SIAM J. NUMER. ANAL.
Vol. 21, No. 4, August 1984

010

NEWTON-TYPE MINIMIZATION VIA THE LANCZOS METHOD*

STEPHEN G. NASH†

Abstract. This paper discusses the use of the linear conjugate-gradient method (developed via the Lanczos method) in the solution of large-scale unconstrained minimization problems. At each iteration of a Newton-type method, the direction of search is defined as the solution of a quadratic subproblem. When the number of variables is very large, this subproblem may be solved using the linear conjugate-gradient method of Hestenes and Stiefel. We show how the equivalent Lanczos characterization of the linear conjugate-gradient method may be exploited to define a modified Newton method which can be applied to problems that do not necessarily have positive-definite Hessian matrices at all points of the region of interest. This derivation also makes it possible to compute a negative-curvature direction at a stationary point.

The idea of a *truncated* Newton method is to perform only a limited number of iterations of the quadratic subproblem. This effectively gives a search direction that interpolates between the steepest-descent direction and the Newton direction. We describe a preconditioned linear conjugate-gradient method that defines a search direction which interpolates between the direction defined by a nonlinear conjugate-gradient-type algorithm and a modified Newton direction.

Key words. unconstrained optimization, truncated-Newton algorithm, modified matrix factorization, Lanczos algorithm, linear conjugate-gradient algorithm

1. Introduction.

In this paper we shall be concerned with methods for the minimization of a bounded, twice continuously-differentiable function $F(x)$. In particular, we are interested in solving problems for which the number of variables n is very large.

When the Hessian matrix is positive definite, the most successful descent methods for minimization are based on finding the direction of search p from the Newton equation

$$G^{(k)}p = -g^{(k)}, \tag{1}$$

where $g^{(k)}$ and $G^{(k)}$ are, respectively, the gradient vector and Hessian matrix of second derivatives of F evaluated at the current iterate $x^{(k)}$. Equation (1) may be solved using the Cholesky factorization

$$G^{(k)} = LDL^T, \tag{2}$$

where L is a unit-lower-triangular matrix with elements l_{ij}, and D a diagonal matrix with jth diagonal element d_j.

When $G^{(k)}$ is a large sparse matrix, the Cholesky factorization can often be computed efficiently using sparse matrix techniques. However, the computation may be impractical due to excessive fill-in. More seriously, it may be impractical to compute $G^{(k)}$ itself. Such problems arise in large-scale linearly constrained and nonlinearly-constrained minimization (see Gill and Murray (1974a), Murtagh and Saunders (1978), (1980), Gill et al. (1980)). In this situation, the Hessian matrices are of the form Z^TWZ where W and Z are large matrices which may be sparse, but whose product Z^TWZ is large and dense.

A well known technique for the solution of large systems of linear equations is the linear conjugate-gradient method of Hestenes and Stiefel (1952). This method can be directly applied to the Newton equations (1) (see, for example, Gill and Murray (1974a, pp. 127–128)). The linear conjugate-gradient algorithm is particularly

* Received by the editors February 17, 1983, and in revised form August 23, 1983.

† Mathematical Sciences Department, The Johns Hopkins University, Baltimore, Maryland 21218. This research was supported by the National Science Foundation under grants MCS-7926009 and ENG77-06761, and by a postgraduate scholarship from the Natural Sciences and Engineering Research Council of Canada.

appropriate when matrix-vector products of the form $G^{(k)}v$ can be computed even though the matrix $G^{(k)}$ or its factorization cannot. The conjugate-gradient algorithm is usually derived as a direct method, in the sense that, theoretically, the solution is found after n iterations or less. However, in practice the algorithm behaves more like an iterative method since it computes a sequence of improving estimates and has the potential of converging in much more or much less than n iterations.

The definition of the search direction given by (1) is only satisfactory if $G^{(k)}$ is positive definite. An indefinite matrix $G^{(k)}$ allows the possibility of p not being a descent direction and this may result in convergence to a nonoptimal point. In the context of minimization, it is preferable not to solve the system (1) if $G^{(k)}$ is indefinite. An alternative is to define p as the solution of a neighboring positive-definite system

$$\bar{G}^{(k)}p = -g^{(k)}.$$

A method for computing $\bar{G}^{(k)}$ when matrix factorizations are feasible is to compute the modified Cholesky factorization of $G^{(k)}$ (Gill and Murray (1974b)). The idea of the Gill–Murray algorithm is to increase the diagonal elements of $G^{(k)}$ during the factorization so that the diagonal elements of D are positive and the subdiagonal elements of L are bounded. If a bound of the form

$$|l_{ij}d_j^{1/2}| \leqq \beta$$

is imposed on the triangular factors, the modified matrix will be of the form

$$\bar{G}^{(k)} = G^{(k)} + \Omega_k(\beta),$$

where Ω_k is a diagonal matrix with nonnegative entries. For an infinite value of β the factorization will be the regular Cholesky factorization with negative d_j replaced by small positive values. As β decreases, the factorization is more likely to be modified. A value of β is computed from the elements of $G^{(k)}$ so that a stable factorization is computed yet the factorization will not be modified if $G^{(k)}$ is sufficiently positive definite.

Modified matrix factorizations are also used within model/trust-region algorithms (Gay (1981)). In this context, they not only safeguard against indefiniteness of the Hessian matrix, but also serve to restrict the step taken when the Newton equations (1) do not accurately model the behavior of the nonlinear function $F(x)$.

In 1975, an important paper of Paige and Saunders demonstrated how the linear conjugate-gradient method may be derived using the Lanczos reduction of a symmetric matrix to tridiagonal form (see Lanczos (1950)). If G is a positive-definite matrix, and if exact arithmetic is used throughout, the Lanczos method determines an orthonormal matrix V and a tridiagonal matrix T such that $V^TGV = T$. If the Cholesky factorization of T is denoted by LDL^T, it is easily verified that the columns of the matrix $V(L^T)^{-1}$ are a set of G-conjugate directions for the minimization problem

$$\min_p \tfrac{1}{2}p^TGp + g^Tp$$

(see (1)). Paige and Saunders derived this formulation of the linear conjugate-gradient algorithm to illustrate when and why the method becomes unstable due to the matrix G becoming indefinite.

Clearly, one possible approach in the indefinite case is to apply the modified Cholesky factorization to T. This has been suggested by Gill and Murray (1974a) and by O'Leary (1982). A conjugate-gradient method based upon the Bunch–Parlett (1971) indefinite factorization has been suggested by Chandra (1978) for those situations in which the solution of an indefinite system is required.

A straightforward application of the modified Cholesky factorization to T requires the a priori computation of β, the bound upon the subdiagonal elements of the lower-triangular factor. The computation of this quantity may be expensive or very difficult if the elements of the Hessian matrix are not known explicitly. In § 2.3 we suggest a modification of the Cholesky factors of T that does not require the computation of β. Further properties of this modified factorization are discussed in §§ 2.4 and 2.5.

In § 3 we describe a modified Newton method for unconstrained minimization based upon the linear conjugate-gradient method. This method is similar to those for small problems in the sense that the method has the ability to detect that $G^{(k)}$ is not sufficiently positive definite and to compute a satisfactory descent direction nevertheless. The application of the traditional linear conjugate-gradient algorithm to equation (1) is not numerically stable for an indefinite system. Moreover, even though the curvature information computed by the traditional algorithm can detect indefiniteness, it is not a reliable indicator of near-singular systems.

