VARIABLE STORAGE CONJUGATE GRADIENT METHODS

Alexandra LeNir

A Thesis
in
The Department
of
Mathematics

Presented in Partial Fulfillment of the Requirements
for the degree of Master of Science at
Concordia University
Montreal, Quebec, Canada

June 1981

ABSTRACT

VARIABLE STORAGE
CONJUGATE GRADIENT METHODS

ALEXANDRA LENIR

QN methods are well known to have a faster rate of convergence than CG methods but they are impractical to use for problems with large n because of their $O(n^2)$ storage requirements. New modified CG methods have recently been developed, including the variable metric CG method of Buckley and the memoryless QN method of Shanno; these require a moderate amount of storage ($O(n)$) and in general converge faster than the standard CG methods. It is shown here that sometimes these methods are identical. Also, in this thesis a new method for solving problems of high dimensionality is introduced which is a mixed CG and QN method. Its derivation is based on Buckley's idea of a variable storage requirement and on Shanno's idea of modifying a standard CG direction into a QN like form. Numerical results given here demonstrate that in general the convergence of this algorithm improves in direct relation to the amount of storage used. This was not achieved by Buckley in his variable storage algorithm and it is not available in Shanno's method.

ACKNOWLEDGEMENTS

This thesis is dedicated to my family and my friend Paula who were very understanding and supportive during my studies.

I am greatly indebted to my teacher and supervisor Dr. A. Buckley for his help, encouragement and many ideas concerning the topic of this thesis. His guidance in the use of computers was much appreciated.

Special thanks also to Heather Duval for her typing.

TABLE OF CONTENTS

PAGE

CHAPTER I

INTRODUCTION

1.1 Background

A problem which arises in many practical situations is to minimize or maximize a given objective function $f(x)$, where x is a real n-dimensional vector which may be subject to a number of constraints. The unconstrained case represents a significant class of practical problems, firstly because many constrained problems can be easily converted to and solved by the methods of unconstrained optimization and secondly because many optimization problems require the solution of unconstrained subproblems.

We shall therefore be concerned with some methods for solving the unconstrained problem

$$\text{minimize } f(x),\ x \in E^n \tag{1.1}$$

where $f(x)$ is assumed to be at least twice continuously differentiable and where the function and the first derivatives can be evaluated at any point x.

Almost all methods developed to solve (1.1) are iterative, i.e. given an initial point x_0 they generate a sequence of points $x_1, x_2, \ldots$ until some stopping criterion is satisfied. The iterative methods we shall be interested in are first theoretically developed to minimize a convex quadratic function in a finite number of iterations. Then they are extended to solve the general problem (1.1) using the fact that a twice continuously differentiable function can be approximated by a quadratic near the minimum.

There are two basic types of methods which just require the evaluation of gradients. First are the quasi-Newton (QN) methods, also called variable metric methods, whose development was initiated by Davidon [9] in 1959. Since then the theory and application of several variations of these methods have been considered by many authors. We shall be interested in a very important and widely used class of such algorithms, the Broyden β-class, introduced by Broyden [3] in 1967, and especially in the BFGS method which is a member of this class. The second type of method using gradients is the conjugate gradient (CG) method developed by Fletcher and Reeves [12] in 1964. It is based on work done by Hestenes and Stiefel [15] in 1952 for solving linear systems. We shall also be discussing some modifications of the conjugate gradient methods, namely Beale's algorithm and the preconditioned conjugate gradient (PCG) algorithm.

In application to a general function, each type of method has some advantages and disadvantages. Recently it was shown by McCormick and Ritter [18] that in general the quasi-Newton methods converge faster and require fewer functional evaluations than the conjugate gradient methods. This agrees with computational experience with these algorithms. However QN methods require computation and storage of an n x n matrix; hence computer storage of n^2

locations. If the function to be minimized has a large number of variables, this may be difficult or impossible to obtain. On the other hand, the conjugate gradient methods require very little storage, 3n to 7n locations depending on the method used, but their convergence is generally slower.

For this reason new methods for solving problems of high dimensionality have been developed. Some use the fact that for large n the Hessian of the function f is very often sparse. Methods of this type were discussed for example in Jadotte [16]. We are interested in a different approach and particularly we shall be concerned with the variable metric conjugate gradient method, the CGQN algorithm, developed by Buckley [7] and the memoryless quasi-Newton method, the MQN algorithm, developed by Shanno [26]. Both are combinations of the quasi-Newton and conjugate gradient approaches and their aim is to keep the storage requirements reasonable while improving the convergence of the conjugate gradient algorithm.

The purpose of this thesis is to introduce a new algorithm developed to solve the problem (1.1) and to compare the theory and computational efficiency of this method with the MQN, CGQN and QN methods. The MQN algorithm, which will be presented in Chapter 3, will be derived with a different approach than the one used in Shanno's paper [26]. The derivation will be based on a new observation about Beale's method to be discussed throughout

Chapter 2. This approach provides a more unified development of the MQN method. It will clarify a relationship which exists between Beale's method and the CGQN algorithm. We will show that for a special case the CGQN and the MQN methods are identical, a point which had not been previously observed.

Numerical results in Shanno [26] and Buckley [7] show that, in general, the MQN algorithm is superior to the CGQN algorithm when the same amount of storage is considered for both cases. However the idea introduced in the CGQN algorithm of varying the storage requirements according to the size of a problem and the computer storage available to a particular user is an interesting one, and in some cases improves the speed of convergence. For this reason we shall discuss several possible modifications which could improve the convergence of the CGQN method. The effect of scaling will be demonstrated with numerical results in Chapter 4. The numerical comparison with the MQN method will indicate that the major drawback of the CGQN algorithm is its requirement of high accuracy in the line searches.

Taking into account the major advantages and disadvantages of the methods discussed above we have developed a new algorithm for solving problems with a large number of variables. This algorithm, to be introduced in Section 3.3 is in some sense an extension of the MQN method motivated by the variable storage idea of the CGQN method and by Theorem 3.1 due to Buckley [6].

Finally, Chapter 4 contains numerical results for different strategies in the implementation of the new variable storage quasi-Newton (VSQN) algorithm as well as a numerical comparison between the CGQN, MQN and QN algorithms. It shows that with increased storage the VSQN method generally outperforms the MQN method.

1.2 Preliminaries

A standard notation encountered in many publications will be used throughout this thesis. Unless otherwise specified, capital leters denote n x n matrices and lower case letters denote column vectors. Other common symbols are explained below:

1. x_k is the kth approximation to x^*, a local minimum of $f(x)$;
2. g_k is the gradient vector of $f(x)$ at x_k, i.e.
$$g_k = g(x_k) = \nabla f(x_k);$$
3. $y_k = g_k - g_{k-1}$;
4. G is the n x n Hessian matrix of f(x), where the (i,j)th element of G is
$$G_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}, \quad i = 1,2,\ldots,n; \; j = 1,2,\ldots,n;$$
5. H_k is an n x n matrix that is the kth approximation to the inverse hessian G^{-1};
6. $\|y\|$ denotes an arbitrary norm of y;
7. q(x) is a quadratic function given by
$$q(x) = \tfrac{1}{2}x^T Ax + b^T x + c$$
where A is a n x n positive definite matrix.

All algorithms which will be discussed are iterative, i.e. given an initial starting point x_0, they generate a sequence of points x_1, x_2,... that is intended to converge to a local minimum of f(x). The point x_k is determined by

$$x_k = x_{k-1} + \lambda_k d_k$$

where d_k is a direction of search determined according to the particular method being used. Since we are interested only in descent methods, d_k will always be required to be downhill, i.e. the condition

$$d_k^T g_{k-1} < 0$$

must hold. The scalar $\lambda_k > 0$ is chosen to minimize the one dimensional function

$$\phi(\lambda) = f(x_{k-1} + \lambda d_k),$$

and the process of determining the minimum point on a given line is called the line search. If λ_k is an exact local minimum of $\phi(\lambda)$ then

$$\phi'(\lambda_k) = d_k^T g_k = 0$$

and the line search will be referred to as an ELS (Exact Line Search).

For the algorithms to be discussed later, there are certain points with respect to the determination of λ_k which should be noted. The basic line search strategy is given by Fletcher [11].

The search along the first direction d_1 usually begins with $\lambda = 1$, and depending on the algorithm used, the search along subsequent directions d_k, k = 2, 3,... begins either with $\lambda = 1$ (the customary approach for QN methods) or

$$\lambda = \frac{\lambda_{k-1}(d_{k-1}^T g_{k-2})}{d_k^T g_{k-1}} \tag{1.2}$$

where λ_{k-1} is the final λ obtained in the previous iteration. Then the trial point $x = x_{k-1} + \lambda d_k$ and $g = g(x)$ are computed. If at x, $g^T d_k < 0$, the step length λ is doubled and a new x is obtained. If $g^T d_k > 0$ the minimum is bracketted so the new λ is determined as max $(\lambda/10 + \hat{\lambda},\ \lambda^* + \hat{\lambda})$ where $\hat{\lambda}$ is the previous best underestimate of the desired λ (initially $\hat{\lambda} \equiv 0$) and λ^* is the minimum of a quadratic fit (see e.g. Luenberger [17]) to the values $\hat{\lambda}$, λ and $\phi'(\lambda)$. The criterion for termination is again specific to the algorithm being used but it is tested for every point computed during the line search. In the case of a CG method, if the condition for terminating is satisfied without the quadratic interpolation having occurred, the interpolation is still forced. This forced interpolation gives the finite termination property when $f = q$. The final λ is then chosen between the original and interpolated λ, whichever produces the lowest function value.

In the study of algorithms it is important to study the rate at which the sequence $\{x_k\}$ converges to the minimum. The terminology defining the speed of convergence is given in the following definition.

<u>Definition 1</u>. Assume that the sequence $\{x_k\}$ converges to x^* and suppose that for some norm $\|\cdot\|$, there exist scalars α, p and $k_o \geq 0$ such that

$$\| x_{k+1} - x^* \| \leq \alpha \| x_k - x^* \|^p, \; k > k_o \; .$$

(i) If $p = 1$ and $\alpha \varepsilon [0,1)$ then we say that the sequence $\{x_k\}$ converges linearly to x^*.

(ii) If $p = 2$ then the sequence $\{x_k\}$ converges quadratically.

(iii) If $1 < p < 2$ then $\{x_k\}$ converges superlinearly.

Superlinear convergence is most commonly attained in practice and it is usually considered satisfactory; linear convergence in general is too slow and therefore undesirable. Although quadratic convergence is preferable it is difficult to obtain in practice and algorithms with this rate of covergence are rare.

CHAPTER II

CG AND QN METHODS

In this chapter we will introduce the basic minimization algorithms under consideration and make a series of observations which will lead to the development of the specialized methods and the new algorithm presented in Chapter 3. The basic algorithms to be discussed are the conjugate gradient (CG), preconditioned conjugate gradient (PCG) and quasi-Newton (QN) methods. They are all iterative methods and are of the following form (not the customary form for standard CG methods):

Given a starting point x_o, the gradient g_o and a matrix Q_o (usually $Q_o = I$), then for $k = 1,2,\ldots,$ iterate with

$$d_k = -Q_{k-1}\, g_{k-1} \quad , \tag{2.1a}$$

$$x_k = x_{k-1} + \lambda_k d_k \quad . \tag{2.1b}$$

The choice of the matrix Q_{k-1} is determined by the specific algorithm and will be discussed in the subsequent sections of this chapter.

2.1 <u>The Conjugate Gradient and Beale Restart Methods</u>

CG methods were first used to solve the general unconstrained minimization problem by Fletcher and Reeves [12] in 1964. They are frequently used, especially for problems with a large number of variables, since they require only a few vectors of length n to be stored.

In this section we define the concept of conjugate directions and present Beale's derivation [2] of the CG method. Then we introduce the CG algorithm for an arbitrary differentiable function and discuss the fundamental problem of restarts.

Definition 2 Given a symmetric positive definite matrix A, the finite set of vectors $d_1, d_2, \ldots, d_k$ is said to be conjugate if

$$d_i^T A d_j = 0 \quad \text{for all } i \neq j.$$

It is well known that such a set of vectors is also linearly independant (see e.g. Luenberger [17]. The importance of conjugate vectors is given by the Expanding Subspace Theorem:

Theorem 1 (EST) Let d_i, $i = 1,2,\ldots,n$ a set of conjugate vectors in E^n and let B_k be the subspace of E^n spanned by $d_1, d_2, \ldots, d_k$. Then for $x_o \in E^n$ the sequence $\{x_k\}$ generated according to

$$x_k = x_{k-1} + \lambda_k d_k$$

has the property that x_k minimizes the quadratic function q on the linear variety $x_o + B_k$, provided all line searches are exact.

Proof See Luenberger [17].

The EST implies that, since x_n minimizes q over $x_o + B_n$, the global minimum x^* of q will be found in at most n steps. This finite termination property is an important consideration when constructing minimization algorithms. The method of conjugate gradients, which we shall now derive, is based on a successive construction of conjugate search directions.

Suppose for now that we know the matrix A defining q. Then we can construct a set of conjugate directions $d_1, d_2, \ldots, d_n$ from an arbitrary set of linearly independent directions $r_1, r_2, \ldots, r_n$ by a Gram-Schmidt process in the following way.

Let $d_1 = r_1$. For $i = 2,3,\ldots,n$ determine successively

$$d_i = r_i + \sum_{j=1}^{i-1} c_{ij}\, d_j , \tag{2.2}$$

where the coefficients c_{ij} must be chosen so that

$$d_i^T A\, d_k = 0 \ , \ k=1,\ldots,i-1 . \tag{2.3}$$

From (2.2) we get

$$d_i^T A\, d_k = r_i^T A\, d_k + \sum_{j=1}^{i-1} c_{ij}\, d_j A d_k . \tag{2.4}$$

By the obvious inductive assumption, substitution of (2.3) into (2.4) yields

$$c_{ij} = - \frac{r_i^T A d_j}{d_j^T A d_j} , \ j = 1,2,\ldots,i-1 . \tag{2.5}$$

Now $d_j \neq 0$, since r_j are assumed to be linearly independent. Hence $d_j^T A d_j > 0$ and c_{ij} is well defined.

