University
Microfilms
International

300 N. ZEEB ROAD, ANN ARBOR, MI 48106
18 BEDFORD ROW, LONDON WC1R 4EJ, ENGLAND

GARG, NARESH KUMAR
A ROBUST MODIFICATION OF NEWTON'S METHOD FOR
NONLINEAR OPTIMIZATION.

RICE UNIVERSITY, PH.D., 1979

University
Microfilms
International 300 N. ZEEB ROAD, ANN ARBOR, MI 48106

RICE UNIVERSITY

A ROBUST MODIFICATION OF NEWTON'S METHOD FOR NONLINEAR OPTIMIZATION

by

Naresh Kumar Garg

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

DOCTOR OF PHILOSOPHY

APPROVED, THESIS COMMITTEE:

Richard A Tapia

Richard A. Tapia, Professor
Dept. of Mathematical Sciences
Chairman

C. C. W

C.-C. Wang, Professor
Dept. of Mathematical Sciences

Thomas W. Parks, Professor
Dept. of Electrical Engineering

Houston, Texas
May 1979

A ROBUST MODIFICATION OF NEWTON'S METHOD FOR NONLINEAR OPTIMIZATION

by

Naresh Kumar Garg

ABSTRACT

A numerically stable algorithm is presented, which essentially uses the preconditioned conjugate gradient method to iteratively solve the linear systems which arise in Newton's method. Directions of negative curvature are obtained and dealt with in an efficient and natural manner. A main feature of the algorithm is that the amount of storage required can be controlled by the choice of the preconditioning matrix. Preliminary numerical experimentation indicates that the method compares favorably with the now standard techniques to solve the particular optimization problem.

ACKNOWLEDGEMENTS

The greatest debt of gratitude that I owe is to my thesis director Professor Richard Tapia, not only for his unstinted advice and encouragement during the course of this study, but also for originally influencing my choice of doctoral research area. I express my deep appreciation to the many teachers and fellow graduate students from whom I have benefitted. The financial support of Rice University during the bulk of my graduate career is gratefully acknowledged, as is the U. S. Department of Energy for support in the form of computer time under contract EY-76-S-05-5046. I wish to express special thanks to Mrs. Velva Power for her expert and expeditious typing. Finally, among the several friends and relatives to whom I am sincerely thankful, I single out my favorite friend Gina and my esteemed brother Dinesh for their generosity and help in many ways.

TABLE OF CONTENTS

1. INTRODUCTION.

This thesis is concerned with the development and implementation of a numerical algorithm for solving finite dimensional nonlinear optimization problems. Our primary objective is to manifest techniques that can be used as practical tools. For this reason, special emphasis is given to robustness, by which is meant the ability of an algorithm to cope with adverse circumstances, whether due to the pathologies of a particular problem or to the shortcomings of finite precision computer arithmetic. The possession of nice theoretical properties is a necessary but far from sufficient condition for an algorithm to be effective for a wide range of practical problems.

Another obvious measure of the complexity of an optimization problem is its size, measured in terms of the number of unknown variables or the number of constraints. Much of the early theory associated with nonlinear optimization concerned the derivation of necessary and sufficient conditions for a solution. In the last ten or fifteen years the question of computation has been seriously dealt with, and several good algorithms for nonlinear optimization have been proposed. However, little work has been done on the development of algorithms that are well-suited for large problems. Most of the algorithms cannot be applied to large problems because of their storage requirements. We will be concerned with the development of an algorithm that can operate effectively with the storage available for a particular problem. Also, a major concern will be strategies that can force convergence from poor starting approximations. The typical assumptions upon which theoretical analyses are usually based--nonsingularity or positive definiteness of matrices, constraint qualifications, etc.--are frequently violated by real-life problems. Such contingencies must be taken into account if the algorithm is to be effective. The algorithms developed in this thesis reduce, in favorable circumstances, to an existing algorithm whose theoretical analysis is well known. As a result, we do not dwell on the particular point of carrying convergence analysis for the proposed algorithms. Instead, we try to deal with questions like "What to do until the local convergence theorem applies, and what if it never applies?"

In order to make this thesis reasonably self-contained, relevant notation and the necessary background material such as definitions, statement of problems to be considered, optimality conditions, and some computational tools from numerical linear algebra are given in section 2. Sections 3 and 4 contain a brief survey of existing methods--most of which are currently in use. Some of the difficulties of these methods are pointed out in section 5, and design goals for the proposed algorithms are set. In section 6 we present the underlying theory for our algorithms. Sections 7 and 8 describe the newly developed algorithms. Computational results of these algorithms when applied to a collection of test problems are presented in section 9. Some thoughts on what might be done next along the lines of the algorithms presented are included in section 10. There is also an appendix which describes a matrix update technique, which we believe to be the most appropriate update method for the purposes of our algorithms.

2. NOTATION AND BACKGROUND

In devising our notation consideration has been given to those symbols in common use which result in the least ambiguity or confusion. The background is taken from Dennis (1978a), Gill and Murray (1974), Heath (1978), Tapia (1977)(1978).

In our work, all vector spaces are finite dimensional; all scalars, vectors and matrices are real. All vectors are column vectors unless transposition is explicitly indicated. The inner product of two vectors x and y in the n-dimensional Euclidean space R^n is denoted by $\langle x,y\rangle$ or by $x^T y$. The Euclidean norm, $\|x\| = \langle x,x\rangle$, will be the only norm used. Usually, but not always, upper case letters are used for matrices, lower case letters for vectors and Greek letters for scalars. The space of all $n \times m$ matrices is denoted by $R^{n \times m}$; the matrix A whose (i,j)-th element is a_{ij} is sometimes denoted by (a_{ij}); and the transpose of A is denoted by A^T . The identity matrix is usually denoted by I, when we wish to emphasize the order we use I_k . The i-th natural basis vector (i.e., the i-th column of I) is denoted by e_i . Approximate equality between numerical quantities is denoted by $\approx$. In the statement of algorithms the symbol $:=$ is used in the sense of replacement, as in some programming languages. The symbol τ is reserved for the machine tolerance which is the smallest floating point number on a given computer such that $1+\tau > 1$.

The notation $[d_1,d_2,\ldots,d_k]$ is used to denote the linear span of the vectors $d_1,d_2,\ldots,d_k \in R^n$. Given a symmetric matrix $A \in R^{n \times n}$, two nonzero vectors d_1 and $d_2 \in R^n$ are said to be A-conjugate (or A-orthogonal) if $\langle Ad_1, d_2\rangle = 0$. If A is also positive definite the vectors d_1 and d_2 can be proved to be linearly independent. The Fréchet derivative of an operator f at x is denoted by $\partial f(x)$, and the Jacobian matrix by $Jf(x)$. We also use the notation $\nabla f(x)$ for $Jf(x)^T$. In this way when f is a functional $\nabla f(x)$ denotes the gradient of f ; and hence our notation makes sense. When f is an operator of several vector variables, say x and λ, we use the subscript x or λ to denote differentiation with respect to x or λ, respectively. In this context, no subscripts imply differentiation with respect to the total variable (x,λ) . Thus we may write

$$\nabla_x f(x,\lambda) = \partial_x f(x,\lambda) \ ; \ \nabla_\lambda f(x,\lambda) = \partial_\lambda f(x,\lambda) \ ;$$

$$\nabla f(x,\lambda) = \partial_{x,\lambda} f(x,\lambda) = \partial f(x,\lambda) \ ; \text{ etc.}$$

For second derivatives, we write

$$\nabla^2 f(x,\lambda) = \partial(\partial f(x,\lambda)) \ ; \ \nabla^2_{x\lambda} f(x,\lambda) = \partial_\lambda(\partial_x f(x,\lambda)) \ ;$$

$$\nabla^2_x f(x,\lambda) = \partial_x(\partial_x f(x,\lambda)) \ ; \text{ etc.}$$

Thus it should be obvious that when f is a functional, $\nabla^2 f(x)$ represents its Hessian matrix at x . An elementary discussion on the differentiation of nonlinear operators may be found in Tapia (1971).

In order to understand the efficiency of an algorithm in terms of its convergence rate we review some fundamental notions of convergence from Ortega and Rheinboldt (1970, Chapter 9). If a sequence $\{x^k\} \subseteq R^n$ converges to x^* , then for $p \in [1,\infty)$ the quantity

$$Q_p\{x^k\} = \overline{\lim_k}\{\|x^* - x^{k+1}\|/\|x^* - x^k\|^p\}$$

is called the quotient factor, and

$$R_p\{x^k\} = \begin{cases} \overline{\lim_k} \|x^* - x^k\|^{1/k}, & \text{if } p=1 \\ \overline{\lim_k} \|x^* - x^k\|^{1/p^k}, & \text{if } p>1 \end{cases}$$

the root factor for the sequence $\{x^k\}$. Moreover, the quantities

$$O_Q\{x^k\} = \inf\{p \in [1,\infty) : Q_p\{x^k\} = \infty\}$$

and

$$O_R\{x^k\} = \inf\{p \in [1,\infty) : R_p\{x^k\} = 1\}$$

are called the Q-order and R-order of convergence of $\{x^k\}$, respectively. The sequence is said to be Q-superlinearly convergent to x^* if

$$Q_1\{x^k\} = 0,$$

and R-superlinearly if

$$R_1\{x^k\} = 0 .$$

It may be observed that Q-convergence guarantees good behavior at each iteration, whereas R-convergence guarantees only that the average behavior is good. Thus the numerical termination criterion of checking the difference between successive iterates is dangerous when the existence of Q-convergence of the sequence is not known.

In this sense it seems desirable for an algorithm to be Q-convergent.

In the two classes of optimization problems that we consider, all the functions are assumed to be as smooth as necessary to support the thoery upon which the algorithms are based. Since a maximization problem can always be transformed to the one in minimization, we treat the optimization problems in terms of minimization. The unconstrained optimization problem of concern may be stated as follows:

$$(2.1) \qquad \text{minimize } f(x); \quad x \in R^n$$

where $f(x)$ is a prescribed nonlinear functional, $f: R^n \rightarrow R$. Notice that we will find the solution x^*, if it exists, among the zeros of the system of nonlinear equations $\nabla f(x) = 0$. It is useful to note that the Jacobian of this system, being the Hessian of f, is symmetric.

The constrained optimization problem of interest is the following:

$$(2.2) \qquad \begin{array}{l} \text{minimize } f(x); \quad x \in R^n \\ \text{subject to } \; h(x) = 0 \end{array}$$

where the objective function $f: R^n \rightarrow R$ and the constraint nonlinear functions $h: R^n \rightarrow R^m$ are prescribed. We restrict our attention only to equality constrained problems. The inclusion of inequality constraints will be considered in future work.

Let us develop some terminology relevant to the above problems which will be useful later. A point x satisfying the constraints is said to be a <u>feasible point</u>. The set of all feasible points is termed the <u>feasible set</u>. In the case of unconstrained optimization, the feasible set is all of R^n . A feasible point x^* is a local minimizer of f if $f(x^*) \leq f(x)$ for all feasible x in some neighborhood of x^* . A feasible point x^* is a global minimizer of f if $f(x^*) \leq f(x)$ for all feasible x . Note that problems (2.1) and (2.2) are stated for global minima. In practice it is extremely difficult to find or verify global minima, so we will content ourselves with algorithms for identifying local minima only.

A feasible point x is called regular if $\nabla h(x)$ has full rank. At a regular feasible point x, the space tangent to the constraint manifold is given by

$$(2.3) \qquad T(x) = \{z : \nabla h(x)^T z = 0\} \ .$$

Clearly, for unconstrained problems the space $T(x)$ is all of R^n for all x . The function

$$(2.4) \qquad P(x,r) = f(x) + \frac{1}{r} h(x)^T h(x)$$

where r is a non-negative scalar; is called the penalty function associated with problem (2.2). The classical Lagrangian function for the problem (2.2) is defined as

$$(2.5) \qquad \ell(x,\lambda) = f(x) + \lambda^T h(x)$$

where the vector $\lambda \in R^m$ is called the vector of Lagrange multipliers. The augmented Lagrangian function for problem (2.2) is given by

$$(2.6) \qquad \mathcal{L}(x,\lambda,c) = f(x) + \lambda^T h(x) + \frac{c}{2} h(x)^T h(x)$$

where λ is as above and c is a non-negative scalar known as penalty constant. The point $x \in R^n$ is said to be a critical point of problem (2.2) if there exist Lagrange multipliers $\lambda \in R^m$ such that $(x,\lambda) \in R^{n+m}$ is a solution of the nonlinear system

$$(2.7) \qquad \nabla \ell(x,\lambda) = 0$$

where by our notational convention

$$\nabla \ell(x,\lambda) = \begin{bmatrix} \nabla_x \ell(x,\lambda) \\ \\ \nabla_\lambda \ell(x,\lambda) \end{bmatrix} = \begin{bmatrix} \nabla f(x) + \nabla h(x)\lambda \\ \\ h(x) \end{bmatrix} .$$

It is well known that if a solution x^* of problem (2.2) is a regular point, then it is also a critical point of problem (2.2), and the Lagrange multipliers λ^* associated with x^* are unique. Observe that a solution of (2.7) also satisfies

$$(2.8) \qquad \nabla \mathcal{L}(x,\lambda,c) = 0 ,$$

and vice-versa. Note that

$$\nabla \mathcal{L}(x,\lambda,c) = \begin{bmatrix} \nabla_x \mathcal{L}(x,\lambda,c) \\ \\ \nabla_\lambda \mathcal{L}(x,\lambda,c) \end{bmatrix} = \begin{bmatrix} \nabla f(x) + \nabla h(x)(\lambda + ch(x)) \\ \\ h(x) \end{bmatrix} .$$

A characterization of the solution is conveyed by the following optimality theorems.

Theorem 2.1.

Sufficient (necessary) conditions that a point x^* be a local minimizer of f in problem (2.1) are

$$\nabla f(x^*) = 0$$

and

$$\nabla^2 f(x^*) \text{ is positive (semi) definite.}$$

Proof. See Luenberger (1973), pp. 110-114.

■

Theorem 2.2.

Sufficient (necessary) conditions that a point x^* be a local constrained minimizer of f in problem (2.2) are

$$\exists \lambda^* \in R^m \text{ such that } \nabla \ell(x^*,\lambda^*) = 0$$

and

$$\nabla_x^2 \ell(x^*,\lambda^*) \text{ is positive (semi) definite on } T(x^*).$$

Proof. See Luenberger (1973), pp. 224-227.

■

Note that for either problem the optimality conditions involve a stationary point of a nonlinear functional (or a solution of a set of nonlinear equations) and a criteria for determining if this stationary point is in fact a local minimizer. This provides a common basis for seeking algorithms to solve these problems and a means to confirm computationally that a solution has been found.

Many advances in iterative algorithms for optimization have gone hand-in-hand with improved techniques in numerical linear algebra. The algorithms discussed in this thesis make vital use of matrix factorizations. We provide below enough of an outline to make this concept understood in the sequel. More details can be found in Householder (1964), Stewart (1973), Wilkinson (1965).

In order to solve a system of simultaneous linear equations

$$Ax = b \;,\; A \in R^{n \times n} \;,\; x,b \in R^n \;;$$

numerical methods usually involve decomposing the matrix A into a product of from two to four simple matrices. The form of decomposition depends on the properties of A and the computing environment, and the candidate factors are such that it is easy to solve the linear system which has one of the factors as the coefficient matrix. In general, if $A = A_1A_2A_3$ then the required solution of the linear system may be obtained by solving $A_1x_1 = b$ for x_1 and then $A_2x_2 = x_1$ for x_2 and finally $A_3x = x_2$ for x. However, in practice, some of the intermediate linear systems are often solved as the decomposition proceeds. As an example, during the traditional Gaussian elimination which corresponds to a factorization $A = P^TLU$ where P is a permutation matrix and L and U are lower and upper triangular respectively, the solution to $P^Tx_1 = b$ as well as that to $Lx_2 = x_1 = Pb$ are often generated as the decomposition stage proceeds. This leaves $Ux = x_2$ as the only obvious system to be solved. We close this section by briefly cataloging the most commonly used factorizations.

The LU factorization: This is essentially Gaussian elimination, and is intended for general nonsingular matrices. The decomposition actually yields

$$PA = LU \quad \text{or} \quad A = P^TLU$$

where P is a permutation matrix, L is unit lower triangular ($\ell_{ii} = 1$) and U is upper triangular. (See Stewart (1973), Wilkinson (1965)).

The Cholesky factorization: When A is symmetric and positive definite, the factorization $A = LL^T$, where L is lower triangular, can be obtained in about half the work and storage required to obtain the LU factorization of a general matrix. Generally, in practice, the decomposition $A = LDL^T$, where L is unit lower triangular and D is a positive diagonal matrix, is used. (See Forsythe and Moler (1967), Stewart (1973)).

The symmetric indefinite factorization: If the matrix A is symmetric but not positive definite, then the decomposition $A = PLDL^TP^T$, where P is a permutation matrix, L is unit lower triangular and D is a block diagonal matrix of 1×1 and 2×2 blocks, may be obtained. (See Bunch and Kaufman (1977), Paige and Saunders (1975)).

The QR decomposition: If A has no special properties (in fact it could even be rectangular), we can write $A = QRP^T$ where P is a permutation matrix, R is upper triangular and Q is an orthogonal matrix $(Q^TQ = I)$. (See Golub (1965), Lawson and Hanson (1974)).

The SVD or singular value decomposition: The decomposition $A = UDV^T$ where U and V are orthogonal matrices and D is a nonnegative diagonal matrix, is very useful. The rank of A is the same as that of D. The SVD is related to the polar decomposition $A = (UV^T)(VDV^T)$. (See Forsythe and Moler (1967), Golub and Reinsch (1970)).

