If $\rho(\theta)$ is such that we cannot easily obtain θ_L and θ_U, we may compute θ_L and θ_U numerically. For example, for the model B, in which $\rho(\theta) = \theta / (1 - \exp(-\theta))$, we should have

$$0.2 \le \left(\theta/[1 - \exp(-\theta)]\right) \le 4.0.$$

These inequalities gives $\theta_L = -2.66039...$ and $\theta_U = 3.92069...$ Having determined the interval for θ, and having selected m, we may construct the look-up table using the m values of θ and $u(\theta)$.

4-5 Estimation of Curvature at x and x*

Curvature estimates at x and/or x^* along the line which connects x and x^* may be evaluated by means of one of the following:

(we (temporarily) define $\pi(\tau) \equiv F(x + \tau s)$.)

(a) Using an interpolating cubic polynomial:

(4-53) $\rho(\tau) \equiv \zeta_3\tau^3 + \zeta_2\tau^2 + \zeta_1\tau + \zeta_0$, say,

between $\tau = 0$ and $\tau = 1$, employing the four items of data $\pi(0)$, $\pi(1)$, $\pi'(0)$ and $\pi'(1)$. We then evaluate the second derivative of this polynomial

(4-54) $\rho''(\tau) \equiv 6\zeta_3\tau + 2\zeta_2$

at $\tau = 0$ or $\tau = 1$, as appropriate.

Substituting $\tau = 0$ and $\tau = 1$ in (4-53) and in the expression for the derivative

(4-55) $\rho'(\tau) = 3\zeta_3\tau^2 + 2\zeta_2\tau + \zeta_1$

gives the following linear system of equations:

(4-56a) $\zeta_3 + \zeta_2 + \zeta_1 + \zeta_0 = \pi(1)$,

(4-56b) $\zeta_0 = \pi(0)$,

(4-56c) $3\zeta_3 + 2\zeta_2 + \zeta_1 = \pi'(1)$,

(4-56d) $\zeta_1 = \pi'(0)$.

Solving this linear system of equations to find ζ_3 and ζ_2, substituting ζ_3 and ζ_2 in (4-54) and finally substituting $\tau = 0$ and $\tau = 1$, we obtain these estimates for the values of the curvature:

(curvature at x) $\approx 6[\pi(1) - \pi(0)] - 2\sigma_1 - 4\sigma_0$

or

(4-57) $= (6\lambda - 2\gamma - 4)\sigma_0$

(using (3-15));

(curvature at x^*) $\approx 4\sigma_1 + 2\sigma_0 - 6[\pi(1) - \pi(0)]$

(4-58) $= (4\gamma + 2 - 6\lambda)\sigma_0$

(using (3-15)).

(b) Using an interpolating quadratic polynomial

between $\tau = 0$ and $\tau = 1$ based on a selection from the four items of data $\pi(0)$, $\pi(1)$, $\pi'(0)$ and $\pi'(1)$. We again evaluate the second derivative of this polynomial at $\tau = 0$ and $\tau = 1$. If the quadratic polynomial is written in the form

$$\nu(\tau) \equiv \zeta_2\tau^2 + \zeta_1\tau + \zeta_0,$$

four equations are obtained as follows:

(4-59a) $\pi(0) = \zeta_0,$

(4-59b) $\pi(1) = \zeta_2 + \zeta_1 + \zeta_0,$

(4-59c) $\pi'(0) = \zeta_1,$

(4-59d) $\pi'(1) = 2\zeta_2 + \zeta_1.$

Using (4-59a), (4-59b), 4-59c) and (3-15), the following estimate is obtained:

(4-60) (curvature estimate at x) $= 2(\lambda - 1)\sigma_0.$

Using (4-59c) and (4-59d), we obtain:

(4-61) (curvature estimate at x or x^*) $= (\gamma - 1)\sigma_0.$

Using (4-59a), (4-59b) and (4-59d), there results:

(4-62) (curvature estimate at x^*) $= 2(\gamma - \lambda)\sigma_0.$

(c) Using $s^TBs = -t^\#\sigma_0$ (that is, an approximation to the curvature s^TGs of F in the direction s at x. $t^\#$ is the "accepted" value of t, as described in chapter 3, when searching along the line x + tp.)

The value of s^TBs may be derived as follows: from (2-1):

(4-63) $Bp = -g.$

By the definition of s and p,

(4-64) $s = t^\#p.$

Equations (4-63) and (4-64) yield

$$Bs = -t^\#g;$$

this leads to

(4-65) $s^T Bs = -t^* s^T g$

or

$$s^T Bs = -t^* \sigma_0,$$

by (3-15).

This curvature estimate has been used by Ford and Saadallah(1987) and Saadallah(1988) in most of their methods.

We employ only positive estimates of curvature at x^* for the following reason:

differentiating F (or π) with respect to τ and evaluating at $\tau = 1$, we obtain

$$\pi''(1) = s^T dg/d\tau|_{\tau=1}.$$

The right hand side of this equation represents the curvature of F at x^* and is estimated by $s^T w$, by equations (4-5) and (4-12), respectively. Therefore it is indicated that, if we require a positive value for $s^T w$ in order to maintain positive definiteness of the matrix B^*, the curvature estimate at x^* should be positive.

Since the estimates of curvature by approach (a) (that is, through formula (4-57) and (4-58)) are obtained via four items of data (as compared with approach (b), which employs only three or two items of data), it is expected that approach (a) will generally be more effective than approach (b). We will discuss the relative effectiveness of these different approaches to estimating curvature in chapter 7.

In the equations defining the required parameter(s) which

employ curvature estimates at x or x^*, we need finally to provide values for c_0 and c_1 (see (4-18e) and (4-18f)). Dividing the right-hand-side of equations (4-61), (4-60), (4-57), and (4-65) by σ_0 gives different values for c_0 (using (4-18)), which are denoted by c0q1, c0q2, c0c and c0t, respectively. They are:

(4-66a) $c0q1 = \gamma - 1,$

(4-66b) $c0q2 = 2(\lambda - 1),$

(4-66c) $c0c = (6\lambda - 2\gamma - 4),$

(4-66d) $c0t = -t^{\#}.$

Dividing the right-hand-sides of equations (4-61), (4-62) and (4-58) by σ_0 gives different values for c_1 (using (4-18)) which we denote by c1q1, c1q2 and c1c, respectively. They are:

(4-67a) $c1q1 = \gamma - 1,$

(4-67b) $c1q2 = 2(\gamma - \lambda),$

(4-67c) $c1c = 4\gamma + 2 - 6\lambda.$

Substituting c0q1 and c1q1 in the corresponding equations often leads to $\theta = 0$ (and, therefore, w = y) because the underlying function model is quadratic. Hence we do not employ these estimates in our algorithms, unless otherwise stated.

Since $\sigma_0 < 0$, using (4-18), we may conclude that c_i will be negative if the curvature estimate (at x or x^*) is positive. For an estimate of c_i to be accepted for use in a method, two conditions should be satisfied. The first condition determines whether the equation (which defines the parameter(s)) has a solution or not, while the second (which is only relevant to c_1)

is concerning with maintaining the property $s^T w > 0$. If either of these conditions is not satisfied, we set $w = y$ or $w = \rho_1 y$, where ρ_1 is a suitable coefficient for y (to be defined later). Different single choices for c_0 or c_1 (as appropriate) for a particular model leads, of course, to different results. An alternative means of utilizing the available estimates is as follows: in a specific iteration, we start with (what is believed to be) the best estimate of c_i. If this value of c_i is acceptable, we determine the parameter(s) and, hence, the vector w. Otherwise, we may select the next best c_i and repeat this process until either we obtain an acceptable c_i or all of the values of c_i have been rejected. In the latter case, we set $w = y$. By this means we obtain an estimate for c_i (if any) which, by analogy with similar issues in computer science, (see, Aho et al(1987)) we call "the greedy value" of c_i. We denote these greedy choices of c_0 and c_1 by "c0gr" and "c1gr", respectively.

4-6 Nomenclature for the Algorithms

To enable the reader to recognize our algorithms more easily (in particular, when the tables of numerical results in chapter 7 are considered), we will denote our algorithms by means of a simple naming scheme. The first character defines the gradient model which is employed (except for one group of algorithms which arise from class II and which do not depend on the precise form of the model; for these algorithms, we use the letter "J"). After

the first character, we denote (by the next character or string) which item(s) of data (that is, function value, curvature estimate at x or curvature estimate at x^*) is/are utilized in the method. For the algorithms which utilize function value (or formula (4-22) or (4-25)), the character is "f". In general, the strings "c0" and "c1" are used to specify algorithms which employ curvature estimate at x and x^*, respectively. Furthermore, for the algorithms which utilize curvature estimates at x or x^*, we use the strings given in the following table to indicate which curvature estimate is in use:

curvature estimate using	string
(4-66b)	"c0q"
(4-66c)	"c0c"
(4-66d)	"c0t"
c0gr (defined above)	"c0gr"
(4-67b)	"c1q"
(4-67c)	"c1c"
c1gr (defined above)	"c1gr"

For example, Ac0c denotes an algorithm which utilizes the model A together with curvature estimate at x using c0c. The algorithms which utilize both c_0 and c_1 use the greedy values of c_0 and c_1. The strings "c-01", "c/01" and "cLs" are used to specify the

algorithms which employ the equations (4-29), (4-28) and the solution by least squares of (4-26) and (4-27), respectively. The string "c0f" is used to specify an algorithm which employs function value and curvature estimate at x.

Since, in practice, using c0t has been found to be less efficient than the use of other values of c_0, we do not employ c0t in those algorithms which employ both c_0 and c_1. For those algorithms which need both c_0 and c_1, we utilize either (4-66b) with (4-67b), or (4-66c) with (4-67c). In either case, we have

$$c_1 + c_0 = 2(\gamma - 1)$$

or

$$c_0 = 2(\gamma - 1) - c_1. \tag{4-68}$$

Instead of evaluating both c_0 and c_1 independently, we therefore evaluate the greedy value of c_1 and then compute c_0 from (4-68).

CHAPTER 5

EXPONENTIAL MODELS

In this chapter we wish to consider four models together with equations which employ function values, curvature estimate at x and curvature estimate at x^*. All of these methods (derived in chapter 4) obtain the parameter θ and then the vector w, which may be used in place of the vector y. These models are, first,

(5-1) $g(\theta,\tau) \equiv (a + b\tau)\exp(\theta\tau)$, (model "A")

(which belongs to the general class I) and, second, the following models from the general class II:

(5-2) $g(\theta,\tau) \equiv \exp(\theta\tau)a + b$, (model "B"),

(5-3) $g(\theta,\tau) \equiv \tau \exp(\theta\tau)a + b$, (model "C"),

and

(5-4) $g(\theta,\tau) \equiv (\tau - 1)\exp(\theta\tau)a + b$, (model "D").

All of these models have the property that no spurious stationary points or points of discontinuity are introduced on the line L (2-5), because they involve $\exp(\theta\tau)$ which is defined and different from zero for all finite values of θ and τ.

The organization of this chapter is as follows: section 1 is allocated to consideration of the model A. We examine the general model II together with curvature estimates at x^* in section 2. In sections 3, 4 and 5, we will study different algorithms which employ the models B, C, and D, respectively. Finally lemmas and computational issues are collected for consideration in section 6.

5-1 The Model A

This model was used by Ford and Saadallah(1987) to develop the first of their algorithms, which employs curvature estimate at x using c0t (from (4-66d)). We will derive a number of algorithms which utilize function values and curvature estimate at x^*. Also, we consider algorithms which utilize curvature estimate at x according to different choices of c_0 including c0t, for the sake of comparison with Ford and Saadallah's algorithm.

The model A is a member of the general class I with

(5-5a) $\varphi(\tau) \equiv \exp(\theta\tau)$.

We will need the following values:

(5-5b) $\varphi_0 = 1;\ \varphi_1 = \exp(\theta);\ \varphi_0' = \theta;\ \varphi_1' = \theta.\exp(\theta)$,

where the left-hand-sides of these expressions are defined in (4-10) and (4-18).

First of all, the common parts of all algorithms related to the model A are considered. Using the relevant parts of (5-5) and (4-31), after simplification, gives the following expression for w:

(5-6) $w = (1 + \theta)g^* - \exp(\theta)g$.

Since the condition $s^Tw > 0$ (which is important for maintaining positive-definiteness of the matrix B) may depend on the equation by which we obtain θ, we consider this issue separately for each algorithm.

5-1-1 Algorithm Af (Ford and Ghandhari(1989))

This algorithm is based on the model A and employs function values. Using (4-22), (5-1) and (5-5) leads to the following equation:

$$(5\text{-}7) \qquad \int_0^1 \exp(\theta\tau)\,d\tau + (\gamma \exp(-\theta) - 1)\int_0^1 \tau \exp(\theta\tau)\,d\tau = \lambda,$$

where γ and λ have been defined in chapter 3.

It is useful, at this point, to define the functions

$$(5\text{-}8) \qquad \psi_i(\theta) \equiv \int_0^1 \tau^i \exp(\theta\tau)\ d\tau, \text{ for } i = 0,\ 1 \text{ and } 2.$$

These functions will be used in this algorithm and in a number of algorithms related to the other models. We may write $\psi_0(\theta)$ and $\psi_1(\theta)$ explicitly (if $\theta \neq 0$) as

$$(5\text{-}9) \qquad \psi_0(\theta) \equiv [\exp(\theta) - 1]/\theta;$$

$$\psi_1(\theta) \equiv [\theta \exp(\theta) + 1 - \exp(\theta)]/\theta^2.$$

The limiting values of these expressions are easily determined to be:

$$(5\text{-}10) \qquad \lim_{\theta\to 0} \psi_0(\theta) = 1; \qquad \lim_{\theta\to 0} \psi_1(\theta) = 0.5.$$

Then, from (5-7) we obtain (using (5-9))

$$(5\text{-}11) \qquad \lambda = \psi_0(\theta) + [\gamma \exp(-\theta) - 1]\psi_1(\theta),$$

or

$$(5\text{-}12) \qquad \lambda = \gamma\Upsilon(-\theta) + \Upsilon(\theta) \quad [\equiv \nu(\theta), \text{ say}],$$

where

(5-13) $\Upsilon(\theta) \equiv \psi_0(\theta) - \psi_1(\theta) \equiv \exp(\theta)\ \psi_1(-\theta)$.

Equation (5-12) provides the means whereby (under certain conditions) θ may be determined. In order to identify the conditions under which equation (5-12) has a solution, we observe that

(5-14) $\lim_{\theta\to+\infty} \Upsilon(\theta) = +\infty; \quad \lim_{\theta\to-\infty} \Upsilon(\theta) = 0.$

It follows that, if $\gamma < 0$, equation (5-12) will have a solution for all values of λ. Furthermore, as we show in the last section of this chapter (Lemma 5-1)), $\Upsilon'(\theta)$ is positive for all values of θ, so that the derivative of the function $\nu(\theta)$ defined by (5-12) is also positive (when $\gamma < 0$). Therefore, (5-12) has one and only one solution when $\gamma < 0$. If $\gamma = 0$, (5-14) shows that (5-12) will have a solution if $\lambda > 0$. This condition is, in fact, satisfied (see (3-17)). Again, the solution is unique. Finally, in the case where γ is positive, we may note that

$$\nu'(\theta) \equiv \Upsilon'(\theta) - \gamma\ \Upsilon'(-\theta)$$

$$\equiv (1 - \gamma \exp(-\theta))\ \Upsilon'(\theta).$$

Since $\Upsilon'(\theta)$ is always positive, $\nu(\theta)$ has only one stationary point (at $\theta_m = \ln \gamma$), which is easily shown to be a minimum. The least value of $\nu(\theta)$ is, therefore,

$$\nu(\theta_m) = (\gamma - 1)/\ln\gamma,$$

which is always positive. Hence, equation (5-12) will have a solution when γ is positive if and only if

(5-15) $\lambda \geq (\gamma - 1)\ln \gamma$.

In this case, two solutions will exist and we choose the one to the right of the minimum θ_m, since (3-16) implies that $\theta_m < 0$. This is important because, as the minimum is approached, F may be expected to behave increasingly like a quadratic function on the line L [equation (2-5)], for which there holds

(5-16) $2\lambda = \gamma + 1$,

(using (4-66a) and (4-66b)). When this relation is substituted into (5-12), the solution $\theta = 0$ results (implying that w = y, as desired). It is therefore necessary (particularly as the minimum is approached) to ensure that, when choosing between the two roots of equation (5-12) in the case when $\gamma > 0$, we select the root which is nearer to zero and hence, will yield (from equation (5-6)) a vector w which more closely resembles y.

To obtain $s^T w > 0$ (the importance of which was previously stated) it follows, from (5-6) that (using (3-15))

(5-17) $s^T w = -\sigma_0(\exp(\theta) - (1+\theta)\gamma)$.

Assuming that p is a descent direction for F at x (so that σ_0 is negative), it may be seen that condition

$$\exp(\theta) > (1 + \theta)\gamma$$

is sufficient to ensure that $s^T w$ is positive. Since $\gamma < 1$ (from (3-16)) and $\exp(\theta) \geq (1 + \theta)$, it follows that

$$\exp(\theta) \leq (1 + \theta)\gamma$$

only if $\theta < -1$ and $\gamma < 0$. In the event that $\theta < -1$, because there would be a risk that $s^T w$ might be negative and since the

coefficient of g^* in (5-6) would then be negative (in sharp contrast to y [= $g^* - g$]), we modify θ to have the value -1 before proceeding to compute w and B^*. On the basis of the tests we have performed to date, such a modification is very rarely necessary (we have found it to be required only once in 100 iterations, on average). A viable alternative would be to take $\theta = 0$, corresponding to a "standard" quasi-Newton update for the current step. With either modification, the new algorithm is guaranteed to generate pd matrices, as long as a suitable updating formula is employed (for example, those which (in terms of y) are commonly known as the DFP and BFGS updates or which, more generally, are members of the Broyden convex family (see chapter 2)).

(Note: in the paper by Ford and Ghandhari(1989), this algorithm is referred to as "EXP1".)

5-1-2 Algorithm Ac0

This algorithm obtains the parameter θ by employing a curvature estimate at x and the model A. Using (5-5) and (4-23) gives

(5-18) $c_0 = \theta - 1 + \gamma\exp(-\theta)$ [$\equiv \delta(\theta)$, say],

where c_0 and γ have been defined in chapters 4 and 3, respectively. Equation (5-18) determines the value of θ, in general. To identify the conditions under which the equation has a solution, we observe that, for $\gamma \leq 0$:

$$\lim_{\theta\to+\infty} \delta(\theta) = +\infty; \quad \lim_{\theta\to-\infty} \delta(\theta) = -\infty;$$

and, for $\gamma > 0$:

$$\lim_{\theta\to+\infty} \delta(\theta) = +\infty; \quad \lim_{\theta\to-\infty} \delta(\theta) = +\infty.$$

It follows that, if $\gamma \leq 0$, equation (5-18) will have a solution for all values of c_0 because

$$\delta'(\theta) = 1 - \gamma\exp(-\theta) > 0,$$

and $\delta(\theta)$ is therefore an increasing function of θ. In the case $\gamma > 0$, we may note that $\theta_m = \ln \gamma$ is a stationary point of $\delta(\theta)$. Since

$$\delta''(\theta) = \gamma\exp(-\theta) > 0,$$

δ is strictly convex and, consequently, the point

$$(\theta_m = \ln \gamma, \ \delta(\theta_m) = \ln \gamma)$$

is the minimum of $\delta(\theta)$. Therefore, (5-18) will have two roots if the condition

$$c_0 > \ln \gamma$$

holds and one (double) root if $c_0 = \ln \gamma$. Again, for the reasons discussed above, when there are two roots we choose that root which is nearer to zero. For a quadratic function, substituting the values of c_0 given by (4-66) (apart from the excluded case (4-66d)) into (5-18) leads to $\theta = 0$, in each case, which implies $w = y$, as desired.

To ensure that the condition $s^T w > 0$ holds, we refer, first, to the discussion in section 5-1-1, which indicated that it was desirable to restrict θ (in the case of this model) to satisfy

$\theta \geq -1$. We may, in the present case, also utilize (5-18), as follows: using (5-17) and substituting the value of γ from (5-18) gives

$$s^T w = -\sigma_0 \exp(\theta)(\theta^2 - (\theta + 1)c_0).$$

Assuming $\sigma_0 < 0$ and $\theta > -1$, we may conclude that a sufficient condition to ensure that $s^T w > 0$ is that $c_0 < 0$, because, in that case,

(5-19) $$\pi(\theta) \equiv \theta^2 - (\theta + 1)c_0 > 0$$

holds. More precisely, (5-19) holds if $\theta > -1$ and

(5-20) $$c_0 < \left(\frac{\theta^2}{1 + \theta}\right) \quad (\geq 0).$$

However, c_0 will be negative if we require any curvature estimate at x to be positive (as we usually do), so that the condition (5-20) will then be satisfied.

5-1-3 Algorithm Ac1

This algorithm employs a curvature estimate at x^* and the model A to obtain θ. Using (5-1), (5-5) and (4-24) gives

(5-21) $$c_1 = \gamma(\theta + 1) - \exp(\theta) \quad [\equiv \zeta(\theta), \text{ say}],$$

where c_1 has been defined in chapter 4. In order to determine the conditions under which equation (5-21) has a solution, we note that, for $\gamma < 0$,

$$\lim_{\theta \to -\infty} \zeta(\theta) = +\infty; \qquad \lim_{\theta \to +\infty} \zeta(\theta) = -\infty;$$

and, for $\gamma > 0$,

$$\lim_{\theta \to -\infty} \zeta(\theta) = -\infty; \quad \lim_{\theta \to +\infty} \zeta(\theta) = -\infty.$$

(We consider the case $\gamma = 0$ below.)

It follows that, if $\gamma < 0$, the equation (5-21) will have a solution for all values of c_1 because $\zeta'(\theta) = \gamma - \exp(\theta) < 0$ and $\zeta(\theta)$ is monotonically decreasing, in this case. For the case when $\gamma > 0$, we may note that the stationary point of ζ (at $\theta_m = \ln \gamma$) is a maximum (because $\zeta''(\theta) = -\exp(-\theta) < 0$). The maximum value of $\zeta(\theta)$ is, therefore,

$$\zeta(\theta_m) = \gamma.(\ln \gamma),$$

which is negative, because $\gamma < 1$ (by (3-16)). Hence equation (5-21) will have two solutions when $c_1 < \gamma.(\ln \gamma)$ and one solution if

$$c_1 = \gamma.(\ln \gamma).$$

In the case when two solutions exist, by analogy with the discussions in section 5-1-1 and 5-1-2, we choose that root which is nearer to zero.

For the case $\gamma = 0$, the equation (5-21) has a solution if $c_1 < 0$ (which, as we have seen, is equivalent to the condition $s^T w > 0$). Therefore, we will always impose the condition $c_1 < 0$ from now on.

In the case of a quadratic function, substituting the values for c_1 given by (4-67) into (5-21) leads to $\theta = 0$, in each case, which implies $w = y$, as desired.

5-2 The General Class II with Curvature Estimate at x^* (Algorithm J)

THe use of any model of the general class II (which includes the models B, C and D also) together with equation (4-27) (which has been obtained by employing curvature estimate at x^*) leads directly to the following explicit form for w:

$$(5\text{-}22) \qquad w = \left(\frac{c_1}{\gamma - 1}\right) y;$$

in other words, w does not depend on the precise form of the function $\varphi(\theta,\tau)$. Therefore, when one utilizes a curvature estimate at x^* and a model from the class II, w may be evaluated directly without the solution of any equation. As indicated in section (5-1-3), to have $s^T w > 0$ we should select negative values for c_1. Alternatively, since $\gamma < 1$ (using (3-16)) and using (5-22) we have

$$s^T w = \left(\frac{c_1}{\gamma - 1}\right) s^T y;$$

then $s^T w > 0$ holds, provided that $c_1 < 0$ and $s^T y > 0$ is satisfied.

Employing the values of c_1 given by (4-67b), (4-67c) or the "greedy value" of c_1 (defined in chapter 4) gives three different algorithms. We call these algorithms "Jc1q", "Jc1c" and "Jc1gr", respectively. The algorithm Jc1c which employs a curvature estimate at x^* by cubic interpolation is, in fact, the same as Biggs's algorithm (Biggs(1973)) for the following reasons: using (4-67c), w in (5-22) becomes

$$w = \left(\frac{4\gamma + 2 - 6\lambda}{\gamma - 1}\right) y.$$

Substituting this value of w in (4-6), we have

$$(5\text{-}23) \qquad B^*s = \frac{4\gamma + 2 - 6\lambda}{\gamma - 1} y.$$

As indicated in section 2-3-1, Biggs's algorithm is a special case of the symmetric Huang family for which (2-37) (that is,

$$B^*s = y/\rho)$$

holds in place of $B^*s = y$ (the secant equation). (2-37) may be written in the form

$$(5\text{-}24) \qquad B^*s = y/\eta^*$$

using (2-40). Substituting δ from (2-38) in (2-39) gives

$$\eta^* = \frac{s^T y}{4s^T g^* + 2s^T g - 6(F^* - F)}$$

or

$$(5\text{-}25) \qquad \eta^* = \frac{\gamma - 1}{4\gamma + 2 - 6\lambda},$$

using (3-15). Substituting (5-25) in (5-24) leads to (5-23).

For a quadratic function, substituting the values for c_1 given by (4-67) leads to

$$[c_1/(\gamma - 1)] = 1,$$

in each case, and results in w = y, as desired.

When c_0 and c_1 are obtained by means of (4-66a) with (4-67a), by (4-66b) with (4-67b), or by (4-66c) with (4-67c),

$$c_0 + c_1 = 2(\gamma - 1).$$

On the assumption that c_0 and c_1 are both negative, it then

follows that

$$\frac{c_1}{\gamma - 1} < 2,$$

which is important from the point of view of theorem 4-1, concerning the condition number of H_ϕ^{*w} in relation to that of H_ϕ^{*y}. Employing (4-42) and $\rho < 2$ leads to

$$K(H_\phi^{*w}) < 2\, K(H_\phi^{*y})$$

provided that $\frac{c_1}{\gamma - 1} > 1$.

5-3 The Model B

It is not immediately clear that this model (which belongs to the general class II, with

$$\varphi(\theta,\tau) \equiv \exp(\theta\tau)),$$

becomes linear as θ tends to zero. However, the following discussion shows that the model does, in fact, have this property. Defining c and d by means of

$$a = c/\theta \text{ and } b = d/\theta$$

leads to

$$g(\theta,\tau) \equiv (c/\theta)\, \exp(\theta\tau) + d/\theta.$$

Substitution of $\tau = 0$ and $\tau = 1$ and straightforward manipulation yields

$$d = \theta g - y/\psi_0(\theta),$$

$$c = y/\psi_0(\theta),$$

where $\psi_0(\theta)$ is defined in (5-9). Thus, the model may be written in the following form:

$$g(\theta,\tau) \equiv g + y\left[[\exp(\theta\tau) - 1]/[\theta\psi_0(\theta)]\right]$$

and, in the limit as $\theta \to 0$, this becomes (using (5-10))

$$g(\theta,\tau) \equiv g + y\tau.$$

Therefore, using (4-32) and (5-5) gives, for w:

$$w = y/\psi_0(-\theta). \tag{5-26}$$

Using (5-10) then yields

$$\lim_{\theta\to 0} w = y,$$

as desired.

At this point we consider the issue of the positiveness of $s^T w$. If the condition $s^T y > 0$ is satisfied in moving from x to x^* (as is certainly true if p is a descent direction for F at x and (3-16) holds), then $s^T w$ will also be positive, by virtue of the definition of ψ_0. [It may also be observed that, for small θ,

$$w \approx (1 + \theta/2)y.]$$

We now consider several algorithms which employ the model B together with function values, curvature estimate at x and the manipulation of a curvature estimate at x and a curvature estimate at x^*.

5-3-1 Algorithm Bf (Ford and Ghandhari(1989))

Using (4-25) and (5-5) gives the following equation by which we may obtain θ:

(5-27) $(\lambda - 1)/(\gamma - 1) = [\psi_0(\theta) - 1]/\theta\psi_0(\theta)$ $[\equiv \mu(\theta)$, say],

where λ has been defined in chapter 3 and ψ_0 has been defined in (5-9). From the relation

(5-28) $\psi_0(\theta) \equiv 1 + \theta\,\Upsilon(\theta)$,

(using (5-13)), it then follows that

(5-29) $(\lambda - 1)/(\gamma - 1) = \Upsilon(\theta)/[1 + \theta\Upsilon(\theta)]$ $[\equiv \mu(\theta)]$.

In order to investigate whether solution(s) of equation (5-29) exist, we note first that the derivative of $\mu(\theta)$ is given by

(5-30) $\mu'(\theta) \equiv [\Upsilon'(\theta) - \{\Upsilon(\theta)\}^2]/[1 + \theta\Upsilon(\theta)]^2$.

We will prove in section 5-6 (Lemma 5-2) that the numerator of this expression (and, hence, $\mu'(\theta)$) is negative for all values of θ, so that μ is a strictly decreasing function. It is straightforward, also, to establish, from the properties of $\Upsilon(\theta)$, that

(5-31) $\lim_{\theta\to-\infty} \mu(\theta) = 1$; $\lim_{\theta\to 0} \mu(\theta) = 0.5$; $\lim_{\theta\to+\infty} \mu(\theta) = 0$.

From these results, it is clear that (5-29) will have a finite solution if and only if

(5-32) $0 < (\lambda - 1)/(\gamma - 1) < 1$.

Since $\gamma < 1$ (using (3-16)), the left-hand inequality becomes the

condition $\lambda < 1$. In Lemma 5-3 in section 5-6 we discuss this condition in detail. The main result is that when $F(x(\tau)) = \{F(x + \tau s)\}$ is convex in the interval between $\tau = 0$ and $\tau = 1$, then $\lambda \leq 1$.

Turning to the right-hand inequality in (5-32) (which can be rewritten as $\lambda > \gamma$), it is immediately evident from (3-17) that if $\gamma \leq 0$, λ will be greater than γ. If $\gamma > 0$, the inequality $\gamma < \lambda$ may, on occasion, be violated, but a simple argument demonstrates that this is only possible in the case of a function whose behaviour is distinctly non-quadratic, since, for a quadratic function, the result (5-16) implies that

$$\lambda = \gamma/2 + 1/2 > \gamma,$$

using (3-16).

In circumstances when λ violates one or both of the restrictions in (5-32), we use $\theta = 0$, resulting in a "standard" quasi-Newton update for that iteration. (It should be noted that, in the case of a quadratic function, (5-16) and (5-31) imply that $\theta = 0$ will be the root of equation (5-29) and $w = y$, as desired.)

