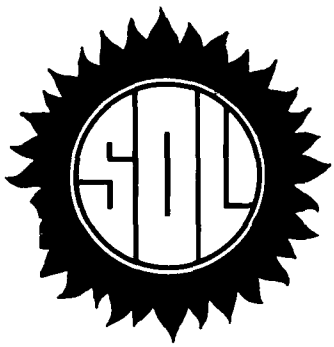
AD-A106 028 STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB F/G 12/1
OPTIMIZATION OF UNCONSTRAINED FUNCTIONS WITH SPARSE HESSIAN MAT--ETC(U)
AUG 81 M N THAPA N00014-75-C-0267
UNCLASSIFIED SOL-81-12 NL

1 OF 1
AD A106 028

END
DATE FILMED
11-81
DTIC

SOL

LEVEL II



Systems
Optimization
Laboratory

AD A106028

DTIC FILE COPY

DTIC
ELECTE
OCT 2 2 1981
S D

A

This document has been approved for public release and sale; its distribution is unlimited.

Department of Operations Research
Stanford University
Stanford, CA 94305

81 10 6 268

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

OPTIMIZATION OF UNCONSTRAINED FUNCTIONS WITH SPARSE HESSIAN MATRICES -- QUASI-NEWTON METHODS

by

Mukund N. Thapa

TECHNICAL REPORT SOL 81-12

August 1981

Research and reproduction of this report were partially supported by the Department of Energy Contract AM03-76SF00326, PA# DE-AT03-76ER72018; New Office of Naval Research Contract N00014-75-C-0267; National Science Foundation Grants MCS-7681259, MCS-7926009 and ECS-8012974.

Reproduction in whole or in part is permitted for any purposes of the United States Government. This document has been approved for public release and sale; its distribution is unlimited.

ABSTRACT

Newton-type methods and quasi-Newton methods have proven to be very successful in solving dense unconstrained optimization problems. Recently there has been considerable interest in extending these methods to solving large problems when the Hessian matrix has a known *a priori* sparsity pattern. This paper treats sparse quasi-Newton methods in a uniform fashion and shows the effect of loss of positive-definiteness in generating updates. These sparse quasi-Newton methods coupled with a modified Cholesky factorization to take into account the loss of positive-definiteness when solving the linear systems associated with these methods were tested on a large set of problems. The overall conclusions are that these methods perform poorly in general — the Hessian matrix becomes indefinite even close to the solution and superlinear convergence is not observed in practice.

For
GRA&I ☑
TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

OPTIMIZATION OF UNCONSTRAINED FUNCTIONS WITH SPARSE HESSIAN MATRICES — QUASI-NEWTON METHODS

1. Introduction

The problem of concern in this paper is the unconstrained minimisation of a twice-continuously differentiable function

$$\underset{x\in\mathbb{R}^n}{\text{minimise}}\ f(x). \tag{1.1}$$

We shall consider the class of quasi-Newton methods applied to problem (1.1) when the Hessian matrix of the function $f(x)$ has a known *a priori* sparsity pattern. The first quasi-Newton method was suggested by Davidon (1959) and extended by Fletcher and Powell (1963). Since then there has been an explosion of interest in these methods. (For a comprehensive survey of quasi-Newton methods see Dennis and Moré, 1977).

The idea behind the most popular quasi-Newton methods for (1.1) is to maintain a positive-definite symmetric matrix that approximates the Hessian matrix of $f(x)$. If we let x_k denote the kth iterate, a quasi-Newton algorithm obtains a descent direction p_k by solving the system of equations

$$B_k p_k = -g_k, \tag{1.2}$$

where B_k is an approximation to the Hessian matrix at iteration k and g_k is the gradient of f at x_k. If B_k is positive-definite, p_k is guaranteed to be a descent direction. Once having obtained p_k, the new point x_{k+1} is given by $x_k + \alpha_k p_k$, where $\alpha_k > 0$ is chosen so that $f(x_{k+1})$ is "sufficiently" less than $f(x_k)$ (for a precise definition of this term, see, for example, Ortega and Rheinboldt, 1970). If the new point x_{k+1} does not satisfy the convergence criteria, a new approximation to the Hessian matrix B_{k+1} is defined by

$$B_{k+1} = \varphi_k B_k + U_k, \tag{1.3}$$

where φ_k is a scalar, and U_k is a matrix chosen so that B_{k+1} is symmetric, positive-definite and satisfies the *quasi-Newton condition*

$$B_{k+1} s_k = y_k, \tag{1.4}$$

with

$$s_k = x_{k+1} - x_k \quad \text{and} \quad y_k = g_{k+1} - g_k .$$

Usually, U_k is a matrix of low rank. The most popular update is the BFGS (Broyden-Fletcher-Goldfarb-Shanno) update (see, e.g. Dennis and Moré, 1977), in which

$$\varphi_k = 1$$
$$U_k = \frac{y_k y_k^T}{s_k^T y_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}. \tag{1.5}$$

To guarantee that the updated matrices are positive definite, we require that $s_k^T y_k > 0$ (which can be ensured by a suitable choice of α_k).

Quasi-Newton methods have been very successful in solving unconstrained and constrained optimization problems of moderate size. The difficulty in applying these methods as described above to large problems is that a symmetric $n \times n$ matrix (or a factorization of a matrix) must be stored. However, many large problems have a sparse Hessian matrix whose sparsity pattern is known (or can be determined) a *priori*. In these cases it would be possible to maintain a suitably sparse approximation to the Hessian matrix. Note that, in general, the quasi-Newton methods described above generate dense approximations to the Hessian, regardless of the sparsity structure of the true Hessian. Much recent research has been directed towards generating Hessian approximations that have the same sparsity structure as the true Hessian (see Dennis and Schnabel, 1978; Marwil, 1978; Shanno, 1980; Toint, 1977, 1978, 1979). This paper treats various aspects of sparse versions of quasi-Newton methods in a uniform manner; and, reports results of extensive numerical testing of sparse quasi-Newton methods combined with a modified Cholesky factorization to provide a positive-definite matrix for computing the search direction.

Section 2 describes the notation to be used in the rest of the paper. Sparse quasi-Newton updates are described in Section 3. Section 4 describes positive-definiteness as related to these quasi-Newton methods. Storage and efficiency are considered in Section 5; and convergence of these methods are described in Section 6. Finally, Section 7 discusses computational aspects of these methods and Section 8 summarizes the important results.

2. Definitions and Notation

In the remainder of this paper, the subscript k will be dropped and the subscript $k+1$ will be replaced by the superscript "+".

Let $G(x)$ denote the Hessian matrix of f at x, and let

$$N = \{(i,j) : [G(x)]_{ij} = 0 \qquad \text{for all } x \in \Re^n\}, \tag{2.1}$$

that is, the set N represents the sparsity pattern of the true Hessian matrix of the function being minimized. Note that the sparsity pattern is assumed to be known and fixed, and that any additional zeros created during execution of the

algorithm are treated as nonzeros. Let

$$\begin{aligned} \overline{N} &= \{(i,j) : i,j = 1,2,\ldots,n\} \setminus N \\ &= \{(i,j) : [G(x)]_{ij} \neq 0 \quad \text{for at least one } x \in \mathbb{R}^n\}. \end{aligned} \tag{2.2}$$

For any symmetric matrix A, define matrices A_N and $A_{\overline{N}}$ as follows

$$[A_N]_{ij} = \begin{cases} A_{ij}, & \text{for } (i,j) \in N, \\ 0, & \text{for } (i,j) \in \overline{N}, \end{cases} \tag{2.3}$$

$$[A_{\overline{N}}]_{ij} = \begin{cases} 0, & \text{for } (i,j) \in N, \\ A_{ij}, & \text{for } (i,j) \in \overline{N}. \end{cases} \tag{2.4}$$

In words, A_N is the matrix A with zeros in the positions corresponding to the nonzeros of $G(x)$; and $A_{\overline{N}}$ is the matrix A with zeros in the positions corresponding to the zeros of $G(x)$. Then A can be written as

$$A = A_N + A_{\overline{N}}. \tag{2.5}$$

Define D_i to be a diagonal matrix whose diagonal elements are 0 or 1 depending on the sparsity pattern of the ith row of $G(x)$, that is,

$$[D_i]_{jj} = \begin{cases} 1, & \text{if } (i,j) \in \overline{N}, \\ 0, & \text{if } (i,j) \in N. \end{cases} \tag{2.6}$$

Finally, define $s^i = D_i s$ for any vector s.

3. Sparse Quasi-Newton Updates

In the dense case, it is possible to obtain an updated Hessian approximation, B^+, that is symmetric, positive-definite, and satisfies the quasi-Newton condition

$$B^+ s = y, \tag{3.1}$$

where $s = x^+ - x$ and $y = g^+ - g$. When generating updates that satisfy sparsity requirements, it is not always possible to obtain an Hessian approximation B^+ that is positive-definite and also satisfies the quasi-Newton condition (see, for example, Shanno, 1980).

Let B be the sparse symmetric $n \times n$ matrix representing the approximation to the Hessian matrix at the start of iteration k of a quasi-Newton algorithm. If

sparsity were the only issue, sparse updates could be developed by setting

$$B^+ = \eta B + U_N, \tag{3.2}$$

where $\eta B + U$ satisfies the quasi-Newton condition (3.1) and U_N is defined by (2.4). Another possibility would be to obtain the factors of B^+ by updating the factors of B (see, for example, Gill, Golub, Murray and Saunders, 1974) while ignoring fill-in. However, neither of these approaches would result in updates that satisfy the quasi-Newton condition (3.1).