3. EXISTING METHODS FOR UNCONSTRAINED PROBLEMS.

A large number of algorithms have been suggested for the solution of unconstrained optimization problems, and it is not our intention to describe them all. Instead we attempt to state some of the most common techniques that are currently in use. The procedures described are of course the ones related to the algorithms developed in this thesis. The exclusion of a particular approach from our discussion does not imply that in our opinion it is either unreliable or inefficient. A more complete survey of the field may be found in texts like Avriel (1975), Himmelblau (1972), Luenberger (1973), Murray (1972), and in survey papers e.g., Dennis and Moré (1977), Dixon (1974), Powell (1971)(1976), in addition to the references cited below.

Following the observation made in the previous section (Theorems 2.1 and 2.2), let us consider an operator $F: R^n \to R^n$ and the problem of finding a point $x \in R^n$ such that

$$(3.1) \qquad F(x) = 0 \ .$$

By a quasi-Newton method for problem (3.1) we mean the iterative procedure

$$(3.2) \qquad \bar{x} = x - B^{-1}F(x)$$

$$(3.3) \qquad \bar{B} = \mathcal{B}(x,\bar{x},B)$$

where $\mathcal{B}(x,\bar{x},B)$ is in some sense an approximation to the Jacobian matrix $JF(x^*)$. Note that, for convenience, we have suppressed the iteration counter and denoted the quantities corresponding to the successive iteration by placing a bar over them.

As special cases of quasi-Newton methods we have

<u>Newton's method</u>:

(3.4) $$\mathcal{B}(x,\bar{x},B) = JF(x)$$

<u>Discrete Newton method</u>:

(3.5) $$\mathcal{B}(x,\bar{x},B) = \left(\frac{1}{t_i}\left[F_i(\bar{x}+t_i e_j) - F_i(\bar{x})\right]\right)$$

where F_i, $i=1,\ldots,n$ represents the ith component of F and t_i, $i=1,\ldots,n$ is a small positive scalar (ideally close to $\sqrt{\tau}$).

<u>Secant methods</u>:

(3.6) $$\mathcal{B}(x,\bar{x},B) = \mathfrak{B}(s,y,B)$$

where $s=\bar{x}-x$, $y=F(\bar{x})-F(x)$ and $\mathfrak{B}$ satisfies the <u>secant</u> equation

(3.7) $$\mathfrak{B}(s,y,B)\cdot s = y\ .$$

Observe that for the case $n=1$, (3.7) completely determines $\bar{B}$ and (3.2) becomes the well-known secant iteration in one dimension. Nevertheless, other names have been used for secant methods in the literature; viz, quasi-Newton, variable metric, modification methods, etc.

<u>Remark 3.1</u>.

We must emphasize that quasi-Newton methods are seldom implemented as in (3.2) - (3.3). Instead, (3.2) is replaced by

$$\bar{x} = x - \alpha B^{-1} F(x)$$

where the step-length parameter α is determined so that an improvement over the current estimate for the solution results. Improvement is measured by means of a scalar valued function ψ, called the merit function, which has a local minimum at the solution and is convex in a neighborhood of the solution. A standard requirement is that the descent condition

$$(3.8) \qquad \psi(x - \alpha B^{-1} F(x)) < \psi(x)$$

be satisfied at each step. In this context, a direction p is said to be a descent direction for ψ at x if

$$(3.9) \qquad \nabla \psi(x)^T p < 0 \ .$$

We now turn to another important class of algorithms, called the conjugate gradient methods. Perhaps the most general class of methods for unconstrained optimization is the one called conjugate direction methods. However, it must be stressed that (as will be obvious below) the conjugate direction method designates a class of algorithms. It is interesting to point out that in the special case of a strictly convex quadratic function, both the secant and the conjugate gradient methods turn out to belong to this class. In fact, more is true than just that, but we will leave this discussion until a later section. We first present below an outline of conjugate gradient methods, and then consider some of the specific forms assumed by secant methods in practice. Further knowledge may be gained from Fletcher (1970a,b), Gill and Murray (1972). Several efficient conjugate direction algorithms are proposed in Davidon (1975), Gill, Murray and Pitfield (1972), Greenstadt (1978), Lenard (1978), Nazareth (1977a,c), Shanno and Phua (1976)(1978b).

For the unconstrained minimization of f in (2.1), a local quadratic approximation to f at a point x is given by the truncanted Taylor's series

$$(3.10) \qquad f(x+\Delta x) \approx f(x) + \nabla f(x)^T \Delta x + \tfrac{1}{2} \Delta x^T \nabla^2 f(x) \Delta x \ .$$

Minimizing this quadratic form leads one to the solution of the linear system

$$(3.11) \qquad \nabla^2 f(x) \cdot \Delta x = -f(x)$$

which is simply Newton's method for problem (2.1). With this in mind, consider the quadratic problem

$$(3.12) \qquad \text{minimize } q(x) = \tfrac{1}{2}\langle Ax,x\rangle - \langle b,x\rangle + c \ ; \quad x \in \mathbb{R}^n$$

where $b \in \mathbb{R}^n$, $c \in \mathbb{R}$ are fixed constants and $A \in \mathbb{R}^{n \times n}$ is a symmetric matrix. The matrix A is assumed to be positive definite unless otherwise stated.

The basic conjugate direction algorithm for the problem (3.12) is defined as (Luenberger (1973))

$$(3.13) \qquad \alpha_k = -\langle g_k, d_k\rangle / \langle Ad_k, d_k\rangle$$

$$(3.14) \qquad x^{k+1} = x^k + \alpha_k d_k$$

where $x^0 \in \mathbb{R}^n$ is an initial guess for the minimizer x^* of q,

$\left\{d_j\right\}_{j=0}^{n-1}$ is a sequence of non-zero A-conjugate vectors in R^n, and g_k represents the gradient vector $\nabla q(x^k)$ of q at x^k, i.e., $g_k = Ax^k - b$. Note that since A is positive definite, x^* is given by $x^* = A^{-1}b$.

Remark 3.2.

Observe that an induction argument using

$$\langle g_{k+1}, d_j \rangle = \langle g_k, d_j \rangle + \alpha_k \langle Ad_k, d_j \rangle, \quad j \leq k$$

and (3.13) shows that g_{k+1} is orthogonal to the linear span $[d_1, d_2, \ldots, d_k]$. Further, since q is strictly convex, it is evident that x^{k+1} minimizes q in the hyperplane $x^0 + [d_1, d_2, \ldots, d_k]$; and thus the algorithm converges to x^* in at most n iterations.

Conjugate gradient (CG) methods sequentially generate the required directions. A useful generalization of CG methods as suggested by Concus, Golub and O'Leary (1976) is as follows:

Given $x^0 \in R^n$ as an initial estimate of the solution x^* for problem (3.12), and $D, H \in R^{n \times n}$ any symmetric positive definite matrices; let

$$d_0 = -Hg_0$$

and for $k \geq 0$, define

$$\alpha_k = -\langle DAd_k, g_k \rangle / \langle Ad_k, DAd_k \rangle$$

$$x^{k+1} = x^k + \alpha_k d_k$$

$$\beta_k = \langle Ad_k, Hg_{k+1} \rangle / \langle Ad_k, d_k \rangle$$

$$d_{k+1} = -Hg_{k+1} + \beta_k d_k$$

With the particular choice $D = A^{-1}$ and $H = I$, the above algorithm reduces to the basic CG method originally introduced by Hestenes and Stiefel (1952), Hestenes (1956) as a means of solving linear systems and later extended to nonlinear optimization by Fletcher and Reeves (1964).

In the case $D = A^{-1}$ and H is an arbitrary symmetric positive definite matrix; the above generalized conjugate gradient algorithm gives rise to the so-called preconditioned conjugate gradient method (PCGM) which was developed by Axelsson (1974)(1975) to solve large sparse linear systems involved in the solution of various partial differential equations. We will make extensive use of PCGM in the sequel, so let us rewrite it as:

Define

(3.15) $$d_0 = -Hg_0$$

and for successive iterates

(3.16) $$\alpha_k = -\langle g_k, d_k\rangle / \langle Ad_k, d_k\rangle$$

(3.17) $$x_{k+1} = x_k + \alpha_k d_k$$

(3.18) $$\beta_k = \langle Ad_k, Hg_{k+1}\rangle / \langle Ad_k, d_k\rangle$$

(3.19) $$d_{k+1} = -Hg_{k+1} + \beta_k d_k \ .$$

Douglas and Dupont (1976) used the above algorithm in Galerkin methods for solving nonlinear Dirichlet problems. Several interesting implementations of conjugate gradient methods have been reported by Nazareth (1977a) and Shanno (1978a,b).

It should be obvious that for problem (2.1), f is a natural choice for the merit function discussed earlier in this section. Similarly, the function q is a natural merit function for problem (3.12). It is now easy to see that the directions $(d_k, k \geq 0)$ generated by the generalized conjugate gradient algorithm are descent directions on q in the sense of (3.9), and that the PCGM also inherits this property.

In the case of quasi-Newton methods for problem (2.1), it is readily seen from (3.2), taking $F(x) = \nabla f(x)$ in (3.1), that the direction $p = -B^{-1}\nabla f(x)$ is a descent direction on f if B is positive definite. Thus the classical steepest descent method $(B = I)$ of Cauchy (1847) makes sense. Also, in view of Theorem 2.1, Newton's method may be expected to generate descent directions at least locally. It is interesting to note that, according to our terminology, both these methods for unconstrained optimization are quasi-Newton methods which are not secant methods (see (3.7)).

For secant methods Davidon (1959), together with some important modifications made by Fletcher and Powell (1963), suggested an update formula (DFP) for the approximation matrix B (see (3.3) and (3.6)) so that the successive approximation $\bar{B}$ inherits symmetry and positive definiteness from B while satisfying the secant equation (3.7). Later, Broyden (1965)(1967)(1969)(1970) introduced a family of such update formulas. Today, a member of this family is usually used for updating the approximation matrix B in (3.3).

The Broyden's class of updates may be given by (assuming $\langle y,s\rangle > 0$ for positive definiteness)

(3.20)
$$\bar{B} = B + \frac{yy^T}{\langle y,s\rangle} - \frac{Bss^TB}{\langle Bs,s\rangle} + \Phi ww^T \; ; \quad \text{where}$$

$$w = \langle Bs,s\rangle^{\frac{1}{2}} \left[\frac{y}{\langle y,s\rangle} - \frac{Bs}{\langle Bs,s\rangle}\right] ,$$

$s = \bar{x} - x$, $y = \nabla f(\bar{x}) - \nabla f(x)$, and $\Phi \geq 0$ may depend on s,y,B .

In particular, $\Phi = 1$ gives the Davidon-Fletcher-Powell (DFP) update formula

(3.21)
$$\bar{B} = B + \frac{(y - Bs)y^T + y(y - Bs)^T}{\langle y,s\rangle} - \frac{\langle y - Bs,s\rangle}{\langle y,s\rangle^2} yy^T \; ;$$

whereas $\Phi = 0$ yields the well-known Broyden-Fletcher-Goldfarb-Shanno (BFGS) update (Broyden (1969),(1970), Fletcher (1970a), Goldfarb (1970), Shanno (1970))

(3.22)
$$\bar{B} = B + \frac{yy^T}{\langle y,s\rangle} - \frac{Bss^TB}{\langle Bs,s\rangle} \; .$$

Note that $O(n^3)$ arithmetic operations are needed to compute $p = -B^{-1}\nabla f(x)$. Toward this end, numerous methods have been suggested to update the Cholesky factors of B thus requiring only $O(n^2)$ operations to calculate p (Fletcher and Powell (1974), Gill, Golub, Murray and Saunders (1974), Gill and Murray (1972), Goldfarb (1976) (1977)). We describe one such technique (given by Dennis (1978b)) in the appendix. Another approach is to approximate the inverse Hessian $\nabla^2 f(x)^{-1}$ by a matrix, say H, and update H instead of B. Inverse updating formulas for the Broyden's class, the DFP and the BFGS updates may be obtained from (3.20), (3.22) and (3.21) respectively, with B, s and y replaced by H, y and s respectively.

To insure descent in secant methods, generally one of the following choices for α (see (3.8)) is suggested

$$\alpha = \arg\ (\underset{\hat{\alpha}>0}{\text{minimize}}\ f(x+\hat{\alpha}p)) \qquad \text{(Cauchy (1847))}$$

or

$$\alpha = \min\ \{\hat{\alpha}>0:\langle\nabla f(x+\hat{\alpha}p),p\rangle = 0\} \qquad \text{(Curry (1944))} \tag{3.23}$$

or

$$\alpha = \min\left\{\begin{array}{l}\hat{\alpha}>0: f(x+\hat{\alpha}p)\leq f(x+\tilde{\alpha}p);\ \ |\tilde{\alpha}-\hat{\alpha}|\leq\epsilon \\ \qquad \text{for small values of}\ \ \epsilon>0\end{array}\right\}$$

As may be realized either choice is unrealistic in practice. However, in the case of quadratic problem (3.12), it is trivial to calculate the unique closed form solution from any of the aforementioned choices as

$$\alpha = -\langle g,p\rangle/\langle Ap,p\rangle\ . \tag{3.24}$$

For general nonlinear function f (problem (2.1)) steplength algorithms have been devised which constitute sophisticated termination criteria for conventional algorithms designed for simple descent or approximate minimization. The main idea is to predict the function decrease that can be expected along the given search direction and stop only when some specified fraction of this decrease has been realized. This work originated with Goldstein (1962) and has been further developed by Armijö (1966), Goldstein and Price (1967), Wolfe (1969)(1971), among others. Important aspects of practical implementation are considered in Gill and Murray (1974b). Two other reliable techniques which may be used for the purpose are given in Bell (1978) and Fox, Lasdon, Tamir and Ratner (1975).

We are now in a position to write the precise secant algorithm for unconstrained optimization. For the sake of definiteness we outline the secant method for problem (3.12) using the inverse Hessian approximation H and steplength given by (3.24) as follows:

Given $x^0 \in R^n$ as an initial estimate for x^* and a symmetric positive definite matrix $H_0 \in R^{n \times n}$, for $k \geq 0$, define

(3.25) $$p_k = -H_k \nabla q(x^k) = -H_k g_k$$

(3.26) $$s_k = (-\langle g_k, p_k \rangle / \langle Ap_k, p_k \rangle) \cdot p_k$$

(3.27) $$x^{k+1} = x^k + s_k$$

(3.28) $$y_{k+1} = \nabla q(x^{k+1}) - \nabla q(x^k) = g_{k+1} - g_k$$

(3.29) $$H_{k+1} = H_k + \frac{s_k s_k^T}{\langle y_k, s_k \rangle} - \frac{H_k y_k y_k^T H_k}{\langle y_k, H_k y_k \rangle} + \tilde{\Phi} v_k v_k^T \;;$$

where $$v_k = \langle y_k, H_k y_k \rangle^{\frac{1}{2}} \left[\frac{s_k}{\langle y_k, s_k \rangle} - \frac{H_k y_k}{\langle y_k, H_k y_k \rangle} \right]$$

and $\tilde{\Phi} \geq 0$ may depend on s_k, y_k, H_k .

Before closing this section, some alternative strategies need be mentioned. Several hybrid-type algorithms which improve the robustness of the straightforward quasi-Newton methods have been proposed (see Heath (1978)). Goldstein and Price (1967) suggested the use of the quasi-Newton direction at each iteration if possible, but if trouble is encountered (indefinite matrix, failure to attain descent, etc.), then to switch to the negative gradient direction for the line search. Gleyzal (1959) had the interesting idea of searching simultaneously along both the quasi-Newton and the gradient directions, but he made no indication as to how such a two-dimensional search might be effectively carried out. A more tractable method along these lines was considered by Levenberg (1944) and Marquardt (1963) and later by Goldfeld, Quandt and Trotter (1966), who suggested shifting the entire spectrum of the Hessian by a positive constant -- only to discover later that an appropriate value for this constant is difficult to obtain. A way around this difficulty was proposed by Powell (1970) with his so-called "dogleg" algorithm. Powell, in his algorithm, avoided the use of line searches and instead defined a <u>region of trust</u>, typically a ball about the current estimate for the solution in which the linearization may be considered adequate and therefore the quasi-Newton step may be employed. A larger value of the radius r of the current trust region biases the step toward the quasi-Newton direction, while a smaller value of r biases the step toward the negative gradient direction. This idea has been further developed by Dennis and Mei (1975) in their so-called "double-dogleg" method, which introduces an early bias toward quasi-Newton direction -- thus increasing the efficiency of the algorithm in the vicinity of a solution.

If at the current iterate x the Hessian $\nabla^2 f(x)$ has a negative eigenvalue, then the corresponding eigenvector, say u, satisfies $\langle \nabla^2 f(x)u,u\rangle < 0$ and the sign of u can be chosen so that $\langle \nabla f(x),u\rangle \leq 0$. Any vector u having these two properties is called a <u>direction of negative curvature</u>, and the above remark shows that at least one such vector exists whenever $\nabla^2 f$ has a negative eigenvalue. Geometrically, along this direction the current point x is on a hill rather than in a valley. Therefore, aside from using quasi-Newton and negative gradient directions, another possibility is to descend along a direction of negative curvature whenever the Hessian matrix is not positive definite. Fiacco and McCormick (1968, pp. 166-167) proposed this strategy in the indefinite case but could only suggest a costly eigenvalue-eigenvector decomposition as a means of computing a suitable direction of negative curvature, so it appeared to be a theoretically interesting but impractical idea. Recently, there has been a lot of research activity in this area. Fletcher and Freeman (1975) suggested symmetric indefinite block-diagonal factorization of $\nabla^2 f(x)$ at each iteration in order to obtain either the quasi-Newton step or a direction of negative curvature, depending on whether the factorization shows $\nabla^2 f(x)$ to be positive definite. Another algorithm using directions of negative curvature was proposed by McCormick (1977) and developed by Moré and Sorenson (1979). In this approach if the Hessian has any negative eigenvalues, then a direction of negative curvature is coupled with an ordinary descent direction (either the negative gradient or the quasi-Newton step), and a line search is performed along a parameterized curve in the plane defined by the two directions. Moré and Sorensen recommend systematic use of the symmetric indefinite block-diagonal factorization (Bunch and Parlett (1971)) to determine if the Hessian is positive definite, perturb it if necessary, solve for the quasi-Newton step, and compute the negative curvature direction. Heath (1978) presented an algorithm which is a synthesis of most of the ideas discussed thus far.