Recently, Dembo and Steihaug (1983) have suggested an "inexact" or "truncated-Newton" method in which the linear conjugate-gradient method is deliberately terminated after q iterations—even though the solution of (1) may not have been determined. The last iterate of the conjugate-gradient iteration is then used as the direction of search p. If a single conjugate-gradient iteration is used, p will be the steepest-descent direction $-g^{(k)}$. If the iterations are not terminated early, and if exact arithmetic is used throughout, p would be the Newton direction defined by (1). Thus the algorithm computes a vector that interpolates between the steepest-descent direction and the Newton direction.

As the solution is approached, their idea is to make the truncated-Newton direction closer to the Newton direction. Consequently, they preserve the excellent asymptotic rate of convergence properties of that method. In order to ensure that the truncated-Newton direction is close to the Newton direction, it may be necessary to do many linear conjugate-gradient steps. The preconditioning techniques described in § 3 show how a good direction may be obtained in a smaller number of linear conjugate-gradient steps.

A disadvantage of a straightforward application of the linear conjugate-gradient algorithm even when $G^{(k)}$ is positive definite is that p interpolates between the steepest-descent direction and Newton's direction. Since even remote from the solution the steepest-descent direction is often a very poor search direction, the resulting algorithm will be inefficient or always require many linear conjugate-gradient steps, unless the Hessian is approximately equal to a multiple of the identity matrix. $F(x)$ could be minimized using a more efficient nonlinear conjugate-gradient algorithm.

To enable the truncated solution to interpolate between a nonlinear conjugate-gradient direction and the Newton direction, it is only necessary to start the process with the appropriate vector. The linear conjugate-gradient method can be implemented using an arbitrary initial vector (Beale (1972)). Such a strategy has two drawbacks. It complicates the case when the Hessian is indefinite, since the corresponding Lanczos method does not produce a tridiagonal projected Hessian T, making modified factorization techniques more difficult. In addition, commencing the linear conjugate-gradient procedure with the right-hand side vector has desirable numerical properties.

Although we use the Lanczos algorithm primarily for stability reasons when the Hessian is indefinite, an added advantage is that it can be started with an arbitrary initial vector without altering the structure of the method. However, like the conjugate-gradient algorithm, there are considerable numerical benefits from starting the Lanczos

algorithm with the right-hand side vector. In order to commence the Lanczos algorithm with the right-hand side vector and also have the property that the search direction interpolates between the nonlinear conjugate-gradient and the Newton directions, it is necessary to transform the equation. The transformation is arranged so that starting with the right-hand-side vector in the transformed variables is equivalent to starting with the nonlinear conjugate-gradient direction in the original variables. An important additional benefit is that the transformed system is designed to be of the type easily solved by the conjugate-gradient algorithm.

2. Solving equations using Lanczos tridiagonalization. In this section we shall consider methods for the solution of linear systems of the form

$$Gp = -g, \tag{3}$$

where G is an $n \times n$ symmetric matrix. We shall use $\{p_q\}$ to denote members of an iterative sequence intended to solve (3). It will be assumed that all the algorithms discussed are performed in exact arithmetic.

The following derivation of the linear conjugate-gradient algorithm from the Lanczos algorithm is essentially due to Paige and Saunders (1975). The derivation is given here in order to introduce ideas and notation required in later sections.

2.1. Positive-definite systems. The conjugate-gradient algorithm can be derived by finding iterates that minimize the quadratic function $Q(p) = \frac{1}{2}p^T Gp + g^T p$.

Let p_q be the qth approximation to the minimum of $Q(p)$ and let $v_1, v_2, \cdots, v_q$ be q linearly independent vectors that span a subspace $\mathcal{V}_q$. The minimum of $Q(p)$ may be computed by minimizing $Q(p)$ over an expanding sequence of linear manifolds that eventually contains R^n. If V_q denotes the matrix with columns $v_1, v_2, \cdots, v_q$ then the minimum of $Q(p)$ over the manifold $p_q + \mathcal{V}_q$ is given by the solution of the problem

$$\underset{w \in R^q}{\text{minimize}}\ Q(p_q + V_q w).$$

If $p_q + V_q w$ is substituted into the quadratic function we find that the optimal w minimizes the function

$$\tfrac{1}{2}w^T V_q^T G V_q w + w^T V_q^T r_q,$$

where $r_q = \nabla Q(p_q) = Gp_q + g$. This quadratic function has a minimum at the point $-(V_q^T G V_q)^{-1} V_q^T r_q$ and consequently, the required minimum over the subspace is given by

$$p_{q+1} = p_q - V_q (V_q^T G V_q)^{-1} V_q^T r_q.$$

Note that r_{q+1}, the gradient of $Q(p)$ at p_{q+1}, is orthogonal to the columns of V_q since

$$\begin{aligned} V_q^T r_{q+1} &= V_q^T (G p_{q+1} + g) \\ &= -V_q^T G V_q (V_q^T G V_q)^{-1} V_q^T r_q + V_q^T r_q \\ &= 0. \end{aligned}$$

The definition of p_{q+1} as a minimum over the manifold $p_q + \mathcal{V}_q$ has special significance if each previous iterate p_j is obtained as the minimum over $p_{j-1} + \mathcal{V}_{j-1}$. In this case r_q will be orthogonal to all the columns of V_q except the last, and the minimum on the

subspace $\mathcal{V}_q$ is given by

$$\begin{aligned} p_{q+1} &= p_q - V_q(V_q^T G V_q)^{-1} V_q^T r_q \\ &= p_q + \gamma V_q (V_q^T G V_q)^{-1} e_q, \end{aligned} \tag{4}$$

where $\gamma = -r_q^T v_q$ and e_q is the qth column of the $q \times q$ identity matrix.

Suppose that the columns of V_q are defined by the Lanczos recurrence relations (Lanczos (1950)). In this case we start with some vector v_1, $v_1^T v_1 = 1$, and form

$$\beta_{j+1} v_{j+1} = G v_j - \alpha_j v_j - \beta_j v_{j-1}, \qquad \alpha_j = v_j^T G v_j, \tag{5}$$

where $v_0 = 0$, and β_{j+1} ($\beta_{j+1} \geqq 0$) is chosen so that $\|v_{j+1}\|_2 = 1$. After the qth step

$$G V_q = V_q T_q + \beta_{q+1} v_{q+1} e_q^T, \tag{6}$$

where

$$T_q = \begin{pmatrix} \alpha_1 & \beta_2 & & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & & \\ & \beta_3 & \alpha_3 & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & \beta_q \\ & & & & & \beta_q & \alpha_q \end{pmatrix},$$

$$V_q^T V_q = I_q \quad \text{and} \quad V_q^T v_{q+1} = 0.$$

The process will be terminated at the first zero β_j, so that in general we assume that β_j is nonzero for $j = 1, 2, \cdots, q$. In this case $V_q^T G V_q = T_q$ and (4) becomes

$$p_{q+1} = p_q + \gamma V_q T_q^{-1} e_q = p_q + \gamma V_q (L_q^T)^{-1} D_q^{-1} L_q^{-1} e_q,$$

where L_q and D_q are the Cholesky factors of T_q. Since L_q has unit diagonal elements, $L_q^{-1} e_q = e_q$ and

$$p_{q+1} = p_q + (\gamma / d_q) V_q (L_q^T)^{-1} e_q = p_q + \sigma_q u_q, \tag{7}$$

where $\sigma_q = -r_q^T v_q / d_q$ and u_q is given by the qth column of the matrix

$$U_q = V_q (L_q^T)^{-1}.$$

Paige and Saunders (1975) show that the columns of U_q can be computed from the recurrence relations

$$u_1 = v_1, \qquad u_q = v_q - l_q u_{q-1},$$

where l_q is the $(q-1)$st subdiagonal element of the lower bidiagonal matrix L_q.

