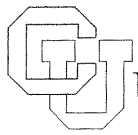
A View of Unconstrained Optimization

J. E. Dennis Jr.
Robert B. Schnabel

CU-CS-376-87



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

A View of Unconstrained Optimization

J.E. Dennis Jr. *
Robert B. Schnabel **

CU-CS-376-87 October 1987

*Department of Mathematical Sciences, Rice University, Houston Texas 772251. Research supported by DOE DE-FG05-86ER25017, SDIO/IST/ARO DAAG-03-86-K-0113 and AFOSR 85-0243.

**Department of Computer Science, Campus Box 430, University of Colorado, Boulder, Colorado, 80309. Research supported by ARO contract DAAG 29-84-K-0140 and NSF grant CCR-870243.

To appear in *Handbooks in Operations Research and Management Science, Vol. 1, Optimization,* G.L. Nemhauser, A.H.G. Rinooy Kan, M.J. Todd, eds., North-Holland, Amsterdam.

A View of Unconstrained Optimization

Abstract

Finding the unconstrained minimizer of a function of more than one variable is an important problem with many practical applications, including data fitting, engineering design, and process control. In addition, techniques for solving unconstrained optimization problems form the basis for most methods for solving constrained optimization problems. This paper surveys the state of the art for solving unconstrained optimization problems and the closely related problem of solving systems of nonlinear equations. First we briefly give some mathematical background. Then we discuss Newton's method, the fundamental method underlying most approaches to these problems, as well as the inexact Newton method. The two main practical deficiencies of Newton's method, the need for analytic derivatives and the possible failure to converge to the solution from poor starting points, are the key issues in unconstrained optimization and are addressed next. We discuss a variety of techniques for approximating derivatives, including finite difference approximations, secant methods for nonlinear equations and unconstrained optimization, and the extension of these techniques to solving large, sparse problems. Then we discuss the main methods used to ensure convergence from poor starting points, line search methods and trust region methods. Next we briefly discuss two rather different approaches to unconstrained optimization, the Nelder-Meade simplex method and conjugate direction methods. Finally we comment on some current research directions in the field, in the solution of large problems, the solution of data fitting problems, new secant methods, the solution of singular problems, and the use of parallel computers in unconstrained optimization.

1. Preliminaries

1.1 Introduction

This chapter discusses the most basic nonlinear optimization problem in continuous variables, the *unconstrained optimization* problem. This is the problem of finding the minimizing point of a nonlinear function of n real variables, and it will be denoted

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; f : \mathbb{R}^n \to R \quad . \tag{1.1}$$

It will be assumed that $f(x)$ is at least twice continuously differentiable.

The basic methods for unconstrained optimization are most easily understood through their relation to the basic methods for a second nonlinear algebraic problem, the *nonlinear equations* problem. This is the problem of finding the simultaneous solution of n nonlinear equations in n unknowns, denoted

$$\text{given } F : \mathbb{R}^n \to \mathbb{R}^n \;, \quad \text{find } x_* \text{ for which } F(x_*) = 0 \tag{1.2}$$

where $F(x)$ is assumed to be at least once continuously differentiable. Therefore, this chapter also will discuss the nonlinear equations problem as necessary for understanding unconstrained optimization.

Unconstrained optimization problems arise in virtually all areas of science and engineering, and in many areas of the social sciences. In our experience, a significant percentage of real-world unconstrained optimization problems are data fitting problems (see Section 6.2). The size of real-world unconstrained optimization problems is widely distributed, varying from small problems, say with n between 2 and 10, to large problems, say with n in the hundreds or thousands. In many cases, the objective function $f(x)$ is a computer routine that is expensive to evaluate, so that even small problems often are expensive and difficult to solve.

The user of an unconstrained optimization method is expected to provide the function $f(x)$ and a starting guess to the solution, x_0. The routine is expected to return an estimate of a *local minimizer* x_* of $f(x)$, the lowest point in some open subregion of $\mathbb{R}^n$. The user optionally may provide routines for evaluating the first and second partial derivatives of $f(x)$, but in most cases they are not provided and instead are approximated in various ways by the algorithm. Approximating these derivatives is one of the main challenges of creating unconstrained optimization methods. The other main challenge is to create

methods that will converge to a local minimizer even if x_0 is far from any minimum point. This is referred to as the *global* phase of the method. The part of the method that converges quickly to x_*, once it is close to it, is referred to as the *local* phase of the method.

This emphasis of this chapter is on modern, efficient methods for solving unconstrained optimization problems, with particular attention given to the main areas of difficulty mentioned above. Since function evaluation so often is expensive, the primary measure of efficiency often is the number of function (and derivative) evaluations required. For problems with large numbers of variables, the number of arithmetic operations required by the method itself (aside from function evaluations) and the storage requirements of the method become increasingly important.

The remainder of this section reviews some basic mathematics underlying this area and the rest of continuous optimization. Section 2 discusses the basic local method for unconstrained optimization and nonlinear equations, Newton's method. Section 3 discusses various approaches to approximating derivatives when they aren't provided by the user. These include finite difference approximations of the derivatives, and secant approximations, less accurate but less expensive approximations that have proven to lead to more efficient algorithms for problems where function evaluation is expensive. We concentrate on the most successful secant method for unconstrained optimization, the "BFGS" method, and as motivation we also cover the most successful secant method for nonlinear equations, Broyden's method. Sections 2 and 3 cover both small and large dimension problems, although the solution of small problems is better understood and therefore is discussed in more detail. Methods that are used when starting far from the solution, called global methods, are covered in Section 4. The two main approaches, line search methods and trust region methods, both are covered in some detail. In Section 5 we cover two important methods that do not fit conveniently in the Taylor series approach that underlies Sections 2 through 4. These are the Nelder-Mead simplex method, an important method for solving problems in very few variables, and conjugate gradient methods, one of the leading approaches to problems with very many variables. Section 6 briefly surveys a number of current research directions in unconstrained optimization. These include further approaches to solving problems in many variables, special methods for problems arising from data fitting, further issues in secant approximation of derivatives, the solution of problems where the Jacobian or Hessian matrix at the solution is singular, and the use of parallel computers in solving unconstrained optimiza-

tion problems.

The scope of this chapter is limited in many important ways which are then addressed by other chapters of this book. We do not consider *constrained optimization* problems, problems where the domain of permissible solutions is limited to those variables x satisfying one or more equality or inequality constraints. Constrained optimization problems in continuous variables are discussed in Chapter 3; the techniques for solving these problems draw heavily upon the techniques for solving unconstrained problems. We do not consider problems where $f(x)$ is not differentiable; these are discussed in Chapter 9. We do not discuss methods that attempt to find the *global minimizer*, the lowest of the possibly multiple local minimizers of $f(x)$. Methods for finding the global minimizer, called *global optimization* methods, are discussed in Chapter 11. Finally, we do not consider problems where some or all of the variables are restricted to a discrete set of values, for example the integers; these problems must be solved by techniques such as those discussed in Chapters 2, 4, 5, and 6. It should be noted that in most nonlinear optimization problems solved today, the variables are continuous, $f(x)$ is differentiable, and a local minimizer provides a satisfactory solution. This probably reflects available software as well as the needs of practical applications.

A number of books give substantial attention to unconstrained optimization and are recommended to readers who desire additional information on this topic. These include Ortega and Rheinboldt [1970], Fletcher [1980], Gill, Murray, and Wright [1981], and Dennis and Schnabel [1983].

1.2 Taylor Series Models

Most methods for optimizing nonlinear differentiable functions of continuous variables rely heavily upon Taylor series expansions of these functions. This section briefly reviews the Taylor series expansions used in unconstrained (and constrained) optimization and in solving systems of nonlinear equations, and a few mathematical properties of these expansions. The material of this section is developed fully in Ortega and Rheinboldt [1970] or Dennis and Schnabel [1983].

The fundamental Taylor series approximation used in solving a system of nonlinear equations (1.2) is the first two terms of the Taylor series approximation to $F(x)$ around a current point x_c,

$$M_c^N(x) = F(x_c) + J(x_c)(x - x_c). \tag{1.3}$$

Here the notation M_c^N stands for the current Newton *model* because we will see in Section 2 that local methods for solving nonlinear equations are based on Taylor series models of the form (1.3), and Newton's method is based specifically on (1.3). $J(x_c)$ denotes the $n \times n$ *Jacobian* matrix of first partial derivatives of $F(x)$ at x_c; in general,

$$[J(x)]_{ij} = \frac{\partial F_i(x)}{\partial x_j} \ , \ i=1,\cdots,n \ , \ \ j=1,\cdots,n \tag{1.4}$$

where $F_i(x)$ denotes the i^{th} component function of $F(x)$.

The local convergence analysis of methods based upon (1.3) depends on the difference between $F(x)$ and $M_c^N(x)$. This and the next error bounds in this subsection, as well as most convergence results in continuous optimization, use the concept of Lipschitz continuity. A function G of the n-vector x is said to be Lipschitz continuous with constant γ in an open neighborhood $D \subset \mathbb{R}^n$, written $G \in Lip_\gamma(D)$, if for all $x, y \in D$,

$$|||G(x) - G(y)||| \leq \gamma \, ||x - y|| \tag{1.5}$$

where $||\cdot||$ and $|||\cdot|||$ are appropriate norms. Now if $J(x) \in Lip_\gamma(D)$ in an open neighborhood $D \in \mathbb{R}^n$ containing x and x_c, using a vector norm $||\cdot||$ and the associated matrix norm, then it is a standard result that

$$||M_c^N(x) - F(x)|| \leq \frac{\gamma}{2} \, ||x - x_c||^2 . \tag{1.6}$$

This result is similar to the familiar Taylor series with remainder results for functions of one variable but requires no mention of higher derivatives of F.

The fundamental Taylor series expansion used in unconstrained optimization is the first three terms of the Taylor series of $f(x)$ around x_c,

$$m_c^N(x_c + d) = f(x_c) + g(x_c)^T d + \tfrac{1}{2} \, d^T H(x_c) d \ . \tag{1.7}$$

Here $g(x_c)$ denotes the n component *gradient* column vector of first partial derivatives of $f(x)$ evaluated at x_c; in general

$$[g(x)]_j = \frac{\partial f(x)}{\partial x_j} \ , \ \ j=1,\cdots,n \ . \tag{1.8}$$

Also, $H(x_c)$ denotes the $n \times n$ *Hessian* matrix of second partial derivatives of $f(x)$ evaluated at x_c; in general

$$[H(x)]_{ij} = \frac{\partial^2 f(x)}{\partial x_i \, \partial x_j} \quad , \quad i=1,\cdots,n, \; j=1,\cdots,n \; . \tag{1.9}$$

Note that $H(x)$ is symmetric if $f(x)$ is twice continuously differentiable.

If $H(x)$ is Lipschitz continuous with constant γ in an open neighborhood of $\mathbb{R}^n$ containing x and x_c, using the norm $\|\cdot\|$, then it is a standard result that

$$|m_c^N(x) - f(x)| \le \frac{\gamma}{6} \|x - x_c\|^3 \; . \tag{1.10}$$

This result is used in convergence analysis of unconstrained optimization methods.

The standard Taylor series with remainder results from the calculus of one variable also extend to single valued functions of multiple variables. For any direction $d \in \mathbb{R}^n$ there exist $t_1, t_2 \in [0,t]$ for which

$$f(x+d) = f(x) + g(x + t_1 d)^T d \tag{1.11}$$

and

$$f(x+d) = f(x) + g(x)^T d + \tfrac{1}{2} d^T H(x + t_2 d) d \; . \tag{1.12}$$

These results are the keys to the necessary and sufficient conditions for unconstrained minimization that we consider next.

1.3 Necessary and Sufficient Conditions for Unconstrained Optimization

Algorithms for solving the unconstrained minimization problem are based upon the first and second order conditions for a point x_* to be a local minimizer of $f(x)$. These conditions are briefly reviewed in this section.

For x_* to be a local minimizer of $f(x)$, it is necessary that $g(x_*) = 0$, where $g(x)$ denotes the gradient defined in (1.8).

Theorem 1.1 Let $f(x) : \mathbb{R}^n \to R$ be continuously differentiable, and let $y \in \mathbb{R}^n$. If $g(y) \neq 0$, then y is not a local minimizer of $f(x)$.

Proof: If $g(y) \neq 0$, then there exist directions $d \in \mathbb{R}^n$ for which $g(y)^T d < 0$; an example is $d = -g(y)$. For any such direction d, we have from (1.11) that

$$f(y + td) - f(y) = t \cdot d^T g(y + t_1 d) \tag{1.13}$$

for some $t_1 \in (0,t)$. Also by the continuity of $f(x)$, there exists $\delta > 0$ such that $g(y + t_1 d)^T d < 0$ for any

$t_1 \in [0,\delta]$. Thus for any stepsize $t<\delta$,

$$f(y+td) < f(y). \tag{1.14}$$

Therefore y cannot be a local minimizer of $f(x)$.

Directions d for which $g(y)^T d < 0$ are called descent directions for f at y. Descent directions play an important role in the global methods for unconstrained optimization discussed in Section 4.

The above argument with $d = g(x_*)$ also shows that $g(x_*) = 0$ is necessary for x_* to be a local maximizer of $f(x)$. To distinguish between minimizers and maximizers it is necessary to consider the second derivative matrix $H(x_*)$ defined in (1.9). First we need the definition of a *positive definite* matrix.

Definition 1.1 Let $H \in \mathbb{R}^{n\times n}$ be symmetric. Then H is positive definite if $v^T H v > 0$ for all nonzero $v \in \mathbb{R}^n$.

There are several equivalent characterizations of positive definite matrices; another common one is that a symmetric matrix H is positive definite if and only if all of its eigenvalues are positive. If $v^T H v \geq 0$ for all v, H is said to be *positive semi-definite*. A negative definite or negative semi-definite matrix is one whose negative is positive definite or positive semi-definite.

Theorem 1.2 Let $f(x) : \mathbb{R}^n \to R$ be twice continuously differentiable, and let $x_* \in \mathbb{R}^n$. If $g(x_*)=0$ and $H(x_*)$ is positive definite, then x_* is a local minimizer of $f(x)$.

Proof : By (1.12) for any $d \in \mathbb{R}^n$,

$$f(x_*+d) = f(x_*) + g(x_*)^T d + \tfrac{1}{2} d^T H(x_*+td)d \tag{1.15}$$

for some $t \in (0,1)$. By the continuity of $f(x)$ and the positive definiteness of $H(x_*)$, there exists $\delta>0$ such that for any direction d with $\|d\|<\delta$ and any scalar t with $|t|\leq 1$, $H(x_*+td)$ is positive definite. Thus for any d with $\|d\|<\delta$, we have from (1.15) and $g(x_*)=0$ that

$$f(x_*+d) > f(x_*). \tag{1.16}$$

Therefore x_* is a local minimizer of $f(x)$.

By a similar argument it is easy to show that a necessary condition for x_* to be a local minimizer of a twice continuously differentiable $f(x)$ is that $g(x_*) = 0$ and $H(x_*)$ is positive semi-definite; in this case, it

is necessary to examine higher order derivatives to determine whether x_* is a local minimizer. If $g(x_*) = 0$ and $H(x_*)$ is negative definite, then the above argument shows that x_* is a local maximizer of $f(x)$. If $g(x_*) = 0$ and $H(x_*)$ has both positive and negative eigenvalues, then x_* is said to be a *saddle point* of $f(x)$. A saddle point is a local minimizer of some cross-section of $f(x)$ and a local maximizer of some other cross-section.

1.4 Rates of Convergence

Most algorithms for nonlinear optimization problems in continuous variables are iterative. They generate iterates $x_k \in \mathbb{R}^n$, $k = 0,1,2,\cdots$ which are meant to converge to a solution x_*. In this case, the rate of convergence is of interest. This subsection reviews the rates of convergence that are important in continuous optimization.

Definition 1.2 Let $x_k \in \mathbb{R}^n$, $k=0,1,\cdots$. Then the sequence $\{x_k\}$ is said to converge to a point $x_* \in \mathbb{R}^n$ if for every i, the i^{th} component $(x_k - x_*)_i$ satisfies

$$\lim_{k \to \infty} (x_k - x_*)_i = 0 . \tag{1.17}$$

If, for some vector norm $\| \cdot \|$, there exist $K \geq 0$ and $\alpha \in [0,1)$ such that for all $k \geq K$,

$$\| x_{k+1} - x_* \| \leq \alpha \| x_k - x_* \| , \tag{1.18}$$

then $\{x_k\}$ is said to converge *q-linearly* to x_* in the norm $\| \cdot \|$. If for some sequence of scalars $\{\alpha_k\}$ that converge to 0

$$\| x_{k+1} - x_* \| \leq \alpha_k \| x_k - x_* \| , \tag{1.19}$$

then $\{x_k\}$ is said to converge *q-superlinearly* to x_*. If $\{x_k\}$ converges to x_* and there exist $K \geq 0$ and $\alpha \geq 0$ such that for all $k \geq K$,

$$\| x_{k+1} - x_* \| \leq \alpha \| x_k - x_* \|^2 \tag{1.20}$$

$\{x_k\}$ is said to converge *q-quadratically* to x_*.

Note that a sequence may be linearly convergent in one norm but not another, but that superlinear and quadratic convergence are independent of the choice of norm on $\mathbb{R}^n$.

Most methods for continuously differentiable optimization problems are *locally q-superlinearly* or *locally q-quadratically convergent*, meaning that they converge to the solution x_* with a superlinear or quadratic rate if they are started sufficiently close to x_*. In practice, local quadratic convergence is quite fast as it implies that the number of significant digits in x_k as an approximation to x_* roughly doubles at each iteration once x_k is near x_*. Locally superlinearly convergent methods that occur in optimization also are often quickly convergent in practice. Linear convergence, however, can be quite slow, especially if the constant α depends upon the problem as it usually does. Thus linearly convergent methods are avoided wherever possible in optimization unless it is known that α is acceptably small. It is easy to define rates of convergence higher than quadratic, e.g. cubic, but they play virtually no role in practical algorithms for multi-variable optimization problems.

The prefix q preceding the words linear, superlinear, or quadratic stands for "quotient" rates of convergence. This notation, commonly used in the optimization literature, is used to contrast with r ("root") rates of convergence, which are a weaker form of convergence. A sequence is *r-linearly* convergent, for example, if the errors $\|x_k - x_*\|$ are bounded by a sequence of scalars $\{r_k\}$ which converge q-linearly to 0. The ellipsoidal algorithm for linear programming (Khachiyan [1979]) is a perfect example of an r-linear method since the radii of the ellipses are a sequence of error bounds that converge q-linearly to zero. Similar definitions apply to other r-rates of convergence; for further detail see Ortega and Rheinboldt [1970]. Thoughout this book, if no prefix precedes the words linear, superlinear, or quadratic, q-order convergence is assumed.

2. Newton's Method

Among nonlinear optimization researchers, there is a strong suspicion that if any iterative method for any problem in any field is exceptionally effective, then it is Newton's method in some appropriate context. This is not to say that Newton's method is always practical.

In this section we derive the Newton iteration for nonlinear equations through the affine or linear Taylor series model (1.3) in subsection 2.1, and the Newton iteration for unconstrained optimization through the corresponding quadratic model (1.7) in subsection 2.2. We give a rigorous, but intuitive, analysis of the local convergence of the corresponding iterative method, called the Newton or Newton-Raphson method, and set the stage for a discussion of some clever and effective modifications. The first of these modifications is controlled inaccuracies in the solutions of the model problems. The resulting inexact Newton method is the topic of subsection 2.3, and it can greatly reduce the expense of computing Newton steps far from the solution. This is especially important for large problems, since then the Newton step itself is likely to be computed by an iterative method. Some additional modifications are discussed Section 3.

2.1. Newton's method for nonlinear equations

The underlying principle in most of continuous optimization is to build, at each iteration, a local model of the problem which is valid near the current solution estimate. The next, improved, solution estimate is gotten at least in part from solving this local model problem.

For nonlinear equations, the local model is the Taylor series model (1.3). The basic Newton iteration can be written as the root of this model,

$$x_+ = x_c - J(x_c)^{-1} F(x_c) .$$

But in keeping with the local model point of view, we prefer to think of it as:

$$\begin{aligned} \text{Solve} \quad & J(x_c)s_c^N = -F(x_c) \\ \text{and set} \quad & x_+ = x_c + s_c^N , \end{aligned} \tag{2.1.1}$$

where s_c^N denotes the current Newton step, because this results directly from

$$0 = F(x_c) + J(x_c)(x - x_c)\,. \tag{2.1.2}$$

Notice that we could also have solved directly for x_+ from (2.1.2), giving

$$J(x_c)x_+ = -F(x_c) + J(x_c)x_c\,. \tag{2.1.3}$$

In exact arithmetic, these formulations are equivalent as long as the columns of $J(x_c)$ span $\mathbb{R}^{n\times n}$. In computer arithmetic, (2.1.1) generally is preferable. The reason is that in practice, we expect the magnitude of the error in solving a linear system $Az = b$ by a matrix factorization and backsolve technique to be estimated by

$$\frac{\|\hat{z} - z\|}{\|z\|} \approx \mu \|A\| \cdot \|A^{-1}\| \equiv \mu\kappa(A)\,,$$

where $\hat{z}$ is the computed solution, the components of A and b are machine numbers, and μ is the quantity known as 'machine epsilon' or 'unit rounding error', i.e., the smallest positive number such that the computed sum of $1+\mu$ is different from 1. Since we generally expect s_c^N to be smaller than x_+, we expect $\|\hat{s_c}^N - s_c^N\|$ in (2.1.1) to be smaller than $\|\hat{x}_+ - x_+\|$ in (2.1.3). Of course, we commit an error of magnitude $\mu\|x_c\|$ when we add $\hat{s_c}^N$ to x_c to get $\hat{x_c}$ in (2.1.1), but this does not involve $\kappa(J(x_c))$. Thus, we will think always of (2.1.1) as Newton's method with the linear system solved using an appropriate matrix factorization. This discussion is relevant for all the methods based on Taylor series type models. For a discussion of the computer solution of linear equations, see Stewart [1973] or Golub and Van Loan [1983].

The main theorem of this section is given below. We use the notation $N(x, \delta)$ to denote the open neighborhood about x of radius δ.

Theorem 2.1.1. Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be continuously differentiable in an open convex set $D \subset \mathbb{R}^n$. If there exists $x_* \in D$ and $r, \beta, > 0$ such that $J(x) \in Lip_\gamma(N(x_*, r))$, $F(x_*) = 0$, and $\|J(x_*)^{-1}\| \le \beta$, then there exists $\varepsilon > 0$ such that for every $x_0 \in N(x_*, \varepsilon)$ the sequence $\{x_k\}$ generated by

$$x_{k+1} = x_k - J(x_k)^{-1}F(x_k),\ k = 0, 1, \cdots$$

is well defined, converges to x_*, and satisfies

$$\|x_{k+1} - x_*\| \le \beta\gamma\|x_k - x_*\|^2\,.$$

Proof: Since Lipschitz continuity implies continuity, and since the determinant is a continuous function of the entries of a matrix, it is easy to see that we can assume without loss of generality that $J(x)$ is invertible

for

$x \in N(x_*, r)$ and that $\|J(x)^{-1}\| \leq 2\beta$.

Thus, if $x \in N(x_*, \varepsilon)$ for $\varepsilon \leq \min\{r, (2\beta\gamma)^{-1}\}$, x_{k+1} exists and

$$\begin{aligned} x_{k+1} - x_* &= x_k - J(x_k)^{-1}F(x_k) - x_* + J(x_k)^{-1}F(x_*) \\ &= J(x_k)^{-1}\left[F(x_*) - F(x_k) - J(x_k)(x_* - x_k)\right]. \end{aligned}$$

Using (1.6), this gives

$$\begin{aligned} \|x_{k+1} - x_*\| &\leq \|J(x_k)^{-1}\| \cdot \|F(x_*) - J(x_k)(x_* - x_k)\| \\ &\leq 2\beta \frac{\gamma}{2} \|x_k - x_*\|^2 \leq \beta\gamma \|x_k - x_*\|^2 \\ &\leq \beta\gamma\varepsilon \|x_k - x_*\| \leq \frac{1}{2}\|x_k - x_*\|, \end{aligned}$$

which establishes both convergence and the quadratic rate of convergence, and completes the proof.

Notice that we could apply Newton's method to $G(x) = J(x_*)^{-1}F(x)$, for which $G(x_*) = 0$, $J_G(x_*) \equiv G'(x_*) = I$, and $J_G(x) \in Lip_{\beta\gamma}N(x_*, r)$, and the identical sequence of iterates and convergence analysis would result. This emphasizes one of the major advantages of Newton's method; it is invariant under any affine scaling of the independent variable. Furthermore, note that the Lipschitz constant of $J_G(x)$, $\beta\gamma$, is a relative measure of the nonlinearity of F as given by the change in J scaled by $J(x_*)^{-1}$. This is satisfying since we would like to believe that F is no more or less nonlinear if we change our basis.

In section 3.3, we give an example of the application of Newton's method to a specific problem. It is compared there to Broyden's method which is superlinearly rather than quadraticly convergent, but which does not share two major disadvantages of Newton's method. Broyden's method does not require the expensive and error-prone computation of $J(x)$ or the finite difference approximation discussed in Section 3.2, and it reduces the be successful to solve the linear system (2.1.1) for the Newton step from $0(n^3)$ to $0(n^2)$ operations. Broyden's method shares the third major disadvantage of Newton's method in that both require a good initial guess x_0 to guarantee success. Because of this, they are referred to as *local methods*, and must be augmented by the techniques discussed in Section 4 to be successful from poor starting points.