Suppose we want to minimize q without first evaluating the Hessian A, but suppose that we can compute the gradient g. Since $x_k = x_{k-1} + \lambda_k d_k$ and $g_k = Ax_k + b$,

$$y_k = g_k - g_{k-1} = \lambda_k\, A d_k . \tag{2.6}$$

Let the set $\{r_1, r_2, \ldots, r_n\}$ be chosen as the set $\{d_1, -g_1, -g_2, \ldots, -g_{n-1}\}$ where d_1 is an arbitrary downhill direction and g_i, $i = 1,2,\ldots,n-1$ are determined successively. It will be shown later (see (2.8) and (2.9)) that all elements of the set chosen are linearly independent. Substituting now (2.6) and $r_i = -g_{i-1}$ $(i \geq 2)$ into (2.5), we obtain

$$c_{ij} = \frac{g_{i-1}^T(g_j - g_{j-1})}{d_j^T(g_j - g_{j-1})} = \frac{g_{i-1}^T y_j}{d_j^T y_j} \quad . \qquad (2.7)$$

Applying the EST, we note that after j line searches along conjugate directions we have found the minimum of q in the hyperplane spanned by $d_1, d_2, \ldots, d_j$. Hence g_j must be orthogonal to the hyperplane, i.e.

$$d_i^T g_j = 0 \quad , \quad i = 1,2,\ldots,j \quad . \qquad (2.8)$$

Also, since the directions d_i, $i = 2,3,\ldots,j$ were constructed as a linear combination of g_i, $i = 1,2,\ldots,j-1$, we see that

$$g_i^T g_j = 0 \quad , \quad 1 \leq i < j \quad . \qquad (2.9)$$

This orthogonality relation makes $c_{ij} = 0$ for $j = 2,3,\ldots,i-2$. Finally (2.2) reduces to the following.

$$d_2 = -g_1 + \frac{g_1^T y_1}{d_1^T y_1} d_1 \qquad (2.10a)$$

$$d_{k+1} = -g_k + \frac{g_k^T y_1}{d_1^T y_1} d_1 + \frac{g_k^T y_k}{d_k^T y_k} d_k \quad , \quad k \geq 2 \quad . \qquad (2.10b)$$

This is the Beale conjugate gradient method. In the regular CG method the first direction d_1 is not arbitrary, but is taken as $-g_o$. Then

$$g_i^T g_j = 0 \quad \text{for} \quad 0 \leq i < j \qquad (2.11)$$

and the coefficient

$$\frac{g_k^T y_1}{d_1^T y_1} = 0$$

The regular CG algorithm then becomes:

Given x_o, define $d_1 = -g_o$ and for $k = 1,2,\ldots$ iterate with

$$x_k = x_{k-1} + \lambda_k d_k, \qquad (2.12a)$$

$$\beta_k = \frac{g_k^T y_k}{d_k^T y_k}, \qquad (2.12b)$$

$$d_{k+1} = -g_k + \beta_k d_k . \qquad (2.12c)$$

The formula (2.12b) is called the Hestenes and Stiefel form of β_k. It can also be written as

$$\beta_k = \frac{g_k^T y_k}{g_{k-1}^T g_{k-1}}, \text{ the Polak-Ribière form, or} \qquad (2.13)$$

$$\beta'_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}, \text{ the Fletcher-Reeves form.} \qquad (2.14)$$

Note that if $f = q$ and line searches are exact, the three formulae for β_k are equivalent, since (2.12b) can then be reduced to (2.13) or (2.14). If $f \neq q$ then they are not the same and give different algorithms. In this thesis we will be using the Hestenes and Stiefel form of β_k, because our main interest will be in algorithms with inexact line searches and (2.12b) does not require ELS.

Now we will describe how the CG algorithm is implemented to solve the problem (1.1) with general f. First we derive a stability condition, i.e. a condition which will guarantee that the functional values decrease on every interation by ensuring downhill directions.

The algorithm (2.12) generates downhill directions when applied to an arbitrary function f, provided that the line searches are exact, since

$$d_{k+1}^T g_k = -g_k^T g_k + \beta_k d_k^T g_k = -\|g_k\|^2 < 0. \qquad (2.15)$$

In fact the ELS requirement can be relaxed since it is clear from (2.15) that a downhill direction can be obtained whenever

$$\beta_k \ (d_k^T g_k) < g_k^T g_k \ .$$

In practice this is usually replaced by slightly stronger condition

$$(d_k^T g_k)\ \beta_k < k_2\ (g_k^T g_k)\ , \tag{2.16}$$

where k_2 is some small constant, say 0.2. The condition (2.16) is attainable in practice for any continuous function.

The regular CG algorithm with any of the three choices of β_k has n-step quadratic convergence. However this is not the case if the starting search direction is not a steepest descent step.

Theorem 2 (Crowder and Wolfe [8]). Let $f = q$ and apply the CG algorithm but with $d_1 \neq -g_0$. Then convergence is linear.

The significance of this theorem for the implementation of the CG algorithm for a general f will become apparent if we replace the general function f, which is approximately quadratic in some neighborhood N of the minimum (a consequence of Taylor's theorem), with a hypothetical function $\hat{f}$, which is still smooth but precisely quadratic in N.

Now suppose we apply the CG algorithm to $\hat{f}$ with the starting point x_0 outside the quadratic region N. Then it is clear that when N is entered, the first direction in N is not the steepest descent step and according to the Theorem 2, convergence to the minimum will be only linear. In practice we do not know when the region N is entered. However, we do know that if the algorithm is restarted in N with the direction of steepest descent, convergence will occur in at most n steps. Therefore Fletcher [12] suggested restarting the CG algorithm after each cycle of n steps with the steepest descent direction. This restarting procedure guarantees superlinear convergence for general f (Crowder and Wolfe [8]).

Unfortunately computational results (Powell [23]) show that, at a restart, the discarded direction defined in (2.12c) generally gives a better value for $f(x_{k+1})$ than the steepest descent step. Recall though, that we can start with an arbitrary downhill direction and still guarantee finite convergence for $f = q$, provided the directions are defined as in (2.10). This is the motivation for Beale's restart (BR) method and gives the following algorithm. Suppose a restart is needed when we are at the point x_t and that d_t was downhill (initially $t = 1$). Compute d_{t+1} as in a regular CG step,

$$d_{t+1} = -g_t + \beta_t d_t \quad , \tag{2.17a}$$

then for $k = t+1, t+2, \ldots$ iterate with

$$x_k = x_{k-1} + \lambda_k d_k \quad , \tag{2.17b}$$

$$\beta_k = \frac{g_k^T y_k}{d_k^T y_k} \quad , \qquad \delta_k = \frac{g_k^T y_t}{d_t^T y_t} \quad , \tag{2.17c}$$

$$d_{k+1} = -g_k + \beta_k d_k + \delta_k d_t \, . \tag{2.17d}$$

Unfortunately, a weakness of the BR method is that the directions generated according to (2.17d) are downhill only when $f = q$ and with ELS since then

$$d_{k+1}^T g_k = -g_k^T g_k + \beta_k d_k^T g_k + \delta_k d_t^T g_k = -g_k^T g_k < 0, \ k > t \, . \tag{2.18}$$

The term $d_k^T g_k$ vanishes when ELS are used and $d_t^T g_k = 0$ when $f = q$ as well (see (2.8)). For a general function f even when all line searches are exact the orthogonality relation $d_t^T g_k = 0$ cannot be guaranteed, so the direction (2.17d) need not be downhill. We will show in the derivation of Shanno's MQN method, and for the new method proposed in Chapter 3, how the direction (2.17d) can be modified to overcome this problem.

A second undesirable feature of the BR algorithm when applied to a general f is that the sequence of points $x_{t+1}, x_{t+2}, \ldots$ may converge to a point other than the minimum x^* of f. To see this consider the rank deficient matrix

$$\hat{H}_t = I - \frac{d_t y_t^T}{d_t^T y_t}$$

and note that (2.17a) and (2.17b) can be rearranged to give

$$d_{t+1} = -\hat{H}_t g_t \, , \tag{2.19a}$$

$$d_{k+1} = -\hat{H}_t g_k + \beta_k d_k \ , \quad k > t \ . \tag{2.19b}$$

It can now be proved by induction that, since d_{t+1} is in the column space of $\hat{H}_t$, d_{k+1} for $k > t$ is also in the same space. Because the column space of $\hat{H}_t$ is of dimension n-1, it cannot be guaranteed that the minimum x^* of f lies in this space and so the points $x_{t+1}, x_{t+2}, \ldots$ may converge to some point with non-zero gradient.

To avoid these drawbacks Powell [23] devised a restart criterion for the BR method which is very successful in practice. Since the gradients in Beale's method are mutually orthogonal when $f = q$ and with ELS, Powell suggests restarting when

$$\frac{g_{k-1}^T g_k}{g_{k-1}^T g_{k-1}} > \rho \tag{2.20}$$

where ρ is some constant less than 1, typically 0.2. The condition (2.20) prevents the convergence to a point with a non-zero gradient.

Restarting is also necessary when the direction (2.17d) is not sufficiently downhill. Therefore Powell recommends restarting whenever

$$1.2 \|g_k\|^2 > d_{k+1}^T g_k > -0.8 \|g_k\|^2 \ , \quad k > t \ . \tag{2.21}$$

Finally the BR algorithm should also be restarted when the full cycle of n iterations without a restart is completed, since then the same problems with linear convergence may occur as in the case of the regular CG method.

Computational results (Powell [23]) show that Beale's method with Powell's restarting criterion is superior to the CG algorithm with Fletcher's restarting technique. In Section 3 we will return to the BR method and show how it can be further improved.

To conclude the discussion about the CG and BR algorithms we will verify that they can be written in the form (2.1). This is not how they usually appear in the literature because the other forms already given indicate more clearly that the algorithms (2.12) and (2.17) do not require storage of matrices. But for our further discussion in Chapter 3 the form (2.1) is more suitable. Rearranging the formula (2.12c), we obtain

$$d_{k+1} = -g_k + \frac{d_k y_k^T}{d_k^T y_k} g_k = -\left(I - \frac{d_k y_k^T}{d_k^T y_k}\right) g_k \equiv -Q_k g_k \quad . \tag{2.22}$$

It is clear that the formula (2.17d) for the Beale direction can be rearranged in a similar fashion to have the form (2.1a) with Q_k defined instead as

$$Q_k = \left(I - \frac{d_k y_k^T}{d_k^T y_k} - \frac{d_t y_t^T}{d_t^T y_t}\right) \; .$$

2.2 Preconditioned Conjugate Gradient Method

The method which will now be described is a modification of the CG method. It is known as the preconditioned conjugate gradient (PCG) method and first appeared in a paper by Axelsson [1] in 1974. It was developed with the object of accelerating the convergence of the CG method by a transformation of variables while keeping the basic properties of the method.

Suppose a nonsingular transformation T is given, such that $\hat{x} \equiv Tx$. The PCG method is derived by first assuming an application of the regular CG algorithm in the new $\hat{x}$-coordinates and then transforming the resulting steps back into the original x-coordinates. The standard CG algorithm (2.12) in the new space is:

Given $\hat{x}_o = Tx_o$, define $\hat{d}_1 = -\hat{g}_o$ and for $k = 1,2,\ldots$, iterate with

$$\hat{x}_k = \hat{x}_{k-1} + \hat{\lambda}_k \hat{d}_k \ , \tag{2.23a}$$

$$\beta_k = \frac{\hat{g}_k^T \hat{y}_k}{\hat{d}_k^T \hat{y}_k} \ , \tag{2.23b}$$

$$\hat{d}_{k+1} = -\hat{g}_k + \hat{\beta}_k \hat{d}_k \ , \tag{2.23c}$$

where $\hat{g} = g(\hat{x})$ and $\hat{y}_k = \hat{g}_k - \hat{g}_{k-1}$. Now consider the transformation of each step (2.23) into the x-coordinates. To do this new relationship between the gradients $\hat{g}$ and g is required. This can be easily derived and is given by

$$\hat{g} = T^{-T}g \quad , \quad \hat{y} = T^{-T}y \ . \tag{2.24}$$

The inner product $\hat{d}_k^T\hat{g}_k$ becomes $d_k^T g_k$ since

$$\hat{d}_k = Td_k \ , \tag{2.25}$$

so $\hat{d}_k^T\hat{g}_k = (Td_k)^T\ (T^{-T}g_k) = d_k^T T^T T^{-T} g_k = d_k^T g_k$.

This relationship implies that the ELS are the same in both coordinate systems, i.e. if $\hat{x}_{k-1} = Tx_{k-1}$ and if both use ELS then the same point is reached in either coordinates. Substituting (2.24) and (2.25) into (2.23b) and (2.23c) we obtain

$$\hat{\beta}_k = \frac{\hat{g}_k^T\hat{y}_k}{\hat{d}_k^T\hat{y}_k} = \frac{(T^{-T}g_k)^T(T^{-T}y_k)}{(Td_k)^T(T^{-T}y_k)} = \frac{g_k^T T^{-1}T^{-T}y_k}{d_k^T y_k} \ ,$$

$$d_{k+1} = T^{-1}\hat{d}_{k+1} = T^{-1}(-\hat{g}_k + \hat{\beta}_k d_k) = -T^{-1}T^{-T}g_k + \hat{\beta}_k d_k .$$

Finally, let $H = T^{-1}T^{-T}$, which implies that H is a symmetric positive definite matrix. The PCG algorithm can now be summarized:

Given x_o and a positive definite matrix H, let $d_1 = -Hg_o$ and for $k = 1,2,\ldots$ iterate with

$$x_k = x_{k-1} + \lambda_k d_k \ , \tag{2.26a}$$

$$\hat{\beta}_k = \frac{g_k^T H y_k}{d_k^T y_k} , \tag{2.26b}$$

$$d_{k+1} = -Hg_k + \hat{\beta}_k d_k . \tag{2.26c}$$

The formulae for β_k in the Fletcher-Reeves or Polak-Ribiere form can be transformed in a similar way. Note that with $H = I$, we have the ordinary CG algorithm (2.12). Also the formula for the direction (2.26c) can be rearranged in a way similar to that for the CG direction (2.22) and we again get

$$d_{k+1} = -\left(H - \frac{d_k y_k^T H}{d_k^T y_k}\right) g_k \equiv -Q_k g_k . \tag{2.27}$$

Note that the matrix Q_k is different than the one given in (2.22). Since the PCG algorithm is just the regular CG method in the $\hat{x}$-coordinates, it has the finite termination property. The directions generated according to (2.26c) with ELS are downhill since H is positive definite so

$$d_{k+1}^T g_k = -g_k^T H g_k + \hat{\beta}_k d_k^T g_k = -g_k^T H g_k < 0 .$$

And again, in practice this is replaced by a slightly stronger condition

$$\hat{\beta}_k(d_k^T g_k) < 0.2\ (g_k^T H g_k) \quad . \tag{2.28}$$

Since CG methods are generally used when limited storage is available, one tries to find a matrix H which improves the convergence of the CG method and requires only a small amount of additional storage. If a full matrix H can be stored then quasi-Newton methods should generally be used, because it has been shown that they are computationally superior to the CG methods (McCormick and Ritter [18], Shanno and Phua [27]). In the next chapter we will introduce a specific type of preconditioning matrix H based on a recursively calculated BFGS update matrix.