After obtaining θ through this algorithm , we may evaluate w by

$$w = (1 + \delta\theta)y \tag{5-33}$$

where

$$\delta = (\lambda - \gamma)/(1 - \gamma).$$

By this formula (whose derivation is given below), we save one evaluation of ψ_0. We may easily verify (using (5-27)) that

$$\mu(-\theta) \equiv 1 - \mu(\theta) \quad (= \delta).$$

Then, since

$$\mu(-\theta) \equiv \frac{1 - \psi_0(-\theta)}{\theta\psi_0(-\theta)},$$

we obtain

$$[\psi_0(-\theta)]^{-1} = 1 + \theta\delta.$$

(Note: in the paper by Ford and Ghandhari(1989), this algorithm is referred to by the name "EXP2".)

5-3-2 Algorithm Bc0

Using (4-26) and (5-5), for the model B, leads to the following equation (which is less complex than (5-29)):

(5-34) $$(\gamma - 1)/c_0 = (\exp(\theta) - 1)/\theta \quad [\equiv \psi_0(\theta)].$$

The circumstances under which equation (5-34) has a solution are determined by noting that

(5-35) $$\lim_{\theta\to-\infty} \psi_0(\theta) = 0; \quad \lim_{\theta\to 0} \psi_0(\theta) = 1; \quad \lim_{\theta\to+\infty} \psi_0(\theta) = +\infty;$$

on the other hand,

(5-36) $$[\psi_0(\theta)]' = \psi_1(\theta) = \exp(\theta)\ \Upsilon(-\theta) > 0,$$

because

$$\Upsilon(-\theta) = [\exp(-\theta) + \theta - 1)]/\theta^2 \geq 0,$$

$$\left(\text{using } \exp(-\theta) \geq 1 - \theta \text{ and } \lim_{\theta\to 0} \Upsilon(-\theta) = 0.5\right).$$

Thus $\psi_0(\theta)$ is an increasing function and positive for all finite values of θ, which implies that $(\gamma - 1)/c_0$ should be positive for solutions to exist.

Since $\gamma - 1 < 0$ (using (3-16)), we must therefore select c_0 to be negative (which is equivalent to choosing a positive curvature estimate at x, as noted above).

After computing θ in this algorithm, we may evaluate w by

$$w = \left(\theta + c_0/(\gamma - 1)\right)y. \tag{5-37}$$

This formula, whose derivation follows below, saves one evaluation of ψ_0. Using (5-34), we may write

$$\exp(-\theta) = \frac{c_0}{\theta(\gamma - 1) + c_0},$$

and substituting this value in (5-26) gives (5-37).

For a quadratic function, all the corresponding values for c_0 from (4-66) (apart from the excluded case (4-66d)) give $\theta = 0$ and $w = y$, as desired.

5-3-3 Algorithm Bc/01

We consider this algorithm (which employs both c_0 and c_1) because, to obtain θ, there is no requirement to solve any equation. Using (4-28) and (5-5) gives the following equation for θ:

$$\exp(\theta) = c_1/c_0 \tag{5-38}$$

or

$$\theta = \ln(c_1/c_0),$$

assuming that $c_1/c_0 > 0$.

To have $s^T w > 0$, c_1 must be negative (see above). Thus, to have a solution for θ, c_0 must also be negative.

Using (4-68), we have

$$\frac{c_1}{c_0} = \frac{2(\gamma - 1)}{c_0} - 1,$$

which is equal to unity for a quadratic function using (5-16) and the values for c_0 in (4-66) (except (4-66d)). This again gives $w = y$ for a quadratic function, as anticipated.

5-3-4 Algorithm Bc-01

We now propose another algorithm which does not require the solution of an equation. Using (4-29) and (5-5) leads to

$$(5\text{-}39) \qquad \frac{\theta\exp(\theta) - \theta}{\exp(\theta) - 1} = \theta = \frac{c_1 - c_0}{\gamma - 1}.$$

To evaluate $c_1 - c_0$, we may employ:

(a) (4-66a) with either of (4-67a) and (4-67b) or,

(b) (4-66b) with either of (4-67a) and (4-67b).

In every case, for a quadratic function, $c_1 - c_0 = 0$, which gives $\theta = 0$, as desired.

5-4 The Model C

In this section we will examine several algorithms which employ the model C defined in (5-3) together with function value and curvature estimate at x. Since, our main purpose in using (4-29) or (4-28) [that is, using $c_1 - c_0$ or c_1/c_0, respectively] is to have a simpler equation or a better result and neither of these aims are achieved, we do not consider the algorithms which are obtained via (4-29) and (4-28) for this model any more.

The model C belongs to the general class II with

(5-40) $\varphi(\theta,\tau) \equiv \tau\exp(\theta\tau)$.

Throughout this section we will make use of the following values, (using (5-40)):

(5-41) $\varphi_0 = 0; \quad \varphi_1 = \exp(\theta); \; \varphi_0' = 1; \; \varphi_1' = \exp(\theta)(1 + \theta)$.

Using (5-41) and (4-32) gives the following expression for w:

(5-42) $w = (1 + \theta)y,$

which is simpler than the vector w in the previous models A and B and, consequently, it is easier to establish conditions under which $s^Tw > 0$. It is obvious that s^Tw is positive (provided that $s^Ty > 0$) if and only if

(5-43) $\theta > -1,$

holds.

5-4-1 Algorithm Cf

In this algorithm we utilize the model C together with function values. Using (5-41) and (4-25) gives the following equation:

$$(5\text{-}44) \qquad \exp(-\theta)\int_0^1 \tau\exp(\theta\tau)\,d\tau = (\lambda - 1)/(\gamma - 1).$$

From (5-8), (5-9) and (5-13), (5-44) becomes

$$(5\text{-}45) \qquad (\lambda - 1)/(\gamma - 1) = \exp(-\theta)\ \psi_1(\theta)$$

$$= \left(\exp(-\theta) + \theta - 1\right)/\theta^2 = \Upsilon(-\theta).$$

As before, we are interested in determining the restrictions under which the equation defining θ has a solution. We note that

$$\lim_{\theta\to-\infty} \Upsilon(-\theta) = +\infty;\quad \lim_{\theta\to 0} \Upsilon(-\theta) = 0.5;\quad \lim_{\theta\to+\infty} \Upsilon(-\theta) = 0;$$

on the other hand, since $\Upsilon(\theta)$ has a positive derivative for all values of θ (see Lemma 5-1 in section 5-6), $\Upsilon(-\theta)$ will have a negative derivative for all values of θ because

$$[\Upsilon(-\theta)]' = -\Upsilon'(-\theta).$$

Therefore provided that $(\lambda - 1)/(\gamma - 1) > 0$, there exists exactly one solution of (5-45). This condition is the left-hand inequality in (5-32), which we considered in section 5-3-1.

In practice, since we need $\theta > -1$, we solve equation (5-45) if the condition

$$[\Upsilon(-(-1)) =]\ (e - 2) > (\lambda - 1)/(\gamma - 1),$$

holds. We may modify this restriction if we wish to bound the possible deterioration in the condition number of the matrix H_{ϕ}^{*w} (see Theorem 4-1).

Usually we restrict the factor ρ (in Theorem 4-1) to lie in the interval [0.2,4.0], in order to bound the condition number of the matrix H_{ϕ}^{*w} in relation to that of the matrix H_{ϕ}^{*y}. The value of ρ in this model is $\theta + 1$. This gives

$$0.2 \leq \theta + 1 \leq 4.0 \text{ or}$$

$$-0.8 \leq \theta \leq 3.0.$$

Since $\Upsilon(-(-0.8)) = 0.6649...$, $\Upsilon(-3) = 0.2277...$ and $\Upsilon(-\theta)$ is a decreasing function, we solve equation (5-45) if

$$0.2277... \leq (\lambda - 1)/(\gamma - 1) \leq 0.6649...$$

holds. For a quadratic function [using (5-16) and (5-45)], we obtain $\theta = 0$ and consequently $w = y$, as expected.

5-4-2 Algorithm Cc0

This is another algorithm which leads to a linear equation in terms of θ. We obtain θ in this algorithm using (5-41) and (4-26), which is the general equation arising from the use of a curvature estimate at x and the model II. The value of θ is

$$\theta = \ln\left(\frac{\gamma - 1}{c_0}\right). \tag{5-46}$$

To have θ defined, c_0 must be negative (since $\gamma < 1$ using (3-16)).

For a quadratic function, $(\gamma - 1)/c_0 = 1$ if we use (4-66a), (4-66b) or (4-66c) and, hence, $w = y$, as desired.

5-5 The Model D

As previously indicated, the condition $s^T w > 0$ is very important. In some methods, we have to establish conditions which ensure that $s^T w > 0$. The motivation for introducing this model is that there is no need to establish any condition for having $s^T w > 0$ (see below). The definition of $\varphi(\theta,\tau)$ in this model (which belongs to the general class II) is:

$$\varphi(\theta,\tau) \equiv (\tau - 1)\exp(\theta\tau),$$

which gives

(5-47) $\varphi_0 = -1;\ \varphi_1 = 0;\ \varphi_0' = 1 - \theta;\ \varphi_1' = \exp(\theta).$

Using (4-32) gives the following expression for the vector w:

$$w = \exp(\theta)\ y$$

which maintains $s^T w > 0$ provided that $s^T y > 0$.

We do not consider (in connection with this model) the equation which arises from using (4-28) and (4-29), because the equations are no simpler than those we are going to describe, nor are the numerical results any better.

5-5-1 The algorithm Df

Using (5-47) and (4-25), we have

$$\int_0^1 \tau\exp(\theta\tau)\,d\tau - \int_0^1 \exp(\theta\tau)\,d\tau + 1 = (\lambda - 1)/(\gamma - 1),$$

which leads to the following equation for θ (employing (5-9)):

$$\psi_1(\theta) - \psi_0(\theta) = (\lambda - \gamma)/(\gamma - 1),$$

or

(5-48) $$(\gamma - \lambda)/(\gamma - 1) = (\exp(\theta) - \theta - 1)/\theta^2 = \Upsilon(\theta),$$

(using (5-13)).

Since $\Upsilon'(\theta) > 0$ (see section 5-6) for all values of θ, it follows that $\Upsilon(\theta)$ is monotonically increasing. We also have (as before)

(5-49) $$\lim_{\theta\to-\infty} \Upsilon(\theta) = 0;\quad \lim_{\theta\to 0} \Upsilon(\theta) = 0.5;\quad \lim_{\theta\to+\infty} \Upsilon(\theta) = +\infty.$$

We now may investigate the conditions under which (5-48) has a solution. Since $\Upsilon(\theta) \geq 0$ for all values of θ and it is ascending, we will have a unique solution only if

(5-50) $$(\gamma - \lambda)/(\gamma - 1) > 0$$

which is exactly the right-hand-side of inequality (5-32), which was considered in section 5-3-1. We remind the reader that the result of that consideration was that this condition may be expected to hold, in general, and, consequently, we may expect equation (5-48) to possess a solution on most iterations.

For a quadratic function, using (5-16) and (5-48) we obtain $\theta = 0$ or $w = y$, as desired.

5-5-2 The Algorithm Dc0

Using (5-47) and (4-26) gives

(5-51) $\theta = 1 - c_0/(\gamma - 1)$,

which is a straightforward means of obtaining θ. Since the inequality $s^T w > 0$ holds for all θ, it appears that we do not need to impose any condition upon the value of c_0. But, from the point of view of the condition number of the matrix H_ϕ^{*w} (see Theorem 4-1), we need to restrict the size of $\exp(\theta)$ (since $w = \exp(\theta)y$ in this model). To restrict $K(H_\phi^{*w})$ in comparison with $K(H_\phi^{*y})$, we may impose the following restrictions:

$0.2 \leq [\rho(\theta) \equiv] \exp(\theta) \leq 4.0$ or,

$\ln 0.2 \leq \theta \leq \ln 4.0$.

Using (5-51), we then have

$2.6094... \geq c_0/(\gamma - 1) \geq -0.3863....$

For a quadratic function using (4-66) (except (4-66d)) and (5-51), we obtain $\theta = 0$ or $w = y$, as expected.

5-6 Computational Considerations and Lemmas

A key component in the implementation of some of the proposed algorithms in this chapter and in the next chapter is (of course) the accurate calculation of ψ_0, ψ_1, Υ, ψ_0' [$\equiv \psi_1$], ψ_1' and Υ'. We may write ψ_0 and ψ_1 in terms of Υ by

(5-52) $\psi_0(\theta) \equiv \theta\Upsilon(\theta) + 1,$

and

(5-53) $\psi_1(\theta) \equiv \exp(\theta)\ \Upsilon(-\theta),$

and, consequently, ψ_0' and ψ_1' in terms of Υ and Υ'. Therefore, for brevity, we may concentrate our attention on the evaluation of Υ. This function and its derivative are given explicitly (for $\theta \neq 0$) by

(5-54a) $\Upsilon(\theta) \equiv (\exp(\theta) - \theta - 1)/\theta^2;$

and

(5-54b) $$\Upsilon'(\theta) \equiv \frac{(\theta - 2)\exp(\theta) + 2 + \theta}{\theta^3},$$

from (5-9) and (5-13). However, these explicit forms cannot be used for small values of θ, because of the danger of severe cancellation. In the case of (5-54a), the observation that

$$\exp(\theta) \approx 1 + \theta$$

for small θ is sufficient to point out the danger, while the more accurate approximation

$$\exp(\theta) \approx 1 + \theta + \theta^2/2$$

produces a similar conclusion (from (5-54b)) for the function $\Upsilon'(\theta)$. This presents a problem, since we have argued that, near

its minimum, the function F may be expected to behave like a quadratic function, leading us to anticipate that the roots of equations which involve Υ (or the related functions) will be small. The straightforward manner of dealing with this difficulty is to employ the standard power series for the exponential function, leading to:

(5-55a) $\Upsilon(\theta) \equiv 1/2 + \theta/3! + \theta^2/4! + \ldots + \theta^i/(i+2)! + \ldots,$

and

(5-55b) $\Upsilon'(\theta) \equiv 1/3! + 2\theta/4! + 3\theta^2/5! + \ldots + i\theta^{i-1}/(i+2)! + \ldots,$

both of which may be expected to converge rapidly for all values of θ that are of interest in this context (large values of θ may be deal with directly via (4-54)).

Alternatively, we might consider the evaluation ψ_0, ψ_1, ψ_0' and ψ_1' through numerical integration by, for instance, adaptive quadrature methods (see, for example, Burden and Faires(1985)), because of the definition given in (5-8) and following expressions:

$$\psi_0' \equiv \psi_1,$$

and

$$\psi_1' \equiv \psi_2.$$

Having obtained ψ_0, ψ_1 and their derivatives, we may evaluate Υ and Υ' using (5-52) or (5-53). As a final alternative, approximations (polynomial, rational or otherwise) may be utilized to evaluate ψ_0, ψ_1, Υ and their derivatives.

We wish to compare the complexity (in terms of μ [number of multiplications and divisions] and α [number of additions and

subtractions]) of the different methods for evaluating ψ_0, ψ_1, Υ and their derivatives. The methods we will consider are polynomial approximation, rational approximation and adaptive quadrature. For brevity, we may concentrate our attention on the evaluation of ψ_0. To evaluate ψ_0 using polynomial approximation may be considered as the basic method, because we can control the accuracy easily. We may write

$$\psi_0(\theta) \equiv \sum_{i=0}^{\infty} \frac{\theta^i}{(i+1)!}.$$

Suppose we stop after N + 1 terms and use $\psi_0^{(N)}(\theta)$ (the expression so far obtained) as our estimate of $\psi_0(\theta)$. Then

$$|\psi_0(\theta) - \psi_0^{(N)}(\theta)| \leq \sum_{i=N+1}^{\infty} \frac{|\theta|^i}{(i+1)!}$$

$$= \frac{|\theta|^{N+1}}{(N+2)!} + \frac{|\theta|^{N+2}}{(N+3)!} + \ldots$$

$$= \frac{|\theta|^{N+1}}{(N+2)!} \left\{1 + \frac{|\theta|}{N+3} + \frac{|\theta|^2}{(N+3)(N+4)} + \ldots\right\}$$

$$\leq \frac{|\theta|^{N+1}}{(N+2)!} \exp(|\theta|) \quad [= \beta(\theta)],$$

where $\beta(\theta)$ is ,therefore, an upper bound for the error in the evaluation of ψ_0 by the polynomial approximation. Thus, by choosing a larger N, we may obtain a more accurate value for ψ_0 (since we will only use this method for evaluating $\psi_0(\theta)$ when $|\theta| \leq 0.5$).

Since for $|\theta| > 0.5$, we may employ $\psi_0(\theta)$ (see above) to evaluate ψ_0 and utilize the polynomial approximation only for $|\theta| \leq 0.5$. In the following table, we give the values for μ and α only for such values of θ.

θ	μ	α
± 0.5	42	28
± 0.3	36	24
± 0.1	30	20
0.01	21	14
0.0001	12	8
10^{-8}	6	4
$\leq 10^{-16}$	3	2

In this table, we have terminated the computation when

$$\frac{|\theta|^N/(N+1)!}{\psi_0^{(N)}(\theta)} \quad [\equiv \alpha(\theta)] < 10^{-16}$$

holds. In that case,

$$\alpha(\theta) = \left\{\beta(\theta)\ \exp(-|\theta|)\ \frac{N+2}{|\theta|}\right\}/\psi_0^{(N)}(\theta) < 10^{-16}$$

or

$$\beta(\theta) < \left\{10^{-16}\ \psi_0^{(N)}(\theta).\exp(|\theta|).|\theta|/(N+2)\right\} < 10^{-16}\psi_0^{(N)}(\theta),$$

because $|\theta| < 0.5$ and $N \geq 1$ and, consequently,

$$\exp(|\theta|)\ |\theta|/(N+2) < 1.$$

Thus, the relative error in the evaluation of $\psi_0(\theta)$ by this means

is less than 1 part in 10^{16}.

We now consider the evaluation of ψ_0 by means of adaptive quadrature, based upon the definition

$$\psi_0(\theta) \equiv \int_0^1 \exp(\theta\tau)\ d\tau.$$

It seems to be unlikely that adaptive quadrature can be competitive with polynomial approximation, because if we evaluate the integrand more than three times, (which is almost certainly the case), we will need at least 48μ and 48α. (For each evaluation of the integrand ($\exp(\theta\tau)$), typically 12μ and 12α (at least) are needed (for example, Tops-10 Software Notebook (September 1983)).

A rational approximation is often more efficient than polynomial approximation (Burden and Faires(1985)). Burden and Faires state that rational functions whose numerator and denominator have the same degree (or nearly the same degree) generally produce approximation results superior to polynomial methods for the same amount of computational effort. Further evidence for the efficiency of rational approximation arises from computer library routines for evaluating standard functions, which often employ rational approximations (for example, see Tops-10 Software Notebook (September 1983)).

Therefore, we may conclude that, for the functions of interest here, polynomial approximation may be expected to be more efficient than adaptive quadrature and since, in general,

rational approximation is more efficient than polynomial approximation, therefore these three methods may be placed in the following tentative order: (i) rational approximation, (ii) polynomial approximation and (iii) adaptive quadrature.

Next, we turn to consider the properties of the function Υ.

Lemma 5-1

The function $\Upsilon(\theta)$ defined by equation (5-54a) (or by equation (5-13)) and by $\Upsilon(0) = 0.5$ has a positive derivative for all values of θ.

Proof: For $\theta \neq 0$, the derivative is given by

(5-56) $\Upsilon'(\theta) \equiv h(\theta)/\theta^3$,

where

$$h(\theta) \equiv (\theta - 2)\exp(\theta) + 2 + \theta,$$

using (5-54b). Furthermore, since (for $\theta \neq 0$)

$$[\Upsilon(\theta) - \Upsilon(-\theta)]/2\theta \equiv [\exp(\theta) - \exp(-\theta) - 2\theta]/2\theta^3$$

$$\equiv 1/6 + o(\theta),$$

it follows that

(5-57) $\Upsilon'(0) = 1/6.$

Suppose, contrary to the statement of the Lemma, that Υ' is negative or zero for some values (which must necessarily be non-zero) of θ. Then, h must possess a zero (call it θ_0) which is not equal to zero. Therefore, by Rolle's Theorem (Hardy(1963)),

$$h'(\theta) \equiv \theta\exp(\theta) + 1 - \exp(\theta)$$

must have a zero (call it θ_1) between θ_0 and 0 (in addition to its zero at $\theta = 0$). A further application of Rolle's Theorem shows that

$$h''(\theta) \equiv \theta\exp(\theta)$$

must have a zero at some point between θ_1 and 0. This is clearly untrue, so that the original assumption (that Υ' is non-positive somewhere) must be false, and the Lemma is proved.■

Lemma 5-2

The function $\Lambda(\theta) \equiv \Upsilon'(\theta) - \{\Upsilon(\theta)\}^2$ is negative for all values of θ.

Proof: For $\theta = 0$, equations (5-55a) and (5-55b) give

(5-58) $\Lambda(0) = 1/6 - 1/4 < 0.$

For $\theta \neq 0$, (5-54a) and (5-54b) yield

(5-59) $\Lambda(\theta) \equiv \{\theta^2\exp(\theta) - (\exp(\theta) - 1)^2\}/\theta^4 \equiv k(\theta)/\theta^4$, say.

If Λ is (in contradiction of the Lemma) to be non-negative for some value of θ, it follows from (5-58) and (5-59) that $k(\theta)$ must be zero for some non-zero value (call it θ_0) of θ, in addition to being zero at $\theta = 0$. Thus, by Rolle's Theorem,

$$k'(\theta) \equiv \exp(\theta)(\theta^2 + 2\theta + 2 - 2\exp(\theta))$$

must be zero for some value (call it θ_1) of θ between θ_0 and 0. Then the function $m(\theta) \equiv \theta^2 + 2\theta + 2 - 2\exp(\theta)$ will also be zero at θ_1 and 0. Hence,

$$m'(\theta) \equiv 2\theta + 2 - 2\exp(\theta) = -2(\exp(\theta) - 1 - \theta)$$

must be zero at some (non-zero) value θ_2 lying between θ_1 and 0. However, this is a contradiction, since $\exp(\theta) > 1 + \theta$ for all $\theta \neq 0$. Thus, k cannot be zero for any non-zero value of θ and, hence, $\Lambda(\theta)$ must be negative for all values of θ.■

Lemma 5-3

Let $F(\tau)$ be a twice continuously differentiable function and $\lambda = [F(1) - F(0)]/\sigma_0$ where $\sigma_0 = F'(0)$ is assumed to be non-zero. If $\lambda > 1$ and $\sigma_0 < 0$ then $F(x)$ is non-convex.

Proof: We prove that, if $F(x)$ is convex in the interval $[0,1]$, then $\lambda \leq 1$. We may write

$$\int_0^\tau F''(\mu)\, d\mu = F'(\tau) - F'(0),$$

or

$$F'(\tau) = \int_0^\tau F''(\mu)\, d\mu + F'(0). \tag{5-60}$$

Since $F(x)$ is assumed to be convex,

$$F''(\mu) \geq 0 \text{ for all } \mu \in [0,1]. \tag{5-61}$$

Therefore

$$F'(\tau) \geq F'(0) = \sigma_0. \tag{5-62}$$

Integrating (5-62) gives

$$\int_0^1 F'(\tau)d\tau = F(1) - F(0) \geq \int_0^1 \sigma_0 d\tau = \sigma_0,$$

which leads to

(5-63) $\lambda \leq 1$

on dividing by $\sigma_0 (< 0)$. ■

Returning to the case of multi-dimensional optimization, it follows from this Lemma that the expression $(\lambda - 1)/(\gamma - 1)$ is non-negative for a convex objective function, because $\gamma < 1$ (using (3-16)).

CHAPTER 6

ALTERNATIVE MODELS

In chapter 5 we discussed four one-parameter gradient models containing the factor $\exp(\theta\tau)$ and which, consequently, have no points of discontinuity through the introduction of the nonlinear term(s). We recall from chapter 4 that, in order to obtain equations using function values (respectively, curvature estimates), we need the integral (derivative) of the gradient model. Thus, using $\exp(\theta\tau)$ in a model often leads to a nonlinear equation (involving $\exp(\theta)$): in particular, when function values (equations (4-22) and (4-25)) are used. All of the models we have investigated which involve $\exp(\theta\tau)$ and which employ function values to define θ, yield nonlinear equations which must be solved numerically. This requires evaluation of functions such as ψ_0, ψ_1 and their derivatives, as we have seen.

It is our intention in this chapter to examine a few non-exponential models for the gradient. Apart from one model (and then only when utilizing function values), the equations by which we obtain θ turn out to be linear and, as a result, θ is obtained directly. In these models, we impose the following restrictions on the parameter(s):

(i) the model should have no points of discontinuity in the interval $0 \leq \tau \leq 1$;

(ii) the condition $s^T w > 0$ is to be satisfied (see chapter 4 for

the motivation for this condition).

The restrictions created by (ii) usually include these implied by restriction (i). Therefore, we usually have no extra restriction on the models in this chapter besides those imposed by the condition

$s^T w > 0$.

The models in this chapter belong to the general class II. Two of them have one parameter each and are:

(6-1) $g(\theta,\tau) \equiv a/(1 + \theta\tau) + b$ (model "E")

and

(6-2) $g(\theta,\tau) \equiv a\tau^{\theta+1} + b$ (model "F").

Two other models have two parameters. They are:

(6-3) $g(\theta,\tau) \equiv a(\tau + \theta_1\tau^{\theta_2}) + b$ (model "G")

and

(6-4) $g(\theta,\tau) \equiv a\tau(1 + \theta_1\tau^m + \theta_2\tau^n) + b$ (model "H"),

where m and n are some positive constants.

We will examine models E, F, G and H in section 1, 2, 3 and 4, respectively.

6-1 The Model E

In this section, we utilize the expression (6-1) for the gradient along the search direction. By comparison with the general class II, the form of φ is

(6-5) $\varphi(\theta,\tau) \equiv 1/(1 + \theta\tau)$,

which gives

(6-6) $\varphi_0 = 1;\ \varphi_1 = 1/(1 + \theta);\ \varphi_0' = -\theta;\ \varphi_1' = -\theta/(1+ \theta)^2$.

First, we obtain the vector w, which is common to all of the algorithms related to this model. Using (6-6) and (4-32), the vector w is obtained (after simplification) as

(6-7) $w = y/(1 + \theta)$.

Assuming $s^T y > 0$, we should have (to maintain $s^T w > 0$)

(6-8) $1 + \theta > 0$, or $\theta > -1$.

On the other hand, the model E is defined, as long as τ does not take the value

(6-9) $\tau = -1/\theta$.

We therefore need to ensure that $-1/\theta$ does not lie in the interval [0,1], which is equivalent to imposing the condition $\theta > -1$. Therefore, as long as $(1 + \theta) > 0$, the model is defined and $s^T w > 0$ holds. In the light of Theorem 4-1, we strengthen the condition $(1 + \theta) > 0$ in practice so that (for example)

$0.2 \le \rho(\theta) \equiv 1/(\theta + 1) \le 4.0$, or (equivalently)

(6-10) $-0.75 \leq \theta \leq 4.0.$

6-1-1 Algorithm Ef

This algorithm (as indicated by its name) is based on the model E and employs function values. Using (4-25) and (6-6) gives

$$\frac{\int_0^1 \frac{d\tau}{1 + \theta\tau} - 1}{\{-\theta/(1+\theta)\}} = \frac{\lambda - 1}{\gamma - 1},$$

or

$$(6\text{-}11) \qquad \frac{(1 + \theta)[\theta - \ln(1 + \theta)]}{\theta^2} = \frac{\lambda - 1}{\gamma - 1}.$$

Clearly, we need to impose the requirement $(1 + \theta) > 0$, which is also necessary and sufficient to ensure that $s^T w > 0$, as we have seen above. We may note that

$$\lim_{\theta \to 0} \frac{\theta - \ln(1 + \theta)}{\theta^2} = 1/2.$$

Utilizing the transformation

$$(6\text{-}12) \qquad r = \ln(1 + \theta),$$

equation (6-11) leads (after simplification) to

$$(6\text{-}13a) \qquad \frac{\exp(r) - r - 1}{\exp(r) + \exp(-r) - 2} = \frac{\gamma - 1}{\lambda - 1},$$

or

$$(6\text{-}13b) \qquad \frac{\gamma - \lambda}{\lambda - 1} = \frac{\Upsilon(-r)}{\Upsilon(r)} \quad [\equiv \nu(r), \text{ say}],$$

where Υ has been defined in (5-13).

Solving (6-13), we obtain r and thus θ [via (6-12)]. In order to determine the conditions under which the equation (6-13) has a solution, we first consider $\nu'(r)$:

$$\nu'(r) = \frac{r[\exp(-r) - \exp(r)] + 2\exp(-r) + 2\exp(r) - 4}{[\exp(r) - r - 1]^2}$$

$$\left[= \frac{\exp(-r)[1 - \exp(r)]}{[\exp(r) - r - 1]^2} [r + 2 + \exp(r)\ (r - 2)] \right],$$

or, after some manipulation,

$$\nu'(r) = \frac{\exp(-r) - 1}{r} \left(\frac{r^2}{\exp(r) - r - 1} \right)^2 \left(\frac{r + 2 + \exp(r)(r - 2)}{r^3} \right)$$

$$= - \psi_0(-r)\ \Upsilon'(r) / [\Upsilon(r)]^2,$$

where ψ_0 and Υ' are defined in (5-9) and (5-56), respectively. Thus, $\nu'(r) < 0$ (when $r \neq 0$) because $\psi_0(-r) > 0$ and $\Upsilon'(r) > 0$ (see Lemma 5-1, section 5-6).

For r = 0,

$$\lim_{r \to 0} \nu'(r) = - 2/3,$$

because

$$\lim_{r \to 0} \psi_0(r) = 1; \quad \lim_{r \to 0} \Upsilon(r) = 1/2; \quad \lim_{r \to 0} \Upsilon'(r) = 1/6;$$

(using (5-52) and (5-55)).