When the vector v_1 used to start the Lanczos process is chosen as a multiple of the right-hand side vector $-g$ and when p_1 is zero, this algorithm is mathematically equivalent to the Hestenes–Stiefel conjugate-gradient algorithm. The derivation given here emphasizes that a whole class of conjugate-direction methods can be generated from different choices for v_1.

Rounding error seriously impairs the performance of Lanczos tridiagonalization by causing a loss of orthogonality in the vectors $\{v_j\}$. The effects of this error are reduced if the starting vector v_1 is taken to be a multiple of $-g$ (Paige and Saunders (1975)). From (6) we have

$$G V_q y = V_q T_q y + \beta_{q+1} v_{q+1} e_q^T y$$

to working precision, and since $p_q = V_q y$ and $T_q y = \beta_1 e_1$,

$$Gp_q = \beta_1 V_q e_1 + \beta_{q+1} v_{q+1} e_q^T y = -g + \beta_{q+1} v_{q+1} e_q^T y. \tag{8}$$

This expression does not depend on the orthogonality of the Lanczos vectors, and indicates that p_q will solve a problem with right-hand side that differs from the true $-g$ by $\beta_{q+1} v_{q+1} e_q^T y$, a quantity that will ultimately be sufficiently small. A relationship analogous to (8) does not hold for arbitrary v_1. Thus we should expect that if a large number of iterations are used to solve the quadratic subproblem, the solution will be more accurate if v_1 is parallel to g rather than an arbitrary vector.

In the light of these observations, we shall always use the term Lanczos iteration to refer to the recurrence relations (5) with v_1 defined as a multiple of $-g$.

2.2. Indefinite systems. When the matrix G is indefinite, Paige and Saunders note that the conjugate-gradient method is unstable and propose a modified Lanczos method based on the LQ factorization of T_q rather than the Cholesky factorization. In the context of minimization however, even the exact solution of an indefinite system is of little practical value since the resulting direction of search is likely to be a nondescent direction. As in the case of modified Newton methods for the factorization of $G^{(k)}$, we can make better use of the solution of a neighboring positive-definite system.

The proposed method is based on the following theorem. We shall assume that the Lanczos iteration, when applied with exact arithmetic to a symmetric matrix G, terminates at iteration s ($s \leqq n$), i.e., β_{s+1} vanishes. As in the last section, we shall use T_q and V_q to denote the $q \times q$ and $n \times q$ matrices associated with the qth stage of the Lanczos iteration.

THEOREM 1. *Let $E_s = \operatorname{diag}(e_{11}, e_{22}, \cdots, e_{ss})$ be a diagonal matrix with nonnegative entries and let E_q denote the $q \times q$ principal submatrix of E_s. If the matrix $T_q + E_q$ is positive definite with Cholesky factors L_q and D_q, the iteration*

$$\bar{p}_1 = 0, \bar{p}_{q+1} = \bar{p}_q + \sigma_q V_q L_q^{-T} e_q,$$

where $\sigma_q = -v_q^T r_q / d_q$, solves the linear system

$$(G + V_s E_s V_s^T)\bar{p} = -g.$$

Proof. Let $\bar{G}$ denote the matrix $G + V_s E_s V_s^T$. Using the orthogonality of the Lanczos vectors

$$V_q^T \bar{G} V_q = V_q^T G V_q + E_q = T_q + E_q = L_q D_q L_q^T. \tag{9}$$

We can apply the ideas of § 2.1 to generate the sequence $\{\bar{p}_j\}$ defined by the recurrence relations

$$\begin{aligned} \hat{r}_q &= \bar{G}\bar{p}_q + g, \\ \bar{p}_{q+1} &= \bar{p}_q - V_q (V_q^T \bar{G} V_q)^{-1} V_q^T \hat{r}_q, \end{aligned} \tag{10}$$

that will solve the linear system $\bar{G}\bar{p} = -g$.

Substituting the expression (9) for $V_q^T \bar{G} V_q$ in (10) and using a similar analysis to that used to obtain (7), we find

$$\bar{p}_{q+1} = \bar{p}_q + \sigma_q V_q L_q^{-T} e_q,$$

where $\sigma_q = -v_q^T \hat{r}_q / d_q$. The scalar $v_q^T \hat{r}_q$ is computed from r_q using

$$V_q^T \hat{r}_q = v_q^T (\bar{G}\bar{p}_q + g) = v_q^T (G\bar{p}_q + g) + v_q^T V_q E_q V_q^T \bar{p}_q = v_q^T r_q + e_{qq} v_q^T \bar{p}_q.$$

As $\bar{p}_q$ lies in the space spanned by $\{v_1, \cdots, v_{q-1}\}$, and V_q is orthogonal, the second term on the right-hand side vanishes. This proves the theorem. □

COROLLARY 1. *The vector $\bar{p}$ obtained from the recurrence relations defined in Theorem 1 satisfies the positive-definite system*

$$(G+VEV^T)\bar{p}=-g,$$

where $\|VEV^T\|=\|E\|$.

Proof. If s is equal to n the corollary follows trivially. If $s<n$ then V will be the matrix $(V_s\bar{V})$, where $\bar{V}$ is the orthogonal complement of V_s. The remaining $n-s$ diagonal elements of E are arbitrary and may be chosen so that $A+VEV^T$ is positive definite. □

2.3. Computing a modified factorization of a tridiagonal matrix. At the first stage of the Lanczos process we need to find the Cholesky factorization of the 1×1 matrix given by

$$T_1=(\alpha_1).$$

If G is not positive definite, α_1 may be negative or zero, and it is replaced by

$$\bar{\alpha}_1=\max\{\alpha_1,\delta\},$$

where δ is a preassigned small positive constant. This is an allowable diagonal modification to T_1 since

$$\bar{\alpha}_1=\alpha_1+\rho_1,$$

where $\rho_1=\max\{0,\delta-\alpha_1\}$. The introduction of the constant δ is necessary to bound the factorization away from singularity. Usually δ will be a multiple of the relative machine precision.

At the second stage of the algorithm we need the modified factorization of the matrix

$$\begin{pmatrix}\bar{\alpha}_1 & \beta_2\\ \beta_2 & \alpha_2\end{pmatrix}. \tag{11}$$

Let the modified Cholesky factors of this matrix be written as

$$\begin{pmatrix}1 & 0\\ l_2 & 1\end{pmatrix}\begin{pmatrix}d_1 & 0\\ 0 & d_2\end{pmatrix}\begin{pmatrix}1 & l_2\\ 0 & 1\end{pmatrix}.$$

Straightforward application of the Cholesky factorization to (11) gives $d_1=\bar{\alpha}_1$, $d_2=\max\{\delta,\phi\}$, and $l_2=\beta_2/\bar{\alpha}_1$, where $\phi=\alpha_2-\beta_2^2/\bar{\alpha}_1$. If ϕ is negative, the matrix is indefinite and a diagonal correction must be made in order to ensure sufficient positive definiteness. The smallest modification to the $(2,2)$ element of (11) that maintains positive definiteness results from redefining d_2 as δ. This modification adds the quantity $-\phi+\delta$ to the second diagonal element of (11). Unfortunately, this algorithm has a serious disadvantage, as the following example will illustrate.

Consider the Cholesky factorization of the matrix

$$T=\begin{pmatrix}\delta & 1\\ 1 & 1\end{pmatrix}. \tag{12}$$

In this case, ϕ is the large negative quantity $1-1/\delta$ and the diagonal d_2 is set to δ. This gives the factorization

$$\begin{pmatrix}1 & 0\\ \delta^{-1} & 1\end{pmatrix}\begin{pmatrix}\delta & 0\\ 0 & \delta\end{pmatrix}\begin{pmatrix}1 & \delta^{-1}\\ 0 & 1\end{pmatrix}=\begin{pmatrix}\delta & 1\\ 1 & \delta+\delta^{-1}\end{pmatrix}.$$

Note that, although we have made the smallest possible modification to d_2, we have produced a very large diagonal correction and the modified matrix is in no way "close" to the original. This has occurred because the lower-triangular element has been allowed to become large.