A sparse version of any quasi-Newton update can easily be obtained by applying Schubert's (1970) technique. For example, a sparse version of the BFGS update (1.5) can be obtained as

$$B^+_{BFGS} = \sum_{i=1}^{n} e_i w_i^T, \tag{3.3}$$

where

$$w_i^T = e_i^T \left(\frac{y(y^i)^T}{s^T(y^i)} - \frac{Bs(s^TB)^i}{(s^TB)^i s} \right). \tag{3.4}$$

However, B^+_{BFGS} is not symmetric. To obtain a symmetric version (see, for example, Shanno, 1980), we redefine the updated matrix as

$$B^+_{BFGS} = \sum_{i=1}^{n} \lambda_i [e_i w_i^T + w_i e_i^T], \tag{3.5}$$

where w_i is defined by (3.4) and the λ_i's are chosen so that B^+_{BFGS} satisfies the quasi-Newton condition (3.1).

One of the disadvantages of this approach is that the matrix in the system of equations obtained from (3.5) to define λ_i need not be positive-definite. Another problem is that the line search used guarantees that $y^T s > 0$ but does not necessarily guarantee that $(y^i)^T s > 0$ and $(s^TB)^i s > 0$ for all i. In fact, this sparse version of the BFGS update has proved to be computationally unstable due to $(y^i)^T s$ and/or $(s^TB)^i s > 0$ being zero or close to zero (see, for example, Shanno, 1980).

An alternative technique for obtaining sparse quasi-Newton updates is to use norm minimization, since some dense quasi-Newton updates can be defined as least-change updates in appropriate norms (see, for example, Dennis and Schnabel, 1978). Toint (1977) showed how to obtain a sparse analog of the Powell-Symmetric-Broyden update (see, for example, Dennis, and Moré, 1977) by finding a matrix E such that $B^+ = B + E$ is closest to B in the Frobenius norm; has the sparsity pattern specified by the set N (Equation 2.1); and satisfies

the quasi-Newton condition (3.1). Formally, the problem can be stated as

$$\text{NM1} \qquad \text{minimize} \quad \|E\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} E_{ij}^2 \tag{3.6}$$

$$\text{subject to} \quad Es = y - Bs \tag{3.7}$$

$$E_{ij} = 0 \qquad (i,j) \in N \tag{3.8}$$

$$E = E^T. \tag{3.9}$$

Theorem 3.1 (Toint, 1977). *The unique solution to problem NM1 is given by*

$$E_{ij} = \begin{cases} 0, & \text{if } (i,j) \in N; \\ \lambda_i s_j + \lambda_j s_i, & \text{if } (i,j) \in \overline{N}; \end{cases} \tag{3.10}$$

where $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)^T$ *is the solution of the linear system*

$$Q\lambda = y - Bs \; (= Es) \tag{3.11}$$

with Q *defined by*

$$Q_{ij} = (s^i)_j (s^j)_i + \|s^i\|_2^2 \, \delta_{ij}, \qquad \text{for all } (i,j) \tag{3.12}$$

and δ_{ij} *is the Kronecker delta.*

Proof. The proof involves finding the stationary point of the Lagrangian of problem NM1. For details, see Toint (1977) or Dennis and Schnabel (1978). ∎

Obtaining the desired update by Toint's method requires the solution of a linear system of equations (3.11). The matrix Q in this system has the same sparsity pattern as the matrix B and is positive-definite if and only if $\|s^i\| > 0$ for all i.

Note that (3.10) and (3.12) can be rewritten in matrix notation as follows:

$$E = \sum_{i=1}^{n} \lambda_i [e_i (s^i)^T + s^i e_i^T], \tag{3.13}$$

$$Q = \sum_{i=1}^{n} [(s^i)_i s_i + \|s^i\|_2^2 e_i] e_i^T. \tag{3.14}$$

In some situations it may be appropriate to use a weighted Frobenius norm as the minimizing function in problem NM1 (see, for example, Dennis and Schnabel, 1978). The difficulty with using an arbitrary symmetric positive-definite weighting matrix W is that it is not easy to obtain a closed form solution to this problem. However, when W is a diagonal matrix with positive diagonal elements, a closed form solution for this problem exists (see, Dennis and Schnabel, 1978; Powell, 1979; Toint, 1977).

Theorem 3.1 indicated how a sparse analog of the Powell-Symmetric-Broyden update could be obtained. Shanno (1980) derived a sparse analog to the BFGS update and indicated that his method could be extended to other symmetric updates. However, his method is rather complicated and not at all easy to apply to other updates. The following theorem shows how sparse analogs to symmetric updates can be derived as a simple extension of Toint's (1977) results.

Theorem 3.2. *Let $\hat{B}^+ = \eta B + U$ where U is symmetric, but, in general, will not have its sparsity pattern specified by N (see Equation 2.1); η is some scale factor; and $\hat{B}^+ s = y$. Then, the symmetric matrix B^+ that is closest to $\hat{B}^+_N$ in the Frobenius norm, and that satisfies the quasi-Newton condition (3.1) is obtained as*

$$B^+ = \hat{B}^+_N + E, \tag{3.15}$$

where E is the solution to problem NM2

NM2 minimize $\|E\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} E_{ij}^2$ *(3.16)*

subject to

$$Es = \hat{B}^+_N s \tag{3.17}$$

$$E_{ij} = 0 \qquad (i,j) \in N \tag{3.18}$$

$$E = E^T. \tag{3.19}$$

Proof. By definition

$$\hat{B}^+_N = U_N \tag{3.20}$$

and

$$\hat{B}^+_N = \eta B_N + U_N. \tag{3.21}$$

From (3.15),

$$\begin{aligned} B^+ s &= (\hat{B}^+_N + E)s \\ &= (\hat{B}^+ - \hat{B}^+_N + E)s \qquad \text{(by definition of } \hat{B}^+_N\text{)} \\ &= y - (\hat{B}^+_N - E)s. \end{aligned}$$

Hence, it follows that B^+ satisfies the quasi-Newton condition (3.1) if and only if $(\hat{B}^+_N - E)s = 0$ or $Es = \hat{B}^+_N s$. This gives us condition (3.17). Conditions (3.18) and (3.19) follow from considerations of sparsity and symmetry on the matrix E. ∎

Corollary 3.3. *The solution to problem NM2 is*

$$E_{ij} = \begin{cases} 0, & \text{if } (i,j) \in N; \\ \lambda_i s_j + \lambda_j s_i, & \text{if } (i,j) \in \overline{N}; \end{cases} \tag{3.22}$$

where $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)^T$ *is the solution of the linear system*

$$Q\lambda = \hat{B}_N^+ s \,(= Es) \tag{3.23}$$

with Q defined by (3.12).

Proof. The proof follows from Theorems 3.1 and 3.2 after noting that problems NM1 and NM2 differ only in equations (3.7) and (3.17). ∎

Analogous to Toint's result with the weighted Frobenius norm, we can extend Theorem 3.2 to the weighted Frobenius norm (see Thapa, 1980).

From the above corollary we see that the sparse analogs of quasi-Newton updates require the solution of a system of equations (see Equation 3.23) whose right-hand side is $\hat{B}_N^+ s$. Shanno (1980) indicates that the computation of $\hat{B}_N^+ s$ does not require the *storage* of the elements of U_N (see Definition 2.1), but does require the *computation* of the elements of U_N. However, the following lemma shows that the elements of U_N need not be computed either.

Lemma 3.4.

$$\hat{B}_N^+ s = y - \eta B s - U_{\overline{N}} s. \tag{3.24}$$

Proof.

$$\begin{aligned} \hat{B}_N^+ s &= U_N s && \text{(from (3.20))} \\ &= (U - U_{\overline{N}}) s && \text{(by definition of } U_N) \\ &= (\hat{B}^+ - \eta B) s - U_{\overline{N}} s && \text{(since } \hat{B}^+ = \eta B + U) \\ &= y - \eta B s - U_{\overline{N}} s. \end{aligned}$$

∎

Toint's update and the sparse analogs of the dense quasi-Newton updates modify all the elements of the Hessian matrix to obtain a sparse version that satisfies the quasi-Newton condition. It may be of some importance to consider obtaining sparse updates that modify the diagonal in a predetermined way. Then, for example, we could obtain a sparse approximation to the BFGS update that has the same diagonal as the diagonal that would be obtained if the dense BFGS update were used at the current iteration of the optimization algorithm. Further details on this approach are given in Thapa, 1980.

4. On Positive-definiteness

An implementation of a quasi-Newton algorithm using a sparse analog of a dense update requires the solution of two systems of equations. First, a system of the form $Q\lambda = r$ is solved to obtain the sparse Hessian approximation. Next, a system of the form $Bp = -g$ is solved to obtain a descent direction p.

4.1 The Hessian Approximation.

If the Hessian approximation B is not positive-definite, the solution of the system of equations $Bp = -g$ may not result in a descent direction. Previous work on quasi-Newton methods has treated an indefinite B with a *trust-region strategy* (see, for example, Dembo et al, 1980; Sorensen, 1980; Steihaug, 1980). The idea behind these methods is to choose a parameter γ such that $\overline{B} = B + \gamma I$ is positive-definite and to define the search direction as the solution of

$$(B + \gamma I)p = -g,$$

so that p is a descent direction.

This paper studies a different method for obtaining a descent direction. We use a method based on the Cholesky factorization (or LDL^T factorization, where L is unit lower triangular and D is a positive diagonal matrix) for symmetric positive-definite matrices. The j-th column of the matrix L is defined from columns 1 through $j-1$ by the equations

$$d_j = B_{jj} - \sum_{s=1}^{j-1} d_s l_{js}^2,$$

$$l_{ij} = \frac{1}{d_j}(B_{ij} - \sum_{s=1}^{j-1} d_s l_{js} l_{is}).$$

We use a numerically stable method to construct a matrix $\overline{B}$ from a *modified Cholesky factorization* of B. (For a detailed description of the modified Cholesky factorization seee Gill, Murray, and Wright (1981)). A descent direction p is then obtained as the solution of $Bp = -g$. With this approach, the Cholesky factors L and D are computed subject to two requirements: all elements of D are strictly positive, and the elements of the factors satisfy a uniform bound, i.e. for $k = 1, \ldots, n$ and some positive value β it holds that

$$d_k > \delta \quad \text{and} \quad |l_{ik} d_k^{\frac{1}{2}}| \leq \beta, \quad i > k. \tag{4.1}$$

The matrices L and D are computed by implicitly increasing the diagonal elements of the original matrix during the factorization in order to satisfy (4.1).