Having established that $\nu'(r) < 0$ for all values of r, we conclude that $\nu(r)$ is a decreasing function of r. On the other hand, the observations:

$$\lim_{r \to -\infty} \nu(r) = +\infty; \quad \lim_{r \to 0} \nu(r) = 1; \quad \lim_{r \to +\infty} \nu(r) = 0;$$

lead us to infer that the equation (6-13b) has a (unique)

solution provided that

$$(6\text{-}14) \qquad \frac{\gamma - \lambda}{\lambda - 1} > 0.$$

In practice, because we wish to restrict θ to lie in the interval [-0.75, 4.0] [by virtue of (6-10)], we therefore require, from (6-13b), the following inequalities to hold before we commence solving for r:-

$$2.5361... \geq \frac{\gamma - \lambda}{\lambda - 1} \geq 0.33859....$$

The argument above guarantees that, if these conditions are satisfied, then (6-13b) will have a unique solution.

For a quadratic function, using (5-16) and (6-11) we obtain $\theta = 0$ and $w = y$, as expected.

6-1-2 Algorithm Ec0

In this algorithm, we employ the model E and a curvature estimate at x to determine θ. Therefore, we need to substitute the values of φ_0, φ_1 and φ_0' from (6-6) into (4-26). This gives the value of θ directly, from the following expression:

$$(6\text{-}15) \qquad \theta = -1 + \frac{c_0}{\gamma - 1}.$$

To maintain $s^T w > 0$, we must have

$$1 + \theta > 0,$$

using (6-8). On the other hand, employing (6-15)

$$1 + \theta = c_0 / (\gamma - 1).$$

Therefore, to have $s^T w > 0$,

$$c_0/(\gamma - 1) > 0$$

must hold. Since $\gamma < 1$ (using (3-16)), we must hence only employ negative values of c_0 or, equivalently, the curvature estimate at x must be positive.

To control the condition number of H_ϕ^{*w} in relation to H_ϕ^{*y}, we enforce the bounds (6-10), which are equivalent to (in this case)

$$(\gamma - 1)/4 > c_0 > 5.0(\gamma - 1).$$

For a quadratic function, substituting the values for c_0 given by (4-66) (apart from the excluded case (4-66d)) into (6-15) leads to $\theta = 0$, in each case, which implies $w = y$, as desired.

6-1-3 Algorithm Ec/01

Using (4-28) and (6-6) leads to

$$\frac{1}{(1 + \theta)^2} = \frac{c_1}{c_0},$$

or

$$\text{(6-16)} \qquad \frac{1}{1 + \theta} = \left(\frac{c_1}{c_0}\right)^{1/2}.$$

Since c_1 must be negative (see chapter 4), to have $1/(1 + \theta)$ defined, c_0 must be negative as well. Using (6-7) and (6-16) then gives

$$w = (c_1/c_0)^{1/2} y.$$

This formula yields a vector w which depends on the curvature estimate at x and at x^*.

Using (4-68), we have

$$\frac{c_1}{c_0} = \frac{2(\gamma - 1)}{c_0} - 1,$$

which is equal to unity (if the function is quadratic) using (5-16) and the values for c_0 in (4-66) (except (4-66d)). This again gives w = y for a quadratic function, as anticipated.

6-1-4 Algorithm EcLS

In this algorithm, we obtain the parameter θ by solving equations (4-26) and (4-27) in an *unweighted* least-squares sense. In this way, θ may be obtained by the following expression:

$$(6\text{-}17) \qquad \theta = \left[\frac{\gamma - 1}{c_1} + \frac{c_0}{\gamma - 1} - 2\right]/2$$

or

$$\theta + 1 = \left[\frac{\gamma - 1}{c_1} + \frac{c_0}{\gamma - 1}\right]/2.$$

This gives (using (6-7))

$$w = \rho y,$$

where

$$\rho = \frac{2c_1(\gamma - 1)}{(\gamma - 1)^2 + c_1 c_0}.$$

Since $\gamma < 1$ (using (3-16)) and $c_1 < 0$, if we select a negative value for c_0, then $\rho > 0$ clearly holds. In that case, $s^T w > 0$,

provided that $s^T y > 0$, and $s^T y > 0$ holds because $\gamma < 1$ and the search direction p is downhill. For a quadratic function, all the corresponding estimates (4-66) and (4-67) (apart from the excluded case (4-66d)) give $\rho = 1$ and $w = y$, as desired.

6-2 The Model F

In this section, we employ the model defined in (6-2), which may be considered to be one of the simplest among all of the suggested models. The parameter θ is determined (a) by function values and (b) by manipulation of function values and curvature estimate at x^*. Since, according to this model, the curvature at x is zero if $\theta > 0$ (and this may be considered as a possible disadvantage of the model), we do not employ equations based upon using curvature estimate(s) at x. Comparing this model and the general model II gives

(6-18) $\varphi(\theta,\tau) \equiv \tau^{\theta+1}$.

φ and its derivative (which may be used to obtain w) are defined provided that

(6-19) $\theta > 0$.

In that case,

(6-20) $\varphi_0 = 0; \quad \varphi_1 = 1; \quad \varphi_1' = \theta + 1.$

Applying (4-32) gives

(6-21) $w = (1+\theta)y$.

To maintain $s^T w > 0$ (provided that $s^T y > 0$), we must have

(6-22) $\theta + 1 > 0.$

6-2-1 Algorithm Ff

Using (4-25) and (6-20), we may write

$$\frac{1}{\theta + 2} = \frac{\lambda - 1}{\gamma - 1}$$

or

(6-23) $$\theta = \frac{\gamma - 1}{\lambda - 1} - 2.$$

This gives w directly in terms of y, as follows:

$$w = \frac{\gamma - \lambda}{\lambda - 1} y,$$

or

(6-24) $$w = (1 + \frac{\eta}{\lambda - 1}) y,$$

where

$$\eta = \gamma + 1 - 2\lambda.$$

(For a quadratic function, using (5-16), (6-24) becomes w = y, as required.)

To maintain $s^T w > 0$ (provided that $s^T y > 0$), we must have

$$\rho = \frac{\gamma - \lambda}{\lambda - 1} > 0.$$

In practice, because we wish to restrict ρ to lie in the interval [0.2, 4.0] (in the light of Theorem 4-1), we therefore require

the following inequalities to hold:-

$$5\lambda - 4 \le \gamma \le \frac{6\lambda - 1}{5} \quad (\text{if } \lambda < 1),$$

and

$$\frac{6\lambda - 1}{5} \le \gamma \le 5\lambda - 4 \quad (\text{if } \lambda > 1).$$

(If $\lambda = 1$, then θ and w are undefined.)

6-2-2 Algorithm FLSfc1

Using (4-27) and (6-20) gives

(6-25) $\quad 1 + \theta = c_1/(\gamma - 1).$

We may obtain θ by an *unweighted* least-squares solution of (6-23) and (6-25). This gives

(6-26) $\quad \theta = \left(\frac{\gamma - 1}{\lambda - 1} + \frac{c_1}{\gamma - 1} - 3\right)/2,$

or

$$w = 0.5\left[\frac{\gamma - \lambda}{\lambda - 1} + \frac{c_1}{\gamma - 1}\right]y.$$

For a quadratic function, using (5-16) and any value of c_1 from (4-67) gives $w = y$.

6-3 The Model G (Algorithm Gfc0)

In this section, we consider the two-parameter model defined by (6-3). For this model, we have one algorithm which employs equations (4-25) and (4-26) to obtain the parameters. In this algorithm we do not need to solve any equation and w is obtained in terms of y in the general form

$$w = \rho y,$$

where ρ is a function of λ, γ and c_0. Therefore, instead of imposing any restriction on the parameters θ_1 and θ_2, we choose to restrict ρ directly. Comparing (6-3) and the general model II gives

$$\varphi(\theta,\tau) \equiv \tau + \theta_1 \tau^{\theta_2}, \tag{6-27}$$

and, consequently,

$$\varphi_0 = 0; \quad \varphi_1 = 1 + \theta_1; \quad \varphi_0' = 1; \quad \varphi_1' = 1 + \theta_1\theta_2. \tag{6-28}$$

As usual, we first obtain w. Substituting φ_1', φ_0 and φ_1 in (4-32) gives

$$w = \frac{1 + \theta_1\theta_2}{1 + \theta_1}\, y \quad [\equiv \rho y]. \tag{6-29}$$

To obtain θ_1 and θ_2 we substitute from (6-28) into (4-26) and (4-25). This gives

$$1 + \theta_1 = \frac{\gamma - 1}{c_0}, \tag{6-30}$$

and

$$1 + \theta_1 = \frac{\theta_2 - 1}{2(\delta\,\theta_2 + \delta - 1)}, \tag{6-31}$$

respectively, where

(6-32) $\delta = (\lambda - 1)/(\gamma - 1)$.

Thus, (6-30) and (6-31) give

$\theta_2 = [c_0 + 2(\lambda - \gamma)]/[c_0 + 2(1 - \lambda)]$, or

(6-33) $\theta_2 = 1 + 2\eta/[2\lambda - 2 - c_0]$,

(where η is defined in (6-24)), while, directly from (6-30), we have

(6-34) $\theta_1 = -1 + (\gamma - 1)/c_0$.

To evaluate θ_2, we substitute a value for c_0 from (4-66). Substituting c0q2 from (4-66b) into (6-33) gives an undefined value for θ_2. Thus, we effectively have <u>two</u> choices for c_0 in this case: c0c and c0t.

Substituting c0c from (4-66c) into (6-33) and (6-34) gives

$\theta_2 = 2$,

and

$\theta_1 = 3\eta/(6\lambda - 2\gamma - 4)$,

respectively. The value 2 for θ_2 indicates that the model chosen for the gradient is quadratic (see (6-27) and (4-20)), so that the underlying function model will be cubic. It is not unexpected, therefore, to find that the resulting method is equivalent to Biggs's method (Biggs(1973)) based on a cubic approximation to the function:

(6-35) $w = \left(1 + 3\eta/(\gamma - 1)\right)y \quad [= \rho_1 y]$.

As previously indicated, $\eta = 0$ for a quadratic function and we

obtain w = y for this case, as expected.

Substituting c0t from (4-66d) into (6-33) and (6-34) gives

$$\theta_2 = 1 + 2\eta/(2\lambda - 2 + t^{\#}),$$

and

$$\theta_1 = -1 + (1 - \gamma)/t^{\#},$$

respectively. Substituting these values in (6-29) leads to

$$(6\text{-}36) \qquad w = \left(1 + \frac{2\eta(\gamma - 1 + t^{\#})}{(2\lambda - 2 + t^{\#})(\gamma - 1)}\right) y \quad [\, = \rho_2 y].$$

We note that, with this choice for c_0, we obtain w = y when the function is quadratic. This occurrence is unusual for the case of $c_0 = -t^{\#}$. This happens because of the use of function values in this algorithm (besides using the curvature estimate at x).

In (6-35) and (6-36), before evaluating w, we evaluate ρ_1 or ρ_2 as appropriate. If ρ_1 (ρ_2) is in the interval [0.2, 4.0] (remembering Theorem 4-1), we evaluate w; otherwise, we either set w = y or set ρ_1 (ρ_2) to the appropriate bound.

6-4 The Model H (Algorithm Hfc0)

This section is devoted to the two-parameter model (6-4) with these restrictions:

(6-37a) $m \neq n$,

(6-37b) $m > 0$, $n > 0$.

The model becomes a one-parameter model if $m = n$. In that case, the new parameter is effectively $(\theta_1 + \theta_2)$. If either of m or n are less than zero, the derivative of the model will be undefined at $\tau = 0$.

Comparing the model (6-4) and the general class II gives

(6-38) $\varphi(\theta,\tau) \equiv \tau(1 + \theta_1\tau^m + \theta_2\tau^n)$

and, consequently,

(6-39) $\varphi_0 = 0; \quad \varphi_1 = 1 + \theta_1 + \theta_2; \quad \varphi'_0 = 1;$

$\varphi'_1 = 1 + (m + 1)\theta_1 + (n + 1)\theta_2.$

Substituting φ_0, φ_1 and φ'_1 in (4-32) gives

(6-40) $$w = \left(1 + \frac{m\theta_1 + n\theta_2}{1 + \theta_1 + \theta_2}\right)y.$$

To obtain θ_1 and θ_2, we employ the equations (4-26) and (4-25). Using (6-39) and (4-26) leads to

(6-41) $1 + \theta_1 + \theta_2 = (\gamma - 1)/c_0,$

and using (6-39) and (4-25) gives

(6-42) $1/2 + \theta_1/(m+2) + \theta_2/(n+2) = \delta(1 + \theta_1 + \theta_2)$,

(where δ is as in (6-32)),

or

(6-43) $1/2 - \delta = \theta_1[\delta - 1/(m+2)] + \theta_2[\delta - 1/(n+2)]$.

(6-41) and (6-42) constitute the following linear system of equations:

$$\begin{cases} \theta_1 + \theta_2 = \dfrac{\gamma - 1}{c_0} - 1 \ [= \zeta] \\ \left[\dfrac{1}{m+2} - \delta\right]\theta_1 + \left[\dfrac{1}{n+2} - \delta\right]\theta_2 = \delta - 0.5. \end{cases}$$

Solving this system,

(6-44a) $$\theta_1 = \frac{m+2}{m-n}\left\{\zeta + (n+2)(0.5 - \delta - \delta\zeta)\right\}$$

and

(6-44b) $$\theta_2 = -\frac{n+2}{m-n}\left\{\zeta + (m+2)(0.5 - \delta - \delta\zeta)\right\}.$$

Having obtained θ_1 and θ_2, we substitute in (6-40). w becomes

(6-45) $w = \rho y$,

where

$$\rho = m + n + 3 - \delta(m+2)(n+2) + mnc_0/2(\gamma - 1),$$

or, using the definition of η in (6-24) and the definition of δ in (6-32) (which gives

$$\delta = \left[1 - \frac{\eta}{\gamma - 1}\right]/2\right),$$

ρ becomes

$$\rho = 1 + \left[mn(c_0 - \gamma + 1) + (m + 2)(n + 2)\eta\right] / 2(\gamma - 1).$$

For a quadratic function, we have $\eta = 0$ and ρ becomes

$$\rho = 1 + \left[mn(c_0 - \gamma + 1)\right] / \left[2(\gamma - 1)\right].$$

Employing either of the values for c_0 from (4-66a) and (4-66b) then leads to $\rho = 1$, as expected. (In the case of

$$c_0 = \gamma - 1,$$

(from (4-66a)) the result is straightforward. In the case of

$$c_0 = 2\lambda - 2$$

[from (4-66b)), we obtain

$$\rho = 1 - mn\eta/2(\gamma - 1),$$

which is equal to unity for a quadratic function, on using (5-16).]

We have tried different values for m and n and most of the numerical results are satisfactory. For brevity, we will present the results corresponding to the choices:

(a) $m = 1;\ n = 2$, which gives

(6-46) $\rho = (6\eta + c_0)/(\gamma - 1)$

and

(b) $m = 1/2;\ n = 3/2$, which leads to

(6-47) $\rho = 0.625 + (0.375c_0 + 4.375\eta)/(\gamma - 1).$

Comment: To distinguish the algorithms corresponding to the cases (a) and (b), we have appended ".a" and ".b" to the conventional

notation defined in section 4-6. (Of course, it is trivial that we may interchange the values of m and n without any effective change in the formula (6-45).)

In this model (as we have stated previously), we do not attempt to control θ_1 and θ_2 directly. Instead we accept ρ, provided that it lies in the chosen "standard" range:

$$0.2 \leq \rho \leq 4.0,$$

in order to control the possible deterioration in the condition number of the matrix H_{ϕ}^{*w} in relation to that of H_{ϕ}^{*y} (see Theorem 4-1).

Substituting different values for c_0, we may obtain simpler expressions for ρ in particular cases. For example, for $m = 1$ with $n = 2$, when we substitute for c_0 from (4-66b), we obtain

$$\rho = 1 + (2 + m + n)\eta/(\gamma - 1), \tag{6-48}$$

while substituting for c_0 from (4-66c) leads to

$$\rho = 1 + (2 + m + n - mn)\eta/(\gamma - 1). \tag{6-49}$$

In both cases, it is clear that $\rho = 1$ and $w = y$ when the function is quadratic (since, in that case, $\eta = 0$ from (5-16)).

CHAPTER 7

NUMERICAL RESULTS AND CONCLUSIONS

In chapter 5 and 6, we have introduced a number of algorithms designed to improve the numerical performance of quasi-Newton methods. In order to evaluate the proposed algorithms, a number of problems (defined in section 1) have been used as a test set. The BFGS method has been selected as a standard for the purposes of comparison. In section 2, we will describe an algorithmic framework within which we have implemented the BFGS method and our new algorithms. Tables and diagrams which contain numerical results are included in section 3 where we discuss the numerical results and also consider numerical evidence concerning the convergence of the algorithms. We will summarize our results in section 4. Finally, we end the chapter with a few suggestions for further research.

7-1 The Test Problems

In our tests, we have tried to solve the following test problems. For each problem, we give the definition of the function, the starting points utilized (denoted by (a), (b), (c), etc.), the minimum (denoted by z) and F(z), the function value at the stated minimum.

1,2,3. Extended Rosenbrock's function (Spedicato(1978)), n = 2,10,20:

$$F(x) \equiv \sum_{i=1}^{n/2} \left\{ 100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2 \right\}.$$

Starting points:-

(a) : (-1.2, 1.0, -1.2, 1.0, -1.2, ...);

(b) : (-120.0, 100.0, -120.0, 100.0, -120.0, ...);

(c) : (-3.635, 5.621, -3.635, 5.621, -3.635, ...);

(d) : (6.390, -0.221, 6.390, -0.221, 6.390, ...).

Minimum : z = (1.0, 1.0, 1.0, 1.0,..., 1.0).

F(z) = 0.

4. Rosenbrock cubic function (Witte and Holst(1964)), n = 2:

$$F(x) \equiv 100(x_2 - x_1^3)^2 + (1 - x_1)^2.$$

Starting points:-

(a) : (-1.2, 1.0);

(b) : (-12.0, 10.0);

(c) : (-3.635, 5.621);

(d) : (6.390, -0.221).

Minimum : z = (1.0, 1.0).

F(z) = 0.

5. A modification of Rosenbrock's cliff function (Rosenbrock and Storey(1966)), n = 2:

$$F(x) \equiv 10^{-4}(x_1 - 3)^2 - (x_1 - x_2) + \exp[20(x_1 - x_2)].$$

Starting points:-

(a) : (-1.2, 1.0);

(b) : (0, 1.0);

(c) : (9.0, 12.0);

(d) : (30.0, 30.0).

Minimum (to 6 decimal places) : z = (3.000003, 3.149790).

F(z) = 1.997866D-001.

6. Brown's badly-scaled function (Sorensen(1977)), n = 2:

$$F(x) \equiv (x_1 - 10^6)^2 + (x_2 - 2\times10^{-6})^2 + (x_1x_2 - 2)^2.$$

Starting points:-

(a) : (1.0, 1.0);

(b) : (100.0, 100.0);

(c) : (10.0, -0.5);

(d) : (-10.0, 0.0).

Minimum (to 6 decimal places) : z = (1000000.0, 0.000002).

F(z) = 0.

7. Powell's badly-scaled function (Sorensen(1977)), n = 2:

$$F(x) \equiv \{10^4x_1x_2 - 1\}^2 + \{\exp(-x_1) + \exp(-x_2) - 1.0001\}^2.$$

Starting points:-

(a) : (0.0, 1.0);

(b) : (0.01, 5.0);

(c) : (3.0, -0.5);

(d) : (-1.0, 5.0).

Minimum : z = (1.098..., 9.106...).

$F(z) = 0.$

8,9,10. Extended Powell's quartic function (Spedicato(1978)), n = 4,12,20:

$$F(x) \equiv \sum_{i=1}^{n/4} \left\{ (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4 \right\}.$$

Starting points:-

(a) : (3.0, -1.0, 0.0, 1.0, 3.0, ...);

(b) : (2.0, 2.0, 3.0, -1.0, 2.0, ...);

(c) : (-3.0, 1.0, -3.0, 1.0, -3.0, ...);

(d) : (10.0, 10.0, 10.0, 10.0, 10.0, ...).

Minimum : z = (0,0,0,...,0).

$F(z) = 0.$

11. Box's "difficult" exponential function (Box(1966)), n = 3:

$$F(x) \equiv \sum_{i=1}^{10} \left\{ x_3 [\exp(t_i x_1) - \exp(t_i x_2)] - \exp(-t_i) + \exp(-10t_i) \right\}^2,$$

where $t_i = 0.1 \times i$.

Starting points:-

(a) : (0.0, -20.0, 1.0);

(b) : (0.0, -10.0, 10.0);

(c) : (-2.660, -3.400, 8.200);

(d) : (0.0, 0.0, 10.0).

Minimum : z = (-1.0, -10.0, 1.0).

F(z) = 0.

12. Biggs's EXP4 function (Biggs(1971)), n = 4:

$$F(x) \equiv \sum_{i=1}^{10} \left\{ x_3 \exp(t_i x_1) + x_4 \exp(t_i x_2) - \exp(-t_i) + 5\exp(-10t_i) \right\}^2,$$

where $t_i = 0.1 \times i$.

Starting points:-

(a) : (-1.0, -2.0, 1.0, -1.0);

(b) : (0.0, -5.0, -1.0, -3.0);

(c) : (-3.562, -3.816, 51.440, -54.060);

(d) : (-5.0, -5.100, 100.0, -101.0).

Minimum : z = (-1.00, -10.0, 1.0, -5.0).

F(z) = 0.

13. Biggs's EXP6 function (Biggs(1971)), n = 6:

$$F(x) \equiv \sum_{i=1}^{13} \left\{ x_4 \exp(t_i x_1) + x_5 \exp(t_i x_2) + x_6 \exp(x_3 t_i) - \exp(-t_i) + 5\exp(-10t_i) - 3\exp(-4t_i) \right\}^2,$$

where t = 0.1 × i.

Starting points:-

(a) : (-1.0, -9.0, -5.0, 1.0, -4.0, 3.0);

(b) : (1.0, -8.0, -5.0, 1.0, 1.0, 1.0);

(c) : (1.0, -8.0, 0.0, 1.0, 0.0, 1.0);

(d) : (0.0, -5.0, -3.0, 3.0, -5.0, 3.0).

Minimum : z = (-1.0, -10.0, -4.0, 1.0, -5.0, 3.0).

F(z) = 0.

14. Wood's function (Colville(1968)), n = 4:

$$F(x) \equiv 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10.1\{(x_2 - 1)^2 + (x_4 - 1)^2\} + 19.8(x_2 - 1)(x_4 - 1).$$

Starting points:-

(a) : (-3.0, -1.0, -3.0, -1.0);

(b) : (-3.0, 1.0, -3.0, 1.0);

(c) : (-1.2, 1.0, -1.2, 1.0);

(d) : (-1.2, 1.0, 1.2, 1.0).

Minimum : z = (1.0, 1.0, 1.0, 1.0).

F(z) = 0.

15. The (Scaled) Thermistor function (Meyer and Roth(1972)), n = 3:

$$F(x) \equiv \sum_{i=1}^{16} \left\{10^{-6}x_1 \exp(1000x_2/[t_i + 100x_3]) - y_i\right\}^2,$$

where $t_i = 5(9 + i)$ and the y_i's are given as follows:

i	y_i	i	y_i	i	y_i	i	y_i
1	3.4780	2	2.8610	3	2.3650	4	1.9630
5	1.6370	6	1.3720	7	1.1540	8	0.9744
9	0.8261	10	0.7030	11	0.6005	12	0.5147
13	0.4427	14	0.3820	15	0.3307	16	0.2872

Starting points:-

(a) : (2.0, 4.0, 2.5);

(b) : (0.6, 6.0, 3.5);

(c) : (0.1, 7.0, 5.0);

(d) : (0.7, 5.5, 3.0).

Minimum (to 6 decimal places) :

z = (0.560964, 6.181344, 3.452235).

F(z) = 8.794586D-007.

16. The Weibull function (Shanno(1970)), n = 3:

$$F(x) \equiv \sum_{i=1}^{99} \left\{ \exp\left[\frac{-(t_i - x_3)^{x_2}}{x_1} \right] - y_i \right\}^2,$$

where $y_i = 0.1 \times i$, and $t_i = 25 + [-50 \ln(y_i)]^{2/3}$.

Starting points:-

(a) : (250.0, 0.3, 5.0);

(b) : (200.0, 0.1, 40.0);

(c) : (25.0, -2.0, 0.0);

(d) : (10.0, 0.0, 10.0).

Minimum : z = (50.0, 1.5, 25.0).

$F(z) = 0.$

17. The "F55" function (Coope(1987)), n = 55:

$$F(x) \equiv \sum_{i=1}^{51} \left\{ [x_{52} + x_i(x_{53} + x_i(x_{54} + x_i x_{55})) - y_i]^2 + (x_i - d_i)^2 \right\}$$

where $d_i = 0.125664(i - 1)$ and $y_i = \sin(d_i)$.

Starting points:-

(a) : (ζ_i) where $\zeta_i = (1.0 + 0.5y_i)d_i$,

(b) : (0.1, 0.2, 0.3, 0.4, 0.5, ...).

Minimum (to 6 decimal places) :

z = (0.108478, 0.192415, 0.282734, 0.379755, 0.483873,
0.595543, 0.715084, 0.842362, 0.976068, 1.113187,
1.249515, 1.381617, 1.508329, 1.630525, 1.749986,
1.868517, 1.987563, 2.108105, 2.230704, 2.355580,
2.482700, 2.611857, 2.742731, 2.874934, 3.008039,
3.141600, 3.275161, 3.408266, 3.540469, 3.671344,
3.800500, 3.927620, 4.052496, 4.175094, 4.295633,
4.414680, 4.533213, 4.652676, 4.774874, 4.901584,
5.033684, 5.170012, 5.307130, 5.440841, 5.568112,
5.687657, 5.799323, 5.903443, 6.000459, 6.090783,
6.174718, -0.266278, 1.998109, -0.913557, 0.096931).

F(z) = 1.324701D-001.

18,19,20. Penalty function I (More et al(1981)), n = 10,20,30:

$$F(x) \equiv \sum_{i=1}^{n+1} f_i^2,$$

where

$$f_i = \sqrt{10^{-5}}\ (x_i - 1),\ 1 \le i \le n,$$

and

$$f_{n+1} = \sum_{j=1}^{n} (x_j^2) - 1/4.$$

Starting points:-

(a) : (1.0, 2.0, 3.0, 4.0, 5.0, ... ,n);

(b) : (-1.0, -2.0, -3.0, -4.0, -5.0, ... ,-n).

For n = 10:

Minimum : z = (0.158..., 0.158..., ... 0.158...).

F(z) = 7.087651D-005.

For n = 20:

Minimum : z = (0.1118.., 0.1118.., ... 0.1118..).

F(z) = 1.577771D-004.

For n = 30:

Minimum : z = (0.091290..., 0.091290..., ... 0.091290...).

F(z) = 2.477253D-004.

21. Watson function (More et al(1981)), n = 12:

$$F(x) \equiv \sum_{i=1}^{n} f_i^2(x),$$

where

$$f_i(x) = \sum_{j=2}^{n} (j-1)x_j t_i^{j-2} - \left(\sum_{j=1}^{n} x_j t_i^{j-1} \right)^2 - 1$$

and $t_i = i/29$.

Starting points:-

(a) : (0, 0, 0,... 0);

(b) : (-0.2, -0.1, 0, 0.1, 0.2, -0.2, -0.1, 0, 0.1, 0.2,...).

Minimum (to 6 decimal places) :

z = (0.109295, 1.011879, 0.111943, 0.338277, 0.117308,
0.076604, 0.078204, 0.066746, 0.047244, 0.029444,
0.016837, 0.009065).

F(z) = 3.638624D-010.

Comment: For this function, different starting-points [including (a) and (b)] converge to different solutions and different function values. It therefore seems that the minimum of this function is not well-defined. We have therefore selected (for presentation) the solution corresponding to the least value of the function among all the different solutions.

22,23,24. Variably-Dimensioned function (More et al(1981)),

$n = 20,30,40$:

$$F(x) \equiv \sum_{i=1}^{n} (x_i - 1)^2 + f^2 + f^4,$$

where

$$f = \sum_{j=1}^{n} j(x_j - 1).$$

Starting points:-

(a) : (ζ_j) where $\zeta_j = 1 - j/n$;

(b) : (ζ_j) where $\zeta_j = 1 - j/(2n)$.

Minimum : $z = (1.0,\ 1.0,\ 1.0, \ldots, 1.0)$.

$F(z) = 0$.

25,26. Discrete Boundary Value function (More et al(1981)),

$n = 20,30$:

$$F(x) \equiv \sum_{i=1}^{n} f_i^2(x),$$

where

$$f_i(x) = 2x_i - x_{i-1} - x_{i+1} + (x_i + t_i + 1)^3 h^2/2,$$

$h = 1/(n+1)$; $t_i = i \times h$ and $x_0 = x_{n+1} = 0$.

Starting points:-

(a) : (ζ_j), where $\zeta_j = t_j(t_j - 1)$;

(b) : (1.0, 2.0, 3.0, 4.0, 5.0, ...);

(c) : (-2.0, -1.0, 0.0, 1.0, 2.0, ...);

(d) : (0.1, 0.2, 0.3, 0.4, 0.5, ...).

For $n = 20$:

Minimum (to 6 decimal places) :

z = (-0.023210, -0.045202, -0.065880, -0.085143, -0.102875,
-0.118948, -0.133219, -0.145532, -0.155706, -0.163542,
-0.168814, -0.171265, -0.170604, -0.166498, -0.158564,
-0.146361, -0.129377, -0.107016, -0.078577, -0.043233).

F(z) = 0.

For n = 30:

Minimum (to 6 decimal places) :

z = (-0.015858, -0.031171, -0.045909, -0.060044, -0.073543,
-0.086372, -0.098496, -0.109874, -0.120464, -0.130221,
-0.139096, -0.147037, -0.153985, -0.159879, -0.164652,
-0.168230, -0.170534, -0.171477, -0.170965, -0.168892,
-0.165145, -0.159599, -0.152116, -0.142541, -0.130707,
-0.116424, -0.099483, -0.079650, -0.056662, -0.030223).

F(z) = 0.

27,28. Penalty function II (More et al(1981)), n = 20,30:

$$F(x) \equiv \sum_{i=1}^{2n} f_i^2(x),$$

where

$$f_1(x) = x_1 - 0.2;$$

$$f_i(x) = a^{1/2}\left(\exp(0.1x_i) + \exp(0.1x_{i-1}) - y_i\right), \quad 2 \le i \le n;$$

$$f_i(x) = a^{1/2}\left(\exp(0.1x_{i-n+1}) + \exp(-0.1)\right), \quad n < i < 2n;$$

$$f_{2n}(x) = \left(\sum_{i=1}^{n} (n - j + 1) x_j^2 \right) - 1;$$

$a = 10^{-5}$, and $y_i = \exp(0.1i) + \exp(0.1(i-1))$.