4. EXISTING METHODS FOR CONSTRAINED PROBLEMS

We now consider the constrained optimization problems. As noted earlier, we will restrict our discussion to equality constrained problems. The major practical difficulty with inequalities is that it is not known at the outset which constraints are active (h_i is *active* at x if $h_i(x) = 0$) at the solution. We review some of the methods which are related in various ways to the algorithm developed for problems with equality constraints. Once again, this survey is not meant to be an exhaustive compilation of methods, nor is the discussion of those methods which are included more than a sketch. The purpose is to acknowledge the influence of previous research in the field and to place the ideas expressed in this thesis in the perspective of existing algorithms. A detailed survey of the methods in this area is given in books such as Avriel (1975), Fiacco and McCormick (1968), Gill and Murray (1974a), Mangasarian (1969), Zangwill (1969), and in survey papers, e.g., Dixon (1975), Fletcher (1977), Murray (1976), Powell (1978a). Equivalence between some of the more commonly used methods for equality constrained problems was established by Tapia (1978).

Methods for minimizing a function subject to nonlinear constraints can be divided broadly into two classes -- those which set up an equivalent unconstrained minimization problem by adding a penalty term to either the objective function (see (2.4)) or the Lagrangian function (see (2.5) and (2.6)), and those which seek to generate a sequence of feasible-descent steps (e.g., Reduced-gradient and Projected-gradient methods). Since our objective is to develop an algorithm which is closely related to one for the unconstrained case, we resort to the former. Regarding methods in the latter, we simply list some of their original references and leave them out of any further discussion.

The projected gradient method is due to Rosen (1960)(1961). The reduced gradient method for linearly constrained problems was introduced by Wolfe (1963), and generalized to include nonlinear constraints by Abadie and Carpentier (1969). Computer implementation of the generalized reduced gradient (GRG) algorithm is mainly due to Lasdon et al (1978). Another method of this type is the gradient-restoration algorithm of Miele et al (1969).

Perhaps the earliest approach to the constrained optimization problem is that of using penalty functions -- originally suggested by Courant (1943) and explored in detail by Fiacco and McCormick (1968). For problem (2.2) a penalty term is added to the objective function, and the resulting function (2.4) is minimized for a decreasing sequence of values of <u>penalty parameter</u> r . Fiacco and McCormick established that there exists some $\hat{r} > 0$ such that for all $0 \leq r < \hat{r}$ a minimizer of the penalty function $P(x,r)$ exists, and if we let $x(r)$ denote this minimizer, then

$$\lim_{r \to 0} x(r) = x^*$$

where x^* is the anticipated solution to problem (2.2). Although this method is robust and has been used successfully in many practical applications, it has some severe disadvantages. First of all, it involves a sequence of exact unconstrained minimizations which can be very expensive. Second, the method may get into serious problems if the initial estimate of the penalty constant r is taken to be larger than $\hat{r}$ -- which is undeterminable (at least in practice). This is illustrated through an example by Dixon (1975). Third, and the most serious drawback with penalty methods, is that as $r \to 0$ the Hessian $\nabla_x^2 P(x,r)$ of the penalty function becomes increasingly ill-conditioned. This feature is imposed by the transformation and is unavoidable, thus limiting the extent to which modifications to the algorithm may alleviate the computational difficulties which exist.

Hestenes (1969), and independently, Powell (1969), proposed adding the penalty term to the classical Lagrangian function (2.5) to get the augmented Lagrangian (2.6), and then to use a "penalty-type" method on this latter function. In order to describe their method and for other approaches to come, let us introduce some definitions following Tapia (1977).

Any nonnegative function $\Pi: R^{n+m+1} \to R$ is called a <u>penalty constant update formula</u> . An operator $U: R^{n+m+1} \to R^m$ is said to be a <u>multiplier update formula</u> if

$$\lambda^* = U(x^*, \lambda^*, c), \quad c \geq 0$$

whenever (x^*, λ^*) is a critical point of problem (2.2). Further, if U is independent of λ (explicitly, i.e., $\nabla_\lambda U(x,\lambda,c) = 0$) we qualify U as a <u>multiplier approximation formula</u>. If (x^*, λ^*) is a critical point of problem (2.2) then x^* is said to be a <u>nonsingular critical point</u> of problem (2.2) if $\nabla^2 \ell(x^*,\lambda^*)$ is invertible. Note that nonsingularity implies regularity.

The basic multiplier method of Hestenes and Powell may now be written as the iterative procedure:

given λ^0 and $c^0 > 0$; repeat the following sequence until the convergence tolerance is met.

- compute $\bar{x}$ such that $\mathcal{L}(\bar{x},\lambda,c) = \min\limits_{x} \mathcal{L}(x,\lambda,c)$
- set $\bar{c} = \Pi(\bar{x},\lambda,c)$
- and $\bar{\lambda} = U(\bar{x},\lambda,c)$.

As with the penalty method, the exact minimization required to update x could be expensive. Note that the Hessian of $\mathcal{L}(x,\lambda,c)$ is given by

$$\text{(4.1)} \quad \nabla^2\mathcal{L}(x,\lambda,c) = \begin{bmatrix} \nabla^2 f(x) + c\nabla h(x)\nabla h(x)^T + \sum\limits_{i=1}^{m} (\lambda_i + ch_i(x))\nabla^2 h_i(x) & \nabla h(x) \\ \nabla h(x)^T & 0 \end{bmatrix}$$

Following Hestenes (1969) (also see Buys (1972)), it may be shown that if (x^*,λ^*) is a solution of problem (2.2) then there exists $\hat{c} > 0$ such that for all $c \geq \hat{c}$, the Hessian $\nabla_x^2\mathcal{L}(x^*,\lambda^*,c)$ is positive definite. But this result makes use of optimality conditions, and so, far away from a solution, a value of c which makes $\nabla_x^2\mathcal{L}(x,\lambda,c)$ positive definite may not exist. The choice of c is rather delicate because too large a value can cause the Hessian to be ill-conditioned, while too small a value can cause it to be indefinite or nearly singular. In fact, Tapia (1977) established that the multiplier method is Q-superlinearly convergent if and only if $c \uparrow \infty$.

While most practical augmented Lagrangian algorithms incorporate heuristic rules for adaptively refining the penalty constant, much of the research has been concerned with efficient update formulas for the Lagrange multipliers. The following list of multiplier update formulas is compiled from Tapia (1978). (We leave out the arguments of a function when they are obvious.)

(4.2) $U(x,\lambda,c) = \lambda + ch$ Hestenes (1969), Powell (1969)

(4.3) $U(x,\lambda,c) = -(\nabla h^T \nabla h)^{-1} \nabla h^T \nabla f$ Rosen (1960)

(4.4) $U(x,\lambda,c) = (\nabla h^T \nabla h)^{-1} (h - \nabla h^T \nabla f)$ Miele et al (1971)

(4.5) $U(x,\lambda,c) = \lambda + (\nabla h^T \nabla_x^2 \mathcal{L}^{-1} \nabla h)^{-1} h$ Buys (1972)

(4.6) $U(x,\lambda,c) = (\nabla h^T D \nabla h)^{-1} (h - \nabla h^T D \nabla f) - ch$ Tapia (1974a)

(4.7) $U(x,\lambda,c) = \lambda + [\nabla h^T D \nabla h + A]^{-1} [h - \nabla h^T D \nabla_x \mathcal{L}]$ Tapia (1977)

where in (4.6) and (4.7) $D \in R^{n \times n}$ and $A \in R^{m \times m}$ may depend on x, λ, and c. A comprehensive discussion of these formulas and their interrelationships is presented by Tapia (1977).

In view of the computational expense of repeated unconstrained minimizations involved in the multiplier method, Tapia (1977) suggested an attractive variant of this approach where the unconstrained minimizations are carried out only as far as one step (however, several steps may optionally be taken), i.e., λ is updated after each step of an iterative method for minimizing the augmented Lagrangian. In this approach, if Newton's method is used for the unconstrained minimization step and formula (4.6) for the multiplier update (with $D = \nabla_x^2 \mathcal{L}^{-1}$), then the algorithm yields essentially Newton's method for solving the extended system (2.7), i.e., the iterative procedure involving the solution of the linear system

$$(4.8) \quad \begin{bmatrix} B & \nabla h \\ \nabla h^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L} \\ h \end{bmatrix}$$

where B represents $\nabla_x^2 \mathcal{L}^{-1}$ in Newton's method (but may be an approximation there of, in secant methods), giving the new estimate of the solution as $(x + \Delta x,\ \lambda + \Delta\lambda)$. It must be carefully noted, however, that the extended system is not solved in Tapia's method -- which may be outlined as follows:

> given x^0, λ^0, c^λ and B_0, repeat until the convergence tolerance is met.
>
> - $\bar{\lambda} = U(x, \lambda, c)$
> - $\bar{x} = x - B^{-1} \nabla_x \mathcal{L}(x, \bar{\lambda}, c)$
> - $\bar{c} = \Pi(\bar{x}, \bar{\lambda}, c)$
> - $\bar{B} = \mathcal{B}(x, \bar{x}, \lambda, \bar{\lambda}, B)$

where $\mathcal{B}(x, \bar{x}, \lambda, \bar{\lambda}, B)$ is an approximation to $\nabla_x^2 \mathcal{L}(x^*, \lambda^*, c)$. Several advantages of this procedure are observed by Tapia, (also by Byrd (1978)), including the fact that the penalty constant need not go to infinity in order to assure local Q-superlinear convergence. In fact, in most practical situations, $c = 0$ is the optimal choice near a solution (see Bertocchi et al (1978)).

An alternative way of implementing Newton's method for solving the nonlinear system (2.7), given by the first order necessary conditions for problem (2.2), is to think of it as forming and solving a succession of linearized subproblems. Toward this end, using a quadratic approximation to the augmented Lagrangian function and a linear approximation to the constraint functions, we get the quadratic programming problem

$$(4.9)\qquad \begin{aligned} &\min_{\Delta x} \mathcal{L}(x,\lambda,c) + \langle \nabla_x \mathcal{L}(x,\lambda,c), \Delta x\rangle + \tfrac{1}{2}\langle B\Delta x, \Delta x\rangle \\ &\text{subject to} \qquad \nabla h(x)^T \Delta x + h(x) = 0 \end{aligned}$$

where B is an approximation to $\nabla_x^2 \mathcal{L}(x,\lambda,c)$. Notice, however, that the solution of (4.9) is given by the linear system (4.8), which also yields the Lagrange multiplier $\Delta\lambda$ for (4.9).

The use of successive quadratic programming subproblems for solving nonlinearly constrained optimization problems is a part of the optimization folklore and was described by Wilson (1963), and has since been treated by Murray (1969), Gill and Murray (1974a), Han (1977a,b), Palomares and Mangasarian (1976), Powell (1977a,b)(1978a,b). Biggs (1972)(1975)(1978) also uses a successive quadratic programming approach, though following a different derivation -- based on the penalty function (2.4). It is interesting to comment that his method turns out to be equivalent to Tapia's (1977) approach with update formula (4.6) ($D = B^{-1}$ and $c = 0$), at least close to the solution.

The inclusion of inequality constraints in the recursive quadratic programming algorithm is obvious. However, Biggs (1976) observed that his recursive quadratic programming method may be efficient starting close to the solution, but the Lagrangian methods display better behavior away from the solution (also Han (1978)). Biggs also noticed that the methods using augmented Lagrangian functions are more suitable for large problems. Similar comments may be concluded from the numerical study conducted by Schittkowski (1978). Tapia (1978) established equivalence between several of these seemingly different approaches.

Another possible linearization is to linearize the constraints but leave the objective function alone, so that the resulting subproblem is a general linearly constrained nonlinear minimization problem. This approach is taken by Rosen and Kreuser (1972), Robinson (1972) and Rosen (1978), a local convergence analysis is given by Robinson (1974).

One major drawback of the last two approaches is that the performance of any computer implementation of these methods is entirely based upon the particular code employed to solve the subproblem involved. This is certainly true for the Rosen-Kreuser method, and also for the quadratic programming subproblem in the presence of inequality constraints. Besides, as with all Newton-based methods, algorithms resulting from simple linearizations cannot be relied upon to converge when started from a poor initial estimate for a solution.

There have been some other interesting develooments, one of which is the search for an "exact penalty function" -- a function whose unconstrained minimizer also solves the constrained optimization problem (Fletcher (1973), Han and Mangasarian (1978)). Some functions with this property are readily suggested, but they are not smooth at the minimizer, which renders them unsuitable for use with established efficient unconstrained minimization techniques.

An algorithm which features a two-part line search, and is based on the explicit recognition of the saddlepoint nature of solutions (x^*,λ^*) to problem (2.2), is due to Bard and Greenstadt (1969). It is interesting to find that their algorithm implicitly uses a special case of (4.6) for the multiplier update formula. The Bard-Greenstadt algorithm offers separate line searches, one on Δx and the other on $\Delta\lambda$, with the clear-cut objective of attaining descent and ascent, respectively. Nevertheless, one main drawback is that these line searches are conducted on the Lagrangian function, which may be unbounded at times (especially when away from the solution).

Heath (1978) also proposed an algorithm using a two-part line search -- one on the range space of the gradients of (active) constraints and the other on the corresponding null space $T(x)$. This approach bears a strong resemblance to the two distinct phases of the projected gradient method. An important difference, however, is that Heath uses a Newton step in the null-space search rather than the negative gradient direction as is done in the projected gradient approach. The effect of this, along with some other differences, is that his algorithm gives Newton-type behavior near a solution and is superlinearly convergent, while the projected gradient algorithm has only linear asymptotic convergence -- at least in its standard formulation.

In general, most of the foregoing approaches have the usual advantages and disadvantages expected of Newton's method, such as having rapid -- but rather localized -- convergence. The most serious unreliability of Newton's method, in the context of constrained optimization, comes from the fact that it is not clear if a solution of (2.7) is a constrained minimizer of problem (2.2), since (2.7) also holds at a constrained maxima or at a constrained saddle point of $f(x)$. (Note that a similar observation may be made for the unconstrained case.)

A consensus seems to be growing -- the use of successive quadratic programming subproblems, simultaneous updating of x and λ in augmented Lagrangian methods, the use of second order terms in projected gradient algorithms, etc. -- which indicates an ever increasing rate of quasi-Newton methods in constrained optimization, with the result that all these algorithms take on a uniform appearance. For this reason, the algorithms we develop in this thesis are based directly on Newton's method, thereby allowing the problem of robustness to be tackled in the open. This approach also simplifies the convergence analysis of other quasi-Newton methods since it make possible the direct application of general theory (Byrd (1978), Dennis and Moré (1977), Glad (1976), Han (1976), Powell (1978b), Robinson (1974), Tapia (1977)).

5. MOTIVATION FOR THE NEW ALGORITHMS.

The similarities among the methods (within each section) discussed earlier are no accident. As Fletcher (1977) observed, the borrowing of ideas which work well in one method for use in another has led to a blurring of the usual distinctions between the various classes of methods. In fact, all the methods are, in some sense, a variant of Newton's method on an appropriate equivalent problem. Perhaps for this reason all the methods depend on the positive definiteness of the Hessian matrix ($\nabla^2 f$ in the unconstrained case, $\nabla_x^2 \ell$ in the space $T(x)$ in the constrained case). Newton's method is locally Q-quadratically convergent, yet unfortunately this rate of convergence is not realized globally; moreover, the method must be modified to insure convergence -- simply because the Hessian matrix is not positive definite in certain regions along the path of the algorithm. If the Hessian is indefinite at a point away from a solution, it may be reasonable to replace it with a positive-definite matrix, as is done in secant methods, and then proceed as usual. However, for problems in which the Hessian is not positive definite at a solution such a practice is open to question and may not, in general, result in Newton-like asymptotic behavior. In fact, in some such cases, even the convergence could be doubtful.

In view of the above we propose to use directions of negative curvature when one appears and otherwise stay on the subspace where the Hessian is positive definite.

Most state of the art algorithms in unconstrained optimization employ secant methods with some form of inexact line search, as discussed in section 3. The basic motivation behind secant methods is to try to obtain the rapid convergence associated with Newton's method without explicitly evaluating the Hessian at each step. This is accomplished by constructing approximations to the Hessian based on the information gathered during the iteration process. Newton's method does not guarantee descent at each step, whereas secant methods do. This is due to the fact that the Hessian approximation is always kept positive definite which leads to a descent direction at each step. Although only local convergence has been established for secant methods (Dennis and Moré (1977)), these methods can be made to exhibit good global convergence properties by using line searches for step length control.