2.2. Newton's Method for Unconstrained Optimization

In this section, we look at Newton's method for the unconstrained minimization problem (1.1). It can be derived by applying the basic iteration (2.1.1) to the first order necessary condition $g(x)=0$ for unconstrained optimization given by Theorem 1.1. This gives the basic Newton iteration

$$\begin{aligned} H(x_c)s_c^N &= -g(x_c) \\ x_+ &= x_c + s_c^N . \end{aligned} \tag{2.2.1}$$

The Newton iteration (2.2.1) is equivalent to solving for a zero of the gradient of the Taylor series quadratic model $m_c^N(x_c+s)$ given by (1.7). If $H(x_c)$ is positive definite, then x_+ is a local minimizer of m_c^N by Theorem 1.2. In fact, it is the global minimizer of the model m_c^N since x_* is the only zero of $\nabla m_c^N(x)$. This leads to a very satisfying interpretation of Newton's method for (1.2) when H_c is positive definite. We construct a uniformly convex quadratic model m_c^N of f by matching function, gradient, and curvature to f at x_c, and then step to the global minimum of the model. Furthermore, since $H(x_c)$ is positive definite, then the Cholesky factorization can be used to solve for s_c^N in (2.2.1). We will see another advantage in Section 4.1. If $H(x_c)$ is not positive definite, however, then we have no reason to believe that (2.1.1) may not lead towards a saddle point or even a maximizer.

Newton's method has all the same disadvantages for unconstrained minimization that it had for nonlinear equations. That is, it is not necessarily globally convergent, and it requires $0(n^3)$ operations per iteration. In addition, we need $H(x_c)$ to be positive definite rather than just nonsingular. Furthermore, both first and second partial derivatives are required. We will address these issues in Sections 3 and 4.

We finish the subsection by stating a convergence theorem for (2.2.1) that follows directly from Theorems 1.2 and 2.1.1.

Theorem 2.2.1. Let $f:\mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable in an open convex set $D \subset \mathbb{R}^n$. Assume there exists $x_* \in D$ and $r, \beta > 0$ such that $g(x_*)=0$, $H(x_*)$ is positive definite, $\|H(x_*)^{-1}\| \le \beta$, and $H(x) \in Lip_\gamma N(x_*, r)$. Then, there exists $\varepsilon > 0$ such that for every $x_0 \in N(x_*, \varepsilon)$, the sequence $\{x_k\}$ generated by

$$x_{k+1} = x_k - H(x_k)^{-1} g(x_k)$$

is well defined, converges to the local minimizer x_* of f, and satisfies

$$\|x_{k+1}-x_*\| \le \beta\gamma\|x_k-x_*\|^2 .$$

2.3. The Inexact Newton Method

This section deals with a topic of importance in the use of Newton's method, particularly for large dimensional problems. In such cases, iterative methods like the Gauss-Seidel or conjugate direction methods are often the only practical way to solve the linear system (2.1.1) or (2.2.1) for the Newton step. The natural question is "how does the inaccuracy in the solution of (2.1.1) for the Newton step affect the rate of convergence in the exact Newton method."

An analysis by Dembo, Eisenstat and Steihaug [1982] leads to the following result.

Theorem 2.3.1. Let the hypotheses of Theorem 2.1 hold, and let $\{\eta_k\}$ be a sequence of real numbers satisfying $0 \le \eta_k \le \eta < 1$. Then for some $\varepsilon > 0$, and any $x_0 \in N(x_*, \varepsilon)$, the inexact Newton method corresponding to any $\{x_k\}$ defined by

$$\begin{aligned} &F'(x_k)s_k = -F(x_k)+r_k, \text{ where } \frac{\|r_k\|}{\|F(x_k)\|} \le \eta_k \\ &x_{k+1} = x_k + s_k \end{aligned} \tag{2.3.1}$$

is well defined and converges at least q-linearly to x_*, with

$$\|x_{k+1}-x_*\| \le \eta\|x_k-x_*\| .$$

If $\{\eta_k\}$ converges to 0, then $\{x_k\}$ converges superlinearly to x_*. If $\{\eta_k\}$ is $O(\{\|F(x_k)\|^p\}$ for $0 \le p \le 1$, then $\{x_k\}$ converges to x_* with q-order $1+p$.

For example, this result guarantees quadratic convergence if at each iteration the approximate Newton step s_k satisfies

$$\frac{\|F(x_k)-F'(x_k)s_k\|}{\|F(x_k)\|} < \min\{1, \|F(x_k)\|\} . \tag{2.3.2}$$

The quantity on the left of (2.3.2), sometimes called the relative residual, is usually monitored in practice when solving a linear system by iteration. Thus, the analysis gives a convenient stopping rule for the linear, or inner, iteration for solving the linear problem (2.1.1) which is embedded in the outer iteration for solving the nonlinear problem (1.2). All these results also apply to the unconstrained optimization problem.

Notice that Theorem 2.3.1 is for an iteration (2.3.1) in which every quantity can be computed if F' and F can be. In the next section, we will consider the effects of using approximations to the Taylor series model. There are strong connections between the analysis of those approaches and the inexact Newton method. In particular, for unconstrained optimization, (2.3.1) is equivalent to a statement about the inaccuracy in an approximation to the gradient $\nabla f(x_k)$.

3. Derivative Approximations

In section 2, we developed the local theory for Newton's method based on the appropriate Taylor series model. In practice, users prefer routines that approximate the required derivatives either by finite differences or by some of the clever and effective multi-dimensional secant methods. In this section, we will consider several instances of these approximations. They all involve using local models of the form

$$M_c(x_c + d) = F(x_c) + B_c d \tag{3.1}$$

to solve (1.2) or of the form

$$m_c(x_c + d) = f(x_c) + g(x_c)^T d + \frac{1}{2} d^T B_c d \tag{3.2}$$

to solve (1.1). We call these quasi-Newton models.

In Section 3.1, we will see the surprisingly simple local convergence analysis for any quasi-Newton iterative method

$$\begin{aligned} &B_c s^{q-N} = -F(x_c) \quad \text{or} \quad B_c s^{q-N} = -g(x_c) \\ &x_+ = x_c + s^{q-N} \end{aligned} \tag{3.3}$$

based on these local models. Section 3.2 will present the finite-difference Newton method.

Sections 3.3 and 3.4 will be devoted to some popular multi-dimensional secant methods for choosing B_c. We will concentrate on Broyden's method for nonlinear equations and the BFGS method for unconstrained optimization. Finally, in Section 3.5, we will consider the incorporation of sparsity into the approaches for approximating derivatives, including some techniques to save significantly on the cost of obtaining B_c by finite differences when $J(x_c)$ has a known sparsity structure.

Since we will deal with sparsity later in this section, it seems appropriate to introduce this important practical property here. In general, we say that a matrix is sparse if few of its entries, say less than 15%, are nonzero. For nonlinear systems of equations, $J(x)$ is sparse if most of the component equations F_i involve very few of the unknowns x_j. For unconstrained minimization, $H(x)$ is sparse if most variables do not interact in the sense that most of the cross partial derivatives are zero.

In practice, the sparsity of $J(x)$ or $H(x)$ is independent of the value of x, and this allows for some important savings at every iteration in solving for the Newton step in (2.1.1). Naturally, we wish to incor-

porate such a useful property into B_c in order to have the same savings in (3.3). Furthermore, it seems intuitive that we should be able to more efficiently approximate $J(x)$ or $H(x)$ if we know that most of its entries are zero.

To some extent, this is true. However, sparsity is entirely a property of the basis used to represent x, and so we may make our approximation methods more dependent than before on the specific representation of the problem. This can lead to practical difficulties of the sort that are usually called 'bad scaling'.

3.1. General convergence theory

In this section, we will consider general local models of the forms (3.1) and (3.2), and we will compare them to the appropriate Taylor series or Newton model to obtain one-step local convergence estimates for the associated iterative methods (3.3).

Lemma 3.1.1. Let F satisfy the hypotheses of Theorem 2.1.1 and let $M_c(x_c+d)$ be given by (3.1). Then for any d,

$$\|F(x_c+d)-M_c(x_c+d)\| \le \frac{\gamma}{2}\|d\|^2+\|[B_c-F'(x_c)]d\| . \tag{3.1.1}$$

In addition, if B_c^{-1} exists and $e_c=x_*-x_c$, $e_+=x_*-x_+$ for x_+ defined by (3.3), then

$$\begin{aligned}\|e_+\| &\le \|B_c^{-1}\|\left[\frac{\gamma}{2}\|e_c\|+\frac{\|[B_c-J(x_c)]e_c\|}{\|e_c\|}\right]\|e_c\| \\ &\le \|B_c^{-1}\|\left[\frac{\gamma}{2}\|e_c\|+\|B_c-J(x_c)\|\right]\|e_c\| .\end{aligned} \tag{3.1.2}$$

Proof: First, we add and subtract the Newton model (1.3) to obtain

$$\begin{aligned}\|F(x_c+d)-M_c(x_c+d)\| &= \|F(x_c+d)\pm M_c^N(x_c+d)-F(x_c)-B_c d\| \\ &\le \|F(x_c+d)-M_c^N(x_c+d)\|+\|F(x_c)+J(x_c)d-M_c(x_c+d)\| .\end{aligned}$$

This says that the error in the model (3.1) is bounded by the sum of the error in the Newton model and the difference between the Newton and quasi-Newton models. Now we apply (1.6) to the first term and simplify the second term to obtain (3.1.1).

To obtain (3.1.2), we use $F(x_*)=0$ and (3.3) to write

$$x_+ - x_* = x_c - x_* - B_c^{-1}[F(x_c) - F(x_*)]$$
$$= B_c^{-1}[F(x_c + e_c) - (F(x_c) + B_c e_c)],$$

so

$$\|e_+\| \leq \|B_c^{-1}\| \cdot \|F(x_c + e_c) - M_c(x_c + e_c)\|$$

and (3.1.2) follows from (3.1.1).

The convergence theorems of Section 2 for Newton's method follow very simply from (3.1.2) with $B_c = J(x_c)$. Indeed, we have reduced the analysis of any quasi-Newton method (3.3) to an analysis of the corresponding Jacobian or Hessian approximation rule. There are three especially important properties to consider.

First, sometimes the error in B_c as an approximation to $J(x_c)$ is controlled by a parameter θ in the approximation rule; we will see that the finite difference Newton method fits this mold. In such rules, it is sometimes the case that if the sequence of iterates $\{x_k\}$ is well defined and converges to x_*, then $\{B_k\}$ converges to $J(x_*)$. Such rules are said to provide *consistent* Jacobian or Hessian approximations. (See Ortega and Rheinboldt [1970].) The reader will immediately see from (3.1.2) that consistent methods are at least q-superlinear. Of course, Newton's method is consistent and q-quadratic, if $J(x)$ is Lipschitz continuous.

Second, Dennis and More [1974] prove that a quasi-Newton method that is convergent with sufficient speed that $\sum_{k=1}^{\infty} \|e_k\| < \infty$ (r-linear is sufficient), will be q-superlinear if and only if the approximations are *directionally consistent* in the sense that

$$\lim_{k \to \infty} [B_k - J(x_k)] \frac{e_k}{\|e_k\|} = 0. \tag{3.1.5}$$

The sufficiency of (3.1.5) for superlinearity is obvious from (3.1.2). The proof of necessity is a bit harder.

These two consistency properties are useful especially for analyzing rates of convergence after convergence of $\{x_k\}$ has been established. It is clear that convergence can be established with much less. In particular, we see from (3.1.2) that if we could ensure that for every k such that $x_0, \ldots, x_k$ is defined, that B_k would be defined and

$$\|B_k^{-1}\| \leq \bar{\beta}, \quad \|B_k - J(x_k)\| \leq \bar{\delta}, \text{ and } \bar{\beta}\bar{\delta} < 1, \tag{3.1.6}$$

then we could get convergence by starting so close that

$$\alpha \equiv \bar{\beta}\,[\gamma\,\|e_0\|/2+\bar{\delta}] < 1\,.$$

This would ensure that $\|e_1\| \le \alpha\|e_0\| < \|e_0\|$ and local q-linear convergence would follow by induction.

A useful property of the Jacobian or Hessian approximation rule in obtaining (3.1.6) is called *bounded deterioration*. There are various useful forms, but the one needed to analyze the Broyden, PSB, Schubert, and DFP secant methods is from Dennis and Walker [1981], pg. 981 where the reader can find a proof of the next two theorems.

Theorem 3.1.2. Let F satisfy the hypothesis of Theorem 2.1.1, and let $B_* \in \mathbb{R}^{n\times n}$ have the property that B_*^{-1} exists and for some operator norm

$$|I - B_*^{-1}F'(x_*)| \le r_* < 1\,.$$

Let $U: \mathbb{R}^n \times \mathbb{R}^{n\times n} \to 2^{\mathbb{R}^{n\times n}}$ be defined in a neighborhood $N = N_1 \times N_2$ of (x_*, B_*) where $N_1 \subset \Omega$ and N_2 contains only nonsingular matrices. Assume that there are nonnegative constants α_1 and α_2 such that for each $(x, B) \in N$, and for $x_+ = x - B^{-1}F(x)$, every $B_+ \in U(x, B)$ satisfies

$$\|B_+ - B_*\| \le [1+\alpha_1\sigma(x,x_+)^p]\cdot\|B - B_*\| + \alpha_2\sigma(x,x_+)^p$$

for $\sigma(x,x_+) = \max\{|x-x_*|, |x_+-x_*|\}$.

Under these hypotheses, for any $r \in (r_*, 1)$, there exist constants ε_r, δ_r such that if $|x_0 - x_*| < \varepsilon_r$ and $|B_0 - B_*| < \delta_r$, then any iteration sequence $\{x_k\}$ defined by

$$x_{k+1} = x_k - B_k^{-1}F(x_k)\,, \quad B_{k+1} \in U(x_k\, B_k)\,,$$

$k = 0, 1, \ldots$, exists, converges q-linearly to x_* with

$$|x_{k+1} - x_*| \le r\cdot|x_k - x_*|\,,$$

and has the property that $\{|B_k|\}$ and $\{|B_k^{-1}|\}$ are uniformly bounded.

There are some important ways in which bounded deterioration may be weaker than consistency. In particular, the essence of the method might be to make B_k not be too much like $J(x_k)$, but to be more convenient to use in (3.3). For example, in the nonlinear Jacobi iteration, B_k is taken to be the diagonal of $J(x_k)$ so that s_k^{-N} defined by (3.3) only costs n divisions. Of course, the Jacobi iteration will not converge

unless the partial derivatives on the diagonal dominate the rest of $J(x_k)$. A standard sufficient condition for this which implies (3.1.6) in the l_∞ norm is called strict diagonal dominance. See Ortega-Rheinboldt [1970]. This brings up another point; the key to a convergence analysis for a specific method is often in choosing the proper norm.

Finally, some methods, like the BFGS secant method, are more readily analyzed by thinking of them as directly generating approximations to the inverse. The following theorem from Dennis and Walker [1981], pg. 982 gives an inverse form of the bounded deterioration principle.

Theorem 3.1.3. Let F satisfy the of Theorem 2.1.1, hypothesis and let K_* be an invertible matrix with $|I - K_* F'(x_*)| \le r_* < 1$.

Let $U: \mathbb{R}^n \times \mathbb{R}^{n\times n} \to 2^{\mathbb{R}^{n\times n}}$ be defined in a neighborhood $N = N_1 \times N_2$ of (x_*, K_*), where $N_1 \subset \Omega$. Assume that there are nonnegative constants α_1, α_2 such that for each (x, K) in N, and for $x_+ = x - KF(x)$, the function U satisfies

$$||K_+ - K_*|| \le [1 + \alpha_1 \sigma(x, x_+)^p]\,||K - K_*|| + \alpha_2 \sigma(x, x_+)^p$$

for each $K_+ \in U(x, K)$. Then for each $r \in (r_*, 1)$ there exist positive constants ε_r, δ_r such that for $|x_0 - x_*| < \varepsilon_r$ and $|K_0 - K_*| < \delta_r$, any sequence $\{x_k\}$ defined by

$$x_{k+1} = x_k - K_k F(x_k), \quad K_{k+1} \in U(x_k, K_k)$$

$k = 0, 1, \ldots$, exists, converges q-linearly to x_* with $|x_{k+1} - x_*| \le r\,|x_k - x_*|$, and has the property that $\{|K_k|\}$ and $\{|K_k^{-1}|\}$ are uniformly bounded.

3.2. Finite-Difference Derivatives

In Section 2 we saw that the local models furnished by the appropriate partial Taylor series are very useful in solving continuous optimization problems. Sometimes, the appropriate partial derivatives are not available analytically, and other times the user is not willing to be bothered providing them. Thus, most good optimization packages provide routines for estimating partial derivatives by finite differences. These routines may (and should) even be used to check for errors in any derivatives the user does provide.

In this section, we will discuss briefly the surprisingly rich topic of how to compute finite-difference approximations to first and second partial derivatives. We will discuss aspects of mathematical accuracy, numerical roundoff, convergence, and efficiency. We begin with accuracy.

We know from elementary calculus that

$$\lim_{h\to 0}\frac{F_i(x+he_j)-F_i(x)}{h}=\frac{\partial F_i(x)}{\partial x_j}$$

where e_j is the j^{th} column of the $n\times n$ identity matrix. This is called the *forward difference* approximation and it suggests approximating the j^{th} column of $J(x_c)$ by

$$\Delta_j F(x_c,h)=\frac{1}{h_j}[F(x_c+h_je_j)-F(x_c)] \tag{3.2.1}$$

for an appropriately chosen vector h of finite-difference steps. We will use the notation $\Delta F(x,h)$ for the Jacobian approximation whose j^{th} column is given by (3.2.1).

Lemma 3.2.1. Let $\|\cdot\|$ be a norm for which $\|e_j\|=1$ and let $F\in Lip_\gamma(D)$ with $x_c, x_c+h_je_j$, $j=1,...,n$ all contained in D. Then

$$\|\Delta_j F(x_c,h)-J(x_c)e_j\|\le\frac{\gamma}{2}|h_j|. \tag{3.2.2}$$

Furthermore, in the l_1 operator norm,

$$\|A\|_1\equiv\max_{1\le j\le n}\sum_{i=1}^{n}|a_{ij}|,$$

it follows that

$$\|\Delta F(x_c,h)-J(x_c)\|_1\le\frac{\gamma}{2}\|h\|_\infty \tag{3.2.3}$$

where $\|h\|_\infty\equiv\max_{1\le j\le n}|h_j|$ is the l_∞ vector norm.

Proof: The proof follows easily from the Taylor series remainder (1.6), since

$$\begin{aligned}\|\Delta_j F(x_c,h)-J(x_c)e_j\| &= |h_j|^{-1}\cdot\|F(x_c+h_je_j)-F(x_c)-J(x_c)h_je_j\|\\ &= |h_j|^{-1}\cdot\|F(x_c+h_je_j)-m_c^N(x_c+h_je_j)\|\\ &\le |h_j|^{-1}\cdot\frac{\gamma}{2}\|h_je_j\|^2=\frac{\gamma}{2}|h_j|.\end{aligned}$$

We get (3.2.3) directly from (3.2.2) since $\|e_j\|_1=1$ and so

$$\| \Delta F(x_c, h) - J(x_c) \|_1 \equiv \max_{1 \le j \le n} \| \Delta_j F(x_c, h) - J(x_c) e_j \|_1$$
$$\le \max_{1 \le j \le n} \frac{\gamma}{2} |h_j| \equiv \frac{\gamma}{2} \|h\|_\infty .$$

Equation (3.2.1) also is used to approximate the gradient by the forward difference approximation

$$\delta_i f(x_c, h) = \frac{f(x_c + h_i e_i) - f(x_c)}{h_i}, \; i = 1, \ldots, n . \tag{3.2.4}$$

Although this forward difference approximation is generally accurate enough, sometimes *central differences* are useful because of roundoff considerations we will discuss later. Since central differences are most often used for gradient approximations, we define them in that context.

$$\delta_i f(x_c, h) = \frac{f(x_c + h_i e_i) - f(x_c - h_i e_i)}{2h_i} \tag{3.2.5}$$

and

$$\delta f(x_c, h) = (\delta_1 f(x_c, h), \ldots, \delta_n f(x_c, h))^T .$$

Lemma 3.2.2. Let $\|\cdot\|$ be a norm such that $\|e_j\| = 1$, and let $H \in Lip_\gamma(D)$, with $x_c, x_c + h_i e_i$, $i = 1, \ldots, n$ all be contained in D. Then $\delta_i f(x_c, h)$ given by (3.2.5) obeys

$$|\delta_i f(x_c, h) - \frac{\partial f}{\partial x_i}(x_c)| \le \tag{3.2.6}$$

and

$$\|\delta f(x_c, h) - g(x_c)\|_\infty \le \frac{\gamma}{6} \|h\|_\infty^2 .$$

Proof: Note that from (1.7), (1.8) we get

$$[f(x_c + h_i e_i) - m_c^N(x_c + h_i e_i)] - [f(x_c - h_i e_i) - m_c^N(x_c - h_i e_i)]$$
$$= f(x_c + h_i e_i) - f(x_c - h_i e_i) - 2h_i \frac{\partial f}{\partial x_i}(x_c) .$$

Thus, from (1.10) and the triangle inequality,

$$|f(x_c + h_i e_i) - f(x_c - h_i e_i) - 2h_i \frac{\partial f}{\partial x_i}(x_c)|3 = \frac{\gamma}{3} |h_i|^3$$

from which (3.2.6) follows directly.

Note that the central difference gradient is more accurate than the forward difference gradient, but that it requires $2n$ rather than n evaluations of $f(x)$ if we assume that $f(x_c)$ is already available.

If the gradient is obtained analytically, but the Hessian is to be approximated by finite differences, then it can be approximated by applying (3.2.1) to $g(x)$, obtaining the approximation $\Delta g(x_c, h)$. However the matrix $\Delta g(x_c, h)$ would not be symmetric although $H(x_c)$ would be. In this case, a sensible strategy would be to use $B_c = \frac{1}{2}[\Delta g(x_c) + \Delta g(x_c)^T]$ as the Hessian approximation. Some theoretical justification for this comes from noting that B_c is the Frobenius norm projection of $\Delta g(x_c)$ into the subspace of all symmetric matrices. Thus, from the Pythagorian Theorem,

$$\| H(x_c) - B_c \|_F \leq \| H_c - \Delta g(x_c, h) \|_F$$

where

$$\| A \|_F \equiv (\sum_{i,j} | a_{ij} |^2)^{\frac{1}{2}} \tag{3.2.7}$$

is the Frobenius norm. This norm will be useful later when we again want an inner product structure on the vector space of real matrices.

It is also possible to obtain an approximate Hessian using $(n^2+3n)/2$ evaluation of $f(x)$. By expanding the Taylor series through third order terms, a stronger version of (3.2.9-10) can be proven in which the $|h_i|^2/|h_j|$ and $|h_j|^2/|h_i|$ terms are removed, and the constant is different.

Lemma 3.2.3. Let $\|\cdot\|$ be a norm such that $\|e_j\| = 1$ and let $H \in Lip_\gamma(D)$ with x_c, $x_c + h_i e_i$, $x_c + h_j e_j$, and $x_c + h_i e_i + h_j e_j$, $i, j = 1, \cdots n$ all contained in D.

$$\text{Let } [H_c]_{ij} \equiv \frac{f(x_c + h_i e_i + h_j e_j) - f(x_c + h_i e_i) - f(x_c + h_j e_j) + f(x_c)}{h_i h_j} . \tag{3.2.8}$$

Then,

$$| [H_c]_{ij} - [H(x_c)]_{ij} | \leq \frac{\gamma}{6} \left[2 \frac{|h_i|^2}{|h_j|} + 3|h_i| + 3|h_j| + 2 \frac{|h_j|^2}{|h_i|} \right] . \tag{3.2.9}$$

If the l_1, l_∞, or Frobenius norm is used, then

$$\| H_c - H(x_c) \| \leq \frac{n\gamma}{6} \max_{i,j} \left[2 \frac{|h_i|^2}{|h_j|} + 3|h_i| + 3|h_j| + 2 \frac{|h_j|^2}{|h_i|} \right] . \tag{3.2.10}$$

Proof: The proof is very much like the previous proof. Let $s_i = h_i e_i, s_j = h_j e_j$ and $s_{ij} = s_i + s_j$, then

$$\begin{aligned} &[f(x_c + s_{ij}) - m_c^N(x_c + s_{ij})] - [f(x_c + s_i) - m_c^N(x_c + s_i)] - [f(x_c + s_j) - m_c^N(x_c + s_j)] \\ &= f(x_c + s_{ij}) - f(x_c + s_i) - f(x_c + s_j) + f(x_c) - h_i h_j [H(x_c)]_{ij} . \end{aligned}$$

From the triangle inequality and (1.10), we get

$$\begin{aligned} |h_i h_j [H_c] - h_i h_j [H(x_c)]_{ij}| &\le \frac{\gamma}{6} [||s_{ij}||^3 + ||s_i||^3 + ||s_j||^3] \\ &\le \frac{\gamma}{6} [(||s_i|| + ||s_j||)^3 + ||s_i||^3 + ||s_j||^3] \\ &\le \frac{\gamma}{6} [(|h_i| + |h_j|)^3 + |h_i|^3 + |h_j|^3] \end{aligned}$$

from which (3.2.9) and (3.2.10) follow.