Starting points:-

(a) : (0.5, 0.5, 0.5,... 0.5);

(b) : (-0.2, -0.1, 0.0, 0.1, 0.2, ...).

For n = 20:

Minimum (to 6 decimal places) :

z = (0.199755, 0.000014, 0.001032, -0.000385, -0.000189,
0.002849, 0.003016, 0.005432, 0.008565, 0.010638,
0.017121, 0.022539, 0.026583, 0.031431, 0.042211,
0.056818, 0.068841, 0.101368, 0.170759, 0.210229).

F(z) = 6.389820D-003.

For n = 30:

Minimum (to 6 decimal places) :

z = (0.182651, 0.000009, 0.000027, -0.000020, 0.000114,
-0.000026, 0.000020, 0.000136, 0.000101, 0.000028,
0.000085, 0.000213, 0.000245, 0.000181, 0.000183,
0.000333, 0.000502, 0.000533, 0.000498, 0.000638,
0.000992, 0.001272, 0.001330, 0.001612, 0.002527,
0.003381, 0.003878, 0.006984, 0.010609, 0.011533).

F(z) = 6.677290D-002.

29. The Chained Rosenbrock function (Toint(1978)), n = 25:

$$F(x) \equiv 1 + \sum_{i=2}^{n} \left[4\alpha_i (x_i - x_{i-1}^2)^2 + (1 - x_{i-1})^2 \right],$$

where the constants α_i's are given as follows:

i	α_i	i	α_i	i	α_i	i	α_i	i	α_i
1	1.25	2	1.40	3	2.40	4	1.40	5	1.75
6	1.20	7	2.25	8	1.20	9	1.00	10	1.10
11	1.50	12	1.60	13	1.25	14	1.25	15	1.20
16	1.20	17	1.40	18	0.50	19	0.50	20	1.25
21	1.80	22	0.75	23	1.25	24	1.40	25	1.60

Starting points:-

(a) : (-1.0, -1.0,... -1.0);

(b) : (0.1, 0.2, 0.3, 0.4, 0.5, 0.1, 0.2,...).

Minimum : z = (1.0, 1.0, 1.0,... 1.0).

F(z) = 1.0.

30. The Chained Wood's function (Conn et al(1986)), n = 32:

$$F(x) \equiv 1 + \sum_{i \in J} \left[(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 + 90(x_{i+3} - x_{i+2}^2)^2 \right.$$

$$\left. + (1 - x_{i+2})^2 + 10(x_{i+1} + x_{i+3} - 2)^2 + 0.1(x_{i+1} - x_{i+3})^2 \right],$$

where n is a multiple of 4 and J = {1, 3, 5, ..., n-3}.

Starting points:-

(a) : (-3.0, -1.0, -3.0, -1.0, -2.0, 0, -2.0, 0,..., -2.0, 0);

(b) : (-1.0, 2.0, -1.0, 2.0,... -1.0, 2.0).

Minimum : z = (1.0, 1.0, 1.0,..., 1.0).

F(z) = 1.0.

31. The Broyden Triangular function (Conn et al(1986)) n = 30:

$$F(x) \equiv 1 + \sum_{i=1}^{n} |(3 - 2x_i)x_i - x_{i-1} - x_{i+1} + 1|^p,$$

where n is even, p = 2 and $x_0 = x_{n+1} = 0$.

Starting points:-

(a) : (-1.0, -1.0,..., -1.0);

(b) : (-0.2, -0.1, 0.0, 0.1, 0.2, -0.2, -0.1, 0,...).

Minimum (to 6 decimal places) :

z = (-0.390601, -0.476943, -0.495177, -0.498993, -0.499789,
-0.499956, -0.499990, -0.499998, -0.499999, -0.499999,
-0.499999, -0.500000, -0.500000, -0.500000, -0.500000,
-0.500000, -0.500000, -0.499999, -0.499999, -0.499999,
-0.499999, -0.499999, -0.499998, -0.499990, -0.499956,
-0.499789, -0.498993, -0.495177, -0.476943, -0.390601).

F(z) = 1.0.

32. The discretization of a variational problem ("VAR") Conn et al(1986)), n = 45:

$$F(x) \equiv 2\sum_{i=1}^{n} [x_i (x_i - x_{i+1})]/h + 2\lambda h \sum_{i=0}^{n} \left([\exp(x_{i+1}) - \exp(x_i)]/(x_{i+1} - x_i) \right),$$

where $h = 1/(n+1)$, $x_0 = x_{n+1} = 0$ and $\lambda = -3.4$.

Starting points:-

(a) : (ζ_i) where $\zeta_i = 0.1 \times i \times h \times (1-i)$;

(b) : (-2.0, -1.0, 0.0, 1.0, 2.0, -2.0, -1.0, 0.0,...).

Minimum (to 6 decimal places) :

z = (0.068121, 0.134522, 0.199085, 0.261687, 0.322202,
0.380499, 0.436446, 0.489907, 0.540746, 0.588827,
0.634013, 0.676171, 0.715171, 0.750887, 0.783200,
0.811998, 0.837179, 0.858651, 0.876333, 0.890158,
0.900072, 0.906036, 0.908026, 0.906036, 0.900072,
0.890158, 0.876333, 0.858651, 0.837179, 0.811998,
0.783200, 0.750887, 0.715171, 0.676171, 0.634013,
0.588827, 0.540746, 0.489907, 0.436446, 0.380499,
0.322202, 0.261687, 0.199085, 0.134522, 0.068121).

F(z) = -8.517243D+000.

Comment: it appears that Conn et al, in their description of this problem, have omitted a factor of 0.1 in the values they give for the minimum point.

7-2 Program Implementation

To implement the algorithms (including the BFGS method, the algorithm 3-1 (see chapter 3) is modified as follows:

(0) Let $k = 0$, $H = I$, and choose x_0 $(\neq z)$;

(1) $p := -Hg$;

if $k \leq n$, then $p := p/\|p\|_2$;

(2) To find a better estimate for the solution, apply a linear-search with the following α- and β- conditions:

$$F^* \leq F + \alpha t^* p^T g,$$

$$p^T g^* > \beta p^T g,$$

where $\alpha = 0.0001$ and $\beta = 0.9$;

(3) In the case of the new algorithms, compute w using one of the formulas for w in chapter 5 or 6 [as appropriate] and replace y by w.

In the case of the BFGS method,

$y = g^* - g$;

If $k = 0$ and $n \geq 10$, scale the matrix H using (2-51c):

$H\hat{} = (s^T y/y^T H y)H$ (see section 2-4-3).

(4) Update H or H^ (as appropriate) to produce H^* using (2-26) [the BFGS formula], using y or w, as appropriate;

(5) If $\|g^*\|_2 < \varepsilon$ then stop (see the comment below)

ELSE

$H := H^*$, $x := x^*$, $g := g^*$, $k := k + 1$, and go to (1).

Comment: The limited accuracy of computation and the precise nature of each function have made it advisable, in our judgment, to utilize different values of ε for different functions. For this reason, we have chosen (after experimentation) $\varepsilon = 10^{-8}$ for Biggs's EXP4 and EXP6 functions, $\varepsilon = 5 \times 10^{-5}$ for the "F55" function, the Penalty function II and the "VAR" function and, finally, $\varepsilon = 10^{-6}$ for all other functions.

The programs were implemented in PASCAL, using double-precision variables throughout with overflow and underflow limits of 1.8×10^{308} and 1.8×10^{-308}, respectively, under Unix System-V on a Senate 68000 microcomputer at the University of Essex.

7-3 Numerical Results and Discussion

To evaluate the new algorithms we need some criteria. The reasons for and motivation of the criteria which have been used to assess the algorithms will be explained in subsection 7-3-1.

Due to the limitation of space and the number of algorithms we have considered (about 30), it is not possible to present all the numerical results in detail. On these grounds, therefore, we have organized the remainder of this section as follows: in subsection 7-3-2, we present an overall view of the numerical results for each algorithm, utilizing various statistics and a diagram which compares each algorithm with the BFGS method. We compare and discuss the numerical results of all algorithms arising from each

model, in order to select the best algorithm for each model. This is carried out in subsection 7-3-3, using a summary table of results.

Full numerical results of the algorithms chosen in subsection 7-3-3 will be presented in 7-3-4, where we will discuss and compare the results and select the best.

In every subsection, we will describe the structure and components of tables, statistics or diagrams, as appropriate.

Finally, in subsection 7-3-5, we consider numerical evidence concerning the convergence of the best algorithms selected in subsection 7-3-4.

7-3-1 Criteria for Comparison

In quasi-Newton methods, the time spent evaluating the function and gradients at the various points required may well dominate the time for the whole minimization process. It is therefore desirable to utilize the number of function evaluations (denoted by "NF") and/or gradient evaluations as one of the most important criteria for comparing different algorithms. Furthermore, the time for housekeeping operations (in our case, the time spent updating the matrices, evaluating the vectors, calling subroutines, setting of variables and entry conditions, and, finally, performing input and output routines) depends directly upon the number of iterations (denoted by "NI"). Thus, NI may be considered as another criterion for comparison. We

will use the figures NF and NI for the 32 test problems and a total of 100 starting-points to compare the new algorithms with the BFGS algorithm and with each other.

To compare a number of algorithms simultaneously, we may award a nominal score of one to an algorithm which "best" solves a given test problem. By "best", we mean that algorithm which solves the problem with the lowest NF total. Ties (if any) are resolved on the basis of NI figures. The summation of scores for each algorithm may then be used for comparison of the algorithms.

Finally, the type of equation (linear or nonlinear) by which we obtain the parameter(s) in each algorithm may be taken as another criterion for comparison.

7-3-2 Statistics and Diagrams for All Algorithms

In this section we present statistics and a diagram for every algorithm to compare each new algorithm with the BFGS method. The statistics and a diagram possesses the following contents:

(i) Summation of NF and NI for each algorithm;

(ii) a comparison of the relative performance of the two algorithms;

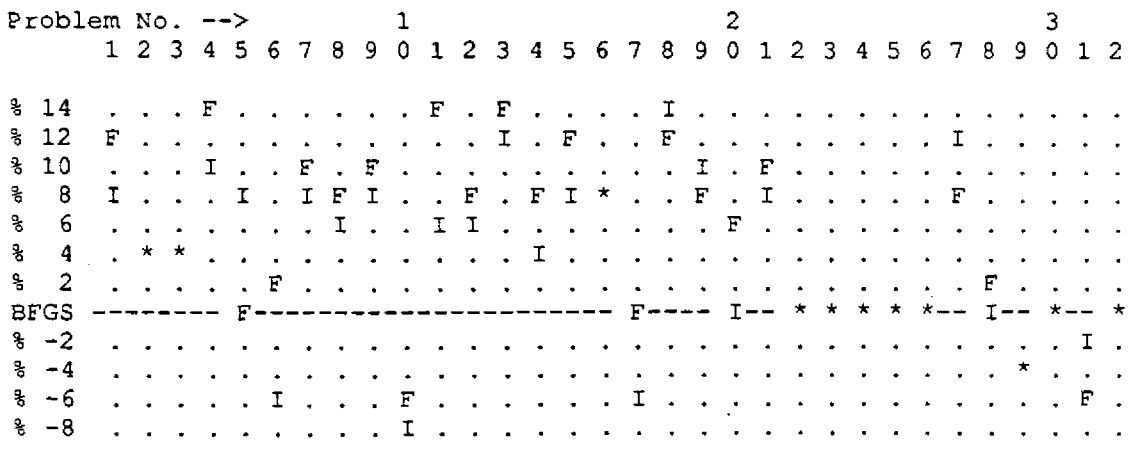
(iii) a diagram which contains the characters "F", "I" and "*". "F" and "I" denote the percentage change (in relation to NF and NI, respectively) of an algorithm for each test problem in comparison with the BFGS method. "*" is used where "F"

and "I" would occur together. To have a clear diagram, we have rounded each percentage figure to the nearest even number. For percentages which are less than -10, we print only those lines which contain the characters "F", "I" or "*". The horizontal and vertical axes are used to denote test problem numbers and the percentages, respectively.

ALGORITHM Ac0gr (COMPARISON WITH BFGS)

	BFGS	Ac0gr
GRAND SUM	(8762, 12384)	(8310, 11561)

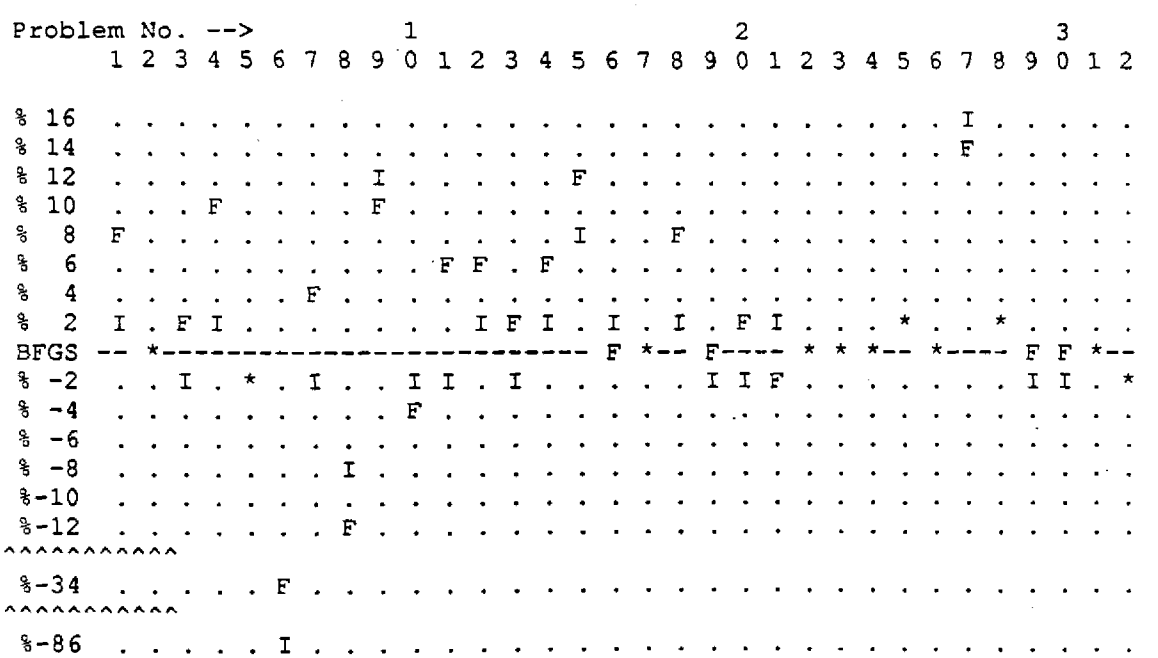
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	63	59
The SAME as	BFGS	14	21
WORSE than	BFGS	23	20



ALGORITHM Ac0t (COMPARISON WITH BFGS)

	BFGS	Ac0t
GRAND SUM	(8762, 12384)	(8691, 11969)

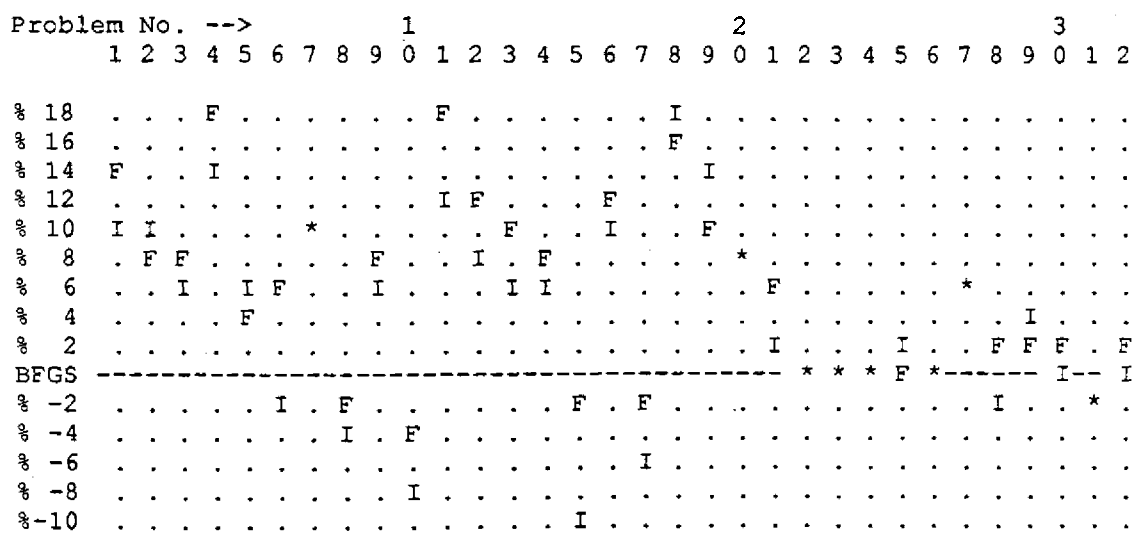
	FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than BFGS	51	45
The SAME as BFGS	29	36
WORSE than BFGS	20	19



ALGORITHM Aclgr (COMPARISON WITH BFGS)

	BFGS	Aclgr
GRAND SUM	(8762, 12384)	(8339, 11525)

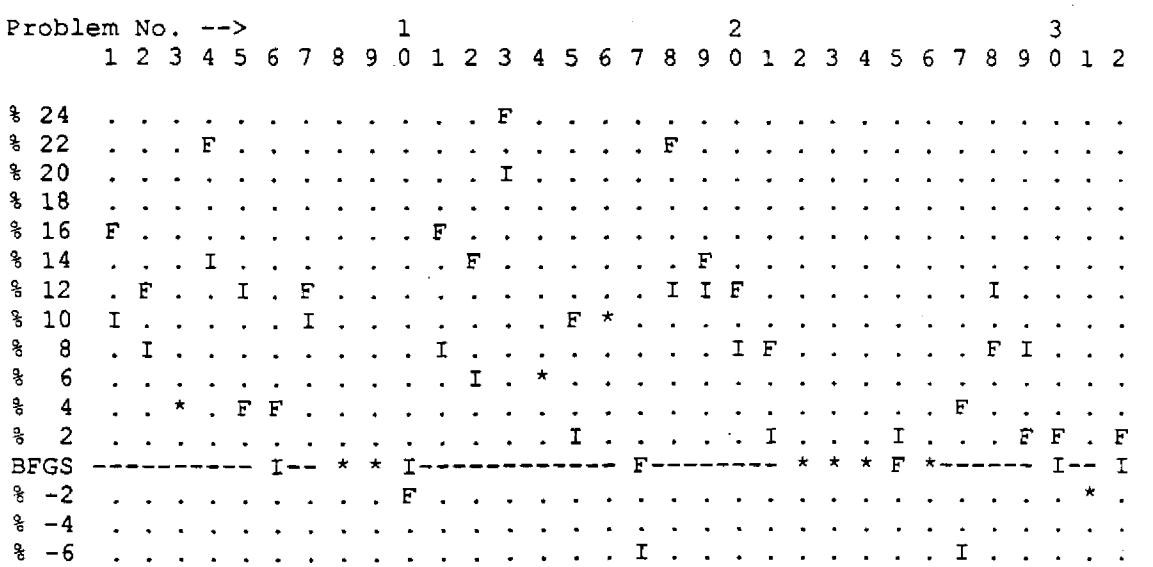
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	64	62
The SAME as	BFGS	18	17
WORSE than	BFGS	18	21



ALGORITHM Af (COMPARISON WITH BFGS)

	BFGS	Af
GRAND SUM	(8762, 12384)	(8212, 11256)

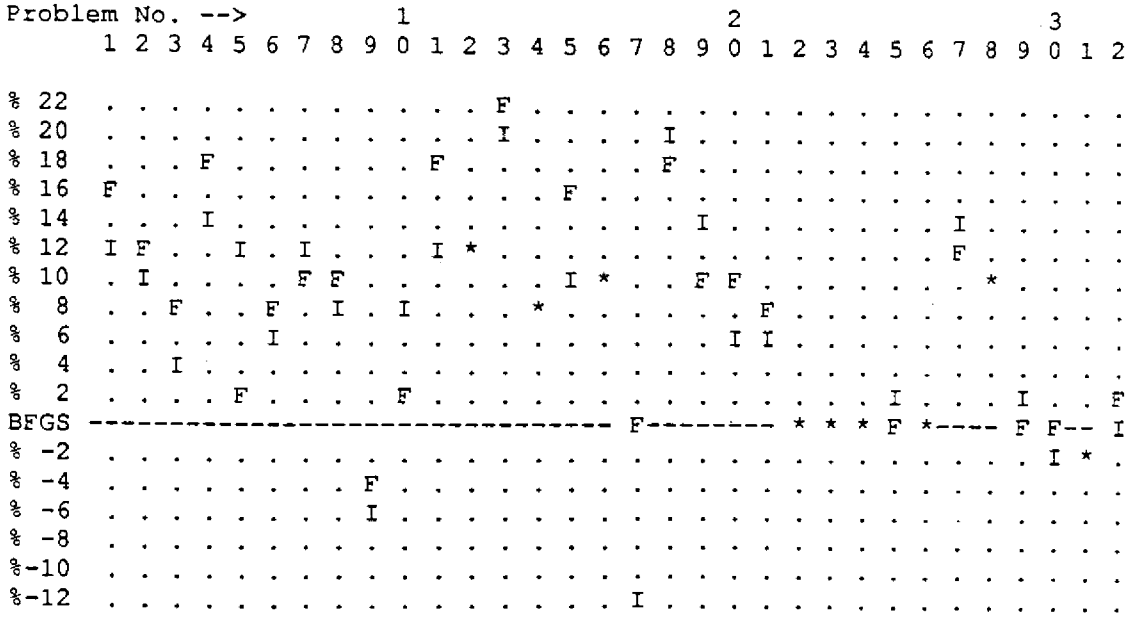
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	67	59
The SAME as	BFGS	14	22
WORSE than	BFGS	19	19



ALGORITHM Bc0gr (COMPARISON WITH BFGS)

	BFGS	Bc0gr
GRAND SUM	(8762, 12384)	(8028, 11220)

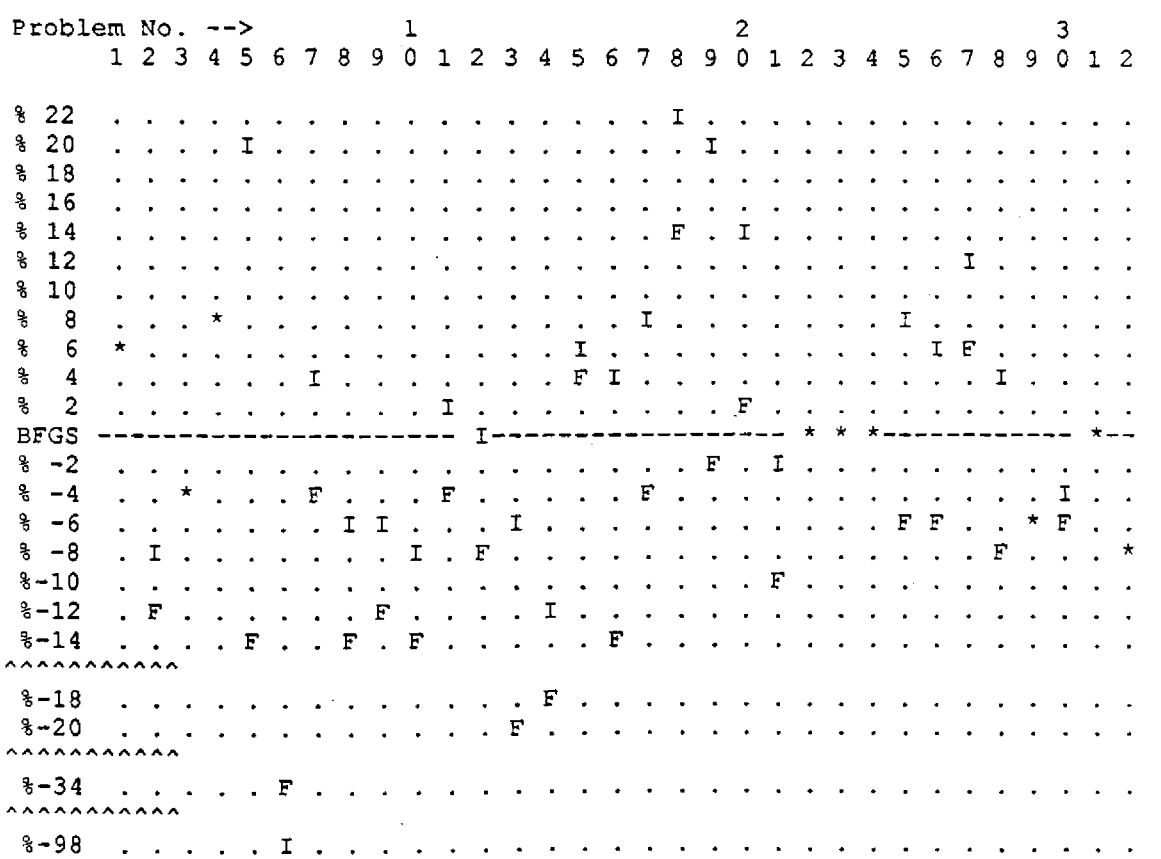
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	71	64
The SAME as	BFGS	15	17
WORSE than	BFGS	14	19



ALGORITHM Bc0t (COMPARISON WITH BFGS)

	BFGS	Bc0t
GRAND SUM	(8762, 12384)	(8575, 12876)

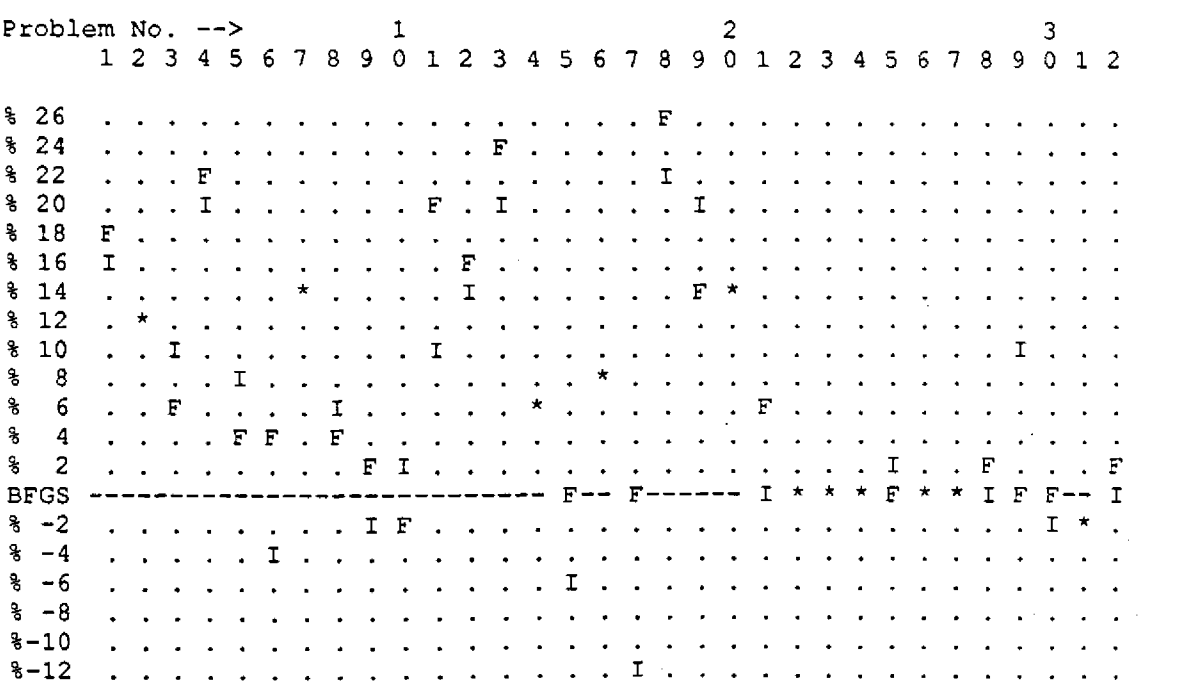
	FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than BFGS	20	41
The SAME as BFGS	69	46
WORSE than BFGS	11	13



ALGORITHM Bc_01 (COMPARISON WITH BFGS)

	BFGS	Bc_01
GRAND SUM	(8762, 12384)	(8108, 11327)

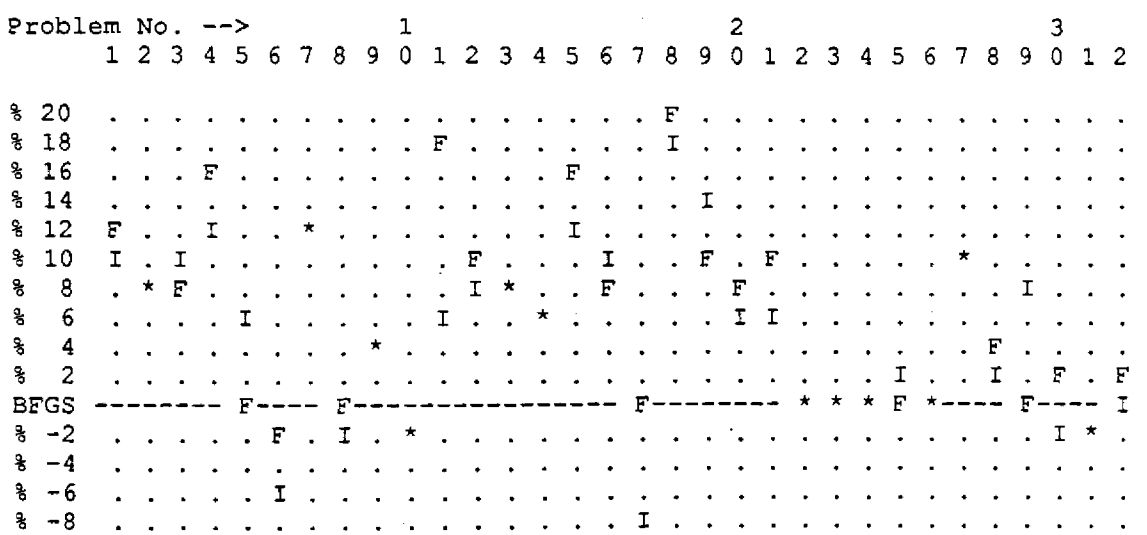
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	63	60
The SAME as	BFGS	22	24
WORSE than	BFGS	15	16



ALGORITHM Bc/01 (COMPARISON WITH BFGS)

	BFGS	Bc/01
GRAND SUM	(8762, 12384)	(8180, 11439)