This suggests the natural question: "Why does an implementation of a secant method with no step length control exhibit disastrous behavior while outside the domain of local convergence of the basic algorithm." Recall the secant equation (3.7), which can be written as

$$(5.1) \qquad \bar{B}s = y$$

where, as usual, the bar notation is used to denote quantities at a subsequent iteration. As mentioned earlier in section 3, this equation forms the basis of all secant methods; and it is trivial to verify that the Broyden's class of rank-2 updates (3.20) satisfies it. If there is no step-length control $(\alpha = 1)$, from (5.1) we have for the unconstrained problem

$$(5.2) \qquad \bar{B}^{-1}(\nabla f(x+p) - \nabla f(x)) = p$$

where p is the secant direction. If p is not small but $\nabla f(x+p) \approx \nabla f(x)$ (which is often possible in nonlinear problems), then $\bar{B}^{-1}$ is necessarily large. However, with the step-length control, we may write (5.1) as

$$\bar{B}^{-1} \frac{\nabla f(x+\alpha p) - \nabla f(x)}{\alpha} = p \ . \tag{5.3}$$

Thus, when α is small -- which is to be expected in secant methods when far away from the solution -- we see that the secant equation (5.1) is approximately

$$\bar{B}^{-1} \nabla^2 f(x) p = p \ . \tag{5.4}$$

The above motivates the use of short steps in secant update formulas, at least far away from the solution. Using short steps we can expect to obtain better approximations to the Hessian away from the solution.

Several scaling techniques (Oren (1973), Oren and Spedicato (1976), Shanno and Phua (1978a), Spedicato (1976)(1978)) have been suggested to obtain a better conditioning of the approximate Hessian matrices B_k . In most of these techniques a scale factor is incorporated directly into the update formula.

It is also felt from the above remarks that it can be fruitful to follow the quadratic model problem (3.10) whenever this is possible. If nothing else, at least that is the motive behind Newton's method. In fact, most methods are invariably invented and analyzed for the pure quadratic problem; and once the techniques work out for this problem they are then extended to more general problems via quadratic approximations. It may be argued that, since near the solution every problem is approximately quadratic, convergence behavior can be expected to be similar to that for the pure quadratic case.

In order to handle large problems we require our algorithms to be able to work with limited storage. With this feature -- the so-called variable storage capability -- in our algorithms, we expect to use full storage whenever it is available, and still solve the problem when full storage is not available.

Some of the above objectives have simply been explained in terms of unconstrained optimization problems just for convenience. It is obvious that they have counter parts in constrained optimization problems (in terms of augmented Lagrangian function). At this point, it is perhaps valuable to summarize some of the main themes behind our motivating philosophy for the optimization algorithms developed.

1. To be competitive the algorithms should be at least locally Q-superlinearly convergent.
2. The algorithms should be able to converge to the desired solution without requiring a close estimate of the solution (robustness).
3. The algorithm should be able to stay clear of, or descend from, saddlepoints and maxima (use of directions of negative curvature).
4. The computational expense involved in using directions of negative curvature should be minimized.
5. We should use short steps in update formulas for Hessian approximation.
6. The algorithms should be able to incorporate variable storage capability in order to handle large problems.

7. The performance of algorithms should not be critically dependent on a judicious choice of parameters such as penalty constants, tolerances, etc., and "fine tuning" should not be required for individual problems.

Some other design goals as desirable of any algorithm for optimization problems include natural use of optimality conditions. This is important not only in guiding the algorithm in seeking a solution but also in verifying that a point to which it has converged is in fact a solution to the problem. Also, the numerical linear algebra techniques involved in an optimization algorithm should be both stable and efficient. For example, factorizations should be updated, if possible, rather than recomputed when the matrix changes. Finally, as usual, although higher derivatives of problem functions may be assumed to exist for theoretical analysis, practical algorithms should require only first derivatives at most, to be analytically defined, and second derivatives, if needed, should be approximated.

Of course, it is unlikely to fully attain all these objectives in one single algorithm. Nevertheless, they serve as a useful guide in choosing among various alternatives in algorithm design and convey the spirit of our approach to the subject (Heath (1978)).

6. A UNIFIED THEORY FOR THE NEW ALGORITHMS.

The algorithms developed in this thesis can be regarded as being somewhat intermediate between the method of steepest descent and Newton's method.

First consider the quadratic problem (3.12) with Hessian matrix A assumed to be positive definite, and note the following properties of the preconditioned conjugate gradient method and the secant methods in this context.

LEMMA 6.1.

If the preconditioned conjugate gradient method (3.15) - (3.19) is applied to problem (3.12) starting at any $x^0 \in R^n$ with an arbitrary symmetric positive definite matrix $H \in R^{n \times n}$, and termination does not occur at x^k, then

$$[Hg_0, Hg_1, \ldots, Hg_k] = [Hg_0, HAHg_0, \ldots, (HA)^k Hg_0] \tag{6.1}$$

$$[d_0, d_1, \ldots, d_k] = [Hg_0, HAHg_0, \ldots, (HA)^k Hg_0] \tag{6.2}$$

$$\langle Ad_j, d_k \rangle = 0 \quad \text{for all} \quad j = k \tag{6.3}$$

$$\alpha_k = \langle Hg_k, g_k \rangle / \langle Ad_k, d_k \rangle \tag{6.4}$$

$$\beta_k = \langle Hg_{k+1}, g_{k+1} \rangle / \langle Hg_k, g_k \rangle \tag{6.5}$$

Proof: The proof follows from a slight modification of the one given for the conjugate gradient method by Luenberger (1973, pp. 174-175). ■

LEMMA 6.2

If the secant method (3.25) - (3.29) is applied to problem (3.12) starting with any $x^0 \in R^n$ and an arbitrary symmetric positive definite matrix $H_0 \in R^n$, then

(6.6) $\langle As_j, s_k \rangle = 0$, $j \neq k$

(6.7) $H_k y_j = s_j$, $j < k$

and the method terminates in at most n iterations. Moreover, $H_n = A^{-1}$.

Proof: This result is well known and may be found in Broyden (1967).

The following theorem establishes the equivalence between the preconditioned conjugate gradient method and the secant methods when applied to the minimization of a strictly convex function.

■

Theorem 6.1.

For the quadratic problem (3.12), starting with the same $x^0 \in R^n$ and the same symmetric positive definite matrix $H_0 \in R^{n \times n}$, the two algorithms -- the preconditioned conjugate gradient method (3.15) - (3.19) and the secant method (3.25) - (3.29) -- yield identical iterates.

Proof: Comparing (3.16), (3.17), and (3.26),(3.27) we notice that if in the two algorithms the iterates x are the same and the directions d and p satisfy $p = \sigma d$ for some $\sigma \neq 0$, then the next iterate $\bar{x}$ would also be the same for these algorithms. We assume that up to the k-th iteration, the iterates x^j, $j \leq k$ are identical in the two algorithms and that $p_j = \sigma_j d_j$ for some $\sigma_j \neq 0$, $j \leq k$, and show by induction that the conditions also hold at the (k+1)-th iteration. From Lemma 6.1 we have

$$[p_0,p_1,\ldots,p_k] = [H_0g_0,H_0g_1,\ldots,H_0g_k] = [d_0,d_1,\ldots d_k] \tag{6.8}$$

and

$$[H_0g_0,H_0g_1,\ldots,H_0g_{k+1}] = [d_0,d_1,\ldots,d_{k+1}] \tag{6.9}$$

We would like to obtain

$$[p_0,p_1,\ldots,p_{k+1}] = [H_0g_0,H_0g_1,\ldots,H_0g_{k+1}] \tag{6.10}$$

because then, using (6.9) gives

$$[p_0,p_1,\ldots,p_{k+1}] = [d_0,d_1,\ldots,d_{k+1}], \tag{6.11}$$

and by the A-conjugation of the vectors d's and p's we would have $p_{k+1} = \sigma_{k+1}d_{k+1}$ for some $\sigma_{k+1} \neq 0$ since we already have $p_k = \sigma_j d_j$ for $\sigma_j \neq 0$, $j \leq k$ (induction hypothesis).

The arithmetical details may be worked out to show that using Lemma 6.2 and the update formula (3.29) we have

$$-p_{k+1} = (1+\theta_{k,1})H_0g_{k+1} + (\delta_{k,1}+\theta_{k,1})p_k$$

$$+ \sum_{j=1}^{k}\left\{H_og_{k+1-j} \sum_{i=k+2-j}^{k+1} a_i\theta_{k-j,i+j-k}\right.$$

$$\left. + \; p_{k-j} \sum_{i=k+2-j}^{k+1} a_i(\delta_{k-j,i+j-k} + \theta_{k-j,i+j-k})\right\}$$

where $\theta_{k,\ell} = -(1-\tilde{\Phi}_k)\langle y_k,H_kg_{k+\ell}\rangle/\langle y_k,H_ky_k\rangle$

$$\delta_{k,\ell} = -\lambda_k\tilde{\Phi}_k\langle y_k,H_kg_{k+\ell}\rangle/\langle y_k,s_k\rangle$$

and a_i denotes the coefficient of $H_0 g_i$, $i \leq k+1$.

Note that the scalar $(1+\theta_{k,1})$ cannot be zero since otherwise $p_{k+1} \in [p_0, \ldots, p_k]$ which using (6.6) implies that $p_{k+1} = 0$. Thus (6.12) gives (6.10) which establishes the theorem. ■

It is immediate from the above theorem that if $H_0 = \gamma I$, $\gamma \neq 0$, then the secant algorithm (3.25) - (3.29) is equivalent to the basic conjugate gradient algorithm, a result of Myers (1968). It must be acknowledged that a result along the lines of Theorem 6.1 might be true was suggested to the author by Peter Percell. Although our work was done independently, we have since learned that Nazareth (1977b) has established essentially the same theorem.

So far we have assumed that the Hessian A of the quadratic function is positive definite. This gives us the nice feature of search directions being directions of descent at every step in either the PCGM or secant methods. In practice, however, the algorithm must be able to accommodate the possible non-positive definiteness of the Hessian matrix which might occur at regions remote from the solution. As observed in section 3, a direction of negative curvature should be a fruitful direction in which to search for a function decrease. Ideally, we would like to define or modify the algorithm so that we do not need to go out of our way to take care of negative curvature at each step, if it exists, and so that the implementation remains simple. The results below have precisely this purpose.

LEMMA 6.3

For the quadratic problem (3.12) if the vectors $\{d_j\}_{j=0}^{k}$, $k<n$ are A-conjugate and if A is positive definite in these directions, then any conjugate direction algorithm of the form (3.13) - (3.14) satisfies

$$\langle g_{k+1}, d_j \rangle = 0 \quad \text{for all} \quad j \leq k \ .$$

Proof: Follows immediately using an induction argument since for the quadratic problem we have $g_{k+1} = g_k + \alpha_k A d_k$ ■

LEMMA 6.4

The preconditioned congugate gradient method, when applied to the quadratic problem (3.12), generates a direction which is A-conjugate if A is positive definite in all the previous directions.

Proof: Follows directly from Lemma 6.1 since positive definiteness of A in all the previous directions suffices for that lemma. ■

LEMMA 6.5

The preconditioned conjugate gradient method, when applied to the quadratic problem (3.12), generates a descent direction for the objective function q if A is positive definite in all the previous directions.

Proof: The assumptions of Lemma 6.3 are fulfilled using Lemma 6.4. The result follows on premultiplying (3.19) by g_{k+1}^T since H is positive definite. ■

THEOREM 6.2

The preconditioned conjugate gradient method, when applied to the quadratic problem (3.12) in which A is indefinite, yields a direction of negative curvature in at most n iteratations.

Proof: Let $\{d_j\}_{j=1}^{k}$ be the directions of positive curvature of A, generated by PCGM. By Lemma 6.4 these directions must be A-conjugate and hence linearly independent. The result now follows by contradiction if $k = n$, because then A would be positive definite. ■

THEOREM 6.3

For the general unconstrained problem (2.1), suppose the preconditioned conjugate gradient method is applied to minimize the quadratic approximation at x

$$q(\Delta x) = f(x) + \langle \nabla f(x), \Delta x \rangle + \tfrac{1}{2}\langle \nabla^2 f(x)\Delta x, \Delta x\rangle \quad ,$$

and the iterates $\Delta x_1, \Delta x_2, \ldots, \Delta x_k$ (taking $\Delta x_0 = 0$) are generated. If the vectors $\{\Delta x_j - \Delta x_{j-1}\}_{j=1}^{k}$ be the directions of positive curvature of f at x, then Δx_k is a direction of descent for f at x .

Proof: Since $\Delta x_0 = 0$, $d_0 = -H\nabla q(0) = -H\nabla f(x)$, the PCGM iterates may be written as

$$\Delta x_k = \Delta x_{k-1} + \alpha_{k-1} d_{k-1} = \sum_{j=0}^{k-1} \alpha_j d_j \quad ; \qquad k \geq 1 \ .$$

Since H is positive definite, d_0 is a direction of descent for f at x and for q at Δx_0 . Hence Δx_1 is a direction of descent for f at x . By induction, let

$\{\Delta x_j\}_{j=1}^{k-1}$ be the directions of descent for f at x. By Lemma 6.5, $\alpha_{k-1}d_{k-1}$ is a direction of descent for q at Δx_{k-1}. Thus, using Lemma 6.4 we have

$$\langle \nabla f(x), \alpha_{k-1}d_{k-1}\rangle < 0 .$$

The result is now immediate from the definition of Δx_k using the induction hypothesis ∎

The significance of the above theorem may be understood by realizing what it offers, namely, that so long as the PCGM directions satisfy $\langle \nabla^2 f(x)d_j, d_j\rangle > 0$ they are all directions of descent for f at x, and the first direction d_k for which $\langle \nabla^2 f(x)d_k, d_k\rangle < 0$ is a direction of negative curvature of f at x. Also if the Hessian calculations are avoided, but the product $\langle \nabla^2 f(x)d, d\rangle$ is available, then it is sufficient to declare whether the Hessian is positive definite in the particular direction d. By Theorem 6.1 these comments also apply to secant methods.

In the case of equality constrained optimization, we know that the variables x and the multipliers λ may be updated by calculating the respective changes Δx and $\Delta\lambda$ obtained by solving the extended linear system (4.8). However, increasing the dimension of the problem to $n+m$ while seeking to solve (4.8) directly seems highly undesirable, yet decoupling this system may not be justified either, at least far away from the solution where regularity conditions might not hold. With this in mind, we consider solving the extended system (4.8) using the preconditioned conjugate gradient method. This also offers the option of not having to solve any linear system, as such, if the pure conjugate gradient method is employed. We postpone details on this issue until section 9, and for now concern outselves with the appropriate modifications that might be required to apply PCGM to the linear system (4.8).

The presence of the zero block in the lower right corner of (4.8) immediately shows that the matrix involved in the system is not positive definite. Using Theorem 2.2, however, this matrix may be expected to be non-singular, at least near a solution. Now suppose the preconditioned conjugate gradient method (3.15) - (3.19) is applied to the linear system $Ax = b$, where A is symmetric and non-singular. In all generality it seems fair in this case to allow having a preconditioning matrix H which is symmetric and nonsingular but not necessarily positive definite. The following results provide our necessary extension.

COROLLARY 6.1

Suppose A is symmetric and non-singular, and the nonzero vectors $d_1, d_2, \ldots, d_k$ are A-conjugate. Then the vector d_j, $1 \leq j \leq k$ is linearly independent of the vectors $\{d_1, d_2, \ldots, d_{j-1}, d_{j+1}, \ldots, d_k\}$ if $\langle Ad_j, d_j \rangle \neq 0$.

COROLLARY 6.2

Let $\{d_j\}_{j=1}^{n}$ be a set of nonzero A-conjugate vectors and $\langle Ad_j, d_j \rangle \neq 0 \ \forall \ j$. Then if the conjugate direction algorithm (3.13)-(3.14) is applied to the symmetric non-singular system $Ax = b$, the iterates converge to the unique solution $x^* = A^{-1}b$ in at most n steps. (As expected, $g_k = Ax^k - b$.)

COROLLARY 6.3

Let the preconditioned conjugate gradient method (3.15) - (3.19) be applied to the symmetric non-singular system $Ax = b$ starting at any $x^0 \in R^n$ with an arbitrary symmetric non-singular matrix $H \in R^{n \times n}$, and for some $k > 0$, let $\langle Hg_j, g_j \rangle \neq 0$ where $g_j = Ax^j - b$, and $\langle Ad_j, d_j \rangle \neq 0 \ \forall \ j \leq k$. Then we have (6.1) - (6.5) if the termination does not occur at x^k.

The proofs of the above corollaries are omitted since they easily follow their positive definite counterparts. The conditions in corollary 6.3 might be thought of as too strong at first sight. Notice, however, that if the preconditioning matrix H is taken to be positive definite, then the condition $\langle Hg_j, g_j \rangle \neq 0$ is no longer required, because then $\langle Hg_k, g_k \rangle = 0$ only if $g_k = 0$, in which case x^k is the desired solution. In our numerical computations with various equality constrained problems, the condition $\langle Ad_j, d_j \rangle \neq 0$ was never violated and $\langle Hg_j, g_j \rangle$ happened to vanish only when $g_j = 0$ -- even when H was only taken to be symmetric non-singular. It seems that a result stronger than Corollary 6.3 may be proved in the context of constrained optimization problems, i.e., when the matrix A takes the special form (4.1). At this stage, however, we proceed to give a result which serves as some consolation besides numerical evidence. For the purpose of the following lemma, assume that the constraints in problem (2.2) are not all satisfied at the point x.

LEMMA 6.6

If the preconditioned conjugate gradient method is used to solve the linear system

$$\begin{pmatrix} \nabla_x^2 \mathcal{L} & \nabla h \\ \nabla h^T & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta\lambda \end{pmatrix} = - \begin{pmatrix} \nabla_x \mathcal{L} \\ h \end{pmatrix}$$

then $\exists\ r > 0$ such that the first direction d_0 (in the PCGM algorithm) is always a descent direction for the penalty function $f + \frac{1}{r} h^T h$.

Proof: The proof follows directly using (3.9). (Also see Biggs (1978)). ∎

The assumption that $h(x) \neq 0$ should not distract us from rendering the above lemma useful, since in practice $h(x) = 0$ is seldom realized, if ever. Note that the result holds even if the preconditioner is not positive definite or is singular. However, the submatrix corresponding to the Hessian $\nabla_x^2 \mathcal{L}$ is assumed to be positive definite, which seems reasonable in view of Theorem (2.2).