This problem does not arise when using the Gill–Murray modified Cholesky factorization since the diagonals are adjusted so as to give lower-triangular elements that are bounded in magnitude by an a priori bound. However, in order to compute this bound it is necessary to determine an accurate bound on $\|G\|$ which may not be possible or convenient if G is large and not stored explicitly.

We propose an alternative method involving modified factorizations of 2×2 diagonal blocks of the matrix T_q. Initially, we factor

$$\begin{pmatrix} \bar{\alpha}_1 & \beta_2 \\ \beta_2 & \alpha_2 \end{pmatrix} + \begin{pmatrix} \sigma_1 & 0 \\ 0 & \rho_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ l_2 & 1 \end{pmatrix} \begin{pmatrix} \bar{d}_1 & 0 \\ 0 & d_2 \end{pmatrix} \begin{pmatrix} 1 & l_2 \\ 0 & 1 \end{pmatrix},$$

so that the quantity $\sigma_1+\rho_2$ is minimized. (We use a different notation for the elements of the diagonal modification because ρ_2 and d_2 may be modified again at the next stage of the factorization.) Thus if we define the quantity

$$\bar{\phi}(\sigma_1, \rho_2) = \alpha_2 + \rho_2 - \beta_2^2/(\bar{\alpha}_1 + \sigma_1),$$

we need to solve the problem

$$\begin{aligned} &\text{minimize } \sigma_1+\rho_2 \\ &\text{subject to } \sigma_1 \geqq 0,\ \rho_2 \geqq 0,\ \bar{\phi}(\sigma_1, \rho_2) \geqq \delta. \end{aligned}$$

Note that $\bar{\phi}(0,0)$ is equal to ϕ, the quantity computed during the regular Cholesky factorization. If $\bar{\phi}(0,0) \geqq \delta$, $\sigma_1 = \rho_2 = 0$. Otherwise, we compute

$$\sigma_1 = |\beta_2| - \bar{\alpha}_1 \quad \text{and} \quad \rho_2 = \delta - \alpha_2 + |\beta_2|,$$

which are the optimal values of σ_1 and ρ_2 ignoring the nonnegativity constraints. If either correction is negative, we use

$$\sigma_1 = 0 \quad \text{and} \quad \rho_2 = \delta - \alpha_2 + \beta_2^2/\bar{\alpha}_1,$$

or

$$\sigma_1 = \beta_2^2/(\alpha_2 - \delta) - \bar{\alpha}_1 \quad \text{and} \quad \rho_2 = 0,$$

choosing the feasible pair that minimizes $\sigma_1+\rho_2$. Since $\delta - \alpha_2 + \beta_2^2/\bar{\alpha}_1 = \delta - \phi > 0$, at least the first pair will be feasible.

When this modified Cholesky factorization is used on the matrix (12), the factors are given by

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \delta \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1+\delta \end{pmatrix},$$

which is optimally "close" to the original matrix under the above set of constraints.

The first column of the Cholesky factorization is unaffected by later computations and consequently l_2 is the required subdiagonal element of L with

$$e_{11} = \rho_1 + \sigma_1.$$

The next stage of the reduction involves the matrix

$$\begin{pmatrix} \bar{\alpha}_2 & \beta_3 \\ \beta_3 & \alpha_3 \end{pmatrix},$$

where $\bar{\alpha}_2 = \alpha_2 + \rho_2$ and by construction, $\bar{\alpha}_2 \geqq \delta$. This matrix is identical in structure to (11) and so the process can be continued to find the modified factorization of T_2, $T_3, \cdots$ etc.

2.4. Bounds on the modified factorization. An advantage of the Gill–Murray modified Cholesky factorization is that it is possible to bound the diagonal modification to the matrix. To do this, their algorithm requires accurate a priori bounds on the elements of the matrix $G^{(k)}$. In our case, the tridiagonal matrix T_q is being factored as it is generated, and the matrix $G^{(k)}$ may not be available. It is still possible, though, to bound the norm of the modification.

THEOREM 2. *Let T be a symmetric tridiagonal matrix, with principal $i \times i$ submatrix T_i. Assume that a modified Cholesky factorization*

$$T_i + E_i = L_i D_i L_i^T$$

is computed using the algorithm described above. If

$$\gamma_i = \max_{j \leqq i} \{|\alpha_j|\}, \qquad \zeta_i = \max_{j \leqq i} \{|\beta_j|\},$$

where $\{\alpha_j\}$ and $\{\beta_j\}$ are the diagonal and subdiagonal elements of T, respectively, and if δ is a positive tolerance for zero, then

$$\|E_i\| \leqq 3(\delta + \gamma_i + \zeta_i).$$

Proof. Initially $T_1 = (\alpha_1)$. If $\alpha_1 > \delta$ then $E_1 = (0)$; otherwise $E_1 = (\delta - \alpha_1)$. In either case $\|E_1\| \leqq \delta + \gamma_1$. (The 2-norm is being used.)

Next, T_2 is the matrix (11). If $\phi = \bar{\phi}(0, 0) \geqq \delta$ then no modification is necessary. Otherwise $\bar{\phi}(\sigma, \rho) = \delta$, and one of the following is used:

a) $\sigma_a = |\beta_2| - \bar{\alpha}_1$, $\rho_a = |\beta_2| - \tilde{\alpha}_2$,

b) $\sigma_b = 0$, $\rho_b = \beta_2^2/\bar{\alpha}_1 - \tilde{\alpha}_2 = \delta - \phi > 0$,

c) $\rho_c = 0$, $\sigma_c = \beta_2^2/\tilde{\alpha}_2 - \bar{\alpha}_1 = \rho_b \cdot (\bar{\alpha}_1/\tilde{\alpha}_2)$,

where $\tilde{\alpha}_2 = \alpha_2 - \delta$. Recall that b) will always be feasible. If a) is used, then $|\sigma| \leqq \delta + \gamma_1 + \zeta_2$, $|\rho| \leqq \delta + \gamma_2 + \zeta_2$. Note that a) is feasible only when $\bar{\alpha}_1 \leqq |\beta_2|$ and $\tilde{\alpha}_2 \leqq |\beta_2|$.

If $\phi < \delta$ and a) is infeasible, then either:

i) $|\beta_2| < \bar{\alpha}_1$: This implies $0 \leqq \rho_b \leqq |\beta_2| - \tilde{\alpha}_2$.

ii) $|\beta_2| < \tilde{\alpha}_2$: This implies $0 \leqq (\bar{\alpha}_1/\tilde{\alpha}_2) \cdot \rho_b = \sigma_c \leqq |\beta_2| - \bar{\alpha}_1$ (since $\bar{\alpha}_1 > 0$ by construction, $0 \leqq |\beta_2| < \tilde{\alpha}_2$ by assumption).

Thus, regardless of whether b) or c) is used, $|\sigma| \leqq \delta + \gamma_2 + \zeta_2$ and $|\rho| \leqq \delta + \gamma_2 + \zeta_2$. This shows that

$$(E_2)_{11} \leqq 2\delta + 2\gamma_2 + \zeta_2, \qquad (E_2)_{22} \leqq \delta + \gamma_2 + \zeta_2.$$

At later stages, only the lower 2×2 block of T is modified. Cascading the above estimates gives the desired result. □

This result is nearly optimal. If $T = -\alpha I$, then $\|E_i\|_2 = \delta + \alpha = \delta + \gamma_i + \zeta_i$. This is only a factor of 3 smaller than the result in the theorem for an algorithm that is clearly local in character. This local character ensures that the low-storage properties of the linear conjugate-gradient algorithm carry over to the modified Lanczos algorithm.