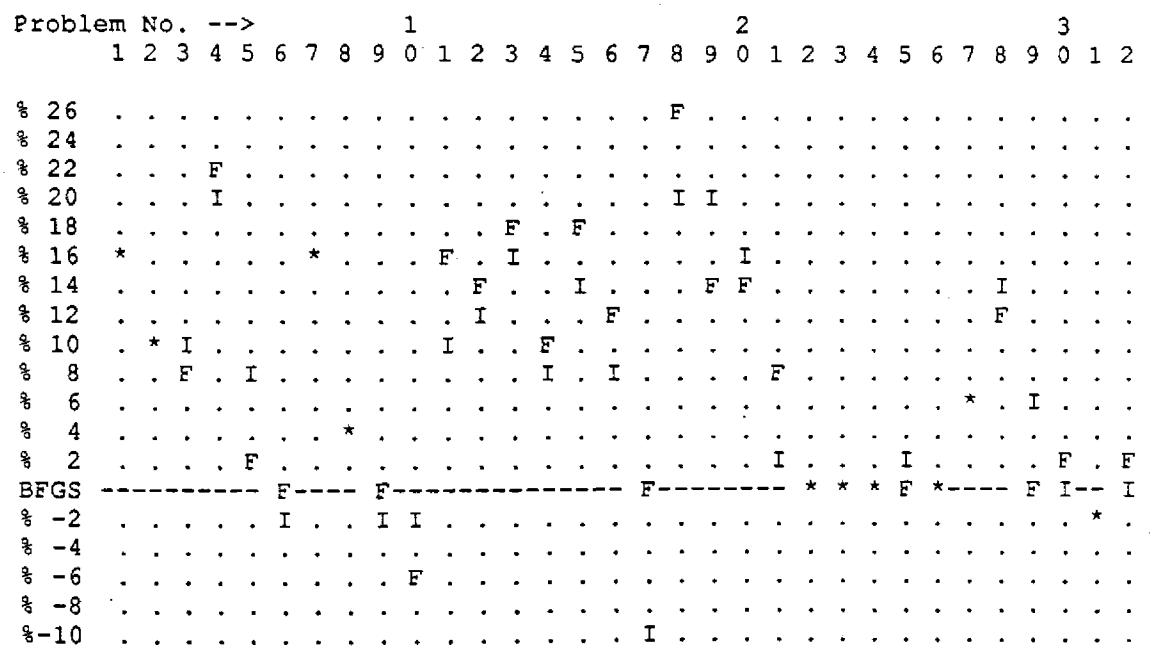
	FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than BFGS	69	65
The SAME as BFGS	15	18
WORSE than BFGS	16	17



ALGORITHM Bf (COMPARISON WITH BFGS)

	BFGS	Bf
GRAND SUM	(8762, 12384)	(7888, 11110)

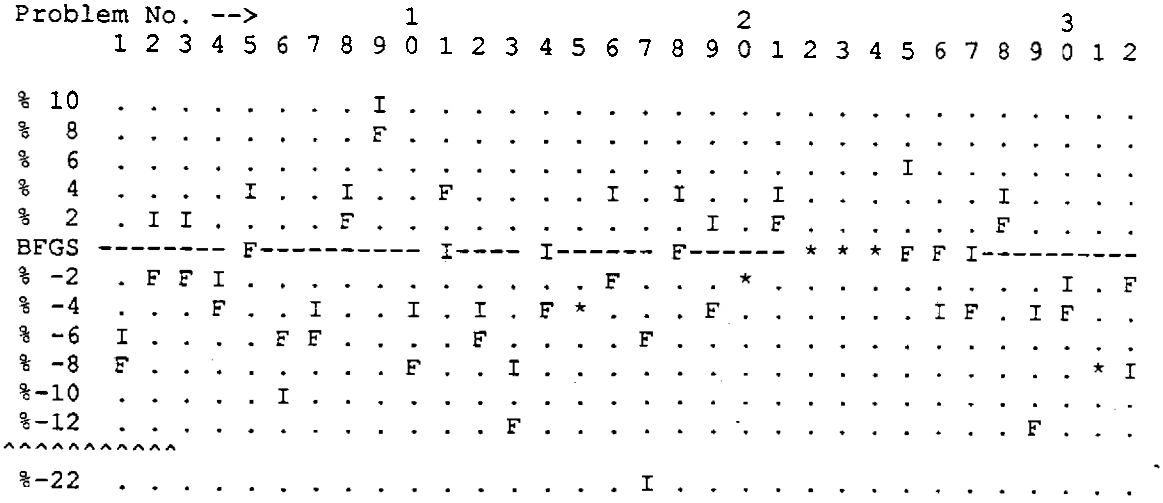
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	70	62
The SAME as	BFGS	14	16
WORSE than	BFGS	16	22



ALGORITHM Cc0gr (COMPARISON WITH BFGS)

	BFGS	Cc0gr
GRAND SUM	(8762, 12384)	(8908, 12793)

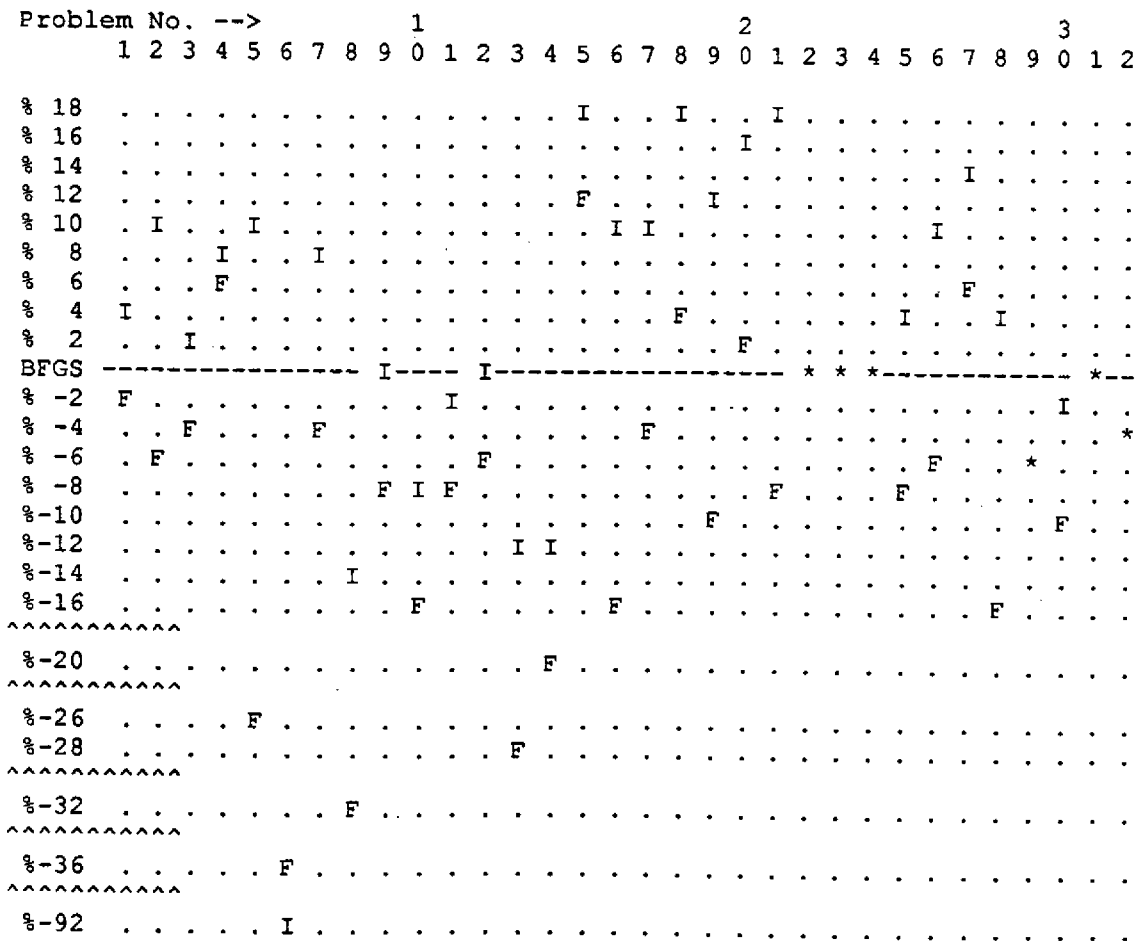
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	20	31
The SAME as	BFGS	60	49
WORSE than	BFGS	20	20



ALGORITHM Cc0t (COMPARISON WITH BFGS)

	BFGS	Cc0t
GRAND SUM	(8762, 12384)	(8406, 13135)

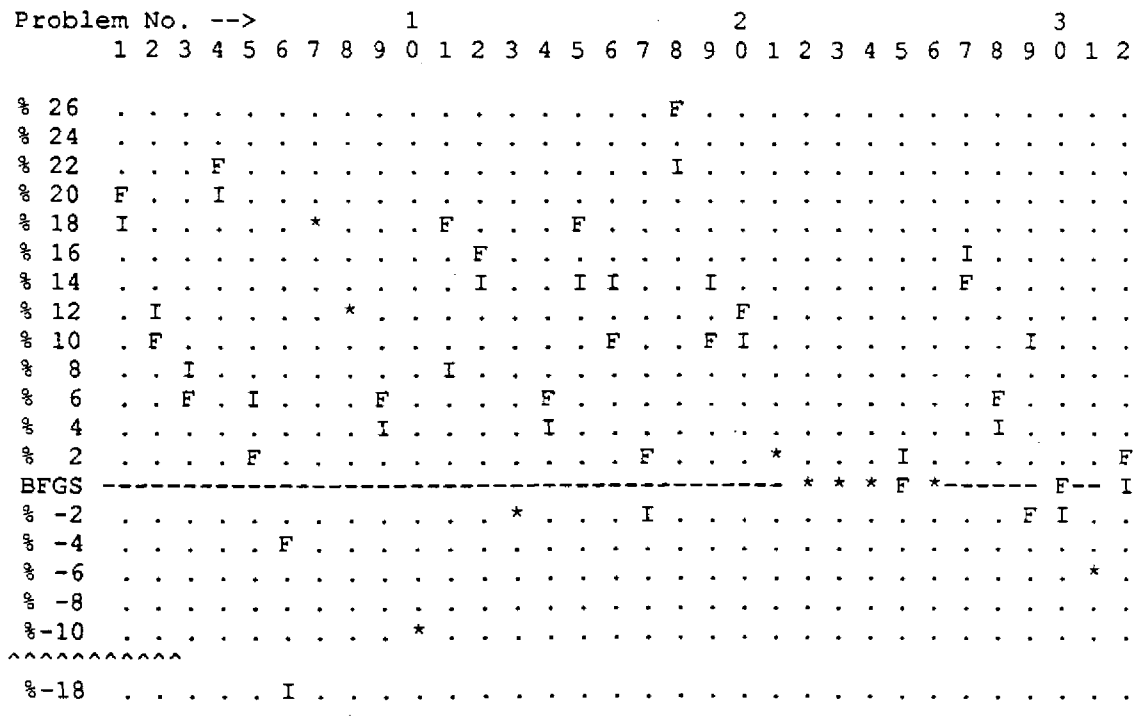
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	14	46
The SAME as	BFGS	76	41
WORSE than	BFGS	10	13



ALGORITHM Cf (COMPARISON WITH BFGS)

	BFGS	Cf
GRAND SUM	(8762, 12384)	(7960, 11210)

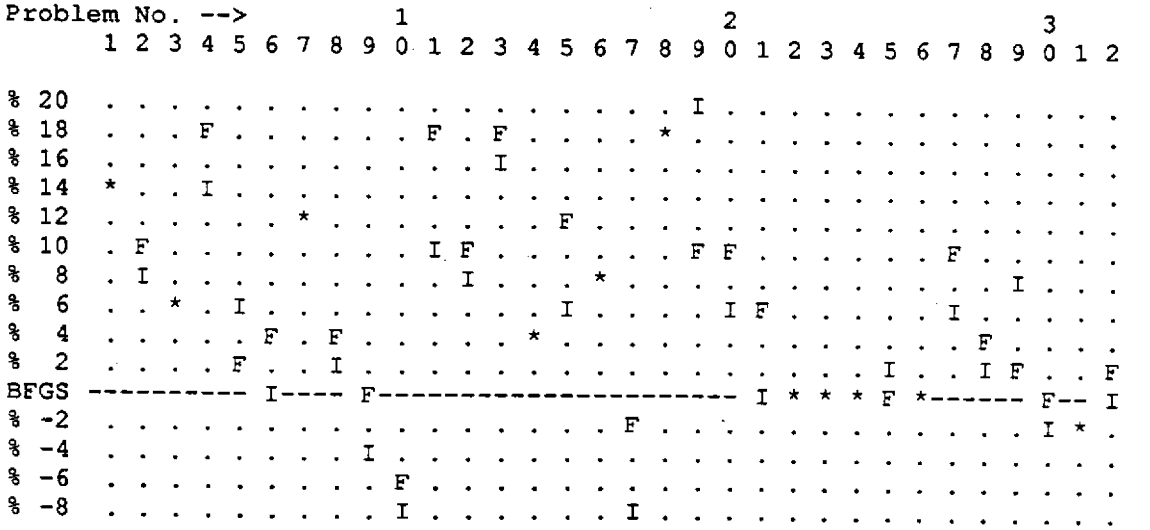
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	65	61
The SAME as	BFGS	14	18
WORSE than	BFGS	21	21



ALGORITHM Dc0c (COMPARISON WITH BFGS)

	BFGS	Dc0c
GRAND SUM	(8762, 12384)	(8197, 11401)

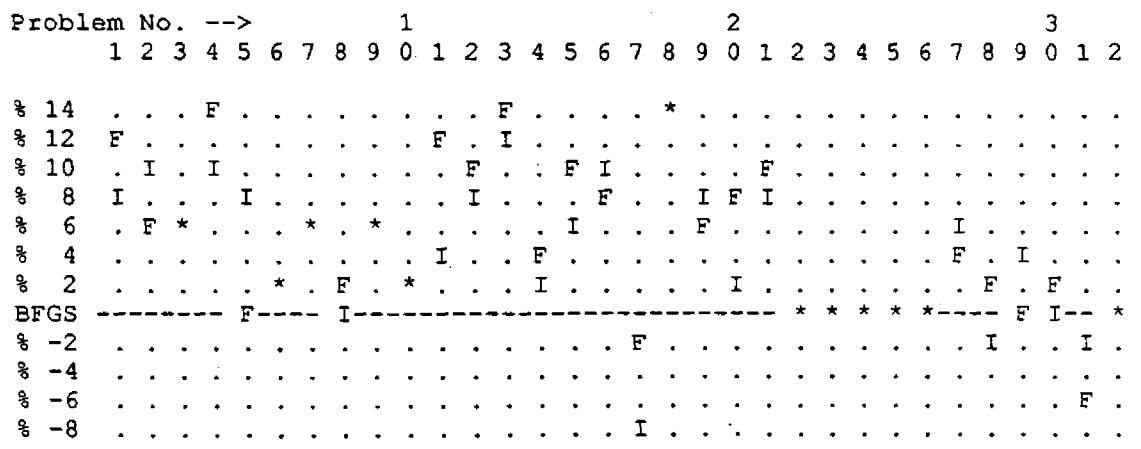
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	66	62
The SAME as	BFGS	16	20
WORSE than	BFGS	18	18



ALGORITHM Dc0q (COMPARISON WITH BFGS)

	BFGS	Dc0q
GRAND SUM	(8762, 12384)	(8322, 11592)

		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	64	63
The SAME as	BFGS	12	15
WORSE than	BFGS	24	22



ALGORITHM Dc0t (COMPARISON WITH BFGS)

	BFGS	Dc0t
GRAND SUM	(8762, 12384)	(8728, 15261)

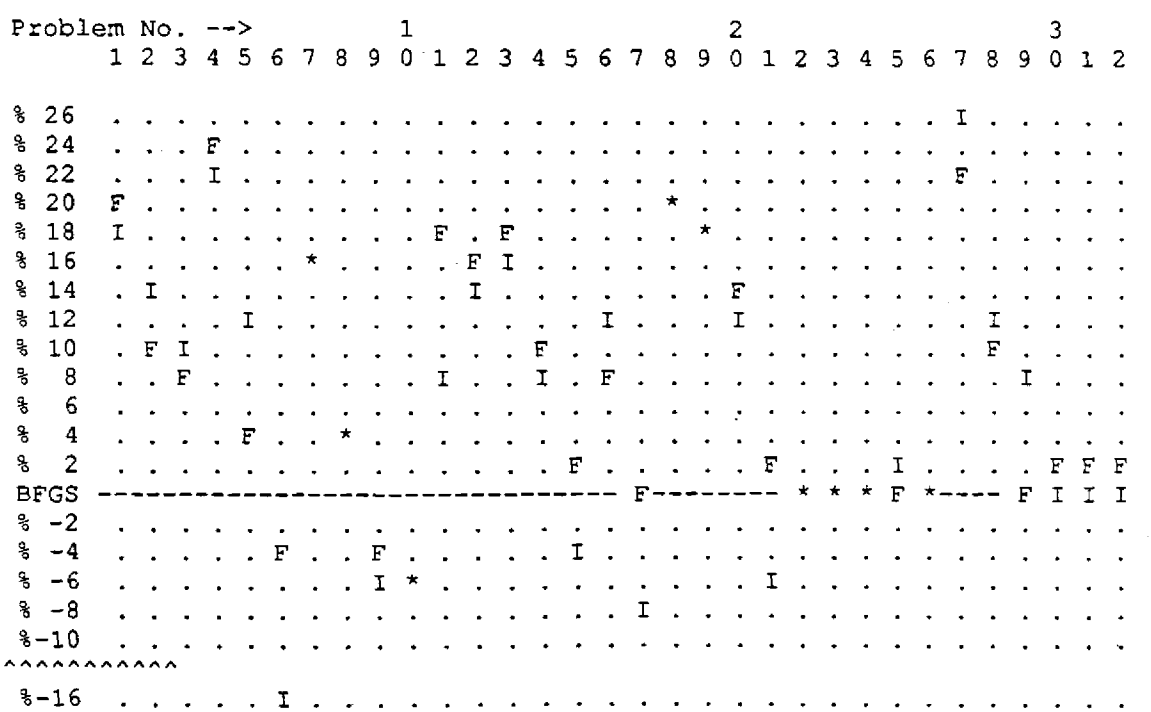
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	3	37
The SAME as	BFGS	89	50
WORSE than	BFGS	8	13

Because of the poor performance of this algorithm and the large size of the diagram, we have chosen to omit the diagram in this case.

ALGORITHM Df (COMPARISON WITH BFGS)

	BFGS	Df
GRAND SUM	(8762, 12384)	(7943, 11132)

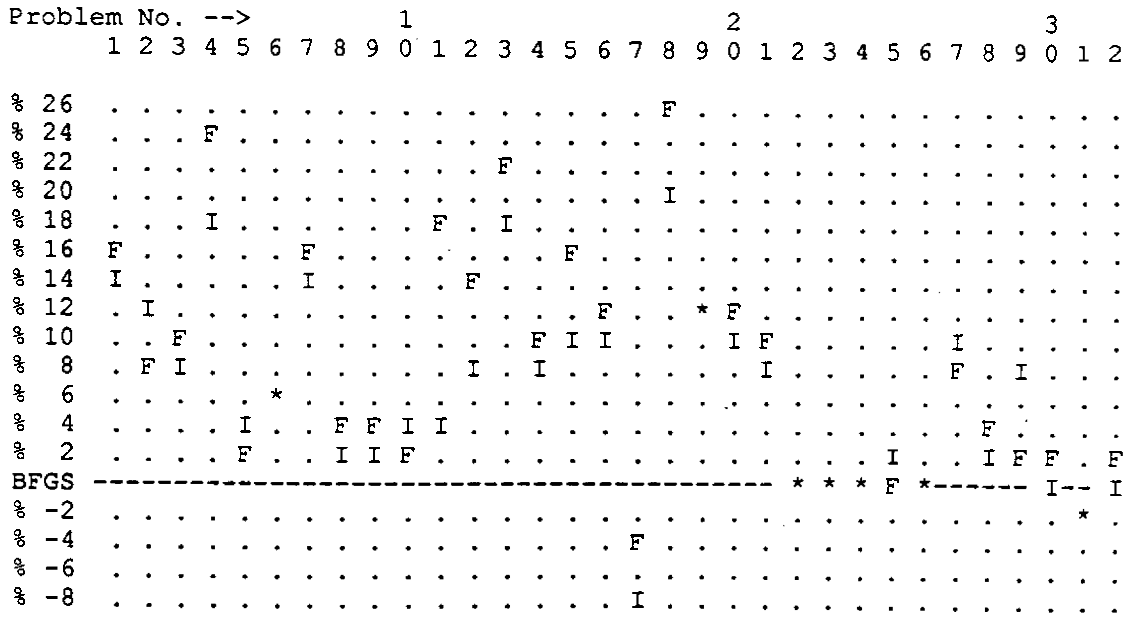
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	65	61
The SAME as	BFGS	20	21
WORSE than	BFGS	15	18



ALGORITHM Ec0gr (COMPARISON WITH BFGS)

	BFGS	Ec0gr
GRAND SUM	(8762, 12384)	(8029, 11090)

		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	71	63
The SAME as	BFGS	12	20
WORSE than	BFGS	17	17



ALGORITHM Ec0t (COMPARISON WITH BFGS)

	BFGS	Ec0t
GRAND SUM	(8762, 12384)	(6875, 15311)

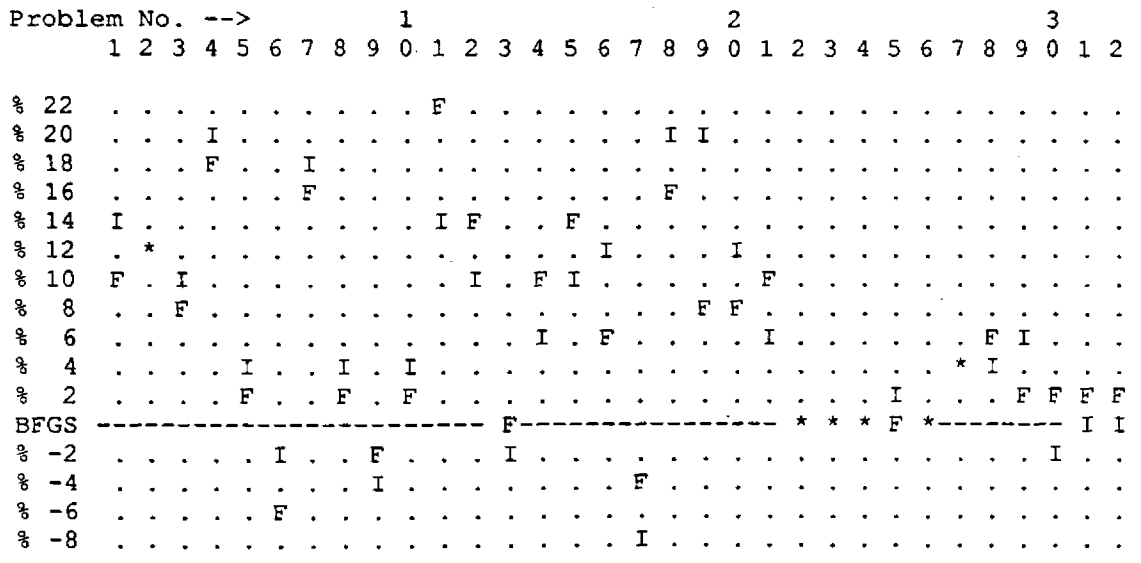
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	6	81
The SAME as	BFGS	84	8
WORSE than	BFGS	10	11

Because of the poor performance of this algorithm and the large size of the diagram, we have chosen to omit the diagram in this case.

ALGORITHM EcLS (COMPARISON WITH BFGS)

	BFGS	EcLS
GRAND SUM	(8762, 12384)	(8065, 11445)

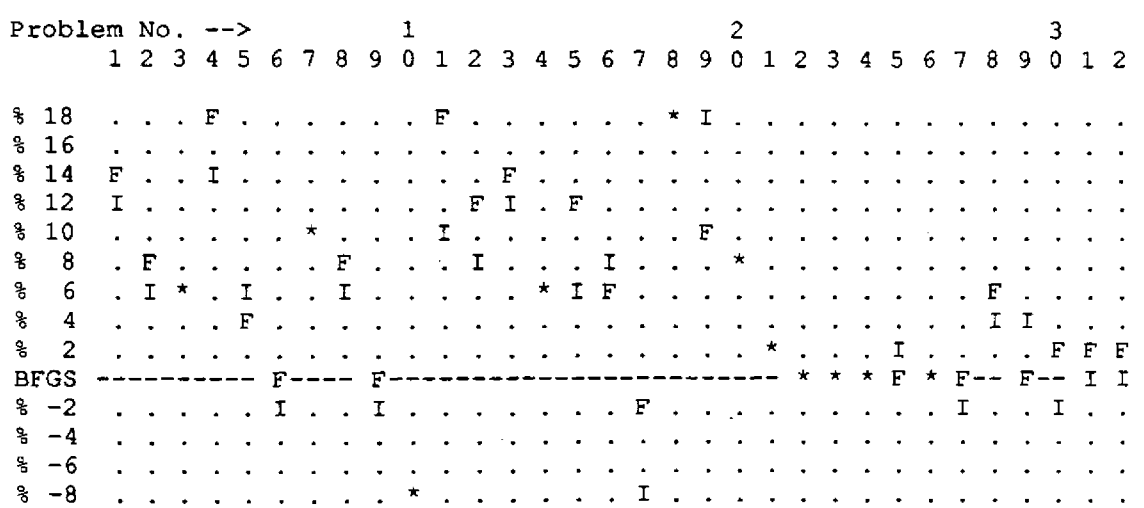
	FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than BFGS	63	64
The SAME as BFGS	20	20
WORSE than BFGS	17	16



ALGORITHM Ec/01 (COMPARISON WITH BFGS)

	BFGS	Ec/01
GRAND SUM	(8762, 12384)	(8224, 11471)

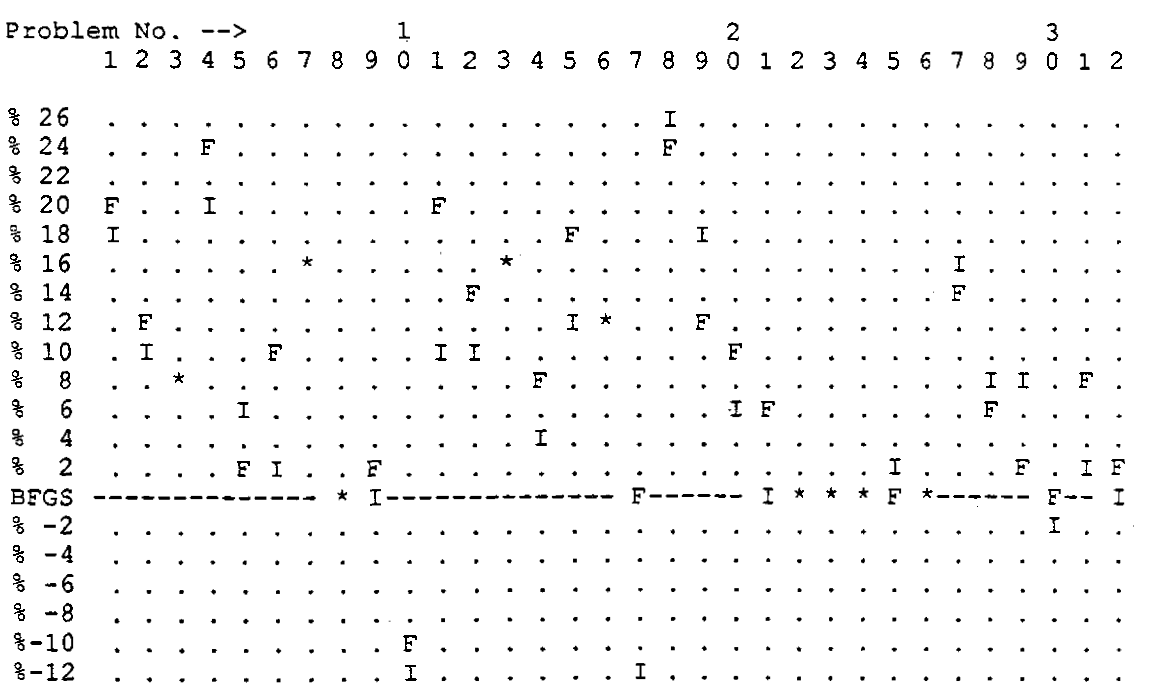
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	64	61
The SAME as	BFGS	19	21
WORSE than	BFGS	17	18



ALGORITHM Ef (COMPARISON WITH BFGS)

	BFGS	Ef
GRAND SUM	(8762, 12384)	(7971, 11057)

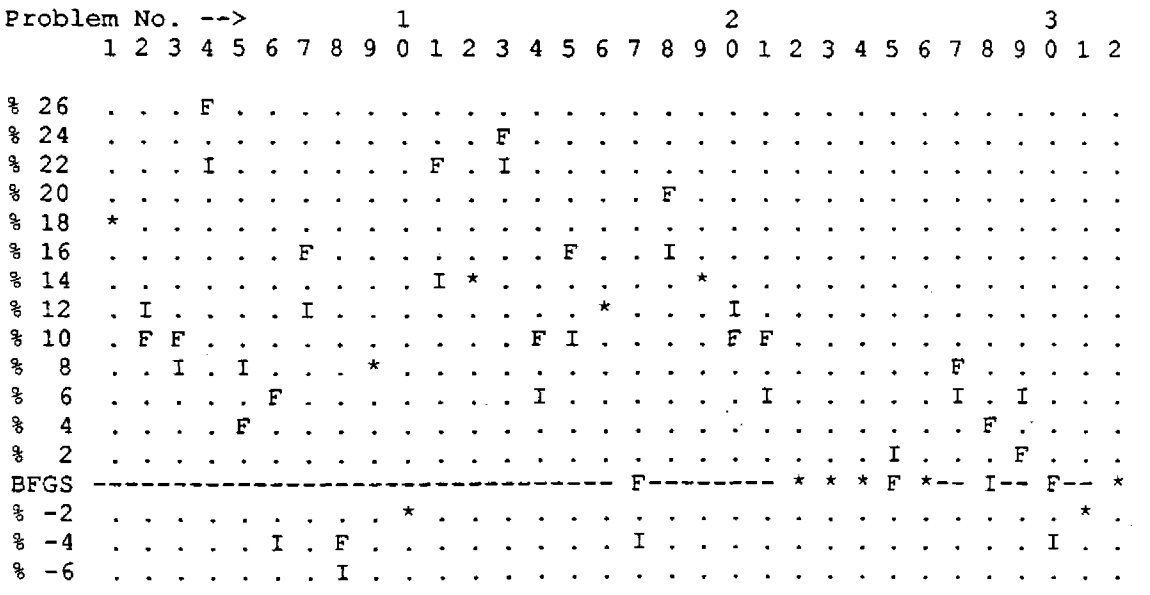
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	68	62
The SAME as	BFGS	17	21
WORSE than	BFGS	15	17