In view of the above lemma, whenever the assumptions of Corollary 6.3 are not satisfied, we may resort to a line search on the penalty function (2.4) along the direction d_0 . In this context the penalty function seems to be a logical choice for the merit function, although several other merit functions have been proposed, e.g., see Han (1977b) .

7. THE UNCONSTRAINED OPTIMIZATION ALGORITHM

Let us recall that Newton's method for the unconstrained problem (2.1) may be thought of as solving the quadratic problem

$$(7.1)\qquad \underset{\Delta x}{\text{minimize}}\; q(\Delta x) = f(x) + \langle \nabla f(x), \Delta x\rangle + \tfrac{1}{2}\langle \nabla^2 f(x)\Delta x, \Delta x\rangle \ .$$

By the optimality condition (Theorem 2.1), if x is near a local minimum of f, then $\nabla^2 f(x)$ can be expected to be positive definite so that problem (7.1) is well defined. Far from the solution when $\nabla^2 f(x)$ is not positive definite we know that Newton's method may lead to trouble. However, using the theory developed in the previous section we may expect to do reasonably well using the PCGM or equivalently secant methods to solve the quadratic model problem (7.1). This suggests setting up an inner loop (say, iterated k times, $k \le n$) in our algorithm. A basic model may be outlined as:

```
given x and B (symmetric, positive definite)

(7.2)
outer loop
    $B := \begin{cases} I \\ B \end{cases}$
    $\Delta x := 0$
    inner loop (k times)
        $u := \nabla^2 f(x)\Delta x + \nabla f(x)$
        $p := -B^{-1}u$
        $y := \nabla^2 f(x)\cdot p$
        $\beta := -\langle u,p\rangle / \langle y,p\rangle$
        $\Delta x := \Delta x + \beta p$
        $B := B + yy^T/\langle y,p\rangle - Bpp^TB/\langle Bp,p\rangle$
    $x := x + \Delta x$
```

Although identical behavior of secant methods with different matrix updates is expected on the quadratic problem (Dixon (1972a)), we have used the BFGS formula above just because numerically it is supposed to be more stable. It is interesting to note that Blue (1977) independently suggested a similar inner loop philosophy for solving nonlinear equations. Since in this case the Jacobian matrix is no longer guaranteed to be symmetric, our theory in section 6 will not be applicable here. Also, recently, Best (1978) approached the unconstrained problem via a modified conjugate direction algorithm which is similar to ours; but in his algorithm, Best does not take advantage of the secant approximation to the Hessian which may be built in our approach, Instead, he generates new directions from a linear combination of the old ones (once n linearly independent directions have been obtained).

Some observations about the model (7.2) must be addressed here. Note that the inner loop index k may be varied to get different algorithms. For $k = n$, the model essentially gives discrete Newton method. With $k = 1$ and B set to I, we have the gradient method. If $k = 1$ and B is updated, we obtain a secant method with a short-step Hessian approximation. In the case that $\langle y,p \rangle < 0$, we immediately have p as a direction of negative curvature. Hessian approximations satisfy the equation (5.4), and the algorithm requires no line searches, except in the case of negative curvature.

Notice that the second order information which appears in the above model algorithm may be computed easily using a finite difference approximation of the form

$$\nabla^2 f(x)\cdot d \approx \frac{\nabla f(x+td)-\nabla f(x)}{t}\ ; \quad t\approx\sqrt{\tau}$$

where τ is the machine tolerance. Also, the inner loop may be written equivalently in terms of PCGM. Although PCGM never updates the preconditioner and yet essentially gives the secant method on a quadratic function there is one drawback, viz., it does not yield an approximation to the Hessian which, otherwise, could be used as the preconditioner for the subsequent inner loop in the next iteration. On the other hand, secant methods do not generate the same iterates if only a part of the Hessian is approximated due to limited storage available. For this reason we intend to use the PCGM in the inner loop and still update the preconditioner to obtain a better approximation to the Hessian at the end of an inner loop. It may be pointed out that it is possible to carry out PCGM and also keep updating the preconditioner, after calculating the new direction, without altering the net effect of the algorithm in the inner loop because $H_j g_k = H_0 g_k$, $j<k$ where H_j is obtained via any secant update formula in Broyden's class. However, this is true only so long as we have full storage at our disposal to approximate the Hessian, i.e., only if we are approximating the full Hessian.

In order to include the possibility of not having enough storage and having to approximate only a part of the Hessian, we will keep a duplicate copy p of the matrix B, so that P will be used as the preconditioner and B will be updated in the inner loop to get an improved approximation of the Hessian. Substituting these ideas into the model algorithm (7.2), we propose the following algorithm.

given x and B (symmetric, positive definite)

```
(7.3)
Outer loop
    B :=   $\begin{cases} I \\ B \end{cases}$
    P :=   $B$
    Δx :=  $0$
    u :=   $\nabla f(x)$
    s :=   $-P^{-1}u$
    Inner loop (k times)
        y :=   $\frac{1}{t}(\nabla f(x+ts) - \nabla f(x)$ ;   $t = \sqrt{\tau} / \|s\|$
        β :=   $-\langle u,s\rangle / \langle y,s\rangle$
        Δx :=  $\Delta x + \beta s$
        B :=   $B + \frac{yy^T}{\langle y,s\rangle} - \frac{Bss^TB}{\langle Bs,s\rangle}$
        u :=   $u + \beta y$
        s :=   $-P^{-1}u + \frac{\langle P^{-1}u,y\rangle}{\langle y,s\rangle} s$
    x :=   $x + \Delta x$
```

At first glance it might seem better to keep an inverse Hessian approximation H rather than the direct Hessian approximation B. But matrix B being symmetric and positive definite may be kept in factored form and these factors may be updated instead of using the BFGS formula on B as given in (3.22). In this case, the work involved in computing $P^{-1}u$ is no more than that in forming Hu . Moreover, it is a general feeling among experts in the field that updating the factors of B is more stable than updating the inverse Hessian approximation H. Since the factors may also be updated within the same order of arithmetic operations, there is no increase in computation (Fletcher and Powell (1974), Gill, Bolub, Murray and Saunders (1974), Goldfarb (1976)). One such technique to update the

Cholesky factors is given in the appendix.

A few more points about the algorithm in (7.3) may be mentioned at this time. First, as noticed before, if $\langle y,s\rangle < 0$ during some inner loop iteration, say the j-th, it must be terminated immediately because the directions that follow may no longer be Hessian-conjugate. If $j > 1$, then according to Theorem 6.3 we have Δx_{j-1} as a descent direction for f and s_j as a direction of negative curvature. This defines a descent pair which can be used in the line search technique of McCormick (1977), Moré and Sorensen (1979). If $j = 1$, then $-\nabla f$ may be regarded as a descent direction and d_1 is still a direction of negative curvature and again we have a descent pair. We may comment here that in our experiments only using d_j in a simple line search routine produced results comparable to those obtained from the use of the descent pair.

Second, it is advisable to check $\|\Delta x\|$ against some step-length bound, say Δ. This is similar to the trust region philosophy discussed in section 3. If $\|\Delta x\| \leq \Delta$ then Δx is accepted, no function check is made. If $\|\Delta x\| > \Delta$ and if $f(x+\Delta x) < f(x)$, Δx is accepted and Δ set to equal $\|\Delta x\|$. If $\|\Delta x\| > \Delta$ and $f(x+\Delta x) \geq f(x)$, Δx is temporarily accepted and the next inner loop is carried out -- say it gives $\overline{\Delta x}$. Now if $f(x+\Delta x+\overline{\Delta x}) < f(x)$, x is set to $x+\Delta x+\overline{\Delta x}$ and $\Delta = \|\overline{\Delta x}\|$. If not, a "weak" line search is performed along Δx at x to compute $\alpha > 0$ such that $f(x+\alpha\Delta x) < f(x)$ and Δ is set to $\alpha\|\Delta x\|$. Note that the existence of α is guaranteed since Δx is a descent direction for f at x. We call this touchstone <u>level-2 descent</u> simply because we allow the jump in function one time, but not twice -- expecting our method to be acting like Newton's method away from the solution.

Third, we recommend that if $\|\beta_j s_j\| \le \epsilon_1 \|\Delta x_{j-1}\|$ for some preassigned small $\epsilon_1 > 0$, then the inner loop should be terminated since the contribution of any further iterations is negligible.

Fourth, for some cases where the function f is known to have maximum and/or saddle points which may hinder the algorithm on its path toward a minimum, it might be beneficial to have, what we call, a pre-optimality loop as an option. In this option the algorithm should be allowed to run a full inner loop with index $k = n$ sometime after $\|\nabla f(x)\| \le \sqrt{\epsilon_0}$ where $\epsilon_0 > 0$ is some preassigned tolerance on $\|\nabla f\|$ which might determine the stopping criteria for the algorithm, viz., stop if $\|\Delta f(x)\| \le \epsilon_0$. Note that the purpose of a full inner loop is to generate a direction of negative curvature (if one exists) which will necessarily exist around a maxima or saddle point. If no such direction is obtained, essentially a discrete Newton step takes place, and if we are in the region of quadratic convergence, the algorithm may be expected to stop right after the pre-optimality loop is completed.

We now turn to the important consideration of incorporating variable storage capability in the algorithm given in (7.3). As a preliminary step, observe that if B is set to I in each outer loop then updating B in the inner loop has no effect, thus both matrices P and B are not needed in the algorithm. Recall that in this case the algorithm uses the basic conjugate gradient method in the inner loop and no matrices are required.

For large problems it may not be possible to store the full Hessian approximation. The actual Hessian is assumed to be positive definite at the solution, and in many cases it may actually be diagonally dominant. Thus it seems reasonable to use the available storage to approximate as many diagonals as possible, starting with the main diagonal, and assume that the rest of the matrix is zero.

Even if the Hessian matrix $\nabla^2 f$ is sparse, we know that the inverse Hessian $\nabla^2 f^{-1}$ may still be full, and hence we must work with an approximation B to $\nabla^2 f$ rather than an approximation H to $\nabla^2 f^{-1}$ to take advantage of sparsity. In this way, sparsity in B can easily be translated into sparsity in the corresponding Cholesky factorization of B, in the case that B is banded along the main diagonal.

To implement variable storage it is advisable to keep B and P (see (7.3)) in the same array. In fact we prefer to think of B and P as different storage blocks (denoted [B] and [P] respectively) of the same array. Let n_s be the total number of locations available for storage of [B] and [P] combined. Clearly, if $n_s \geq (n^2+n)/2$ then ample space is available to carry out the inner loop using the secant method directly, without the use of [P] (see (7.2)). If $n_s < 2n$ then not enough storage is available even to store a diagonal matrix in [B] and in [P] separately so that they may be changed from step to step. In this case, [B] and [P] are both assumed to represent identity matrices and no updates are performed, so the basic conjugate gradient method is used for the inner loop. If $2n \leq n_s < (n^2+n)/2$ then the number, say n_d, such that

$$n_d = \max\{k \mid (2n+1-k)k \le n_s\}$$

gives the maximum number of diagonals that may be updated in [B] and used as a preconditioner in [P]. Note here that due to symmetry n_d only counts the main diagonal and the super diagonals (or sub diagonals).

Before concluding this section, we mention that the problem of having to preserve symmetry, positive definiteness and sparsity along with satisfying the secant equation using a rank-2 update was considered by Marwil (1978), and independently by Toint (1977). Their procedure requires the solution of an additional sparse linear system. Moreover, in the case of a full Hessian, the Marwill-Toint update reduces to the Powell-symmetric-Broyden (PSB) formula which does not preserve positive definiteness. (Also see Toint (1978).)

Schubert (1970), and independently Broyden (1971), presented a modification of Broyden's (1965) method for solving nonlinear equations which takes sparsity into account. This method, although very successful in sparse nonlinear equations, unfortunately does not retain symmetry. One straightforward approach would be to carry out Schubert's sparse update and then symmetrize the resulting matrix. If required, positive definiteness may also be forced, but the secant equation would be violated.

An ad-hoc technique, which we find easy to use, is to update the sparse Cholesky factors of B and then zero out the appropriate parts (incomplete Cholesky factorization). In this case the update inherits symmetry, positive definiteness and sparsity, but the secant equation is not necessarily satisfied.

8. THE CONSTRAINED OPTIMIZATION ALGORITHM.

Consider Newton's method using the augmented Lagrangian to solve the equality constrained problem (2.2). As noticed earlier in section 4, the system of linear equations that must be solved in order to get x and λ corrections is

$$(8.1) \qquad \begin{bmatrix} \nabla_x^2 \mathcal{L}(x,\lambda,c) & \nabla h(x) \\ \nabla h(x)^T & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}(x,\lambda,c) \\ h(x) \end{bmatrix}$$

where $\nabla_x^2 \mathcal{L}(x,\lambda,c)$ is given by

$$(8.2) \qquad \nabla_x^2 \mathcal{L}(x,\lambda,c) = \nabla^2 f(x) + c\nabla h(x)\nabla h(x)^T + \sum_{i=1}^{m} (\lambda_i + ch_i(x))\nabla^2 h_i(x)$$

As in unconstrained optimization, suppose that $\nabla_x^2 \mathcal{L}(x,\lambda,c)$ is being approximated by a matrix B which will be improved using secant update formulas as the iteration progresses. Recall that following Hestenes (1969), the matrix $\nabla_x^2 \mathcal{L}(x,\lambda,c)$ may be assumed to be positive definite for c large enough, at least in the neighborhood of a solution. Following the approach in section 7, a model algorithm may now be written as (8.3) on the following page. Note that the subvectors v_x and v_λ of an $n+m$ vector v denote the parts of v corresponding to x and λ, respectively.

As implied by Corollary 6.3, the inner loop in the algorithm (8.3) may only be run as long as both $\langle u,p \rangle$ and $\langle y,p \rangle$ are non zero. Note that since B is taken to be positive definite and is maintained that way through the secant updates $\mathfrak{B}$, the matrix P is nonsingular if ∇h is of full rank. If not, modified Gaussian decomposition for given x,λ,c and B (symmetric, positive definite)

```
(8.3)
Outer loop
    $\Delta x := 0$
    $\Delta\lambda := 0$
    Inner loop (k times)
        $P := \begin{bmatrix} B & \nabla h(x) \\ \nabla h(x)^T & 0 \end{bmatrix}$
        $u := \nabla^2\mathcal{L}(x,\lambda,c)\begin{pmatrix}\Delta x \\ \Delta\lambda\end{pmatrix} + \nabla\mathcal{L}(x,\lambda,c)$
        $p := -P^{-1}u$
        $y := \nabla^2\mathcal{L}(x,\lambda,c)p$
        $\beta := -\langle u,p\rangle / \langle y,p\rangle$
        $\Delta x := \Delta x + \beta p_x$
        $\Delta\lambda := \Delta\lambda + \beta p_\lambda$
        $B := \mathfrak{B}(p_x, \nabla_x^2\mathcal{L}(x,\lambda,c)p_x, B)$
    $x := x + \Delta x$
    $\lambda := \lambda + \Delta\lambda$
    $c := \Pi(x,\lambda,c)$
```

symmetric indefinite matrix may be used to force nonsingularity in P. Moreover, if PCGM is used in the inner loop then this decomposition will be required only in the outer loop.

Also the second order information involved in the algorithm may be avoided by using finite differences to calculate $\nabla_x^2\mathcal{L}(x,\lambda,c)$ and the rest of the information is already available. It must be noted here that $\langle y,p\rangle < 0$ is of no significance to us; instead, if $\langle\nabla_x^2\mathcal{L}(x,\lambda,c)p_x,p_x\rangle < 0$ then we conclude that p_x is a direction of negative curvature of $\nabla_x^2\mathcal{L}(x,\lambda,c)$. Unfortunately, we do not know the component of p_x which lies in the orthogonal complement of $\nabla h(x)$. For this reason, we must try to keep $\nabla_x^2\mathcal{L}(x,\lambda,c)$ positive definite all the time by the chöice of a big enough c . In fact, the penalty constant update formula $\Pi(x,\lambda,c)$ that we have used in the outer loop above is used primarily to indicate this situation. If, however, $\nabla_x^2\mathcal{L}(x,\lambda,c)$ cannot be made positive definite--which could be true far away from the solution--we could perform a line search along the first direction in the inner loop using the penalty function (Lemma 6.6).

Some interesting observations from the model are apparent as follows. Of course, for $k = n$ the method gives essentially Newton's method. For $k = 1$ it turns into Tapia's method as discussed in section 4 with the update formula (4.6) (U_* in Tapia (1977)). With $k = 1$ and P set to the diagonal matrix $\begin{bmatrix} I_n & 0 \\ 0 & -I_m \end{bmatrix}$, the model yields gradient methods on both x and λ variables -- on λ in terms of the dual problem -- see Tapia (1977). As in the unconstrained case, the Hessian approximations obtained here may be expected to be numerically stable because of short-steps. Also the only line searches that the algorithm requires are those to take care of the case when $\nabla_x^2 \mathcal{L}(x,\lambda,c)$ is indefinite.

The variable storage capability may be incorporated following the presentation in the previous section. One extreme case is of importance, however. This is the case when P is set to the diagonal matrix $\begin{bmatrix} I_n & 0 \\ 0 & -I_m \end{bmatrix}$. Note that in this case no matrix storage is required and no linear system needs to be solved. Besides, this option could be very useful far away from the solution since it takes the first step along the steepest descent direction as noted above.

Putting together some of the preceding ideas the algorithm may be stated in the form of (8.4).

