TENSOR METHODS FOR LARGE, SPARSE UNCONSTRAINED OPTIMIZATION*

ALI BOUARICHA†

Abstract. Tensor methods for unconstrained optimization were first introduced by Schnabel and Chow [*SIAM J. Optimization*, 1 (1991), pp. 293-315], who describe these methods for small to moderate-size problems. The major contribution of this paper is the extension of these methods to large, sparse unconstrained optimization problems. This extension requires an entirely new way of solving the tensor model that makes the methods suitable for solving large, sparse optimization problems efficiently. We present test results for sets of problems where the Hessian at the minimizer is nonsingular and where it is singular. These results show that tensor methods are significantly more efficient and more reliable than standard methods based on Newton's method.

Key words. tensor methods, unconstrained optimization, sparse problems, large-scale optimization, singular problems

AMS subject classifications. 65K

1. Introduction.

In this paper we describe tensor methods for solving the unconstrained optimization problem

$$\text{given } f : \Re^n \to \Re, \text{ find } x_* \in \Re^n \text{ such that } f(x_*) \le f(x) \text{ for all } x \in D, \tag{1.1}$$

where D is some open set containing x_*, and f is convex on D. We assume that f is at least twice continuously differentiable, and n is large.

Tensor methods for unconstrained optimization are general-purpose methods primarily intended to improve upon the performance of standard methods, especially on problems where $\nabla^2 f(x_*)$ has a small rank deficiency. They are also intended to be at least as efficient as standard methods on problems where $\nabla^2 f(x_*)$ is nonsingular.

Tensor methods for unconstrained optimization base each iteration upon the fourth-order model of the objective function $f(x)$,

$$M_T(x_c + d) = f(x_c) + \nabla f(x_c) \cdot d + \frac{1}{2}\nabla^2 f(x_c) \cdot d^2 + \frac{1}{6} T_c \cdot d^3 + \frac{1}{24} V_c \cdot d^4, \tag{1.2}$$

where $d \in \Re^n$, x_c is the current iterate, $\nabla f(x_c)$ and $\nabla^2 f(x_c)$ are the first and second analytic derivatives of f at x_c, or finite difference approximations to them, and the tensor terms at x_c, $T_c \in \Re^{n\times n\times n}$ and $V_c \in \Re^{n\times n\times n\times n}$, are symmetric. (We use the notation $\nabla f(x_c) \cdot d$ for $\nabla f(x_c)^T d$, and $\nabla^2 f(x_c) \cdot d^2$ for $d^T \nabla^2 f(x_c) d$ to be consistent with the tensor notation $T_c \cdot d^3$ and $V_c \cdot d^4$. Also, for simplicity, we abbreviate terms of the form dd, ddd, and $dddd$ by d^2, d^3, and d^4, respectively.) Before proceeding, we define the tensor notation used above.

Definition 1.1. Let $T \in \Re^{n\times n\times n}$. Then for $u, v, w \in \Re^n$, $T \cdot uvw \in \Re$, $T \cdot vw \in \Re^n$, with

$$T \cdot uvw = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} T(i,j,k)u(i)v(j)w(k),$$

*Part of this work was performed while the author was research associate at CERFACS (Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique).

†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 60439 (bouarich@@mcs.anl.gov). This work was supported in part by the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38.

$$(T \cdot vw)(i) = \sum_{j=1}^{n} \sum_{k=1}^{n} T(i,j,k)v(j)w(k), \quad i = 1, \cdots, n.$$

Definition 1.2. Let $V \in \Re^{n \times n \times n \times n}$. Then for $r, u, v, w \in \Re^n$, $V \cdot ruvw \in \Re$, $V \cdot uvw \in \Re^n$ with

$$V \cdot ruvw = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V(i,j,k,l)r(i)u(j)v(k)w(l),$$

$$(V \cdot uvw)(i) = \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} V(i,j,k,l)u(j)v(k)w(l), \quad i = 1, \cdots, n.$$

The tensor terms are selected so that the model interpolates a small number of function and gradient values from previous iterations. This results in T_c and V_c being low-rank tensors, which is crucial for the efficiency of the tensor method. The tensor method requires no more function or derivative evaluations per iteration and hardly more storage or arithmetic operations than does a standard method based on Newton's method.