ALGORITHM FLSfc1 (COMPARISON WITH BFGS)

	BFGS	FLSfc1
GRAND SUM	(8762, 12384)	(7969, 11108)

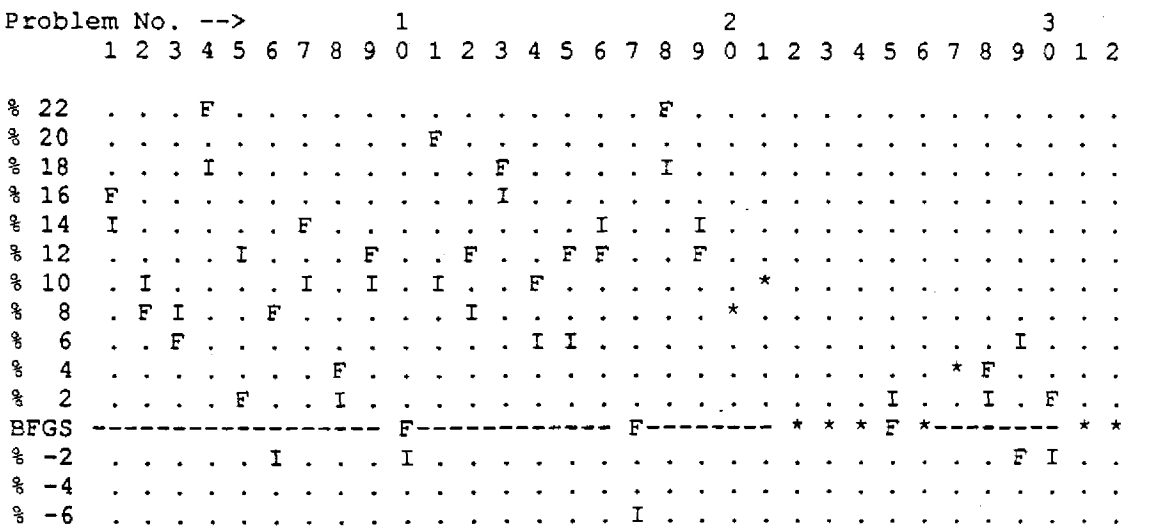
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	70	65
The SAME as	BFGS	13	18
WORSE than	BFGS	17	17



ALGORITHM Ff (COMPARISON WITH BFGS)

	BFGS	Ff
GRAND SUM	(8762, 12384)	(8090, 11266)

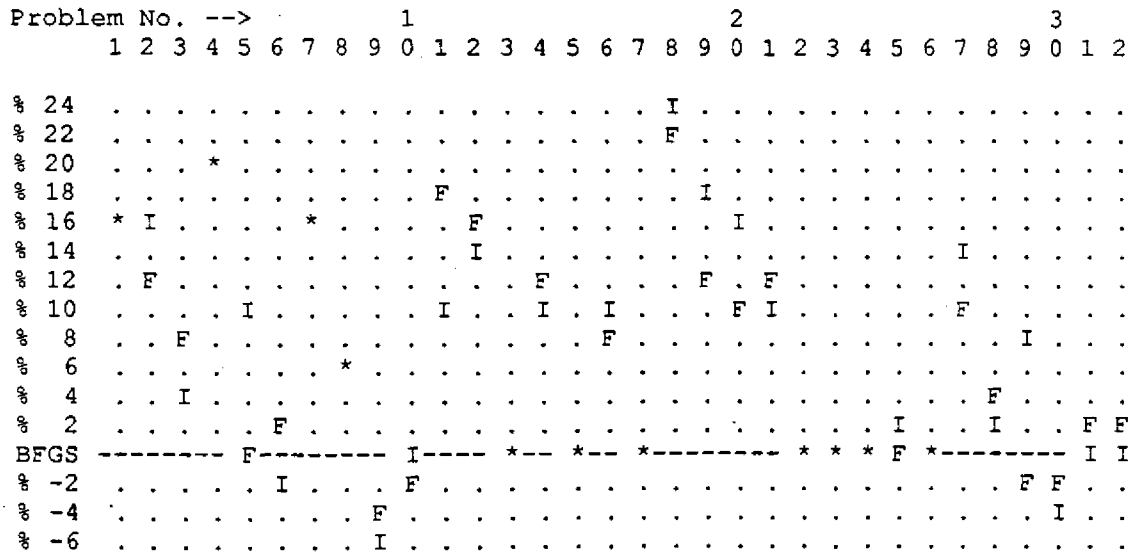
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	71	65
The SAME as	BFGS	11	13
WORSE than	BFGS	18	22



ALGORITHM Gfc0c (COMPARISON WITH BFGS)

	BFGS	Gfc0c
GRAND SUM	(8762, 12384)	(8058, 11387)

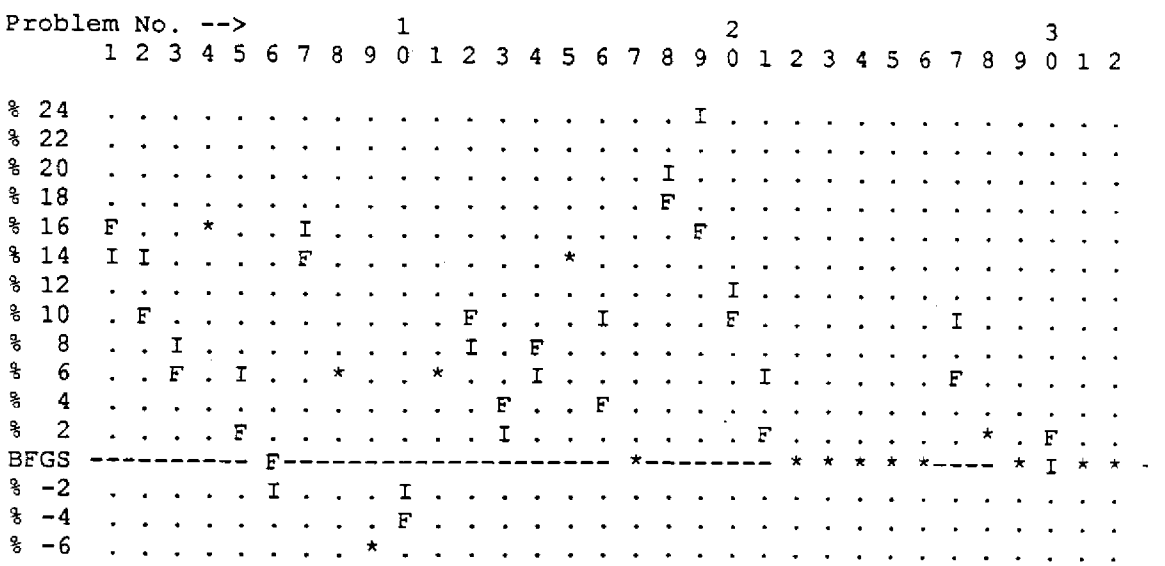
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	65	65
The SAME as	BFGS	20	20
WORSE than	BFGS	15	15



ALGORITHM Gfc0t (COMPARISON WITH BFGS)

	BFGS	Gfc0t
GRAND SUM	(8762, 12384)	(8050, 11471)

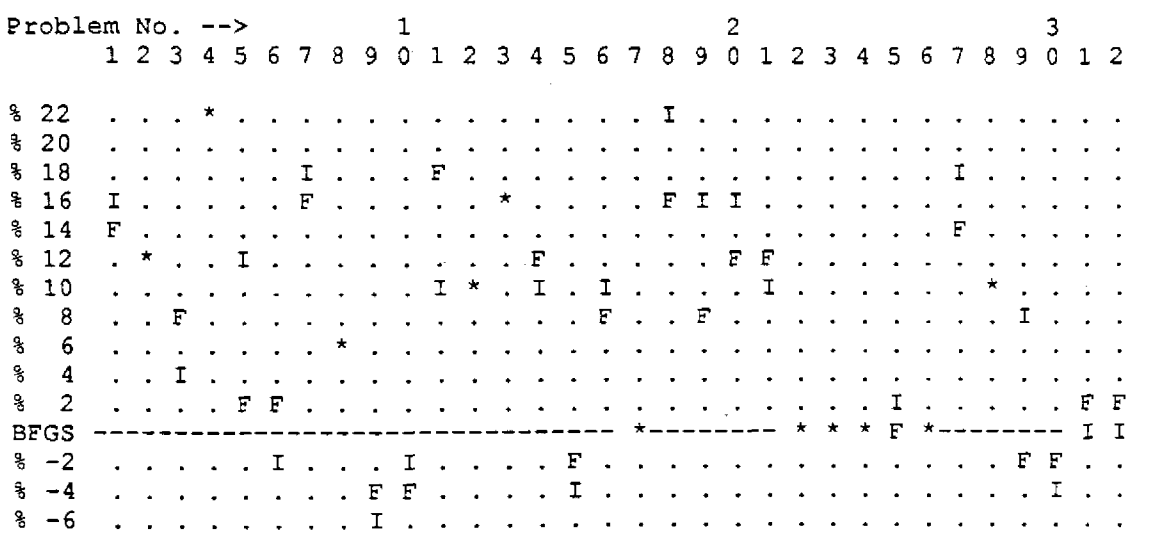
	FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than BFGS	59	55
The SAME as BFGS	12	12
WORSE than BFGS	29	33



ALGORITHM Hfc0c.a (COMPARISON WITH BFGS)

	BFGS	Hfc0c.a
GRAND SUM	(8762, 12384)	(7984, 11350)

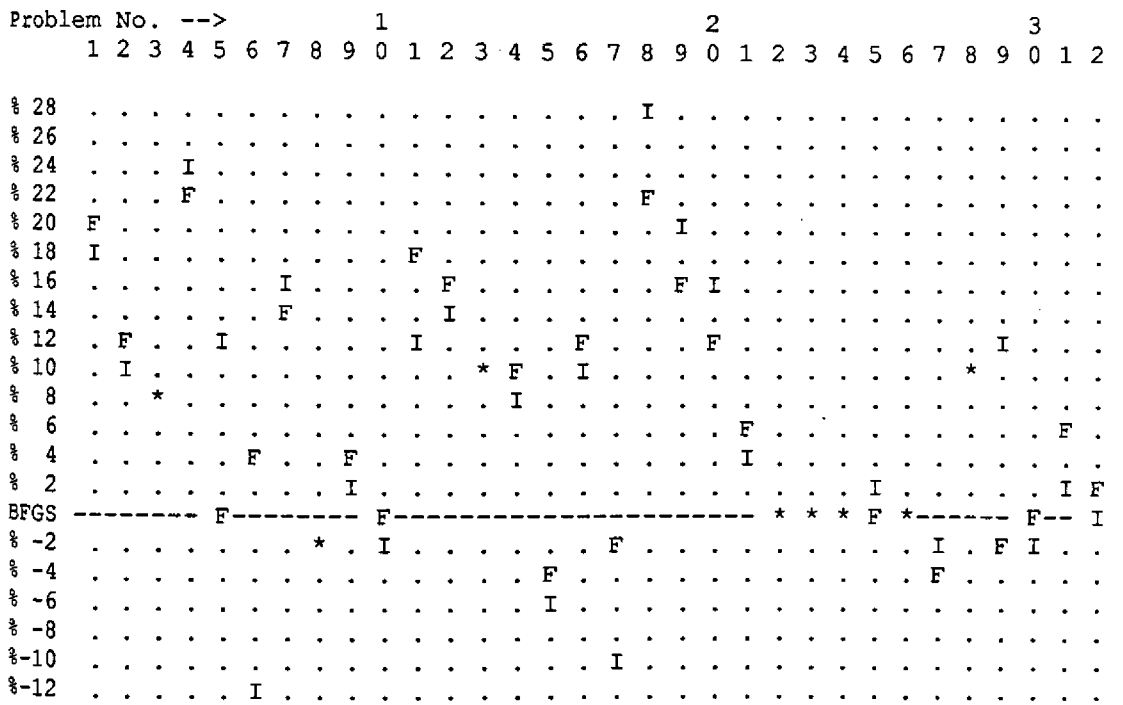
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	68	65
The SAME as	BFGS	17	18
WORSE than	BFGS	15	17



ALGORITHM Hfc0c.b (COMPARISON WITH BFGS)

	BFGS	Hfc0c.b
GRAND SUM	(8762, 12384)	(8026, 11360)

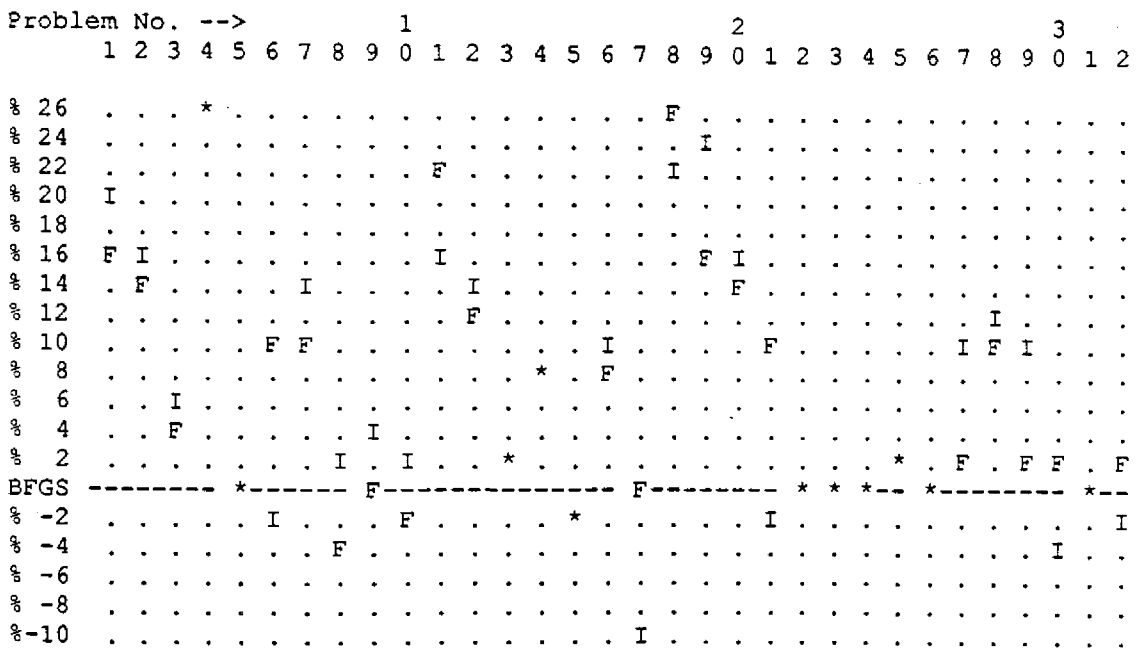
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	63	63
The SAME as	BFGS	21	22
WORSE than	BFGS	16	15



ALGORITHM Hfc0q.a (COMPARISON WITH BFGS)

	BFGS	Hfc0q.a
GRAND SUM	(8762, 12384)	(7937, 11361)

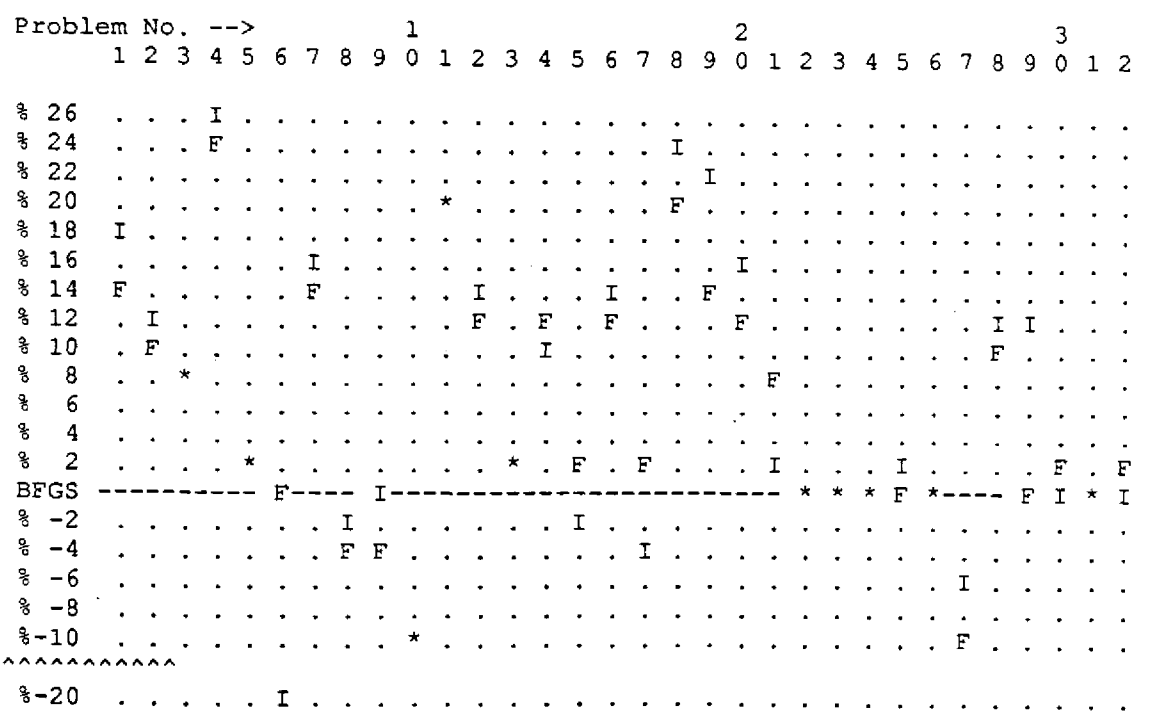
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	63	61
The SAME as	BFGS	22	24
WORSE than	BFGS	15	15



ALGORITHM Hfc0q.b (COMPARISON WITH BFGS)

	BFGS	Hfc0q.b
GRAND SUM	(8762, 12384)	(7983, 11453)

		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	60	61
The SAME as	BFGS	23	20
WORSE than	BFGS	17	19



ALGORITHM Hfc0t.a (COMPARISON WITH BFGS)

	BFGS	Hfc0t.a
GRAND SUM	(8762, 12384)	(16256, 18753)

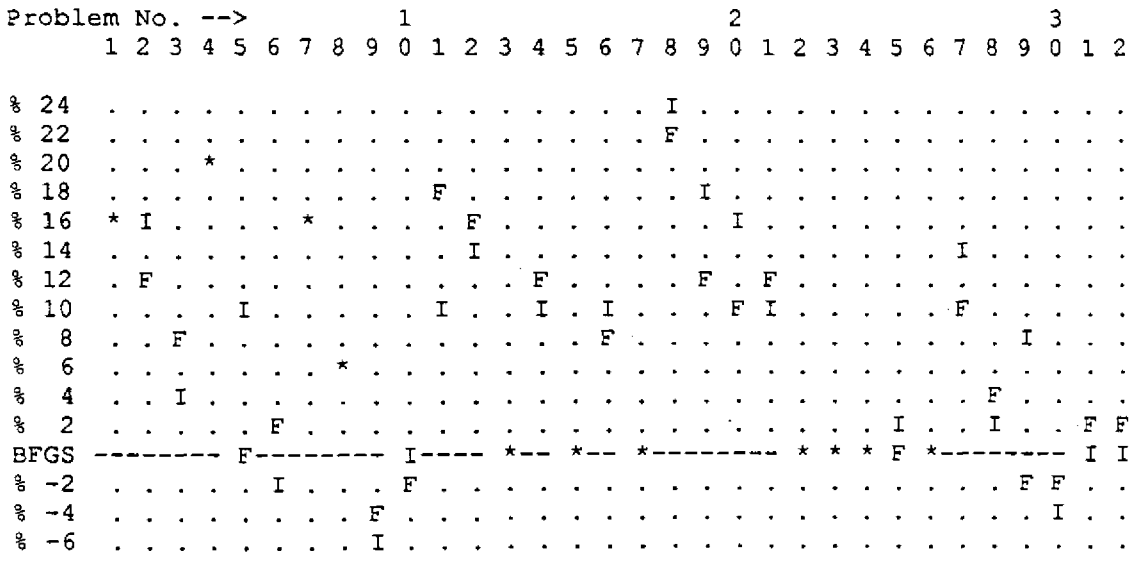
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	0	2
The SAME as	BFGS	91	89
WORSE than	BFGS	9	9

Because of the poor performance of this algorithm and the large size of the diagram, we have chosen to omit the diagram in this case.

ALGORITHM Jc1c (COMPARISON WITH BFGS)

	BFGS	Jc1c
GRAND SUM	(8762, 12384)	(8058, 11387)

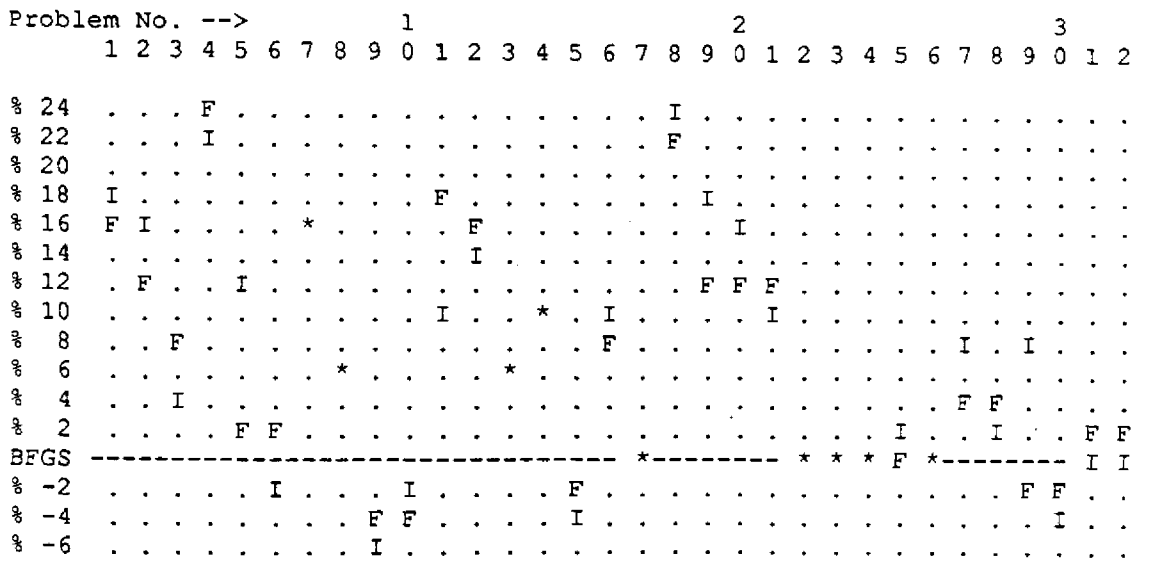
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	65	65
The SAME as	BFGS	20	20
WORSE than	BFGS	15	15



ALGORITHM Jclgr (COMPARISON WITH BFGS)

	BFGS	Jclgr
GRAND SUM	(8762, 12384)	(8041, 11382)

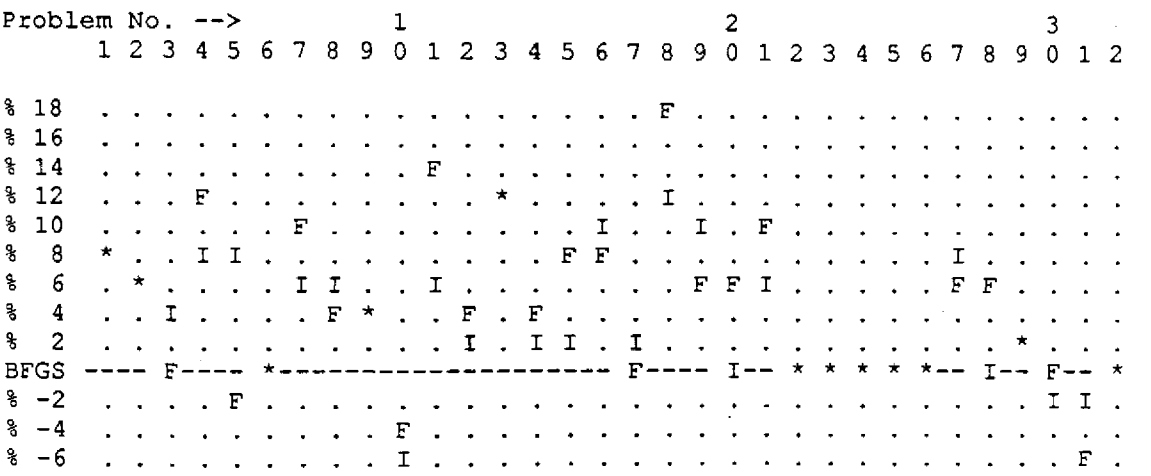
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	65	63
The SAME as	BFGS	19	21
WORSE than	BFGS	16	16



ALGORITHM Jc1q (COMPARISON WITH BFGS)

	BFGS	Jc1q
GRAND SUM	(8762, 12384)	(8399, 11690)

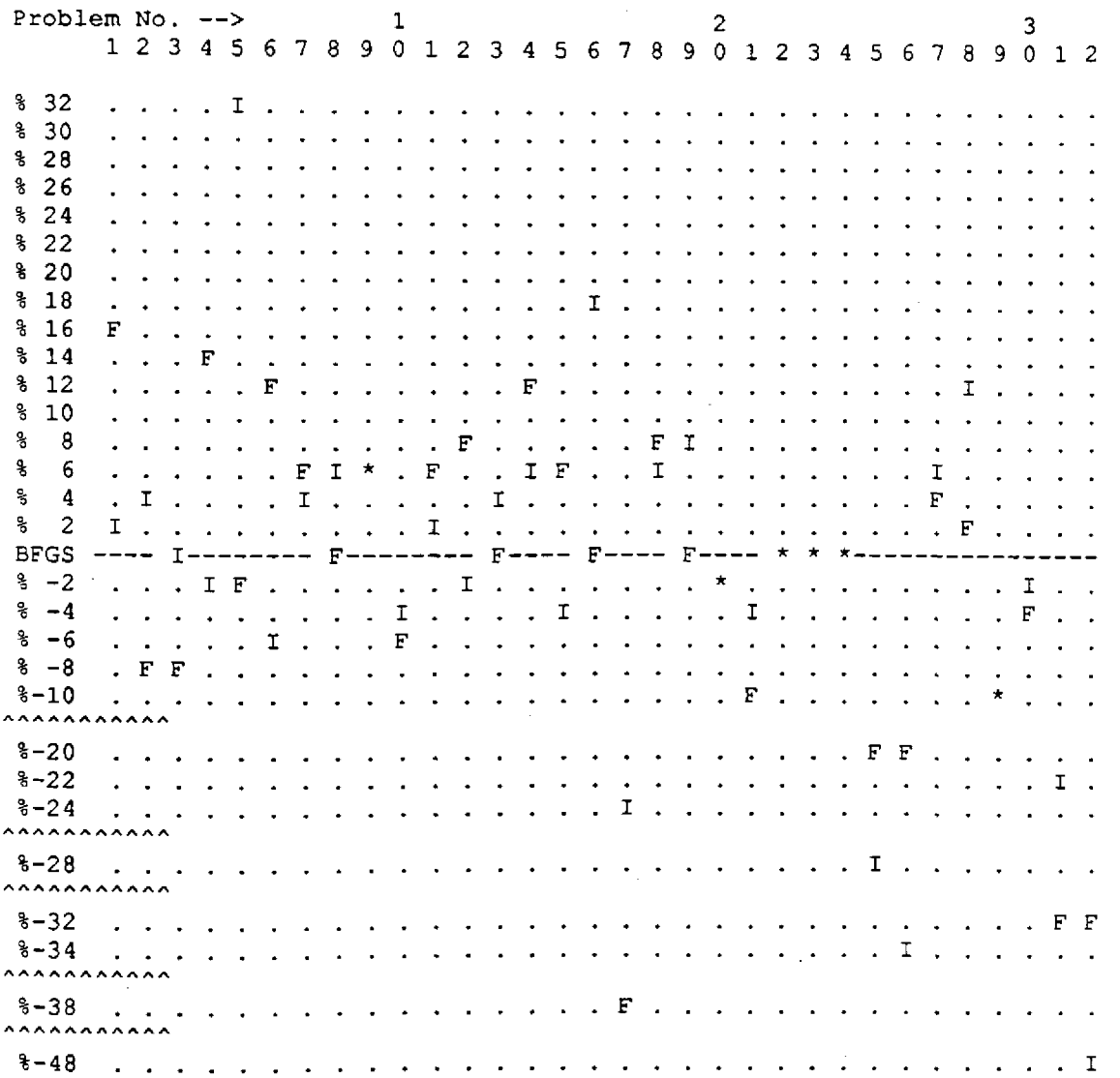
		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	59	57
The SAME as	BFGS	19	20
WORSE than	BFGS	22	23



ALGORITHM FS-B2 (COMPARISON WITH BFGS) (Ford and Saadallah (1987))

	BFGS	FS-B2
GRAND SUM	(8762, 12384)	(8831, 12417)

		FUNCTION EVALUATION COUNT	ITERATION COUNT
BETTER than	BFGS	46	51
The SAME as	BFGS	45	42
WORSE than	BFGS	9	7



Before starting to consider the diagrams and their associated statistics, we recall that the estimate $s^T Bs$ for the curvature at x (or, equivalently, employing c0t from (4-66d)) was first used by Ford and Saadallah(1986) and Saadallah(1988) in a number of algorithms including (what we denote here by) FS-B2 which was their best. We have utilized curvature estimates at x and x^* via polynomial approximation (that is, c0q, c0c or c0gr from (4-66) for the case of curvature at x, and employing c1q, c1c or c1gr from (4-67), for the case of curvature at x^*). We have utilized c0t in our algorithms in order to compare its efficiency (or otherwise) with that of c0c, c0q and c0gr.