Now we have seen some useful rules for approximating derivatives by differences of function values, and we have analyzed the accuracy of these approximations. So far, it seems as though the obvious thing to do is to choose the vector h to be very small in order to make the approximate derivatives more accurate. The difficulty is that these approximation rules are certain to lose more accuracy due to finite-precision cancellation errors as h becomes smaller.

In section 2.1, we introduced *machine epsilon* μ as the smallest positive quantity for which the floating-point sum $1+\mu$ would be different from the floating point representation of 1. Thus, even if we ignore the fact that all functions are to be evaluated in finite-precision, we see that $|h_j|$ must be large enough that $x_c + h_j e_j$ is not the same as x_c in finite precision arithmetic, or else the numerator in the forward difference would be zero. This means that $|h_j| \ge \mu |[x_c]_j|$.

If we remember that function values are computed in finite precision with t digits then we can believe that if we obtain a value of say $F_i(x_c) = (.d_1 ..., d_t)10^e$, it is reasonable that d_t or d_{t-1} are much less likely to be correct than d_1 or d_2. This can lead to real problems when coupled with the fact that $F_i(x_c + h_j e_j) = (.d_1' \cdots d_t') \cdot 10^{e'}$ will probably have $e' = e$, and $d_k' = d_k$ for $k = 1, 2, ..., \bar{t} < t$, with $\bar{t}$ closer to t the smaller $|h_j|$ is. In other words, the smaller we take $|h_j|$, the more of the most accurate leading digits of $F(x)$ are canceled out by the subtraction $F_i(x_c + h_j e_j) - F_i(x_c)$ in the numerator of the forward-difference formula (3.2.1). This means that the difference will have at most $(t - \bar{t})$ meaningful digits, which are computed using the less trustworthy trailing digits of $F(x)$. Thus, $|h_j|$ must at least be large enough so that $F_i(x_c)$ and $F_i(x_c + h_j e_j)$ differ in some trustworthy digits.

The standard rule to use in setting the entire vector h for (3.2.1) or (3.2.4) is

$$h_c = \mu_F^{\frac{1}{2}} x_c \,, \tag{3.2.11}$$

where μ_F is the relative accuracy in the subroutine used to evaluate F or f. In other words, if the routine is accurate to $t_F < t$ digits, then $\mu_F = 10^{-t_F}$ while if it is accurate in all its digits we take $\mu_F = \mu$. If some component $[x]_i = 0$, then we take $h_i = \mu_F^{\frac{1}{2}}$ for lack of a better choice. This rule attempts to balance the mathematical and numerical error in (3.2.1); see e.g. Dennis and Schnabel [1983].

The reader will see that all these numerical difficulties clearly are compounded in the Hessian approximation (3.2.8), which calls for a larger relative magnitude of h since the inaccuracies in the numerator are divided by h^2. We also see that the bound in (3.2.9) points toward choosing all the components of h to be about the same magnitude. Thus, we recommend that x_c be scaled as well as possible, and an analysis suggests that

$$h_c = \mu_F^{\frac{1}{3}} x_c \ . \tag{3.2.12}$$

This rule is also suggested for use in (3.2.5).

Next we state a theorem on the theoretical rate of convergence of finite-difference methods. The reader can easily furnish a proof by combining Lemma 3.2.1 with Lemma 3.1.1.

Theorem 3.2.4. Let F and x_* obey the hypotheses of Theorem 2.1.1 in the l_1 norm. There exists $\varepsilon, \eta > 0$ such that if $\{h_k\}$ is a sequence in $\mathbb{R}^n$ with $0 \le \| h_k \|_\infty \le \eta$, and $x_0 \in N(x_*, \varepsilon)$, then the sequence $\{x_k\}$ generated by

$$B_k e_j = \begin{cases} \dfrac{F(x_k + (h_k)_j e_j) - F(x_k)}{(h_k)_j}, & (h_k)_j \neq 0 \\ \dfrac{\partial F}{\partial x_j}(x_k), & (h_k)_j = 0, \end{cases}$$

$$x_{k+1} = x_k - B_k^{-1} F(x_k), \quad k = 0, 1, \cdots$$

is well defined and converges q-linearly to x_*. If $\lim_{k \to} \| h_k \|_\infty = 0$, then the convergence is q-superlinear. If there exists some constant C_1 such that $\| h_k \|_\infty \le C_1 \| x_k - x_* \|_1$, or equivalently, a constant C_2 such that $\| h_k \|_\infty \le C_2 \| F(x_k) \|_1$, then the convergence is q-quadratic.

Even though Theorem 3.2.4 does not consider the finite precision effects discussed above, it reflects the practical experience with finite difference approximations : if the stepsizes are properly selected, then finite difference methods give similar performance to the same methods using analytic derivatives. The main disadvantage of using these approximations is their cost. The forward difference approximations

(3.2.1) to $J(x)$ or (3.2.4) to $g(x)$ require n additional evaluations of $F(x)$ or $f(x)$, respectively, while the central difference approximation (3.2.5) to $g(x)$ requires $2n$ additional evaluations of $f(x)$. The approximations to $H(x)$ require either n additional evaluations of $g(x)$ for (3.2.1) or, for (3.2.8), $(n^2+3n)/2$ evaluations of $f(x)$. These costs can be considerable for problems where function evaluation is expensive. In the remainder of this section we discuss cheaper approximations to $J(x)$ and $H(x)$ that may be used instead.

It turns out that if $g(x)$ is not available analytically, it is almost always approximated by finite differences, because an accurate gradient is crucial to the progress and termination of quasi-Newton methods, and the secant approximations that we discuss next are not accurate enough for this purpose. In Section 5.1 we discuss a class of methods that does not require gradients.

3.3. Broyden's Method

In one dimension, the secant method is an effective local method for solving nonlinear equations. It can be viewed as a forward-difference method in which the step size h_+ used in constructing the new iterate from x_+ is taken to be $x_c - x_+$, so that the local model derivative B_+ is $[F(x_+ + x_c - x_+)) - F(x_+)]/[x_c - x_+]$. Thus, no extra F values are needed to determine B_+ and build the new local model since $F(x_+ + h_+) = F(x_c)$.

From the local model point of view, the secant method follows from assuming that we will determine the approximate derivative B_+ in the new model of $F(x_+ + d)$, $M_+(x_+ + d) = F(x_+) + B_+ d$, by requiring $M_+(x_c)$ to match $F(x_c)$. This means that

$$F(x_c) = M_+(x_+ + (x_c - x_+)) = F(x_+) + B_+(x_c - x_+)$$

is used to determine B_+. This results in the system of linear equations

$$B\, s_c = y_c \tag{3.3.1}$$

where $s_c = x_+ - x_c$ and $y_c = F(x_+) - F(x_c)$.

For $n = 1$, (3.3.1) uniquely determines B_+. For $n > 1$, there is an $n \times (n-1)$ dimensional linear manifold in $\mathbb{R}^{n \times n}$ of solutions B to (3.3.1). The most commonly used secant method for systems of nonlinear equations, Broyden's method, makes a specific selection of B_+ from this manifold which costs only a small

multiple of n^2 to compute and is a rank-one correction to B_c. It also has a very elegant geometric interpretation given by the following lemma from Dennis and More [1977].

Lemma 3.3.1. Let $B_c \in \mathbb{R}^{n\times n}$, $s_c, y_c \in \mathbb{R}^n$, $s \neq 0$. Then Broyden's update

$$B_+ = B_c + \frac{(y_c - B_c s_c)s_c^T}{s_c^T s_c} \tag{3.3.2}$$

is the unique solution to

$$\min \|B - B_c\|_F \quad \text{subject to} \quad Bs_c = y_c .$$

Proof: If $Bs_c = y_c$, then $B_+ - B_c = [B - B_c]\frac{s_c s_c^T}{s_c^T s_c}$ and so

$$\|B_+ - B_c\|_F \leq \|B - B_c\|_F \cdot \|\frac{s_c s_c^T}{s_c^T s_c}\|_2 = \|B - B_c\|_F .$$

Thus, Broyden's method generalizes the 1-dimensional secant method by changing the current derivative approximation as little as possible in the Frobenius norm consistent with satisfying the secant or quasi-Newton equation (3.3.1). For this reason it is called a *least-change secant update.* Of course, this leaves the problem of finding B_0 to start the process. Generally B_0 is obtained by finite differences.

Broyden, Dennis and More [1973] proved that Broyden's method is locally q-superlinearly convergent. The proof is in two parts, and it is typical of all the least-change secant method proofs. First, one establishes local q-linear convergence by proving bounded deterioration and applying Theorem 3.1.2. Then, one gets q-superlinearity by establishing (3.1.5).

Lemma 3.3.2. Let $D \subset \mathbb{R}^n$ be a convex domain containing x_c, x_+, and x_*. Let $J \in Lip_\gamma(D)$, $B_c \in \mathbb{R}^{n\times n}$, and let B_+ be given by Broyden's update (3.3.2). Then,

$$\|B_+ - J(x_*)\|_F \leq \|B_c - J(x_*)\|_F + \gamma\sigma(x_c, x_+) \tag{3.3.3}$$

Proof: Remember that $\sigma(x_c, x_+) \equiv \max\{\|x_c - x_*\|_2, \|x_+ - x_*\|_2\}$.

$$\begin{aligned} B_+ - J(x_*) &= B_c - J(x_*) + \frac{(y_c \pm J(x_*)s_c - B_c s_c)s_c^T}{s_c^T s_c} \\ &= [B_c - J(x_*)]\left[I - \frac{s_c s_c^T}{s_c^T s_c}\right] + \frac{y_c - J(x_*)s_c)s_c^T}{s_c^T s_c} \\ &= [B_c - J(x_*)]\left[I - \frac{s_c s_c^T}{s_c^T s_c}\right] + \int_0^1 [J(x_c + t\,s_c) - J(x_*)]\,dt\,\frac{s_c s_c^T}{s_c^T s_c} \end{aligned}$$

It is then straightforward to obtain

$$\begin{aligned} \|B_+ - J(x_*)\|_F &\le \|B_c - J(x_*)\|_F \cdot \|I - \frac{s_c s_c^T}{s_c^T s_c}\|_2 \\ &\quad + \gamma \int_0^1 \|x_c + t s_c - x_*\|_2 \, dt \cdot \|\frac{s_c s_c^T}{s_c^T s_c}\|_2 \\ &\le \|B_c - J(x_*)\|_F + \gamma \sigma(x_+, x_c), \end{aligned}$$

since $I - \frac{s_c s_c^T}{s_c^T s_c}$ and $\frac{s_c s_c^T}{s_c^T s_c}$ are l_2 projection matrices and so they have unit norm.

This completes the proof, but we can't resist some comments about the geometry of the proof and its relation to the more general derivation of least-change secant methods. Notice that

$$B_+ - J(x_*) = [B_+ - J(x_*)] \cdot \left[I - \frac{s_c s_c^T}{s_c^T s_c} \right] + [B_+ - J(x_*)] \left[\frac{s_c s_c^T}{s_c^T s_c} \right].$$

In $\mathbb{R}^{n \times n}$ with the Frobenius norm inner product, this is just the orthogonal decomposition of $B_+ - J(x_*)$ into its projection into the subspace of matrices that annihilate s_c and the orthogonal complement of the annihilators of s_c. But the essence of Broyden's method is that B_+ has the same projection as B_c into the annihilators of s_c. Thus,

$$B_+ - J(x_*) = [B_c - J(x_*)] \left[I - \frac{s_c s_c^T}{s_c^T s_c} \right] + [B_+ - J(x_*)] \frac{s_c s_c^T}{s_c^T s_c}. \tag{3.3.4}$$

Now the norm of the first term on the right hand side of (3.3.4) will be smaller than $\|B_c - J(x_*)\|$, while the magnitude of the second term is totally dependent on the sagacity of our choice of $y_c = B_+ s_c$ as an approximation to $J(x_*)s_c$.

In general, the idea of a least-change secant method is to adopt an inner product structure on matrix space and a secant condition (3.3.1). In addition, one may require B_c and B_+ to be in some closed linear manifold A of matrices defined by another desirable property like symmetry or sparsity. One then obtains B_+ by orthogonally projecting B_c into the generalized intersection of A with the matrices that satisfy (3.3.1), using the chosen inner product. By generalized intersection, we mean the projection of the set of secant matrices into A.

The next subsection contains another application of this approach. Dennis and Schnabel [1979] derive many more interesting updates based on this approach. Dennis and Walker [1981] give conditions on the secant equations and the inner products used at each iteration so that the resulting quasi-Newton

method is locally q-linearly as fast as the method that uses $B_k \equiv B_*$. Schnabel [1983] considers multiple secant equations. Dennis and Walker [1985] consider the case when an appropriate secant condition is imperfectly known. Grzegorski [1985], Flachs [1986] relax the conditions on A to convexity.

Now we return to the consideration of Broyden's method. In order to complete the proof of superlinear convergence, we need to show that (3.1.5) holds. In fact, it is not hard to show that

$$\lim_{k\to\infty} \frac{\|[B_k - J(x_*)]s_k\|}{\|s_k\|} = 0 \tag{3.3.5}$$

is also equivalent to superlinear convergence under the same hypotheses. Notice that

$$\frac{\|[B_k - J(x_*)]\, s_k\|_2}{\|s_k\|_2} = \left\| \frac{[B_k - J(x_*)]\, s_k s_k^T}{s_k^T s_k} \right\|_F \tag{3.3.6}$$

and so (3.3.5) seems a reasonable condition to try for in light of our previous discussion. For completeness, we state the theorem. For an elementary complete proof, see page 177 of Dennis and Schnabel [1983].

Theorem 3.3.3. Let F satisfy the hypotheses of Theorem 2.1.1. There exist positive constants ε, δ such that if $\|x_0 - x_*\|_2 < \varepsilon$ and $\|B_0 - J(x_0)\| < \delta$, then the sequence $\{x_k\}$ generated by Broyden's method

$$\begin{aligned} x_{k+1} &= x_k - B_k^{-1} F(x_k), \quad k = 0, 1, \cdots \\ B_{k+1} &= B_k + \frac{(y_k - B_k s_k) s_k^T}{s_k^T s_k} \\ y_k &= F(x_{k+1}) - F(x_k), \quad s_k = x_{k+1} - x_k \end{aligned}$$

is well defined and converges q-superlinearly to x_*.

Proof: We will sketch the proof. First, notice that Lemma 3.3.2 shows that the hypotheses of the bounded deterioration result, Theorem 3.1.2, holds with $B_* = J(x_*)$. Thus the method is at least q-linearly convergent. Let $E_k = B_k - J(x_*)$.

In order to show (3.3.5), we need a technical lemma. From the Pythagorian Theorem in $\mathbb{R}^{n\times n}$, we have

$$\begin{aligned} \|E_k \left(I - \frac{s_k s_k^T}{s_k^T s_k} \right)\|_F &= \left(\|E_k\|_F^2 - \left\| \frac{E_k s_k s_k^T}{s_k^T s_k} \right\|_2^2 \right)^{\frac{1}{2}} \\ &\le \|E_k\|_F - \frac{1}{2\|E_k\|_F} \left\| \frac{E_k s_k s_k^T}{s_k^T s_k} \right\|_2^2, \end{aligned} \tag{3.3.7}$$

where the inequality follows from : $\alpha \geq |\beta| \geq 0$ implies $(\alpha^2-\beta^2)^{\frac{1}{2}} \leq \alpha - \beta^2/2\alpha$.

By (3.3.4), (3.3.6), and (3.3.7),

$$\begin{aligned} \|E_{k+1}\|_F &\leq \|E_k\left(I - \frac{s_k s_k^T}{s_k^T s_k}\right)\|_F + \|E_{k+1}\frac{s_k s_k^T}{s_k^T s_k}\|_F \\ &\leq \|E_k\|_F - \frac{1}{2\|E_k\|_F}\left(\frac{\|E_k s_k\|_2}{\|s_k\|_2}\right)^2 + \frac{\|E_{k+1}s_k\|_2}{\|s_k\|_2}. \end{aligned}$$

Thus, from some manipulation and the proof of Lemma 3.2.2,

$$\left(\frac{\|E_k s_k\|_2}{\|s_k\|_2}\right)^2 \leq 2\|E_k\|_F\left(\|E_k\|_F - \|E_{k+1}\|_F + \gamma\sigma(x_{k+1},x_k)\right).$$

Now, since Lemma 3.3.2 shows that $\{\|B_k\|_F\}$ is uniformly bounded and $\|x_k - x_*\|_2$ converges to zero at least q-linearly, $\{\|E_k\|_F\}$ is uniformly bounded by some b and $\sum_{j=0}^{\infty}\sigma(x_{j+1},x_j) < \infty$. This gives that

$$\begin{aligned} \sum_{j=0}^{k}\left(\frac{\|E_k s_k\|_2}{\|s_k\|_2}\right)^2 &\leq 2b\left(\|E_0\|_F - \|E_{k+1}\|_F + \gamma\sum_{j=0}^{k}\sigma(x_{k+1},x_k)\right) \\ &< \infty, \end{aligned}$$

and (3.3.5) follows.

A natural question is whether $\{B_k\}$ always converges to $J(x_*)$. The answer is that it does not and that the final approximation may be arbitrarily far from $J(x_*)$, c.f. pg. 185 of Dennis and Schnabel [1983]. On the same page is the following example which provides a comparison with Newton's method.

Let

$$F(x) = \begin{bmatrix} x_1^2 + x_2^2 - 2 \\ e^{x_1 - 1} + x_2^3 - 2 \end{bmatrix},$$

which has a root $x_* = (1,1)^T$. The sequences of points generated by Broyden's method and Newton's method from $x_0 = (1.5,2)^T$ with $B_0 = J(x_0)$ for Broyden's method, are shown below.

Broyden's Method			Newton's Method	
1.5	2.0	x_0	1.5	2.0
0.8060692	1.457948	x_1	0.8060692	1.457948
0.7410741	1.277067	x_2	0.8901193	1.145571
0.8022786	1.159900	x_3	0.9915891	1.021054
0.9294701	1.070406	x_4	0.9997085	1.000535
1.004003	1.009609	x_5	0.999999828	1.000000357
1.003084	0.9992213	x_6	0.99999999999992	1.0000000000002
1.000543	0.9996855	x_7	1.0	1.0
0.99999818	1.00000000389	x_8		
0.9999999885	0.999999999544	x_9		
0.99999999999474	0.99999999999998	x_{10}		
1.0	1.0	x_{11}		

The final approximation to the Jacobian generated by Broyden's method is

$$A_{10} \approx \begin{bmatrix} 1.999137 & 2.021829 \\ 0.9995643 & 3.011004 \end{bmatrix}, \text{ whereas } J(x_*) = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}.$$

We proved superlinear convergence by proving that $\sum_{k=0}^{\infty} \left(\frac{\|E_k s_k\|}{\|s_k\|} \right)^2 < \infty$. It is easy to see that if $\sum_{k=0}^{\infty} \left(\frac{\|E_k s_k\|}{\|s_k\|} \right) < \infty$, then $\{B_k\}$ converges, because

$$B_{k+1} - B_k = \frac{(y_k - B_k s_k) s_k}{s_k^T s_k} = \frac{E_k s_k s_k^T}{s_k^T s_k} + O(\sigma(x_{k+1}, x_k)).$$

Furthermore, if we just knew how fast $\frac{\|E_k s_k\|}{\|s_k\|}$ goes to zero, we could say more about the rate of convergence of $\{x_k\}$ to x_*. The only related result we know about is due to Gay [1979]. He proves that $x_{2n+1} \equiv x_*$ if $J(x)$ is constant for all x, i.e., F is affine. This allows him to prove the $2n$-step q-quadratic convergence of $\{x_k\}$ to x_*, and that in turn implies r-order at least $2^{\frac{1}{2n}}$.

Broyden's method is very popular in practice, for two main reasons. First, it generally requires fewer function evaluations than a finite difference Newton's method. Second, it can be implemented in ways that require only $0(n^2)$ arithmetic operations per iteration. We conclude this section by mentioning three basic ways to do this.

If n is small enough to allow storage of a full $n \times n$ Jacobian approximation and use of a QR factorization to solve (3.3), then we recommend a scheme due to Gill, Golub, Murray, and Saunders [1974]. In

this scheme, one has

$$B_c = Q_c R_c \quad \text{but wants} \quad B_+ = Q_+ R_+$$

and, since $B_+ = B_c + \dfrac{(y_c - B_c s_c) s_c^T}{s_c^T s_c} \equiv B_c + uv^T$,

$$B_+ = Q_c \left[R_c + (Q_c^T u) v^T\right] \equiv Q_c \left[R_c + wv^T\right].$$

Now a low multiple of n^2 operations is sufficient to get a QR factorization of a rank-one update to an upper triangular matrix and so

$$B_+ = Q_+ R_+ \text{ where } R_c + wv^T = \tilde{Q} R_+ \text{ and}$$

The second and third ways of implementing the update are both based on the Sherman-Morrison-Woodbury formula (c.f. Dennis and Schnabel [1983] pg. 188). It is easy to see that if B is nonsingular and $u, v \in \mathbb{R}^n$, then $B + uv^T$ is nonsingular if and only if $1 + v^T B^{-1} u \equiv \omega \neq 0$. Then,

$$(B + uv^T)^{-1} = B^{-1} - \frac{1}{\omega} (B^{-1} u) v^T B^{-1} \tag{3.3.8}$$

$$= \left[I - \frac{1}{\omega} (B^{-1} u) v^T\right] B^{-1}. \tag{3.3.9}$$

Broyden [1965] suggests updating the sequence of approximate inverse Jacobians using (3.3.8), i.e.,

$$(B^{-1})_+ = (B^{-1})_c + \frac{(s_c - (B^{-1})_c y_c) s_c^T (B^{-1})_c}{s_c^T (B^{-1})_c y_c}.$$

This is only feasible for about the same class of dense problems as the QR updating scheme. The QR scheme is more trouble to implement, but it has the advantage that the condition number of the current Jacobian approximation can always be monitored by using standard techniques to estimate the condition number of an upper triangular matrix. Approaches based on (3.3.8) may become of increased importance on parallel computers.

The final way of implementing Broyden's method is very useful for large problems, and it is based on (3.3.9). The idea can be found in Matthies and Strang [1979]. At the k^{th} step, we assume that we have some way of solving $B_0 x = b$ for any b; for example, we might have a sparse factorization of B_0. This allows us to recursively solve

$$B_{k+1} s_{k+1} = -F(x_{k+1})$$

by using (3.3.9) and solving

$$B_k w_k = \frac{1}{\|s_k\|_2}(y_k - B_k s_k),$$

computing $\omega_k = 1 + \frac{s_k^T}{\|s_k\|} w_k,$

solving $B_k \bar{s}_{k+1} = -F(x_{k+1}),$

and computing $s_{k+1} = \bar{s}_{k+1} - \frac{1}{\sigma_k} w_k \frac{s_k^T \bar{s}_{k+1}}{\|s_k\|_2}.$

This scheme must be restarted whenever the number of vectors allocated to store the update vectors is filled.

3.4. BFGS Method for Unconstrained Minimization

Now we discuss the selection of a secant approximation B_c to the Hessian matrix in the model (3.3.1). We could simply apply Broyden's method to find a root of $g(x)=0$. This is not done because more effective alternatives exist based on using more of the structure of the Hessian matrix which we are trying to approximate. In this section we will try to acquaint the reader with the high points of this rich material. For more information, see Dennis and More [1977], Dennis and Schnabel [1979], Fletcher [1980], or Dennis and Schnabel [1983].

The analog of (3.3.1) for solving $g(x)=0$ is the same except for the definition of y_c:

$$Bs_c = y_c = g(x_+) - g(x_c), \tag{3.4.1}$$

where B is meant to approximate $H(x_+)$. Equation (3.4.1) causes the quadratic model

$$m(x_+ + d) = f(x_+) + g(x_+)^T d + \frac{1}{2} d^T B_+ d$$

to interpolate $f(x_+)$, $g(x_+)$, and $g(x_c)$. However $H(x_+)$ is symmetric, while the Broyden update B_+ of B_c generally will not be. Also

$$\|\frac{1}{2}[B_+ + B_+^T] - H(x_+)\|_F \le \|B_+ - H(x_+)\|_F$$

indicates that we can approximate $H(x_+)$ more closely with a symmetric B_+.

Thus, it is natural to use the least change ideas of the last section to select B_+ as the projection of B_c onto the intersection of the matrices obeying (3.4.1) with the subspace $\mathcal{A}$ of symmetric matrices in $\mathbb{R}^{n\times n}$.

Then we obtain the PSB or Powell symmetric Broyden update

$$B_+ = B_c + \frac{(y_c - B_c s_c)\, s_c^T + s_c (y_c - B_c s_c)^T}{s_c^T s_c} - \frac{s_c^T (y_c - B_c s_c)\, s_c s_c^T}{(s_c^T s_c)^2}. \tag{3.4.2}$$

Note that B_+ will inherit symmetry from B_c. This update can be quite effective, and we will meet it again in the next section. It is not generally used in practice for two reasons. First, it can have difficulty with poorly scaled problems. Second, we will see in Section 4 that it is desirable that each B_+ be positive definite, but B_+ will only inherit positive definiteness from B_c under some conditions more restrictive than those required for (3.4.1) to have a symmetric positive definite solution B.