To see an example of the use of a preconditioner without the requirement of storage of a full matrix H consider a quadratic function $q(x) = \frac{1}{2}\ x^T Dx$, where D is a diagonal positive definite matrix. Applying the regular CG method (2.12) to this function, the minimum is obtained in at most n iterations. But if the PCG algorithm with the metric $H = D^{-1}$ is used, then the minimum $x^* = 0$ is reached from any starting point x_o in one iteration, since for $\lambda = 1$,

$$d_1 = -Hg_o = -D^{-1}g_o \quad , \quad \text{and}$$

$$x_1 = x_o + \lambda d_1 = x_o - \lambda D^{-1} g_o = x_o - x_o = x^* .$$

The diagonal elements of D^{-1} can be computed without computing second derivatives, since $g_o = Dx_o$ and therefore

$$[\text{diag } D^{-1}]^T = \left(\frac{x_{o1}}{g_{o1}} , \frac{x_{o2}}{g_{o2}} , \ldots , \frac{x_{on}}{g_{on}} \right)^T \qquad (2.29)$$

Here, x_{oi} and g_{oi} are the ith elements of the vectors x_o and g_o respectively. This example also suggests a possible technique for scaling the initial direction d_1 when minimizing a general function f.

As a further example of a PCG method we will now show that the Beale restart method is in fact a PCG method with a special preconditioner $\hat{H}_t$. This is a new observation and it will be important in our derivation of the MQN and the new algorithm in Chapter 3. Note that Beale's direction (2.19b) agrees with a PCG direction (2.26c) with $H = \hat{H}_t$ except for the term multiplying d_k. However, consider the expression $g_k^T \hat{H}_t y_k$ when $f = q$ and with ELS:

$$g_k^T \hat{H}_t y_k = g_k^T \left(I - \frac{d_t y_t^T}{d_t^T y_t} \right) y_k$$

$$= g_k^T y_k - \frac{(g_k^T d_t)(y_t^T y_k)}{d_t^T y_t} = g_k^T y_k \qquad (2.30)$$

since by (2.8) $g_k^T d_t = 0$ for $k \geq t$. Substituting (2.30) into (2.19b) we can rewrite the Beale restart algorithm as

$$d_{t+1} = -\hat{H}_t g_t \ , \tag{2.31a}$$

$$d_{k+1} = -\hat{H}_t g_k + \frac{g_k^T \hat{H}_t y_k}{d_k^T y_k} d_k = -\hat{H}_t g_k + \hat{\beta}_k d_k \ . \tag{2.31b}$$

Thus, the Beale algorithm for $f \neq q$ and ELS is equivalent to the PCG algorithm with the preconditioner $\hat{H}_t$. However the matrix $\hat{H}_t$ is not positive definite and it cannot be guaranteed that the directions in (2.31) are downhill. In Chapter 3 we will show how this problem can be overcome.

Finally we will introduce the preconditoned Beale restart (PBR) algorithm since it will be used in Section 3.3 in connection with the CGQN algorithm. The PBR algorithm can be derived in exactly the same way as the PCG algorithm and is given as follows:

Given a positive definite matrix H, a starting point x_{t-1} and an arbitrary downhill direction d_t, compute

$$x_t = x_{t-1} + \lambda_t d_t \ ,$$

$$d_{t+1} = -Hg_t + \frac{g_t^T H y_t}{d_t^T y_t} d_t \ , \text{ and then}$$

for $k = t+1, t+2, \ldots,$ iterate with

$$x_k = x_{k-1} + \lambda_k d_k \quad , \tag{2.32a}$$

$$\hat{\beta}_k = \frac{g_k^T H y_k}{d_k^T y_k} \quad , \quad \hat{\delta}_k = \frac{g_k^T H y_t}{d_t^T y_t} \quad ; \tag{2.32b}$$

$$d_{k+1} = -Hg_k + \hat{\beta}_k d_k + \hat{\delta}_k d_t \; . \tag{2.32c}$$

2.3 Quasi-Newton Methods

The development of QN methods (also called variable metric methods) began in 1959 with Davidon [9] introducing the ideas of the DFP method, which were later developed into a viable algorithm by Fletcher and Powell [13] in 1963. Broyden [3] in 1967 generalized the theory of the DFP algorithm to a family of algorithms, now commonly called the Broyden β-class. Quasi-Newton methods were developed from Newton methods which we shall first briefly introduce.

The basic idea of the Newton method is to approximate a given function f in each iteration by a quadratic function q. Assuming that we are at the point x_k we evaluate $f(x_k)$, g_k and the Hessian G_k. The quadratic function q_k which coincides with these values of f(x) at x_k is given by

$$q_k(x) = \tfrac{1}{2}(x-x_k)^T G(x-x_k) + (x-x_k)^T g_k + f(x_k) .$$

The minimum x* of q_k satisfies $\nabla q_k = G(x-x_k) + g_k = 0$ which implies that $x^* = x_k - G^{-1} g_k$. When minimizing a general function f the Newton method uses x* as a new point for the next iteration. The iteration formula is then

$$d_k = -G_k^{-1} g_k \quad ,$$

$$x_{k+1} = x_k + \lambda_k d_k \quad , \quad k = 0,1,2,\ldots .$$

The value of λ_k is sometimes taken as $\lambda_k = 1$ or it may be determined by a line search, depending on which variation of the Newton method one uses. However near the solution $\lambda_k = 1$ is always satisfactory.

The main advantage of Newton type methods is that if they do converge, convergence is usually quadratic. However, they have several serious drawbacks which often make them impractical for general problems. First, the method often fails to converge to a solution from a poor initial estimate. Second, the explicit evaluation of second derivatives for general functions is often impossible or at least very difficult and time consuming. The third disadvantage is the necessity of solving a set of linear equations at each iteration.

Quasi-Newton methods were developed to overcome these disadvantages while preserving a fast rate of convergence. The main idea underlying the variable metric methods is to build an approximation H_k to the inverse Hessian using the information obtained during the descent process. The conditions imposed on H_k vary depending on the particular algorithm. However the so-called quasi-Newton equation

$$H_k y_k = s_k \qquad (2.33)$$

where $s_k = x_k - x_{k-1}$, is always required to hold. Note that equation (2.33) is exactly satisfied if we apply the Newton method to q, where the Hessian and its inverse are constant. We shall restrict our attention to the Broyden family of algorithms, where $H_k = H_{k-1} + C_k$, C_k being a rank one or rank two correction matrix.

The algorithm is:

Given x_0 and an initial approximation H_0 to the inverse Hessian, define

$d_1 = -H_0 g_0$. For $k = 1,2,\ldots$, iterate with

$$x_k = x_{k-1} + \lambda_k d_k \quad , \qquad (2.34a)$$

$$H_k = H_{k-1} + \frac{H_{k-1} y \; y^T H_{k-1}}{a} + \frac{s s^T}{b} + \beta \; a \; w \; w^T \quad , \qquad (2.34b)$$

$$d_{k+1} = -H_k g_k \qquad (2.34d)$$

where $a = y^T H_{k-1} y$,

$b = s^T y$

$$w = \frac{H_{k-1} y}{a} + \frac{s}{b}$$

and the missing subscripts here and elsewhere are k. The scalar $\beta \geq 0$ determines the update formula and should not be confused with β_k used in the CG algorithm. When $\beta = 0$, (2.34b) is called the DFP update H_k^D. Throughout this thesis we shall be referring to (2.34b) with $\beta = 1$. This is the BFGS update formula which was independently developed in papers by Broyden [4], Fletcher [11], Goldfarb [14] and Shanno [25] in 1970. There is now strong computational and theoretical evidence that this is one of the best updates of the Broyden class (see e.g. Shanno and Phua [27]). The formula for the BFGS update is often written as

$$H_k = \left(I - \frac{sy^T}{s^Ty}\right) H_{k-1} \left(I - \frac{ys^T}{s^Ty}\right) + \frac{ss^T}{s^Ty} . \qquad (2.35)$$

In this thesis we will often write $H^* = U(x_k, H)$ meaning H^* is the BFGS update of the positive definite matrix H computed at x_k.

We now state some important properties of the rank-2 correction methods. The following theorem, which is a special case of Powell's theorem [24], states that the algorithm (2.34) preserves positive definiteness in the updates generated.

<u>Theorem 3</u> Let H_{k-1} be a symmetric positive definite matrix, let s_k be any non-zero vector in the space spanned by columns of H_{k-1} and let y_k be any vector that satisfies the condition

$$s_k^T y_k > 0 \ . \tag{2.36}$$

Then, if the vector w is non-zero and $\beta \geq 0$, the matrix H_k given by (2.34b) is positive definite.

Proof We consider the value $\beta = 0$, i.e. the DFP update H_k^D . It can be shown from the positive definiteness of H_{k-1} and from the Cauchy-Schwarz inequality that H_k^D is positive definite (see e.g. Luenberger [17]). In particular, for $\beta = 0$, H_k^D has n eigenvalues greater than zero. Note that (2.34b) can be written as

$$H_k = H_k^D + \beta \ a \ w \ w^T \ , \quad (a > 0).$$

The dependence of eigenvalues on the parameter β (Wilkinson [32]) is such that, if $\beta > 0$, the eigenvalues will right shift. Specifically if $u_1 \leq u_2 \leq \ldots \leq u_n$ are ordered eigenvalues of the matrix H_k^D, and $v_1 \leq v_2 \leq \ldots \leq v_n$ are eigenvalues of H_k, and if $\beta > 0$ and $a > 0$, then $u_1 \leq v_1 \leq u_2 \leq \ldots \leq u_n \leq v_n$. Hence all the eigenvalues of H_k will be positive. QED.

It is important to note that Theorem 3 does not require any ELS or quadratic f. Any value of λ_k chosen so that $s_k^T y_k > 0$ will guarantee positive definiteness in the next update, and hence a downhill direction d_{k+1} given by (2.34c). The condition (2.36) is in practice generally replaced by the slightly stronger line search criterion

$$\left| d_k^T g_k \right| < k_1 \left| d_k^T g_{k-1} \right| , \qquad (2.37a)$$

where usually $k_1 = 0.9$. In addition to (2.37a), the condition

$$f(x_k) - f(x_{k-1}) < k_2 \; (s_k^T g_{k-1}) \qquad (2.37b)$$

with $k_2 = 0.0001$ should hold to assure reasonable progress to the minimum (Fletcher [11]).

As we mentioned previously, QN methods were developed to preserve the convergence properties of the Newton methods. The Broyden updates are chosen so that the directions are mutually conjugate when minimizing $f = q$ (Broyden [3]). Then, by the EST, finite termination is guaranteed. This results in superlinear convergence when the algorithm is applied to a general f (Stachurski [30]).

A modification of the Broyden family of methods was recently developed by Oren [19], Oren and Luenberger [20] and Oren and Spedicato [21]. They introduced scaling into the Broyden updates in order to improve the stepwise rate of convergence. These algorithms alter Broyden's single parameter family to a double parameter family. The matrix H_{k-1} is now updated by the formula

$$H_k = \left(H_{k-1} - \frac{H_{k-1} y y^T H_{k-1}}{a} + \beta \, a \, w \, w^T \right) \gamma + \frac{s s^T}{b} \quad . \qquad (2.38)$$

The scaling parameter γ is chosen to make the sequence of points generated according to (2.34) invariant when the objective function is multiplied by a constant. This would be the case if the Newton algorithm were applied; if $\hat{f} = cf$ then $\hat{g} = cg$ and $\hat{G} = cG$ and the Newton iteration yields

$$x_{k+1} = x_k - \hat{G}_k^{-1}\hat{g}_k = x_k - c^{-1}G_k^{-1}cg_k = x_k - G_k^{-1}g_k, \quad k \geq 1.$$

Besides the invariancy condition, other conditions are imposed on the choice of γ. Oren and Spedicato [21] considered the problem of minimizing the upper bound of the condition number of the matrix $H_{k-1}^{-1}H_k$ since it was shown (e.g. Luenberger [17]) that reducing this condition number is important for decreasing the roundoff error. They derived the relationship

$$\beta = \frac{b(c - b\gamma)}{\gamma(ac - b^2)} \qquad (2.39)$$

where $c = s^T H_{k-1} s$ and a, b and β are defined as before.

Based on numerical results, Shanno and Phua [28] recommended using γ when $\beta = 1$, especially for moderate to large size problems. When $\beta = 1$, (2.39) simplifies to

$$\gamma = \frac{b}{a} \qquad (2.40)$$

and substitution of $\beta = 1$ and (2.40) into (2.38) gives

$$H_k = \left(H_{k-1} - \frac{H_{k-1} y y^T H_{k-1}}{a} + a\, w\, w^T\right) \frac{b}{a} + \frac{s s^T}{b} \quad . \qquad (2.41)$$

The update (2.41) will be referred to as the scaled BFGS update.

In the next chapter we will show how the idea of updates and γ scaling introduced here for QN methods can be applied to algorithms with limited storage requirements.

CHAPTER III

COMBINED CG AND QN METHODS

We now consider algorithms developed to solve the problem (1.1) with moderate or large n which are in general computationally superior to the standard CG methods and still require just 0(n) locations of storage. All methods to be discussed in this chapter are modifications of the CG algorithm (2.12) inspired by some ideas introduced for QN methods in Section 2.3. In particular, it will be shown how the BFGS update can be successfully implemented into CG methods without any major increase in their storage requirements.

First, in Section 3.1 we will introduce the memoryless quasi-Newton (MQN) method (Shanno [26]) which uses 7n locations of storage. Then in Section 3.2, the mixed CG and QN (CGQN) method by Buckley [7] will be described. This method is designed to use a variable amount of storage, the minimum being 8n locations. Finally, in Section 3.3 we will make some observations regarding the relationship between the MQN and CGQN methods. These observations and a study of advantages and disadvantages of the two methods will lead us to development of a new algorithm (VSQN) with variable storage requirements. This method with 7n locations of storage is exactly the same as the MQN method. But as Chapter 4 will show, the computational efficiency of this method improves in direct relation to the amount of storage available.

3.1 The MQN Algorithm

In this section the MQN method given by Shanno [26] will be introduced. The derivation of the method will be based on the observation in Chapter 2 that the Beale restart method is in fact a special form of a preconditioned conjugate gradient method. This approach, which is different from the one given by Shanno, points out a certain analogy to an algorithm given by Buckley [7]. This will be discussed in subsequent sections of this chapter.

First we introduce Shanno's observation that conjugate gradient methods are BFGS quasi-Newton methods where the approximation to the inverse Hessian is indeed updated at every step, but always from the identity matrix.