When this process is completed, the matrices L and D are the Cholesky factors of a positive-definite matrix $\overline{B}$ that satisfies

$$\overline{B} = LDL^T = B + P,$$

where P is a non-negative diagonal matrix whose j-th element is P_{jj}.

The modified Cholesky factorization is a numerically stable method that produces a positive-definite matrix differing from the original matrix only in its diagonal elements. The diagonal modification is *optimal* in the sense that an *a priori* bound upon the norm of P is minimized, subject to the requirement that sufficiently positive-definite matrices are left unaltered. In practice, the actual value of P is almost always substantially less than the *a priori* bound.

Once having obtained a direction of descent by the modified Cholesky factorization we determine the steplength along p such that the function value at the new point $x + \alpha p$ is "sufficiently" lower than the function value at the current point x. In order to obtain such a decrease, the most important condition on the parameter α is

$$|g(x + \alpha p)^T p| \leq -\eta g^T p^T, \quad \text{for } 0 \leq \eta \leq 1,$$

where η is called the *linesearch accuracy* parameter.

Above we discussed techniques to obtain a descent direction p when the Hessian approximation is not positive-definite. The question then is whether it is possible to alter the approximation B to obtain B^+ such that B^+ is symmetric, satisfies the quasi-Newton condition (3.1) and is positive-definite. Toint (1979) has shown that it is indeed possible to do so under certain restrictions on the sparsity pattern of B. His result is of theoretical importance, but is not practically implementable.

Note that when using the Levenberg-Marquardt procedure or the sparse modified Cholesky factorization we solve the modified system $\overline{B}p = -g$ (where $\overline{B} = B + \lambda I$ in the former case and $\overline{B} = B + P$, where P is a diagonal matrix, in the latter case). An obvious approach to obtaining a positive-definite Hessian approximation would be to replace B by $\overline{B}$. In this case $\overline{B}$ would have the required sparsity pattern, be symmetric and positive-definite, but would not satisfy the quasi-Newton condition (3.1). However, it would be hoped that the next approximation $\overline{B}^+$ obtained from $\overline{B}$ would be "better" than the approximation B^+ obtained from B. Unfortunately, it turns out that $\overline{B}^+$ can be more indefinite than B^+. To show this we first need some new definitions and notation. It should be noted that the technique described below can be applied to any method if we regard $\overline{B}$ as the starting positive-definite approximation.

P – Diagonal modification to make $B + P$ sufficiently

positive-definite. (4.2)

$\overline{B} = B + P.$ (4.3)

$B^+ = B + U_N + E.$ (4.4)

$\overline{B}^+ = \overline{B} + \overline{U}_N + \overline{E}.$ (4.5)

λ – multipliers for obtaining E. (4.6)

$\overline{\lambda}$ – multipliers for obtaining $\overline{E}$. (4.7)

U – Update obtained using B. (4.8)

$\overline{U}$ – Update obtained using $\overline{B}$. (4.9)

$h_1(P) = \overline{U} - U.$ (4.10)

$h(P) = h_1(P) + \eta P.$ (4.11)

From (4.11) and (4.10),

$$[h(P)]_N = [h_1(P)]_N \tag{4.12}$$

and

$$[h(P)]_{\overline{N}} = [h_1(P)]_{\overline{N}} + \eta P. \tag{4.13}$$

Next, note that

$$\eta\overline{B} + \overline{U} = \eta B + U + h(P). \tag{4.14}$$

Now, from Corollary 3.3, B^+ can be obtained from B as follows :

$$B^+_{ij} = \begin{cases} 0, & \text{if } (i,j) \in N; \\ \hat{B}^+_{ij} + \lambda_i s_j + \lambda_j s_i, & \text{if } (i,j) \in \overline{N}; \end{cases} \tag{4.15}$$

where $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)^T$ solves

$$Q\lambda = \hat{B}^+_N s \tag{4.16}$$

with Q defined by equation (3.12) and

$$\hat{B}^+ = \eta B + U. \tag{4.17}$$

Again , from Corollary 3.3, it follows that $\overline{B}^+$ can be obtained from $\overline{B}$ as

$$\overline{B}^+_{ij} = \begin{cases} 0, & \text{if } (i,j) \in N; \\ \hat{\overline{B}}^+_N + \overline{\lambda}_i s_j + \overline{\lambda}_j s_i, & \text{if } (i,j) \in \overline{N}; \end{cases} \tag{4.18}$$

where $\overline{\lambda} = (\overline{\lambda}_1, \overline{\lambda}_2, \ldots, \overline{\lambda}_n)^T$ solves

$$Q\overline{\lambda} = \hat{\overline{B}}^+_N s \tag{4.19}$$

with Q defined by equation (3.12) and

$$\hat{\overline{B}}^+ = \eta\overline{B} + \overline{U}. \tag{4.20}$$

The next theorem shows the relationship between $\overline{\lambda}$ and λ and between $\overline{B}^+$ and $\overline{B}$.

Theorem 4.1. *Assume* $||s^i|| > 0$ *for all* i. *Let* z *solve*

$$Qz = [h(P)]_{\overline{N}} s. \tag{4.21}$$

Then

$$(a)\ \overline{\lambda} = \lambda - z \tag{4.22}$$

$$(b)\ \overline{B}^+_{ij} = \begin{cases} 0, & \text{if } (i,j) \in N; \\ B^+ + [h(P)]_{ij} - (z_i s_j + z_j s_i), & \text{if } (i,j) \in \overline{N}; \end{cases} \tag{4.23}$$

where Q *is defined by (3.12);* $\overline{N}$ *and* N *are defined by (2.1) and (2.2) ; and the rest of the quantities are defined by equations (4.2) – (4.14).*

Proof.

$$\begin{aligned} \hat{\overline{B}}^+_N s &= y - \eta \overline{B} s - \overline{U}_{\overline{N}} s \\ &= y - \eta(B+P)s - (U_{\overline{N}} - [h_1(P)]_{\overline{N}})\, s \\ &= y - \eta B s - U_{\overline{N}} s - (\eta P + [h_1(P)]_{\overline{N}})\, s \\ &= y - \eta B s - U_{\overline{N}} s - [h(P)]_{\overline{N}} s \\ &= \hat{B}^+_N s - [h(P)]_{\overline{N}} s, \end{aligned} \tag{4.24}$$

where the first and fifth equalities follow from Lemma 3.7, the second equality follows from (4.3) and (4.10), and the fourth equality follows from (4.11). From (4.19) we obtain,

$$\begin{aligned} Q\overline{\lambda} &= \hat{\overline{B}}^+_N s \\ &= \hat{B}^+_N s - [h(P)]_{\overline{N}} s && \text{from (4.24)} \\ &= Q\lambda - [h(P)]_{\overline{N}} s && \text{from (4.16).} \end{aligned}$$

Since $||s^i|| > 0$ for all i , it follows from Lemma 3.2 (b) that Q^{-1} exists. Hence

$$\begin{aligned} \overline{\lambda} &= \lambda - Q^{-1}[h(P)]_{\overline{N}} s \\ &= \lambda - z, \end{aligned}$$

which is (4.22).

Next, we prove (4.23). From (4.18) it follows that for $(i,j) \in N$, $B^+_{ij} = 0$; and for $(i,j) \in \overline{N}$

$$\begin{aligned} \overline{B}_{ij} &= \hat{\overline{B}}^+_{ij} + \overline{\lambda}_i s_j + \overline{\lambda}_j s_i \\ &= \hat{B}^+_{ij} + [h(P)]_{ij} + (\lambda_i - z_i)s_i + (\lambda_j - z_j)s_i && \text{from (4.14) and (4.22)} \\ &= B^+_{ij} + [h(P)]_{ij} - (z_i s_j + z_j s_i). && \text{from (4.15)} \end{aligned}$$

∎

The above theorem shows the effect of a diagonal modification on sparse updates. From equation (4.23) in the statement of the theorem, it is clear that if the z_i and s_j are of the appropriate sign and magnitude then it is possible for $\overline{B}^+$ to be more indefinite than B^+.

The following example illustrates that $\overline{B}^+$ can be more indefinite than B. We shall only summarize the results. The full calculations are given in Thapa (1980). We use Toint's update and a sparse modified Cholesky factorization to compute $\overline{B}^+$.

Example 4.1. Note that for Toint's update, $h(P) = P$, $\varphi = 1$, $U = 0$. Hence,

$$z = Q^{-1}Ps$$

and

$$B^+_{ij} = \begin{cases} 0, & \text{for } (i,j) \in N \\ B^+_{ii} + P_{ii} - 2z_i s_i, & \text{for } i = 1, \ldots, n \\ B^+_{ij} - (z_i s_j + z_j s_i), & \text{for } (i,j) \in \overline{N},\ i \neq j. \end{cases}$$

Let

$$B = \begin{pmatrix} 25 & 5 & 3 & 0 & 0 \\ 5 & 12 & 0 & 0 & 9 \\ 3 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 5 & 4 \\ 0 & 9 & 0 & 4 & 1 \end{pmatrix},$$

$g = (1, 1, 20, 1, -2)^T$, $y = (2, 4, 3, -1, -2)^T$, and the step-length $\alpha = 1$.