2.5. Computing a direction of negative curvature. At a stationary point, the equation (1) will have a right-hand side equal to zero, and the Lanczos algorithm will compute a zero solution to this equation. If G is positive definite, we have found a local minimum x^*. If G is indefinite, further progress can be made by moving along a direction of negative curvature.

Suppose that $\|g\|$ is small, i.e. $\|g\| \leqq \varepsilon$, some prespecified tolerance for zero depending on the machine precision and the size of the problem. We wish to determine if G is indefinite, and if so, compute p such that $p^T G p < 0$. We choose v_1 randomly, $\|v_1\| = 1$, as the initial vector for the Lanczos process. The Lanczos iteration is performed as long as T_j is positive semi-definite. (Parlett (1980) reports that the extreme eigenvalues of T_j will be good approximations to the extreme eigenvalues of G after $2n^{1/2}$ iterations; if G is indefinite, one of its extreme eigenvalues will be negative.) The tolerance δ should be set to zero.

Assume that T_{q+1} is the first indefinite matrix and (for the moment) that $\beta_{q+1} \neq 0$. Then $T_{q+1} = L_{q+1} D_{q+1} L_{q+1}^T$, where

$$L_{q+1} = \begin{pmatrix} L_q & 0 \\ 0 & 1 \end{pmatrix}, \qquad D_{q+1} = \begin{pmatrix} D_q & b_{q+1} \\ b_{q+1}^T & \alpha_{q+1} \end{pmatrix},$$

and $b_{q+1}^T = (0, \cdots, 0, \beta_{q+1})$. Only the last diagonal element of D_q can be zero. To determine a direction of negative curvature, we perform an orthogonal spectral decomposition of the bottom 2×2 block of D_{q+1}:

$$\begin{pmatrix} d_q & \beta_{q+1} \\ \beta_{q+1} & \alpha_{q+1} \end{pmatrix} = Q \begin{pmatrix} \lambda_q & 0 \\ 0 & -\lambda_{q+1} \end{pmatrix} Q^T, \quad Q = \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{pmatrix}, \quad Q^T Q = I,$$

with $\lambda_{q+1} > 0$. This is possible because T_{q+1} has exactly one negative eigenvalue. Let $p = u_{q+1}$, where

$$U_{q+1} = (u_1 | \cdots | u_{q+1}) = V_{q+1} \tilde{L}_{q+1}^{-T}, \qquad \tilde{L}_{q+1} = L_{q+1} \begin{pmatrix} I & 0 \\ 0 & Q \end{pmatrix},$$

$u_{q+1} = q_{12} u_1 + q_{22} v_{q+1}$. This is nearly the same as the formula used to compute the linear conjugate-gradient search direction, so that no additional vector storage is required to implement this idea. If $e_{q+1}^T = (0, \cdots, 0, 1)$,

$$\begin{aligned} p^T G p &= e_{q+1}^T U_{q+1}^T G U_{q+1} e_{q+1} \\ &= e_{q+1}^T \tilde{L}_{q+1}^{-1} V_{q+1}^T G V_{q+1} \tilde{L}_{q+1}^{-T} e_{q+1} = -\lambda_{q+1} < 0. \end{aligned}$$

Thus, p is a direction of negative curvature. It should be noted that, since the Lanczos algorithm seeks out the most negative eigenvalue of G, $p^T G p / \|p\|^2$ will be close to its minimum value. Thus, near $x^{(k)}$, $F(x)$ will decrease rapidly along the direction p.

If $\beta_{q+1} = 0$, the situation is more complex. If T_q is positive definite and v_1 contains components of all eigenvectors of G, then we are at a local minimum. This last condition is difficult to verify. To guarantee that G is positive definite, we require a more complex procedure. For example, we could carry out the Lanczos procedure using a series of initial vectors. Choose v_1^1 as above; choose v_1^2 orthogonal to $\{v_1^1, Gv_1^1, \cdots\}$, and so forth until R^n is spanned, or until an indefinite T_q is encountered. However, this procedure is impractical for large problems.

If G is indefinite, then even if the initial vector v_1 contains no component in the negative eigenspace, any rounding error would amost certainly introduce one, allowing a negative eigenvalue to develop in T_q as desired. This justifies using a single starting vector when seeking a direction of negative curvature. If G is only positive semidefinite,

then it is not possible to determine a direction of negative curvature for the quadratic approximation.

2.6. Preconditioning. When exact arithmetic is used, the number of iterations required to solve $Gp=-g$ using the conjugate-gradient method (or the equivalent Lanczos scheme) is equal to the number of distinct eigenvalues of G (Luenberger (1973, pp. 176–178)). Therefore the conjugate-gradient method will be more efficient when the original system is replaced by an equivalent system in which the coefficient matrix has many clustered eigenvalues. The purpose of preconditioning is to construct a transformation to have this effect (Axelsson (1977)).

Let C be a symmetric, positive-definite matrix. The solution of $Gp=-q$ can be found by solving the system

$$C^{-1/2}GC^{-1/2}y=-C^{-1/2}g,$$

and forming $p=C^{-1/2}y$. Let R denote the matrix $C^{-1/2}GC^{-1/2}$; then R has the same eigenvalues as $C^{-1}G$, since $C^{-1/2}RC^{1/2}=C^{-1}G$. The idea is to choose C so that many of the eigenvalues of $C^{-1}G$ are close to unity. The matrix C is known as the preconditioning matrix.

Ignoring the finite-termination peroperty, the unpreconditioned method converges linearly, with rate $(\kappa^{1/2}-1)/(\kappa^{1/2}+1)$, where κ is the condition number of G. This rate can be improved via preconditioning if $C^{-1}G$ has a smaller condition number than G.

Given a preconditioning matrix C, we can apply the Lanczos algorithm to the transformed system without forming R and without finding the square root of the matrix C. In practice, C will often not be explicitly available. It will only be available as an operator, and all that will be possible is to solve systems of linear equations with coefficient matrix C.

The recurrence relations analogous to (5) for the transformed system are

$$\beta_{j+1}v_{j+1}=C^{-1}Gv_j-\alpha_jv_j-\beta_jv_{j-1}, \qquad \alpha_j=v_j^TGv_j,$$

where $v_0=0$ and $\beta_1v_1=C^{-1}g$, and the vectors v_i are normalized so that $\|v_i\|_C=1$ ($\|v\|_C^2=v^TCv$). After the qth step we have

$$GV_q=CV_qT_q+\beta_{q+1}Cv_{q+1}e_q^T.$$

Note that the matrices C, T_q, and V_q satisfy the relationships

$$V_q^TCV_q=I_q \quad \text{and} \quad V_q^TGV_q=T_q.$$

This preconditioned Lanczos algorithm allows us to solve the system of equations (3) in the same way as in § 2.1. The crucial fact in the derivation in that section is that the matrix V_q transforms the matrix G to tridiagonal form. This is still true for the preconditioned algorithm.

We shall discuss the choice of preconditioning matrix in § 3.2.

3. Methods for unconstrained minimization. In this section we shall discuss the implication of using variants of the linear conjugate-gradient method to solve the unconstrained minimization problem. All the methods discussed are descent methods of the following general form.

ALGORITHM D (descent method for unconstrained minimization). Let $x^{(k)}$ be the current estimate of a minimum of $F(x)$.

D1. (test for convergence). If the conditions for convergence are satisfied, the algorithm terminates with $x^{(k)}$ as the solution.