Shanno's work was motivated by a method proposed by Perry [22], who noted that a CG direction can be rewritten as

$$d_{k+1} = -\left(I - \frac{d_k y_k^T}{d_k^T y_k}\right) g_k \qquad (3.1)$$

as observed in Section 2.1. Perry suggested adding a correction term to the matrix transforming g_k in (3.1) and substituting $s_k = \lambda_k d_k$, thus obtaining

$$d_{k+1} = \left(I - \frac{s_k y_k^T}{s_k^T y_k} + \frac{s_k s_k^T}{s_k^T y_k}\right) g_k \tag{3.2}$$

$$\equiv -P_k g_k \; .$$

He justified his choice of the correction term in (3.2) by noting that the matrix P_k satisfies the equation

$$P_k^T y_k = s_k \; ,$$

which is similar but not identical to the QN equation (2.33), since P_k is not symmetric. He also noted that when ELS are done, i.e. $s_k^T g_k = 0$, the direction (3.2) reduces to the standard CG direction (3.1). Perry tested his new method, without exact line searches, against the Fletcher-Reeves and Polak-Ribière algorithms and showed some increase in computational efficiency for his choice of test functions.

However, since the matrix P_k is not positive definite, the search directions are not necessarily downhill and this can cause the method to be numerically unstable. To avoid this problem Shanno suggested adding to P_k in (3.2) the correction term

$$- \frac{y_k s_k^T}{s_k^T y_k}$$

to obtain a symmetric matrix. Then he showed that if the QN equation (2.33) is required to be satisfied, the matrix must be modified further; it is given by

$$Q_k^* = I - \frac{s_k y_k^T + y_k s_k^T}{s_k^T y_k} + \left(1 + \frac{y_k^T y_k}{s_k^T y_k}\right)\frac{s_k s_k^T}{s_k^T y_k} . \qquad (3.3)$$

If ELS are done, $s_k^T y_k = 0$ and the direction

$$d_{k+1} = -Q_k^* g_k$$

again reduces to the standard CG direction (3.1). Note that Q_k^* can be rewritten as

$$Q_k^* = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) I \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{s_k^T y_k}$$

which is the BFGS update of the identity matrix. Thus, Shanno points out that the conjugate gradient method is precisely the BFGS quasi-Newton method, where the approximation to the inverse Hessian is reset to the identity matrix at every step. Also we wish to emphasise that the derivation of this result introduces an important idea which will be used throughout this chapter. This is the idea of modifying a rank deficient matrix defining a transformation of the steepest descent step into a positive definite matrix, so that when ELS are done, the resulting transformation is identical to the original one. Based on these observations Shanno then develops the memoryless QN method, which we now derive.

Consider the Beale restart method with Powell's restart technique introduced in Section 2.1. Powell [23] showed that this method is more efficient than the Fletcher-Reeves or Polak-Ribiere algorithms restarted every n steps. However, to obtain a stable algorithm, line searches are required to be fairly exact, which usually implies more functional evaluations per iteration. This typifies CG algorithms and it is considered to be their major disadvantage. So it is desirable to find some modifications of the above method in which the line search can be relaxed.

It was shown in Section 2.2 that assuming f = q and ELS, the BR method is equivalent to a preconditioned CG method with matrix

$$\hat{H}_t = \left(I - \frac{d_t y_t^T}{y_t^T d_t} \right). \tag{3.4}$$

However the matrix $\hat{H}_t$ is not suitable as a preconditioner since it is singular, hence not positive definite so non-downhill directions may be generated. So as a first step leading to Shanno's algorithm, consider modifying the matrix $\hat{H}_t$ using the same idea as when (3.1) was changed into (3.3). Thus we obtain a positive definite matrix

$$H_t = I - \frac{s_t y_t^T}{y_t^T s_t} + \left(1 + \frac{y_t^T y_t}{s_t^T y_t}\right)\frac{s_t s_t^T}{s_t^T y_t} . \tag{3.5}$$

Now the PCG directions in (2.26), with $\hat{H}_t$ replaced by H_t, are given by

$$d_{t+1} = -H_t g_t , \tag{3.6a}$$

$$d_{k+1} = -H_t g_t + \frac{y_k^T H_t g_k}{d_k^T y_k} d_k , \quad k > t . \tag{3.6b}$$

Note that the PCG algorithm with the directions ~~(3.6)~~ is identical to the original BR method if $f = q$ and ELS, since then $s_t^T g_k = 0$ for $k \geq t$, and

$$H_t g_k = g_k - \frac{s_t y_t^T g_k + y_t (s_t^T g_k)}{y_t^T s_t} + \left(1 + \frac{y_t^T y_t}{s_t^T y_t}\right)\frac{s_t (s_t^T g_k)}{s_t^T y_t}$$

$$= g_k - \frac{s_t y_t^T g_k}{y_t^T s_t} = \hat{H}_t g_k .$$

Replacing $H_t g_k$ by $\hat{H}_t g_k$ in (3.6), we get back the PCG directions with the matrix $\hat{H}_t$, i.e. the original BR directions.

Observe that the line search requirements for the PCG algorithm with H_t are similar to those of CG algorithm; hence they still need to be quite accurate. So as a second step leading to the MQN method, note that the direction (3.6b) can be written in QN-like form as

$$d_{k+1} = -\left(H_t - \frac{d_k y_k^T H_t}{d_k^T y_k}\right) g_k = -\hat{H}_k g_k . \quad (3.7)$$

As before, the matrix $\hat{H}_k$ can be modified with extra terms to get a symmetric matrix satisfying the QN equation (2.33); then instead of $\hat{H}_k$ we get

$$H_k = H_t - \frac{s_k y_k^T H_t + H_t y_k s_k^T}{s_k^T y_k} + \left(1 + \frac{y_k^T H_t y_k}{s_k^T y_k}\right) \frac{s_k s_k^T}{s_k^T y_k} . \quad (3.8)$$

Using the modified PCG method with the matrix H_t, written in the QN-like form as in (3.7) but with $\hat{H}_k$ replaced by H_k, we have the following algorithm:

Given x_t and a downhill direction d_t, let

$$d_{t+1} = -H_t g_t ; \quad (3.9a)$$

then for $k = t + 1, t + 2, \ldots,$ iterate with

$$x_k = x_{k-1} + \lambda_k d_k , \quad (3.9b)$$

$$d_{k+1} = -H_k g_k , \quad (3.9c)$$

where H_t and H_k are given by (3.5) and (3.8) respectively. Restarts are done according to Powell's criterion (2.20).

The matrix H_k is a BFGS update of the positive definite matrix H_t, the vectors s_k $(k > t)$ are all in the column space of H_t since H_t has rank n, so by Theorem 3, H_k is positive definite provided the condition

$$s_k^T y_k > 0 , \; k > t \quad (3.10)$$

is satisfied. Therefore the directions in (3.9a) and (3.9c) are downhill whenever the line search ensures that (3.10) holds.

Assuming f = q and ELS, the algorithm (3.9) reduces to the BR method, because then $s_k^T g_k = 0$ which implies that

$$-H_k g_k = -H_t g_k + \frac{s_k y_k^T H_t g_k + H_t g_k (s_k^T g_k)}{s_k^T y_k} - \left(1 + \frac{y_k^T H_t y_k}{s_k^T y_k}\right) \frac{s_k (s_k^T g_k)}{s_k^T y_k}$$

$$= -H_t g_k + \frac{y_k^T H_t g_k}{s_k^T y_k} s_k = -\hat{H}_k g_k, \quad k > t \ , \qquad (3.11)$$

which is the same as (3.7), hence equivalent to the BR direction. Thus the algorithm (3.9) has the finite termination property, and it is the basic MQN method.

Now consider scaling. The QN methods build an approximation to the inverse Hessian and as the search proceeds they behave in a similar way to Newton methods so $\lambda_k = 1$ can often be used successfully. However this is not true of the method given by (3.9) since the approximation to the inverse Hessian is at each iteration just an update of the matrix H_t which is updated from the identity matrix at the restart point x_t. Thus the length of the direction generated in (3.9) does not have any relation to the true step size to the optimum. Shanno considered several scaling techniques used in other methods including the Fletcher scaling [10] and the γ scaling discussed in Section 2.3. Fletcher suggested scaling the CG direction by

$$\hat{d}_{k+1} = \frac{2(f_k - f_{k-1})}{g_k^T d_{k+1}} d_{k+1} , \qquad (3.12)$$

which is used in practical CG methods with good results. Based on numerical experience, Shanno and Phua [28] suggested that using γ given by (2.40) at every step of the BFGS algorithm is harmful for general nonlinear functions, since it introduces both truncation and approximation errors in the estimate of the inverse Hessian. However they also found that using γ at the initial step was important, especially for large problems. It reduces the truncation error which results from using the identity matrix as an initial approximation to the inverse Hessian.

Motivated by these results, Shanno suggested scaling the matrix H_t in the algorithm (3.9), but not the matrix H_k. Using (2.41) with H_{k-1} replaced by I the scaled matrix H_t is given by

$$H_t = \left(I - \frac{s_t y_t^T + y_t s_t^T}{s_t^T y_t} + \frac{y_t^T y_t}{s_t^T y_t} \; \frac{s_t s_t^T}{s_t^T y_t}\right) + \frac{s_t s_t^T}{s_t^T y_t} . \qquad (3.13)$$

Shanno also tested the application of the Fletcher scaling and found that in general such an approach is better than (3.13). However in some cases (3.13) was far superior. Therefore he proposed using only the scaled H_t at restart steps, and at each non-restart step to scale according to the Fletcher scaling. The computational results demonstrated that this seems to be the best choice, the major reason for this being that, together with Powell's restart criterion, the proposed method has the ability to select automatically the preferable scaling.

The MQN algorithm is then the double update algorithm (3.9) with H_t given by (3.13) where the direction (3.9c) is scaled by (3.12). As to the implementation of the MQN method, we first note that no matrix needs to be stored since (3.9c) can be written as

$$d_{k+1} = -H_t g_k + \frac{s_k^T g_k}{s_k^T y_k} H_t y_k - \left(\left(1 + \frac{y_k^T H_t y_k}{s_k^T y_k}\right) \frac{s_k^T g_k}{s_k^T y_k} - \frac{y_k^T H_t g_k}{s_k^T y_k}\right) s_k \tag{3.14}$$

where the vectors $H_t g_k$ and $H_t y_k$ are defined by

$$H_t g_k = \frac{s_t^T y_t}{y_t^T y_t} g_k - \frac{s_t^T g_k}{y_t^T y_t} y_t + \left(2 \frac{s_t^T g_k}{s_t^T y_t} - \frac{y_t^T g_k}{y_t^T y_t}\right) s_t \tag{3.15}$$

and

$$H_t y_k = \frac{s_t^T y_t}{y_t^T y_t} y_k - \frac{s_t^T g_k}{y_t^T y_t} y_t + \left(2 \frac{s_t^T y_k}{s_t^T y_t} - \frac{y_t^T y_k}{y_t^T y_t}\right) s_t . \tag{3.16}$$

It follows that only 7 vectors are required to be stored, as for the BR method. From the formulas (3.14) - (3.16) it is also evident that all the computations needed to obtain d_{k+1} are of order n, which is again the same as for the CG methods.

The line search criterion for the MQN algorithm is given by (2.37) the same as for QN methods, since again only the condition (3.10) is required to hold to guarantee downhill directions.

Shanno showed by numerical comparisons that the MQN method works better than the BR algorithm with Powell's restarts where the line search criterion is much stricter. We will see similar results in Chapter 4 when we compare Shanno's algorithm with Buckley's method.

3.2 The CGQN Algorithm

This is a variable metric CG method developed by Buckley [7] and it combines the CG and QN methods in an attempt to preserve their main advantages, i.e. the low storage requirements of CG methods and the rapid convergence of the QN methods. The CGQN algorithm is implemented to use a variable amount of storage depending on the availability of space, with a minimum requirement of 8n locations. The idea of variable storage will be used again in the next section where it will be shown how it can be successfully applied to Shanno's algorithm.

Now we will make a few observations which led Buckley to develop his algorithm. Consider the PCG algorithm (2.26) applied to a general f. It was indicated previously that restarting every n iterations with the direction $d_{t+1} = -Hg_t$ assures good ultimate convergence, provided the algorithm is continued with the same matrix H from that point on. So the matrix H can be changed every n steps without effecting the convergence properties. But Buckley suggests that it might be desirable to restart and change the metric H quite frequently to suit the observed behaviour of the objective function. He proposes the QN updates as a natural choice for the matrix H, noting that the usual storage requirement of n^2 locations can be reduced.

The reason for this is that the QN updates consist of a sequence of rank 1 or rank 2 corrections to an initial matrix which is usually defined as the identity. Instead of storing the updated matrix, one can just store the vectors defining these corrections. Therefore if the number of QN updates is limited, the storage needed to record the updated matrix can be much less than $O(n^2)$ locations.

Before we continue to make some more observations, we outline briefly the proposed strategy for the CGQN algorithm:

Given x_0 and a matrix H_0 (usually $H_0 = I$), set $i = 0$ and $t = 0$ and let

$$d_{t+1} = -H_i g_t \ . \tag{3.17a}$$

Then for $k = t+1,\ t+2, \ldots,$ iterate with (3.17b)
(2.26a) - (2.26c), the PCG algorithm using
the fixed matrix H_i. When a restart is
indicated (the criterion for restarting will
be given later) go to the next step.

Reset t to the current k. Update the matrix (3.17c)
H_i by the BFGS formula to get the matrix

$$H_{i+1} = H_i - \frac{vs^T}{b} + s\left(\frac{b+a}{b^2}\,s - \frac{v}{b}\right)^T$$

where $a = y^T H_i y$, $b = s^T y$, $v = H_i y$ and
the missing subscripts are t.

Record the update H_{i+1} by storing the two
vectors v and s and the two scalars a and b.

Replace i by i + 1 and repeat from (3.17a).

Note that the matrix H_{i+1} is not explicitly computed and that only (2n + 2) locations of storage are needed to record the corrections to the matrix H_i. Thus if the algorithm is started with $H_o = I$, the update H_m, $m = 0,1,2,\ldots,$ requires m(2n + 2) locations of storage to be completely recorded.