All the numbers shown have been rounded to four digits. The modified Cholesky factors of B are

$$L_B = \begin{pmatrix} 1 & & & & \\ .2 & 1 & & & \\ .12 & -.05455 & 1 & & \\ 0 & 0 & 0 & 1 & \\ 0 & .8182 & 2.547 & .8 & 1 \end{pmatrix},$$

$$D_B = \text{diag}(25,\ 11,\ .1927,\ 5,\ 10.81),$$

with

$$P_B = \text{diag}(0,\ 0,\ .3855,\ 0,\ 21.63). \tag{4.25}$$

Therefore,

$$B^+ = \begin{pmatrix} 25.03 & 6.219 & 2.868 & 0 & 0 \\ 6.219 & 10.39 & 0 & 0 & 2.722 \\ 2.868 & 0 & .3693 & 0 & 0 \\ 0 & 0 & 0 & 3.140 & 2.438 \\ 0 & 2.722 & 0 & 2.438 & 7.356 \end{pmatrix},$$

$$B^+ = \begin{pmatrix} 25.01 & 5.122 & 2.965 & 0 & 0 \\ 5.122 & 11.83 & 0 & 0 & 9.260 \\ 2.965 & 0 & .3826 & 0 & 0 \\ 0 & 0 & 0 & 5.061 & 4.052 \\ 0 & 9.260 & 0 & 4.052 & 22.42 \end{pmatrix}.$$

Hence, we get

$$||P_B||_2^2 = 468.0,$$
$$||P_{B^+}||_2^2 = .00113,$$

and

$$||P_{B^+}||_2^2 = 760.5.$$

Therefore,

$$||P_{B^+}||_2^2 > ||P_{B^+}||_2^2.$$

Also $||P_{B^+}||_2^2 < ||P_B||_2^2$ and $||P_{B^+}||_2^2 > ||P_B||_2^2$.

The example described above was generated by using a modified Cholesky factorization. However, it would be easy to generate a similar example for trust-region methods.

4.2 Solving $Q\lambda = r$.

The matrix Q in Equation (3.12) is positive-definite if and only if $||s^i|| > 0$ for $i = 1, \ldots, n$ (see Toint, 1977). If any of the s^i has a zero norm, then the matrix Q is singular (and positive semi-definite). In this case the system $Q\lambda = r$ may have no solution. Toint (1977) suggested setting λ_j and r_j to zero for all j such that $||s^j|| = 0$, and solving the reduced system obtained by deleting the zero rows and columns of Q. Another technique for solving the system when Q is singular is to use the sparse version of the modified Cholesky factorization (see Thapa, 1980). Both these techniques generate λ_i's that produce Hessian approximations that do not necessarily satisfy the quasi-Newton condition (3.1).

Numerical experience has shown that not only is it possible for Q to be singular, but that Q may remain singular for many iterations. The following example illustrates this fact.

Example 4.2. Consider a generalization of the well known two-dimensional Rosenbrock function (Rosenbrock, 1960)

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

The Hessian matrix for this function is tridiagonal.

If $B_0 = I$, and $x_0 = (-1.2, 1, -1.2, 1, 1, 1, \ldots, 1)^T$, then $f_0 = 533.4$ and $g_0 = (-215.6, 792, -655.6, -88, 0, 0, \ldots, 0)^T$. The solution of $B_0 p_0 =$

$-g_0$ is $p_0 = -g_0$. With a linesearch accuracy $\eta = .9$, α is taken as .000958. Clearly, the last $n-6$ rows and columns of Q are all zero. The updated Hessian B_1 then has all zero off diagonal elements in the last $n-5$ columns. On the next iteration the last $n-7$ rows and columns of Q are all zero. A total of $n-6$ iterations are required before Q becomes positive-definite. (However, for this particular example, the system of equations $Q\lambda = r$ is consistent and the Hessian approximations generated do indeed satisfy the quasi-Newton condition (3.1).)

5. Storage and Efficiency

The dense quasi-Newton updates are all either rank-one or rank-two symmetric updates and most of them yield a positive-definite Hessian approximation when $y^T s > 0$ (an exception being the Powell-Symmetric-Broyden update). Provided that these low rank updates yield positive-definite Hessian approximations, it is possible to update the Cholesky factors of the Hessian in $O(n^2)$ operations (for example, see Gill, Golub, Murray, and Saunders, 1974). Hence, solving $Bp = -g$ at each iteration to obtain a descent direction p requires $O(n^2)$ arithmetic operations.

In the sparse case, however, the updates are all of rank n and there does not exist any known method of efficiently updating the factors of the Hessian approximation. This necessitates solving a new linear system of equations to obtain a descent direction p in each iteration of the algorithm. Furthermore, updating the Hessian approximation at each iteration requires the solution of an additional linear system of equations of the form $Q\lambda = r$ (see Section 3) where Q has the same sparsity pattern as the Hessian matrix. In general, the Cholesky factorization requires $O(n^3)$ arithmetic operations, and, hence, it would be undesirable to solve even one system of linear equations at each iteration. However, it is expected that the presence of sparsity would reduce the arithmetic operations needed to compute the Cholesky factors from scratch to $O(n^2)$ or less.

In the dense case the factors are updated at each iteration and hence we need space only to store the factors. In the sparse case, however, a copy of the Hessian approximation as well as its Cholesky factors must be stored. Extra space is not required for Q, since Q can be stored at the end of the vector storing the lower-triangular factors of the Hessian approximation, and the factors of Q can be overwritten on Q.

6. Convergence

It is important to be able to show that an algorithm using the sparse quasi-Newton updates of Section 3 generates a sequence of points $\{x_k\}_{k=0}^{\infty}$ that converge to a point x^* such that $g(x^*) = 0$. For a proof of one such result see

Thapa, 1980. Being able to prove that an algorithm converges is not sufficient to make it useful from the point of view of practical applications. It is usually desirable to have a fast rate of convergence. Toint (1979a) showed that under certain conditions his algorithm, which employs a trust-region procedure to solve for the descent direction, converges superlinearly. This is an interesting theoretical result; however, it did not hold in many tests performed by the author.

Table 6.1 shows the last few iterations of Toint's sparse quasi-Newton method using a modified Cholesky factorization to solve for the descent direction. If Toint's (1979a) theory of superlinear convergence is to hold we should observe superlinear convergence at this stage regardless of whether we use a trust region strategy or a modified Cholesky factorization to solve for the descent direction. However, superlinear convergence usually is not observed in practice, as Table 6.1 indicates. The table is typical of numerous tests in which superlinear convergence was not observed, even when the algorithm was started close to the solution with the exact Hessian matrix!

A number of possible reasons exist to explain the failure of superlinear convergence in practice. One of the reasons is that the sequence of updates generated by Toint's method consistently failed to remain positive-definite even close to the solution (as shown by the nonzero entries in column P_D of Table 6.1 which indicates that the Hessian matrix was modified). Another reason is that the region around the solution in which x_k converges superlinearly is so small that the limitations of finite precision make it impossible to improve the initial estimates.

TABLE 6.1
Toint's Update on Test Function *Genrose*
Dimension 4, linesearch accuracy 0.9

k	α_k	$f_k - f^*$	$\|x_k - x^*\|_2$	P_D	$\|g_k\|_2$
46	.011	7.3×10^{-11}	4.4×10^{-7}	.14	3.9×10^{-4}
47	.947	1.8×10^{-14}	2.3×10^{-7}	0.0	3.0×10^{-6}
48	1.00	8.6×10^{-15}	1.9×10^{-7}	5.7	1.3×10^{-7}
49	.223	6.6×10^{-16}	1.1×10^{-9}	0.0	1.3×10^{-6}
50	.911	8.7×10^{-19}	1.3×10^{-9}	0.0	2.8×10^{-8}
51	.775	0.0	9.7×10^{-10}	7.0	3.7×10^{-9}

KEY

k – Iteration number.
α_k – Step length at iteration k.
x_k – Approximation to the solution at iteration k.
x^* – The solution.
f^* – Function value at x^*.
P_D – Maximum addition to the diagonal during factorisation. A nonzero value indicates an indefinite matrix.
g_k – Gradient at the point x_k.

7. Numerical Results and Discussion

In this section we discuss the numerical performance of the various quasi-Newton type algorithms. The algorithms were tested on a wide range of problems. Thus, it is hoped that the numerical results will be valuable in analysing the strong and weak points of the various methods, and determining the circumstances under which the methods are most successful.

Section 7.1 discusses criteria for comparing the numerical performance of some of the algorithms tested. A key to the algorithms tested is given in Section 7.2 and the test problems used for the comparison of these algorithms are described in Appendix C. The last section discusses the numerical results. Only a small sample of the algorithms tested are discussed here. For more detailed numerical results, see Thapa (1980).

7.1 Basis for Comparison

For the purpose of comparing algorithms it is necessary to have a uniform standard of comparison (Gill and Murray, 1979a), which will be called an *assessment* criterion. A termination criterion is not a desirable basis for comparison

for at least two reasons. Firstly, there is no universal agreement on the best termination criterion for any given situation. Secondly, a wide variation in accuracy of the solution may be obtained for two different algorithms using the same termination criterion.

The assessment criterion used here is that suggested by Gill and Murray (1979a). The first point x_k is taken for which

$$f_k - f^* < \psi(1 + |f^*|). \tag{7.1.1}$$

In all the tests carried out, ψ was chosen to be 10^{-5}.

7.2 Key to the Algorithms Tested

This section lists the algorithms tested. All the algorithms described are implemented with the sparse modified Cholesky factorization (see Thapa, 1980), the line search of Gill et al. (1979b), and the assessment criterion of Section 7.1.

Three types of sparse quasi-Newton are described here.

TOINT – The update described by Toint (Section 2) to maintain a sparse approximation to the Hessian matrix that satisfies the quasi-Newton condition.

BFGS – The update suggested by Shanno (Section 2) to modify the BFGS update to obtain a sparse approximation that satisfies the quasi-Newton condition.

DFP – The Davidon-Fletcher-Powell update (see, for example, Dennis and Moré, 1977) modified (as described in Theorem 3.2) to yield a sparse approximation that satisfies the quasi-Newton condition.

The following algorithms were compared with the above algorithms:

QNM – A quasi-Newton method using the full $n \times n$ BFGS update to approximate the Hessian matrix.

UBFGS – This method updates the sparse Cholesky factors by using algorithm C1 described in the paper by Gill, Golub, Murray and Saunders (1974) and ignoring all fill-in (in the factors) when the dense BFGS update is used to approximate the Hessian matrix.