An obviously necessary condition for (3.4.1) to have a positive definition solution B is that

$$s_c^T y_c = s_c^T B s_c > 0. \tag{3.4.3}$$

We will now prove by construction that (3.4.3) is also sufficient for the existence of a symmetric and positive definite solution B to (3.4.1), by constructing the BFGS method due to Broyden [1969], Fletcher [1970], Goldfarb [1970] and Shanno [1970].

If we assume that B_c is symmetric and positive definite, then it can be written in terms of its Cholesky factors

$$B_c = L_c L_c^T, \quad L_c \text{ lower triangular}.$$

In fact, L_c will probably be available as a by-product of solving (3.3). We want B_+ to be a symmetric positive definite secant matrix which is equivalent to the existence of a nonsingular J_+ for which

$$B_+ = J_+ J_+^T \quad \text{and} \quad J_+ J_+^T s_c = y_c.$$

Let $v_c = J_+^T s_c$, so that $J_+ v_c = y_c$ and $v_c^T v_c = y_c^T s_c$ if J_+ exists. If we knew v_c, then it would seem reasonable to take

$$J_+ = L_c + \frac{(y_c - L_c v_c) v_c^T}{v_c^T v_c} \tag{3.4.4a}$$

in view of the success of Broyden's method. Transposing (3.4.4a), multiplying both sides by s_c, and using $J_+^T s_c = v_c$ and $v_c^T v_c = y_c^T s_c$ gives

$$v_c = J_+^T s_c = L_c^T s_c + v_c \cdot \left(1 - \frac{v_c^T L_c^T s_c}{y_c^T s_c}\right)$$

which simplifies to

$$v_c = \left[\frac{y_c^T s_c}{s_c^T B_c s_c} \right]^{\frac{1}{2}} L_c^T s_c \tag{3.4.4b}$$

and which exists if $y_c^T s_c > 0$.

Equations (3.4.4a, b) define J_+ such that $J_+ J_+^T = B_+$ is the BFGS update. It is customary to write the BFGS update as

$$B_+ = B_c + \frac{y_c y_c^T}{y_c^T s_c} - \frac{B_c s_c s_c^T B_c}{s_c^T B_c s_c} \tag{3.4.5}$$

but it is not efficient to actually calculate and factor such a matrix at each iteration. Instead Goldfarb [1976] recommends updating the Cholesky factorization $L_c L_c^T$ of B_c by applying the QR update scheme of subsection 3.3.3 to

$$J_+^T = L_c^T + u_c v_c^T$$

to get $J_+^T = Q_+ L_+^T$ in $0(n^2)$ operations. In fact, we don't need to form Q_+ since we only care about

$$B_+ = J_+ J_+^T = L_+ Q_+^T Q_+ L_+^T = L_+ L_+^T .$$

This Cholesky update form is the recommended way to implement the BFGS method if a full Cholesky factor can be stored. For large problems, updates can be saved and the Sherman-Morrison-Woodbury formula applied in the manner of Subsection 3.3.3. It is also possible to obtain the direct update to the inverse,

$$(B^{-1})_+ = (B^{-1})_c + \frac{(s_c - (B^{-1})_c y_c) s_c^T + s_c (s_c - (B^{-1})_c y_c)^T}{y_c^T s_c} - \frac{y_c^T (s_c - (B^{-1})_c y_c)^T s_c s_c^T}{(y_c^T s_c)^2} \tag{3.4.6}$$

The BFGS method can also be derived as the least-change symmetric secant update to $(B^{-1})_c$ in *any* inner product norm of the form $|||(\cdot)||| \equiv ||M_+(\cdot)M_+^T||$ where $M_+ M_+^T s_c = y_c$. In other words, the BFGS method is the result of projecting $(B_c)^{-1}$ into the symmetric matrices that satisfy $B^{-1} y_c = s_c$, and the projection is independent of a large class of inner product norms.

If instead we project B_c into the symmetric matrices for which $B_c s_c = y_c$, in any norm of the form $||M_+^{-1}(\cdot)M_+^{-T}||_F \equiv |||(\cdot)|||$ where $M_+ M_+^T s_c = y_c$, the result is the update formula

$$B_+ = B_c + \frac{(y_c - B_c s_c)y_c^T + y_c(y_c - B_c s_c)^T}{y_c^T s_c} - \frac{s_c^T(y_c - B_c s_c)\, y_c y_c^T}{(y_c^T s_c)^2}. \tag{3.4.7}$$

Equation (3.2.7) is called the DFP update for Davidon [1959], Fletcher and Powell [1963], and it also passes positive definiteness from B_c to B_+ whenever $y_c^T s_c > 0$.

Theorem 3.4.1 is a local superlinear convergence result for all three methods we have discussed in this subsection. This is a combination of three theorems from Broyden, Dennis and Moré [1973]. The proofs follow the same outline as the proof for Broyden's method.

Theorem 3.4.1. Let f satisfy the hypothesis of Theorem 2.2.1. Then, there exist $\varepsilon > 0$, $\delta > 0$ such that the sequences generated respectively by the BFGS, DFP, or PSB methods exist and converge q-superlinearly to x_* from any x_0, B_0 for which $\|x_* - x_0\| \le \varepsilon$ and $\|B_0 - H(x_*)\| \le \delta$. Furthermore, the PSB method converges if $H(x_*)$ is nonsingular but not positive definite.

The BFGS method seems to work especially well in practice. Generally it requires more iterations to solve a given problem than a finite-difference Newton method would, but fewer function and gradient evaluations. Most experts feel that a property which contributes to the success of the DFP and BFGS methods is that the iteration sequences are invariant with respect to linear basis changes under reasonable hypotheses. This property is not shared by the PSB method. The superiority of the BFGS to the DFP is still an interesting research topic and is discussed briefly in Section 6.3.

A final issue is how to choose the initial Hessian approximation B_0 in a BFGS method. In analogy to Section 3.3 we could choose B_0 to be a finite difference approximation to $\nabla^2 f(x_0)$, but besides being expensive, an initial Hessian often is indefinite and then may be perturbed anyhow as we will see in Section 4. Instead, the common practice is to set $B_0 = I$ so that B_0 is positive definite and the first step is in the steepest descent direction. This choice may not correctly reflect the scale of the Hessian, and so a one-time scaling correction often is applied during the first iteration; see Shanno and Phua [1978a] or Dennis and Schnabel [1983].

3.5 Sparse Finite-Difference and Secant Methods

In this section, we will consider briefly the incorporation of sparsity into the methods of the previous three subsections. We will see that there are important opportunities for efficiency in finite-difference Jacobian and Hessian approximations. The development of effective sparse secant methods will be seen to be more problematical.

Medium size dense problems are usually solvable by library subroutines based on local models we have already considered. When a problem is large enough to make sparsity an important consideration, it often is useful to incorporate the source of the problem into the method. In Section 6.1 we will discuss some promising approaches to this end. Here we discuss general purpose approaches.

We begin with a discussion of special finite-difference methods for sparse problems.

Curtis, Powell and Reid [1974] began one of the most elegant and useful lines of research on sparse nonlinear problems. They noticed that if the sparsity structure of $J(x)$ is such that some subset C_J of the column indices has the property that there is at most one nonzero in any row of the submatrix composed of these columns, then all the nonzero element in this submatrix of $\Delta F(x_c, h)$ could be calculated from the single function difference

$$F(x_c + \sum_{j \in C_J} h_j e_j) - F(x_c) .$$

In order to illustrate the enormous savings possible in finite-difference methods, notice that if $J(x)$ is tridiagonal, then only three extra values of $F(x)$ are needed for any n to build $\Delta F(x_c, h)$. They are

$$\begin{aligned} F(x_c + h_1 e_1 + h_4 e_4 h_7 e_7 + \cdots) &\equiv F(x_c + d_1) \\ F(x_c + h_2 e_2 + h_5 e_5 h_8 e_8 + \cdots) &\equiv F(x_c + d_2) \\ F(x_c + h_3 e_3 + h_6 e_6 h_9 e_9 + \cdots) &\equiv F(x_c + d_3) . \end{aligned}$$

The submatrix consisting of the first, forth, seventh, tenth, ... columns of $\Delta F(x_c, h)$ is then determined from the function difference $F(x_c + d_1) - F(x_c)$. The other two submatrices are determined in the same way from $F(x_c + d_2) - F(x_c)$ and $F(x_c + d_3) - F(x_c)$.

Curtis, Powell, and Reid suggested some effective heuristics to keep the number of submatrices, and hence the number of function evaluations, small. Coleman and Moré [1983] and Coleman, Garbow, and More´* [1984] exploited the connection between a subclass of graph coloring problems and the problem of

determining how to partition the columns of $J(x)$ to save further on function evaluations. Although they proved that the problem of minimizing the number of submatrices is NP-complete, they developed some very useful heuristics.

Li [1986] noticed that if no component of $s_c = x_+ - x_c$ is zero, then the function value at the past iteration $F(x_-)$ can be used to reduce the number of extra function evaluations in either of these approaches by one. In the tridiagonal case, this reduces to using $h = s_c$ and $F(x_c) - F(x_c - d_1)$, $F(x_c - d_1) - F(x_c - d_1 - d_2)$, and $F(x_c - d_1 - d_2) - F(x_-)$ respectively to calculate the three submatrices. He suggests leaving the j^{th} column unchanged if $(s_c)_j = 0$. His analytical and computational results support this approach.

This approach can also be applied to approximate a sparse Hessian from gradient values. Powell and Toint [1979] developed methods to further reduce the number of gradient evaluations required in this case. To illustrate their main idea, assume that $H(x)$ is diagonal except for a full last row and column. We can then approximate the last column by $[g(x_c + h_n e_n) - g(x_c)] \cdot h_n^{-1}$ and the last row by its transpose. The first $n-1$ components of $g(x_c + \sum_{j=1}^{n-1} h_j e_j) - g(x_c)$ suffice to obtain an approximation to the rest of $H(x_c)$. Thus two extra gradient evaluations suffice while n would be required for the same sparsity pattern without symmetry. Powell and Toint also suggest a more complex indirect approximation method. Coleman and Moré [1984] and Coleman, Garbow, and More´* [1985] showed that problem of minimizing the number of extra gradient evaluations also is related to graph coloring, and again developed useful heuristics although this problem is also NP-complete. Goldfarb and Toint [1984] show the connection between the finite-difference approximation and tiling the plane when the nonlinear problem arises from numerically solving a partial differential equation.

Now we consider sparse secant methods. Schubert [1970] and Broyden [1971] independently suggested a sparse form of Broyden's method. Reid [1973] showed that their update is the Frobenius norm least change secant update to B_c with A taken to be the set of matrices with the sparsity of $J(x)$.

In order to state the algorithm, let us define P_i to be the l_2 projection of $\mathbb{R}^n$ onto the subspace z_i of $\mathbb{R}^n$ consisting of all vectors with zeros in every row position for which the corresponding column position of the i^{th} row of $J(x)$ is always zero. That is, $P_i v$ zeroes the elements of v corresponding to the zero ele-

ments of row i of $J(x)$ and leaves the others unchanged. We also need the pseudoinverse notation a^+ to denote 0 when the real scalar is $a=0$ and a^{-1} otherwise. Then the sparse Broyden or Schubert method is:

Given B_c sparse, and x_c

solve $B_c s_c = -F(x_c)$.

Set $x_+ = x_c + s_c$ and

$$B_+ = B_c + \sum_{i=1}^{n} [P_i(s_c)^T P_i(s_c)]^+ e_i^T (y_c - B_c s_c) e_i P_i(s_c)^T . \tag{3.5.1}$$

Note that (3.5.1) reduces to Broyden's method if $J(x)$ has no nonzeroes.

Marwil [1979] gave the first complete proof that (3.5.1) is locally q-superlinearly convergent under the hypothesis of Theorem 3.3.3. In practice, this method is cheaper than the Broyden update, but the savings are small because there is no reduction over the cost of Newton's method in solving for the quasi-Newton step. An important practical use of this method is to update a submatrix of an approximate Jacobian, the other columns of which might be generated by finite differences as suggested above. Dennis and Li [1986] test and analyze a strategy based on using heuristics of Coleman and Moré to pick out subsets of columns that can be very efficiently approximated by finite differences. The remaining columns are then updated by (3.5.1). The results are very good, and there are indications that similar approaches are useful in engineering applications.

For unconstrained optimization, Marwil [1978] and Toint [1977] constructed a sparse analog of the PSB update (3.4.2). Toint analyzed the method under some safeguarding, and Dennis and Walker [1981] give a complete proof of local q-superlinear convergence. An example by Sorensen [1981], however, raises doubts about the utility of the method. We will not dwell on this topic because the update seems to share the shortcomings of the PSB, and in addition, it requires the solution of an extra $n \times n$ positive definite linear system with the same sparsity as $H(x)$ for the update of B_c to B_+. Thus this method has not had a major practical impact.

Professor Angelo Lucia of Clarkson reports good results using the cheap and simple expedient of projecting B_+ defined by (3.5.1) into the subspace of symmetric matrices, i.e., he uses the approximate Hessian defined by $\frac{1}{2}[B_+ + B_+^T]$. Steihaug [1980] had shown this method to be locally q-superlinearly convergent.

Unfortunately, extending the BFGS and DFP algorithms to sparse problems seems at a dead end. One problem is that a sparse positive definite approximation may not exist in some cases where $y_c^T s_c > 0$. A more pervasive problem is that sparsity is not invariant under general linear basis changes, while the essence of these secant methods is their invariance to any linear basis changes. We will comment in Section 6.1 on some work that uses more fundamental problem structure to get around the problems of the straightforward approach to least change secant methods for sparse unconstrained minimization.

4. Globally Convergent Methods

The methods for unconstrained optimization discussed in Sections 2 and 3 are *locally convergent* methods, meaning that they will converge to a minimizer if they are started sufficiently close to one. In this section we discuss the modifications that are made to these methods so that they will converge to a local minimizer from a poor starting point x_0. Methods with this property are called *globally convergent.*

Two main approaches have emerged for making the methods of Sections 2 and 3 more globally convergent while retaining their excellent local convergence properties. They are *line search* methods and *trust region* methods. Both are used in successful software packages, and neither has been shown clearly superior to the other. Furthermore, both approaches certainly will play important roles in future research and development of optimization methods. Therefore we cover both approaches, in Sections 4.2 and 4.3 respectively. We briefly compare these approaches in Section 4.4.

The basic idea of both line search and trust region methods, is that they use the quickly convergent local methods of Sections 2 and 3 when they are close to a minimizer, and that when these methods are not sufficient, they use some reliable approach that gets them closer to the region where local methods will work. The basic concept behind this global phase is that of a *descent direction*, a direction in which $f(x)$ initially decreases from the current iterate x_c. Descent directions and their relation to local methods are discussed in Section 4.1. Included in Section 4.1 is a discussion of the well-known, but slow, *method of steepest descent.*

Global strategies for solving systems of nonlinear equations are obtained from global strategies for unconstrained optimization, for example by applying the methods of this section to $f(x) = \| F(x) \|^2$. For a discussion of this topic, see e.g. Dennis and Schnabel [1983], Section 6.5.

4.1 Descent Directions

The basic strategy of most globally convergent methods for unconstrained optimization is to decrease $f(x)$ at each iteration. Fundamental to this is the idea of a *descent direction* from x_c, a direction d from x_c in which $f(x)$ initially decreases.

The proof of Theorem 1.1 (the necessary condition for unconstrained optimization) showed that d is a descent direction from x_c if and only if

$$\nabla f(x_c)^T d < 0,$$

i.e., the directional derivative in the direction d is negative. In this case, for all sufficiently small $\varepsilon > 0, f(x_c+\varepsilon d) < f(x_c)$. Thus given a descent direction d, one can always choose a new iterate $x_+ = x_c+\varepsilon d$ so that $f(x_+) < f(x_c)$. This property is the basis of globally convergent methods.

A natural question to ask is whether the local methods for unconstrained optimization discussed in Sections 2 and 3 yield steps in descent directions. These methods were derived by considering the local quadratic model of $f(x)$ around x_c, which in general had the form

$$m_c(x_c+d) = f(x_c) + \nabla f(x_c)^T d + \frac{1}{2} d^T H_c d. \tag{4.1.1}$$

They then chose $d = -H_c^{-1}\nabla f(x_c)$ causing $x_+=x_c+d$ to be the critical point of (4.1.1).

If H_c is positive definite, x_+ is the unique minimizer of the model; furthermore

$$\nabla f(x_c)^T d = -\nabla f(x_c)^T H_c^{-1} \nabla f(x_c) < 0$$

so that d is a descent direction. On the other hand, if H_c is not positive definite, then not only doesn't the model have a minimizer, but also d may not be a descent direction.

In implementations of the leading secant methods for unconstrained minimization such as the BFGS method, H_c always is positive definite. Thus the steps generated by these methods always are in descent directions.

When $H_c = \nabla^2 f(x_c)$, however, it may not be positive definite when x_c is far from a local minimizer. Thus the Newton step $d = -\nabla^2 f(x_c)^{-1}\nabla f(x_c)$ is not necessarily in a descent direction. We will see that line search and trust region methods deal with indefinite Hessian matrices in different ways.

The idea of choosing x_+ to be a step from x_c in a descent direction d also leads naturally to the idea of taking steps in the "steepest" descent direction. By this one means the direction d for which the initial rate of decrease from x_c in the direction d is greatest. For this definition to make sense, the direction d must be normalized; then we can define the "steepest" descent direction as the solution to

$$\min_{d \in R^n} \nabla f(x_c)^T d \quad \text{subject to } \|d\|=1. \tag{4.1.2}$$

The solution to (4.1.2) depends on the choice of norm; for the Euclidean norm it is $d = -\nabla f(x_c) / \| \nabla f(x_c) \|_2$, which is generally known as the *steepest descent direction.*

One classic minimization algorithm is based solely on the steepest descent direction. It is to choose each new iterate x_{k+1} to be the minimizer of $f(x)$ in this direction, i.e.

$$\text{choose } t_k \text{ to solve} \quad \underset{t>0}{\text{minimize}}\, f(x_k - t\ \nabla f(x_k))$$

$$x_{k+1} = x_k - t_k \nabla f(x_k). \tag{4.1.3}$$

This is known as the *method of steepest descent.*

The method of steepest descent is an example of a globally convergent method. By this we mean that if a sequence of iterates $\{x_k\}$ is generated by (4.1.3) for a continuously differentiable $f(x)$ that is bounded below, then $\lim_{k\to\infty} \nabla f(x_k) = 0$. However the method has several important practical drawbacks. Most importantly, it usually is slow. If the method converges to x_*, then it can be shown that

$$\limsup_{k\to\infty} \frac{\|M\ (x_{k+1} - x_*)\|}{\|M\ (x_k - x_*)\|} \le c \tag{4.1.4}$$

where $M^2 = \nabla^2 f(x_*)$, and $c = (\lambda_1 - \lambda_n)/(\lambda_1 + \lambda_n)$ for (λ_1, λ_n) the largest and smallest eigenvalues of $\nabla^2 f(x_*)$. Furthermore, for any $f(x)$, (4.1.4) can be shown to be an equality for some starting x_0. Thus the method is only linearly convergent and may be very slowly convergent for problems with even slightly poorly conditioned $\nabla^2 f(x_*)$. Secondly, as written (4.1.3) requires the solution of an exact one-variable minimization problem at each iteration. The steepest descent method may be implemented with an inexact minimization and still retain (4.1.4) but the work per iteration may still be large. Thirdly, the method is very sensitive to transformations of the variable space. If the variable space is changed to $\hat{x} = T \cdot x$, the Hessian matrix in this new variable space becomes $T^{-T}\ \nabla^2 f(x)\ T^{-1}$, so that by the above discussion the rate of convergence may be significantly altered. Indeed, the effectiveness of even a single steepest descent iteration in reducing $f(x)$ may depend significantly upon the units used in defining the variables. In contrast, the performance of the Newton or BFGS methods is unaffected by linear transformations of the variable space.

For these reasons, the method of steepest descent is not recommended as a general purpose optimization method. We will see, however, that steepest descent steps play a role in the trust region methods discussed in Section 4.2 and in the conjugate gradient methods of Section 5.2. Furthermore, versions of the

method of steepest descent continue to be used successfully in some practical applications, for example problems where x_0 is close to x_* and the cost of computing $\nabla^2 f(x)$ is prohibitive, even though conjugate gradient or BFGS methods may be more efficient for such problems.

4.2 Line Search Methods

The general idea of a *line search method* is to choose a descent direction from x_c at each iteration, and select the next iterate x_+ to be a point in this direction that decreases $f(x)$. That is

$$\text{choose } d_c \text{ for which } \nabla f(x_c)^T d_c < 0$$

$$\text{choose } x_+ = x_c + t_c d_c,\ t_c > 0, \text{ so that } f(x_+) < f(x_c).$$

Note that the method of steepest descent fits this framework.

Modern line search methods differ from the method of steepest descent in three important ways: 1) d_c usually is chosen to be the Newton or secant direction; 2) t_c is chosen by a procedure that requires much less work than an exact minimization; and 3) $t_c = 1$ is used whenever possible, so that the line search method reduces to Newton's method or a secant method close to a local minimizer. In this section we summarize the convergence properties of such methods, the selection of the search direction d_c, and practical procedures for calculating t_c.

Starting with the work of Armijo [1966] and Goldstein [1967], it has been shown that line search methods will be globally convergent if each step satisfies two simple conditions. The first is that the decrease in $f(x)$ is sufficient in relation to the length of the step $s_c = t_c d_c$; the relation

$$f(x_+) < f(x_c) + \alpha \nabla f(x_c)^T s_c \tag{4.2.1}$$

usually is chosen to implement this condition where $\alpha \in (0,1)$ is some constant. Note that for any sufficiently small step in a descent direction d_c, (4.2.1) is satisfied for any $\alpha < 1$. The second condition is that the step is not too short. The equation

$$\nabla f(x_+)^T s_c \geq \beta \nabla f(x_c)^T s_c \tag{4.2.2}$$

is most commonly chosen to implement this condition, where $\beta \in (\alpha,1)$; it says that the step must be long enough so that the directional derivative increases by some fixed fraction of its original magnitude.

The main value of equations (4.2.1) and (4.2.2) is that incorporating them into a line search algorithm leads to a practical and globally convergent method. Theorem 4.2.1 says that given any descent direction d_c and $0 < \alpha < \beta < 1$, it is always possible to choose $t_c > 0$ so that $x_+ = x_c + t_c d_c$ satisfies (4.2.1) and (4.2.2) simultaneously. Theorem 4.2.2 says that if every iterate is chosen in this way and the directions d_c are selected reasonably, the method will be globally convergent. For proofs of these theorems, see Wolfe [1969,1971] or Dennis and Schnabel [1983].

Theorem 4.2.1 Let $f : R^n \to R$ be continuously differentiable and bounded below. Let $x_k \in R^n$, $d_k \in R^n$ satisfy $\nabla f(x_k)^T d_k < 0$. Then if $0 < \alpha < \beta < 1$, there exist $0 < t_1 < t_2$ such that for any $t_k \in [t_1, t_2]$, $x_{k+1} = x_k + t_k d_k = x_k + s_k$ satisfies

$$f(x_{k+1}) < f(x_k) + \alpha \nabla f(x_k)^T s_k \tag{4.2.3}$$

and

$$\nabla f(x_{k+1})^T s_k \geq \beta \nabla f(x_k)^T s_k \,. \tag{4.2.4}$$

Theorem 4.2.2 Let $f : R^n \to R$ be continuously differentiable and bounded below, and assume $\nabla f(x)$ is Lipschitz continuous in R^n. Given $x_0 \in R^n$, suppose the sequence $\{x_k\}$ is defined by $x_{k+1} = x_k + d_k$, $k = 0,1,...$, and that there exists $\sigma > 0$ such that for each $k > 0$,

$$\nabla f(x_k)^T s_k < -\sigma \|\nabla f(x_k)\|_2 \|s_k\|_2 \tag{4.2.5}$$

and (4.2.3), (4.2.4) are satisfied. Then either $\nabla f(x_k) = 0$ for some k, or $\lim_{k \to \infty} \nabla f(x_k) = 0$.

The only restriction in Theorem 4.2.2 that we have not yet discussed is equation (4.2.5). This simply says that each step direction must be a descent direction where in addition, the angle between s_c and the negative gradient is less than some fixed angle less than 90° . For example, if $s_k = -H_k^{-1} \nabla f(x^k)$ where H_k is $\nabla^2 f(x_k)$ or any approximation to it, then (4.2.5) is satisfied if the condition numbers of H_k are uniformly bounded above. This is not a big restriction in practice although not all methods can be shown to enforce it in theory.

Other conditions can be substituted for (4.5) and still allow Theorem 4.2.1 and 4.2.2 to hold. A common substitution for (4.2.2) is

$$|\nabla f(x_+)^T s_c| \leq \beta |\nabla f(x_c)^T s_c| \tag{4.2.6}$$

again with $\beta \in (\alpha,1)$; as $\beta \to 0$, (4.2.6) causes x_+ to approach a minimizer of $f(x)$ along the line in the direction d_c from x_c . Another substitution for (4.2.2) which causes our theorems to remain true is

$$f(x_+) \geq f(x_c) + \gamma \nabla f(x_c)^T s_c$$

for some $\gamma \in (0,\alpha)$.