From the total set of results, we draw the following general conclusions:

(i) algorithms which employ curvature estimate via polynomial approximation produce better results than those algorithms which utilize c0t. This indicates that curvature estimation via polynomial approximation is more accurate than curvature estimation via the Hessian approximation. We may justify this conclusion as follows:

(1) B (the Hessian approximation matrix) is (usually) initialized to the unit matrix (or some scalar multiple thereof) and then accumulates information about the function's behaviour as it is updated at each iteration. Thus, it is likely to be only a crude approximation in the early iterations and becomes more accurate and reliable as the solution is approached. Therefore Ford and Saadallah recommended that c0t should not be used in the early

iterations and, thus, an algorithm (such as FS-B2) relying on such a curvature estimate will only commence to part company with the parent quasi-Newton method when sufficient iterations have been completed.

(2) On the other hand, curvature estimation via polynomial approximation relies on function values and can be used from the first iteration.

(ii) Both the model employed and the technique for obtaining the parameter(s) are influential in determining the efficiency of algorithms; as is shown by the following observations:

(1) different numerical results for algorithms whose techniques for obtaining the parameter(s) are identical (for example, all algorithms which utilize function values directly), while based upon different models for the gradient;

(2) differing results for algorithms from one model indicate that the technique for obtaining the parameter(s) is influential in determining the efficiency of algorithms.

(iii) In those models from which we have generated algorithms using function values, curvature estimate at x and/or curvature estimate at x^*, the best results (in terms of NF and NI) correspond to algorithms which utilize function values. This is to be expected since, in those algorithms which use function values, a precise item of data (namely, the function value) is utilized, while, in the others, an estimate (of curvature) is used.

(iv) Apart from algorithms Ac0t and Gfc0t, other algorithms which

employ c0t produce results worse than the BFGS method. {In Bc0t, Cc0t and Ec0t, the grand sum of NI is smaller than that of "BFGS", while the grand sum of NF (the main criterion for assessment) is higher than that of the BFGS algorithm, in each case.}

(v) Among algorithms which utilize c0gr, only Cc0gr is not efficient. Excluding Ac0t, Cc0t, Dc0t, Ec0t, Cc0gr and FS-B2, the remainder of algorithms have (on average) about 10% decrease in NF and (on average) about 9% decrease in NI.

7-3-3 Selecting the Best Algorithms

To select the best algorithm from each model, we refer, in each case, to the appropriate table. Every table (A-H and J) contains the following information for the given sets of algorithms and for the complete set of test problems:

(i) summation of NF and NI for each algorithm;

(ii) summation of scores for each algorithm (where the scores are determined with reference to the subset of algorithms currently under consideration in the table).

(iii) equation number (denoted by "EQN.No.") and type of equation (denoted by "TYPE") by which we obtain the parameter(s). ("L" and "NL" stand for "linear" and "nonlinear" equation, respectively.)

Model A

Considering table A, we observe that algorithm Af is the best in the sense it has the smallest NF and NI and, also, the largest score among all the algorithms of this model. The next best algorithm is probably Ac1gr. Bearing in mind that Ac1gr requires the solution of a simpler equation in comparison with equation corresponding to Af, one might possibly prefer Ac1gr to Af when the function is not complicated.

Table A

	BFGS	Af	Ac0gr	Ac0t	Ac1gr
GRAND (NI:)	8762,	8212,	8310,	8691,	8339,
SUMS (NF:)	12384	11256	11561	11969	11525
SCORES	16	51	24	22	33
TYPE		NL	NL	NL	NL
EQN.No.		(5-12)	(5-18)	(5-18)	(5-21)

Model B

Algorithm Bc-01 is selected as the best algorithm of this model, for the following reasons:

(i) Bc-01 has the highest score and its corresponding equation (to obtain the parameter) is linear.

(ii) Although Bf has the smallest NF and NI, its related equation is nonlinear and these values are only slightly less than those for Bc-01.

(iii) Bc0gr (which has the highest score after Bc-01) also involves a nonlinear equation and its NF and NI are (again) only slightly less than those of Bc-01.

However, (as with model A) one may prefer Bf in place of Bc-01 when F is complicated.

Table B

	BFGS	**Bf**	**Bc0gr**	**Bc0t**	**Bc/01**	**Bc-01**
GRAND (NI:)	8762,	7888,	8028,	8575,	8180,	8108,
SUMS (NF:)	12384	11110	11220	12876	11439	11327
SCORES	19	35	41	8	31	42
TYPE		NL	NL	NL	L	L
EQN.No.		(5-29)	(5-34)	(5-34)	(5-38)	(5-39)

Model C

In this model (viewing table C), Cf is clearly (on all numerical counts) the best algorithm, even though its associated equation is nonlinear.

Table C

	BFGS	Cf	Cc0gr	Cc0t
GRAND (NI:)	8762,	7960,	8908,	8406,
SUMS (NF:)	12384	11210	12793	13135
SCORES	29	73	17	16
TYPE		NL	L	L
EQN.No.		(5-45)	(5-46)	(5-46)

Model D

Examining table D, we may conclude that algorithm Df is better (on all numerical counts) than the others. The next best method is Dc0c and this has the added advantage of having a linear equation to solve (as opposed to Df, which requires the solution of a nonlinear equation). We have, however, decided to select Df from this group.

Table D

	BFGS	Df	Dc0c	Dc0q	Dc0t
GRAND (NI:)	8762,	7943,	8197,	8322,	8728,
SUMS (NF:)	12384	11132	11401	11592	15261
SCORES	20	61	35	29	8
TYPE		NL	L	L	L
EQN.No.		(5-48)	(5-51)	(5-51)	(5-51)

Model E

Ec0gr may be viewed as the best algorithm of this model, for these reasons:

(i) Algorithm Ef has only a slightly higher score, and NF and NI for algorithm Ec0gr are only fractionally higher than those of Ef.

(ii) The equation by which we obtain the parameter in Ef is nonlinear, while that in Ec0gr is linear.

Again, for complicated functions, Ef might be more efficient than Ec0gr.

Table E

	BFGS	Ef	Ec0gr	Ec0t	Ec/01	EcLS
GRAND (NI:)	8762,	7971,	8029,	6875,	8224,	8065,
SUMS (NF:)	12384	11057	11090	15311	11471	11445
SCORES	20	48	40	9	26	33
TYPE		NL	L	L	L	L
EQN.No.		(6-13)	(6-15)	(6-15)	(6-16)	(6-17)

Model F

Since both Ff and FLSfc1 have linear equations (to obtain the parameter), FLSfc1 is evidently the better algorithm from this model.

Table F

	BFGS	Ff	FLSfc1
GRAND (NI:)	8762,	8090,	7969,
SUMS (NF:)	12384	11266	11108
SCORES	23	48	66
TYPE		L	L
EQN.No.		(6-23)	(6-26)

Model G

For this model, the algorithm Gfc0c (which is equivalent to the algorithm Jc1c (and the algorithm derived by Biggs(1973))), is taken to be the better algorithm, on the basis of the score and the NF value.

Table G

	BFGS	Gfc0c	Gfc0t
GRAND (NI:)	8762,	8058,	8050,
SUMS (NF:)	12384	11387	11471
SCORES	25	70	37
TYPE		L	L
EQN.No.		(6-33/34)	(6-33/34)

Model H

In this model we have three algorithms for group a (m = 1 and n = 2) and two algorithms for group b (m = 1/2 and n = 3/2). We have excluded Hfc0t.b from the table because its results are very similar to those for Hfc0t.a and the results of Hfc0t.a are much worse than those of the BFGS algorithm. Apart from Hfc0t.a, the other results are very similar. Since the equations corresponding to all algorithms of this model are linear and Hfc0c.a has the highest score and lowest NF value, Hfc0c.a is selected as the best algorithm from this model. We recognize, however, that Hfc0q.a (or even Hfc0c.b or Hfc0q.b) could well have been chosen.

Table H

	BFGS	**Hfc0c.a**	**Hfc0c.b**	**Hfc0q.a**	**Hfc0q.b**	**Hfc0t.a**
GRAND (NI:)	8762,	7984,	8026,	7937,	7983,	16256,
SUMS (NF:)	12384	11350	11360	11361	11453	18753
SCORES	24	39	29	37	31	7
TYPE		L	L	L	L	L
EQN. No.		(6-44)	(6-44)	(6-44)	(6-44)	(6-44)

Model J

Considering table J, we observe that algorithm Jc1gr possesses the highest score and the lowest NF and NI values. Since we do not need to solve an equation for any of the three algorithms, Jc1gr is therefore selected as the best algorithm for this group. Again, we have an algorithm (Jc1c) which is very close, in its performance, to the method selected.

Table J

	BFGS	Jc1q	Jc1c	Jc1gr
GRAND (NI:)	8762,	8399,	8058,	8041,
SUMS (NF:)	12384	11690	11387	11382
SCORES	26	31	59	61
TYPE		L	L	L
EQN.No.		(5-22)	(5-22)	(5-22)

7-3-4 The Best Algorithms Overall

In the previous subsection, we discussed the determination of the best algorithm for each model. It is our intention in this subsection to compare and discuss (in more detail) the numerical results for those selected algorithms, together with the BFGS and the FS-B2 algorithms, and to select the best algorithms overall. We present full numerical results for these algorithms in table 1 (below) which contains the following information:

(i) NF and NI for every test function and every starting-point;

(ii) summation of NF and NI for each test function;

(iii) summation of NF and NI for each algorithm over all test functions;

(iv) summation of the scores for each algorithm overall the test functions.

(v) type of equation (denoted by "TYPE") by which we obtain the parameter(s). ("L" and "NL" stands for "linear" and "nonlinear" equation, respectively.)

TABLE 1

	BFGS	Gfc0c	FS-B2	Af	Cf	Df	Bc_01	Ec0gr	FLSfc1	Hfc0c.a	Jc1gr
PN:	1- Rosenbrock's function (n = 2); gtol = 1.0D-6										
(a)	32, 42	28, 32	34, 39	30, 35	30, 36	30, 35	29, 33	30, 37	31, 37	28, 32	28, 32
(b)	379,520	316,429	366,423	348,432	302,405	309,415	319,428	326,436	313,414	318,447	311,436
(c)	49, 61	46, 58	54, 59	43, 53	43, 57	43, 53	43, 56	41, 53	42, 54	44, 56	44, 54
(d)	62, 81	48, 68	55, 69	50, 69	48, 67	44, 61	47, 67	48, 66	47, 69	48, 68	48, 68
Sum	522,704	438,587	509,590	471,589	423,565	426,564	438,584	445,592	433,574	438,603	431,590
PN:	2- Extended Rosenbrock's function (n = 10); gtol = 1.0D-6										
(a)	26, 37	30, 41	33, 44	28, 38	29, 40	31, 41	30, 41	29, 39	31, 41	30, 41	30, 41
(b)	137,201	105,165	125,207	123,172	114,173	110,171	111,166	113,181	115,178	115,170	105,165
(c)	51, 65	42, 52	50, 67	46, 56	44, 55	43, 57	44, 55	45, 58	42, 54	42, 52	42, 52
(d)	34, 36	30, 37	32, 46	32, 34	33, 36	31, 33	32, 33	33, 35	31, 35	30, 37	30, 37
Sum	248,339	207,295	240,364	229,300	220,304	215,302	217,295	220,313	219,308	217,300	207,295
PN:	3- Extended Rosenbrock's function (n = 20); gtol = 1.0D-6										
(a)	23, 40	27, 44	28, 42	29, 47	28, 47	24, 43	28, 46	25, 42	27, 44	26, 43	26, 43
(b)	144,245	131,212	130,252	129,226	117,221	122,217	117,217	127,214	127,212	132,214	131,212
(c)	51, 63	50, 62	50, 61	52, 63	51, 59	48, 58	50, 61	48, 56	46, 57	51, 62	50, 62
(d)	26, 33	27, 34	34, 53	26, 33	27, 34	27, 34	27, 34	26, 33	26, 33	27, 34	27, 34
Sum	244,381	235,352	242,408	236,369	223,361	221,352	222,358	226,345	226,346	236,353	234,351

TABLE 1 (Continued)

	BFGS	Gfc0c	FS-B2	Af	Cf	Df	Bc_01	Ec0gr	FLSfc1	Hfc0c.a	Jc1gr
PN: 4- Rosenbrock cubic function (n = 2); gtol = 1.0D-6											
(a)	41, 49	34, 39	39, 45	35, 39	34, 40	27, 29	33, 39	28, 38	28, 36	27, 33	32, 37
(b)	418,569	342,476	438,494	373,454	340,455	341,449	345,452	351,438	329,440	333,461	329,455
(c)	98,133	77, 90	88,104	75,100	72,103	72, 95	73, 94	78, 96	75, 95	76, 97	73, 89
(d)	199,276	159,210	201,238	170,214	157,213	155,206	160,220	166,204	151,197	159,203	155,201
Sum	756,1027	612,815	766,881	653,807	603,811	595,779	611,805	623,776	583,768	595,794	589,782
PN: 5- Rosenbrock's cliff function (n = 2); gtol = 1.0D-6											
(a)	21, 30	19, 28	12, 30	17, 28	18, 28	17, 28	18, 28	18, 29	19, 28	19, 28	19, 28
(b)	21, 25	14, 25	16, 25	18, 24	20, 24	18, 24	18, 24	19, 23	20, 24	14, 25	14, 25
(c)	18, 26	17, 26	12, 27	17, 26	17, 26	17, 26	17, 26	18, 26	16, 26	17, 26	17, 26
(d)	19, 25	21, 26	14, 26	18, 24	19, 25	17, 24	19, 24	21, 26	18, 24	19, 24	19, 24
Sum	79,106	71,105	54,108	70,102	74,103	69,102	72,102	76,104	73,102	69,103	69,103
PN: 6- Brown's badly scaled function (n = 2); gtol = 1.0D-6											
(a)	14, 50	14, 48	17, 44	16, 50	16, 49	18, 51	18, 51	13, 47	17, 50	14, 48	14, 48
(b)	32, 65	31, 56	25, 53	21, 56	34, 65	29, 66	24, 51	21, 50	25, 51	31, 56	31, 56
(c)	14, 51	17, 59	17, 46	18, 51	21, 59	20, 55	16, 52	21, 55	20, 54	17, 59	17, 59
(d)	14, 51	13, 48	19, 47	19, 53	17, 53	19, 53	19, 53	15, 50	15, 50	13, 48	13, 48
Sum	74,217	75,211	78,190	74,210	88,226	86,225	77,207	70,202	77,205	75,211	75,211

TABLE 1 (Continued)

	BFGS	Gfc0c	FS-B2	Af	Cf	Df	Bc_01	Ec0gr	FLSfc1	Hfc0c.a	Jc1gr
PN: 7- Powell's badly scaled function (n = 2); gtol = 1.0D-6											
(a)	147,189	126,158	152,172	138,164	127,157	137,168	130,161	132,156	131,164	128,165	126,158
(b)	146,182	117,158	136,171	122,166	114,159	111,158	121,158	122,159	129,154	112,156	118,159
(c)	176,218	147,180	164,209	156,194	144,179	148,178	149,183	152,189	154,183	145,172	147,180
(d)	160,202	134,162	150,192	145,171	130,160	137,168	143,173	131,157	136,169	137,166	134,162
Sum	629,791	524,658	602,744	561,695	515,655	533,672	543,675	537,661	550,670	522,659	525,659
PN: 8- Powell's quartic function (n = 4); gtol = 1.0D-6											
(a)	39, 42	30, 34	38, 41	36, 39	30, 34	33, 36	34, 38	39, 42	41, 45	30, 34	30, 34
(b)	39, 40	48, 49	38, 44	45, 48	39, 40	40, 43	43, 44	37, 38	50, 51	48, 49	48, 49
(c)	53, 57	42, 44	41, 45	37, 41	43, 45	42, 44	41, 44	43, 45	43, 45	42, 44	42, 44
(d)	42, 49	42, 49	47, 59	55, 60	39, 46	51, 57	46, 53	51, 57	49, 55	42, 49	42, 49
Sum	173,188	162,176	164,189	173,188	151,165	166,180	164,179	170,182	183,196	162,176	162,176
PN: 9- Extended Powell's quartic function (n = 12); gtol = 1.0D-6											
(a)	43, 50	46, 52	38, 46	38, 46	38, 45	41, 48	40, 45	37, 42	35, 44	46, 52	46, 52
(b)	62, 68	58, 65	57, 63	58, 64	57, 63	58, 64	58, 64	58, 64	57, 63	58, 65	58, 65
(c)	54, 61	65, 72	53, 59	54, 63	55, 60	64, 70	58, 63	55, 62	53, 58	65, 72	65, 72
(d)	70, 82	75, 84	65, 78	79, 89	70, 79	81, 90	76, 85	73, 84	67, 77	75, 84	75, 84
Sum	229,261	244,273	213,246	229,262	220,247	244,272	232,257	223,252	212,242	244,273	244,273

TABLE 1 (Continued)

	BFGS	Gfc0c	FS-B2	Af	Cf	Df	Bc_01	Ec0gr	FLSfc1	Hfc0c.a	Jc1gr
PN: 10- Extended Powell's quartic function (n = 20); gtol = 1.0D-6											
(a)	43, 57	35, 58	33, 61	28, 48	36, 55	34, 52	32, 53	31, 48	39, 56	41, 61	36, 57
(b)	37, 54	34, 49	54, 69	48, 61	48, 62	51, 65	44, 58	38, 51	36, 49	35, 49	38, 51
(c)	46, 57	52, 62	45, 54	49, 60	54, 66	46, 58	49, 63	48, 60	48, 61	52, 62	52, 62
(d)	79, 91	82, 97	80, 93	81, 94	87,102	85, 99	76, 91	81, 94	87,100	82, 97	82, 97
Sum	205,259	203,266	212,277	206,263	225,285	216,274	201,265	198,253	210,266	210,269	208,267
PN: 11- Box's "difficult" exponential function (n = 3); gtol = 1.0D-6											
(a)	25, 30	23, 26	23, 32	26, 29	24, 30	25, 27	23, 27	24, 26	24, 27	23, 26	23, 26
(b)	38, 54	35, 43	36, 59	34, 44	38, 43	37, 42	38, 43	43, 48	35, 40	35, 43	35, 43
(c)	54, 96	44, 76	58, 83	49, 76	44, 73	48, 76	48, 78	47, 76	48, 78	47, 78	46, 77
(d)	90,118	83,101	87,108	83,100	83, 97	80, 97	79, 93	83, 96	71, 90	81, 97	83,101
Sum	207,298	185,246	204,282	192,249	189,243	190,242	188,241	197,246	178,235	186,244	187,247
PN: 12- Bigg's EXP4 function (n = 4); gtol = 1.0D-8											
(a)	33, 36	29, 32	35, 39	30, 33	29, 33	29, 33	29, 33	30, 34	28, 31	29, 32	29, 32
(b)	31, 35	29, 31	32, 34	29, 32	29, 31	28, 30	29, 31	31, 33	31, 33	29, 31	29, 31
(c)	149,221	130,191	165,206	149,204	137,202	134,196	130,188	141,200	134,202	137,202	130,191
(d)	256,337	217,271	249,301	231,275	208,263	214,271	220,281	230,273	211,272	228,298	213,271
Sum	469,629	405,525	481,580	439,544	403,529	405,530	408,533	432,540	404,538	423,563	401,525

TABLE 1 (Continued)

	BFGS	Gfc0c	FS-B2	Af	Cf	Df	Bc_01	Ec0gr	FLSfc1	Hfc0c.a	Jc1gr
PN: 13- Bigg's EXP6 function (n = 6); gtol = 1.0D-8											
(a)	54, 69	54, 68	52, 74	54, 66	55, 69	55, 69	55, 69	56, 67	54, 68	54, 68	54, 68
(b)	110,125	108,118	113,133	104,110	108,119	105,117	99,106	98,106	91,102	99,110	95,105
(c)	137,157	147,176	126,144	71, 75	155,178	83, 90	77, 82	86, 93	78, 83	97,112	130,148
(d)	89, 94	80, 86	81, 95	83, 87	81, 88	83, 89	80, 84	79, 83	82, 87	78, 87	84, 96
Sum	390,445	389,448	372,446	312,338	399,454	326,365	311,341	319,349	305,340	328,377	363,417
PN: 14- Wood's function (n = 4); gtol = 1.0D-6											
(a)	78, 97	69, 83	69, 85	75, 93	78, 91	76, 90	81, 99	74, 88	71, 85	69, 82	69, 83
(b)	80,101	76, 90	70, 86	75, 92	77, 96	75, 94	74, 96	76, 95	75, 91	77, 92	73, 94
(c)	68, 96	62, 90	67, 81	64, 91	64, 89	59, 87	57, 80	62, 83	66, 89	62, 90	62, 90
(d)	38, 49	32, 41	41, 51	34, 43	33, 44	33, 41	34, 45	33, 41	34, 42	32, 41	32, 41
Sum	264,343	239,304	247,303	248,319	252,320	243,312	246,320	245,307	246,307	240,305	236,308
PN: 15- Thermistor function (n = 3); gtol = 1.0D-6											
(a)	231,309	199,253	246,281	220,258	211,260	208,261	211,258	209,253	199,243	208,263	206,257
(b)	138,182	118,149	139,165	126,151	123,153	123,152	127,156	123,148	121,151	122,149	120,148
(c)	107,134	183,239	124,146	215,252	84,102	185,221	188,232	103,123	116,141	184,250	189,249
(d)	127,162	108,142	119,146	32, 41	106,134	112,132	114,137	113,140	106,126	111,147	107,144
Sum	603,787	608,783	628,738	593,702	524,649	628,766	640,783	548,664	542,661	625,809	622,798

TABLE 1 (Continued)

	BFGS	Gfc0c	FS-B2	Af	Cf	Df	Bc_01	Ec0gr	FLSfc1	Hfc0c.a	Jc1gr
PN: 16- Weibull function (n = 3); gtol = 1.0D-6											
(a)	62, 80	50, 69	51, 77	46, 64	48, 70	48, 66	52, 68	51, 66	49, 66	51, 69	50, 68
(b)	55, 71	52, 65	43, 70	57, 68	51, 67	46, 63	56, 70	52, 63	52, 63	53, 66	52, 65
(c)	32, 47	35, 50	29, 44	32, 47	31, 45	38, 53	35, 51	33, 47	30, 44	35, 50	35, 50
(d)	81,100	72, 90	65,108	70, 91	66, 85	71, 91	69, 84	72, 85	70, 88	69, 90	72, 90
Sum	230,298	209,274	188,299	205,270	196,267	203,273	212,273	208,261	201,261	208,275	209,273
PN: 17- "F55" function (n = 55); gtol = 5.0D-5											
(a)	120,235	122,233	155,354	135,229	120,228	124,227	132,228	130,231	126,232	122,233	122,233
(b)	78,124	77,123	91,141	75,132	82,126	89,133	90,130	85,142	79,126	77,123	77,123
Sum	198,359	199,356	246,495	210,361	202,354	213,360	222,358	215,373	205,358	199,356	199,356
PN: 18- Penalty function I (n = 10); gtol = 1.0D-6											
(a)	77, 94	59, 77	66, 87	70, 76	61, 71	63, 75	61, 72	66, 71	68, 80	62, 82	62, 75
(b)	165,210	124,163	163,192	142,164	127,153	133,166	127,154	130,154	134,166	125,175	124,163
Sum	242,304	183,240	229,279	212,240	188,224	196,241	188,226	196,225	202,246	187,257	186,238

TABLE 1 (Continued)

	BFGS	Gfc0c	FS-B2	Af	Cf	Df	Bc_01	Ec0gr	FLSfc1	Hfc0c.a	Jc1gr
PN: 19- Penalty function I (n = 20); gtol = 1.0D-6											
(a)	89,113	73, 96	78,121	77, 97	76, 99	74, 97	73,100	79,101	75,100	72, 96	72, 97
(b)	161,202	134,181	151,197	145,175	138,182	130,161	129,172	140,175	139,170	140,191	134,181
Sum	250,315	207,277	229,318	222,272	214,281	204,258	202,272	219,276	214,270	212,287	206,278
PN: 20- Penalty function I (n = 30); gtol = 1.0D-6											
(a)	89,141	76,131	88,152	81,130	84,131	81,127	77,129	84,132	83,134	76,132	76,132
(b)	155,232	130,202	161,232	142,199	135,200	135,192	132,195	137,199	134,199	129,198	128,199
Sum	244,373	206,333	249,384	223,329	219,331	216,319	209,324	221,331	217,333	205,330	204,331
PN: 21- Watson function (n = 12); gtol = 1.0D-6											
(a)	40, 76	38, 63	42, 78	40, 65	40, 65	40, 65	40, 65	36, 61	37, 62	38, 63	38, 63
(b)	36, 61	31, 58	37, 74	35, 60	35, 70	40, 68	36, 64	34, 61	34, 61	31, 58	31, 58
Sum	76,137	69,121	79,152	75,125	75,135	80,133	76,129	70,122	71,123	69,121	69,121

TABLE 1 (Continued)

	BFGS	Gfc0c	FS-B2	Af	Cf	Df	Bc_01	Ec0gr	FLSfc1	Hfc0c.a	Jc1gr
PN: 22- Variably Dimensioned function (n = 20); gtol = 1.0D-6											
(a)	10, 43	10, 43	10, 43	10, 43	10, 43	10, 43	10, 43	10, 43	10, 43	10, 43	10, 43
(b)	9, 50	9, 50	9, 50	9, 50	9, 50	9, 50	9, 50	9, 50	9, 50	9, 50	9, 50
Sum	19, 93	19, 93	19, 93	19, 93	19, 93	19, 93	19, 93	19, 93	19, 93	19, 93	19, 93
PN: 23- Variably Dimensioned function (n = 30); gtol = 1.0D-6											
(a)	9, 39	9, 39	9, 39	9, 39	9, 39	9, 39	9, 39	9, 39	9, 39	9, 39	9, 39
(b)	9, 42	9, 42	9, 42	9, 42	9, 42	9, 42	9, 42	9, 42	9, 42	9, 42	9, 42
Sum	18, 81	18, 81	18, 81	18, 81	18, 81	18, 81	18, 81	18, 81	18, 81	18, 81	18, 81
PN: 24- Variably Dimensioned function (n = 40); gtol = 1.0D-6											
(a)	10, 41	10, 41	10, 41	10, 41	10, 41	10, 41	10, 41	10, 41	10, 41	10, 41	10, 41
(b)	10, 59	10, 59	10, 59	10, 59	10, 59	10, 59	10, 59	10, 59	10, 59	10, 59	10, 59
Sum	20,100	20,100	20,100	20,100	20,100	20,100	20,100	20,100	20,100	20,100	20,100

TABLE 1 (Continued)

	BFGS	Gfc0c	FS-B2	Af	Cf	Df	Bc_01	Ec0gr	FLSfc1	Hfc0c.a	Jc1gr
PN: 25- Discrete Boundary Value function (n = 20); gtol = 1.0D-6											
(a)	43,101	43,101	74,137	43,101	43,101	43,101	43,101	43,101	43,101	43,101	43,101
(b)	68, 77	65, 75	71, 83	65, 75	65, 75	65, 75	65, 75	65, 75	65, 75	65, 75	65, 75
(c)	67, 85	66, 84	75, 93	66, 84	66, 84	66, 84	66, 84	66, 84	66, 84	66, 84	66, 84
(d)	49, 86	49, 86	69,108	49, 86	49, 86	49, 86	49, 86	49, 86	49, 86	49, 86	49, 86
Sum	227,349	223,346	289,421	223,346	223,346	223,346	223,346	223,346	223,346	223,346	223,346
PN: 26- Discrete Boundary Value function (n = 30); gtol = 1.0D-6											
(a)	71,175	71,175	119,232	71,175	71,175	71,175	71,175	71,175	71,175	71,175	71,175
(b)	98,117	97,116	114,132	97,116	97,116	97,116	97,116	97,116	97,116	97,116	97,116
(c)	93,131	93,131	114,151	93,131	93,131	93,131	93,131	93,131	93,131	93,131	93,131
(d)	74,140	74,140	103,165	74,140	74,140	74,140	74,140	74,140	74,140	74,140	74,140
Sum	336,563	335,562	450,680	335,562	335,562	335,562	335,562	335,562	335,562	335,562	335,562
PN: 27- Penalty function II (n = 20); gtol = 5.0D-5											
(a)	137,201	126,202	124,197	141,186	122,186	120,184	135,204	115,180	120,179	116,183	141,214
(b)	196,319	158,262	188,304	210,316	160,264	124,224	199,317	186,297	192,303	157,267	164,284
Sum	333,520	284,464	312,501	351,502	282,450	244,408	334,521	301,477	312,482	273,450	305,498

TABLE 1 (Continued)

	BFGS	Gfc0c	FS-B2	Af	Cf	Df	Bc_01	Ec0gr	FLSfc1	Hfc0c.a	Jc1gr
PN: 28- Penalty function II (n = 30); gtol = 5.0D-5											
(a)	347,444	344,425	299,436	330,420	333,404	318,395	343,421	325,408	341,418	318,399	333,415
(b)	207,315	195,301	183,313	162,274	195,304	170,281	210,316	213,318	208,311	180,285	212,318
Sum	554,759	539,726	482,749	492,694	528,708	488,676	553,737	538,726	549,729	498,684	545,733
PN: 29- Penalty function II (n = 25); gtol = 1.0D-6											
(a)	50, 80	42, 79	59, 90	44, 74	43, 80	42, 79	43, 78	44, 77	43, 78	42, 79	42, 79
(b)	39, 63	40, 66	39, 66	38, 65	37, 65	40, 65	37, 66	38, 62	41, 61	40, 66	40, 66
Sum	89,143	82,145	98,156	82,139	80,145	82,144	80,144	82,139	84,139	82,145	82,145
PN: 30- Chained Rosenbrock function (n = 32); gtol = 1.0D-6											
(a)	386,460	395,459	382,466	388,455	392,449	386,445	384,451	382,444	405,467	394,457	395,459
(b)	290,386	314,402	306,412	291,379	299,389	294,392	303,396	293,378	294,387	313,403	314,402
Sum	676,846	709,861	688,878	679,834	691,838	680,837	687,847	675,822	699,854	707,860	709,861

TABLE 1 (Continued)

	BFGS	Gfc0c	FS-B2	Af	Cf	Df	Bc_01	Ec0gr	FLSfc1	Hfc0c.a	Jc1gr
PN: 31- Broyden Triangular function (n = 30); gtol = 1.0D-6											
(a)	18, 81	17, 73	22,106	18, 81	18, 81	18, 81	17, 72	18, 80	18, 80	17, 73	17, 73
(b)	22, 96	23,101	27,127	23,100	24,107	22, 93	24,107	23,100	23,101	23,101	23,101
Sum	40,177	40,174	49,233	41,181	42,188	40,174	41,179	41,180	41,181	40,174	40,174
PN: 32- The "VAR" function (n = 45); gtol = 5.0D-5											
(a)	58, 83	58, 83	62, 85	58, 83	58, 83	58, 83	58, 83	58, 83	58, 83	58, 83	58, 83
(b)	60,109	61,107	112,167	61,107	61,107	61,107	61,107	61,107	60,109	61,107	61,107
Sum	118,192	119,190	174,252	119,190	119,190	119,190	119,190	119,190	118,192	119,190	119,190
GRAND SUM	8762, 12384	8058, 11387	8831, 12417	8212, 11256	7960, 11210	7943, 11132	8108, 11327	8029, 11090	7969, 11108	7984, 11350	8041, 11382
SCORES	15	25	13	26	25	30	21	30	30	26	26
TYPE		L	L	NL	NL	NL	L	L	L	L	L

Examining the NF and NI values and the scores at the end of table 1, we suggest the following algorithms (in order) as exhibiting the best numerical performance: (Ec0gr, FLSfc1), Df, Af, Hfc0c.a, Jc1gr, Cf, Gfc0c and Bc-01. The first three have equal highest scores and the last six have the next highest scores. The grand sums for NF and NI in these nine algorithms are smaller than those for the remainder. Consideration of the nine algorithms in subsection 7-3-2 shows that, on average, these algorithms are less efficient than the BFGS method (typically) only for 2-5 test problems out of a total of 32 as far as NF is concerned..