We shall now consider an important feature particular to the CGQN algorithm, the intermittent restarts and intermittent updating of the matrix H_i as described in (3.17), and what effect this has on the finite termination property. We already know that if the PCG algorithm (2.26) is applied to $f = q$ then the minimum x^* is reached in at most n iterations. Now suppose we apply the algorithm (3.17) to q with restarts done intermittently. Then if restarts are always done before the nth iteration, a consequence may be that the minimum point x^* is never found. But Buckley [6] proved that this situation does not occur, if H_i is updated by the BFGS formula as in (3.17c). He proved that for $f = q$ and ELS, the sequence of points $x_o, x_1, x_2, \ldots$ obtained by the PCG algorithm with a preconditioner H_o and the sequence of points $\bar{x}_o, \bar{x}_1, \bar{x}_2, \ldots$ determined by the CGQN algorithm are exactly the same, provided the starting point is the same in both cases. This is independent of how often the matrix H_i is changed, i.e. of the frequency of restarts. We shall now state this important result with a simpler proof than in [6]. To simplify the notation let $H^* = U(x_k, H)$, let d_{k+1} be the PCG direction (2.26), let $\bar{d}_{k+1}$ be the CGQN direction and let S_k be the vector space spanned by $H_o g_o, H_o g_1, \ldots, H_o g_k$.

<u>Theorem 4</u> (Buckley [6]). Suppose $x_i = \bar{x}_i$,

for $i = 0, 1, \ldots, k$, and assume

$$Hv = H_o v, \text{ for all } v \perp S_{k-1} . \tag{3.18}$$

Then

$$\bar{d}_{k+1} = d_{k+1} \text{ and} \tag{3.19}$$

$$H^*v = H_o v, \text{ for all } v \perp S_k. \tag{3.20}$$

<u>Proof</u> First recall (section 2.1) that in the PCG algorithm, with $f = q$ and with ELS,

$$s_k \varepsilon S_{k-1} \quad \text{and} \tag{3.21}$$

$$g_k \perp S_{k-1}, \quad \text{so} \tag{3.22}$$

$$s_k^T g_k = 0 \quad . \tag{3.23}$$

Suppose no update is done at $\bar{x}_k$. Then $H^* = H$ and (3.20) is immediate from (3.18) since if $v \perp S_k$ then

$v \perp S_{k-1}$ ($S_{k-1} \subseteq S_k$). Also (3.18) and (3.22) imply that

$$\begin{aligned} \bar{d}_{k+1} &= -Hg_k + \frac{y_k^T H g_k}{d_k^T y_k} d_k \\ &= -H_o g_k + \frac{y_k^T H_o g_k}{d_k^T y_k} d_k \\ &\equiv d_{k+1} . \end{aligned}$$

On the other hand, if we update H to H* at $\bar{x}_k$, we have

$$\bar{d}_{k+1} = -H^* g_k$$

with

$$H^* = \left(I - \frac{s_k y_k^T}{s_k^T y_k}\right) H \left(I - \frac{y_k s_k^T}{s_k^T y_k}\right) + \frac{s_k s_k^T}{s_k^T y_k}. \qquad (3.24)$$

Applying (3.23) to (3.24), we get from (3.18) and (3.22), as above,

$$\begin{aligned} \bar{d}_{k+1} &= -H^* g_k \\ &= -Hg_k + \frac{y_k^T H g_k}{d_k^T y_k} d_k \\ &= -H_o g_k + \frac{y_k^T H_o g_k}{d_k^T y_k} d_k \\ &\equiv d_{k+1} \end{aligned}$$

and (3.19) is proved.

Now suppose $v \perp S_k$. Then $v \perp S_{k-1}$ and $s_k^T v = 0$ so from (3.18) and (3.24)

$$H^* v = Hv - \frac{y_k^T H v}{d_k^T y_k} d_k$$

$$= H_o v - \frac{y_k^T H_o v}{d_k^T y_k} d_k$$

$$= H_o v.$$

QED.

Applying this theorem inductively then shows that the algorithms generate identical points from start to finish.

Buckley [6] also noted that Theorem 4 is specific to the BFGS algorithm. If other updates from the β-class are used, then H^*g_k contains components along directions other than g_k and d_k. In this case Theorem 4 does not apply and the finite termination property cannot be guaranteed.

Now consider the implementation of the CGQN algorithm, beginning with the restarting criterion. If the CG algorithm is applied to $f = q$, the gradients are mutually orthogonal. The corresponding result for the PCG algorithm is that

$$g_i^T H g_j = 0 \text{ for } i \neq j \;.$$

In the case of a general function f one can not expect this to hold. But a substantial deviation from zero, especially near the minimum where it should be zero, could indicate that the preconditioning matrix is not simulating the local behaviour of f too well. So Buckley suggests a restart when

$$\left|\frac{g_{k-1}^T H_i g_k}{g_k^T H_i g_k}\right| > 0.2 \tag{3.25}$$

which is similar to Powell's criterion (2.20).

The line search strategy used is the same as for the PCG algorithm (2.26) except that the condition $s^T y > 0$ must be included to assure that the matrix H_i is positive definite. Those conditions are more restrictive than the ones given for Shanno's routine. In Chapter 4 we will give some computational results to demonstrate how the difference in line search strategies effect the performance of the algorithms.

We noted earlier that the CGQN method does not require storing any matrix and that it is designed to work with a variable amount of storage. The amount of storage available to the routine must be specified by the user. We have not yet mentioned what strategy to use when the assigned storage limit is reached. Suppose that m updates were recorded (recall that $m(2n + 2)$ locations of storage are then used) and the given limit is reached. Then there are several possibilities. One can delete the vectors which record some of the old updates. But computational results (Buckley [7]) showed that this usually causes the new update H_{m+1} to become non-positive definite and then the stability of the algorithm is affected. The simplest strategy of discarding all previous updates and starting with the identity again proved to be computationally most efficient according to Buckley.

Buckley [7] showed that his method in general outperforms the Fletcher and Reeves CG algorithm and it compares with the BR method with Powell's restart criterion. In the next section we will show how this method can be modified to improve the computational efficiency.

3.3 The VSQN Algorithm

We will first make some observations about the CGQN and the MQN methods and show that although they were derived in different manners, they are identical under certain conditions. Then we shall review some advantages and disadvantages of these method and discuss possible modifications to avoid some of their drawbacks. Finally, motivated by these ideas we develop a new algorithm which could be interpreted as a variable storage extension of Shanno's MQN algorithm.

We will now show that the CGQN algorithm applied to a quadratic q with ELS generates the same directions as the preconditioned Beale's method (2.32) between two consecutive restarts. Suppose the CGQN algorithm is used. Let x_t be the restart point and let H denote the matrix defined when a step is taken from the point x_{t-1}. Let $H^* = U(x_t,H)$ and assume there are no more restarts after the one at x_t, so the CGQN directions $d_{t+1}, d_{t+2} \ldots$ are the PCG directions (2.26c) with the preconditioner H^*.

Now, since $s_t^T g_k = 0$ for $k \geq t$, we see that

$$H^* g_k = Hg_k - \frac{s_t y_t^T H g_k}{s_t^T y_t}, \quad k \geq t. \tag{3.27}$$

Also

$$\hat{\beta}_k = \frac{y_k^T H^* g_k}{d_k^T y_k} = \frac{1}{d_k^T y_k}\left(y_k^T H g_k - \frac{(y_k^T s_t) y_t^T H g_k}{d_t^T y_t} \right) \tag{3.28}$$

$$= \frac{y_k^T H g_k}{d_k^T y_k}, \quad k > t .$$

Substituting (3.27) and (3.28) into the formulas defining the PCG directions in (2.26) we obtain

$$d_{t+1} = -H^* g_t = Hg_t - \frac{y_t^T H g_t}{d_t^T y_t} d_t$$

and

$$d_{k+1} = -H^* g_k + \hat{\beta}_k d_k$$

$$= -Hg_k + \frac{y_t^T H g_k}{d_t^T y_t} d_t + \frac{y_k^T H g_k}{d_k^T y_k} d_k, \quad k > t ,$$

which are in fact the PBR directions (2.32).

Now consider the CGQN method where only the correction vectors for 1 update are allowed to be stored throughout the computation. According to the strategy (3.17) of the CGQN method, the new matrix H^* is then computed at each restart point as the BFGS update of the identity matrix, i.e. $H^* \equiv H_t$, H_t defined by (3.5). Then $H \equiv I$ and the method reduces to the BR method. Now consider the MQN method, but without the γ scaling of H_t. We have shown by (3.7) and (3.11) that it also reduces to BR method under the same condition. Hence if $f = q$ and with ELS the CGQN method with 1 update stored and the MQN method without the γ scaling are identical. In fact they are equivalent even when $f \neq q$ but with ELS, since it was shown in (3.11) that the MQN method reduces to the PCG method with the preconditioner H_t.

The basic difference in implementing the two methods, the MQN and the CGQN with 1 update, are the requirements for the line searches. Since the CGQN uses the PCG method, the line searches need to be quite accurate to produce a stable algorithm. Hence in general more functional evaluations per iteration are required then if the MQN algorithm is applied.

To relax the accuracy requirement for the line searches in the CGQN algorithm, consider applying the same technique used by Shanno in modifying the PCG direction in the derivation of his algorithm. We would then obtain the CGQN algorithm modified in the following way. At the restart point x_t compute the BFGS update H^* of the previous preconditioner H and the direction $d_{t+1}=-H^*g_t$. The direction d_{k+1} is modified by Shanno's strategy by adding extra terms and is given by

$$d_{k+1} = -H_k g_k, \ k > t$$

where $H_k = U(x_k,H^*)$ as in (3.8) with H^* used instead of H_t. The requirements for the line searches are then the same as for the MQN algorithm, i.e. the condition $s^Ty > 0$ is all that must be satisfied to obtain downhill directions. Note that the modified CGQN method reduces back to the CGQN algorithm (3.17) if ELS are assumed; hence the finite termination property is not affected.

Now let us examine in some detail the updating strategy of the CGQN method. Suppose the storage is available to record m updates and we have reached the point $x_{t,i}$, $i < m$, the ith restart point. According to the strategy of the CGQN algorithm, the matrix $H_i = U(x_{t,i}, H_{i-1})$ is computed and then one continues with the PCG method (2.26) with the metric H_i until the next restart point $x_{t,i+1}$. The matrix H_i is the ith BFGS update and contains some information about the behaviour of the function f at the points $x_{t,1}, x_{t,2}, \ldots, x_{t,i}$. One would expect that the convergence of the algorithm improves if more updates are stored, i.e. with larger m. However the computational results show that this is not usually true, unless m is close to n.

One possible reason for this behaviour is the following. Suppose that the quadratic region N of the hypothetical function $\hat{f}$ is reached, a restart is indicated and the matrix H_i, $i < m$ is computed in the usual way. Most of the informations contained in the matrix H_i was accumulated at the previous restart points outside N, which are no longer relevant to the current behaviour of the function $\hat{f}$. Thus the preconditioner $H_i = U(x_{t,i}, H_{i-1})$ might not be more useful in speeding up the convergence than the matrix given by $H_i = U(x_{t,i}, I)$.

However if more of the recent information could be contained in the matrix H_i, then H_i should be a better approximation to the inverse Hessian in the quadratic region than the matrix $H_i = U(x_{t,i}, I)$. This suggests revising the updating strategy in the following way. When a restart is indicated at some point x_t, discard all previous updates and compute the matrix $H_t = U(x_t, I)$. Then continue with the steps of the BFGS QN algorithm, only storing the correction vectors defining the updates as described for the CGQN method. When the storage limit for m updates is reached, i.e. when the last update H_{t+m-1} is recorded, switch to the PCG method (2.26) with the preconditioner H_{t+m-1} and continue until the next restart point is reached.

The finite termination property is not affected by this modification in the updating strategy. This can be seen by considering the m QN iterations as m consecutive restarts and applying Buckley's Theorem 4. This theorem applies to any number of intermittent restarts interleaved with PCG steps; hence it also applies in our case.

Two possible modifications of the CGQN algorithm have now been considered; first the modification of the PCG direction into the QN-like form and second, revising the updating strategy by updating continuously until the storage limit is reached. Combining these we obtain a new algorithm which we will refer to as a variable storage QN (VSQN) algorithm. This algorithm can be viewed as an extension of Shanno's 2-update method to a multiple update method.

The VSQN algorithm with m updates is given as follows.

Given x_o and the initial positive definite matrix

$H_o \equiv I$, compute

$d_1 = -H_o g_o$,

$x_1 = x_o + \lambda_1 d_1$.

For $k = 1,2,\ldots, m$ iterate with

$$H_k = U(x_k, H_{k-1}) , \tag{3.29a}$$

$$d_{k+1} = -H_k g_k , \tag{3.29b}$$

$$x_{k+1} = x_k + \lambda_{k+1} d_{k+1} . \tag{3.29c}$$

For $k = m+1, m+2,\ldots$ replace (3.29a) with

$$H_k = U(x_k, H_m) . \tag{3.29d}$$

When a restart is indicated, say at x_t, reset $t = 1$, $H_0 = I$ and repeat from (3.29a).

Note that if $m = 1$, the algorithm above is identical to Shanno's MQN algorithm.

The implementation of the VSQN algorithm is much the same as for the CGQN except of course for the line search requirements which are the same as for the QN methods given in Section 2.3. Again, no matrix is stored, but only the correction vectors as in the CGQN algorithm and $5n + m(2n + 2)$, $m = 1,2,...$ locations of storage required are specified by the user. With $m = 1$, 1 update is stored and we have the MQN method. The effect on the computational performance of the VSQN method when m is increased will be seen in Chapter 4. The restart criterion for the new algorithm could be either (3.25) as for the CGQN but with $i = m$, or (2.20), Powell's restart criterion, since by Buckley's Theorem 4 when $f = q$ and with ELS they are equivalent. In Chapter 4 we will give test results for the VSQN method with (3.25) and (2.20) and it will be shown which one should be preferred in practice. In the next chapter we will discuss some more details regarding the implementation of the VSQN routine.

CHAPTER IV

NUMERICAL RESULTS

In this chapter we compare the performance of some of the algorithms of interest when they are applied to a variety of standard test problems.

This comparison can be done in several ways. We have chosen a commonly used approach which is to compare the total number of function and gradient evaluations necessary to reach the desired result. The reason for this choice is that in practical applications the computer time used in evaluating the function is often the major part of the total machine time required to solve the problem. We will however also compare the total CPU times. This is important because our main objective is to compare algorithms when applied to problems of moderate to high dimensions, in which case the time used for housekeeping computations may not be negligible.

4.1 Algorithms and Functions Tested

In Section 4.2 we will give test results for the new VSQN algorithm (3.29) with different choices for the line search and restart strategies.