Note that (4.2.2) (or (4.2.6)) and $\nabla f(x_c)^T d_c < 0$ imply

$$(\nabla f(x_+) - \nabla f(x_c))^T s_c \geq (\beta-1) \nabla f(x_c)^T s_c > 0$$

which is the condition (3.4.3) that we saw is necessary and sufficient for the existence of symmetric and positive definite secant updates. Thus by enforcing (4.2.2) or (4.2.6) in a BFGS method, we also ensure that it is possible to make a positive definite update at each iteration.

Two practical issues remain in applying Theorems 4.2.1 and 4.2.2 to obtain a practical line search algorithm : the choices of the step direction d_c , and the efficient calculation of the step length t_c to satisfy (4.2.1) and (4.2.2). In addition, we need to show how we retain the fast local convergence of the methods discussed in Sections 2 and 3.

Our methods are based upon the quadratic model (4.1.1) where $H_c = \nabla^2 f(x_c)$ or an approximation to it. When H_c is a BFGS approximation, then it is positive definite and line search methods use $d_c = -H_c^{-1} \nabla f(x_c)$. We saw in Section 4.1 that this d_c is always a descent direction. Generally BFGS based line search methods do not explicitly enforce (4.2.5); it can be shown to be true in theory under certain assumptions (see e.g. Broyden, Dennis and More [1973], Powell [1976]).

When $H_c = \nabla^2 f(x_c)$ or a finite difference approximation to it, then H_c may or may not be positive definite. The standard practice, due to Gill and Murray [1974] is to attempt the Cholesy factorization of H_c in such a way that the result is the factorization $L_c L_c^T$ (or $L_c D_c L_c^T$) of $(H_c + E_c)$. Here L_c is lower triangular, D_c is positive diagonal, and E_c is a non-negative diagonal matrix which is zero if H_c is positive definite and not too badly conditioned. Then d_c is obtained by solving $L_c L_c^T d_c = -\nabla f(x_c)$, so that $d_c = -(H_c + E_c)^{-1} \nabla f(x_c)$. Thus d_c is the Newton direction if H_c is safely positive definite, as it will be near a strong local minimizer and usually at most other iterations as well. Otherwise d_c is some descent direction related to the Newton direction. In all cases, the cost of the factorization is only $0(n^2)$ operations more than a normal Cholesky factorization, d_c obeys (4.2.5), and the size of E_c is bounded in terms of the size of H_c . Schnabel and Van Vleck [1987] recently have developed a new version of this modified

Cholesy decomposition which appears to reduce the size of E_c while improving the conditioning of $H_c + E_c$.

Thus any line search method, whether it chooses H_c to be the Hessian or a BFGS approximation to it, will use $d_c = -H_c^{-1} \nabla f(x_c)$ as the search direction in the neighborhood of a minimizer x_* where $\nabla^2 f(x_*)$ is positive definite. If the steplength $t_c = 1$ is permissible, this means that the global convergence results of this section are consistent with the fast local convergence of Sections 2.2 and 3.4. Theorem 4.2.3, due to Dennis and More [1974], shows that this is the case as long as $\alpha < \frac{1}{2}$ in (4.2.1).

Theorem 4.2.3 Let $f : \mathbf{R}^n \to \mathbf{R}$ have a Lipschitz continuous Hessian in an open convex set D . Consider a sequence x_k generated by $x_{k+1} = x_k + t_k d_k$, where $\nabla f(x_k)^T d_k < 0$ for all k and t_k is chosen to satisfy (4.2.1) with $\alpha < \frac{1}{2}$, and (4.2.2). If x_k converges to a point $x_* \in D$ at which $\nabla^2 f(x_k)$ is positive definite, and if

$$\lim_{k \to \infty} \frac{\|\nabla f(x_k) + \nabla^2 f(x_k) d_k\|}{\|d_k\|} = 0 , \tag{4.2.7}$$

then there is an index $k_0 \geq 0$ such that for all $k \geq k_0$, $t_k = 1$ is admissible. Furthermore, $\nabla f(x_*) = 0$, and if $t_k = 1$ for all $k \geq k_0$, then $\{ x_k \}$ converges q-superlinearly to x_* .

If exact Hessians are used, d_k will be $-\nabla^2 f(x_k)^{-1} \nabla f(x_k)$ for all x_k sufficiently close to x_* , so that (4.2.7) is trivially true and quadratic convergence is achieved by using $t_k = 1$. In a BFGS method, the analysis of Broyden, Dennis, and More [1973] or Powell [1976] shows that (4.2.7) is true so that q-superlinear convergence can be retained.

From Theorem 4.2.3, we see that to combine fast local convergence with global convergence, a practical procedure for selecting the steplength t_c should always try $t_c = 1$ first (at least near a minimizer) and use it if it is admissible. Beyond this, experience has shown that a practical procedure for choosing t_c to satisfy (4.2.1) and (4.2.2) should aim to be as efficient as possible, in that it chooses α and β so that there is a wide range of points satisfying (4.2.1) and (4.2.2), and uses the first point that it finds in this range rather than trying to closely approximate the minimizer of $f(x)$ along the line $x_c + t\, d_c$. Many strategies for accomplishing these goals efficiently have been proposed, and probably every line search that is coded is unique in some way. Algorithm 4.2.1 indicates the structure of a representative line search.

There are four possible stages in this line search. If $t_c = 1$ satisfies both (4.2.1) and (4.2.2), then $x_+ = x_c + d_c$, and no further line search calculations are performed. If $t_c = 1$ is not admissible because it fails (4.2.1), then t_c will be decreased. This is most often done by safeguarded quadratic interpolation. In this procedure the minimizer t_m of the one-dimensional quadratic approximation $q(t)$ to $f(x_c + td_c)$ that interpolates $f(x_c)$, $\nabla f(x_c)^T d_c$, and $f(x_c + t_c d_c)$ is calculated by

$$t_m = \frac{-t_c^2 \nabla f c^T d_c}{2[f(x_c + t_c d_c) - f(x_c) - t_c \nabla f(x_c)^T d_c]}, \tag{4.2.8}$$

and the next stepsize is then set to max $\{t_m, c_1 t_c\}$, where typically $c_1 = 0.1$. (It can be shown that $t_m > t_c/2(1-\alpha)$). This Init-Decrease stage may be repeated one or more times if the new $x_c + t_c d_c$ continues to fail (4.2.1). Sometimes a form of safeguarded cubic interpolation is used instead in this stage; see e.g. Dennis and Schnabel [1983].

Alternately, if $t_c = 1$ satisfies (4.2.1) but not (4.2.2), t_c will be increased. Generally a simple rule like $t_c = 2t_c$ is used although more sophisticated strategies are possible. This Init-Increase stage also may be

---

Algorithm 4.2.1 Line Search Structure

Given $f(x) : \mathbf{R}^n \to \mathbf{R}$, x_c, $f(x_c)$, $\nabla f(x_c)$, descent direction d_c

```
tlow := 0 , tup := ∞ , done := false, t_c := 1
Repeat
    evaluate f(x_c + t_c d_c)
    If x_c + t_c d_c satisfies (4.2.1) then
        evaluate ∇f(x_c + t_c d_c)
        If x_c + t_c d_c satisfies (4.2.2) Then
            done := true
        Else
            tlow := t_c
            If tup = ∞ then
                t_c := Init-Increase (t_c)
            Else t_c := Refine (t_c , tlow , tup)
    Else
        tup := t_c
        If tlow = 0 then
            t_c := Init-Decrease (t_c)
        Else t_c := Refine (t_c , tlow , tup)
Until done = true
```

---

repeated if $x_c + t_c d_c$ continues to satisfy (4.2.1) and fail (4.2.2).

After one or more repetitions of either the Init-Increase or Init-Decrease phase, either an admissible $x_c + t_c d_c$ is found, or it must be the case that the last two values of t_c that have been tried bracket an acceptable value of t. That is, the one with the lower value of t_c, $tlow$ must satisfy (4.2.1) but not (4.2.2), while the one with the higher value of t_c, tup, must fail (4.2.1). In this case an admissible t_c must be in $(tlow, tup)$, and it is identified by the final, Refine, phase of the line search. Let $\delta = tup - tlow$. A typical Refine phase would calculate

$$t_m = tlow - \frac{\delta^2 \nabla f(tlow)^T d_c}{f(tup) - f(tlow) - \delta \nabla f(tlow)^T d_c}$$

the minimizer of the one dimensional quadratic interpolating $f(tlow)$, $\nabla f(tlow)^T d_c$, and $f(tup)$, and then set $t_c = \min\{\max\{t_m, tlow + c_2\delta\}, tup - c_2\delta\}$ where typically $c_2 = 0.2$. This phase may also be repeated one or more times.

In theory it can be shown that our line search terminates in a finite number of iterations. In practice, very little work usually is necessary. Experience has shown that line search algorithms with relatively loose tolerances generally produce algorithms that require fewer total number of function and derivative evaluations to reach the minimizer than algorithms with tight tolerances. Typical line search algorithms set $\alpha = 10^{-4}$ in (4.2.1), so that virtually any decrease in $f(x)$ is acceptable, and β between 0.7 and 0.9 in (4.2.2), so that only a small decrease in the magnitude of the directional derivative is required. Due to these tolerances, $t_c = 1$ is admissible much of the time, and when it is not, generally one or at most two more values of t_c must be attempted. Thus the three procedures described above, Init-Decrease, Init-Increase, and especially Refine, are used only infrequently.

The above line search is related to the ones described by Shanno and Phua [1978b], Fletcher [1980], Dennis and Schnabel [1983], and many other authors. Many line searches have been proposed, especially variants that deal differently with the case when the Hessian is indefinite; see e.g. McCormick [1977], More and Sorenson [1979]. Some line searches only allow $t_c \leq 1$ and only enforce (4.2.1); in this case Algorithm 4.2.1 is much simpler as the Init-Increase and Refine stages and the check for (4.2.2) are eliminated. The techniques of Shultz, Schnabel, and Byrd [1985] show that such a line search still leads to a globally convergent algorithm, but satisfaction of the condition (3.4.3) for positive definite secant updates

is not guaranteed. Finally, some line search algorithms do not start with $t_c=1$ if the previous step was very short; good strategies for doing this are closely related to the trust region approach that we discuss next.

4.3 Trust Region Methods

Trust region methods are the other main group of methods for ensuring global convergence while retaining fast local convergence in optimization algorithms. The fundamental difference between line search and trust region methods is how they combine the use of the quadratic model with the choice of the step length. We saw in Section 4.2 that in a line search method, the quadratic model $m_c(x_c + d)$ given by (4.1.1) is used to obtain a search direction $d_c=-H_c^{-1}g_c$ (or $-(H_c + E_c)^{-1}g_c$ if H_c is not positive definite), and then a steplength is chosen. The procedure for choosing the steplength does not make further use of the Hessian (approximation) H_c .

A trust region method takes the different philosophy that one *first* chooses a trial step length Δ_c , and then uses the quadratic model to select the best step of (at most) this length for the quadratic model by solving

$$\underset{s\in\mathbf{R}^n}{\text{minimize}} \quad m_c(x_c + s)=f(x_c) + \nabla f(x_c)^T s + \frac{1}{2}s^T H_c s \qquad (4.3.1)$$

$$\text{subject to} \quad \|s_c\| \le \Delta_c$$

The trial step length Δ_c is considered an estimate of how far we *trust* the quadratic model, hence it is called a *trust radius* and the resultant method is called a *trust region* method. We will see below that Δ_c is closely related to the length of the successful step at the previous iteration, and may be adjusted as the current iteration proceeds. First we describe the solution to (4.3.1) in Theorem 4.3.1. An early proof of much of Theorem 4.3.1 is given in Goldfeldt, Quandt, and Trotter [1966]; other seminal references include Gay [1981] and Sorensen [1982].

Theorem 4.3.1 Let $g_c = \nabla f(x_c)\in\mathbf{R}^n$, $H_c\in\mathbf{R}^{n\times n}$ symmetric, $\Delta_c > 0$. Let $\lambda_1\in\mathbf{R}$ denote the smallest eigenvalue of H_c and let $v_1\in\mathbf{R}^n$ denote the corresponding eigenvector. Then if H_c is positive definite and $\|H_c^{-1}g_c\| \le \Delta_c$, $s_c=-H_c^{-1}g_c$ is the unique solution to (4.3.1). Otherwise the solution to (4.3.1) satisfies $\|s_c\|=\Delta_c$ and

$$(H_c + \mu_c I)s_c = -g_c$$

for some $\mu_c \geq 0$ where $H_c + \mu_c I$ is at least positive semi-definite. Furthermore either $H_c + \mu_c I$ is positive definite and

$$s_c = -(H_c + \mu_c I)^{-1} g_c \tag{4.3.2}$$

for the unique $\mu_c > \max\{0, -\lambda_1\}$ for which $\|s_c\| = \Delta_c$, or $\mu_c = -\lambda_1$ and

$$s_c = -(H_c - \lambda_1 I)^{+} g_c + \omega v \ , \tag{4.3.3}$$

where + denotes the Moore-Penrose pseudouniverse, and $\omega \in \mathbf{R}$ is chosen so that $\|s_c\| = \Delta_c$. If H_c is positive definite, the solution must be given by (4.3.2); the case (4.3.3) only occurs if H_c is indefinite and $\|(H_c + \mu_c I)^{-1} g_c\| < \Delta_c$ for all $\mu_c \geq -\lambda_1$.

Theorem 4.3.1 indicates several differences between the step taken in line search and trust region methods. From (4.3.2) we see that, even if H_c is positive definite, the trust region step is not always in the Newton direction $-H_c^{-1}\nabla f(x_c)$. In fact, it is straightforward to show that for all $\mu \geq \max\{0, -\lambda_1\}$, $\|(H_c + \mu I)^{-1}\nabla f(x_c)\|$ is a monotonically decreasing function of μ. Thus as $\Delta_c \to 0$, $\mu \to \infty$, and $-(H_c + \mu I)^{-1}\nabla f(x_c) \to -\nabla f(x_c)/\mu$. Therefore for small Δ_c, the trust region step is nearly in the steepest descent direction, while as Δ_c increases, the trust region step approaches and ultimately becomes the Newton step $-H_c^{-1}\nabla f(x_c)$ as long as H_c is positive definite. This is depicted in Figure 4.3.1.

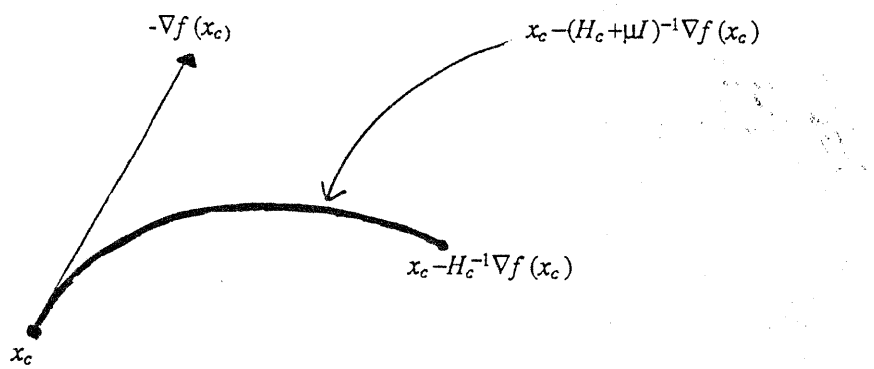


Figure 4.3.1 The trust region curve

A second difference between line search and trust region methods is how they deal with the case when H_c is not positive definite. Since the minimization of an indefinite or negative definite quadratic model is not mathematically well-posed, a line search method must perturb a non-positive definite Hessian to be positive definite, as was seen in Section 4.2. On the other hand, the minimization of such a model within some closed region is well defined and reasonable, and this is what the trust region method does. Indeed, even if x_c is a maximum or saddle point, (4.3.1) is well-defined with a solution given by (4.3.3).

The above discussion indicates two attractive properties of trust region methods. First, small steps are in the steepest descent direction, the best direction for sufficiently small steps, while the Newton step is used when it is within the trust region and H_c is positive definite, thereby hopefully preserving fast local convergence. We will see that this property is retained by all the practical approximations to the ideal trust region step that are discussed later in this section. Second, the trust region method deals naturally with indefinite Hessian matrices. This will be seen to lead to stronger convergence results than were possible for the line search methods of Section 4.2.

On the other hand, the ideal trust region step described in Theorem 4.3.1 is difficult to calculate. The main difficulty is that there is no closed formula that gives the unique $\mu_c \geq \max\{0,-\lambda_1\}$ for which $\|(H_c + \mu_c I)^{-1}\nabla f(x_c)\| = \Delta_c$. Instead, this μ_c must be calculated by an iterative process with each iteration requiring the Cholesky factorization of a matrix of the form $H_c + \mu I$. In contrast, the line search methods of Section 4.2 require only one matrix factorization per iteration. Furthermore, it is possible that the step (4.3.3) will be required which necessitates an eigenvalue-eigenvector calculation. This case is rare, especially in finite precision arithmetic, but in cases that are close to this the calculation of (4.3.2) becomes more difficult.

For these reasons, efficient computational implementations of trust region methods solve (4.3.1) approximately. Before we present these approximate solution methods, we discuss the overall schema of a trust region method including the adjustment of the trust radius Δ_c, and the convergence properties of a method that uses this schema while solving (4.3.1) exactly. This theorem will help justify our continued interest in these methods. We will then see that these convergence properties are retained when using various efficient, approximate trust region steps in place of the exact solution to (4.3.1).

After the trust region s_c is calculated, a trust region method must evaluate $f(x_c + s_c)$ to see whether $x_c + s_c$ is a satisfactory next iterate. It may not be if the quadratic model does not accurately reflect $f(x)$ within the trust region. In this case the trust radius is decreased and the trust region step recalculated. Otherwise, $x_c + s_c$ becomes the next iterate and the new trust radius must be calculated. Such an approach is outlined in Algorithm 4.3.1.

The reader will recognize the step acceptance condition

$$actual\text{-}reduction \leq \alpha_1 \, (predicted\text{-}reduction) \tag{4.3.4}$$

as being very similar to the sufficient decrease condition (4.2.1) used in line search algorithms. The only difference is that the second order term $\frac{1}{2} s_c^T H_c s_c$ is included on the right hand side of (4.3.4). Again, $\alpha_1 = 10^{-4}$ is typical. No analog to condition (4.2.2) is needed by trust region methods because the strategy for adjusting the trust radius prevents the step from being too short.

If (4.3.4) is failed, the current iteration is repeated with a smaller trust radius. The procedure for decreasing Δ_c is similar or identical to the procedure Init-Decrease for decreasing t_c in a line search.

Algorithm 4.3.1 Trust Region Iteration

Given $f : \mathbf{R}^n \rightarrow \mathbf{R}$, x_c, $f(x_c)$, $\nabla f(x_c)$, Hessian (approximation) H_c,
trust radius $\Delta_c > 0$, $0 < \alpha_1 < \alpha_2 < 1$, $0 < c_1 < c_2 < 1 < c_3 < c_4$

done := false
Repeat
s_c := exact or approximate solution to (4.3.1)

evaluate $f(x_c + s_c)$
$actual\text{-}reduction := f(x_c + s_c) - f(x_c)$
$predicted\text{-}reduction := \nabla f(x_c)^T s_c + \frac{1}{2} s_c^T H_c s_c$

If $actual\text{-}reduction \leq \alpha_1 \, predicted\text{-}reduction$ Then
done := true
$x_+ := x_c + s_c$
If $actual\text{-}reduction \leq \alpha_2 \, predicted\text{-}reduction$ Then
Δ_+ := Increase-Δ (Δ_c) (* $\Delta_+ mem [c_3, c_4] \Delta_c$ *)
Else $\Delta_+ := \Delta_c$ (* or $\Delta_+ mem [c_1, 1] \Delta_c$ *)
Else Δ_c := Decrease-Δ (Δ_c) (* new Δ_c $mem [c_1, c_2]$ old Δ_c *)
Until *done* = true

Typically, quadratic interpolation (equation (4.2.8)) with a safeguard such as new $\Delta_c \in [0.1, 0.5]$ old Δ_c is used.

If (4.3.4) is satisfied, $x_c + s_c$ is acceptable as the next iterate x_+ and the new trust radius Δ_+ must be determined. Algorithm 4.3.1 increases Δ_+ over Δ_c if the quadratic model and function have agreed quite well; the condition $actual-reduction \leq \alpha_2$ $(predict-reduction)$ tests this with α_2=0.75 typical. Some methods instead allow the *current* iteration to be continued with this larger trust radius in this case, which complicates the description considerably and doesn't affect the theoretical properties. This may help in practice by saving gradient evaluations. In either case, the Increase-Δ procedure usually doubles Δ_c, analogous to the Init-Increase portion of the line search. Otherwise, $\Delta_+=\Delta_c$ in Algorithm 4.3.1. Some methods may set the new $\Delta_+ < \Delta_c$ if agreement between the actual function and the model was rather poor, for example if (4.3.4) was failed with α=0.1. This also does not affect the method's convergence properties.

Theorem 4.3.2 gives the main global and local convergence properties of a method based on the trust region iteration of Algorithm 4.3.1, in the case where (4.3.1) is solved exactly. Similar results may be found in many references including Fletcher [1980], Gay [1981], and Sorensen [1982].

Theorem 4.3.2 Let $f : \mathbf{R}^n \rightarrow \mathbf{R}$ be twice continuously differentiable and bounded below. Also, for $x_0 \in \mathbf{R}^n$ and some $\beta_1, \beta_2 > 0$, let $\nabla^2 f(x)$ be uniformly continuous and satisfy $\|\nabla^2 f(x)\| \leq \beta_1$ for all $x \in \{x \in \mathbf{R}^n : f(x) \leq f(x_0)\}$. Let $\{x_k\}$ be the sequence produced by iterating Algorithm 4.3.1 starting from x_0, and using $H_c = \nabla^2 f(x_c)$ or any symmetric approximation with $\|H_c\| \leq \beta_2$ at each iteration, and the exact solution to (4.3.1) to calculate d_c. Then $\lim_{k \rightarrow \infty} \|\nabla f(x_k)\| = 0$. If in addition each $H_c = \nabla^2 f(x_c)$, then for any limit point x_* of the sequence $\{x_k\}$, $\nabla f(x_*) = 0$ and $\nabla^2 f(x_*)$ is at least positive semi-definite. Furthermore if each $H_c = \nabla^2 f(x_c)$,then if $\{x_k\}$ converges to x_*, $\nabla^2 f(x_*)$ is positive definite, and $\nabla^2 f(x)$ is Lipschitz continuous around x_*, then the rate of convergence is q-quadratic.

Theorem 4.3.2 shows that a trust region method that solves the trust region problem exactly has attractive convergence properties. The same first order result is established as for line search methods, and no assumption about the condition number of H_c is needed. In addition, if exact second derivatives are used, then the second order necessary conditions for a minimum are satisfied by any limit point, which means that saddle points and maxima are avoided. The analysis also shows that near a local minimum with

$\nabla^2 f(x_*)$ positive definite, asymptotically the trust region constraint becomes inactive so that only Newton steps are taken and local quadratic convergence is retained.

These nice convergence properties help motivate the interest in trust region methods which follow the general schema of Algorithm 4.3.1, but where the step is calculated by a considerably more efficient procedure than that required for the exact solution of problem (4.3.1). There are two obvious relaxations to problem (4.3.1) that may make it easier to solve : the trust region constraint may be satisfied only approximately, and/or the quadratic model may be minimized only approximately. We will see that both may be relaxed substantially without weakening the theoretical properties of the method.

In fact, two general classes of efficient methods for approximately solving (4.3.1) have arisen, corresponding to these two possible relaxations of (4.3.1). In the first, *approximate optimal step methods*, mainly the trust region constraint is relaxed; the step generally is still of the form $-(H_c + \mu I)^{-1}\nabla f(x_c)$ for some positive definite $H_c + \mu I$. These methods are summarized below. In the second, *dogleg methods*, the minimization of the quadratic model is relaxed; we defer the consideration of these methods until later in this section.

Hebden [1973] and More [1978] were the first to construct efficient approximate optimal step methods. They developed an efficient procedure for finding a $\mu > 0$ for which

$$\|(H_c + \mu I)^{-1}\nabla f(x_c)\| \approx \Delta_c \tag{4.3.5}$$

in the case when H_c is positive definite and $\|H_c^{-1}\nabla f(x_c)\| > \Delta_c$. Their algorithms are based on applying Newton's method in μ to

$$\frac{1}{\|(H_c + \mu I)^{-1}\nabla f(x_c)\|} - \frac{1}{\Delta_c} = 0 \ , \tag{4.3.6}$$

following the observation that (4.3.6) is more nearly linear in μ near the desired solution than is (4.3.5). This results in the μ-iteration

$$\mu_+ = \mu - \frac{\|s_c\|^2 (\|s_c\| - \Delta_c)}{\Delta_c \, (s_c^T (H_c + \mu I)^{-1} s_c)} \tag{4.3.7}$$

where $s_c = -(H_c + \mu I)^{-1}\nabla f(x_c)$, which requires one factorization of a matrix of the form $H_c + \mu I$ for each μ-iteration. Typically the trust region constraint is relaxed to $\|s_c\| \in [0.9, 1.1]\,\Delta_c$; then usually only one or two iterations of (4.3.7), and the same number of matrix factorizations, are required for each iteration of the optimization algorithm.