D2. (compute a search direction). Compute an n-vector p_k, the direction of search.
D3. (compute a step length). Compute a positive scalar α_k, the step length, such that $F(x^{(k)}+\alpha_k p_k) < F(x^{(k)})$.
D4. (update the estimate of the minimum) Set $x^{(k+1)} \leftarrow x^{(k)}+\alpha_k p_k$, $k \leftarrow k+1$, and go back to step D1.

Before we discuss the application of the linear conjugate-gradient method, we need to introduce some background material.

3.1. Nonlinear conjugate-gradient-type methods. Until recently, conjugate-gradient-type algorithms were the only class of algorithms suitable for problems in which the Hessian cannot be sorted. All these algorithms define the search direction as a linear combination of the gradient vector and a subset of the previous search directions. Many of these method are members of the class of limited-memory quasi-Newton methods. These methods derive their name from the class of quasi-Newton methods for unconstrained optimization. The direction of search for a quasi-Newton method can be defined as

$$p_k = -H_k g^{(k)},$$

where H_k is an $n \times n$ matrix which is stored explicitly and is an approximation to the inverse Hessian matrix. After computing the change in x, $s_k \equiv x^{(k+1)} - x^{(k)}$ and the corresponding change in the gradient vector, $y_k \equiv g^{(k+1)} - g^{(k)}$, the approximate Hessian is updated to include the new curvature information obtained during the kth iteration. For example, the BFGS formula for H_{k+1} is given by

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)s_k^T + s_k(s_k - H_k y_k)^T}{y_k^T s_k} - \frac{(s_k - H_k y_k)^T y_k}{(y_k^T s_k)^2} s_k s_k^T \tag{13}$$

(see Dennis and Moré (1977)). If exact linear searches are made and F is a positive-definite quadratic function, the matrix H_{k+1} satisfies the so-called quasi-Newton condition for k pairs of vectors $\{s_j, y_j\}$, i.e.,

$$s_j = H_{k+1} y_j, \qquad j = 1, 2, \cdots, k.$$

In this case, if the Hessian of F is G, then $Gs_j = y_j$ and consequently

$$s_j = HGs_j,$$

and the matrix HG has k unit eigenvalues with eigenvectors $\{s_j\}$.

Limited-memory quasi-Newton methods define the direction of search as $-Hg^{(k)}$. The matrix H is never stored explicitly; rather, only the vectors $\{s_j, y_j\}$ that define the rank-one corrections are retained (see Shanno (1978), Gill and Murray (1979) and Nocedal (1980)).

Different methods can be developed by varying the number of vectors $\{s_j, y_j\}$ stored and the choice of quasi-Newton updating formula. For example, if we define the matrix H to be the identity matrix updated by one iteration of the BFGS formula (13), and if exact line searches are performed, the algorithm will be equivalent to the Fletcher–Reeves nonlinear conjugate-gradient method.

3.2. Truncated-Newton methods. The modified Lanczos method (see § 2) may be applied directly to the Newton equations (1). When exact arithmetic is used and the Lanczos iteration is continued until β_{q+1} is zero (i.e., q steps are computed), the

nonlinear algorithm is a modified Newton method with a positive-definite approximation to the Hessian

$$G^{(k)}+\Omega_k,$$

where Ω_k is the matrix $V_q E_q V_q^T$. Note that, unlike the modification produced by the direct application of the modified Cholesky factorization, Ω_k is not a diagonal matrix. The orthogonality of the Lanczos vectors implies that

$$\| V_q E_q V_q^T \| = \| E_q \|.$$

Consequently, when the diagonal modification to T_q is small, the modification to $G^{(k)}$ will be small also.

The truncated-Newton method of Dembo and Steihaug (1983) "solves" (1) by performing a limited number of iterations of the linear conjugate-gradient method. The iterations are terminated ("truncated") before the system is solved exactly. The final iterate of the truncated sequence is then taken as an approximate solution of (1). If a single linear iteration is used, p_k will be the steepest-descent direction $-g^{(k)}$; if the sequence is not truncated, p_k will be the solution of (1). Thus, the algorithm computes a vector that interpolates between the steepest-descent direction and the Newton direction.

Dembo and Steihaug showed that, if the initial iterate of the linear conjugate-gradient scheme is the steepest-descent drection $-g^{(k)}$ and if the iteration is terminated when indefiniteness is detected, then the search direction computed will be a descent direction with respect to $F(x)$. We propose that the modified Lanczos algorithm be used to compute the direction of search. The following theorem indicates that the direction of search obtained by terminating the modified Lanczos scheme will always be a direction of descent.

THEOREM 3. *Let $\{p_q\}$ denote the sequence of iterates computed by the modified Lanczos algorithm with p_0 equal to the zero vector. Then $p_q^T g^{(k)} < 0$ for all $q \geqq 1$.*

Proof. When p_0 is zero, p_q $(q \geqq 1)$ is the solution of the minimization problem

$$\underset{p \in \mathcal{V}_{q-1}}{\text{minimize}} \tfrac{1}{2} p^T (G^{(k)}+\Omega_k) p + p^T g^{(k)}.$$

Thus

$$p_q = -V_{q-1}(V_{q-1}^T (G^{(k)}+\Omega_k) V_{q-1})^{-1} V_{q-1}^T g^{(k)}.$$

Direct premultiplication by $(g^{(k)})^T$ gives

$$p_q^T g^{(k)} = -(g^{(k)})^T V_{q-1}(V_{q-1}^T (G^{(k)}+\Omega_k) V_{q-1})^{-1} V_{q-1}^T g^{(k)} < 0,$$

since, by construction, $V_{q-1}^T (G^{(k)}+\Omega_k) V_{q-1}$ is positive definite. □

We believe that a truncated-Newton method will be successful only when a good direction can be produced in a small number of linear conjugate-gradient iterations, and hence the use of preconditioning is essential. The matrix H from an r-step limited-memory quasi-Newton method is one choice for a preconditioning matrix. In this case, the vector $-Hg^{(k)}$ will be the first nontrivial member of the sequence $\{p_q\}$ and this direction is far more likely to give a good reduction in the function than the negative gradient. Thus the search direction will interpolate between the Newton direction and a quasi-Newton direction, rather than the steepest-descent direction. As we approach the solution, and $F(x)$ looks more and more like a quadratic function, r of the eigenvalues of the matrix $HG^{(k)}$ will tend to unity, thereby reducing the number of iterations required to compute a good search direction.

A diagonal scaling of the variables can further improve the performance of a limited-memory quasi-Newton (Gill and Murray (1979)). A diagonal scaling can be developed by using the matrix/vector products from the modified-Lanczos algorithm to approximate the diagonal of $G^{(k)}$. The diagonal of a BFGS quasi-Newton approximation to $G^{(k)}$ is maintained, using the vector pair $(u_q, G^{(k)}u_q)$ in place of the traditional curvature information (s_q, y_q) (in this context, the "objective function" is the quadratic model $Q(p)$; thus $s_q = \alpha_q u_q = p_{q+1} - p_q$ and $y_q = \alpha_q G^{(k)} u_q = r_{q+1} - r_q$, where $r_q = G^{(k)} p_q + g^{(k)}$). The diagonal scaling matrix $D^{(k)}$ that is developed while finding $p^{(k)}$ is used to precondition the next major iteration for finding $p^{(k+1)}$. $D^{(k)}$ is also used to initialize the approximation to the diagonal of $G^{(k+1)}$. This approach is effective because of the equivalence between the BFGS algorithm and the linear conjugate-gradient algorithm when applied to a quadratic function. For details of these techniques, see Nash (1982).

With these facts in mind, in the numerical tests the modified Lanczos algorithm will be preconditioned using a two-step limited-memory quasi-Newton update (13), initialized with this approximation to the diagonal of $G^{(k)}$.