It should be remarked that in the options for P in (8.4), the inverse notation simply indicates that a modified Gaussian elimination of the corresponding matrix is contained in P. Of course, in its first option the matrix P is not actually used. The computation of vector Pu is carried out via an elimination process rather than actual matrix-vector multiplication. The decomposition which we have been calling modified Gaussian elimination is simply a symmetric indefinite factorization of Bunch-Parlett type which forces nonsingularity in the matrix.

Note that z represents the vector $\nabla_x^2 \mathcal{L}(x,\lambda,c)s_x$, and the components of vector y are obtained using simple matrix-vector multiplication. Once again, the penalty constant update formula simply indicates that c is updated if $\nabla_x^2 \mathcal{L}(x,\lambda,c)$ turned out to be given x,λ,c and B (symmetric, positive definite)

```
(8.4)
Outer Loop
    $B := \begin{cases} I \\ B \end{cases}$
    $P := \begin{cases} \begin{bmatrix} I_n & 0 \\ 0 & -I_m \end{bmatrix}^{-1} \\ \begin{bmatrix} B & \nabla h(x) \\ \nabla h(x)^T & 0 \end{bmatrix}^{-1} \end{cases}$
    $\Delta x := 0$
    $\Delta\lambda := 0$
    $u_x := \nabla_x \mathcal{L}(x,\lambda,c)$
    $u_\lambda := h(x)$
    $s := -Pu$
    Inner Loop (k times)
        $z := \dfrac{\nabla_x \mathcal{L}(x+ts_x,\lambda,c) - \nabla_x \mathcal{L}(x,\lambda,c)}{t}$ ;   $t = \sqrt{\tau}/\|s_x\|$
        $y_x := z + \nabla h(x)s_\lambda$
        $y_\lambda := \nabla h(x)^T s_x$
        $\beta := -\langle u,s\rangle / \langle y,s\rangle$
        $\Delta x := \Delta x + \beta s_x$
        $\Delta\lambda := \Delta\lambda + \beta s_\lambda$
        $B := B + \dfrac{zz^T}{\langle z,s_x\rangle} - \dfrac{Bs_x s_x^T B}{\langle Bs_x,s_x\rangle}$
        $u := u + \beta y$
        $s := -Pu + (\langle Pu,y\rangle / \langle y,s\rangle)s$
    $x := x + \Delta x$
    $\lambda := \lambda + \Delta\lambda$
    $c := \Pi(x,\lambda,c)$
```

indefinite. Some updating rules as indicated by Pierre and Lowe (1975), and Glad (1976), may be used successively in order to maintain $\nabla_x^2\mathcal{L}(x,\lambda,c)$ positive definite. Also notice that each time the penalty constant c changes to $\bar{c}$, $\nabla_x^2\mathcal{L}(x,\lambda,c)$ changes by the amount

$$(\bar{c}-c)(\nabla h(x)\nabla h(x)^T+\sum_{i=1}^{m} h_i(x)\nabla^2 h(x)).$$

Since $\nabla h(x)$ is known, the first term can be compensated for by replacing B with

$$B+(\bar{c}-c)\nabla h(x)\nabla h(x)^T .$$

The penalty parameter r in the penalty function (2.4) deserves mention here. Recall that in Lemma 6.6 we establish that the first direction in each inner loop of (8.4) is a descent direction on the penalty function. If the product $\langle\nabla P,s_x\rangle$ is simplified, then it turns out that starting with

$$(8.5) \qquad r=\frac{2\gamma\langle h,h\rangle}{|\langle h,\lambda\rangle|} \ ; \quad 0<\gamma<1$$

the result holds for some $r>0$. In practice, however, the above choice of r usually suffices for the descent property to hold. If not, r may be replaced by $\bar{r}$ where

$$(8.6) \qquad \bar{r}=\frac{2\langle h,h\rangle}{\frac{2}{r}\langle h,h\rangle+\langle\nabla P,s_x\rangle+\epsilon} \ ; \quad \epsilon>0$$

to get the descent property to be satisfied. Also note, by the way, that instead of (8.5)

$$r = \frac{2\gamma\|g\|}{\|\lambda\|} \; ; \quad 0<\gamma<1 \tag{8.7}$$

may alternatively be used.

9. COMPUTATIONAL RESULTS

Experimental computer programs have been coded which implement the algorithms given in (7.3) and (8.4) with sufficient generality to subsume the different options provided in these algorithms. These programs are intended as a pilot project to test the fundamental validity of the ideas developed in this thesis. The programs have not had the benefit of a lengthy development process nor the exhaustive testing necessary to produce high quality mathematical software. For these reasons the programs will not be documented here nor will the coding be discussed in great detail. In particular, no extraordinary efforts were spent on the efficiency of the programs in obtaining solutions and in economizing execution time. Indeed, a number of potential savings in this area were deliberately foregone in order to avoid complications in the coding. In view of the rather preliminary nature of the numerical testing at hand, precise comparisons with other well-documented algorithms in the literature are difficult to make. However, we hope to address these issues in subsequent work.

A simple descent strategy is employed in line searches. The original search direction is normalized and a unit step size is tried. If the unit step size is acceptable (the merit function attains lower value), the step size is doubled successively until a point with a higher merit function value than its precedent is reached; and then the preceding step size is accepted. If the step size of one is not acceptable, the step size is successively halved until a point with a lower merit function value is found.

In the unconstrained optimization algorithm, Cholesky factors of the Hessian approximation B are updated using the BFGS formula in the composite-t algorithm of Fletcher and Powell (1974). In the constrained optimization algorithm, the matrix B is updated directly using the BFGS formula as in (8.4). A modified Cholesky decomposition is used to obtain the indefinite symmetric factorization of matrix P and to force nonsingularity in it. Also in either case, the initial matrix B, if used, is taken to be the identity matrix.

Most of the test problems selected are well-known examples taken from the literature. For each problem all published starting points were used and in most cases several additional starting points were added, usually to probe some form of difficulty, such as violation of optimality conditions (Theorems 2.1 and 2.2) or starting relatively far from a solution.

A fair summary of the results is that the algorithms converged to a correct solution from all of the starting points for most of the problems and from most of the starting points for virtually all of the problems. Although precise comparisons with other algorithms seem pointless to report at this time, yet the performance of our programs in this regard was generally satisfactory, ranging from quite good on several problems to fairly poor on a few. Thus the methods developed here appear to show great promise as the basis for robust algorithms in nonlinear optimization.

We now list numerical results for the test problems. The convergence test consisted simply of a tolerance of 10^{-5} for $\|\nabla f\|$ in the unconstrained case and for $\max\{\|\nabla_x \mathcal{L}\|, \|h\|\}$ in the constrained case. In the tables that follow, the number of evaluations of the problem functions (NF) and their derivatives (ND) is given, along with an approximate total CPU time to attain the accuracy noted above. As a means of preliminary comparison, in the unconstrained case, similar information was obtained from Shanno and Phua's (1976) quasi-Newton algorithm (denoted by QNSP below) and from Polak-Ribiere conjugate gradient algorithm (denoted by CGPR below) (see Klessig and Polak (1972)) using Lasdon's line search routine (see Fox et al (1975)). In the constrained case, Biggs' (1975) program, OPRQP, using recursive quadratic programming was employed. All computations were performed in double precision arithmetic on the ITEL AS/6 computer using the same FORTRAN compiler.

Unconstrained Optimization Problems (UCOP)

UCOP1: Chebyquad function (Fletcher (1965))

$$f(x) = \sum_{i=1}^{n} (\Phi_i(x))^2$$

where

$$\Phi_i(x) = -\int_0^1 T_i(\xi)d\xi + \frac{1}{n} \sum_{j=1}^{n} T_i(x_j)$$

and T_i is the i-th Chebychev Polynomial on (0,1).

Starting points:

(1) $$x_i^0 = i/(n + 1) \ .$$

UCOP2: Mancino function (Shanno and Phua (1978a))

$$f(x) = \sum_{i=1}^{n} (\Phi_i(x))^2$$

where

$$\Phi_i(x) = \sum_{\substack{j=1 \\ j\neq i}}^{n} \left[\sqrt{x_j^2 + i/j}\left(\sin^{\alpha}\log\sqrt{x_j^2 + i/j} + \cos^{\alpha}\log\sqrt{x_j^2 + i/j}\right)\right]$$

$$+ \beta n x_i + (i - n/2)^{\gamma}$$

$\alpha = 5, \quad \beta = 14, \quad \gamma = 3$.

Starting points:

(1) $$x^0 = \frac{-n\beta}{\beta^2 n^2 - (\alpha + 1)^2 (n - 1)^2} (\Phi_1(0), \Phi_2(0), \ldots, \Phi_n(0))$$

UCOP3: Oren function (Oren (1973))

$$f(x) = \left(\sum_{i=1}^{n} i x_i^2\right)^p \quad \text{with} \quad p = 2 \ .$$

Starting points:

(1) $x^0 = (1,1,1,\ldots)$

UCOP4: Rosenbrock extended function (Shanno (1978a))

$$f(x) = \sum_{i=1}^{n-1} (x_i - 1)^2 + 100 \sum_{i=1}^{n-1} (x_i^2 - x_{i+1})^2$$

Starting points:

(1) $x^0 = (70,70,\ldots)$

(2) $x^0 = (50,-50,50,-50,\ldots)$

(3) $x^0 = (2,2,2\ldots)$

(4) $x^0 = (-3,-3,-3,\ldots)$

UCOP5: Rosenbrock separated function (Moré et al (1978))

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

Starting points:

(1) $x^0 = (-1,2,1,1,1,\ldots)$

(2) $x^0 = (-1.2,1,-1.2,1,-1.2,1,\ldots)$

(3) $x^0 = (2,3,2,3,2,3,\ldots)$

UCOP6: Sine-Exponential function (unpublished)

$$f(x) = \sum_{i=1}^{n}(i+2n)e^{x_i} + \sum_{i=1}^{n}\left(\frac{n+1}{2}\right) x_i^2 + \prod_{i=1}^{n} \sin x_i$$

Starting points:

(1) $x^0 = (15,15,15,\ldots)$

(2) $x^0 = (1,2,3,1,2,3,1,2,3,\ldots)$

(3) $x^C = (-2,-2,-2,\ldots)$

Equality Constrained Optimization Problems (ECOP)

ECOP1: (Miele, et al (1971))

$n = 3, \quad m = 1$

$f(x) = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4$

$h_1(x) = x_1(1 + x_2^2) + x_3^4 - 4 - 3\sqrt{2}$

Solutions obtained:

(1) $x^* \approx (1.1049, 1.1967, 1.5353)$

Starting points:

(1) $x^0 = (11,12,15)$

(2) $x^0 = (2.7,2.9,3.8)$

(3) $x^0 = (1.4,1.5,1.9)$

ECOP2: (Miele et al (1971))

$n = 5, \; m = 2$

$f(x) = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_3 - 1)^2 + (x_4 - 1)^4 + (x_5 - 1)^6$

$h_1(x) = x_4 x_1^2 + \sin(x_4 - x_5) - 2\sqrt{2}$

$h_2(x) = x_2 + x_3^4 x_4^2 - 8 - \sqrt{2}$

Solutions obtained:

(1) $x^* \approx (1.1661, 1.1821, 1.3802, 1.5060, .6109)$

(2) $x^* \approx (-.9868, -.9142, -1.303, 1.893, .4976)$

Starting points:

(1) $x^0 = (2,2,2,2,2)$

(2) $x^0 = (-1,3,-.5,-2,-3)$

(3) $x^0 = (12,13,14,15,7)$

(4) $x^0 = (5.7,5.9,6.9,7.5,3.1)$

<u>ECOP3</u>: (Miele et al (1971))

$n = 5,\ M = 3$

$$f(x) = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_4)^4 + (x_4 - x_5)^4$$

$$h_1(x) = x_1 + x_2^2 + x_3^3 - 2 - 3\sqrt{2}$$

$$h_2(x) = x_2 - x_3^2 + x_4 + 2 - 2\sqrt{2}$$

$$h_3(x) = x_1 x_5 - 2$$

Solutions obtained:

(1) $x^* \approx (1.1911,1.3626,1.4728,1.6350,1.6790)$

(2) $x^* \approx (-.7662,2.667,-.4682,-1.619,-2.610)$

(3) $x^* \approx (-2.702,-2.990,.1719,3.848,-.7401)$

Starting points:

(1) $x^0 = (2,2,2,2,2)$

(2) $x^0 = (-1,3,-.5,-2,-3)$

(3) $x^0 = (5.9,6.8,7.3,8.1,8.4)$

(4) $x^0 = (150,160,170,180,190)$

ECOP4: (Powell (1969))

$n = 5,\ m = 3$

$f(x) = x_1x_2x_3x_4x_5$

$h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10$

$h_2(x) = x_2x_3 - 5x_4x_5$

$h_3(x) = x_1^3 + x_2^3 + 1$

Solutions obtained:

(1) $x^* \approx (-1.7172,1.5957,1.8272,.7636,.7636)$

(2) $x^* \approx (-1.7172,1.5957,1.8272,-.7636,-.7636)$

(3) $x^* \approx (-.6991,-.870,-2.790,-.6967,-.6967)$

(4) $x^* \approx (.3920,-1.020,0.,2.968,0.)$

Starting points:

(1) $x^0 = (-1,2,1,-2,-2)$

(2) $x^0 = (-2,2,2,2,2)$

(3) $x^0 = (-2,2,2,-1,-1)$

(4) $x^0 = (-1,-1,-1,-1,-1)$

(5) $x^0 = (-100,100,100,50,50)$

ECOP5: (Himmelblau (1972), problem 4a, pp. 396)

$n = 10,\ m = 3$

$$f(x) = \sum_{i=1}^{10} \left\{ e^{x_i} \left[c_i + x_i - \ln\left(\sum_{j=1}^{10} e^{x_j} \right) \right] \right\}$$

$$h_1(x) = e^{x_1} + 2e^{x_2} + 2e^{x_3} + e^{x_6} + e^{x_{10}} - 2$$

$$h_2(x) = e^{x_4} + 2e^{x_5} + e^{x_6} + e^{x_7} - 1$$

$$h_3(x) = e^{x_3} + e^{x_7} + e^{x_8} + 2e^{x_9} + e^{x_{10}} - 1$$

and c_i are given by

$c = (-6.089, -17.164, -34.054, -5.914, -24.721, -14.986, -24.1,$

$-10.708, -26.662, -22.179)$

Solutions obtained:

(1) $x^* \approx (-3.2, -1.91, -.24, -6.56, -.72, -7.27, -3.6, -4.02, -3.29, -2.33)$

(2) $x^* \approx (-2.3, -.35, -15.2, -6.46, -.76, -6.51, -2.78, -3.06, -1.61,$

$-.713)$

Starting points:

(1) $x^0 = (.5, .75, 2.2, 1.5, 1.7, 1.5, .7, .75, .5, .25)$

(2) $x^0 = (-.4, -.7, -2, -1.5, -1.5, -1.4, -.75, -.8, -.6, -.3)$

(3) $x^0 = (.1, .2, .3, .4, .5, .6, .7, .8, .9, .7)$

(4) $x^0 = (7, 9, -6, 3, 8, 8, 7, 6, 7, 8)$

TABLE 1: UNCONSTRAINED TEST RESULTS

			Algorithm (7.3)			
Problem	n	x^0	CGPR NF/ND CPU sec.	B: =I NF/ND CPU sec.	B: =B NF/ND CPU sec.	QNSP NF/ND CPU sec.
UCOP1	5	(1)	28/4 .0284	4/13 .0339	4/12 .0376	31/31 .0447
	7	(1)	62/12 .0591	11/34 .0774	11/46 .1376	34/34 .0759
	9	(1)	148/30 .1537	12.71 .1682	30/102 .3917	46/46 .1589
UCOP2	10	(1)	44/6 .3881	13/7 .1703	13/7 .1712	13/13 .2251
	15	(1)	43/6 .8474	13/7 .3645	13/7 .3661	11/11 .4183
	20	(1)	58/8 2.0243	17/9 .8262	17/9 .8279	13/13 .8666
	25	(1)	57/8 3.1340	17/9 1.2873	17/9 1.2893	13/13 1.3525
UCOP3	10	(1)	64/11 .0465	47/24 .0482	47/24 .0482	140/140 .3808
	15	(1)	72/14 .0558	69/35 .0780	69/35 .0780	211/211 .9265
	20	(1)	86/16 .0679	23/84 .1601	23/39 .2169	262/262 1.7260
	25	(1)	94/18 .0794	21/102 .2087	21/45 .3699	316/316 2.9676
UCOP4	20	(1)	2478/474 1.3152	42/392 .6906	26/105 .7946	322/322 2.0331
	20	(2)	481/101 .2866	12/406 .7033	12/220 1.7161	334/334 2.1517
	20	(3)	339/74 .2118	16/112 .2058	16/51 .3756	60/60 .3250
	20	(4)	885/199 .5398	21/391 .6777	21/211 1.6087	197/197 1.1734
UCPO5	20	(1)	148/23 .0783	10/35 .0686	10/33 .2249	40/40 .1873
	20	(2)	148/23 .0787	48/143 .2408	17/144 .9113	60/60 .3425
	20	(3)	80/12 .0498	8/30 .0606	8/37 .2727	67/67 .3475
UCOP6	20	(1)	54/9 .1029	23/23 .1166	23/27 .2246	99/99 .8625
	20	(2)	46/8 .0907	9/13 .0705	9/14 .1283	24/24 .1710
	20	(3)	34/6 .0712	3/14 .0699	3/21 .2125	20/20 .1906

TABLE 2: EQUALITY CONSTRAINED TEST RESULTS

			ALGORITHM (8.4)	BIGGS' OPRQP
	x^0	x^*	NF/ND	NF/ND
Problem	used	found	CPU sec.	CPU sec.
ECOP1	(1)	(1)	12/34 .074	37/29 .056
	(2)	(1)	10/28 .064	31/28 .054
	(3)	(1)	6/16 .043	19/19 .046
ECOP2	(1)	(1)	9/33 .095	52/35 .114
	(2)	(2)	32/124 .310	3911/1000* 2.916
	(3)	(1)	18/68 .178	219/92 .285
	(4)	(1)	17/65 .169	41/43 .128
ECOP3	(1)	(1)	8/22 .073	17/17 .070
	(2)	(2)	6/16 .057	16/16 .067
	(3)	(1)	11/27 .095	30/24 .094
	(4)	(3)	28/82 .233	(1)**
ECOP4	(1)	(2)	10/28 .089	23/22 .088
	(2)	(1)	7/19 .064	21/20 .082
	(3)	(2)	8/22 .073	27/23 .091
	(4)	(3)	10/28 .089	26/23 .093
	(5)	(4)	19/55 .158	(13)***
ECOP5	(1)	(1)	14/94 .458	946/461 3.955
	(2)	(2)	14/105 .480	(28)***
	(3)	(2)	14/100 .473	(36)***
	(4)	(2)	21/151 .755	95/31****

*: Algorithm stopped due to excessive function evaluations.