Several authors, including Gay [1981], Sorensen [1982], and More and Sorensen [1983], have investigated generalizations of these approximate optimal step methods that extend to the case when H_c is indefinite. More and Sorensen present an efficient algorithm that guarantees that their approximate solution to (4.3.1) reduces the quadratic model $m_c(x_c + s)$ by at least γ times the amount that the exact solution to (4.3.1) would, for any fixed $\gamma < 1$. Their algorithm combines the Hebden-More procedure mentioned above with a use of the LINPACK condition number estimator to obtain a satisfactory solution when the exact solution is in or near the "hard case" (4.3.3). No eigenvalue/eigenvector calculations are required. They show that their method still generally requires only 1-2 matrix factorizations per iteration, and that it retains the convergence properties of Theorem 4.3.2.

The analyses of these approximate optimal step methods are subsumed by Theorem 4.3.3 below, due to Shultz, Schnabel, and Byrd [1985]. Similar first order results are proven by Powell [1970, 1975] and Thomas [1975]. Details of the interpretations that are given following Theorem 4.3.3 are also found in Shultz, Schnabel, and Byrd [1985].

Theorem 4.3.3 Let the assumptions in the first two sentences of Theorem 4.3.2 hold. Let $\{x_k\}$ be the sequence produced by iterating Algorithm 4.3.1 starting from x_0, using $H_c = \nabla^2 f(x_c)$ or any symmetric approximation with $\|H_c\| \leq \beta_2$ at each iteration. If there exist $c_1, c_2 > 0$ such that each s_c satisfies,

$$\nabla f(x_c)^T s_c + \frac{1}{2} s_c^T H_c s_c \leq -c_1 \|\nabla f(x_c)\| \min\{\Delta_c, \frac{c_2 \|\nabla f(x_c)\|}{\|H_c\|}\}, \tag{4.3.8}$$

then $\lim_{k \to \infty} \|\nabla f(x_k)\| = 0$. If in addition each $H_c = \nabla^2 f(x_c)$ and there exists $c_3 > 0$ such that each d_c satisfies

$$\nabla f(x_c)^T s_c + \frac{1}{2} s_c^T H_c s_c \leq -c_3(-\lambda_1(H_c))\Delta_c^2 \tag{4.3.9}$$

where $\lambda_1(H_c)$ denotes the smallest eigenvalue of H_c, then for any limit point x_* of $\{x_k\}$, $\nabla f(x_*) = 0$ and $\nabla^2 f(x_*)$ is at least positive semi-definite. Also, if each $H_c = \nabla^2 f(x_c)$, each s_c satisfies (4.3.8), and there exists $c_4 \in (0, 1]$ such that $s_c = -H_c^{-1} \nabla f(x_c)$ whenever H_c is positive definite and $\|H_c^{-1} \nabla f(x_c)\| \leq c_4 \Delta_c$, then if $\{x_k\}$ converges to x_* with $\nabla^2 f(x_*)$ positive definite and $\nabla^2 f(x)$ Lipschitz continuous around x_*, then the rate of convergence is q-quadratic.

Theorem 4.3.3 contains two equations, (4.3.8) and (4.3.9), that say what conditions a trust region step needs to satisfy in order to be globally convergent to points satisfying the first and second order necessary

conditions for minimization, respectively. Equation (4.3.8) gives a condition on the sufficient use of descent directions in order to assure first order global convergence. With $c_1=\frac{1}{2}$ and $c_2=1$, it implies that the step s_c provides at least as much decrease in the quadratic model as the best permissible step in the steepest descent direction. (Here "permissible" means "obeying the trust region constraint"). This interpretation forms part of the motivation for the dogleg methods for approximating (4.3.1) that are discussed below. By using smaller c_1 and c_2 in (4.3.8), condition (4.3.8) says that each s_c provides at least a fixed fraction of the decrease in the quadratic model that would be obtained by the best permissible step in some direction p_c of "bounded" descent (i.e. the angle between p_c and $\nabla f(x_c)$ is uniformly bounded above by a constraint < 90°). This is all that is needed to assure $\lim_{k\to\infty}\{\nabla f(x_k)\}=0$. Thus Theorem 4.3.3 also applies to a line search method where the trust region bound Δ_c is used to determine the steplength.

Equation (4.3.9) gives a condition on the sufficient use of negative curvature directions that, in conjunction with (4.3.8), assures global convergence to a point satisfying second order necessary conditions for minimization. Equation (4.3.9) can be interpreted as saying that, whenever H_c is indefinite, s_c provides at least a fixed fraction of the reduction in the quadratic model that would be obtained by the best permissible step in some direction v_c of "bounded" negative curvature (i.e. $(v_c^T H_c v_c)/(\lambda_1 v_c^T v_c)$ is uniformly bounded below by a fixed positive constant). This constitutes a considerable relaxation of the "hard case" (4.3.3) in the exact solution of (4.3.1).

Thus any trust region method that uses the schema of Algorithm 4.3.1 and chooses its steps to satisfy conditions (4.3.8-9) has strong global convergence properties. It is straightforward to show that (4.3.8-9) implies the condition on a trust region step used by More and Sorensen [1983], that the reduction in the quadratic model by s_c be at least a fixed fraction of the reduction from solving (4.3.1) exactly. The converse is only true under stronger assumptions on H_c (see Byrd, Schnabel, and Shultz [1987]). Now we return to the second class of methods for approximately solving (4.3.1), dogleg methods, which are now easily seen as another way to satisfy conditions (4.3.8-9).

The earliest trust region method, of Powell [1970], is the original dogleg method. Given a quadratic model $m_c(x_c+s)$ with H_c positive definite, and $\Delta_c > 0$, it selects $x_c + s_c$ to be the Newton point $x_N = x_c - H_c^{-1}\nabla f(x_c)$ if it is inside the trust region. Otherwise it selects $x_c + s_c$ to be the point on the piecewise

linear curve connecting x_c, x_{Cp}, and x_N which is distance Δ_c from x_c. (See Fig. 4.3.2.) Here x_{Cp} is the *Cauchy point*, the minimum of the quadratic model in the steepest descent direction $-\nabla f(x_c)$ from x_c. (It is straightforward to show that $\|x_{Cp} - x_c\| \leq \|x_N - x_c\|$, so that the intersection of the dogleg curve with the trust radius is unique.) Dennis and Mei [1979] constructed a generalization, the *double dogleg method*, which selects $x_c + s_c$ to be the unique point on the piecewise linear curve connecting x_c, x_{Cp}, γx_N, and x_N which is distance Δ_c from x_c , or $x_c + s_c = x_N$ if x_N is inside the trust region, for a particular $\gamma < 1$. For either method, it can be shown that as one moves along the piecewise linear curve from x_c to x_N, the distance from x_c increases and the value of the quadratic model decreases.

Thus these dogleg methods take small steps in the steepest descent direction when the trust radius is small, take Newton steps when the trust radius is sufficiently large, and take steps in a linear combination of the steepest descent and Newton directions otherwise. Due to the use of steepest descent steps when $\Delta_c \leq \|x_{Cp} - x_c\|$ and to the monotonic decrease of the quadratic model along the entire dogleg curve, they always obtain at least as much descent on the quadratic model as the best steepest descent step of length at most Δ_c. Thus they obey (4.3.8) and by Theorem 4.3.3 are globally convergent in the sense that the sequence of gradients of the iterates converges to zero.

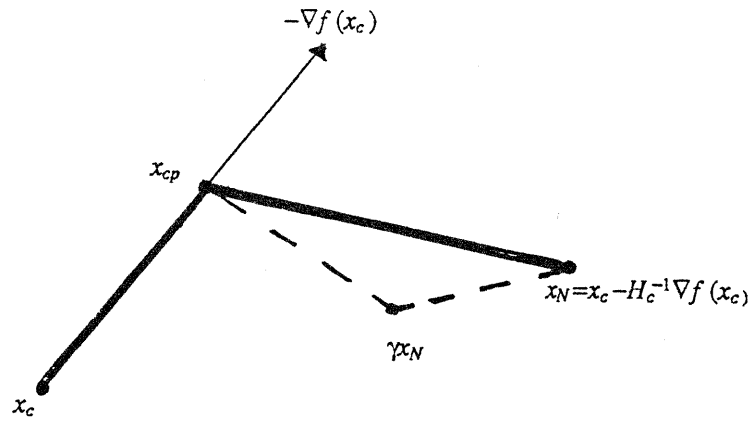


Figure 4.3.2 The dogleg curve
(dotted part is double dogleg modification)

An attraction of these dogleg methods, in comparison to the approximate optimal step methods discussed above, is that they only require one matrix factorization (of H_c) per iteration. All the remaining calculations are easily seen to require at most $O(n^2)$ operations, and no additional factorizations are required if the trust region must be decreased during the current iteration. On the other hand, neither version described above makes explicit use of an indefinite H_c or satisfies (4.3.9). Thus no second order global convergence results can be proven for them.

These dogleg methods are closely related to minimizing the quadratic model $m_c(x_c + s_c)$ over the two dimensional subspace spanned by the steepest descent and Newton directions, subject to the trust region constraint. This two dimensional trust region problem also is easy to solve using only the factorization of H_c, and the inclusion of the steepest descent direction assures satisfaction of (4.3.8) and hence first order global convergence. Shultz, Schnabel, and Byrd [1985] propose an algorithm along these lines where, if H_c is indefinite, the two dimensional subspace is changed to the one spanned by $-\nabla f(x_c)$ and some direction of bounded negative curvature which is fairly efficient to compute. Thus the algorithm obeys (4.3.9) as well and has the same theoretical convergence properties as an approximate optimal step method. Byrd, Schnabel, and Shultz show that an optimization algorithm based on this approach is very competitive with a modern approximate optimal step method in robustness and efficiency.

In practice both approximate optimal step methods and various dogleg methods are used in solving unconstrained optimization problems and in other contexts. Some additional comments on their relative merits are contained in Section 4.4.

4.4 Comparison of Line Search and Trust Region Methods

Sections 4.2 and 4.3 have presented two classes of methods, line searches and trust regions, for obtaining globally convergent unconstrained optimization methods while also retaining fast local convergence. The reasons for presenting both approaches are that neither appears to be consistently superior to the other in practice, that both are used in modern software, and that both can be expected to play important roles in the future development of optimization methods. This section elaborates briefly on these remarks.

Gay [1983] and Schnabel, Koontz, and Weiss [1985] have conducted comparisons of line search and trust region codes on a standard set of test problems from More, Garbow, and Hillstrom [1981]. Gay compared a BFGS method utilizing a line search with a BFGS method using a double-dogleg trust region step. Schnabel, Koontz, and Weiss tested methods using finite difference Hessians and methods using BFGS approximations, in both cases comparing line search, double-dogleg, and approximate optimal trust region steps. Both studies showed that while there can be considerable differences between the performance of line search, dogleg, and optimal step methods on individual problems, no one method is consistently more efficient or robust than any other. Indeed, the average differences in efficiency between the line search and trust region methods tested were quite small, and they had similar success rates.

In modern numerical software libraries, one finds both line searches and trust regions used in conjunction with both (finite difference) Hessians or BFGS approximations. Philosophically, some people prefer to use line searches in conjunction with BFGS methods because the necessary condition (3.4.3) for positive definite updates can be guaranteed to be satisfied at each iteration; in trust region methods no such guarantee is possible and occasionally (3.4.3) is not satisfied and the update must be skipped. Similarly, some people advocate using trust region methods in codes where the (finite difference) Hessian is used, because a "natural" treatment of indefiniteness is possible and it can be guaranteed that saddle points and local maxima are avoided. But as the previous paragraph has indicated, neither of these theoretical arguments has been shown to correspond to any significant computational advantage.

Algorithm developers in areas such as constrained optimization, least squares data fitting, and large scale optimization often need to choose between using line searches or trust regions in developing new codes. Some trade-offs are fairly consistent. Line searches often are a little simpler to code, but sometimes it is not clear how to deal with indefiniteness, rank deficiency, or other factors that may cause the line search direction to be in an unacceptable direction. Trust region methods often offer a mathematical solution to these problems, but usually require some additional linear algebra cost. In addition it sometimes is challenging to construct efficient, approximate solution algorithms for the appropriate trust region problem. The result is that both approaches are used; for example there currently is considerable research activity in both line search and trust region methods for nonlinearly constrained optimization. The two-dimensional trust region technique mentioned at the end of Section 4.3 seems to offer a good compromise in cases

where the trust region approach seems desirable to assure acceptable step directions, but an (approximate) optimal solution to the exact trust region problem is very difficult or expensive to find.

One case where there appears to be a discernible difference between line search and trust region methods is in Gauss-Newton methods for nonlinear least squares (see Section 6.2). In this case the underlying local method is at best linearly convergent on most problems. For such algorithms, trust region algorithms, which may be viewed as combining two linearly convergent directions, the standard Gauss-Newton direction and the steepest descent direction, appear generally to be more robust and efficient than line search algorithms which only use the Gauss-Newton direction. For a detailed discussion of such algorithms, see Dennis and Schnabel [1983], Ch. 10.

5. Non-Taylor Series Methods

This section presents two fundamentally different algorithmic approaches that have proven themselves useful for unconstrained minimization. First, we will describe the Nelder-Mead simplex algorithm, Nelder and Mead [1965], an effective pattern search technique for problems of very low dimension which is beloved of users but generally ignored by optimization researchers. Then, we will provide a unifying framework for the proliferation of conjugate direction algorithms that have been devised for solving problems with large numbers of variables.

5.1 The Nelder-Mead Simplex Algorithm

The Nelder-Mead algorithm moves a simplex through the domain space with the goal of getting the function minimizer x_* in the interior of the simplex. Once *ad hoc* tests indicate that the minimizer has been surrounded, the algorithm shrinks the simplex toward the vertex corresponding to the lowest function value and returns to the process of trying to get x_* into the interior of the (smaller) simplex.

We will confine ourselves here to a description of the four basic moves of the algorithm. Each iteration begins with a set of $n+1$ current points $x_c^1,...,x_c^{n+1}$ in general position, i.e., the convex hull S_c of $\{x_c^1,...,x_c^{n+1}\}$ is an n-dimensional simplex. Furthermore, these vertices are assumed to be sorted on their objective function values so that $f(x_c^i) \le f(x_c^{i+1})$, $i = 1,...,n$. The first goal of each iteration is to replace the worst vertex x_c^{n+1} with a better one by moving the simplex away from x_c^{n+1}. If this fails, we tacitly assume that we are close enough to the minimizer to need smaller moves for improvement. Thus, we keep the best vertex and shrink the simplex along each edge by replacing each of the other vertices by its average with the old best vertex. We then evaluate f at the n new vertices and sort and label them to obtain $S_+ = \langle x_+^1,...,x_+^{n+1} \rangle$. The convention we use is that the older vertex is numbered lower when two vertices have equal function values.

We have described the *shrinkage step* through which the algorithm tries to close in on x_*; now we describe the moves aimed at getting the larger simplex close by moving away from the worst vertex.

The first trial step in each iteration is to consider the *reflection* $x_c^r = 2\bar{x}_c - x_c^{n+1}$ of x_c^{n+1} through the centroid $\bar{x}_c = \frac{1}{n}\sum_{i=1}^{n} x_c^i$ of the best n-face of S_c. If success is so great that $f(x_c^r) < f(x_c^1)$, then we try

expanding the simplex in the same direction by testing $f(x_c^e) < f(x_c^1)$, where $x_c^e = 2x_c^r - x_c^{n+1}$. If the expansion is successful, we drop x_c^{n+1} in favor of x_c^e, otherwise we drop x_c^{n+1} in favor of x_c^r, and sort the vertices to obtain S_+. There are two things to note about a successful *expansion* step. The first is that we do not continue in the same vein even if $f(x_c^e) < f(x_c^r) < f(x_c^1)$. This is because we want to avoid a simplex that gets elongated enough that its vertices are not numerically in general position. The second point is that we accept the expansion vertex even if it is not as good as the reflection vertex. Thus, the best approximate minimizer we have found so far might not be retained as a vertex. This is in keeping with our view of trying to move the simplex over the minimizer before we start to close in on it.

The reflection vertex is taken without an expansion attempt if $f(x_c^1) \le f(x_c^r) < f(x_c^n)$. In this case, x_c^n will become x_+^{n+1} after sorting. If $f(x_c^r) \ge f(x_c^{n+1})$, then we try once to contract the simplex internally along the reflection direction by testing $f(x_c^c) < f(x_c^n)$, where the contraction vertex is $x_c^c = \frac{1}{2}[\bar{x}_c + x_c^{n+1}]$. If the *contraction* fails, then we shrink the simplex as explained above. If it succeeds, then we replace x_c^{n+1} by x_c^c and sort to prepare for the next iteration.

There is one more possibility to consider for the trial of a reflection step. If $f(x_c^n) \le f(x_c^r) < f(x_c^{n+1})$, then we can see immediately that if we replace x_c^{n+1} by x_c^r, the outcome would be that x_+^{n+1} would be x_c^r and that the subsequent reflection step would be rejected, because $x_+^1 = x_c^{n+1}$, in favor of a trial contraction. Thus, we pass over this 'shadow' iteration and compute the indicated shadow contraction directly from the current vertices as $x_c^{s'} = \frac{1}{2}(x_c^r + \bar{x}_c)$. We finish the iteration exactly as we did for a regular contraction.

Many users have suggested minor modifications of this algorithm. The best known modifications and the history of the algorithm are collected in Woods [1985]. Woods also gives a novel application of one of the three major advantages of the algorithm. He applies the algorithm to multicriteria optimization by exploiting the fact that we only use the objective function f to decide which is the 'better' of two vertices. In common use, this indicates that the algorithm should be robust with respect to noise or inaccuracies in the values of f. For example, we have experience with an engineer who was able to use it to resolve parameters to .5% after only reaching 5% resolution with a standard finite-difference Newton code.

The three main strengths of the Nelder-Mead simplex method are its tolerance of function noise, its nonreliance on any gradient approximations, and the extreme simplicity of its implementation. It takes less

than 100 statements to implement in any high level language, and this is an important factor in its popularity with users who still distrust black boxes.

The major weaknesses of the algorithm are that it can be very slow for more than about 5 variables, and that it can converge to a nonminimizer. The only example of the latter that we know of is given in Dennis and Woods [1987] and it is mitigated somewhat by the nonminimizing limit being a point where the derivative doesn't exist. Both these shortcomings are the subject of current research.

It is worth noting that the example referred to in the paragraph above shows that the algorithm is not safe for nonsmooth problems despite the fact that it uses no gradient information. Incidental to this *caveat* is that the example is convex.

5.2 Conjugate Direction Methods

The second class of methods that we discuss in this section is the conjugate direction algorithms. It is not entirely accurate to say that they are not based on quadratic models, but there is never a need to store a full Hessian. Therefore, these methods are especially suited for large dimensional problems where $f(x)$ and $g(x)$ are available but n is too large to store or factor an $n \times n$ matrix at each iteration.

Conjugate direction algorithms are usually presented as they would be programmed. This demonstrates their most important property, computational simplicity and little storage, but it obscures the geometric elegance that better helps the novice gain an overview of the whole class of methods.

Conjugate direction methods are most simply presented as methods for minimizing strictly convex quadratic functions. We will follow the point of view taken in Dennis and Turner [1986] where the reader can find all the proofs of results claimed here. We will adopt the standard convenience of taking $x_0 = 0$.

Assume that we are at the k^{th} iterate and that we have a scheme for generating a descent direction d_{k+1} for $q(x) = \frac{1}{2} x^T H x - h^T x$, H symmetric and positive definite, from x_k. Suppose that x_k minimizes $q(x)$ on a k-dimensional subspace spanned by the previous iterates. Choose x_{k+1} to be the unique minimizer of $q(x)$ on the $k+1$-dimensional subspace formed by adding d_{k+1} to the previous spanning set. These two sentences characterize the conjugate direction algorithms. It is interesting to note that this point of view is identical to the definition given by Cantrell [1969], Cragg and Levy [1969], and Miele and Cantrell

[1969] of a 'memory gradient' method. It is completely developed in Dennis and Turner [1986]. Nazareth [1986] gives a corresponding algorithm for general minimization problems.

It is easy to show that if we define at each iteration $p_j = x_j - x_{j-1}$, then $p_j^T H p_i = 0$ for $1 \le j < i \le k$ and $p_j^T \nabla q(x_k) = 0$ for $1 \le j \le k$. Thus, solving the $k+1$ dimensional minimization problem on span $\{p_1, ..., p_k, d_{k+1}\}$ for x_{k+1} requires the solution of a $(k+1)$-dimensional symmetric positive definite linear system which is diagonal except for a full last row and column. This system can be solved explicitly, but the expense of saving all the previous p_i's is too high for large problems.

If one can arrange to have $d_k^T H p_i = 0$ for the oldest p_i's, then those p_i do not have to be saved because they are not needed explicitly in solving for x_{k+1}. It turns out that $d_{k+1} = -\nabla q(x_k) = h - Hx_k$ accomplishes this purpose for $1 \le i \le k-1$ so that p_{k+1} is a linear combination of p_k and d_{k+1}. This fortunate circumstance follows from a more general result.

Let B be an arbitrary matrix and select each $d_{j+1} \in span\{d_1, Bp_1, ..., Bp_j\}$. Then, it is easy to show that x_k minimizes $q(x)$ on span $\{d_1, Bd_1, ..., B^{k-1}d_1\} \equiv K(d_1, B, k-1)$, and that $-\nabla q(x_k)^T B p_j = 0$ for $j = 1, ..., k-1$. Thus, if we choose any B and generate d_{k+1} so that $d_{k+1}^T H = -\nabla q(x_k)^T B$, then we only need x_k, p_k and d_{k+1} to generate

$$p_{k+1} = d_{k+1} + \beta_k \cdot p_k, \ \beta_k = -\frac{d_{k+1}^T H p_k}{p_k^T H p_k} \tag{5.2.1}$$

and

$$x_{k+1} = x_k + \alpha_k \cdot p_{k+1}, \ \alpha_k = \frac{d_{k+1}^T \nabla q(x_k)}{d_{k+1}^T H p_{k+1}}. \tag{5.2.2}$$

In this case, the p_k's are to be thought of as directions rather than iterative steps as they were defined above, but there is no difficulty introduced into the discussion above by doing so. Of course $\pm d_{k+1}$ must also be a descent direction for q from x_k, i.e., $d_{k+1}^T \nabla q(x_k) \neq 0$. The choice $B = H$ gives rise to $d_{k+1} = -\nabla q(x_k)$ mentioned above. This is called the *conjugate gradient* algorithm. The subspace $K(d_1, B, k-1)$ is called the k-dimensional Krylov subspace generated from d_1 by B.

If M is any symmetric positive definite matrix for which $d_{k+1} = -M^{-1}\nabla q(x_k)$ is easy to compute, then d_{k+1} is a descent direction and corresponds to $B = M^{-1}H$. The resultant method is called the *preconditioned conjugate gradient algorithm* and M is called the *preconditioner*. The use of a preconditioner can significantly improve the conjugate gradient algorithm. Let us now discuss briefly some factors in

choosing M.

Intuitively, there are two things we might want to accomplish in our choice of M and hence B. We can try for a big reduction in q by choosing M^{-1} to be a good approximation to H^{-1}. In other words, we can try to make our new directions d_k approximate the Newton direction for q. This is the point of most iterative methods, like SOR or Gauss-Seidel for example, and it is common to exploit such methods as preconditioners for problems where they were once used alone. In such preconditioners, one never needs to work explicitly with M since $M^{-1}\nabla q(x_k)$ is the iterative step. It is worth noting that SOR does not correspond to a symmetric positive definite preconditioner M and so SSOR, which involves a forward and then backward sweep, is generally used because it does correspond to a symmetric positive definite M. See Golub and Van Loan [1983].

This way of choosing an iterative method as a preconditioner for a conjugate direction algorithm lends itself to two popular points of view: an optimizer would feel that the iterative method is being used to accelerate the conjugate direction algorithm, but the numerical partial differential equations solver would be more likely to feel that conjugate directions is being used to accelerate the basic iterative method being used as a preconditioner.

From a purely matrix algebra point of view, this first way of choosing M, which we have been discussing, corresponds to making the condition number of $M^{-\frac{1}{2}}HM^{-\frac{1}{2}}$ small, since this is the Hessian and $M^{-1}\nabla q(x_k)$ is the steepest descent direction for q at x_k in the transformed variables $x' = M^{\frac{1}{2}}x$. It is worth pointing out the result that the DFP or BFGS applied to minimize $q(x)$ with exact line searches and initial Hessian approximation M generates exactly the same points in exact arithmetic as the conjugate gradient algorithm preconditioned by M.

The second point of view in choosing M, and hence $M^{-1}H = B$, is to try to make $K(M^{-1}h, B, n-1)$ have the smallest possible dimension, say $p \ll n$. The reason is that in this case the algorithm solves the problem in p steps. This can be seen from the fact that $\nabla q(x_p)$ is orthogonal to $K(M^{-1}h, B, p-1)$. Thus, $M^{-1}\nabla q(x_p)$ cannot lie in $K(M^{-1}h, B, p-1)$ unless it is zero. Then it must be zero since $M^{-1}\nabla q(x_p) \in K(M^{-1}h, B, p)$ and the Krylov subspaces have stopped increasing their dimension so $K(M^{-1}h, B, p) = K(M^{-1}h, B, p-1)$.