In Section 4.3 we will compare the VSQN algorithm with the BFGS and the MQN routines, both documented by Shanno. The BFGS routine is an implementation of the BFGS method with initial γ scaling as described in Section 2.3. It requires $n(n + 1)/2$ locations of storage and computations are of order n^2.. The MQN routine is an implementation of Shanno's conjugate gradient algorithm with inexact line searches (3.9), i.e. the memoryless QN method. It requires $7n$ locations of storage and computations are of order n. The BFGS and the MQN routines are generally considered to be very successful implementations of the QN and CG classes of algorithms. We also include some test results for the CGQN algorithm (3.17) documented by Buckley.

All of the routines described above were linked to a test package TESTPACK developed by Buckley [5]. This test package provides a means for a uniform testing procedure for unconstrained minimization algorithms. It contains a large number of standard test problems from which we have selected the functions with the largest dimensions. All the runs were made on a CDC-CYBER 170-800 computer.

The functions tested, with the initial point x_o and dimension n, are:

EXTROS: An extended Rosenbrock function [28].

$$f(x) = \sum_{i=1}^{n/2} \left(100\,(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i})^2\right),$$

for n even ;

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

$n = 10$ and $n = 20$.

TRIDIA: A tridiagonal quadratic function [29].

$$f(x) = \sum_{i=2}^{n} (i-1)\,(2x_i - x_{i-1})^2 \ ,\ n \geq 2\ ;$$

$$x_o^T = (-1.0,\ -1.0,\ \ldots,\ -1.0)\ ;$$

$n = 20$ and $n = 30$.

NONDIA: A nondiagonal variant of the Rosenbrock function [29].

$$f(x) = \sum_{i=2}^{n} \left(100\,(x_1 - x_i^2)^2 + (1 - x_i)^2\right),\ n \geq 2\ ;$$

$$x_o^T = (-1.0,\ -1.0,\ \ldots,\ -1.0)\ ;$$

$n = 20$ and $n = 30$.

MANCIN: The Mancino function [28].

$$f(x) = \sum_{i=1}^{n} f_i^2, \text{ where}$$

$$f_i = \sum_{\substack{i=1 \\ i \neq j}}^{n} v_{ij} (\sin^5 \log v_{ij} + \cos^5 \log v_{ij}) + 14nx_i + (i-n/2)^3,$$

and $v_{ij} = (x_j^2 + i/j)^{\frac{1}{2}}$;

$x_o^T = (a\ f_1(0),\ a\ f_2(0),\ \ldots,\ a\ f_n(0))$,

where $a = -7n \ / \ (80n^2 + 36n - 18)$;

$n = 20$

CHAROS: A chained extension of the Rosenbrock function [31].

$$f(x) = \sum_{i=2}^{n} \left(4\ \alpha_i\ (x_{i-1} - x_i^2)^2 + (1 - x_i^2)\right),\ n \geq 2\ ,$$

where α_i are constants defined in [31] ;

$x_o^T = (-1.0,\ -1.0,\ \ldots,\ -1.0)$;

$n = 10$ and $n = 25$.

POWELL: Powell's function with a singular minimum [28].

$$f(x) = \sum_{i=1}^{n/4} \left((x_{4i-3}+10x_{4i-2})^2 + 5(x_{4i-1}-x_{4i})^2 \right.$$

$$\left. + (x_{4i-2}-2x_{4i-1})^4 + 10(x_{4i-3}-x_{4i})^4 \right) ;$$

$$x_o^T = (3.0,\ -1.0,\ 0.0,\ 1.0,\ -1.0,\ \ldots,\ 1.0) ;$$

$n = 60$ and $n = 80$.

OREN: Oren's power function [28].

$$f(x) = \left(\sum_{i=1}^{n} i\, x_i^2 \right)^2 ;$$

$$x_o^T = (1.0,\ 1.0,\ \ldots,\ 1.0) ;$$

$n = 50$ and $n = 75$.

4.2 Numerical Results For the VSQN Algorithm

Test results for the VSQN algorithm (3.29) with different line search and restarting strategies will be given here. In all cases computation was terminated when $\|g\|_2 \leq 10^{-5}$ and the line search was terminated when (2.37) was satisfied.

We shall first point out the main differences in the line strategy (Section 1.2) for the BFGS and the MQN algorithm. In the BFGS routine the search starts with $\lambda = 1$ on every step since a reasonable scaling of the length of a direction is provided automatically (Section 2.3). Also, the best results are obtained with only the termination criterion (2.37) being satisfied; a quadratic interpolation is not required at every step. This usually implies more iterations but fewer functional calls, i.e. the ratio of the number of total functional evaluations to the number of iterations (RATIO) is then significantly less than 2. On the other hand the line search for the MQN method starts with $\lambda = 1$ only on the first or the first restart iteration, and the initial step length along the other directions is defined by (1.2). Shanno found that the MQN method works better if the quadratic interpolation is forced at every iteration. This results in general in fewer iterations but more functional evaluations; RATIO in this case is usually more than 2.

We have tested the VSQN algorithm with different combinations of the line search strategies described above. The QN strategy was modified by setting the initial step length $\lambda = 1$ only for all "QN" steps. Specifically, if m corrections defining the m BFGS updates are allowed to be stored, the search along each of $d_1, d_2, \ldots, d_{m+1}$

begins with $\lambda = 1$ (recall, at restarts, $t = 1$). For the following "CG" directions, $d_{m+2}, d_{m+3}, \ldots$ the line search begins with λ defined by (1.2). The quadratic interpolation was forced at every iteration right from d_1. In Tables 1 and 2 (see appendix) we give an example of the results obtained with this line search strategy for $m = 2$ and $m = 8$ respectively. Note that the results did not improve significantly with the increased number of updates stored, and RATIO did not decrease, as would be expected.

Table 3 - 6 give results when $m = 2,4,6$, and 8, respectively, and when the line search was the same as above but with the quadratic interpolation forced only on the "CG" steps, $d_{m+2}, d_{m+3}, \ldots$. The results in Tables 3 - 6 indicate that the VSQN algorithm exhibits the expected behaviour discussed in Section 3.3, i.e. increasing the storage available substantially improves the performance of the algorithm. Also note that RATIO decreases as more QN updates are stored.

We have also tested the VSQN algorithm with different restarting criteria, as described in Section 3.3. The results in Tables 1 - 6 were obtained when the Powell restart criterion (2.20) was used. The test results obtained with the restart condition (3.25) are given in Tables 7 - 10, again with $m = 2,4,6$ and 8, respectively. Comparison, for example, of Table 3 and 7 where $m = 2$ indicates that the simpler restart criterion (2.20) should be preferred in this case. Also comparing Table 4 - 6 and 8 - 10 shows that the algorithm with the Powell restart criterion improves in a more uniform fashion with increased storage. Since there is no significant benefit obtained for the extra work required in computing (3.25), the criterion (2.20) is more suitable in general for the algorithm.

Finally we would like to note that tests were carried out for the VSQN method with m = 10,15, and 20. The results were not much better than the ones given for m = 8. The total number of function evaluations for the same 13 problems tested fluctuated between 650 and 685. We have not found any satisfactory explanation for this behaviour.

4.3 Comparison of the Algorithms

We now compare the BFGS, the MQN, the CGQN and the VSQN methods. In Table 11 a comparison of the number of iterations (ITER) and functional evaluations (FEVAL) is made. Note that results for two versions of the CGQN routine are given. First, CGQN is the algorithm described in Section 3.2. In general it outperforms the standard CG algorithm (see Buckley [7]), although it is inferior to the MQN method. In Section 3.3 we have suggested some modifications to improve its performance. To demonstrate the effect of the γ scaling of the initial approximation to the inverse Hessian, the results obtained using the CGQNG (CGQN with γ) are included. Both versions used 8n locations of storage, i.e. only 1 update was stored at a time. The entries for the number of functional evaluations in Table 11 clearly show that the MQN routine is superior to either version of CGQN which uses exact searches.

A comparison of the entries in Table 11 shows that, by providing additional storage, the number of functional evaluations can be substantially reduced. As expected the BFGS algorithm is still in many cases superior, but note that for many problems the difference is slight.

In Table 12 the total machine time required for solving each problem (MSEC) and the part of this time used for evaluation of the function (FSEC) are documented. We see that an increase of storage for the VSQN method does not lead to a major increase in time, since the increase in computational effort is only of order n. Of course, if m were larger, the increase in run time would be substantial. To demonstrate this, we include Table 13 where results are given for the VSQN routine with enough storage available so that a QN step is done at every iteration without restarting or switching to the modified CG steps. (This does not apply to the runs made for the two verions of the Oren function, where due to insufficient storage a few restarts were made.) Thus the iteration and functional evaluation counts in Table 13 are in general comparable to the ones given for the BFGS routine, except that the CPU times are larger for the VSQN routine. However, the object of the new algorithm is to operate with limited storage and therefore this situation is not realistic.

The run times in Table 12 for the MQN and VSQN algorithm are superior to the times given for BFGS, because the computations in BFGS are of order n^2. The difference in the CPU times is most apparent for the Powell function with $m = 80$. However, if the function is very expensive to evaluate, even a few extra function evaluations significantly increase the total run time. This in fact occurs for the Mancino function. Hence the BFGS method is preferable. In summary, if storage is limited the VSQN method with several updates seems to be the best alternative among the algorithms under discussion.

4.4 Conclusion

We have introduced an algorithm to solve moderate to large-scale unconstrained optimization problems. It is an extension of Shanno's 2 update BFGS quasi-Newton method as a multiple update variable storage method such as the one introduced by Buckley. We have shown on a variety of test problems that this idea does indeed lead to improved convergence over Shanno's method, even when only a few more vectors are stored throughout the computations.

APPENDIX

TABLES

TABLE 1 (2 UPDATES, FORCED QI ON EVERY STEP)
TEST BEING EXECUTED AT 10:52 A.M., MAY 18, 1981

STANDARD CONTROL PARAMETERS
TERMINATION NORM = 2 (EUCLIDEAN)
TERMINATION TYPE = 1 (GRADIENT)
ACCURACY SPECIFIED .100E-04

PR#	FN#	NAME	DIM	ITS	FNCS	GRDS	FVALUE	GVALUE	MSECS	FSECS	ER	RS
31	8	EXTROS10	10	26	66	66	.52E-16	.81E-14	.156	.050	0	12
46	8	EXTROS20	20	26	66	66	.52E-16	.81E-14	.260	.082	0	12
26	6	TRIDIA20	20	38	77	77	.51E-12	.90E-10	.342	.083	0	2
27	6	TRIDIA30	30	51	103	103	.22E-12	.24E-10	.652	.152	0	2
23	5	NONDIA20	20	24	58	58	.20E-18	.40E-15	.195	.058	0	11
24	5	NONDIA30	30	23	54	54	.92E-14	.79E-10	.255	.088	0	11
37	10	MANCIN20	20	6	13	13	.15E-17	.47E-12	17.902	17.864	0	1
33	9	CHAROS10	10	32	65	65	.26E-11	.68E-10	.168	.041	0	9
34	9	CHAROS25	25	47	95	95	.18E-11	.77E-10	.531	.156	0	7
21	11	POWELL60	60	34	69	69	.31E-10	.34E-11	.568	.091	0	15
47	11	POWELL80	80	37	75	75	.52E-10	.22E-11	.784	.130	0	17
35	7	OREN50	50	42	85	85	.57E-08	.34E-10	.593	.113	0	20
40	7	OREN75	75	40	81	81	.11E-07	.95E-10	.817	.154	0	19
				ITERS	FUNCS	GRADS	RATIO		MSECS	FSECS		
TOTALS				426	907	907	2.13		23.223	19.062		

NONE OF THE PROBLEMS WERE FLAGGED WITH ERRORS.

COMPUTATION DONE WITH THE BUCKLEY-LENIR EXTENSION OF SHANNO'S ROUTINE.

METH	NTERMS	NTEST	ATEST	CTEST	SCDIAG	SCGAMA	SCGAMF	HTEST	RO	BETA
0	2	0	0	3	F	F	T	F	.20	1.00

```
-----------------------------------------------------------------------
!            TABLE 2 ( 8 UPDATES, FORCED QI ON EVERY STEP )           !
!           TEST BEING EXECUTED AT 11:02 A.M., MAY 18, 1981           !
-----------------------------------------------------------------------
!                    STANDARD CONTROL PARAMETERS                      !
!                 TERMINATION NORM =  2 (EUCLIDEAN)                   !
!                 TERMINATION TYPE =  1 (GRADIENT )                   !
!                 ACCURACY SPECIFIED        .100E-04                  !
-----------------------------------------------------------------------
```

PR#	FN#	NAME	DIM	ITS	FNCS	GRDS	FVALUE	GVALUE	MSECS	FSECS	ER	RS
31	8	EXTROS10	10	27	69	69	.33E-13	.89E-11	.238	.062	0	4
46	8	EXTROS20	20	27	69	69	.33E-13	.89E-11	.399	.096	0	4
26	6	TRIDIA20	20	38	77	77	.98E-12	.86E-10	.483	.072	0	3
27	6	TRIDIA30	30	48	97	97	.19E-12	.30E-10	.991	.138	0	2
23	5	NONDIA20	20	25	57	57	.56E-19	.30E-15	.298	.058	0	3
24	5	NONDIA30	30	23	54	54	.93E-14	.58E-10	.366	.080	0	3
37	10	MANCIN20	20	6	13	13	.89E-17	.28E-11	17.858	17.820	0	1
33	9	CHAROS10	10	39	79	79	.39E-12	.36E-10	.268	.053	0	5
34	9	CHAROS25	25	51	103	103	.35E-12	.17E-10	.843	.175	0	4
21	11	POWELL60	60	22	45	45	.12E-09	.67E-10	.533	.063	0	3
47	11	POWELL80	80	25	51	51	.13E-09	.44E-10	.831	.089	0	3
35	7	OREN50	50	44	89	89	.36E-08	.29E-10	.949	.122	0	6
40	7	OREN75	75	48	97	97	.89E-08	.68E-10	1.560	.185	0	6
				ITERS	FUNCS	GRADS	RATIO		MSECS	FSECS		
TOTALS				423	900	900	2.13		25.617	19.013		

NONE OF THE PROBLEMS WERE FLAGGED WITH ERRORS.

COMPUTATION DONE WITH THE BUCKLEY-LENIR EXTENSION OF SHANNO'S ROUTINE.

METH	NTERMS	NTEST	ATEST	CTEST	SCDIAG	SCGAMA	SCGAMF	HTEST	RO	BETA
0	8	0	0	3	F	F	T	F	.20	1.00

! TABLE 3 (2 UPDATES, POWELL REST. CRITERION (2.20)) !
! TEST BEING EXECUTED AT 11:07 A.M., MAY 18, 1981 !

! STANDARD CONTROL PARAMETERS !
! TERMINATION NORM = 2 (EUCLIDEAN) !
! TERMINATION TYPE = 1 (GRADIENT) !
! ACCURACY SPECIFIED .100E-04 !