Considering the first three algorithms, the NF and NI sums for Df are very close to those of Ec0gr and FLSfc1. The equation by which we obtain the parameter is nonlinear for Df, while it is linear for the two others. On these grounds, we select Ec0gr and FLSfc1 as the best algorithms overall.

7-3-5 Numerical Study of Convergence

In this section we intend to consider (numerically) the convergence behaviour of seven algorithms (including the BFGS and the best six new algorithms). To judge whether an algorithm has, locally, q-superlinear convergence behaviour, t^* must be equal to unity for each step under consideration (Broyden, Dennis and More(1973)). Therefore, we investigate data (for three functions) from the final j ($\leq$ 10) iterations, for which the values of t^*

are equal to unity. The data presented in the body of the tables below are:

k : iteration number, and

$$\omega_1 = \frac{\|x_k - z\|}{\|x_{k-1} - z\|},$$

respectively. The numerical results for the six top algorithms indicate that ω_1 generally decreases as the solution is approached (although not always monotonically), as it does for BFGS. This signifies that the new algorithms may well have, locally, q-superlinear convergence behaviour, similar to the BFGS algorithm.

We have also considered the issue of n-step q-quadratic convergence for the seven algorithms. Since the evidence was inconclusive we have not included the corresponding results.

Algorithm: BFGS

Rosenbrock's function	Biggs's EXP4	Penalty function I
	24, 4.7591D-001	
	25, 3.8631D-001	
25, 6.6548D-001	26, 2.0321D-001	
26, 4.5856D-001	27, 1.3648D-001	
27, 1.8261D-001	28, 6.5402D-001	
28, 6.6855D-001	29, 3.4164D-001	
29, 2.6916D-001	30, 2.5410D-002	74, 5.9025D-001
30, 2.7750D-002	31, 3.6088D-001	75, 4.2897D-001
31, 7.7660D-002	32, 2.8402D-001	76, 3.5089D-001
32, 6.7770D-002	33, 3.7010D-002	77, 2.6602D-001

Algorithm: Af

Rosenbrock's function	Biggs's EXP4	Penalty function I
	21, 1.3833D+000	
	22, 9.7938D-001	
	23, 7.0594D-001	
24, 7.8205D-001	24, 3.1092D-001	
25, 4.2387D-001	25, 1.6915D-001	
26, 1.3206D-001	26, 4.8742D-001	66, 7.2890D-001
27, 6.8598D-001	27, 8.5200D-002	67, 3.2634D-001
28, 7.7410D-002	28, 1.8744D-001	68, 4.9520D-002
29, 8.6400D-002	29, 1.5400D-002	69, 1.1460D+000
30, 8.0000D-004	30, 4.0270D-002	70, 7.8414D-001

Algorithm: Df

Rosenbrock's function	Biggs's EXP4	Penalty function I
21, 4.6386D-001		
22, 7.1426D-001		
23, 5.0200D-001		
24, 4.0180D-002		
25, 1.2633D+000		
26, 2.0312D-001	25, 2.6576D-001	
27, 1.0065D-001	26, 2.2371D-001	60, 6.8501D-001
28, 4.7500D-003	27, 1.2012D-001	61, 2.7791D-001
29, 1.5817D-001	28, 3.4500D-003	62, 7.7350D-002
30, 1.7320D-002	29, 1.3150D-002	63, 1.0039D+000

Algorithm: Ec0gr

Rosenbrock's function	Biggs's EXP4	Penalty function I
21, 8.3994D-001		57, 8.7328D-001
22, 6.5229D-001		58, 6.7481D-001
23, 6.1496D-001	23, 2.6134D-001	59, 8.4336D-001
24, 4.3802D-001	24, 9.1240D-002	60, 6.6991D-001
25, 5.0973D-001	25, 2.2880D-001	61, 5.0263D-001
26, 5.7290D-002	26, 9.2150D-001	62, 7.8612D-001
27, 3.5972D-001	27, 2.3681D-001	63, 5.8266D-001
28, 6.7580D-002	28, 9.6820D-002	64, 2.7271D-001
29, 1.9380D-002	29, 1.3170D-002	65, 4.9835D-001
30, 9.5000D-004	30, 4.1660D-002	66, 1.6448D-001

Algorithm: FLSfc1

Rosenbrock's function	Biggs's EXP4	Penalty function I
22, 4.9564D-001	19, 5.3954D-001	59, 5.1448D-001
23, 6.1454D-001	20, 4.5101D-001	60, 1.0458D+000
24, 2.1196D-001	21, 5.9090D-001	61, 8.5008D-001
25, 4.9728D-001	22, 3.3533D-001	62, 5.8353D-001
26, 6.3952D-001	23, 3.4938D-001	63, 3.4179D-001
27, 3.6885D-001	24, 5.6397D-001	64, 1.0916D+000
28, 2.3740D-002	25, 3.1028D-001	65, 6.6254D-001
29, 3.4430D-002	26, 1.5350D-002	66, 3.7787D-001
30, 1.0740D-001	27, 2.6900D-002	67, 9.0710D-002
31, 1.9390D-002	28, 3.1160D-002	68, 7.1927D-001

Algorithm: Hfc0c.a

Rosenbrock's function	Biggs's EXP4	Penalty function I
	20, 3.0878D-001	53, 8.7946D-001
	21, 5.5828D-001	54, 6.7619D-001
21, 4.7120D-001	22, 2.6948D-001	55, 8.3193D-001
22, 2.8329D-001	23, 4.5012D-001	56, 4.7114D-001
23, 1.2159D+000	24, 8.8192D-001	57, 1.0503D+000
24, 3.7705D-001	25, 2.4126D-001	58, 7.4939D-001
25, 6.3370D-002	26, 4.2150D-002	59, 4.7134D-001
26, 5.6400D-003	27, 1.9710D-002	60, 1.4312D-001
27, 6.0330D-002	28, 8.4030D-002	61, 8.9958D-001
28, 3.3300D-003	29, 8.9370D-002	62, 3.5528D-001

Algorithm: Jc1gr

Rosenbrock's function	Biggs's EXP4	Penalty function I
	20, 3.0878D-001	
	21, 5.5828D-001	
21, 4.7120D-001	22, 2.6948D-001	
22, 2.8329D-001	23, 4.5012D-001	
23, 1.2159D+000	24, 8.8192D-001	57, 7.7290D-001
24, 3.7705D-001	25, 2.4126D-001	58, 3.9005D-001
25, 6.3370D-002	26, 4.2150D-002	59, 5.9956D-001
26, 5.6400D-003	27, 1.9710D-002	60, 2.3310D-001
27, 6.0330D-002	28, 8.4030D-002	61, 1.7200D-001
28, 3.3300D-003	29, 8.9370D-002	62, 7.7407D-001

7-4 Summary and Conclusions

Our aim, in this thesis, has been the improvement of quasi-Newton methods, by the use of objective function values. Quasi-Newton methods are based on the following relation (Broyden(1965)):

$$Bs = \frac{dg}{dt}\Big|_{t=t^*} \text{ or}$$

$$= \frac{dg}{d\tau}\Big|_{\tau=1},$$

where g is regarded be a function of t or τ, respectively (see chapter 2). In standard quasi-Newton formulas, the value of

$$\frac{dg}{d\tau}\Big|_{\tau=1}$$

is estimated by $g^* - g = y$, which may be obtained through a linear model for the gradient. Ford and Saadallah(1986) and Saadallah(1988) have introduced a few nonlinear parametric models for the gradient and have obtained the parameter by using s^TBs as a curvature estimate at x (or equivalently, $c_0 = c0t$ from (4-66d)).

We have suggested a number of nonlinear parametric models (including one of those studied by Ford and Saadallah) and have obtained the parameter(s) using values of the objective functions, either directly (equation (4-14)) or, indirectly, by estimating curvature at x and/or x^* (equations (4-16) and (4-17), respectively). (We have also used s^TBs as a curvature estimate at x, for the sake of comparison.)

Therefore, algorithms introduced in the previous chapters may be categorised as follows:

(a) algorithms which employ function values directly;

(b) algorithms which employ curvature estimates at x and/or x^*, obtained via polynomial approximation (thus using function values indirectly);

(c) algorithms which utilize curvature estimates at x via the Hessian approximation.

We have considered, in section 7-3, the new algorithms from the viewpoints of numerical results and of the type of equation(s) which must be solved to obtain the parameter(s). Briefly speaking, the numerical results indicate that algorithms in group (a) are more efficient than those in group (b) (disregarding the type of equation which must be solved), and algorithms in group (b) are more efficient than those in group (c) (see section 7-3 for the details). Taking into consideration the type of equation which must be solved to obtain the parameter(s), algorithms in group (b), for some models, may be viewed, nevertheless, as better than those in group (a). For example, Ec0gr (which is in group (b)) has been selected as the best algorithm in model E because its numerical results are very close to those of Ef (in group (a)) and its corresponding equation is linear, while that of Ef is nonlinear. Finally, algorithms Ec0gr and FLSfc1 were selected as the best algorithms overall.

7-5 Suggestions for Further Research

In all of the new algorithms, we have taken care to maintain the condition

(7-1) $s^T w > 0$

at all times, which is a necessary and sufficient condition (in the case of updates selected from the convex class, for instance) to have B (the Hessian approximation matrix) positive-definite (which guarantees downhill search-directions). (We will refer to other quasi-Newton methods as the nonconvex class.) When we impose the condition (7-1):

(a) we must (often) restrict the parameter(s) to lie in certain interval(s);

(b) the curvature estimate at x^* (which is $s^T w$) and the actual value of the curvature at x^* may be inconsistent in sign, because $s^T w$ is restricted to be positive, while the actual value of the curvature may be positive , negative or zero.

Following on from (a) and (b), the new algorithms may be more efficient when implemented with updates from the nonconvex class (for example, SR1) than with updates from the convex class, for the following reasons:

(i) imposing the restriction for the parameter in the convex class leads to difficulties when the parameter does not lie in the predefined interval. If we revert to use of a "standard" quasi-Newton method in this case, this means that we are not using a nonlinear gradient model any more. On the other hand, on

utilizing a member of the nonconvex class, we do not have to impose restrictions (in this context) for the parameter(s) and we may then utilize the gradient model in more cases;
(ii) any inconsistency between curvature estimate and the actual value of curvature (at least in sign) may be removed. However, it must be borne in mind that there is, then, no guarantee (for such update formulas belonging to the nonconvex class) that they will produce a descent direction (in the case of utilizing a line-search) or define a quadratic model with a minimum (in the case of utilizing a trust-region technique).

All the new algorithms have been implemented using a line-search technique. When we employ either a line-search technique or a trust-region technique, in each iteration we move from the current point to a new point. It is important to observe that the updating of the matrices does not depend on the technique (for example, line-search or trust-region) used to move to the new point. Therefore we may employ (with the models proposed here) a trust-region technique in place of line-searches, which may lead to different results and would be worthy of investigation. {We also recall, in relation to the issues discussed above that Conn et al(1988) have recently claimed that the SR1 formula (using trust-region techniques) produces better results than the well-known BFGS method.}

We have briefly studied (numerically) the convergence properties of some of the new methods. A more detailed investigation of the convergence properties of the new methods is

clearly necessary.

REFERENCES

Aho, A.V., Hopcroft, J.E. and Ullman, J.D.(1987): "Data Structures and Algorithms", Addison-Wesley, London.

Al-Baali, M.(1985): "Descent property and global convergence of the Fletcher-Reeves method with inexact line search", I.M.A. J. Num. Anal., **5**, 121-124.

Allwright, J.C.(1973): "Improving the conditioning of optimal control problems using simple models", in "Recent Mathematical Developments in Control" (ed. D.J. Bell), Academic Press, London.

Armijo, L.(1966): "Minimization of functions having Lipschitz continuous first-partial derivatives", Pacific J. Math., **16**, 1-3.

Bard, Y.(1968): "On a numerical instability of Davidon-like methods", Maths. Comp., **22**, 665-666.

Beale, E.M.L.(1972): "A derivation of conjugate gradients" in "Numerical methods for nonlinear optimization" (ed. F.A. Lootsma), Academic Press, London.

Biggs, M.C.(1971): "Minimization algorithms making use of non-quadratic properties of the objective function", J. Inst. Math. Appl., **8**, 315-327.

Biggs, M.C.(1973): "A note on minimization algorithms which make use of nonquadratic properties of the objective function". J. Inst. Math. Appl., **12**, 337-338.

Box, M.J.(1966): "A comparison of several current optimization methods, and the use of transformations in

constrained problems", Computer J., **9**, 67-77.

Broyden, C.G.(1965): "A class of methods for solving nonlinear simultaneous equations", Maths. Comp., **19**, 577-593.

Broyden, C.G.(1967).: "Quasi-Newton methods and their application to function minimization", Maths. Comp., **21**, 368-381.

Broyden, C.G.(1970): "The convergence of a class of double-rank minimization algorithms", part I and II, J. Inst. Math. Appl., **6**, 76-90 and 222-231.

Broyden, C.G.(1972): "Quasi-Newton methods" in "Numerical methods for unconstrained optimization" (ed. W. Murray), Academic Press, London.

Broyden, C.G., Dennis,J.E. and More,J.J.(1973): "On the local and superlinear convergence of quasi-Newton methods", J. Inst. Math. Appl., **12**, 223-246.

Buckley, A.(1978a): "Extending the relationship between the conjugate gradient and BFGS algorithms", Math. Prog., **15**, 343-348.

Buckley, A.(1978b): "A combined conjugate gradient quasi-Newton minimization algorithm", Math. Prog., **15**, 200-210.

Buckley, A. and Lenir, A.(1983): "Quasi-Newton-like variable-storage conjugate gradients", Math. Prog., **27**, 155-175.

Bunday, B.D. and Garside, G.R.(1987): "Optimization Methods in Pascal", Edward Arnold, London.

Burden, R.L. and Faires, J. D.(1985): "Numerical Analysis" (third edition), Prindle, Weber & Schmidt, Boston.

Byrd, R.H., Schnabel, R.B. and Shultz G.A.(1988):

"Approximate solution of the trust region problem by minimization over two-dimensional subspaces " Math. Prog., **40**, 247-263.

Cohen, A.(1970): "Rate of convergence for root finding and optimization algorithms", (Ph.D. Dissertation), Univ. of California, Berkeley.

Colville, A.R.(1968): "A comparative study of nonlinear programming codes", I.B.M. Technical Report No. 320-2949.

Conn, A.R., Gould, N.I.M. and Toint, Ph. L.(1986): "Testing a class of methods for solving minimization problems with simple bounds on the variables", Research Report CS-86-45, Univ. of Waterloo.

Conn, A.R., Gould, N.I.M. and Toint, Ph. L.(1988): "Convergence of quasi-Newton matrices generated by the symmetric rank one update", Report No. C.S.S. 208, Computer Science and Systems Division, Harwell Laboratory Oxfordshire OX11 ORA.

Coope, I.D.(1987): "A conjugate direction implementation of the BFGS algorithm with automatic scaling", Report No. **42**, Dept. of Mathematics, Univ. of Canterbury Chrischurch, New zealand.

Crockett, J.B. and Chernoff,H.(1955): "Gradient methods of maximization", Pacific J. Math., **5**, 33-50.

Curry, H.B.(1944): "The method of steepest descent for nonlinear minimization problems", Quart. Appl. Math., **2**, 258-261.

Davidon, W.C.(1959): "Variance metric methods for minimization", A.E.C. Res. and Dev. Report ANL-5990 (revised).

Davidon, W.C.(1968): "Variance algorithm for minimization", Computer J., **10**, 406-410.

Davidon, W.C.(1975): "Optimally conditioned optimization algorithms without line searches", Math. Prog., **9**, 1-30.

Davidon, W.C.(1980): "Conic approximations and collinear scalings for optimizers", SIAM J. Num. Anal., **17**, 268-281.

Dembo, R.S., and Steihaug, T.(1983): "Truncated-Newton Algorithms for large-scale unconstrained optimization", Math. Prog., **26**, 190-212

Dennis, J.E.(1972): "On some methods based on Broyden's secant approximation to the Hessian", in "Numerical methods for non-linear optimization" (ed. F. Lootsma), Academic Press, London.

Dennis, J.E. and Martinez, H.J. and Tapia, R.A.(1989): "Convergence theory for the structured BFGS secant method with an application to nonlinear least-squares", J. Opt. Theo. Appl., **61**, 161-178.

Dennis, J.E. and More, J.J.(1977): "Quasi-Newton Methods, Motivation and Theory", SIAM Review, **19**, 46-89.

Dennis, J.E. and Mei, H.H.W.(1979): "Two new unconstrained optimization algorithms which use function and gradient values", J. Opt. Theo. Appl., **28**, 453-482.

Dennis, J.E. and Schnabel, R.B.(1981): "A new derivation of symmetric positive definite secant updates" in "Nonlinear programming **4**" (ed. O.L. Mangasarian, R.R. Meyer and S.M. Robinson), 167-197.

Dennis, J.E. and Schnabel, R.B.(1983): "Numerical methods for unconstrained optimization and nonlinear equations",

Prentice-Hall, Englewood Cliffs, New Jersey.

Dennis, J.E. and Walker, H.F.(1981): "Convergence theorems for least-change secant update methods", SIAM J. Num. Anal., **18**, 949-987.

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

Dixon, L.C.W., Gomulka, J. and Szego, G.P.(1975): "Towards a global optimization technique" in "Towards global optimization" (ed. L.C.W. Dixon and G.P. Szego), North-Holland Publishing Company, Amsterdam.

Dixon, L.C.W. and Patel, K.D.(1981). "The place of parallel computation in numerical global optimization II; The Multiextremal global optimization problem", Technical Report No. 119, Numerical Optimization Centre, The Hatfield Polytechnic.

Fiacco, A.V. and McCormick, G.P.(1968): "Nonlinear Programming", John Wiley, New York.

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

Fletcher, R.(1980): "Practical methods of optimization", vol. 1 of "Unconstrained optimization", John Wiley and Sons, New York.

Fletcher, R.(1987): "Practical Methods Of Optimization" (second edition), John Wiley and Sons, New York.

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

Fletcher, R. and Reeves, C.M.(1964): "Function minimization by conjugate gradients", Computer J., **7**, 149-154.

Ford, J.A.(1986): "On the use of additional function evaluations in Quasi-Newton methods", Department of Computer Science, Technical Report CSM-82, University of Essex.

Ford, J.A. and Saadallah, A.F.(1986): "A rational function model for unconstrained optimization", Colloquia Mathematica Societatis Janos Bolyai, **50**, Numerical Methods, Miskolc, 539-563.

Ford, J.A. and Saadallah, A.F.(1987): "On the construction of minimization methods of quasi-Newton type", Journal of Computational and Applied Mathematics, **20**, 239-246.

Ford, J.A. and Ghandhari R.A.(1989) "On the use of function-values in unconstrained optimization", Journal of Computational and Applied Mathematics (to appear).

Gill, P.E. and Murray, W.(1979): "Conjugate gradient methods for large-scale nonlinear optimization", Technical Report SOL 79-15, Dept. Operations Research, Stanford Univ.

Gill, P.E. Murray, W. and Wright, M.H.(1981): "Practical Optimization", Academic Press, London.

Goldfarb, D.(1970): "A family of variable metric methods derived by variational means", Maths. Comp., **24**, 23-26.

Goldfeld, S.M., Quandt,R.E., and Trotter, H.F.(1966): "Maximization by quadratic hill-climbing", Econometrica, **34**, 541-551.

Goldstein, A.A.(1967): "Constructive Real Analysis", Harper and Row, New York.

Goldstein, A.A. and Price, J.F.(1967): "An effective algorithm for minimization", Num. Math., **10**, 184-189.

Grandinetti, L.(1979): "Factorization versus nonfactorization in quasi-Newtonian algorithms for differentiable optimization", Dipartimento di Sistemi Report N5, Universita della Calabria.

Greenstadt, J.L.(1970): "Variations of variable-metric methods", Maths. Comp., **24**, 1-22.

Grippo, L., Lampariello, F. and Lucidi, s.(1986): "A nonmonotone Line Search Technique for Newton's Method", SIAM J. on Num. Anal., **23**, 707-716.

Hardy, G.H.(1963): "A course of pure mathematics", Cambridge University Press, Cambridge.

Hestenes, M.R. and Stiefel, E.(1952): "Methods of conjugate gradients for solving linear systems", J. Res. N.B.S., **49**, 409-436.

Huang, H.Y.(1970): "Unified approach to quadratically convergent algorithms for function minimization", J. Opt. Theo. Appl., **5**, 405-423.

Isaacson, E. and Keller, H.B.(1966): "Analysis of numerical methods", John Wiley and Sons, London.

Johnson, L.W. and Riess, R.D.(1982): "Numerical Analysis" (second edition), Addison-Wesley, London.

Levenberg, K.(1944): "A method for the solution of certain nonlinear problems in least squares", Quart. Appl. Math., **2**, 164-168.

Luenberger, D.G.(1984): "Linear And Nonlinear Programming" (second edition), Addison-Wesley, London.

Marquardt, D.W.(1963): "An algorithm for least-squares

estimation of nonlinear parameters", SIAM J. Appl. Maths., **11**, 431-441.

McCormick, G.P. and Pearson, J.D.(1969): "Variable metric methods and unconstrained optimization", in "Optimization" (ed. R. Fletcher), Academic Press, London.

McCormick, G.P. and Ritter, K.(1972): "Methods of conjugate directions versus quasi-Newton methods", Math. Prog., **3**, 101-116.

McGuire, M.F. and Wolfe, P.(1973): "Evaluating a restart procedure for conjugate gradients", Report RC-4382, IBM Research Centre, Yorktown Heights.

Meyer, R.R. and Roth, P.M.(1972): "Modified damped least squares: An algorithm for nonlinear estimation", J. Inst. Math. Appl., **9**, 218-233.

More, J.J.(1982): "Recent developments in algorithms and software for trust-region methods", Technical Memorandum No. **2**, Argonne National Laboratory.

More, J.J., Garbow, B.S. and Hillstrom, K.E.(1981): "Testing unconstrained optimization software", A.C.M. T.O.M.S., **7**, 17-41.

Murtagh, B.A. and Sargent, R.W.H.(1969): "A constrained minimization method with quadratic convergence" in "Optimization" (ed. R. Fletcher), Academic Press, London.

Nazareth, L.(1977): "A conjugate direction algorithm without line-searches", J. Opt. Theo. Appl., **23**, 373-387.

Nazareth, L.(1979): "A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms", SIAM J. Num. Anal., **16**, 794-800.

Nazareth, L. and Nocedal, J.(1978): "A study of conjugate gradient methods", Technical Report SOL 78-29, Dept. Operations Res., Stanford Univ.

Nazareth, L. and Nocedal, J.(1982): "Conjugate direction methods with variable storage", Math. Prog., **23**, 326-340.

Nelder, J.A. and Mead, R.(1965): "A Simplex method for function minimization", Computer J., **7**, 308-313.

Nocedal, J.(1980): "Updating quasi-Newton matrices with limited storage", Maths. Comp., **35**, 773-782.

Oren, S.S.(1973): "Self-scaling variable metric algorithms without line-search for unconstrained minimization", Maths. Comp., **27**, 873-885.

Oren, S.S.(1974): "On the selection of parameters in self-scaling variable metric algorithms", Math. Prog., **7**, 351-367.

Oren, S.S. and Spedicato, E.(1976): "Optimal conditioning of self-scaling variable metric algorithms", Math. Prog., **10**, 70-90.

Ortega, J.M. and Rheinboldt, W.C.(1970): "Iterative Solution Of Nonlinear Equations In Several Variables", Academic Press, London.

Perry, A.(1976): "A modified conjugate gradient algorithm", Discussion paper No. 229, Centre for Mathematical Studies in Economics and Management Science, Northwestern University.

Polak, E.(1971): "Computational methods in optimization: A unified approach", Academic Press, New York.

Powell, M.J.D.(1964): "An efficient method for finding the

minimum of a function of several variables without calculating derivatives", Computer J., **7**, 155-162.

Powell, M.J.D.(1970a): "A hybrid method for nonlinear equations" in "Numerical methods for nonlinear algebraic equations" (ed. P. Rabinowitz), Gordon and Breach, London.

Powell, M.J.D.(1970b): "A Fortran subroutine for solving systems of nonlinear algebraic equations" in "Numerical methods for nonlinear algebraic equations" (ed. P. Rabinowitz), Gordon and Breach, London.

Powell, M.J.D.(1970c): "A new algorithm for unconstrained optimization", in "Nonlinear Programming" (ed. J.B. Rosen, O.L. Mangasarian and K. Ritter), 31-65, Academic Press, New York.

Powell, M.J.D.(1971): "On the convergence of the variable metric algorithm", J. Inst. Math. Appl., **7**, 21-36.

Powell, M.J.D.(1972): "Some properties of the variable metric algorithm", in "Numerical methods for nonlinear optimization" (ed. F. Lootsma), Academic Press, London.

Powell, M.J.D.(1975): "Convergence properties of a class of minimization algorithms" in "Nonlinear Programming **2**" (ed.O.L. Mangasarian, R.R. Meyer and S.M. Robinson), Academic Press, New York.

Powell, M.J.D.(1976): "Some global convergence properties of a variable metric algorithm for minimization without exact line-searches" in "SIAM-AMS Proceedings, vol. IX" (ed. R.W. Cottle and C.E. Lemke), SIAM publications, Philadelphia.

Powell, M.J.D.(1977): "Restart procedures for the conjugate

gradient method", Math. Prog., **12**, 241-254.

Powell, M.J.D.(1984): "Nonconvex minimization calculations and the conjugate gradient method", in "Numerical Analysis" (ed. D.F. Griffiths), Springer, Berlin.

Powell, M.J.D.(1985): "How bad are the BFGS and DFP methods when the objective function is quadratic?", DAMTP Report 85/NA4, Univ. of Cambridge.

Powell, M.J.D.(1986): "Convergence properties of algorithms for nonlinear optimization", SIAM Review, vol. 28, 487-500.

Rosenbrock, H.H.(1960): "An automatic method for finding the greatest or least value of a function", Computer J., **3**, 175-184.

Rosenbrock, H.H. and Storey, C.(1966): "Computational techniques for chemical engineers", Pergamon Press.

Saadallah, A.F.(1988): "A new approach to quasi-Newton methods for minimization", (Ph.D. thesis), University of Essex, Colchester, U.K.

Schnabel, R.B.(1976): "Optimal conditioning in the convex class of rank two updates", Report No. C.S.S. 36, Computer Science and Systems Division, A.E.R.E. Harwell, Didcot, Oxon.

Shanno, D.F.(1970): "Conditioning of quasi-Newton methods for function minimization", Maths. Comp., **24**, 647-656.

Shanno, D.F.(1978): "Conjugate gradient methods with inexact searches", Mathematics of Operations Research, vol. **3**, 244-256.

Shanno, D.F. and Phua, K.H.(1978): "Matrix conditioning and nonlinear optimization", Math. Prog., **14**, 149-160.

Shultz, G.A., Schnabel, R.B. and Byrd, B.H.(1985): "A family

of trust-region based algorithms for unconstrained minimization with strong global convergence properties", SIAM J. Num. Anal., **22**, 47-67.

Silverman, R.A.(1985): "Calculus with Analytic Geometry", Prentice-Hall, London.

Sorensen, D.C.(1977): "Updating the symmetric indefinite factorization with applications in a modified Newton's method", Report ANL-77-49, Argonne National Laboratory.

Sorensen, D.C.(1980): "The Q-superlinear convergence of a collinear scaling algorithm for unconstrained optimization", SIAM J. Num. Anal., **17**, 84-114.

Sorensen, D.C.(1981): "Trust region methods for unconstrained minimization" in "Nonlinear optimization" (ed. M.J.D. Powell), Academic Press, New York.

Sorensen, D.C.(1982): "Newton's method with a model trust region modification", SIAM J. Num. Anal., **19**, 409-426.

Spedicato, E.(1975): "Computational experience with quasi-Newton algorithms for minimization problems of moderately large size", in "Towards global optimization **2**" (ed. L.C.W. Dixon, and G.P. Szego), North-Holland Publishing Company, Oxford.

Spedicato, E.(1978): "Quasi-Newton methods for nonlinear unconstrained minimization: A review", in "Towards global optimization **2**" (ed. L.C.W. Dixon and G.P. Szego), North-Holland Publishing Company, Oxford.

Spendley, W., Hext, G.R. and Himsworth, F.R.(1962): "Sequential application of simplex designs in optimization and

evolutionary operation", Technometrics, **4**, 441-461.

Strang, G.(1980): "Linear algebra and its applications", Academic Press, London.

Toint, Ph. L.(1978): "Some numerical results using a sparse matrix updating formula in unconstrained optimization", Maths. Comp., **32**, 839-851.

Tops-10 Software Notebooks(September 1983): "Tops-10/Tops-20 Common Math. Library; Reference Manual", Digital Equipment Corporation, Marlboro, Massachusetts.

Witte, B.F.W. and Holst, W.R.(1964): Paper submitted at the 1964 Spring Joint Computer Conference, Washington D.C.

Wolfe, P.(1969): "Convergence conditions for ascent methods", SIAM Review, **11**, 226-235.

Wolfe, P.(1971): "Convergence conditions for ascent methods II: Some corrections", SIAM Review, **13**, 185-188.