**: Lagrange multiplier equations singular at the indicated iteration.

***: Search direction uphill at the indicated iteration.

****: Line search failed; $f \approx -10^4$, $||h|| \approx 10^3$.

10. CONCLUDING REMARKS

The equivalence of preconditioned conjugate gradient methods and secant methods for quadratic problems is used to develop numerically stable algorithms for unconstrained and equality constrained problems of nonlinear optimization. Robust algorithms for nonlinear optimization are by necessity complex. The strategy underlying any such algorithm inevitably depends on using local information to deduce global properties that lead to an improved estimate of the solution. Our algorithms tend to stay as close to Newton's method as possible without getting influenced by the short comings of the Newton iteration. A successful strategy should not depend on conditions that hold only for special cases, or in a close neighborhood of the optimum. The algorithms described in this thesis are able to adapt to difficult circumstances, so that progress can be guaranteed far from the solution, even when local indications are misleading.

The algorithms presented are well suited for large problems, because it is possible to implement them without involving any matrices. The numerical results obtained by using an ad-hoc method (described in section 7) are not satisfactory.

The algorithm for equality constrained optimization needs two major improvements. First of all, a clever use of the penalty constant in the augmented Lagrangian is desired. We believe the inner loop iterations in (8.4) should predict an appropriate value for this parameter. Second, it is necessary to modify the algorithm to incorporate inequality constraints. Newton's method, in the usual sense, is not directly applicable to inequality constrained problems. Therefore, a number of methods have been proposed which convert the inequality constrained problem into an equality constrained problem or an unconstrained problem.

APPENDIX: A FACTORIZED SECANT UPDATE PROCEDURE.

Since the detailed enumeration of any numerical algorithm can make especially dull reading and need be absorbed only by the most enthusiastic of readers, we describe a procedure to obtain rank-2 updates of the form (3.20) in this appendix so as not to disrupt the continuity of the main body of this thesis.

Notice that since the Hessian approximation matrix B is always kept symmetric and positive definite ($\langle y,s\rangle > 0$ in (3.20)), a nonsingular lower triangular matrix L can be obtained via Cholesky factorization of B such that $B = LL^T$. We consider a method to update L to get $\bar{L}$ such that $\bar{B} = \bar{L}\bar{L}^T$ where $\bar{B}$ is obtained from B by a rank-2 update of the type (3.20).

First suggested by Gill and Murray (1972), several elegant methods (Gill, Golub, Murray and Saunders (1974), Fletcher and Powell (1974)) have been proposed to update the Cholesky factor L when a rank-1 correction is added to B. Since the rank-2 correction term in (3.20) can be expressed as the sum of two symmetric rank-1 terms, these procedures need to be performed twice. Goldfarb (1976) suggested direct methods to carry out such updates on L. Recently, a clever derivation of (3.20) was presented by Dennis (1978b) requiring only a rank-1 Broyden update on L to effectively get a rank-2 update on B. We describe this approach below. For the sake of convenience, we restrict our discussion to the BFGS update formula (3.22).

Since we want the updated matrix $\bar{B}$ to be symmetric, positive definite ($\langle y,s\rangle > 0$) and satisfy the secant equation $\bar{B}s = y$, we seek to find a vector v and a matrix J such that

(A.1) $$v = J^T s \quad \text{and} \quad y = Jv ,$$

and then reduce J by an orthogonal transformation Q to a lower triangular matrix $\bar{L}$ so that

$$\bar{B} = JJ^T = \bar{L}Q^T Q\bar{L}^T = \bar{L}\bar{L}^T$$

implies that $\bar{L}$ is the required Cholesky factor of $\bar{B}$.

Suppose we know v, then J can be obtained by updating L using the rank-1 Broyden formula

(A.2) $$J = L + \frac{(y - Lv)v^T}{\langle v,v\rangle} = L + zv^T \; ; \qquad z = \frac{y - Lv}{\langle v,v\rangle}$$

which satisfies $Jv = y$. Using the other condition $v = J^T s$ of (A.1) and (A.2) we obtain

(A.3) $$v = \sqrt{\frac{y^T s}{s^T LL^T s}}\; L^T s \; .$$

It is easy to verify that $\bar{B} = JJ^T$ is exactly the BFGS update on $B(= LL^T)$.

In order to reduce J to a lower triangular matrix, note that

(A.4) $$J = L(I + \tilde{z}v^T) \; ; \quad \text{where} \quad L\tilde{z} = z \; .$$

Goldfarb (1976) suggested two efficient methods using 1) Givens' plane rotations and 2) Householder's transformations to determine orthogonal matrix Q such that $(I + \tilde{z}v^T)Q = \tilde{L}$, where $\tilde{L}$ is lower triangular. Thus the required $\bar{L}$ matrix is given by (using A.4)

(A.5) $$\bar{L} = L\tilde{L} \; .$$

Given L, s and y we summarize the algorithmic approach for updating $L=(\ell_{ij})$. Since Goldfarb's method based on Householder's transformations seems to require slightly fewer computations, we adopt this to triangularize $I+\tilde{z}v^T$ in the algorithm below. As for storage, it requires three extra vectors as work space.

Step 1. Set $w:=L^Ts$ and then $v:=\sqrt{y^Ts/w^Tw}\cdot w$

Step 2. If $|v_i|<\tau\|v\|$, $1\leq i\leq n$ consider $v=0$ and hence there is is no change in L, thus terminate. Otherwise, determine $m=\max\{i;\,|v_i|\geq\tau\|v\|\}$; and set $w_m:=0$. If $m=1$, go to step 4 .

Step 3. For $j=m,m-1,\ldots,2$; set $w_{j-1}:=w_j+v_j^2$

Step 4. Compute $z:=(y-Lv)/(w_1+v_1^2)$

Step 5. Set $z_1:=z_1/\ell_{11}$

For $i=2,\ldots,n$; set $z_i:=\left(z_i-\sum_{j=1}^{i-1}\ell_{ij}z_j\right)\Big/\ell_{ii}$

Set $\xi:=1$, $\eta:=0$ and $j:=1$. If $m=1$ go to step 8.

Step 6. Set, in order, $\delta:=\xi z_j-\eta v_j$

$$\theta:=1+\delta v_j$$

$$\gamma:=-(\mathrm{sgn}(\theta))\sqrt{\theta^2+\delta^2w_j}$$

$$\mu:=\theta v_j+\delta w_j$$

$$\sigma:=\mu\xi/\gamma$$

$$\mu:=(\delta-\mu\eta)/\gamma$$

$$\xi:=-\xi/\gamma$$

$$\eta:=-(\eta+\delta^2/(\theta-\gamma))/\gamma$$

```
Step 7.  For $i = j, j+1, \dots n$ ; set $\ell_{ij} := \ell_{ij}\gamma$ .
         For $k = j+1, j+2, \dots, n$ ; do the following sequence:
                 Set $\theta := \mu v_k + \sigma z_k$
                 For $i = k, k+1, \dots, n$ ; set $\ell_{ij} := \ell_{ij} + \ell_{ik}\theta$ .
         If $j = m$, updating of L is complete; terminate.
         Otherwise set $j := j+1$ . If $j < m$, go to step 6 .
Step 8.  Set $\gamma = 1 + (\xi z_m - \eta v_m)v_m$ . If $m < n$; set $\mu = 0$, $\sigma = \xi v_m$ ,
         and go to step 7. Otherwise set $\ell_{nn} := \ell_{nn}\gamma$; and terminate.
```

Note that $v = 0$ if and only if $s = 0$ because L is nonsingular. Thus, checking $\|L^T s\|$ in step 1 is recommended to avoid division by zero. Observe that step 2 takes care of the possibility $v_i = 0$ for $m+1 \le i \le n$ and $v_m \neq 0$, in which case the last $n - m$ columns of L are unchanged. (Recall that τ is the machine tolerance.) Step 5 solves $L\tilde{z} = z$ for $\tilde{z}$. Step 6, repeated $m - 1$ times, reduces $I + \tilde{z}v^T$ and follows directly from Method 2 of Goldfarb (1976). Step 7 multiples $L\tilde{L}$ as a new column of $\tilde{L}$ is generated.

REFERENCES

1. J. Abadie and J. Carpentier,"Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints," in Optimization, ed. R. Fletcher, Academic Press, London (1969).

2. L. Armijö, "Minimizations of Functions Having Lipschitz-continuous First Partial Derivatives," Pacific Journal of Mathematics, vol. 16, pp. 1-3 (1966).

3. M. Avriel, Nonlinear Programming--Analysis and Methods, Prentice Hall, New Jersey (1975).

4. O. Axelsson, "On Preconditioning and Convergence Acceleration in Sparse Matrix Problems" CERN Report 74-10, European Organization for Nuclear Research, Geneva (1974).

5. O. Axelsson, "A class of Iterative Methods for Finite Element Equations," Report 75-03R, Dept. of Computer Science, Chalmers University of Technology, Göteborg (1975)

6. Y. Bard and J. Greenstadt, "A Modified Newton Method for Optimization with Equality Constraints," in Optimization, ed. R. Fletcher, Academic Press, London (1969).

7. Bradley M. Bell, "An Improved Line Search Technique," SIGMAP Bulletin, No. 25, pp. 33-38 (1978).

8. M. Bertocchi, E. Covalli and E. Spedicato, "Computational Experience with Diagonalized Multiplier Quasi-Newton Methods for Nonlinear Optimization with Equality Constraints," Technical Report, Universitȁ de Bergamo, Italy (1978).

9. M. J. Best, "A Quasi-Newton Method can be Obtained from a Method of Conjugate Directions, Mathematical Programming, vol. 15, pp. 189-199 (1978).

10. M. C. Biggs, "Constrained Minimization Using Recursive Equality Quadratic Programming," in Numerical Methods for Nonlinear Optimization, ed. F. A. Lootsma, Academic Press, London (1972).

11. M. C. Biggs, "Constrained Minimization Using Recursive Quadratic Programming: Some Alternative Subproblem Formulations," in Toward Global Optimization, eds. L.C.W. Dixon and G.P. Szegö, North Holland Publishing Co., Amsterdam (1975).

12. M. C. Biggs, "A Numerical Comparison Between Two Approaches to the Nonlinear Programming Problems," Technical Report No. 77, Numerical Optimization Centre, The Hatfield Polytechnic, Hertfordshire, England (1976).

13. M. C. Biggs, "On the Convergence of Some Constrained Minimization Algorithms Based on Recursive Quadratic Programming," *J. Inst. Maths. Applics*. 21, pp. 67-81 (1978).

14. J. L. Blue, "Robust Algorithms for Solving Systems of Nonlinear Equations," Technical Report, Bell Laboratories, Murray Hill, New Jersey (1977).

15. C. G. Broyden, "A Class of Methods for Solving Nonlinear Simultaneous Equations," *Math. Comp*., Vol. 19, pp. 577 - 593 (1965).

16. C. G. Broyden, "Quasi-Newton Methods and Their Application to Function Minimization," *Math. Comp*., Vol. 21, pp. 368-381 (1967).

17. C. G. Broyden, "A New Double-rank Minimization Algorithm," *Notices Amer. Math. Soc*., Vol. 16, p. 670 (1969).

18. C. G. Broyden, "The Convergence of a Class of Double-rank Minimization Algorithms," Parts I and II, *J. Inst. Math. Appl*., Vol. 6, pp. 76-90 and 222-231 (1970).

19. C. G. Broyden, "The Convergence of an Algorithm for Solving Sparse Nonlinear Systems," Math. Comp., Vol. 25, pp 285-294 (1971).

20. J. R. Bunch and L. Kaufman, "Indefinite Quadratic Programming," Computing Science Technical Report No. 61, Bell Laboratories, Murray Hill, New Jersey (1977).

21. J. R. Bunch and B. N. Parlett, "Direct Methods for Solving Symmetric Indefinite Systems of Linear Equations," SIAM J. Numer. Anal., Vol. 8, pp. 639-655 (1971).

22. J. D. Buys, "Dual Algorithms for Constrained Optimization," Ph.D. thesis, University of Leiden, Leiden, The Netherlands (1972).

23. R. H. Byrd, "Local Convergence of the Diagonalized Method of Multipliers," *JOTA*, to appear.

24. A. Cauchy, "Méthode Générale pour la Resolution des Systémes D'équations Simultanées," Compt. Rend., Vol. 25, pp. 536-538 (1947).

25. P. Concus, G. H. Golub and D. P. O'Leary, "A Generalized Conjugate Gradient Method for the Numerical Solution of Elliptic Partial Differential Equations, in *Sparse Matrix Computations*, ed. J. R. Bunch and D. J. Rose, Academic Press, New York (1976).

26. R. Courant, "Variational Methods for the Solution of Problems of Equilibrium and Vibrations," Bull. Amer. Math. Soc., Vol. 49, pp. 1-23 (1943)

27. H. B. Curry, "The Method of Steepest Descent for Nonlinear Minimization Problems," Quart. Appl. Math., Vol. 2, pp. 258-261 (1944).

28. W. C. Davidon, " Variable Metric Method for Minimization," Report ANL-5990 Rev., Argonne National Laboratories, Argonne, Illinois (1959).

29. W. C. Davidon, "Optimally Conditioned Optimization Algorithms Without Line Searches," Mathematical Programming, Vol. 9, pp. 1-30 (1975).

30. J. E. Dennis, Jr., "A Brief Introduction to Quasi-Newton Methods," in Numerical Analysis, ed. Gene H. Golub and Joseph E. Oligar, American Mathematical Society, Providence, Rhode Island (1978a).

31. J. E. Dennis, Jr., "Variable Metric Secant Updates from Matrix Factorizations for Sparse Problems," Techincal paper in preparation, Department of Computer Science, Cornell University, Ithaca, New York (1978b).

32. J. E. Dennis, Jr. and H. H. W. Mei, "An Unconstrained Optimization Algorithm Which Uses Function and Gradient Values," Technical Report TR75-246, Cornell University, Ithaca, New York (1975).

33. J. E. Dennis, Jr. and Jorge J. More, "Quasi-Newton Methods, Motivation and Theory," SIAM Review, Vol. 19, No. 1, pp. 46-89. (1977).

34. L. C. W. Dixon, "Variable Metric Algorithms: Necessary and Sufficient Conditions for Identical Behavior on Nonquadratic Functions," J. of Optimization Theory and Applications, Vol. 10, No. 1, pp. 34-40 (1972a)

35. L. C. W. Dixon, "Quasi-Newton Algorithms Generate Identical Points," Math. Prog., Vol. 2, pp. 383-387(1972b).

36. L. C. W. Dixon, "Nonlinear Optimization: A Survey of the State of the Art," in Software for Numerical Mathematics, ed. by D. J. Evans, Academic Press, London (1974).

37. L. C. W. Dixon, "Nonlinear Programming: A View of the State of the Art," in Towards Global Optimization, ed. L. C. W. Dixon and G. P. Szegö, North-Holland Publishing Co., Amsterdam (1975).

38. J. Douglas, Jr. and T. Dupont, "Preconditioned Conjugate Gradient Iteration Applied to Galerkin Methods for a Mildly Nonlinear Dirichlet Problem," in Sparse Matrix Computations, ed. J. R. Bunch and D. J. Rose, Academic Press, New York (1976).

39. A. V. Fiacco and G. P. McCormick, "Nonlinear Programming: Sequential Unconstrained Minimization Techniques," Wiley, New York, (1968).

40. R. Fletcher, "Function Minimization Without Calculating Derivatives--A Review," Computer Journal, Vol. 8, pp. 33-41 (1965).

41. R. Fletcher, "A New Approach to Variable Metric Algorithms," Computer Journal 13, 317-322 (1970a).

42. R. Fletcher, "A Class of Methods for Nonlinear Programming With Termination and Convergence Properties," in Integer and Nonlinear Programming, ed. J. Abadie, North-Holland Publishing Co., Amsterdam (1970b).

43. R. Fletcher, "A Survey of Algorithms for Unconstrained Optimization, in Numerical Methods for Unconstrained Optimization, ed. W. Murray, Academic Press, London (1972).

44. R. Fletcher, "An Exact Penalty Function for Nonlinear Programming with Inequalities," Math. Prog., 5, pp. 129-150 (1973).

45. R. Fletcher, "Methods for Solving Nonlinearly Constrained Optimization Problems," in The State of the Art in Numerical Analysis, ed. D. Jacobs, Academic Press, London (1977).

46. R. Fletcher and T. L. Freeman, "A Modified Newton Method for Minimization," Technical Report No. 7, Department of Mathematics, University of Dundee, U.K. (1975).