Standard methods for solving unconstrained optimization problems are widely described in the literature; general references on this topic include Dennis and Schnabel [9], Fletcher [12], and Gill, Murray, and Wright [14]. In this paper, we propose extensions to standard methods that use analytic or finite-difference gradients and Hessians.

The standard method for unconstrained optimization, Newton's method, bases each iteration upon the quadratic model of $f(x)$,

$$M_N(x_c + d) = f(x_c) + \nabla f(x_c) \cdot d + \frac{1}{2}\nabla^2 f(x_c) \cdot d^2. \tag{1.3}$$

This method is defined when $\nabla^2 f(x_c)$ is nonsingular and consists of setting the next iterate x_+ to the minimizer of (1.3), namely,

$$x_+ = x_c - \nabla^2 f(x_c)^{-1} \nabla f(x_c). \tag{1.4}$$

A distinguishing feature of Newton's method is that if $\nabla^2 f(x_c)$ is nonsingular at a local minimizer x_*, then the sequence of iterates produced by (1.4) converges locally quadratically to x_*. However, Newton's method is generally linearly convergent at best if $\nabla^2 f(x_*)$ is singular [15].

Methods based on (1.2) have been shown to be more reliable and more efficient than standard methods on small to moderate-size problems [19]. In the test results obtained for both nonsingular and singular problems, the improvement by the tensor method over Newton's method is substantial, ranging from 30% to 50% in iterations and in function and derivative evaluations. Furthermore, the tensor method solves several problems that Newton's method fails to solve.

The tensor algorithms described in [19] are QR-based algorithms involving orthogonal transformations of the variable space. These algorithms are very effective for minimizing the tensor model when the Hessian is dense because they are very stable numerically, especially when the Hessian is singular. They are not efficient for sparse problems, however, because they destroy the sparsity of the Hessian due to the orthogonal transformation of the variable space. To preserve the sparsity of the Hessian, we have developed an entirely new way of solving the tensor model that employs a sparse variant of the Cholesky decomposition. This makes our new algorithms very well suited for sparse problems.

The remainder of this paper is organized as follows. In §2 we briefly review the techniques introduced by Schnabel and Chow [19] to form the tensor model. In §3 we describe efficient algorithms for minimizing the tensor model when the Hessian is sparse. In §§4 and 5 we discuss the globally convergent modifications for tensor methods for large, sparse unconstrained optimization. These consist of line search backtracking and model trust region techniques. A high-level implementation of the tensor method is given in §6. In §7 we describe comparative testing for an implementation based on the tensor method versus an implementation based on Newton's method, and we present summary statistics of the test results. Finally, in §8, we give a summary of our work and a discussion of future research.

2. Forming the Tensor Model. In this section, we briefly review the techniques that were introduced in [19] for forming the tensor model for unconstrained optimization.

As was stated in the preceding section, the tensor method for unconstrained optimization bases each iteration upon the fourth-order model of the nonlinear function $f(x)$ given by (1.2).

The choices of T_c and V_c in (1.2) cause the third-order term $T_c \cdot d^3$ and the fourth-order term $V_c \cdot d^4$ to have simple and useful forms. These tensor terms are selected so that the tensor model interpolates function and gradient information at a set of p not necessarily consecutive past iterates $x_{-1}, \cdots, x_{-p}$.

In the remainder of this paper, we restrict our attention to $p = 1$. The reasons for this choice are that the performance of the tensor version that allows $p \geq 1$ is similar overall to that constraining p to be 1, and that the method is simpler and less expensive to implement in this case. (The derivation of the third- and fourth-order tensor terms for $p \geq 1$ is explained in detail in [19].)