DFD – Direct method for finite differencing (See Powell and Toint, 1979) with the sparse modified Cholesky factorization.

7.3 Test Problems

The generation of an adequate set of test problems to compare a set of algorithms is not an easy task. It is important that a large carefully-selected set of problems should be used to test the algorithms. The set of problems should be sufficiently diverse so that one or more of the algorithms do not exploit peculiarities common to the set by adjusting certain free parameters in the algorithms. Furthermore, the set of problems should be chosen to appropriately test the algorithms under consideration. For example, it is pointless to test sparsity exploiting algorithms exclusively on problems of small dimension. On the other hand, testing large problems can become prohibitively expensive in terms of CPU time. A list of the test problems and the starting points used by the algorithms on these problems can be found in Appendix C.

7.4 Discussion of Numerical Results

All the algorithms are coded in double precision Fortran IV. The runs were made on a DEC-20 System, for which the machine precision, ϵ, is approximately 10^{-18}, and the largest number representable is approximately 10^{38}.

A total of 27 problems were solved. Each problem was solved using the values 0.9, 0.1 and 0.001 for η, the accuracy of the line search (see Section 4.1). Many different algorithms (see Thapa, 1980), including those described in Section 7.2, were tested. Each algorithm requires two parameters in addition to the line search accuracy: S_{max}, a bound upon the change $\|x_{k+1} - x_k\|$ at each iteration, and f_{est}, an estimate of the value of the objective function at the solution. In all the cases the value of S_{max}(see Section 7.4.3) was set to 10^5 (essentially implying an upper bound of infinity), and the value of f_{est} was set to the value of $f(x)$ at the solution. The algorithms also require an estimate of the space required by the nonzeros of the Cholesky factors of the Hessian matrix.

The results of all the tests are displayed in Appendices A and B. Appendix A contains tables of storage requirements of the Hessian matrix and its Cholesky factors; and execution times of the various routines. Appendix B contains the results of testing the algorithms described in Section 7.2.

7.4.1 Storage Required for the Hessian Matrix and its factors.

Table A1 in Appendix A is a comparison of the space required in double word lengths by the Hessian matrix and its Cholesky factors for the dense quasi-Newton method, one finite-difference scheme and the sparse quasi-Newton methods. For details on the computation of these numbers, see Thapa (1980). It is interesting to see that for a function like the 7Diagonal, the space required by the sparsity-exploiting methods is not much less than the space required by a dense quasi-Newton method. This is because there is considerable fill-in in the

Cholesky factors of the Hessian matrix. For situations such as these, where the factors fill in considerably, it is necessary to obtain a symmetric permutation of the matrix that would reduce the fill-in. However, even this may not necessarily resolve the difficulty (as is the case for the 7Diagonal function). For large problems with considerable fill-in, the modified-Newton algorithm utilising a finite-difference scheme to obtain the Hessian matrix can be used by rejecting all fill-in in the factors, or by rejecting some fill-in by utilising some sort of a partial factorization scheme (see Thapa, 1980). It is interesting to compare the space requirements for the calculus of variation problems ranging in dimension from 10 to 1000. These problems have a block-tridiagonal structure. For $n = 10$, the sparse quasi-Newton methods require more space than the dense quasi-Newton methods. As n grows large, the space required by the dense quasi-Newton method grows rapidly and it becomes impractical to implement the method from the point of view of the storage. The space required by the sparse quasi-Newton methods grows much faster than for the finite-difference method. For example, when $n = 1000$ the space required to store the approximation to the Hessian matrix and its LDL^T factors, is greater for sparse quasi-Newton methods, than for a modified-Newton method using the direct method (Algorithm DFD) for finite-differencing, by 2745 double word lengths. Thus, the maximum size of the problem that can be solved by the sparse quasi-Newton method is smaller than the maximum size of the problem that can be solved by a modified-Newton algorithm utilising a finite-difference approximation scheme.

7.4.2 Time Required for Different Tasks.

Table A2 in Appendix A compares the CPU time in seconds for the various tasks performed in each of the algorithms. All the numbers in the table were obtained by one computer run when the system was free of any other jobs. Each task (excepting GENPAT) was executed 50 times and the average time is reported in Table A2. Even so, these numbers are not very accurate and are merely meant to compare the time spent in executing the various tasks for different functions. Note that the time required to obtain a finite-difference approximation to the Hessian matrix includes the time spent in evaluating the gradient vectors for the different groups. The tasks of generating the pattern of the Hessian matrix (GENPAT), of forming groups for finite-differencing (GROUP), and of analyzing the Hessian matrix to determine the fill-in in the Cholesky factors (ANALYZ) need be done only once for a given function. Once these tasks have been completed, the appropriate information can be read in. It is especially advantageous to read in the pattern of the Hessian matrix, as the results on Calvar2 for $n = 500$ show (Table A2). It is interesting to note that the time spent in obtaining the Cholesky factors (FACTOR) and in solving a system of equations (SOLVE) is usually small (except for the function 7Diagonal where

there is considerable fill-in and the time for FACTOR is large). For the chained Rosenbrock function (ChaRose) the time to evaluate the function and gradient at any point is smaller than the time spent in FACTOR and SOLVE. For functions of this type the standard measure of function evaluations only is not suitable; and, thus it would be more useful to compare the various algorithms on the basis of numbers of iterations in addition to the number of function evaluations. In fact, for the functions ChaRose and QOR the time required to obtain a sparse quasi-Newton update is more than the time required to obtain a finite-difference approximation to the Hessian matrix. A comparison of the times to FACTOR and SOLVE for the function Calvar2 for n=100 to 500 shows the interesting fact that the CPU time increases linearly with n.

7.4.3 Influence of Stepmx.

The numerical results obtained in our implementation of Toint's update differ from the results shown in Toint's (1978) paper. One of the reasons for this is that Toint (1978) uses a "trust-region method" whereas we use a "step-length method".

In both these methods the $\|x_{k+1} - x_k\|$ can be bounded by a scalar. In a step-length algorithm a uniform bound, S_{max}, is used for all iterations; whereas, in a trust-region method, a scalar Δ_k (the size of the trust-region) is adjusted at each iteration. By choosing different initial estimates of Δ_0 it is possible to obtain better results, as is the case in the results shown by Toint (1978), where Δ_0 is varied considerably (in one case a different value for Δ_0 is used when comparing different algorithms on the same function). It is possible to duplicate, within a few function evaluations, Toint's result by choosing an appriopriate value for S_{max} in the linesearch. However, for uniformity of testing, S_{max} was set to 10^5 (essentially implying an upper bound of infinity) on all the functions tested. A large upper bound was chosen to avoid biasing any of the results — a smaller bound could possibly influence the performance of some of the algorithms on certain functions.

7.4.4 Comparison of the Algorithms.

The function evaluations and iterations quoted in Table B (Appendix B) are all computed using the assessment criterion of Section 7.1 with a tolerance of 10^{-5}. All the algorithms were run to a maximum of 2000 function evaluations. The algorithms were all still reducing the function (albeit slowly) when they were stopped at 2000 function evaluations. In some cases some of the algorithms failed. By this it is meant that the line-search routine could not find a lower point. This was usually due to the fact that the search direction p was arbitrarily close to zero or almost orthogonal to g. The exception was the algorithm UBFGS, where the failure was due to p not being a descent direction, since the updated

factors had lost their positive-definiteness. In all the cases examined, the matrix Q (see Equation 3.12) had been singular for many iterations before failure of the algorithm, which meant that the sparse quasi-Newton updates did not satisfy the quasi-Newton condition.

Table B compares the function evaluations and iterations required by three sparse quasi-Newton updates (TOINT, BFGS, DFP). The performance of these algorithms is poor in general — especially on the large problems. On a few functions, these methods do slightly better than the finite-difference algorithm. However, their performance is much worse in terms of number of iterations in most of the cases (with the exception of the Trigonometric function). An interesting result holds for the quadratic function (which is a diagonal function). At each iteration the new Hessian approximation is in fact obtained as a finite difference approximation with the finite difference interval being $\max\{|s_i|, \delta\}$ for the ith diagonal element, where δ is given by

$$\delta = \max\{\epsilon||B||, \epsilon\},$$

where ϵ is the machine precision. That is,

$$B_{ii}^{+} = \frac{y_i}{\max\{|s_i|, \delta\}}.$$

Besides these three algorithms, a sparse version of the self-scaled BFGS update was also tested (results not shown here). The best of these four methods is Toint's update, which is a little surprising considering that the dense BFGS update has been found to be superior to the other dense quasi-Newton methods in practice. However, Toint's update is by no means competitive when compared with a modified-Newton algorithm using a finite-difference scheme to generate approximations to the Hessian matrix. In fact, in spite of a proof of superlinear convergence (Toint, 1979a), superlinear convergence was not observed in the test problems within the current implementation of Toint's update (for a discussion, see Section 6).

On moderate size problems ($n = 50$ *to* $n = 100$), the dense quasi-Newton method performs significantly better than the sparse quasi-Newton methods. Thus, there does not seem to be much truth in the speculation that supplying more information (in the form of sparsity) to quasi-Newton methods should cause them to converge faster. An interesting variation is the method UBFGS which uses the C1 algorithm (Gill, Golub, Murray and Saunders, 1974) to update the factors of the BFGS update ignoring all fill-in in the factors. When it does converge its performance is superior to that of the sparse quasi-Newton methods. As noted previously, its failure is due to the loss of positive definiteness of the product of the Cholesky factors. It is remarkable that this method does better than the sparse quasi-Newton methods since the updates obtained by the

UBFGS method do not satisfy the quasi-Newton condition (3.1). However, these good results with UBFGS should be viewed with some caution since the method performed well mostly on functions with diagonally dominant Hessian matrices.

The table clearly shows the superiority of the modified-Newton method over the others. In most of the cases the finite-difference method does better than the others in terms of function evaluations. It does better than all the others in terms of iterations, as Table B shows.