PR#	FN#	NAME	DIM	ITS	FNCS	GRDS	FVALUE	GVALUE	MSECS	FSECS	ER	RS
31	8	EXTROS10	10	35	51	51	.73E-15	.46E-13	.152	.034	0	17
46	8	EXTROS20	20	35	51	51	.73E-15	.46E-13	.265	.067	0	17
26	6	TRIDIA20	20	45	81	81	.35E-12	.35E-10	.376	.082	0	6
27	6	TRIDIA30	30	52	90	90	.85E-12	.80E-10	.595	.131	0	8
23	5	NONDIA20	20	36	59	59	.12E-15	.67E-13	.248	.066	0	17
24	5	NONDIA30	30	36	61	61	.17E-18	.23E-14	.349	.092	0	17
37	10	MANCIN20	20	8	10	10	.23E-18	.74E-13	13.824	13.789	0	4
33	9	CHAROS10	10	59	77	77	.53E-11	.38E-10	.237	.065	0	26
34	9	CHAROS25	25	50	73	73	.38E-12	.32E-10	.456	.123	0	16
21	11	POWELL60	60	56	81	81	.54E-10	.12E-10	.866	.109	0	23
47	11	POWELL80	80	71	99	99	.62E-08	.89E-10	1.386	.178	0	31
35	7	OREN50	50	34	37	37	.94E-08	.51E-10	.395	.053	0	17
40	7	OREN75	75	39	43	43	.64E-08	.32E-10	.661	.080	0	19
				ITERS	FUNCS	GRADS	RATIO		MSECS	FSECS		
TOTALS				556	813	813	1.46		19.809	14.869		

NONE OF THE PROBLEMS WERE FLAGGED WITH ERRORS.

COMPUTATION DONE WITH THE BUCKLEY-LENIR EXTENSION OF SHANNO'S ROUTINE.

METH	NTERMS	NTEST	ATEST	CTEST	SCDIAG	SCGAMA	SCGAMF	HTEST	RO	BETA
0	2	0	0	0	F	F	T	F	.20	1.00

TABLE 4 (4 UPDATES, POWELL REST. CRITERION (2.20))
TEST BEING EXECUTED AT 11:10 A.M., MAY 18, 1981

STANDARD CONTROL PARAMETERS
TERMINATION NORM = 2 (EUCLIDEAN)
TERMINATION TYPE = 1 (GRADIENT)
ACCURACY SPECIFIED .100E-04

PR#	FN#	NAME	DIM	ITS	FNCS	GRDS	FVALUE	GVALUE	MSECS	FSECS	ER	RS
31	8	EXTROS10	10	34	47	47	.32E-13	.88E-11	.184	.033	0	8
46	8	EXTROS20	20	34	47	47	.32E-13	.88E-11	.316	.060	0	8
26	6	TRIDIA20	20	40	69	69	.47E-12	.64E-10	.414	.071	0	3
27	6	TRIDIA30	30	51	86	86	.47E-12	.27E-10	.706	.131	0	6
23	5	NONDIA20	20	32	46	46	.26E-18	.22E-14	.241	.045	0	8
24	5	NONDIA30	30	29	39	39	.80E-15	.10E-10	.310	.060	0	7
37	10	MANCIN20	20	6	8	8	.10E-15	.31E-10	11.024	10.991	0	2
33	9	CHAROS10	10	53	71	71	.61E-12	.45E-10	.266	.051	0	11
34	9	CHAROS25	25	64	76	76	.19E-11	.81E-10	.601	.119	0	15
21	11	POWELL60	60	43	55	55	.17E-08	.13E-11	.782	.073	0	10
47	11	POWELL80	80	89	121	121	.42E-09	.13E-10	2.127	.222	0	21
35	7	OREN50	50	36	40	40	.21E-08	.88E-11	.521	.057	0	9
40	7	OREN75	75	44	53	53	.26E-08	.14E-10	.937	.096	0	11
				ITERS	FUNCS	GRADS	RATIO		MSECS	FSECS		
TOTALS				555	758	758	1.37		18.429	12.009		

NONE OF THE PROBLEMS WERE FLAGGED WITH ERRORS.

COMPUTATION DONE WITH THE BUCKLEY-LENIR EXTENSION OF SHANNO'S ROUTINE.

METH	NTERMS	NTEST	ATEST	CTEST	SCDIAG	SCGAMA	SCGAMF	HTEST	RO	BETA
0	4	0	0	0	F	F	T	F	.20	1.00

TABLE 5 (6 UPDATES, POWELL REST. CRITERION (2.20))
TEST BEING EXECUTED AT 11:12 A.M., MAY 18, 1981

STANDARD CONTROL PARAMETERS
TERMINATION NORM = 2 (EUCLIDEAN)
TERMINATION TYPE = 1 (GRADIENT)
ACCURACY SPECIFIED .100E-04

PR#	FN#	NAME	DIM	ITS	FNCS	GRDS	FVALUE	GVALUE	MSECS	FSECS	ER	RS
31	8	EXTROS10	10	42	54	54	.16E-16	.20E-13	.259	.040	0	7
46	8	EXTROS20	20	42	54	54	.16E-16	.20E-13	.452	.078	0	7
26	6	TRIDIA20	20	58	64	64	.15E-11	.89E-10	.481	.067	0	10
27	6	TRIDIA30	30	64	78	78	.18E-11	.92E-10	.753	.116	0	11
23	5	NONDIA20	20	33	45	45	.54E-15	.40E-13	.282	.048	0	6
24	5	NONDIA30	30	31	43	43	.49E-17	.98E-14	.382	.065	0	5
37	10	MANCIN20	20	6	8	8	.41E-16	.13E-10	11.039	11.004	0	1
33	9	CHAROS10	10	41	55	55	.73E-12	.38E-10	.234	.048	0	6
34	9	CHAROS25	25	62	78	78	.28E-11	.63E-10	.673	.127	0	10
21	11	POWELL60	60	35	65	65	.56E-07	.63E-10	.762	.084	0	6
47	11	POWELL80	80	34	64	64	.78E-07	.96E-10	.961	.115	0	6
35	7	OREN50	50	37	40	40	.88E-08	.69E-10	.623	.051	0	6
40	7	OREN75	75	47	50	50	.26E-08	.88E-11	1.142	.095	0	8
			ITERS		FUNCS	GRADS	RATIO		MSECS	FSECS		
TOTALS				532	698	698	1.31		18.043	11.938		

NONE OF THE PROBLEMS WERE FLAGGED WITH ERRORS.

COMPUTATION DONE WITH THE BUCKLEY-LENIR EXTENSION OF SHANNO'S ROUTINE.

METH	NTERMS	NTEST	ATEST	CTEST	SCDIAG	SCGAMA	SCGAMF	HTEST	RO	BETA
0	6	0	0	0	F	F	T	F	.20	1.00

TABLE 6 (8 UPDATES, POWELL REST. CRITERION (2.20))
TEST BEING EXECUTED AT 11:15 A.M., MAY 18, 1981

STANDARD CONTROL PARAMETERS
TERMINATION NORM = 2 (EUCLIDEAN)
TERMINATION TYPE = 1 (GRADIENT)
ACCURACY SPECIFIED .100E-04

PR#	FN#	NAME	DIM	ITS	FNCS	GRDS	FVALUE	GVALUE	MSECS	FSECS	ER	RS
31	8	EXTROS10	10	41	53	53	.23E-16	.39E-14	.298	.040	0	5
46	8	EXTROS20	20	41	53	53	.23E-16	.39E-14	.516	.070	0	5
26	6	TRIDIA20	20	53	63	63	.65E-12	.43E-10	.500	.063	0	7
27	6	TRIDIA30	30	61	83	83	.18E-12	.38E-10	.948	.128	0	6
23	5	NONDIA20	20	30	43	43	.87E-14	.56E-10	.289	.043	0	4
24	5	NONDIA30	30	32	42	42	.56E-17	.60E-13	.427	.060	0	4
37	10	MANCIN20	20	6	8	8	.41E-16	.13E-10	11.011	10.976	0	1
33	9	CHAROS10	10	49	57	57	.55E-12	.62E-10	.298	.042	0	6
34	9	CHAROS25	25	52	63	63	.11E-11	.52E-10	.674	.106	0	6
21	11	POWELL60	60	37	46	46	.19E-10	.66E-10	.836	.065	0	5
47	11	POWELL80	80	37	44	44	.16E-07	.26E-10	1.093	.084	0	5
35	7	OREN50	50	43	47	47	.27E-08	.11E-10	.824	.058	0	6
40	7	OREN75	75	48	55	55	.85E-08	.75E-10	1.361	.099	0	6
				ITERS	FUNCS	GRADS	RATIO		MSECS	FSECS		
TOTALS				530	657	657	1.24		19.075	11.834		

NONE OF THE PROBLEMS WERE FLAGGED WITH ERRORS.

COMPUTATION DONE WITH THE BUCKLEY-LENIR EXTENSION OF SHANNO'S ROUTINE.

METH	NTERMS	NTEST	ATEST	CTEST	SCDIAG	SCGAMA	SCGAMF	HTEST	RO	BETA
0	8	0	0	0	F	F	T	F	.20	1.00

TABLE 7 (2 UPDATES, REST. CRITERION WITH H (3.25))
TEST BEING EXECUTED AT 11:20 A.M., MAY 18, 1981

STANDARD CONTROL PARAMETERS
TERMINATION NORM = 2 (EUCLIDEAN)
TERMINATION TYPE = 1 (GRADIENT)
ACCURACY SPECIFIED .100E-04

PR#	FN#	NAME	DIM	ITS	FNCS	GRDS	FVALUE	GVALUE	MSECS	FSECS	ER	RS
31	8	EXTROS10	10	34	55	55	.84E-15	.81E-12	.191	.043	0	13
46	8	EXTROS20	20	34	55	55	.84E-15	.81E-12	.332	.067	0	13
26	6	TRIDIA20	20	45	81	81	.35E-12	.35E-10	.441	.078	0	6
27	6	TRIDIA30	30	57	91	91	.54E-12	.36E-10	.711	.137	0	13
23	5	NONDIA20	20	35	50	50	.13E-13	.45E-10	.249	.051	0	16
24	5	NONDIA30	30	30	46	46	.17E-14	.12E-11	.318	.069	0	13
37	10	MANCIN20	20	8	10	10	.23E-18	.74E-13	13.746	13.705	0	4
33	9	CHAROS10	10	42	61	61	.33E-12	.12E-10	.206	.044	0	15
34	9	CHAROS25	25	73	114	114	.34E-12	.35E-10	.819	.190	0	18
21	11	POWELL60	60	109	150	150	.45E-07	.62E-10	1.909	.206	0	46
47	11	POWELL80	80	106	158	158	.29E-08	.14E-11	2.492	.274	0	44
35	7	OREN50	50	34	37	37	.94E-08	.51E-10	.444	.047	0	17
40	7	OREN75	75	39	43	43	.64E-08	.32E-10	.750	.081	0	19
				ITERS	FUNCS	GRADS	RATIO		MSECS	FSECS		
TOTALS				646	951	951	1.47		22.608	14.992		

NONE OF THE PROBLEMS WERE FLAGGED WITH ERRORS.

COMPUTATION DONE WITH THE BUCKLEY-LENIR EXTENSION OF SHANNO'S ROUTINE.

METH	NTERMS	NTEST	ATEST	CTEST	SCDIAG	SCGAMA	SCGAMF	HTEST	RO	BETA
0	2	0	0	0	F	F	T	T	.20	1.00

! TABLE 8 (4 UPDATES, REST. CRITERION WITH H (3.25)) !
! TEST BEING EXECUTED AT 11:23 A.M., MAY 18, 1981 !

STANDARD CONTROL PARAMETERS
TERMINATION NORM = 2 (EUCLIDEAN)
TERMINATION TYPE = 1 (GRADIENT)
ACCURACY SPECIFIED .100E-04

PR#	FN#	NAME	DIM	ITS	FNCS	GRDS	FVALUE	GVALUE	MSECS	FSECS	ER	RS
31	8	EXTROS10	10	35	47	47	.64E-16	.15E-13	.212	.036	0	8
46	8	EXTROS20	20	35	47	47	.64E-16	.15E-13	.366	.064	0	8
26	6	TRIDIA20	20	42	63	63	.28E-11	.54E-10	.458	.060	0	6
27	6	TRIDIA30	30	51	86	86	.47E-12	.27E-10	.853	.134	0	6
23	5	NONDIA20	20	32	46	46	.26E-18	.22E-14	.267	.047	0	8
24	5	NONDIA30	30	30	44	44	.13E-14	.90E-11	.359	.076	0	7
37	10	MANCIN20	20	6	8	8	.10E-15	.31E-10	11.109	11.074	0	2
33	9	CHAROS10	10	40	68	68	.60E-12	.38E-10	.288	.045	0	5
34	9	CHAROS25	25	44	75	75	.10E-12	.61E-11	.647	.132	0	5
21	11	POWELL60	60	63	78	78	.13E-10	.22E-11	1.199	.110	0	16
47	11	POWELL80	80	78	107	107	.57E-09	.26E-10	2.011	.190	0	19
35	7	OREN50	50	36	40	40	.21E-08	.88E-11	.562	.046	0	9
40	7	OREN75	75	43	47	47	.26E-08	.52E-10	.975	.084	0	11
				ITERS	FUNCS	GRADS	RATIO		MSECS	FSECS		
TOTALS				535	756	756	1.41		19.306	12.098		

NONE OF THE PROBLEMS WERE FLAGGED WITH ERRORS.

COMPUTATION DONE WITH THE BUCKLEY-LENIR EXTENSION OF SHANNO'S ROUTINE.