4. Numerical results. In this section, we discuss the numerical behavior of the methods discussed earlier. The results given here are intended only to show the promise of truncated-Newton algorithms. We have only discussed in detail preconditioning using a nonlinear conjugate-gradient-type algorithm. We believe that practical truncated-Newton algorithms will also exploit information computed during the linear iterations. This topic is discussed in Nash (1982), and more complete numerical results are presented there.

Three of the test problems arise from the discretization of problems in the calculus of variations. The general problem is to find the minimum of the functional

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

over the set of piecewise differentiable curves with the boundary conditions $x(0) = a$, $x(1) = b$. When $x(t)$ is expressed as a linear sum of functions that span the space of piecewise cubic polynomials, then minimization of J becomes a finite-dimensional problem with a block-tridiagonal Hessian matrix.

Example 1. Cal 1.

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

with the boundary conditions $x(0) = 1$, $x(1) = 2$; $n = 50, 100$.

Example 2. Cal 2.

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

with the boundary conditions $x(0) = x(1) = 0$; $n = 50, 100$.

Example 3. Cal 3.

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

with the boundary conditions $x(0) = 1$, $x(1) = 0$; $n = 50,100$.

For these three examples, the starting point was $x^{(0)} = (0, \cdots, 0)^T$. The remaining two examples follow.

Example 4. GenRose (Generalized Rosenbrock function).

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

with $n = 50, 100$. The Hessian is tridiagonal.

Example 5. Chebyquad.

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

where

$$f_i(x) = \int_0^1 T_i^*(x)\, dx - \frac{1}{n} \sum_{j=1}^{n} T_i^*(x_j),$$

and $T_i^*(x)$ is the ith order shifted Chebyshev polynomial. The Hessian matrix is dense; $n = 20$.

For the final two examples, the starting point is $x^{(0)} = (1/(n+1), \cdots, n/(n+1))^T$. In all cases, the eigenvalues of the Hessian are not clustered, making the problems challenging for a truncated-Newton method. For further details on these examples, see Gill and Murray (1979).

4.1. Details of the algorithms. We have compared two truncated-Newton algorithms to a modified Newton method, and to a nonlinear conjugate-gradient-type method. One of the truncated-Newton methods, BTN (Basic TN), is a truncated-Newton routine without a preconditioning strategy. This was included to show the necessity of preconditioning in a practical algorithm. The modified Newton algorithm MNA uses the modified-Cholesky factorization of Gill and Murray (1974b). The nonlinear conjugate-gradient-type method PLMA is a two-step BFGS limited-memory quasi-Newton method with a diagonal scaling (Gill and Murray (1979)). With a different diagonal scaling, PLMA corresponds to the preconditioning operator used for the truncated-Newton method TN, described in §§ 2 and 3. PLMA is the most successful method of its type discussed in the survey of Gill and Murray (1979). All the routines are coded in double precision FORTRAN IV. The runs were made on an IBM 370/168, for which the relative machine precision ε is approximately 10^{-15}.

The truncated-Newton routines require the computation of matrix/vector products of the form $G^{(k)}v$. For routine BTN and for Example 5, these were computed by differencing the gradient along the vector v (Gill and Murray (1974a), O'Leary (1982)). The difference parameter used was $\varepsilon^{1/2}(1 + \|x^{(k)}\|)$, where ε is the machine precision. Elsewhere, sparse finite-differencing techniques (Thapa (1980)) were used to approximate $G^{(k)}$ at the beginning of each major iteration, and this approximation was used to compute the matrix/vector products. The difference parameter used here was $\varepsilon^{1/2}$ (this parameter was set internally by the sparse-finite-differencing package). Because our interest is in methods that do not require second derivatives, tests were not made using exact second-derivative information.

For both truncated-Newton algorithms, a fairly stringent criterion was used to terminate the modified Lanczos iterations. Following Dembo and Steihaug (1983), the modified Lanczos iterations are terminated if

$$\frac{\|r_q\|}{\|g^{(k)}\|} \leq \min\{1/k, \|g^{(k)}\|\},$$

where r_q is the qth residual of the linear system. This criterion forces the algorithm to behave like a conjugate-gradient algorithm near the beginning of the iteration and

like Newton's method near the solution. We stress, however, that when second derivatives are not available, or the cost of the matrix/vector product $G^{(k)}v$ is high, a criterion must be used that always leads to a small number of linear iterations. Because the computation of the search direction can be degraded by loss of orthogonality, at most $n/2$ modified Lanczos iterations were allowed at each major step.

Each problem was solved using three values of η, the step-length accuracy; these values were 0.25, 0.1, and 0.001. (The step length α must satisfy $|g(x+\alpha p)^T p| \leqq -\eta g(x)^T p$, where $0 \leqq \eta < 1$.) Each algorithm requires two additional user-specified parameters. The first (λ) limits the change in x at each iteration (the quantity $\|x^{(k+1)} - x^{(k)}\|_2$). The value of λ was set at 10 for all problems to avoid overflow during the computation of the objective function. The second parameter is an estimate of the value of the objective function at the solution and is used to compute the initial step for the step-length algorithm. In each case, this parameter was set to the value of $F(x)$ at the solution.

The results are contained in Table 1. Each table entry refers to the iteration at which

$$F^{(k)} - F(x^*) < 10^{-5}(1 + |F(x^*)|),$$

where x^* is the solution.

4.2. Discussion of results. We emphasize that we do not intend the results contained in Table 1 to be used as a thorough comparison for the four methods PLMA, TN, BTN, and MNA. The last algorithm requires second derivatives and therefore a direct comparison cannot be made. However, the table does indicate that TN gives results more similar to those of MNA than to those of PLMA.

With the exception of the results for TN, each entry is a pair of numbers: the first is the number of major iterations: the second is the number of function/gradient evaluations required to solve the problem (for BTN, this reflects both the line search and the matrix/vector products). For TN, more detailed results are given. The first pair of numbers gives the total number of iterations, and the number of function/gradient evaluations used in the line search. The finite-difference column records the number of gradient evaluations used to compute the matrix/vector products. The next column is the total number of modified Lanczos iterations (each iteration will normally be dominated by the cost of the matrix/vector product, comparable to a gradient evaluation). The final column combines the line-search cost with the inner-iteration cost to give a measure of the total cost of the minimization; two totals are given: the first combines the line-search costs with the finite differencing costs, and the second with the iteration costs.

The derivation of the conjugate-gradient algorithm via the Lanczos algorithm was carried out primarily to produce a safer method for computing a search direction when the Hessian is indefinite. Examination of the results for the test functions Chebyquad and GenRose indicates that this approach is also effective computationally (indefiniteness was not encountered when minimizing the other functions). The truncated-Newton algorithm (as measured using column TN in the table) performs 2 to 8 times as well as MNA, the modified-Newton algorithm. (The totals column is not used to evaluate algorithm TN because the MNA results do not reflect the cost of second-derivative information either.) This is surprising since the truncated-Newton method is a compromise of Newton's method for use on large-scale problems. For the calculus of variations problems Cal 1–3, MNA performed better than TN. However, the number of iterations used by both methods are nearly the same; and the line-search results for TN are not much greater than for MNA.