If we ignore the influence of the initial direction on the dimension of $K(M^{-1}h, B, n-1)$, then an upper bound on that dimension is the degree of the minimal polynomial of B. This is easy to see since if $m(B)=\sum_{i=0}^{p}\alpha_i B^i=0$ is the minimal polynomial for B, then $0=m(B)M^{-1}h$ is a nonzero linear combination of the $p+1$ generators for $K(M^{-1}h, B, p)$ whose dimension must be less than $p+1$. Since B is similar to a symmetric matrix, p is the number of distinct eigenvalues of $B=M^{-1}H$. See Hoffman and Kunze [1971].

It is not unusual for strictly convex quadratics arising from discretized partial differential equations to be solved with $p \sim n/10^3$. Such spectacularly successful preconditionings nearly always come from deep insight into the problem and not from matrix theoretic considerations. They often come from discretizing and solving a simplified problem.

The choice of preconditioners for optimization problems is not nearly so well understood. This may be because most of our effort has been directed toward algorithms and software for general library use. We have generally used conjugate direction methods only for nonquadratic problems, and then only for problems so large that we have no other choice (see Section 6.1).

For nonquadratics, the conjugate gradient formulas are generally given by

$$p_{k+1} = -\nabla f(x_k) + \beta_k \cdot p_k, \quad \beta_k = \frac{\nabla f(xk)^T \nabla f(x_k)}{\nabla f(x_{k-1})^T \nabla f(x_{k-1})} \tag{5.2.3}$$

and

$$x_{k+1} = x_k + \alpha_k \cdot p_{k+1}, \ \alpha_k \text{ minimizes } f \text{ along } p_{k+1}. \tag{5.2.4}$$

If $f(x)$ is quadratic, (5.2.3-4) are equivalent to (5.2.1-2), but no matrix is required by (5.2.3-4). In general, the line search is not done exactly but it has to be done fairly accurately; see Gill, Murray, and Wright [1981].

Of course, the conjugate directions do not remain conjugate in finite precision implementations, and there is no reason they should be conjugate for nonquadratics. The standard way to handle this is to save some past vectors and make sure d_k is made conjugate with respect to these few vectors (see Vinsome [1976], Young and Jea [1981]), or to restart the method, perhaps by taking d_k periodically to be a linear combination of some restart vector and the d_k that would have been chosen if a restart were not due. See Beale [1972] or Powell [1977].

6. Some Current Research Directions

In this section we will give brief summaries of some interesting areas of ongoing activity. We will discuss large problems, data fitting, secant updates, singular problems, and parallel computation.

6.1 Large Problems

There are three different scenarios we wish to consider here:

(i) A quadratic model can be formed and the model Hessian can be factored if sparsity is taken into account.

(ii) A quadratic model can be formed but linear iterative methods must be used to solve the model problem in place of matrix factorizations.

(iii) The problem is too large for any explicit use of a model Hessian.

It is important to note that the number of variables alone does not determine which class fits a particular problem. If a big problem has a nice enough sparsity structure in the Hessian it fits in class (i); if the sparsity structure is not so nice it fits in (ii); and if it isn't sparse enough for (ii) it fits in (iii).

For problems in class (i), our first choice would be to use a Newton or finite difference Newton model as outlined in Section 3.5. If we wish to use a secant method, then we should use a so-called "limited memory" method in the spirit of the last implementation in Section 3.3. Probably, these methods fit better with a linesearch rather than a trust region. Buckley and Lenir [1983] is a limited memory method which has been highly recommended to us by users.

For problems of class (ii), there is an elegant generalization of the dogleg algorithm (Steihaug [1980], Toint [1981]) which has shown its mettle in dealing with seismic inversion. This algorithm can be viewed as a trust region implementation of the conjugate direction inexact Newton method. The idea is simple. Given a quadratic model and a trust radius, perform conjugate direction iterations to compute the Newton step until either the Newton step is computed inside the trust region and taken as the trial step, or until some conjugate direction iterate lands outside the trust region. When the latter happens, the trial step is taken to be at the intersection of the trust region and the last conjugate direction step, or a direction of

negative curvature for the quadratic model is generated at some conjugate direction iterate inside the trust region, and the trial step is taken where this negative curvature direction strikes the boundary of the trust region. If a preconditioner is used, it can be thought of as defining the shape of the trust region.

As computer storage has become less expensive, fewer problems must be relegated to class (iii). Generally, these problems are solved using the non-matrix form of conjugate direction methods mentioned at the end of Section 5.2.

Finally, Griewank and Toint [1982a,b] have suggested and analyzed an interesting approach to obtaining Hessian approximations for problems where the objective function can be written as the sum of objective functions that each involves its own subset of the variables. Generally speaking, the summand functions should have small dense Hessians which are positive semidefinite at the point formed by selecting the relevant subset of the components of the minimizer of the problem. This allows an approximate Hessian of the sum to be assembled from (for example) BFGS updates of the summand Hessians. The reader familiar with research on sparse matrices will recognize the connection with so-called clique or finite-element storage. See Duff, Erisman, and Reid [1986].

6.2 Data Fitting

One of the most common sources of optimization problems is in parameter estimation arising from fitting mathematical models to data. Typically, data

$$(t_i, x_i)\ ,\ i=1,\cdots,m \tag{6.2.1}$$

has been collected, and one wants to select the free parameters $x \in \mathbb{R}^n$ in the model $m(t,x)$ so that

$$m(t_i, x) \cong y_i,\ i=1,...,m\ . \tag{6.2.2}$$

Such problems arise in almost all areas of science, engineering, and social science. A more common notation in these applications is (x_i, y_i), $i=1,...,n$ for the data, and $f(x,\theta)$, $\theta \in \mathbb{R}^p$ for the model (θ may be replaced by β or other symbols), but we will use (6.2.1-2) to be consistent with the remainder of this book. Generally, there are far more data points than parameters, that is $m \gg n$ in (6.2.2).

Usually it is assumed that each t_i is known exactly in (6.2.1) but that y_i is measured with some error. In that case it makes sense to achieve (6.2.2) by making each *residual*

$$r_i(x) = m(t_i, x) - y_i$$

as small as possible. Let

$$R(x) = (r_1(x), \ldots, r_m(x))^T .$$

Then we wish to choose x so that the vector $R(x)$ is as small as possible, in some sense.

If we choose x to

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \|R(x)\|_1 \quad \text{or} \quad \underset{x \in \mathbb{R}^n}{\text{minimize}} \|R(x)\|_\infty \tag{6.2.3}$$

then we have a non-differentiable unconstrained optimization problem. There has been considerable research into methods for solving (6.2.3), see e.g. Gill, Murray, and Wright [1981], Murray and Overton [1980,1981], Bartels and Conn [1981] and Conn [1985]. Research is continuing into producing algorithms and software for such problems.

It is much more common to use the l_2 norm instead of (6.2.3), i.e.

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} f(x) = \frac{1}{2} \sum_{i=1}^{m} r_i(x)^2 = \frac{1}{2} R(x)^T R(x) \tag{6.2.4}$$

If each $y_i = m(t_i, \hat{x}) + e_i$ for some true parameter value $\hat{x}$ and some random error e_i, and if the m random errors arise from independent and identical normal distribution with mean 0, then (6.2.3) produces the maximum likelihood estimator of $\hat{x}$. In any case, (6.2.4) generally is easier to solve than (6.2.3).

If $R(x)$ is linear in x then (6.2.4) is a *linear least squares* problem and it can be solved in $0(mn^2)$ operations; see Stewart [1973], Lawson and Hanson [1974], or Golub and Van Loan [1983]. Otherwise it is a *nonlinear least squares* problem. The nonlinear least squares problem is a particular case of unconstrained optimization and can be solved by the techniques of the preceding sections. But many special purpose methods have arisen that take advantage of the special form of the derivatives of $f(x)$,

$$\nabla f(x) = \sum_{i=1}^{m} \nabla r_i(x) r_i(x) \equiv J(x)^T R(x)$$

where $J(x) \in \mathbb{R}^{m \times n}$ denotes the Jacobian matrix of $R(x)$, and

$$\nabla^2 f(x) = \sum_{i=1}^{m} (\nabla r_i(x) \nabla r_i(x)^T + \nabla^2 r_i(x) r_i(x)) \equiv J(x)^T J(x) + S(x) \tag{6.2.5}$$

where $S(x) \in \mathbb{R}^{n \times n}$ is the part of $\nabla^2 f(x)$ that is a function of second derivatives of $R(x)$.

If both first and second derivatives of $R(x)$ are available (and n is not too large) then the nonlinear least squares problem should probably just be solved by standard unconstrained optimization methods. Otherwise the goal of most special purpose nonlinear least squares methods is to produce efficient methods that require only analytic or finite difference first derivatives of $R(x)$. This supplies $\nabla f(x)$ and the first part of $\nabla^2 f(x)$, but not the second order part $S(x)$. Many strategies for using this special structure exist, and we summarize them very briefly. For more detail see Dennis and Schnabel [1983].

Gauss-Newton and Levenberg-Marquardt methods simply omit $S(x)$ and base their step on the model

$$m(x_c+d) = f(x_c) + \nabla f(x_c)^T d + \frac{1}{2} d^T J(x_c)^T J(x_c) d \qquad (6.2.6)$$

This is very reasonable if the data (6.2.1) is fit well by the optimal model $m(t_i, x_*)$ since in this case $R(x_*)$ and hence $S(x_*)$ will be nearly zero, and omitting $S(x)$ will cause little harm. Methods that use a line search in conjunction with (6.2.6) are generally called Gauss-Newton methods, while the use of a trust region leads to More 's [1978] derivation of the Levenberg-Marquardt method. This method is implemented in MINPACK (More , Garbow, and Hillstrom [1980] and has been quite effective in practice. Note that (6.2.6) is equivalent to

$$m(x_c+d) = \frac{1}{2} \| R(x_c) + J(x_c) d \|_2^2$$

and so these methods can be derived from the linear Taylor series model of $R(x)$.

Alternatively, secant methods for nonlinear least squares construct approximations to $\nabla^2 f(x)$ given by (6.2.5). Some codes have used the methods of Section 3.4 to approximate all of $\nabla^2 f(x)$, but more successful methods have arisen by approximating $\nabla^2 f(x_c)$ by

$$J(x_c)^T J(x_c) + A_c$$

where A_c approximates $S(x_c)$. That is, the available part of the Hessian is used and only the unavailable part is approximated. Dennis, Gay and Walsch [1981a,b] constructed a very effective method along these lines. In general, such quasi-Newton methods are more effective than modern Levenberg-Marquardt methods on problems where $R(x_*)$, and hence $S(x_*)$ is large, and of comparable effectiveness otherwise.

Research is continuing on various aspects of nonlinear least squares calculations, including large residual problems (Salane [1987]), large dimensional problems (Toint [1987]), secant approximation (Al-

Baali and Fletcher [1983], and application of the tensor methods mentioned in Section 6.5 (Hanson and Krogh [1987], Bouaricha and Schnabel [1987]).

An interesting variant of the data fitting problem occurs when there is experimental error in both t_i and y_i. In this case it becomes appropriate to measure the distance between the data point (t_i, y_i) and the fitting function $m(t,x)$ by the (weighted) Euclidean distance from the point to the curve. Boggs, Byrd, and Schnabel [1987] show that minimizing the sum of the squares of these distances leads to the problem

$$\underset{x\in\mathbb{R}^n,\ \delta\in\mathbb{R}^m}{\text{minimize}} \sum_{i=1}^{m} ((m(t_i+\delta_i,x)-y_i)^2+w_i^2\delta_i^2) \tag{6.2.7}$$

for appropriate weights w_i. Problem (6.2.7) is a nonlinear least squares problems in $m+n$ variables, but Schwetlick and Tiller [1985] and Boggs, Byrd, and Schnabel [1987] show how to solve it efficiently using essentially the same work per iteration as for the standard nonlinear least squares problem (6.2.4). Boggs, Byrd, Donaldson, and Schnabel [1987] provide a software package for (6.2.7). The case when $m(t,x)$ is linear in both t and x is addressed in Golub and Van Loan [1980].

Finally, there is an increasing amount of cross-fertilization between the optimization and statistical communities in the study of data fitting problems. Areas of interest include the application of modern optimization techniques to specialized data fitting problems (see e.g. Bates and Watts [1987], Bunch [1987], and the statistical analysis of parameter estimates obtained by nonlinear least squares (see e.g. Bates and Watts [1980], Cook and Witmer [1985], Donaldson and Schnabel [1987].

6.3 Secant Updates

The investigation of alternative secant methods for unconstrained optimization is enjoying a recent revival. This was a very active area in the 1960's and 70's, starting with the discovery of the first secant update, the DFP update (3.4.7), and continuing with the discovery of the BFGS update (3.4.5). Much interest was focused on the Broyden family of updates (Broyden [1970])

$$B_+(\theta) = \theta B_+^{DFP} + (1-\theta)B_+^{BFGS} \tag{6.3.1}$$

which differ from each other only by multiples of a rank-one matrix; this topic is discussed extensively in Fletcher [1980]. Several convergence results for any choice of $\theta\in[0,1]$ have been proven; see e.g. Griewank and Toint [1982b] and Stachurski [1981]. But the consensus for over 10 years has been that the

BFGS is best in practice. One hint is a powerful convergence result of Powell [1976] for the BFGS that has never been extended to the DFP.

Some recent research has attempted to explain theoretically the superiority of the BFGS. Powell [1986] uses a simple example to show that the DFP can be very much slower than the BFGS. Byrd, Nocedal, and Yuan [1987] extend Powell's 1976 convergence result to any $\theta \in (0,1]$, i.e. any convex combination of the DFP and the BFGS except the DFP, and in doing so show that the DFP lacks a self-correcting term in its update formula. Dennis, Martinez, and Tapia [1987] show that the BFGS has an optimal bounded deterioration property in the convex class.

Other recent research has resumed computational and theoretical investigation of choices other than the BFGS ($\theta = 0$) in (6.3.1). Zhang and Tewarson [1986] consider using $\theta<0$; they extend Powell's convergence result to their strategy and their computational results show that it may produce better results than the BFGS in practice. Conn, Gould, and Toint [1981] revisit the symmetric-rank on update,

$$B_+ = B_c + \frac{(y_c - B_c s_c)(y_c - B_c s_c)^T}{(y_c - B_c s_c)^T s_c}$$

the choice $\theta = (y_c^T s_c)/(y_c - B_c s_c)^T s_c$ in (6.3.1), which may have a zero denominator, and show that it may be competitive with the BFGS when used in conjunction with a trust region method and safeguarding of the denominator.

Dennis and Walker [1985] and Vu [1984] have studied the problem of dealing with noise in y for the least change Frobenius norm updates, but the more important analysis for the BFGS update is not so well understood.

A very important practical problem in secant updating comes from constrained optimization. We want to extend the BFGS method to maintain an approximation to the Hessian of the Lagrangian with respect to the primal variables. The most commonly used method is due to Powell [1978], but he shows in Powell [1985] that it can lead to ill-conditioning in the approximate Hessians. Tapia [1984] suggests and analyzes a promising procedure.

Some very elegant work on secant methods for *nonlinear* problems that come from discretization of infinite dimensional problems has been done by Griewank [1983] and Kelley and Sachs [1986]. They give a new derivation of the method for nonlinear two point boundary value problems suggested in Hart and

Soul [1973]. The idea behind this work is to consider the operator equation in its natural function space setting. Roughly speaking, one defines a least-change secant method for the operator equation using the norm in which Newton's method could be shown to have its familiar convergence properties. This gives a point-wise quasi-Newton method equivalent to a local affine model which one discretizes and solves. In other words, linearization precedes discretization instead of the standard approaches in which one discretizes the operator equation and then iteratively linearizes it.

Finally, the advent of parallel computers is leading to revived interest in *multiple secant updates.* These are methods that attempt to satisfy more than one secant equation at each iteration in order to interpolate more than one previous value of $g(x)$ for optimization, or of $F(x)$ for nonlinear equations. They were first mentioned by Barnes [1965] for nonlinear equations and shown by Gay and Schnabel [1978] and Schnabel and Frank [1987] to lead to small gains over Broyden's method for nonlinear equations. The application of this approach to unconstrained optimization has fundamental limitations, see Schnabel [1983]. But these methods now seem naturally suited to parallel computation where multiple values of $g(x)$ or $F(x)$ may be calculated at one iteration; see e.g. Byrd, Schnabel, and Shultz [1987].

6.4 Singular Problems

There are a number of practical problems for which $J(x_*)$ (for nonlinear equations) or $\nabla^2 f(x_*)$ (for optimization) are singular or nearly singular. We call these *singular problems.* None of the convergence results of Sections 2-3 apply to singular problems, and by considering one variable problems we can see that slow convergence is to be expected. For example, applying Newton's method for nonlinear equations to solve $x^2 = 0$ produces linear convergence with constant $\frac{1}{2}$, while applying Newton's method for optimization to minimize x^4 gives linear convergence with constant $\frac{2}{3}$. Clearly the standard linear and quadratic models are less helpful in these cases since all the derivatives used in the standard models approach zero as x converges to x_*.

There has been a considerable amount of recent research analyzing the behavior of standard methods on singular problems, and suggesting improved methods. Most of this research has been for nonlinear equations problems and is summarized excellently in Griewank [1985]. We will give a very brief

indication of this work.

Many researchers have analyzed the convergence of Newton's method for solving singular systems of nonlinear equations. For "regular" singularities, results such as those in Decker and Kelley [1980a,b] or Griewank [1980] show that one can still expect linear convergence with the constant converging to $\frac{1}{2}$. Decker and Kelley [1985] show that Broyden's method, like the one-dimensional secant method, is linearly convergent on singular problems with constant converging to $(\sqrt{5}-1)/2 \equiv 0.62$, from certain starting points.

Various acceleration schemes for solving singular systems of equations have been proposed, and they are surveyed in Griewank [1985]. Many authors have suggested methods related to the one variable idea of doubling the Newton step that are intended solely for singular problems. One difficulty in applying these techniques is that one may not know a priori whether the problem is singular. Some methods from curve following are also applicable to singular systems of equations. (see e.g. Moore and Spense [1980]). Griewank [1985] and Schnabel and Frank [1984,1987] have proposed methods that are applicable to both singular and nonsingular problems. These methods append a simple low rank quadratic term to the standard linear model, in a way that doesn't significantly increase the costs of forming, storing, or solving the model. Schnabel and Frank report that their "tensor" methods lead to significant improvements on both singular and nonsingular test problems.

The solution of singular unconstrained optimization problems is more complex. This is related to the fact that for one variable problems, if $f^{ii}(x_*) = 0$, then we must also have $f^{iii}(x_*) = 0$ and $f^{iv}(x_*) \geq 0$. Similarly, Griewank and Osborne [1983] show that if $\nabla^2 f(x_*)v = 0$ at a minimizer x_*, then we must have $\nabla^3 f(x_*)vvv = 0$, $\nabla^4 f(x_*)vvvv \geq 0$, and $\nabla^3 f(x)vv$ not too large. These conditions imply that the singularity is "irregular" and invalidate most of the approaches mentioned above. Schnabel and Chow [1987] introduce a "tensor" method that appends low rank third and fourth order terms to the standard quadratic model, without requiring any additional function or derivative evaluations and without appreciably increasing the cost of forming, storing, or solving the model. They report that their approach leads to a substantial reduction in the cost of solving both singular and nonsingular test problems.

6.5 Parallel Computation

An important recent development in computing is the commercial availability of *parallel computers*, computers that can perform multiple operations concurrently. These include *MIMD* (Multiple Instruction Multiple Data) *computers* that allow different instruction streams to be executed concurrently, *processor arrays* that apply the same instruction stream to multiple data streams concurrently, and *vector computers* that use data pipelining to rapidly perform pairwise addition or multiplication of vectors. Since it appears that many of the significant future gains in computer speed will come from effectively utilizing such machines, it is becoming important to design optimization methods that utilize them efficiently. Virtually all the research in this area is quite recent, and we will simply indicate some of the approaches that are emerging. These are primarily oriented towards MIMD computers, which seem to be the class of parallel computers best suited towards parallel optimization because they support concurrency at a coarse-grain algorithmic level.

One approach towards parallel optimization is to design general purpose parallel variants of the Newton or quasi-Newton methods discussed in Sections 2-4. These types of methods have have two potentially significant costs that must be considered in constructing parallel versions. They are the evaluations of functions and derivatives, and the linear algebra costs in solving linear systems or updating secant approximations. Both can be adapted to parallel computers.

The most obvious way to use a parallel computer effectively during the evaluation of functions or derivatives is to perform the multiple function evaluations of a finite difference gradient or Hessian calculation concurrently (Dixon [1981], Patel [1982], Lootsma [1984]). Schnabel [1986] introduces the idea of performing a *speculative* finite difference gradient evaluation concurrently with the evaluation of $f(x)$ at a trial point, before it is known whether this point will be accepted as the next iterate. Since the acceptance rate for trial points usually is at least 70%, this gradient value will usually be needed, so this simple strategy will utilize $n+1$ or fewer processors quite efficiently if function evaluation is the dominant cost. Byrd, Schnabel, and Shultz [1987] investigate the more difficult question of effectively utilizing more than $n+1$ (but fewer than $\frac{1}{2}n^2$) processors; this leads to new optimization methods that usually require significantly fewer iterations that the BFGS method. An alternative (see e.g. Patel [1982]) is to evaluate $f(x)$ at many

trial points simultaneously, but this appears unlikely to produce as much increase in speed.

An n become large it is also important to perform the linear algebra calculations in parallel. For methods that use the Hessian, parallel matrix factorizations and backsolves are required and many effective algorithms have been produced (see e.g. Heller [1978], Geist and Heath [1986]). For BFGS methods, Han [1986] has proposed an implementation that sequences a factorization ZZ^T of the inverse of the Hessian approximation and is well suited to parallel computers. An alternative is to utilize the unfactored inverse update (3.4.6), which appears to parallelize as well and require fewer arithmetic operations. Traditionally there has been some concern over the numerical stability of these approaches (see e.g. Powell [1987]); this issue is now being re-examined since they seem better suited to parallel computation than the Cholesy factorization update (3.4.4).

It will also be increasingly important to develop specific parallel optimization methods for particular classes of expensive optimization problems. Some early examples include Dixon, Ducksbury, and Singh [1982], and Dixon and Spedicato [1985]. There has also been work on parallel methods related to other optimization methods we have discussed; for example see Housos and Wing [1980] for a parallel conjugate direction method, and Lescrenier [1986] for a parallel approach to partially separable optimization.

REFERENCES

AL-BAALI, and FLETCHER [1983]. Variational methods for nonlinear least squares, Department of Mathematical Sciences Report NA/71, University of Dundee, Dundee, Scotland.

ARMIJO, L. [1966]. Minimization of functions having Lipschitz-continuous first partial derivatives, *Pacific Journal of Mathematics*, 16, pp. 1-3.

BARNES, J. [1965]. An algorithm for solving nonlinear equations based on the secant method, *Computer Journal*, 8, pp. 66-72.

BARTELS, R.H. and A.R. CONN [1981]. An approach to nonlinear l_1 data fitting, Computer Science Department Report CS-81-17, University of Waterloo, Waterloo, Canada.

BATES, D.M. and D.G. WATTS [1980]. Relative curvature measures of nonlinearity, *Journal of the Royal Statistical Society*, Ser. B, 42, pp. 1-25.

BATES, D.M. and D.G. WATTS [1987]. A generalized Gauss-Newton procedure for multi-response parameter estimation, *SIAM Journal on Scientific and Statistical Computing*, 8, pp. 49-55.

BEALE, E.M.L. [1972]. A derivation of conjugate gradients, in *Numerical Methods for Nonlinear Optimization*, F.A. Lootsma, ed., Academic Press, London, pp. 39-43.

BOGGS, R.H. BYRD, J.R. DONALDSON, and R.B. SCHNABEL [1987]. ORDPACK- Software for weighted orthogonal distance regression, Department of Computer Science, Report CU-CS-360-87, University of Colorado, Boulder, Colorado.

BOUARICHA, A. and R.B. SCHNABEL [1987]. A software package for tensor methods for nonlinear equations and nonlinear least squares, in preparation.

BROYDEN, C.G. [1965]. A class of methods for solving nonlinear simultaneous equations, *Mathematics of Computation*, 19, pp. 577-593.

BROYDEN, C.G. [1969]. A new double-rank minimization algorithm, *Notices of the American Mathematical Society*, 16, p. 670.

BROYDEN, C.G. [1970]. The convergence of a class of double-rank minimization algorithms, Parts I and II, *Journal of the Institute of Mathematics and its Applications*, 6, pp. 76-90, 222-236.

BROYDEN, C.G. [1971]. The convergence of an algorithm for solving sparse nonlinear systems, *Mathematics of Computation*, 25, pp. 285-294.

BROYDEN, C.G., J.E. DENNIS Jr., and J.J. MORE [1973]. On the local and superlinear convergence of quasi-Newton methods, *IMA Journal of Applied Mathematics*, 12, pp. 223-246.