8. Conclusions

All the algorithms were tested using a modified Cholesky factorization. The overall conclusions reached are that sparse quasi-Newton methods perform poorly in general. Superlinear convergence was not observed and the quasi-Newton updates consistently lost the property of positive-definiteness. Furthermore, they require more storage than modified-Newton methods that utilize a finite-difference scheme that exploits sparsity in the Hessian matrix; and they may require a significant amount of time to perform the linear algebra needed to obtain a sparse quasi-Newton update. Modified-Newton methods utilizing a finite-difference scheme that exploit sparsity and symmetry in the Hessian matrix perform extremely well. These Newton-type methods perform very well even when all fill-in is ignored in the modified Cholesky factors (see Thapa, 1980).

A crude implementation of preconditioned conjugate-gradient methods (see Thapa, 1980) that utilize sparse quasi-Newton updates was seen to perform well in comparison to sparse quasi-Newton methods. It is expected that a refined implementation of such methods may prove to be very successful. However, much work remains to be done on such methods.

Surprisingly, UBFGS, a crude implementation of a method that updated the Cholesky factors using algorithm C1 of Gill, et al. (1974) but ignoring all fill-in in the factors, performed well on the set of problems on which the updated factors remained positive definite. However, it should be noted that UBFGS performed well mostly on functions that were diagonally dominant. It would be interesting to develop this method further.

9. Acknowledgements

I would like to thank Dr. Margaret H. Wright and Dr. Philip E. Gill for their invaluable guidance and tremendous amount of help in making this paper possible.

APPENDIX A

TABLE A1
Space Required in *Double Word Lengths* by
the Hessian Matrix and its Cholesky factors
in a Dense Quasi-Newton Method compared to
a Modified-Newton Method using Finite-Differences
and to Sparse Quasi-Newton Methods

PROBLEM	n	QNM	DFD	SPQN
Quadratic	$n = 25$	325	57.75	65
GenRose	$n = 25$	325	92.25	122.5
QOR	$n = 50$	1275	627.5	755
C1GOR	$n = 50$	1275	644.5	820
7Diagonal	$n = 60$	1830	1368	1530
C1dia7	$n = 60$	1830	1391.25	1581.25
Calvar1	$n = 10$	55	52.5	75
Calvar1	$n = 20$	210	112.5	162.5
Calvar1	$n = 30$	465	172.5	250
Calvar1	$n = 50$	1275	292.5	425
Calvar1	$n = 100$	5050	592.5	862.5
Calvar1	$n = 200$	20100	1192.5	1737.5
Calvar1	$n = 300$	45150	1792.5	2612.5
Calvar1	$n = 400$	80200	2392.5	3487.5
Calvar1	$n = 500$	125250	2992.5	4362.5
Calvar1	$n = 1000$	500500	5992.5	8737.5

KEY

QNM - Dense quasi-Newton method.
DFD - Direct method for finite-differencing.
NDFD - New Direct method for finite-differencing.
SPQN - Sparse quasi-Newton update.

TABLE A2
CPU seconds Required for Various Tasks Performed by
the Unconstrained Optimisation Algorithms

PROBLEM	ChaRose $n = 25$	QOR $n = 50$	GOR $n = 50$	7Diagonal $n = 60$	Calvar1 $n = 100$
GENPAT	.08239	.46832	1.7793	3.4964	13.714
GROUP	.00584	.02631	.02628	.03320	.04531
ANALYZ	.00916	.09643	.09652	.27808	.17235
FACTOR	.00502	.10487	.10440	.45444	.04284
SOLVE	.00234	.01619	.01614	.03704	.01351
SFUN	.00283	.00792	.03446	.05731	.13113
DFD	.01388	.07878	.31867	.52722	.94332
TOINT	.01220	.13674	.13699	.51904	.08978
BFGS	.01680	.15063	.15056	.53581	.16203
DFP	.01697	.15114	.15106	.53739	.11993

PROBLEM	Calvar2 $n = 100$	Calvar2 $n = 200$	Calvar2 $n = 300$	Calvar2 $n = 400$	Calvar2 $n = 500$
GENPAT	8.8004	34.752	77.927	139.07	217.55
GROUP	.04537	.09226	.13989	.18905	.23885
ANALYZ	.17182	.67572	1.5377	2.7874	4.41204
FACTOR	.04246	.08589	.12952	.17396	.21801
SOLVE	.01362	.02705	.04094	.05537	.06896
SFUN	.08445	.16839	.25288	.33888	.42442
DFD	.61740	1.2315	1.8444	2.4788	3.1124
TOINT	.08920	.18043	.27287	.36742	.46021
BFGS	.14189	.31575	.48811	.66463	.83609
DFP	.11936	.24161	.36554	.49153	.61596

KEY

GENPAT – Generate the pattern of the Hessian matrix.
GROUP – Form the groups for the Direct Method for finite-differencing.
ANALYZ – Determine the pattern of the LDL^T FACTORS.
FACTOR – Obtain the LDL^T factors.
SOLVE – Solve an $n \times n$ system of equations.
SFUN – Compute the function value and the gradient vector.
DFD – Direct Method for finite-differencing.
TOINT – Toint's sparse quasi-Newton update.
BFGS – Sparse version of the BFGS update.
DFP – Sparse version of the DFP update.

APPENDIX B

TABLE B
Number of *Function Evaluations* and *Iterations* Required

PROBLEM	η	TOINT	BFGS	DFP	UBFGS	DFD	QNM
Calvar1	$\eta = .9$	60,40	73,53	369,273	40,33	36,5	47,42
Start 1	$\eta = .1$	98,46	57,27	189,88	61,30	31,4	54,24
$n = 10$	$\eta = .001$	102,39	65,24	240,97	63,24	28,3	68,24
Calvar1	$\eta = .9$	371,180	*	1222,873	59,46	37,5	83,73
Start 1	$\eta = .1$	401,160	*	362,170	96,48	32,4	90,40
$n = 20$	$\eta = .001$	712,243	*	229,87	152,57	38,4	113,41
Calvar1	$\eta = .9$	895,374	*	446,233	113,85	45,6	118,100
Start 1	$\eta = .1$	895,342	*	315,136	160,77	41,5	125,56
$n = 30$	$\eta = .001$	948,296	*	1181,435	219,85	39,4	162,55
Calvar2	$\eta = .9$	435,229	*	692,412	55,32	15,2	36,22
Start 1	$\eta = .1$	580,282	*	420,209	59,28	15,2	37,19
$n = 30$	$\eta = .001$	672,264	*	697,297	65,28	17,2	39,19
Calvar3	$\eta = .9$	52,36	45,35	65,48	30,27	22,3	35,31
Start 1	$\eta = .1$	63,31	*	114,56	36,19	22,3	37,16
$n = 10$	$\eta = .001$	73,31	46,20	94,42	46,20	26,3	42,16
Calvar3	$\eta = .9$	152,95	*	200,139	54,45	29,4	57,48
Start 1	$\eta = .1$	128,62	107,54	240,118	70,35	24,3	67,30
$n = 20$	$\eta = .001$	151,61	*	147,64	83,36	27,3	77,30
GenRose	$\eta = .9$	48,25	49,29	88,55	134,63	61,13	107,43
Start 3	$\eta = .1$	71,26	139,64	76,34	163,64	65,11	130,40
$n = 25$	$\eta = .001$	104,28	91,27	104,37	178,51	76,11	157,44
ChaRose	$\eta = .9$	61,28	44,21	40,23	68,36	62,13	97,48
Start 5	$\eta = .1$	66,26	45,19	45,17	63,28	45,8	116,44
$n = 10$	$\eta = .001$	92,23	76,20	79,21	91,29	87,12	165,47

KEY

TOINT – Toint' sparse quasi-Newton update.
BFGS – Shanno's sparse BFGS update.
DFP – Sparse version of the DFP update.
UBFGS – Updating the Cholesky factors of the BFGS update ignoring fill-in.
DFD – Direct method for finite-differencing.
QNM – Dense quasi-Newton method.
xx,yy – Number of function evaluations, Number of iterations.
* – Exceeded 2000 function evaluations.
F – Failed to converge.
NR – Not Run.

TABLE B (continued)
Number of *Function Evaluations* and
Iterations Required

PROBLEM	η	TOINT	BFGS	DFP	UBFGS	DFD	QNM
Quadratic	$\eta = .9$	3,2	3,2	3,2	22,21	3,1	31,30
Start 1	$\eta = .1$	4,2	4,2	4,2	45,24	3,1	41,20
$n = 25$	$\eta = .001$	4,2	4,2	4,2	49,24	3,1	41,20
QOR	$\eta = .9$	19,12	27,17	17,11	F	10,1	39,23
Start 1	$\eta = .1$	23,12	23,12	24,13	F	10,1	27,13
$n = 50$	$\eta = .001$	25,12	25,12	27,13	F	10,1	27,13
GOR	$\eta = .9$	74,43	*	56,33	F	37,4	60,30
Start 1	$\eta = .1$	88,36	*	84,37	F	39,4	59,29
$n = 50$	$\eta = .001$	126,38	185,67	118,37	F	44,4	72,29
PSP	$\eta = .9$	6,4	6,4	6,4	4,3	10,1	5,4
Start 1	$\eta = .1$	9,4	9,4	9,4	7,3	10,1	7,3
$n = 50$	$\eta = .001$	9,4	9,4	9,4	7,3	10,1	7,3
C1GOR	$\eta = .9$	422,303	*	541,369	F	63,6	155,153
Start 1	$\eta = .1$	587,266	*	645,303	F	58,5	169,68
$n = 50$	$\eta = .001$	633,241	*	739,278	F	64,5	217,69
7Diagonal	$\eta = .9$	46,39	44,34	60,42	F	37,4	22,20
Start 6	$\eta = .1$	44,20	71,32	59,26	F	37,4	24,9
$n = 60$	$\eta = .001$	72,27	79,31	77,29	187,53	44,4	26,8
C1Dia7	$\eta = .9$	*	*	1871,604	F	72,7	92,56
Start 1	$\eta = .1$	*	*	*	F	58,5	142,50
$n = 60$	$\eta = .001$	*	*	*	F	64,5	278,49
Trig	$\eta = .9$	22,15	23,16	23,16	F	61,6	NR
Start 7	$\eta = .1$	29,15	29,15	32,16	F	62,6	NR
$n = 100$	$\eta = .001$	36,15	36,15	37,16	F	68,6	NR

KEY

TOINT – Toint' sparse quasi-Newton update.
BFGS – Shanno's sparse BFGS update.
DFP – Sparse version of the DFP update.
UBFGS – Updating the Cholesky factors of the BFGS update ignoring fill-in.
DFD – Direct method for finite-differencing.
QNM – Dense quasi-Newton method.
xx,yy – Number of function evaluations, Number of iterations.
* – Exceeded 2000 function evaluations.
F – Failed to converge.
NR – Not Run.