47. R. Fletcher and M. J. D. Powell, "A Rapidly Convergent Descent Method for Minimization," Computer Journal, Vol. 6, No. 2, pp. 163-168 (1963).

48. R. Fletcher and M. J. D. Powell, "On the Modification of LDL^T Factorizations," Math. of Comp., Vol. 28, No. 128, pp. 1067-1087 (1974).

49. R. Fletcher and C. M. Reeves, "Function Minimization by Conjugate Gradients," Computer Journal, Vol. 7, No. 2, pp. 149-154 (1964).

50. G. E. Forsythe and C. B. Moler, Computer Solution of Linear Algebraic Systems, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1967).

51. R. L. Fox, L. S. Lasdon, A. Tamir and M. Ratner, "An Efficient One-Dimensional Search Procedure," Management Science, Vol. 22, No. 1, pp. 42-50 (1975).

52. P. E. Gill, G. H. Golub, W. Murray and M. A. Saunders, "Methods for Modifying Matrix Factorizations," Math. of Comp., Vol. 28, No. 126, pp. 505-535 (1974).

53. P. E. Gill and W. Murray, "Quasi-Newton Methods for Unconstrained Optimization," J. Inst. Maths. Applics, Vol. 9, pp. 91-108 (1972).

54. P. E. Gill and W. Murray, eds., Numerical Methods for Constrained Optimization," Academic Press, London (1974a).

55. P. E. Gill and W. Murray, "Safeguarded Steplength Algorithms for Optimization Using Descent Methods," National Physical Laboratory Technical Report NAC37, Division of Numerical Analysis and Computing, Teddington, Middlesex, U.K. (1974b).

56. P. E. Gill, W. Murray and R. A. Pitfield, "The Implementation of Two Revised Quasi-Newton Algorithms for Unconstrained Optimization," Report NAC 11, National Physical Laboratory, Teddington, England (1972).

57. S. T. Glad, "Properties of Updating Methods for the Multipliers in Augmented Lagrangians," Ph.D. thesis, University of Lund, Lund, Sweden (1976).

58. A. N. Gleyzal, "Solution of Nonlinear Equations," Quart. Appl. Math., Vol. 17, pp. 95-96 (1959).

59. Donald Goldfarb, "A Family of Variable-Metric Methods Derived by Variational Means," Math. Comp., 24, pp. 23-26 (1970).

60. Donald Goldfarb, "Factorized Variable Metric Methods for Unconstrained Optimization," Math. Comp., Vol. 30, No. 136, pp. 796-811 (1976).

61. Donald Goldfarb, "Generalized Conjugate Directions Without Line Searches Using Factorized Variable Metric Updating Formulas," Mathematical Programming, Vol. 13, pp. 94-110 (1977).

62. S. M. Goldfeld, R. E. Quandt and H. F. Trotter, "Maximization by Quadratic Hill-Climbing," Econometrica, Vol. 34, pp. 541-551 (1966).

63. A. A. Goldstein, "Cauchy's Method for Minimization," Numer. Math., Vol. 4, pp. 146-150 (1962).

64. A. A. Goldstein and J. F. Price, "An Effective Algorithm for Minimization," Numer. Math., Vol. 10, pp. 184-189 (1967).

65. G. H. Golub, "Numerical Method for Solving Linear Least Squares Problems," Numer. Math., Vol. 7, pp. 206-216 (1965).

66. G. H. Golub and C. Reinsch, "Singular Value Decomposition and Least Squares Solutions," Numer. Math., Vol. 14, pp. 403-420 (1970).

67. John Greenstadt, "Revision of a Derivative-Free Quasi-Newton Method, Math. Comp., Vol. 32, No. 141, pp. 201-221 (1978).

68. S. P. Han, "Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems," Math. Prog. Vol. 11, pp. 263-282 (1976).

69. S. P. Han, "Dual Variable Metric Algorithms for Constrained Optimization," SIAM J. of Control and Optimization, Vol. 15, No. 4, pp. 546-565 (1977a).

70. S. P. Han, "A Globally Convergent Method for Nonlinear Programming," JOTA, Vol. 22, No. 3, pp. 297-309 (1977b).

71. S. P. Han, "A Hybrid Method for Nonlinear Programming," in Nonlinear Programming 3, ed. O.L. Mangasarian, R. R. Meyer and S. M. Robinson, Academic Press, New York (1978).

72. S. P. Han and O. L. Mangasarian, "Exact Penalty Functions in Nonlinear Programming," Technical Summary Report #1897, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin (1978).

73. M. T. Heath, "Numerical Algorithms for Nonlinearly Constrained Optimization," Technical Report STAN-CS-78-656, Computer Science Department, Stanford University, Stanford, California, (1978).

74. M. R. Hestenes, "The Conjugate-Gradient Method for Solving Linear Systems," Proceedings of Symposium in Applied Mathematics, Vol. 6, Numerical Analysis, pp. 83-102 (1956).

75. M. R. Hestenes, "Multiplier and Gradient Methods," JOTA, Vol. 4, pp. 303-320 (1969).

76. M. R. Hestenes and Eduard Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," Journal of Research of the National Bureau of Standards, Vol. 49, No. 6, pp. 409-436 (1952).

77. D. M. Himmelblau, Applied Nonlinear Programming, McGraw Hill Book Co., New York (1972).

78. A. S. Householder, The Theory of Matrices in Numerical Analysis, Blaisdell Publishing Company, New York (1964).

79. R. Klessig and E. Polak, "Efficient Implementations of the Polak-Ribiere Conjugate Gradient Algorithm," SIAM J. Control, Vol. 10, pp. 524-549 (1972).

80. L. S. Lasdon, A. D. Waren, A. Jain and M. Ratner, "Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming," ACM Transactions on Mathematical Software, Vol. 4, No. 1, pp. 34-50 (1978).

81. C. L. Lawson and R. J. Hanson, Solving Least Squares Problems, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1974).

82. M. L. Lenard, "Accelerated Conjugate Direction Methods for Unconstrained Optimization," JOTA, Vol. 25, No. 1, pp. 11-31 (1978).

83. K. Levenberg, "A Method for the Solution of Certain Nonlinear Problems in Least Squares," Quart. Appl. Math., Vol. 2, pp. 164-168 (1944).

84. F. A. Lootsma, "A Survey of Methods for Solving Constrained Minimization Problems via Unconstrained Minimization," in Numerical Methods for Nonlinear Optimization, ed. F. A. Lootsma, Academic Press, London (1972).

85. D. G. Luenberger, Introduction to Linear and Nonlinear Programming, Addison-Wesley Publishing Co., California (1973).

86. O. L. Mangasarian, Nonlinear Programming, McGraw-Hill, New York (1969).

87. O. L. Mangasarian, "Unconstrained Methods in Nonlinear Programming," in Nonlinear Programming, ed. R. W. Cottle and C. E. Lemke, SIAM-AMS Proc., Vol. 9, pp. 169-184 (1976).

88. D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," J. Soc. Indust. Appl. Math., Vol. 11, pp. 431-441 (1963).

89. Earl Marwil, "Local and Linear Convergence of an Algorithm for Solving a Sparse Minimization Problem," Technical Report, Dept. of Computer Science, Cornell University, Ithaca, New York (1978).

90. G. P. McCormick, "A Modification of Armijö's Step-Size Rule for Negative Curvature," Mathematical Programming, Vol. 13, pp. 111-115 (1977).

91. A. Miele, E. E. Cragg, R. R. Iyer and A. V. Levy, "Use of the Augmented Penalty Function in Mathematical Programming Problems," Parts 1 and 2, JOTA, Vol. 8, pp. 115-153 (1971).

92. A. Miele, H. Y. Huang and J. C. Heideman, "Sequential Gradient-Restoration Algorithm for the Minimization of Constrained Functions--Ordinary and Conjugate Gradient Versions," JOTA, Vol. 4, pp. 213-243 (1969).

93. J. J. Moré, B. S. Garbow and K. E. Hillstrom, "Testing Unconstrained Optimization Software," Technical Report-324, Applied Mathematics Division, Argonne National Laboratory, Argonne, Illinois (1978).

94. J. J. Moré and D. C. Sorensen, "On the Use of Directions of Negative Curvature in a Modified Newton Method," Mathematical Programming, Vol. 16, pp. 1-20 (1979).

95. Walter Murray, "Constrained Optimization," National Physical Laboratory Report MA79, Division of Numerical Analysis and Computing, Teddington, Middlesex, U.K. (1969).

96. Walter Murray, editor, Numerical Methods for Unconstrained Optimization, Academic Press, London (1972).

97. Walter Murray, "Methods for Constrained Optimization," in Optimization in Action, ed. L. C. W. Dixon, Academic Press, London (1976).

98. Geraldine E. Myers, "Properties of the Conjugate Gradient and Davidon Methods," JOTA, Vol. 2, No. 4, pp. 209-219 (1968).

99. L. Nazareth, "A Conjugate Direction Algorithm Without Linear Searches," JOTA, Vol. 23, No. 3, pp. 373-387 (1977a).

100. L. Nazareth, "A Relationship Between the BFGS and Conjugate Gradient Algorithms," ANL-AMD Technical Memo 282 (Rev.), Applied Mathematics Division, Argonne National Laboratory, Argonne, Illinois (1977b).

101. L. Nazareth, "A FORTRAN Implementation of Davidon's Optimally Conditioned Method," Technical Report TM-306, Argonne National Laboratory, Argonne, Illinois (1977c).

102. L. Nazareth and J. Nocedal, "Properties of Conjugate Gradient Methods with Inexact Linear Searches," Technical Report SOL 78-1, Dept. of Operations Research, Stanford University, Stanford, California (1978).

103. J. Nocedal, "On the Method of Conjugate Gradients for Function Minimization," Ph.D. Thesis, Dept. of Mathematical Sciences, Rice University, Houston, Texas (1978).

104. S. S. Oren, "Self-Scaling Variable Metric Algorithms Without Line Search for Unconstrained Minimization," Math. Comp., Vol. 27, No. 124, pp. 873-885 (1973).

105. S. S. Oren and E. Spedicato, "Optimal Conditioning of Self-Scaling Variable Metric Algorithms," Math. Prog., Vol. 10, pp. 70-90 (1976).

106. J. M. Ortega and W. C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York (1970).

107. C. C. Paige and M. A. Saunders, "Solution of Sparse Indefinite Systems of Linear Equations," SIAM J. Numer. Anal., Vol. 12, No. 4, pp. 617-629 (1975).

108. U. M. Garcia Palomares and O. L. Mangasarian, "Superlinearly Convergent Quasi-Newton Algorithms for Nonlinearly Constrained Optimization Problems," Math. Prog., Vol. 11, pp. 1-13 (1976).

109. D. A. Pierre and M. J. Lowe, Mathematical Programming via Augmented Lagrangians, Addison-Wesley Publishing Co., Reading, Massachusetts (1975).

110. M. J. D. Powell, "A Method for Nonlinear Constraints in Minimization Problems," in Optimization, ed. R. Fletcher, Academic Press, London, pp. 283-298 (1969).

111. M. J. D. Powell, "A New Algorithm for Unconstrained Optimization," in Nonlinear Programming, ed. J. B. Rosen, O. L. Mangasarian and K. Ritter, Academic Press, New York (1970).

112. M. J. D. Powell, "Recent Advances in Unconstrained Optimization," Math. Prog., Vol. 1, pp. 26-57 (1971).

113. M. J. D. Powell, "A View of Unconstrained Optimization," in Optimization in Action, ed. L.C.W. Dixon, Academic Press, London (1976).

114. M. J. D. Powell, "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," presented at the Dundee Conference on Numerical Analysis (1977a).

115. M. J. D. Powell, "Variable Metric Methods for Constrained Optimization," presented at the Third International Symposium on Computing Methods in Applied Sciences and Engineering, Paris (1977b).

116. M. J. D. Powell, "Algorithms for Nonlinear Constraints that Use Lagrangian Functions," Mathematical Programming, Vol. 14, pp. 224-248 (1978a).

117. M. J. D. Powell, "The Convergence of Variable Metric Methods for Nonlinearly Constrained Optimization Calculations," in Nonlinear Programming 3, ed. O. L. Mangasarian, R. R. Meyer and S. M. Robinson, Academic Press, New York (1978b).

118. S. M. Robinson, "A Quadratically-Convergent Algorithm for General Nonlinear Programming Problems," Mathematical Programming, Vol. 3, pp. 145-156 (1972).

119. S. M. Robinson, "Perturbed Kuhn-Tucker Points and Rates of Convergence for a Class of Nonlinear Programming Algorithms," Mathematical Programming, Vol. 7, pp. 1-16 (1974).

120. J. B. Rosen, "The Gradient Projection Method for Nonlinear Programming. Part I: Linear Constraints," SIAM Journal, Vol. 8, pp. 181-217 (1960).

121. J. B. Rosen, "The Gradient Projection Method for Nonlinear Programming. Part II: Nonlinear Constraints," SIAM Journal, Vol. 9, pp. 514-532 (1961).

122. J. B. Rosen, "Two-phase Algorithm for Nonlinearly Constraint Problems," in Nonlinear Programming 3, ed. O. L. Mangasarian, R. R. Meyer and S. M. Robinson, Academic Press, New York (1978).

123. J. B. Rosen and J. Kreuser, "A Gradient Projection Algorithm for Nonlinear Constraints," in Numerical Methods for Nonlinear Optimization, ed. F. A. Lootsma, Academic Press, London (1972).

124. Klaus Schittkowski, "A Numerical Comparison of Optimization Software Using Randomly Generated Test Problems--an Intermediate Balance," Preprint #43, Institut für Angewandte Mathematik und Statistik, Universität Würzburg, Am Hubland, Würzburg, West Germany (1978).

125. L. K. Schubert, "Modification of a Quasi-Newton Method for Nonlinear Equations with a Sparse Jacobian," Math. Comp., Vol. 24, pp. 27-30 (1970).

126. D. F. Shanno, "Conditioning of Quasi-Newton Methods for Function Minimization," Math. Comp., Vol. 24, pp. 647-656 (1970).

127. D. F. Shanno, "Conjugate Gradient Methods with Inexact Searches," Math. of Oper. Res., Vol. 3, No. 3, pp. 244-256 (1978a).

128. D. F. Shanno, "On the Convergence of a New Conjugate Gradient Algorithm," SIAM J. Num. Analysis, Vol. 15, No. 6, pp. 1247-1257 (1978b).

129. D. F. Shanno, "On Variable-Metric Methods for Sparse Hessians," MIS Technical Report No. 26, College of Business and Public Administration, University of Arizona, Tucson, Arizona (1978c).

130. D. F. Shanno and K. H. Phua, "Minimization of Unconstrained Multivariate Functions," ACM Transactions on Mathematical Software, Vol. 2, No. 1, pp. 87-94 (1976).

131. D. F. Shanno and K. H. Phua, "Matrix Conditioning and Nonlinear Optimization," Math. Prog., Vol. 14, No. 2, pp. 149-160 (1978a).

132. D. F. Shanno and K. H. Phua, "A Variable Method Subroutine for Unconstrained Nonlinear Minimization," MIS Technical Report No. 28, College of Business and Public Administration, University of Arizona, Tuscon, Arizona (1978b).

133. E. Spedicato, "A Note on the Determination of the Scaling Parameters in a Class of Quasi-Newton Methods for Unconstrained Minimization," Technical Research Report, CISE, Segrate, Italy (1976).

134. E. Spedicato, "On a Conjecture of Dixon and Other Topics in Variable Metric Methods," Math. Prog., Vol. 15, pp. 123-129 (1978).

135. G. W. Stewart, Introduction to Matrix Computations, Academic Press, New York (1973).

136. R. A. Tapia, "The Differentiation and Integration of Nonlinear Operators," in Nonlinear Functional Analysis and Applications, ed. L. B. Rall, Academic Press, New York (1971).

137. R. A. Tapia, "Newton's Method for Optimization Problems with Equality Constraints," SIAM J. of Num. Anal., Vol. 11, pp. 174-196, 874-886 (1974a).

138. R. A. Tapia, "A Stable Approach to Newton's Method for General Mathematical Programming Problems in R^n," JOTA, Vol. 14, No. 5, pp. 453-476 (1974b).

139. R. A. Tapia, "Diagonalized Multiplier Methods and Quasi-Newton Methods for Constrained Optimization," JOTA, Vol. 22, pp. 135-194 (1977).

140. R. A. Tapia, "Quasi-Newton Methods for Equality Constrained Optimization: Equivalence of Existing Methods and a New Implementation," in Nonlinear Programming 3, ed. O. L. Mangasarian, R.R. Meyer and S. M. Robinson, Academic Press, New York (1978).

141. Ph. L. Toint, "On Sparse and Symmetric Matrix Updating Subject to a Linear Equation," Math. Comp., Vol. 31, No. 140, pp. 954-961 (1977).

142. Ph.L. Toint, "Some Numerical Results Using a Sparse Matrix Updating Formula in Unconstrained Optimization," Math Comp., Vol. 32, pp. 839-852 (1978).

143. J. H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, London (1965).

144. R. B. Wilson, A Simplicial Algorithm for Concave Programming, Ph. D. Dissertation, Harvard University, Cambridge, Mass. (1963).

145. P. Wolfe, "Methods of Nonlinear Programming," in Recent Advances in Mathematical Programming, ed. R. L. Graves and P. Wolfe, McGraw-Hill Book Co., New York (1963).

146. P. Wolfe, "Convergence Conditions of Ascent Methods," SIAM Review, Vol. 11, pp. 226-235 (1969).

147. P. Wolfe, "Convergence Conditions of Ascent Methods II: Some Corrections," SIAM Review, Vol. 13, pp. 185-188 (1971).

148. W. I. Zangwill, Nonlinear Programming: A Unified Approach, Prentice Hall, Inc., Englewood Cliffs, New Jersey (1969).