The interpolation conditions at the past point x_{-1} are given by

$$f(x_{-1}) \;=\; f(x_c) \;+\; \nabla f(x_c) \cdot s \;+\; \frac{1}{2}\nabla^2 f(x_c) \cdot s^2 + \frac{1}{6}T_c \cdot s^3 + \frac{1}{24}V_c \cdot s^4 \tag{2.1}$$

and

$$\nabla f(x_{-1}) \;=\; \nabla f(x_c) \;+\; \nabla^2 f(x_c) \cdot s \;+\; \frac{1}{2}T_c \cdot s^2 \;+\; \frac{1}{6}V_c \cdot s^3, \tag{2.2}$$

where

$$s \;=\; x_{-1} - x_c.$$

Schnabel and Chow [19] choose T_c and V_c to satisfy (2.1) and (2.2). They first show that the interpolation conditions (2.1) and (2.2) uniquely determine $T_c \cdot s^3$ and $V_c \cdot s^4$. Multiplying (2.2) by s yields

$$\nabla f(x_{-1}) \cdot s \;=\; \nabla f(x_c) \cdot s \;+\; \nabla^2 f(x_c) \cdot s^2 \;+\; \frac{1}{2}T_c \cdot s^3 \;+\; \frac{1}{6}V_c \cdot s^4. \tag{2.3}$$

Let α, $\beta \in \Re$ be defined by

$$\alpha = T_c \cdot s^3,$$

$$\beta = V_c \cdot s^4.$$

Then from (2.1) and (2.3) they obtain the following system of two linear equations in the two unknowns α and β:

$$\frac{1}{2}\alpha \; + \; \frac{1}{6}\beta \; = \; q_1, \tag{2.4}$$

$$\frac{1}{6}\alpha \; + \; \frac{1}{24}\beta \; = \; q_2, \tag{2.5}$$

where q_1, $q_2 \in \Re$ are defined by

$$q_1 \; = \; \nabla f(x_{-1}) \cdot s \; - \; \nabla f(x_c) \cdot s \; - \; \nabla^2 f(x_c) \cdot s^2,$$

$$q_2 \; = \; f(x_{-1}) \; - \; f(x_c) \; - \; \nabla f(x_c) \cdot s \; - \; \frac{1}{2}\nabla^2 f(x_c) \cdot s^2.$$

The system (2.4)–(2.5) is nonsingular; therefore the values of α and β are uniquely determined. Hence, the interpolation conditions uniquely determine $T_c \cdot s^3$ and $V_c \cdot s^4$. Since these are the only interpolation conditions, the choice of T_c and V_c is vastly underdetermined.

Schnabel and Chow [19] choose T_c and V_c by first selecting the smallest symmetric V_c, in the Frobenius norm, for which

$$V_c \cdot s^4 \; = \; \beta,$$

where β is determined by (2.4)–(2.5). Then they substitute this value of V_c into (2.2), obtaining

$$T_c \cdot s^2 = a, \tag{2.6}$$

where

$$a \; = \; 2(\nabla f(x_{-1}) \; - \; \nabla f(x_c) \; - \; \nabla^2 f(x_c) \cdot s \; - \; \frac{1}{6} V_c \cdot s^3). \tag{2.7}$$

This is a set of n linear equations in n^3 unknowns $T_c(i,j,k)$, $1 \le i,j,k \le n$. More precisely, Schnabel and Chow [19] choose the smallest symmetric T_c and V_c, in the Frobenius norm, that satisfy the equations (2.6)–(2.7). That is,

$$\min_{V_c \in \Re^{n \times n \times n \times n}} \; \| \, V_c \, \|_F \tag{2.8}$$

$$\text{subject to } V_c \cdot s^4 \; = \; \beta, \text{ and } V_c \text{ is symmetric,}$$

and

$$\min_{T_c \in \Re^{n \times n \times n}} \; \| \, T_c \, \|_F \tag{2.9}$$

$$\text{subject to } T_c \cdot s^2 \; = \; a, \text{ and } T_c \text{ is symmetric.}$$

The solution to (2.8) is

$$V_c \;=\; \gamma\; (s \otimes s \otimes s \otimes s), \quad \gamma = \frac{\beta}{(s^T s)^4},$$

where the tensor $V_c = s \otimes s \otimes s \otimes s \in \Re^{n\times n\times n\times n}$ is called a fourth-order rank-one tensor for which $V_c(i,j,k,l) = s(i)s(j)s(k)s(l),\ 1 \le i,j,k,l \le n$. (We use the notation $\otimes$ to be consistent with [19].)