TABLE B (continued)
Number of *Function Evaluations* and
Iterations Required

PROBLEM	η	TOINT	BFGS	DFP	UBFGS	DFD	QNM
Calvar1	$\eta = .9$	*	*	1913,858	220,157	38,5	191,162
Start 1	$\eta = .1$	*	*	1502,606	203,89	39,5	299,149
$n = 50$	$\eta = .001$	*	*	*	395,153	48,5	258,88
Calvar2	$\eta = .9$	739,391	*	1682,872	107,59	15,2	53,28
Start 1	$\eta = .1$	1052,460	*	1967,963	111,54	15,2	54,28
$n = 50$	$\eta = .001$	1119,440	*	*	119,52	17,2	57,28
Calvar3	$\eta = .9$	411,242	*	1138,627	128,93	36,5	114,90
Start 1	$\eta = .1$	556,254	*	871,406	169,84	31,4	131,59
$n = 50$	$\eta = .001$	898,350	*	1365,568	203,85	36,4	154,59
Calvar1	$\eta = .9$	*	*	*	478,322	45,6	372,298
Start 1	$\eta = .1$	*	*	*	662,323	49,6	384,166
$n = 100$	$\eta = .001$	*	*	*	812,316	50,5	476,164
Calvar2	$\eta = .9$	*	*	*	199,100	15,2	103,53
Start 1	$\eta = .1$	*	*	*	202,100	15,2	106,53
$n = 100$	$\eta = .001$	*	*	*	218,97	17,2	107,53
Calvar3	$\eta = .9$	1383,757	*	*	243,178	36,5	187,138
Start 1	$\eta = .1$	1754,721	*	*	306,158	31,4	206,97
$n = 100$	$\eta = .001$	*	*	*	361,154	36,4	258,97
Calvar2	$\eta = .9$	F	NR	*	390,216	8,1	NR
Start 2	$\eta = .1$	*	NR	*	468,235	8,1	NR
$n = 200$	$\eta = .001$	*	NR	*	464,212	9,1	NR
Calvar3	$\eta = .9$	*	NR	*	893,533	113,14	NR
Start 4	$\eta = .1$	*	NR	*	1092,524	135,15	NR
$n = 200$	$\eta = .001$	*	NR	*	1164,512	140,14	NR

KEY

TOINT – Toint' sparse quasi-Newton update.
BFGS – Shanno's sparse BFGS update.
DFP – Sparse version of the DFP update.
UBFGS – Updating the Cholesky factors of the BFGS update ignoring fill-in.
DFD – Direct method for finite-differencing.
QNM – Dense quasi-Newton method.
xx,yy – Number of function evaluations, Number of iterations.
* – Exceeded 2000 function evaluations.
F – Failed to converge.
NR – Not Run.

APPENDIX C

C.1 Test Problems

For the purpose of comparing the algorithms described in Section 7.2, test problems ranging in size from dimension 10 to dimension 500 were used. The problems tested are described below. The following notation will be used in the rest of the paper.

N_G – Number of nonzeros below the diagonal of the Hessian matrix.

N_L – Number of nonzeros below the unit diagonal of the lower-triangular Cholesky factor.

N_{GD} – Number of groups used to obtain the finite difference approximation to the Hessian by the Powell-Toint direct method (Algorithm D in Thapa, 1980).

Test Function 1. GenRose (Gill and Murray, 1979b)

This is a generalization of the well known two-dimensional Rosenbrock function (Rosenbrock, 1960)

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

The Hessian matrix is tridiagonal and

$$N_G = n - 1, \qquad N_L = n - 1,$$
$$N_{GD} = 3.$$

Test Function 2. ChaRose (Toint, 1978)

$$f(x) = 1 + \sum_{i=2}^{25} \left(4\alpha_i(x_{i-1} - x_i^2)^2 + (1 - x_i)^2\right),$$

This is a modification and generalization of the well known two-dimensional Rosenbrock function (Rosenbrock, 1960) where the constants α_i are given by Toint (1978). The Hessian matrix is tridiagonal and

$$N_G = 24, \qquad N_L = 24,$$
$$N_{GD} = 3.$$

Test Function 3. QOR (Toint, 1978)

$$f(x) = \sum_{i=1}^{50} \alpha_i x_i^2 + \sum_{i=1}^{33} \beta_i \left(d_i - \sum_{j \in A(i)} x_j + \sum_{j \in B(i)} x_j\right)^2,$$

where the constants α_i, β_i, d_i and the sets $A(i)$, $B(i)$ are described by Toint (1978). The function is convex with a sparse Hessian matrix and

$$N_G = 115, \qquad N_L = 389,$$
$$N_{GD} = 8.$$

Test Function 4. GOR (Toint, 1978)

$$f(x) = \sum_{i=1}^{50} c_i(x_i) + \sum_{i=1}^{33} b_i(y_i),$$

where

$$c_i(x_i) = \begin{cases} \alpha_i x_i \log_e(1 + x_i), & x_i \geq 0, \\ -\alpha_i x_i \log_e(1 + x_i), & x_i < 0, \end{cases}$$

$$y_i = d_i - \sum_{j \in A(i)} x_j + \sum_{j \in B(i)} x_j$$

and

$$b_i(y_i) = \begin{cases} \beta_i y_i^2 \log_e(1 + y_i), & y_i \geq 0, \\ \beta_i y_i^2, & y_i < 0. \end{cases}$$

The constants α_i, β_i, d_i and the sets $A(i)$, $B(i)$ are the same as for QOR. The function is convex but has discontinuous second derivations. The Hessian matrix has the same sparsity pattern as the Hessian matrix for QOR. The quantities $\bar{N}_G$, $\bar{N}_L$, and N_{GD} are the same as for QOR.

Test Function 5. PSP (Toint, 1978)

$$f(x) = \sum_{i=1}^{50} \alpha_i (x_i - 5)^2 + \sum_{i=1}^{33} \beta_i h_i(y_i),$$

where

$$y_i = d_i - \sum_{j \in A(i)} x_j + \sum_{j \in B(i)} x_j,$$

$$h_i(y_i) = \begin{cases} 1/y_i, & y_i \geq 0.1, \\ 100(0.1 - y_i) + 10, & y_i < 0.1, \end{cases}$$

The constants α_i, β_i, d_i and the sets $A(i)$, $B(i)$ are the same as for QOR. The Hessian matrix has the same sparsity pattern as the Hessian matrix for QOR. The quantities $\bar{N}_G$, $\bar{N}_L$, and N_{GD} are the same as for QOR.

Test Function 6. Quadratic (Gill and Murray)

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

where ρ is a constant. The function is convex with a diagonal Hessian matrix that is ill-conditioned and

$$\bar{N}_G = 0, \qquad \bar{N}_L = 0,$$
$$N_{GD} = 1.$$

The next three functions are similar to those that arise in the numerical solution of optimal control problems. The general continuous problem is of the form

$$\min_{x(t)} J(x(t)) = \int_0^1 f(t, x(t), x'(t))\, dt,$$

subject to the boundary conditions $x(0) = a$, $x(1) = b$. These problems are known as calculus of variations problems. A numerical procedure to solve these problems is to discretize them. The first three functions described below are discretized by expressing $x(t)$ as a linear sum of functions that span the space of piecewise cubic polynomials. This gives rise to a function with a block triangular Hessian matrix.

Test Function 7. Calvar (Gill and Murray, 1973)

$$J(x(t)) = \int_0^1 \left\{ x(t)^2 + x'(t)\tan^{-1} x'(t) - \log(1 + x'(t)^2)^{\frac{1}{2}} \right\} dt,$$

with the boundary conditions $x(0) = 1$, $x(1) = 2$. The Hessian matrix is block tridiagonal and

$$\begin{aligned} N_G &= -5 + 2.5n, & N_L &= N_G, \\ N_{GD} &= 6. \end{aligned}$$

Test Function 8. Calvar2 (Gill and Murray, 1973)

$$J(x(t)) = \int_0^1 \left\{ 100(x(t) - x'(t)^2)^2 + (1 - x'(t))^2 \right\} dt,$$

with the boundary conditions $x(0) = x(1) = 0$. The Hessian matrix is block tridiagonal and

$$\begin{aligned} N_G &= -5 + 2.5n, & N_L &= N_G, \\ N_{GD} &= 6. \end{aligned}$$

Test Function 9. Calvar3 (Gill and Murray, 1973)

$$J(x(t)) = \int_0^1 \left\{ e^{-2x(t)^2}(x'(t)^2 - 1) \right\} dt,$$

with the boundary conditions $x(0) = 1$, $x(1) = 0$. The Hessian matrix is block tridiagonal and

$$\begin{aligned} N_G &= -5 + 2.5n, & N_L &= N_G, \\ N_{GD} &= 6. \end{aligned}$$

Test Function 10. 7-Diagonal (Toint, 1978)

This is a modified version of the function described in Toint's paper.