BUCKLEY, A. and A. LENIR [1983]. QN-like variable storage conjugate gradients, *Mathematical Programming*, 27, pp. 155-175.

BUNCH, D.S [1987]. Maximum likelihood estimation of probabilistic choice models, *SIAM Journal on Scientific and Statistical Computing*, 8, pp. 56-70.

BYRD, R.H., J. NOCEDAL and Y. YUAN [1987]. Global convergence of a class of quasi-Newton methods on convex problems, *SIAM Journal on Numerical Analysis*, to appear.

BYRD, R.H., R.B. SCHNABEL and G.A. SHULTZ [1985]. A trust region algorithm for nonlinearly constrained optimization, Department of Computer Science Report CU-CS-313-85, University of

Colorado, Boulder, Colorado, to appear in *SIAM Journal on Numerical Analysis*.

BYRD, R.H., R.B. SCHNABEL and G.A. SHULTZ [1987]. Using parallel function evaluations to improve Hessian approximations for unconstrained optimization, Department of Computer Science Report CU-CS-361-87, University of Colorado, Boulder, Colorado.

CANTRELL, J.W. [1969]. Relation between the memory gradient method and the minimization of functions, *Journal of Optimization Theory and its Applications*, 4, pp. 67-71.

COLEMAN, T.F., B. GARBOW and J.J. MORE [1984]. Software for estimating sparse Jacobian matrices, *ACM Transactions on Mathematical Software*, 10, pp. 329-347.

COLEMAN, T.F., B. GARBOW and J.J. MORE [1985]. Software for estimating sparse Hessian matrices, *ACM Transactions on Mathematical Software*, 11, pp. 363-378.

COLEMAN, T.F. and J.J. MORE [1983]. Estimation of sparse Jacobian matrices and graph coloring problems, *SIAM Journal on Numerical Analysis*, 20, pp. 187-209.

COLEMAN, T.F. and J.J. MORE [1984]. Estimation of sparse Hessian matrices and graph coloring problems, *SIAM Journal on Numerical Analysis*, 28, pp. 243-270.

CONN, A.R. [1985]. Nonlinear programming, exact penalty functions and projection techniques for nonsmooth functions, in *Numerical Optimization 1984*, P.T. Boggs, R.H. Byrd, and R.B. Schnabel, eds., SIAM, Philadelphia, pp. 3-25.

COOK, R.D. and J.A. WITMER [1985]. A note on parameter-effects curvature, *Journal of the American Statistical Association*, 80, pp. 872-878.

CRAGG, E.E. and A.V. LEVY [1969]. Study on a supermemory gradient method for the minimization of functions, *Journal of Optimization Theory and its Applications*, 4, pp. 191-205.

CURTIS, A., M.J.D. POWELL and J.K. REID [1974]. On the estimation of sparse Jacobian matrices, *IMA Journal of Applied Mathematics*, 13, pp. 117-120.

DAVIDON, W.C. [1959]. Variable metric methods for minimization, Argonne National Labs Report ANL-5990.

DECKER, D.W. and C.T. KELLEY [1980a]. Newton's method at singular points I, *SIAM Journal on Numerical Analysis*, 17, pp. 66-70.

DECKER, D.W. and C.T. KELLEY [1980b]. Newton's method at singular points II, *SIAM Journal on Numerical Analysis*, 17, pp. 465-471.

DECKER and KELLEY [1985]. Broyden's method for a class of problems having singular Jacobian at the root, *SIAM Journal on Numerical Analysis*, 22, pp. 566-574.

DEMBO, R.S., S.C. EISENSTAT and T. STEIHAUG [1982]. Inexact Newton methods, *SIAM Journal on Numerical Analysis*, 19, pp. 400-408.

DENNIS, J.E. Jr., D.M. GAY, and R.E. WELSCH [1981]. An adaptive nonlinear least-square algorithm, *ACM Transactions on Mathematical Software*, 7, pp. 348-368.

DENNIS, J.E. JR. and G. LI [1986]. A hybrid algorithm for solving sparse nonlinear systems of equations, *Mathematics of Computation*, in press

DENNIS, J.E. Jr., H.J. MARTINEZ, and R.A. TAPIA [1987]. A convergence theory for the structured BFGS secant method with an application to nonlinear least squares, TR87-15, Department of

Mathematical Sciences, Rice University, Houston, Texas.

DENNIS, J.E. Jr. and H.H. MEI [1979]. Two new unconstrained optimization algorithms which use function and gradient values, *Journal of Optimization Theory and its Applications*, 28, pp. 453-482.

DENNIS, J.E. Jr. and J.J. MORE [1974]. A characterization of superlinear convergence and its application to quasi-Newton methods, *Mathematics of Computation*, 28, pp. 549-560.

DENNIS, J.E. Jr. and J.J. MORE [1977]. Quasi-Newton methods, motivation and theory, *SIAM Review*, 19, pp. 46-89.

DENNIS, J.E. Jr. and R.B. SCHNABEL [1979]. Least change secant updates for quasi-Newton methods, *SIAM Review*, 21, pp. 443-459.

DENNIS, J.E. Jr. and R.B. SCHNABEL [1981]. A new derivation of symmetric positive definite secant updates, *Nonlinear Programming* 4, O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, eds., Academic Press, New York.

DENNIS, J.E. Jr. and R.B. SCHNABEL [1983]. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey.

DENNIS, J.E. Jr. and K. TURNER [1986]. Generalized Conjugate Directions, *Journal for Linear Algebra and Applications*, 88/89, pp.187-209.

DENNIS, J.E. Jr. and H.F. WALKER [1981]. Convergence theorems for least-change secant update methods, *SIAM Journal on Numerical Analysis*, 18, pp. 949-987.

DENNIS, J.E. Jr. and H.F. WALKER [1985]. Least-change sparse secant update methods with inaccurate secant conditions, *SIAM Journal on Numerical Analysis*, 22, pp. 760-778.

DENNIS, J.E. Jr. and D.J. WOODS [1987]. Optimization on microcomputers: The Nelder-Mead simplex algorithm, in *New Computing Environments: Microcomputers in Large-Scale Computing*, A. Wouk, ed., SIAM, Philadelphia, pp. 116-122.

DIXON, L.C.W. [1981]. The place of parallel computation in numerical optimization I, the local problem, Technical Report No. 118, Numerical Optimisation Centre, The Hatfield Polytechnic.

DIXON, L.C.W, P.G. DUCKSBURY and P. SINGH [1982]. A parallel version of the conjugate gradient algorithm for finite element problems, Technical Report No. 132, Numerical Optimisation Centre, The Hatfield Polytechnic.

DIXON, L.C.W. and E. SPEDICATO [1985]. Software developments for numerical on-line parallel optimisation of engine fuel consumption, Technical Report No. 172, Numerical Optimisation Centre, The Hatfield Polytechnic.

DONALDSON, J.R. and R.B. SCHNABEL [1987]. Computational experience with confidence regions and confidence intervals for nonlinear least squares, *Technometrics*, 29, pp. 67-82.

DUFF, I.S., A.M. ERISMAN and J. K. REID [1986]. *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford.

FLACHS, J. [1986]. On the generation of updates for quasi-Newton methods, *Journal of Optimization Theory and its Applications*, 48, pp. 379-418.

FLETCHER, R. [1970]. A new approach to variable metric methods, *Computer Journal*, 13, pp. 317-322.

FLETCHER, R. [1980]. *Practical Methods of Optimization, vol. 1, Unconstrained Optimization*, John Wiley and Sons, New York.

FLETCHER, R. and M.J.D. POWELL [1963]. A rapidly convergent descent method for minimization, *Computer Journal*, 6, pp. 163-168.

FLETCHER, R. and C.M. REEVES [1964]. Function minimization by conjugate gradients, *Computer Journal*, 7, pp. 149-154.

GAY, D.M. [1979]. Some convergence properties of Broyden's method, *SIAM Journal on Numerical Analysis*, 16, pp. 623-630.

GAY, D.M. [1981]. Computing optimal locally constrained steps, *SIAM Journal on Scientific and Statistical Computing*, 2, pp. 186-197.

GAY, D.M. [1983]. Subroutines for unconstrained minimization using a model/trust-region approach, *ACM Transactions on Mathematical Software*, 9, pp. 503-524.

GAY, D.M. and R.B. SCHNABEL [1978]. Solving systems of nonlinear equations by Broyden's method with projected updates, *Nonlinear Programming 3*, L. Mangasarian, R.R. Meyer, and S.M. Robinson, eds., Academic Press, New York, pp. 245-281.

GEIST, G.A. and M.T. HEATH [1986]. Matrix computation on distributed memory multiprocessors, in *Hypercube Multiprocessors 1986*, M.T. Heath ed., SIAM, Philadelphia, pp. 161-180.

GILL, P.E., G.H. GOLUB, W. MURRAY and M.A. SAUNDERS [1974]. Methods for modifying matrix factorizations, *Mathematics of Computation*, 28, pp. 505-535.

GILL, P.E. and W. MURRAY [1974]. Newton-type methods for unconstrained and linearly constrained optimization, *Mathematical Programming*, 28, pp. 311-350.

GILL, P.E., W. MURRAY and M.H. WRIGHT [1981]. *Practical Optimization*. Academic Press, London, New York, Toronto, Sydney and San Francisco.

GOLDFARB, D. [1970]. A family of variable metric methods derived by variational means, *Mathematics of Computation*, 24, pp. 23-26.

GOLDFARB, D. [1976]. Factorized variable metric methods for unconstrained optimization, *Mathematics of Computation*, 30, pp. 796-811.

GOLDFARB, D. and PH.L. TOINT [1984]. Optimal estimation of Jacobian and Hessian matrices that arise in finite difference calculations, *Mathematics of Computation*, 43, pp. 69-88.

GOLDFELDT, S.M., R.E. QUANDT, and H.F. TROTTER [1966]. Maximization by quadratic hill-climbing, *Econometrica*, 34, pp. 541-551.

GOLDSTEIN, A.A. [1967]. *Constructive Real Analysis*, Harper & Row, New York.

GOLUB, G.H. and C.F. VAN LOAN [1983]. *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland.

GOLUB, G.H. and C.F. VAN LOAN [1980]. An analysis of the total least squares problem, *SIAM Journal on Numerical Analysis*, 17, pp. 883-893.

GRIEWANK, A.O. [1980a]. Analysis and modification of Newton's method at singularities, Ph.D. thesis, Australian National University.

GRIEWANK, A.O. [1980b]. Starlike domains of convergence for Newton's method at singularities, *Numerische Mathematik*, 35, pp. 95-111.

GRIEWANK, A.O. [1983]. Note on the weighting of Schubert's update for discretizations of ordinary differential equations, Technical Report, Department of Mathematics, Southern Methodist University.

GRIEWANK, A.O. [1985]. On solving nonlinear equations with simple singularities or nearly singular solutions, *SIAM Review*, 27, pp. 537-564.

GRIEWANK, A.O. and PH.L. TOINT [1982a]. Partitioned variable metric updates for large sparse optimization problems, *Numerische Mathematik*, 39, pp. 119-137.

GRIEWANK, A.O. and PH.L. TOINT [1982b]. Local convergence analysis for partitioned quasi-Newton updates, *Numerische Mathematik*, 39, pp. 429-448.

GRZEGORSKI, S.M. [1985]. Orthogonal projections on convex sets for Newton-like methods, *SIAM Journal on Numerical Analysis*, 22, pp. 1208-1219.

HART, W.E. and S.O.W. SOUL [1973]. Quasi-Newton methods for discretized nonlinear boundary value problems, *IMA Journal of Applied Mathematics*, 11, pp. 351-359.

HAN, S.P. [1986]. Optimization by updated conjugate subspaces, in *Numerical Analysis: Pitman Research Notes in Mathematics Series 140*, D.F. Griffiths and G.A. Watson, eds., Longman Scientific and Technical, Burnt Mill, England, pp. 82-97.

HANSON and KROGH [1987]. A new algorithm for constrained nonlinear least squares problems, manuscript.

HEBDEN, M.D. [1973]. An algorithm for minimization using exact second derivatives, Technical report TP515, Atomic Energy Research Establishment, Harwell, England.

HELLER, D. [1978]. A survey of parallel algorithms in numerical linear algebra, *SIAM Review*, 20, pp. 740-777.

HOFFMAN, K. and R. KUNZE [1971]. Linear Algebra, Prentice-Hall, Englewood Cliffs, N.J.

HOUSOS, E.C. and O. WING [1980]. Parallel nonlinear minimization by conjugate directions, *Proceedings of the 1980 International Conference on Parallel Processing*, pp. 157-158.

KHACHIYAN, L.G. [1979]. A polynomial algorithm in linear programming, *Doklady Akademiia Nauk SSSR Novaia Seriia*, 224, pp. 1093-1096.

LESCRENIER, M. [1986]. Partially separable optimization and parallel computing, Report No. 86/5, Department of Mathematics, Facultes Universitaires ND de la Paix, Belgium.

LAWSON, C.L. and R.J. HANSON [1974]. Solving least squares problems, Prentice Hall, Englewood Cliffs, NJ.

LEVENBERG, K. [1944]. A method for the solution of certain problems in least squares, *Quart. Appl. Math.*, 2, pp. 164-168.

LI, G. [1986]. Algorithms for solving sparse nonlinear systems of equations, TR86-6, Department of Mathematical Sciences, Ph.D. Thesis, Rice University.

LOOTSMA, F.A. [1984]. Parallel unconstrained optimization methods, Report 84-30. Reports of the Department of Mathematics and Informatics. Delft University of Technology.

MARQUARDT, D. [1963]. An algorithm for least-squares estimation of nonlinear parameters, *SIAM Journal on Applied Mathematics*, 11, pp. 431-441.

MARWIL, E.S. [1978]. Exploiting sparsity in Newton-type methods, Cornell Applied Math. Ph.D. Thesis.

MARWIL, E.S. [1979]. Convergence results for Schubert's method for solving sparse nonlinear equations, *SIAM Journal on Numerical Analysis*, 16, pp. 588-604.

MATTHIES, H. and G. STRANG [1979]. The solution of nonlinear finite element equations, *International Journal on Numerical Methods in Engineering*, vol. 14, pp. 1613-1626.

MCCORMICK, G.P. [1977]. A modification of Armijo's step-size rule for negative curvature, *Mathematical Programming*, 13, pp. 111-115.

MIELE, A. and J.W. CANTRELL [1969]. Study on a memory gradient method for the minimization of functions, *Journal of Optimization Theory and its Applications*, 3, pp. 459-470.

MOORE, G. and A. SPENCE [1980]. The calculation of turning points of nonlinear equations, *SIAM Journal on Numerical Analysis*, 17, pp. 567-576.

MORE, J.J. [1977]. The Levenberg-Marquardt algorithm: implementation and theory, *Numerical Analysis*, G.A. Watson, ed., pp. 105-116, Lecture Notes in Mathematics 630, Springer-Verlag, Berlin, Heidelberg and New York.

MORE, J.J. [1978]. The Levenberg-Marquardt algorithm: implementation and theory, *Numerical Analysis, Dundee 1977. Lecture Notes in Mathematics 630*, G.A. Watson, ed., Springer-Verlag, Berlin, pp. 105-116.

MORE, J.J., B.S. GARBOW, and K.E. HILLSTROM [1980]. User guide for MINPACK-1, Argonne National Laboratory Report ANL-80-74.

MORE, J.J., B.S. GARBOW and K.E. HILLSTROM [1981]. Fortran subroutines for testing unconstrained optimization software, *ACM Transactions on Mathematical Software*, 7, pp. 136-140.

MORE, J.J. and D.C. SORENSEN [1979]. On the use of directions of negative curvature in a modified Newton method, *Mathematical Programming*, 16, pp. 1-20.

MORE, J.J. and D.C. SORENSEN [1982]. Newton's method, Argonne National Labs Report ANL-82-8, Argonne, Illinois.

MORE, J.J. and D.C. SORENSEN [1983]. Computing a trust region step, *SIAM Journal on Scientific and Statistical Computing*, 4, pp. 553-572.

MURRAY, W. and M.L. OVERTON [1980]. A projected Lagrangian algorithm for nonlinear minimax optimization, *SIAM Journal on Scientific and Statistical Computing*, 1, pp. 345-370.

MURRAY, W. and M.L. OVERTON [1981]. A projected Lagrangian algorithm for nonlinear l_1 optimization, *SIAM Journal on Scientific and Statistical Computing*, 2, pp. 207-224.

NAZARETH, J.L. [1986]. The method of successive affine reduction for nonlinear minimization, *Mathematical Programming*, 35, pp. 97-109.

NELDER, J.A. and R. MEAD [1965]. A simplex method for function minimization, *Computer Journal*, 7, pp. 308-313.

ORTEGA, J.M. and W.C. RHEINBOLDT [1970]. *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York.

PATEL, K.D. [1982]. Implementation of a parallel (SIMD) modified newton algorithm on the ICL DAP, Technical Report 131, The Hatfield Polytechnic Numerical Optimisation Centre.

POWELL, M.J.D. [1970a]. A hybrid method for nonlinear equations, in *Numerical Methods for Nonlinear Algebraic Equations*, P. Rabinowitz, ed., Gordon and Breach, London, pp. 87-114.

POWELL, M.J.D. [1970b]. A new algorithm for unconstrained optimization, in *Nonlinear Programming*, J.B. Rosen, O.L. Mangasarian and K. Ritter, eds., Academic Press, New York, pp. 31-65.

POWELL, M.J.D. [1975]. Convergence properties of a class of minimization algorithms, *Nonlinear Programming*, 2, O.L. Mangasarian, R.R. Meyer, S.M. Robinson, eds., Academic Press, New York, pp. 1-27.

POWELL, M.J.D. [1976]. Some global convergence properties of a variable metric algorithm without exact line searches, in *Nonlinear Programming*, R. Cottle and C. Lemke, eds., AMS, Providence, Rhode Island, pp. 53-72.

POWELL, M.J.D. [1977]. Restart procedures for the conjugate-gradient method, *Mathematical Programming*, 12, pp. 241-254.

POWELL, M.J.D. [1978]. The convergence of variable metric methods for nonlinearly constrained optimization problems, *Nonlinear Programming*, 3, O.L. Mangasarian, R. Meyer, S. Robinson, eds., Academic Press, New York, pp. 27-63.

POWELL, M.J.D. [1984]. On the global convergence of trust region algorithms for unconstrained minimization, *Mathematical Programming*, 29, pp. 297-303.

POWELL, M.J.D. [1985]. The performance of two subroutines for constrained optimization on some difficult test problems, in *Numerical Optimization 1984*, P.T. Boggs, R.H. Byrd, R.B. Schnabel, eds., SIAM, Philadelphia, pp. 160-177.

POWELL, M.J.D. [1986]. How bad are the BFGS and DFP methods when the objective function is quadratic?, *Mathematical Programming*, 34, pp. 34-47.

POWELL, M.J.D. [1987]. Updating conjugate directions by the BFGS formula, *Mathematical Programming*, 38, pp. 29-46.

POWELL, M.J.D. and PH.L. TOINT [1979]. On the estimation of sparse Hessian matrices, *SIAM Journal on Numerical Analysis*, 16, pp. 1060-1074.

REID, J.K. [1973]. Least squares solution of sparse systems of non-linear equations by a modified Marquardt algorithm, in *Proceedings* of the NATO Conf. at Cambridge, July 1972, North Holland, Amsterdam, pp. 437-445.

REINSCH, C. [1971]. Smoothing by spline functions II, *Numerische Mathematik*, 16, pp. 451-454.

REID, J.K. [1973]. Least squares solution of sparse systems of non-linear equations by a modified Marquardt algorithm, in *Proceedings* of the NATO Conf. at Cambridge, July 1972, North Holland, Amsterdam, pp. 437-445.

SALANE, D.E. [1987]. A continuation approach for solving large-residual nonlinear least squares problems, *SIAM Journal on Scientific and Statistical Computing*, 8, pp. 655-671.

SCHNABEL, R.B. [1983]. Quasi-Newton methods using multiple secant equations, Technical Report CU-CS-247-83, Department of Computer Science, University of Colorado at Boulder.

SCHNABEL, R.B. [1986]. Concurrent function evaluations in local and global optimization, Technical

Report CU-CS-345-86, Department of Computer Science, University of Colorado at Boulder, to appear in *Computer Methods in Applied Mechanics and Engineering.*

SCHNABEL, R.B. and CHOW [1987]. Tensor methods for unconstrained optimization, in preparation.

SCHNABEL, R.B. and P.D. FRANK [1984]. Tensor methods for nonlinear equations, *SIAM Journal on Numerical Analysis*, 21-5, pp. 815-843.

SCHNABEL, R.B. and P.D. FRANK [1987]. Solving systems of nonlinear equations by tensor methods, in *The State of the Art in Numerical Analysis*, A. Iserles and M.J.D. Powell, eds., Clarendon Press, Oxford, pp. 245-271.

SCHNABEL, R.B., J.E. KOONTZ and B.E. WEISS [1985]. A modular system of algorithms of unconstrained minimization, *ACM Transactions on Mathematical Software*, 11, pp. 419-440.

SCHNABEL, R.B. and E. VAN VLECK [1987]. A new modified Cholesky decomposition algorithm, in preparation.

SCHUBERT, L.K. [1970]. Modification of a quasi-Newton method for nonlinear equations with a sparse Jacobian, *Mathematics of Computation*, 24, pp. 27-30.

SCHWETLICK, H. and V. TILLER [1985]. Numerical methods for estimating parameters in nonlinear models with errors in the variables, *Technometrics*, 27, pp. 17-24.

SHANNO, D.F. [1970]. Conditioning of quasi-Newton methods for function minimization, *Mathematics of Computation*, 24, pp. 647-657.

SHANNO, D.F. and K.H. PHUA [1978a]. Matrix conditioning and nonlinear optimization, *Mathematical Programming*, 14, pp. 145-160.

SHANNO, D.F. and K.H. PHUA [1978b]. Numerical comparison of several variable metric algorithms, *Journal of Optimization Theory and its Applications*, 25, pp. 507-518.

SHULTZ, G.A., R.B. SCHNABEL and R.H. BYRD [1985]. A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties, *SIAM Journal on Numerical Analysis*, 22, pp. 47-67.

SORENSEN, D.C. [1981]. An example concerning quasi-Newton estimation of a sparse Hessian, *ACM SIGNUM Newsletter*, 16, 2, pp. 8-10.

SORENSEN, D.C. [1982]. Newton's method with a model trust region modification, *SIAM Journal on Numerical Analysis*, 19, pp. 409-426.

STACHURSKI, A. [1981]. Superlinear convergence of Broyden's bounded Θ-class of methods, *Mathematical Programming*, 20, pp. 196-212.

STEWART, G.W. III [1973]. *Introduction to Matrix Computations*, Academic Press, New York.

STEIHAUG, T. [1980]. Quasi-Newton Methods for Large Scale Nonlinear Problems, Ph.D dissertation, SOM Technical Report #49, Yale University.

STEIHAUG, T. [1981]. The conjugate gradient method and trust regions in large scale optimization, *SIAM Journal on Numerical Analysis*, 20, pp. 626-637.

TAPIA, R.A. [1984]. On secant updates for use in general constrained optimization, TR84-3, Department of Mathematical Sciences, Rice University, Houston, Texas. Revised January 1987. To appear in *Mathematics of Computation.*

THOMAS, S.W. [1975]. Sequential estimation techniques for quasi-Newton algorithms, Technical Report TR75-227, Department of Computer Science, Cornell University.

TOINT, PH.L. [1977]. On sparse and symmetric matrix updating subject to a linear equation, *Mathematics of Computation*, 31, pp. 954-961.

TOINT, PH.L. [1978]. Some numerical results using a sparse matrix updating formula in unconstrained optimization, *Mathematics of Computation*, 32, pp. 839-851.

TOINT, PH.L. [1981]. Towards an efficient sparsity exploiting Newton method for minimization, in *Sparse matrices and their uses*, I.S. Duff, ed., Academic Press, London, pp. 57-88.

TOINT, PH.L. [1987]. On a large scale nonlinear least squares calculations, *SIAM Journal on Scientific and Statistical Computing*, 8, pp. 416-435.

VINSOME, P.K.W. [1976]. Orthomin, an iterative method for solving sparse sets of simultaneous linear equations, in Society of Petroleum Engineers of AIME, *Proceedings* of the Fourth Symposium on Reservoir Simulation.

VU, P.A. [1984]. Symmetric secant updates with inaccurate secant conditions. Ph.D. Thesis, University of Houston.

WOLFE, P. [1969]. Convergence conditions for ascent methods, *SIAM Review*, 11, pp. 226-235.

WOLFE, P. [1971]. Convergence conditions for ascent methods II: some corrections, *SIAM Review*, 13, pp. 185-188.

WOODS, D.J. [1985]. An interactive approach for solving multi-objective optimization problems, Ph.D. Thesis, Rice University, available as Math. Science TR85-5.

YOUNG, D.M. and K.C. JEA [1980]. Generalized conjugate gradient acceleration of iterative methods, *Linear Algebra and its Applications*, vol. 34, pp. 159-194.

ZHANG, Y. and R.P. TEWARSON [1986]. On the development of algorithms superior to the BFGS method in Broyden's family of updates, Department of Applied Mathematics and Statistics Report AMS 86-69, State University of New York, Stony Brook, New York.