The solution to (2.9) is

$$T_c \;=\; b \otimes s \otimes s + s \otimes b \otimes s + s \otimes s \otimes b, \tag{2.10}$$

where the notation $T = u \otimes v \otimes w,\ u,v,w \in \Re^n,\ T \in \Re^{n\times n\times n}$, is called a third-order rank-one tensor for which $T(i,j,k) = u(i)v(j)w(k)$. Here $b \in \Re^n$ is the unique vector for which (2.10) satisfies (2.6). It is given by

$$b \;=\; \frac{3a(s^T s) \;-\; 2s(s^T a)}{3(s^T s)^3}.$$

T_c and V_c determined by the minimum norm problems (2.9) and (2.8) have rank 2 and 1, respectively. This is the key to form, store, and solve the tensor model efficiently. The whole process of forming the tensor model requires only $O(n^2)$ arithmetic operations. The storage needed for forming and storing the tensor model is only a total of $6n$.

For further information we refer to [19].

3. Solving the Tensor Model When the Hessian Is Sparse. In this section we give algorithms for finding a minimizer of the tensor model (1.2) efficiently, when the Hessian is sparse.

The substitution of the values of T_c and V_c into (1.2) results in the tensor model

$$\begin{aligned} M_T(x_c+d) \;&=\; f(x_c) + \nabla f(x_c)\cdot d + \frac{1}{2}\nabla^2 f(x_c)\cdot d^2 \\ &\quad + \frac{1}{2}(b^T d)(s^T d)^2 + \frac{\gamma}{24}(s^T d)^4. \end{aligned} \tag{3.1}$$

As we stated in §2, we only consider the case $p = 1$ where the tensor model interpolates $f(x)$ and $\nabla f(x)$ at the previous iterate. The generalization for $p \ge 1$ is fairly straightforward. This constraint is mainly motivated by our computational results. When we allow $p \ge 1$, our test results showed almost no improvement over the case where $p = 1$. The tensor method is therefore considerably simpler, as well as cheaper in terms of storage and cost per iteration.

3.1. Case 1: The Hessian Is Nonsingular. We show that the minimization of (3.1) can be reduced to the solution of a third-order polynomial in one unknown, plus the solution of three systems of linear equations that all involve the same coefficient matrix $\nabla^2 f(x_c)$. For conciseness, we use the notation $g = \nabla f(x_c)$ and $H = \nabla^2 f(x_c)$.

A necessary condition for d to be a local minimizer of (3.1) is that the derivative of the tensor model with respect to d must be zero. That is,

$$\nabla M_T(x_c+d) = g + Hd + (b^T d)(s^T d)s + \frac{1}{2}(s^T d)^2 b + \frac{\gamma}{6}(s^T d)^3 s = 0,$$

which yields

$$d = -H^{-1}(g + (b^T d)(s^T d)s + \frac{1}{2}(s^T d)^2 b + \frac{\gamma}{6}(s^T d)^3 s). \tag{3.2}$$

If we first premultiply the equation (3.2) by s^T on both sides, we obtain a cubic equation in the unknowns $\beta = s^T d$ and $\theta = b^T d$,

$$s^T H^{-1} g + \beta + s^T H^{-1} s\theta\beta + \frac{1}{2} s^T H^{-1} b\beta^2 + \frac{\gamma}{6} s^T H^{-1} s\beta^3 = 0. \tag{3.3}$$

If we then premultiply the equation (3.2) by b^T on both sides, we obtain another cubic equation in the unknowns β and θ,

$$b^T H^{-1} g + \theta + b^T H^{-1} s\theta\beta + \frac{1}{2} b^T H^{-1} b\beta^2 + \frac{\gamma}{6} b^T H^{-1} s\beta^3 = 0. \tag{3.4}$$

Thus, we obtain a system of two cubic equations in the two unknowns β and θ which can be solved analytically.