METH	NTERMS	NTEST	ATEST	CTEST	SCDIAG	SCGAMA	SCGAMF	HTEST	RO	BETA
0	4	0	0	0	F	F	T	T	.20	1.00

TABLE 9 (6 UPDATES, REST. CRITERION WITH H (3.25))
TEST BEING EXECUTED AT 11:26 A.M., MAY 18, 1981

STANDARD CONTROL PARAMETERS
TERMINATION NORM = 2 (EUCLIDEAN)
TERMINATION TYPE = 1 (GRADIENT)
ACCURACY SPECIFIED .100E-04

PR#	FN#	NAME	DIM	ITS	FNCS	GRDS	FVALUE	GVALUE	MSECS	FSECS	ER	RS
31	8	EXTROS10	10	37	50	50	.34E-14	.92E-12	.260	.041	0	6
46	8	EXTROS20	20	37	50	50	.34E-14	.92E-12	.451	.068	0	6
26	6	TRIDIA20	20	48	64	64	.28E-12	.62E-10	.539	.066	0	6
27	6	TRIDIA30	30	51	79	79	.49E-12	.84E-10	.980	.113	0	4
23	5	NONDIA20	20	33	45	45	.54E-15	.40E-13	.307	.047	0	6
24	5	NONDIA30	30	31	44	44	.10E-13	.57E-10	.415	.066	0	5
37	10	MANCIN20	20	6	8	8	.41E-16	.13E-10	11.004	10.971	0	1
33	9	CHAROS10	10	57	72	72	.25E-12	.26E-10	.351	.054	0	9
34	9	CHAROS25	25	62	98	98	.95E-12	.66E-10	1.056	.156	0	5
21	11	POWELL60	60	35	65	65	.56E-07	.63E-10	.816	.090	0	6
47	11	POWELL80	80	34	64	64	.78E-07	.96E-10	1.036	.116	0	6
35	7	OREN50	50	37	40	40	.88E-08	.69E-10	.670	.058	0	6
40	7	OREN75	75	47	50	50	.26E-08	.88E-11	1.231	.095	0	8

	ITERS	FUNCS	GRADS	RATIO	MSECS	FSECS
TOTALS	515	729	729	1.42	19.116	11.941

NONE OF THE PROBLEMS WERE FLAGGED WITH ERRORS.

COMPUTATION DONE WITH THE BUCKLEY-LENIR EXTENSION OF SHANNO'S ROUTINE.

METH	NTERMS	NTEST	ATEST	CTEST	SCDIAG	SCGAMA	SCGAMF	HTEST	RO	BETA
Q	6	0	0	0	F	F	T	T	.20	1.00

TABLE 10 (8 UPDATES, REST. CRITERION WITH H (3.25))
TEST BEING EXECUTED AT 11:28 A.M., MAY 18, 1981

STANDARD CONTROL PARAMETERS
TERMINATION NORM = 2 (EUCLIDEAN)
TERMINATION TYPE = 1 (GRADIENT)
ACCURACY SPECIFIED .100E-04

PR#	FN#	NAME	DIM	ITS	FNCS	GRDS	FVALUE	GVALUE	MSECS	FSECS	ER	RS
31	8	EXTROS10	10	38	52	52	.20E-13	.33E-10	.296	.039	0	5
46	8	EXTROS20	20	38	52	52	.20E-13	.33E-10	.508	.076	0	5
26	6	TRIDIA20	20	53	63	63	.65E-12	.43E-10	.541	.053	0	7
27	6	TRIDIA30	30	45	75	75	.13E-12	.31E-10	1.091	.111	0	2
23	5	NONDIA20	20	29	40	40	.24E-14	.16E-10	.305	.046	0	4
24	5	NONDIA30	30	31	45	45	.16E-13	.39E-10	.461	.064	0	4
37	10	MANCIN20	20	6	8	8	.41E-16	.13E-10	10.992	10.955	0	1
33	9	CHAROS10	10	43	62	62	.72E-12	.56E-10	.342	.038	0	5
34	9	CHAROS25	25	44	62	62	.61E-12	.63E-10	.717	.109	0	4
21	11	POWELL60	60	35	47	47	.42E-08	.33E-10	.940	.064	0	4
47	11	POWELL80	80	32	44	44	.38E-07	.73E-10	1.120	.079	0	4
35	7	OREN50	50	43	47	47	.27E-08	.11E-10	.880	.062	0	6
40	7	OREN75	75	48	55	55	.85E-08	.75E-10	1.448	.107	0	6

	ITERS	FUNCS	GRADS	RATIO	MSECS	FSECS
TOTALS	485	652	652	1.34	19.641	11.803

NONE OF THE PROBLEMS WERE FLAGGED WITH ERRORS.

COMPUTATION DONE WITH THE BUCKLEY-LENIR EXTENSION OF SHANNO'S ROUTINE.

METH	NTERMS	NTEST	ATEST	CTEST	SCDIAG	SCGAMA	SCGAMF	HTEST	RO	BETA
0	8	0	0	0	F	F	T	T	.20	1.00

TABLE 11

ITER/FEVAL

PROBLEM	BFGS	MQN	VSQN m=2	VSQN m=4	VSQN m=6	VSQN m=8	CGQN	CGQNG
EXTROS								
n=10	36/44	23/55	35/51	34/47	42/54	41/53	23/68	24/73
n=20	36/44	23/55	35/51	34/47	42/54	41/53	23/68	24/73
TRIDIA								
n=20	33/34	43/87	45/81	40/69	58/64	53/63	53/94	45/83
n=30	41/42	53/107	52/90	51/86	64/78	61/83	56/98	59/98
NONDIA								
n=20	32/43	23/53	36/59	32/46	33/45	30/43	24/70	23/68
n=30	30/41	17/40	36/61	29/39	31/43	32/42	25/74	24/67
MANCIN								
n=20	7/8	6/13	8/10	6/8	6/8	6/8	11/17	7/19
CHAROS								
n=10	37/39	36/75	59/77	53/71	41/55	49/57	49/98	37/79
n=25	53/75	59/121	50/73	64/76	62/78	52/63	73/138	55/105
POWELL								
n=60	41/42	28/59	56/81	43/55	35/65	37/46	45/111	49/122
n=80	37/38	33/69	71/99	89/121	34/64	37/44	49/120	38/99
OREN								
n=50	F/200	30/61	34/37	36/40	37/40	43/47	68/142	25/58
n=75	F/200	37/75	39/43	44/53	47/50	48/55	78/159	31/75
TOTAL	783/832	411/870	556/813	555/758	532/698	530/657	577/1257	441/1019
RATIO	1.06	2.12	1.46	1.37	1.31	1.24	2.18	2.31
TOTAL WITHOUT OREN	383/432	344/734	483/733	475/665	448/608	439/555	431/956	385/886

Key: F failed to converge

TABLE 12

MSEC/FSEC

PROBLEM	BFGS	MQN	VSQN m=2	VSQN m=4	VSQN m=6	VSQN m=8
EXTROS						
n=10	.312/.033	.136/.043	.167/.039	.193/.040	.271/.040	.332/.044
n=20	.974/.060	.237/.079	.280/.073	.324/.066	.455/.072	.538/.074
TRIDIA						
n=20	.724/.037	.418/.090	.403/.099	.428/.068	.483/.073	.504/.064
n=30	1.83/.064	.771/.170	.681/.150	.743/.127	.760/.102	.960/.126
NOND						
n=20	.702/.049	.192/.063	.268/.067	.253/.055	.306/.048	.307/.046
n=30	1.31/.063	.207/.065	.378/.104	.313/.059	.390/.067	.467/.067
MANC						
n=20	11.4/11.3	18.9/18.8	14.6/14.5	11.5/11.5	11.3/11.3	11.4/11.3
CHAROS						
n=10	.261/.029	.205/.056	.244/.058	.271/.047	.235/.039	.308/.043
n=25	1.70/.107	.768/.227	.486/.125	.628/.134	.715/.134	.699/.110
POWELL						
n=60	6.373/.060	.570/.088	.915/.119	.807/.078	.811/.088	.867/.065
n=80	9.882/.068	.894/.129	1.489/.192	2.204/.217	.989/.120	1.137/.08
OREN						
n=50	22.6/.253	.453/.084	.415/.051	.541/.055	.642/.055	.864/.064
n=75	51.3/.378	.867/.150	.697/.090	.963/.120	1.15/.09	1.43/.11
TOTAL	109.3/12.5	24.6/20.1	21.0/15.7	19.2/12.5	18.5/12.2	19.8/12.2
TOTAL WITHOUT OREN'S FUNCTION	35.4/11.8	23.3/19.8	19.9/15.6	17.7/12.4	16.7/12.1	17.5/12.0

TABLE 13 (VSQN WITH QN STEP AT EVERY ITER.(EXCEPT FOR OREN F.))
TEST BEING EXECUTED AT 11:35 A.M., MAY 18, 1981

STANDARD CONTROL PARAMETERS
TERMINATION NORM = 2 (EUCLIDEAN)
TERMINATION TYPE = 1 (GRADIENT)
ACCURACY SPECIFIED .100E-04

PR#	FN#	NAME	DIM	ITS	FNCS	GRDS	FVALUE	GVALUE	MSECS	FSECS	ER	RS
31	8	EXTROS10	10	36	44	44	.37E-13	.46E-10	.762	.040	0	1
46	8	EXTROS20	20	36	44	44	.37E-13	.46E-10	1.284	.070	0	1
26	6	TRIDIA20	20	33	34	34	.38E-13	.68E-11	.796	.043	0	1
27	6	TRIDIA30	30	41	42	42	.46E-12	.72E-10	1.661	.065	0	1
23	5	NONDIA20	20	32	46	46	.30E-13	.22E-12	.784	.049	0	1
24	5	NONDIA30	30	31	41	41	.58E-13	.16E-10	1.009	.063	0	1
37	10	MANCIN20	20	7	8	8	.72E-18	.22E-12	11.096	11.052	0	1
33	9	CHAROS10	10	37	39	39	.51E-12	.35E-10	.624	.031	0	1
34	9	CHAROS25	25	53	57	57	.11E-11	.67E-10	2.353	.100	0	1
21	11	POWELL60	60	41	42	42	.76E-11	.28E-10	2.970	.060	0	1
47	11	POWELL80	80	37	38	38	.45E-09	.30E-10	3.188	.065	0	1
35	7	OREN50	50	143	144	144	.13E-07	.85E-10	13.393	.182	0	3
40	7	OREN75	75	135	136	136	.11E-07	.89E-10	12.956	.249	0	4
				ITERS	FUNCS	GRADS	RATIO		MSECS	FSECS		
TOTALS				662	715	715	1.08		52.876	12.069		

NONE OF THE PROBLEMS WERE FLAGGED WITH ERRORS.

COMPUTATION DONE WITH THE BUCKLEY-LENIR EXTENSION OF SHANNO'S ROUTINE.

METH	NTERMS	NTEST	ATEST	CTEST	SCDIAG	SCGAMA	SCGAMF	HTEST	RO	BETA
0	0	0	0	0	F	F	T	F	.20	1.00

REFERENCES

[1] AXELSSON, O., "On preconditioning and convergence acceleration in sparse matrix problems", CERN Data Handling Division Report 74-10, (1974).

[2] BEALE, E.M.L., "A derivation of conjugate gradients" in Num. Methods for Nonlinear Optimization, F.A. Lootsma ed. (Academic Press, 1072).

[3] BROYDEN, C.G., "Quasi-Newton methods and their application to function minimization", Mathematics of Computation, 21 (1967), pp. 368-381.

[4] BROYDEN, C.G., "The convergence of a class of double-rank minimization algorithms II", JIMA, 6, pp. 222-231, 1970.

[5] BUCKLEY, A.G., "A portable package for testing minimization algorithms", (To appear in the Proceedings of COAL Conference, Boulder, Colorado, Jan 81)

[6] BUCKLEY, A.G., "Extending the relationship between the conjugate gradient and BFGS algorithm", Math. Prog. 15 (1978).

[7] BUCKLEY, A.G., "A combined Conjugate Gradient Quasi Newton minimization algorithm", Math. Prog. 15 (1978)

[8] CROWDER, H. and WOLFE, P., "Linear convergence of the conjugate gradient method", IBM Journal of Research and Development 16 (1972).

[9] DAVIDON, W.C., "Variable metric method for minimization", A.E.C. Research and Development Report, ANL-5990(1959).

[10] FLETCHER, R., "A Fortran subroutine for minimization by the method of Conjugate Gradients", Rpt. R-7073, A.E.R.E. Harwell (1972).

[11] FLETCHER, R., "A new approach to variable metric algorithms", Computer Journal, 13(1970).

[12] FLETCHER, R. and REEVES, C.M., "Function minimization by conjugate gradients", Computer Journal, 7(1964).

[13] FLETCHER, R. and POWELL, M.J.D. "A rapidly convergent descent method for minimization", Computer Journal, 7(1963).

[14] GOLDFARB, D., "A family of variable metric methods derived by variational means", Math. of Computation, 29 (1974).

[15] HESTENES, M.R. and STIEFEL, E., "Methods of conjugate gradients for solving linear systems", J. Res. Nat. Bureau Standard, 49 (1959).

[16] JADOTTE, J.T., "Conjugate gradient versus sparsity exploiting quasi Newton algorithms in unconstrained minimization", Master's thesis (Concordia University, Dept. of Math.)

[17] LUENBERGER, D., "Introduction to Linear and Nonlinear programming", Addison-Wesley (1973).

[18] MCCORMICK, G. and RITTER, K., "Methods of conjugate directions versus Quasi Newton methods", Math. Prog., 3 (1972).

[19] OREN, S.S., "Self-scaling variable metric algorithm. Part II", Management Science, 20, 1974, pp. 863-874

[20] OREN, S.S. and LUENBERGER, D., "Self-scaling variable metric algorithm, part I", Management Science, 20 (1974).

[21] OREN, S.S. and SPEDICATO, E., "Optimal conditioning or self-scaling variable metric algorithms", Math. Prog., 10 (1976).

[22] PERRY, A., "A modified Conjugate Gradient algorithm", Discussion paper No. 229. Center for Mathematical Studies in Economics and Management Science, Northwestern University, 1976.

[23] POWELL, M.J.D., "Restart procedure for the conjugate gradient methods", Math. Prog., 12 (1977)

[24] POWELL, M.J.D., "Quadratic termination properties of a class of double-rank minimization algorithms", Rpt. T.P. 471, A.E.R.E., Harwell. (1972)

[25] SHANNO, D.F., "Conditioning of a Quasi-Newton method for function minimization", Math. of Computation, 24 (1970).

[26] SHANNO, D.F., "Conjugate gradient methods with inexact searches", MIS (Univ. of Arizona, Tucson), Tech. Rpt. No. 22 (1977)

[27] SHANNO, D.F. and PHUA, K.H., "Numerical comparison of several variable metric algorithms", JOTA 25 (1978).

[28] SHANNO, D.F. and PHUA, K.H., "Matrix conditioning and non-linear optimization", Math. Prog. 14 (1978) 149 - 160.

[29] SHANNO, D.F. and PHUA, K.H., "On variable-metric methods for sparse Hessians II. The new method". MIS Tech. Rept. No. 27, p. 10.

[30] STACHURSKI, A., "Superlinear convergence of Broyden's bounded β-class of methods", Math. Prog., 2 (1981).

[31] TOINT, Ph.L., "Some numerical results using a sparse matric updating formula in unconstrained minimization", Math. of Computation, 32(1978).

[32] Wilkinson, J.H., The Algebraic Eigenvalue Problem, Oxford University Press, (1965).