Let

$$\hat{f}(x) = \sum_{i=1}^{60} |y_i|^{\frac{7}{3}} + \sum_{i=1}^{30} |x_i + x_{i+30}|^{\frac{7}{3}};$$

then

$$f(x) = \hat{f}(x) + \sum_{i=1}^{60} (x_i - 5)^2,$$

where

$$\begin{aligned} y_1 &= -\left(3 - \frac{x_1}{2}\right)x_1 + 2x_2 - 1, \\ y_i &= x_{i-1} - \left(3 - \frac{x_i}{2}\right)x_i + 2x_{i+1} - 1 \qquad i = 2, \ldots, n, \\ y_{60} &= x_{59} - \left(3 - \frac{x_{60}}{2}\right)x_{60} - 1. \end{aligned}$$

The Hessian matrix has the pattern shown in Toint(1978) and

$$\begin{aligned} N_G &= 147, \qquad & N_L &= 957, \\ N_{GD} &= 8. \end{aligned}$$

Test Function 11. Trigonometric (Toint, 1978)

This is a modification of the function suggested by Toint. The modification guarantees that the same minimum is found by all the algorithms if the same starting point is used.

Choose a set of pairs of indices $J = \{(i, j) \mid 1 \leq i \leq n, \text{ and } 1 \leq j \leq i\}$

Let

$$\hat{f}(x) = \sum_{(i,j) \in J} \alpha_{ij} \sin[\beta_i x_j + \beta_j x_i + c_{ij}];$$

then

$$f(x) = \hat{f}(x) + \sum_{i=1}^{n} (x_i - 5)^2,$$

where

$$\begin{aligned} \alpha_{ij} &= 5[1 + \mathrm{mod}(i, 5) + \mathrm{mod}(j, 5)], \\ \beta_i &= 1 + i\frac{1}{10}, \\ c_{ij} &= \frac{(i + j)}{10}. \end{aligned}$$

The Hessian matrix has its pattern defined by the set J and

$$\begin{aligned} N_G &= 268, \qquad & N_L &= 1212, \\ N_{GD} &= 9. \end{aligned}$$

Test Function 12. C1GOR

This is a combination of the Calvar1 (Test Function 7) and GOR (Test Function 4) with $n = 50$. That is,

$$f(x) = f_{\text{GOR}} + f_{\text{Calvar1}}.$$

The Hessian matrix has a pattern that is a combination of the patterns of GOR and the 1st calculus of variations and

$$N_G = 163, \qquad N_L = 393,$$
$$N_{GD} = 9.$$

Test Function 13. C1DIA7

This is a combination of Calvar1 (Test function 7) and the 7-diagonal function (Test function 11) for $n = 60$. That is,

$$f(x) = f_{\text{Calvar1}} + f_{\text{7-Diagonal}}.$$

The Hessian matrix has a pattern that is a combination of the patterns of the 1st calculus of variations and the 7 diagonal function; and

$$N_G = 175, \qquad N_L = 970,$$
$$N_{GD} = 9.$$

C.2 Starting Points

The starting points used are as follows.

Start 1

$$x_0 = (0, 0, \ldots, 0, 0)^T.$$

Start 2

$$x_0 = \left(\frac{1}{n+1}, \frac{2}{n+1}, \ldots, \frac{n}{n+1}\right)^T.$$

Start 3

$$x_0 = (-1.2, 1, -1.2, 1, 1, 1, \ldots)^T.$$

Start 4

$$x_0 = (-1.2, 1, -1.2, 1, -1.2, 1, \ldots)^T.$$

Start 5

$$x_0 = (1, -1, 1, -1, 1, -1, \ldots)^T.$$

Start 6

$$x_0 = (-1, -1, -1, \ldots)^T.$$

Start 7

$$x_0 = (1, 1, 1, \ldots)^T.$$

References

Davidon, W. C. (1959). Variable metric methods for minimization, A. E. C. Res. and Develop. Report ANL-5990, Argonne National Laboratory.

Dembo, R. S., Eisenstat, S. C., and Steihaug, T. (1980). Inexact Newton Methods, Technical Report (Series 47), School of Organization and Management, Yale University.

Dennis, J. E. and Moré, J. J. (1977). Quasi-Newton methods, motivation and theory, *SIAM Review* **19**, pp. 46–89.

Dennis, J. E., Jr. and Schnabel, R. B. (1978). Least change secant update for quasi-Newton methods, Technical Report (TR78–344), Department of Computer Science, Cornell University, Ithaca, New York 14853.

Fletcher, R. and Powell, M. J. D. (1973). A rapidly convergent descent method for minimization, *Computer J.* **6**, pp. 163–168.

Gill, P. E., Golub, G. H., Murray, W. and Saunders, M. (1974). Modifying matrix factorizations, *Math. Comp.* **28**, pp. 504–535.

Gill, P. E. and Murray, W. (1973). "The numerical solution of a problem in the calculus of variations", in *Recent Mathematical Developments in Control* (D. J. Bell, ed.), pp. 97–122, Academic Press, London and New York.

Gill, P. E. and Murray, W. (1979a). "Performance evaluation for nonlinear optimization", in *Performance Evaluation for Numerical Software* (L. Fosdick, ed.), North-Holland.

Gill, P. E. and Murray, W. (1979b). Conjugate-gradient methods for large-scale nonlinear optimization, Report SOL 79-15, Department of Operations Research, Stanford University.

Gill, P. E., Murray, W., Saunders M. A. and Wright M. H. (1979). Two step-length algorithms for numerical optimization, Report SOL 79-25, Department of Operations Research, Stanford University.

Gill, P. E., Murray, W., and Wright, M. H. (1981). *Practical Opimization*, Academic Press, London and New York.

Levenberg, K. (1944). A method for the solution of certain problems in least-squares, *Quart. Appl. Math.* **2**, pp. 164–168.

Marquardt, D. (1963). An algorithm for least-squares estimation of nonlinear

parameters, *SIAM J. Appl. Math.* 11, pp. 431–441.

Marwill, E. S. (1978). Exploiting sparsity in Newton-like methods, PhD Thesis, Cornell University, Ithaca, New York 14853.

Powell, M. J. D. (1979). Quasi-Newton formulae for sparse second derivative matrices, Internal Report DAMTP 1979/NA7, Dept. of Applied Math. and Theoretical Physics, University of Cambridge, England.

Powell, M. J. D. and Toint, Ph. L. (1979). On the Estimation of sparse Hessian matrices, *Siam J. Numerical Analysis* 16, pp. 1060–1073.

Rosenbrock, H. H. (1960). An automatic method for finding the greatest or least value of a function, *Comput. J.* 3, pp. 175–184.

Schubert, L. K. (1970). Modification of a quasi-Newton method for nonlinear equations, *Math. Comp.* 24, pp. 27–30.

Shanno, D. F. (1980). On variable metric methods for sparse Hessians, *Math. Comp.*. 34, pp. 499–514.

Sorensen, D. C. (1980). Newton's Method with a Trust-Region Modification, Technical Report (ANL-80-106), Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439.

Steihaug T. (1980). Quasi-Newton Methods for Large Scale Nonlinear Problems, Technical Report (Series 49), School of Organization and Management, Yale University.

Thapa, M. N. (1980). Optimization of Unconstrained Functions with Sparse Hessian Matrices, PhD Thesis, Dept. of Operations Research, Stanford University, Stanford, California 94305.

Toint, P. L. (1977). On sparse and symmetric matrix updating subject to a linear equation, *Math. Comp.* 31, pp. 954–961.

Toint, P. L. (1978). Some numerical results using a sparse matrix updating formula in unconstrained optimization, *Math. Comp.* 32, pp. 839–851.

Toint, P. L. (1979a). On the superlinear convergence of an algorithm for solving a sparse minimization problem, *Siam J. Numer Anal.* 16, pp. 1036–1045.

Toint, P. L. (1979b). A note about sparsity exploiting quasi-Newton Updates, Technical 79/5, Department of Mathematics, Facultés, Universitaires de Namur, Rampart de la Vierge 8, B-5000 Namur, Belgium.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 81-12	2. GOVT ACCESSION NO. AD-A106028	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) OPTIMIZATION OF UNCONSTRAINED FUNCTIONS WITH SPARSE HESSIAN MATRICES--QUASI-NEWTON METHODS		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Mukund N. Thapa		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0267
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-143
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research - Dept. of the Navy 800 N. Quincy Street Arlington, VA 22217		12. REPORT DATE August 1981 13. NUMBER OF PAGES 35
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

unconstrained
quasi-Newton
sparse
numerical
Cholesky
positive-definite
superlinear

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

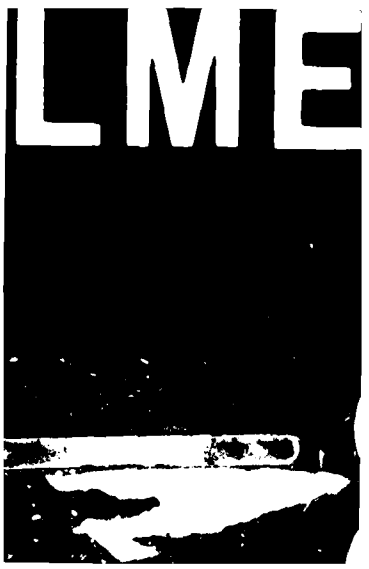
SEE ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SOL 81-12: OPTIMIZATION OF UNCONSTRAINED FUNCTIONS WITH SPARSE HESSIAN MATRICES -- QUASI-NEWTON METHODS; by M.N. Thapa

Newton-type methods and quasi-Newton methods have proven to be very successful in solving dense unconstrained optimization problems. Recently there has been considerable interest in extending these methods to solving large problems when the Hessian matrix has a known a priori sparsity pattern. This paper treats sparse quasi-Newton methods in a uniform fashion and shows the effect of loss of positive-definiteness in generating updates. These sparse quasi-Newton methods coupled with a modified Cholesky factorization to take into account the loss of positive-definiteness when solving the linear systems associated with these methods were tested on a large set of problems. The overall conclusions are that these methods perform poorly in general -- the Hessian matrix becomes indefinite even close to the solution and superlinear convergence is not observed in practice.


LME