We now show how to compute the solutions of this system of two cubic equations in two unknowns by computing the solutions of a single cubic equation in the unknown β. Let $u = s^T H^{-1} g$, $v = s^T H^{-1} b$, $w = s^T H^{-1} s$, $y = b^T H^{-1} g$, and $z = b^T H^{-1} b$. We first calculate the value of θ as a function of β using the equation (3.3):

$$\theta = -\frac{(u + \beta + \frac{1}{2} v\beta^2 + \frac{\gamma}{6} w\beta^3)}{w\beta}. \tag{3.5}$$

Note that the denominator of (3.5) is equal to zero if either $\beta = 0$ or $w = 0$. We assume that $\beta \neq 0$; otherwise the tensor model would be reduced to the Newton model. Now, if $w = 0$, then (3.3) would be quadratic in β and

$$\beta = \frac{-1 \pm \sqrt{1 - 2uv}}{2}.$$

In this case, real-valued minimizers of the tensor model (3.1) may exist only if $1 - 2uv \geq 0$. It is easy to check that in order for θ to have a defined value, $1 + v\beta$ cannot be zero.

If $\beta \neq 0$ and $w \neq 0$, we substitute the expression for θ into (3.4) and obtain

$$-u + (yw - uv - 1)\beta - \frac{3}{2} v\beta^2 + (\frac{1}{2} wz - \frac{\gamma}{6} w - \frac{1}{2} v^2)\beta^3 = 0, \tag{3.6}$$

which is a third-order polynomial in the one unknown β. The roots of (3.6) are computed analytically. We substitute the values of β into (3.5) to calculate the values of θ. Then we simply substitute the values of β and θ into (3.2) to obtain the values of d. The major cost in this whole process is the calculation of $H^{-1}g$, $H^{-1}b$, and $H^{-1}s$.

After we compute the values of d, we determine which of them are potential minimizers. Our criterion is to select those values of d that guarantee that there is a descent path from x_c to $x_c + d$ for the model $M_T(x_c + d)$. Then among the selected steps, we choose the one that is closest to the current iterate x_c in the Euclidean norm sense. If the tensor model has no minimizer, we use the standard Newton step as the step direction for the current iteration.

3.2. Case 2: The Hessian Is Rank Deficient. If the Hessian matrix is rank deficient, we transform the tensor model given in (3.1) by the following procedure. Let $d = \hat{d} + \delta$ for a fixed $\hat{d}$, where δ is the new unknown. Substituting this expression for d into (3.1) yields the following tensor model, which is a function of δ:

$$\begin{aligned} M_T(x_c + d) &= f(x_c) + \nabla f(x_c) \cdot \hat{d} + \frac{1}{2}\nabla^2 f(x_c) \cdot \hat{d}^2 + \frac{1}{2}(b^T \hat{d})(s^T \hat{d})^2 \\ &\quad + \frac{\gamma}{24}(s^T \hat{d})^4 + (\nabla f(x_c) + \nabla^2 f(x_c)\hat{d} + (b^T \hat{d})(s^T \hat{d})s \\ &\quad + \frac{1}{2}(s^T \hat{d})^2 b + \frac{\gamma}{24}(s^T \hat{d})^3 s) \cdot \delta + \frac{1}{2}(\nabla^2 f(x_c) \\ &\quad + (b^T \hat{d} + \frac{\gamma}{2} s s^T) \cdot \delta^2 + (s^T \hat{d})(b^T \delta)(s^T \delta) + \frac{1}{2}(b^T \delta)(s^T \delta)^2 \\ &\quad + \frac{\gamma}{6}(s^T \hat{d})(s^T \delta)^3 + \frac{\gamma}{24}(s^T \delta)^4. \end{aligned} \tag{3.7}$$