TABLE 1

Function	η	PLMA		MNA		BTN		TN		Finite-difference	Lanczos iterations	Totals
Cal 1	.25	194	366	7	9	16	978	10	18	60	181	78/199
	.1	204	401	6	11	17	1151	9	19	54	139	73/158
$n = 50$	.001	205	456	6	17	13	787	9	34	54	132	88/166
Cal 2	.25	64	106	4	4	8	133	7	8	42	61	50/69
	.1	61	118	4	6	8	133	7	8	42	61	50/69
$n = 50$	.001	60	123	4	4	8	135	7	10	42	61	52/71
Cal 3	.25	80	152	6	6	7	206	7	8	42	104	50/112
	.1	78	155	5	7	7	206	7	8	42	104	50/112
$n = 50$	.001	77	161	5	11	8	289	7	17	42	96	59/113
Cal 1	.25	423	819	NR		26	3855	10	19	60	390	79/409
	.1	429	854	NR		23	3214	10	25	60	484	85/409
$n = 100$	.001	416	905	NR		26	3948	10	35	60	337	95/372
Cal 2	.25	112	204	NR		8	256	8	9	48	108	57/117
	.1	107	206	NR		8	256	8	9	48	108	57/117
$n = 100$	.001	113	228	NR		8	259	8	11	48	111	59/122
Cal 3	.25	143	270	NR		9	508	7	8	42	151	50/159
	.1	142	281	NR		9	636	7	9	42	159	51/168
$n = 100$	.001	138	284	NR		9	617	7	18	42	158	60/176
GenRose	.25	108	201	62	202	33	499	31	75	93	255	168/330
	.1	119	263	66	257	32	518	33	99	99	249	198/348
$n = 50$	.001	119	330	88	392	35	614	34	143	102	252	245/395
GenRose	.25	191	365	NR		60	1150	57	164	171	420	335/684
	.1	192	410	NR		62	1153	60	190	180	585	370/775
$n = 100$	.001	188	528	NR		60	1197	58	248	174	534	422/782
Chebyquad	.25	38	75	29	121	10	104	7	16	37	37	53/53
	.1	33	71	24	116	9	96	8	17	51	51	68/68
$n = 20$	.001	33	90	30	161	11	164	9	29	61	61	90/90

NR—not run (due to expense). For paired entries (n_1, n_2), n_1 is the number of major iterations required, and n_2 is the number of function/gradient evaluations used.

For Cal 1–3, TN is 1.5 to 3 times as efficient as PLMA (using the "totals" column for comparison). TN performs at least as well on Chebyquad; on GenRose, the result of the comparison depends on which set of totals for TN is used. Since PLMA is a high-quality routine for large-scale optimization, the truncated-Newton algorithm is performing effectively on these problems.

By comparing the results for BTN (the unpreconditioned algorithm) with TN, it is seen that BTN is 1.5 to 10 times as expensive to use as the preconditioned algorithm. BTN is particularly slow for function Cal 1. This indicates the importance of preconditioning strategies in practical computations.

Finally, we compare these results with those in O'Leary (1982), where a different matrix factorization approach is used to stabilize the linear conjugate-gradient iteration. For the function GenRose ($n = 50$), that algorithm required 1373 gradient evaluations; for GenRose ($n = 100$), 2616. These totals are considerably higher than for the routine

TN (330 and 684). The two sets of tests are not exactly comparable because of differences in the computers and line-search algorithms used. The remaining test functions used in that paper were not considered here because they are either small (at most 6 variables, so there is little difference between a truncated-Newton and a Newton algorithm) or easy to solve.

The techniques described in § 2.5 to compute a direction of negative curvature were never required. These techniques are unlikely to be needed unless the initial guess $x^{(0)}$ is a stationary point that is not a local minimum.

5. Conclusions. We have described several techniques designed to make truncated-Newton methods an effective tool for large-scale function minimization. The methods have low-storage requirements, and require only first-derivative function information. If the search direction is computed using the linear conjugate-gradient method, derived via the Lanczos method, the algorithm can be made stable even when the Hessian is indefinite. This approach also makes progress possible from a stationary point that is not a local minimum.

In practice, a truncated-Newton method must be preconditioned to make it efficient. One preconditioning strategy was discussed that is obtained dynamically by the algorithm as the problem is being solved, without a priori information. This technique allows the truncated-Newton algorithm to compute a search direction that interpolates between a nonlinear conjugate-gradient direction and the Newton direction.

The numerical tests indicated that a truncated-Newton algorithm can perform more like Newton's method than like a nonlinear conjugate-gradient-type method. The performance is especially remarkable on nonconvex problems.

Acknowledgments. The author would like to thank his thesis advisors Philip Gill, Gene Golub, and Walter Murray for their many helpful comments. Thanks also to Chris Paige, Michael Saunders, and Margaret Wright for their careful reading of earlier drafts of this report. We are grateful to Mukund Thapa for kindly providing the subroutines for computing the sparse Hessian matrices of the test examples.

REFERENCES

[1] O. AXELSSON, *Solution of linear systems of equations: iterative methods*, in Sparse Matrix Techniques, V. A. Barber, ed., Lecture Notes in Mathematics 572, Springer-Verlag, Berlin, 1977.

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

[3] J. R. BUNCH AND B. N. PARLETT, *Direct methods for solving symmetric indefinite systems of linear equations*, this Journal, 8 (1971), pp. 639–655.

[4] R. CHANDRA, *Conjugate-gradient methods for partial differential equations*, Report # 129, Dept. Computer Science, Yale Univ., New Haven, CT, 1978.

[5] R. S. DEMBO AND T. STEIHAUG, *Truncated-Newton algorithms for large-scale unconstrained optimization*, Math. Prog., 26 (1983), pp. 190–212.

[6] J. E. DENNIS AND J. J. MORÉ, *Quasi-Newton methods, motivation and theory*, SIAM Rev., 19 (1977), pp. 46–89.

[7] D. M. GAY, *Computing optimal locally constrained steps*, SIAM J. Sci. Stat. Comput., 2 (1981), pp. 186–197.

[8] P. E. GILL AND W. MURRAY, eds., *Numerical Methods for Constrained Optimization*, Academic Press, London and New York, 1974 (=(1974a)).

[9] ———, *Newton-type methods for unconstrained and linearly constrained optimization*, Math. Prog., 28 (1974), pp. 311–350 (=(1974b)).

[10] ———, *Conjugate-gradient methods for large-scale nonlinear optimization*, Report SOL 79-15, Operations Research Dept., Stanford Univ., Stanford, CA, 1979.

[11] P. E. GILL, W. MURRAY, M. A. SAUNDERS AND M. H. WRIGHT, *QP-based methods for large-scale nonlinearly constrained optimization*, presented at the Nonlinear Programming 4 Symposium, Madison, WI, July 1980.

[12] M. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. National Bureau of Standards, 49 (1952), pp. 409–436.

[13] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. National Bureau of Standards, 45 (1950), pp. 255–282.

[14] D. G. LUENBERGER, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1973.

[15] B. A. MURTAGH AND M. A. SAUNDERS, *Large-scale linearly constrained optimization*, Math. Prog., 14 (1978), pp. 41–72.

[16] ———, *The implementation of a Lagrangian-based algorithm for sparse nonlinear constraints*, Report SOL 80-1, Dept. Operations Research, Stanford Univ., Stanford, CA, 1980.

[17] S. G. NASH, *Preconditioning of truncated-Newton methods*, Report 371, Mathematical Sciences Dept., The Johns Hopkins Univ., Baltimore, MD, 1982.

[18] J. NOCEDAL, *Updating quasi-Newton matrices with limited storage*, Math. Comp., 35 (1980), pp. 773–782.

[19] D. P. O'LEARY, *A discrete Newton algorithm for minimizing a function of many variables*, Math. Prog., 23 (1982), pp. 20–33.

[20] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, this Journal, 12 (1975), pp. 617–629.

[21] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[22] D. F. SHANNO, *Conjugate gradient methods with inexact searches*, Math. Oper. Res., 3 (1978), pp. 244–256.

[23] M. THAPA, *Optimization of unconstrained functions with sparse Hessian matrices*, Ph.D. thesis, Operations Research Dept., Stanford Univ., Stanford, CA, 1980.