If we let $\hat{\beta} = s^T \hat{d}$, $\hat{\theta} = b^T \hat{d}$, $\hat{g} = \nabla f(x_c) + \nabla^2 f(x_c)\hat{d} + \hat{\theta}\hat{\beta}s + \frac{1}{2}\hat{\beta}^2 b + \frac{\gamma}{6}\hat{\beta}^3 s$, $c = b^T \hat{d} + \frac{\gamma}{2}$, and $\hat{H} = \nabla^2 f(x_c) + css^T$, then we obtain the modified tensor model

$$\begin{aligned} M_T(x_c + d) &= M_T(x_c + \hat{d}) + \hat{g} \cdot \delta + \frac{1}{2}\hat{H} \cdot \delta^2 + \hat{\beta}(b^T \delta)(s^T \delta) \\ &\quad + \frac{1}{2}(b^T \delta)(s^T \delta)^2 + \frac{\gamma}{6}\hat{\beta}(s^T \delta)^3 + \frac{\gamma}{24}(s^T \delta)^4. \end{aligned} \tag{3.8}$$

The advantage of this transformation is that the matrix $\hat{H}$ is likely to be nonsingular if the rank of $\nabla^2 f(x_c)$ is at least $n - 1$. A necessary and sufficient condition for $\hat{H}$ to be nonsingular is given in the following lemma. Let g and H denote $\nabla f(x_c)$ and $\nabla^2 f(x_c)$, respectively.

Lemma 3.1. Let $H \in \Re^{n \times n}$, $s \in \Re^n$.

$$H + css^T \textit{ is nonsingular if and only if } M = \begin{bmatrix} H & cs \\ cs^T & -c \end{bmatrix} \textit{ is nonsingular.}$$

(Note that the $\begin{bmatrix} s^T & -1 \end{bmatrix}$ submatrix was premultiplied by the constant c to symmetrize the augmented matrix M.)

Proof. We prove that there exists $v \in \Re^n$, $v \neq 0$, for which $(H + css^T)v = 0$, if and only if there exist $\bar{v} \in \Re^n$, $w \in \Re$, for which

$$\begin{bmatrix} H & cs \\ cs^T & -c \end{bmatrix} \begin{bmatrix} \bar{v} \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} \bar{v} \\ w \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{3.9}$$

Suppose first that $(H + css^T)v = 0, v \neq 0$. Then for $\bar{v} = v, w = s^T v$, $(\bar{v}, w)$ satisfies (3.9). Conversely, if there exists $(\bar{v}, w)$ satisfying (3.9), then $s^T \bar{v} = w$, so $(H + css^T)\bar{v} = 0$, and $\bar{v} \neq 0$; otherwise, $w = 0$, which contradicts (3.9). Thus $(H + css^T)$ is singular if and only if M is singular.

Corollary 3.2. Let $H \in \Re^{n\times n}$, $s \in \Re^n$.

$$\textit{If } H + css^T \textit{ is nonsingular, then } \left[\begin{array}{cc} H & cs \end{array}\right] \textit{ has full row rank.}$$

Proof. Follows from Lemma 3.1.

Lemma 3.3. Let $H \in \Re^{n\times n}$, $\mathrm{rank}(H) = n-1$, $s \in \Re^n$.

$$H + css^T \textit{ is nonsingular if and only if } \left[\begin{array}{cc} H & cs \end{array}\right] \textit{ has full row rank.}$$

Proof. The *only if* part follows from Corollary 3.2. Now assume $\left[\begin{array}{cc} H & cs \end{array}\right]$ has full row rank. Since H has rank $n-1$, $H = H_1 H_2^T$, where $H_1, H_2 \in \Re^{n\times(n-1)}$ have full column rank. Since $\left[\begin{array}{cc} H & cs \end{array}\right]$ has full row rank,

$$(v^T H = 0 \text{ and } v^T s = 0) \Rightarrow v = 0. \tag{3.10}$$

From $H = H_1 {H_2}^T$ and the fact that H_2 has full column rank, (3.10) is equivalent to

$$(v^T H_1 = 0 \text{ and } v^T s = 0) \Rightarrow v = 0.$$

Thus the $n \times n$ matrix $\left[\begin{array}{cc} H_1 & cs \end{array}\right]$ is nonsingular. Analogously, the $n\times n$ matrix $\left[\begin{array}{cc} H_2 & s \end{array}\right]$ is nonsingular. Therefore

$$\left[\begin{array}{cc} H_1 & cs \end{array}\right] \left[\begin{array}{c} H_2^T \\ s^T \end{array}\right] = H_1 {H_2}^T + css^T = H + css^T$$

is nonsingular. □

Even though $\hat{H}$ is not sparse in general, Lemma 3.1 will be used later on to exploit the sparsity of H when working with $\hat{H}$.

For δ to be a local minimizer of (3.8) the derivative of the tensor model (3.8) with respect to δ must be zero. That is,

$$\begin{aligned} \nabla M_T(x_c+\delta) &= \hat{g} + \hat{H}\delta + \hat{\beta}(s^T\delta)b + \hat{\beta}(b^T\delta)s + (s^T\delta)(b^T\delta)s \\ &\quad + (\frac{1}{2}b + \frac{\gamma}{2}\hat{\beta}s)(s^T\delta)^2 + \frac{\gamma}{6}(s^T\delta)^3 s = 0, \end{aligned} \tag{3.11}$$

which yields

$$\begin{aligned} \delta &= -\hat{H}^{-1}(\hat{g} + \hat{\beta}(s^T\delta)b + \hat{\beta}(b^T\delta)s + (s^T\delta)(b^T\delta)s \\ &\quad + (\frac{1}{2}b + \frac{\gamma}{2}\hat{\beta}s)(s^T\delta)^2 + \frac{\gamma}{6}(s^T\delta)^3 s). \end{aligned} \tag{3.12}$$

Premultiplying (3.12) by s^T on both sides results in a cubic equation (in β) in the two unknowns $\beta = s^T\delta$ and $\theta = b^T\delta$:

$$\begin{aligned} & s^T\hat{H}^{-1}\hat{g} + (1 + \hat{\beta}s^T\hat{H}^{-1}b)\beta + \hat{\beta}s^T\hat{H}^{-1}s\theta + s^T\hat{H}^{-1}s\beta\theta \\ & + (\frac{1}{2}s^T\hat{H}^{-1}b + \frac{\gamma}{2}\hat{\beta}s^T\hat{H}^{-1}s)\beta^2 + \frac{\gamma}{6}s^T\hat{H}^{-1}s\beta^3 = 0. \end{aligned} \tag{3.13}$$

The premultiplication of (3.12) by b^T on both sides yields another cubic equation (in β) in the two unknowns β and θ:

$$\begin{aligned} & b^T\hat{H}^{-1}\hat{g} + (1 + \hat{\beta}b^T\hat{H}^{-1}s)\theta + \hat{\beta}b^T\hat{H}^{-1}b\beta + b^T\hat{H}^{-1}s\beta\theta \\ & + (\frac{1}{2}b^T\hat{H}^{-1}b + \frac{\gamma}{2}\hat{\beta}b^T\hat{H}^{-1}s)\beta^2 + \frac{\gamma}{6}b^T\hat{H}^{-1}s\beta^3 = 0. \end{aligned} \tag{3.14}$$

Therefore, we obtain a system of two cubic equations in the two unknowns β and θ, which we can solve analytically.

Since (3.13) is linear in θ, we can compute θ as a function of β and then substitute its expression into (3.14) to obtain an equation in the one unknown β. Let $u = s^T\hat{H}^{-1}\hat{g}$, $v = s^T\hat{H}^{-1}b$, $w = s^T\hat{H}^{-1}s$, $y = b^T\hat{H}^{-1}\hat{g}$, and $z = b^T\hat{H}^{-1}b$. Equation (3.13) yields

$$\begin{aligned}\theta = \frac{1}{w(\hat{\beta}+\beta)}&(yw\hat{\beta} - u - uv\hat{\beta} + (yw + zw\hat{\beta}^2 - 2v\hat{\beta} - v^2\hat{\beta}^2 - uv - 1)\beta \\ &+(\frac{3}{2}zw\hat{\beta} - \frac{\gamma}{2}w\hat{\beta} - \frac{3}{2}v - \frac{3}{2}v^2\hat{\beta}) + (\frac{1}{2}zw - \frac{\gamma}{6}w - \frac{v^2}{2})\beta^3).\end{aligned} \tag{3.15}$$

The denominator of (3.15) is equal to zero if either $\hat{\beta}+\beta = 0$ or $w = 0$. If $w = 0$, then (3.13) would be quadratic in β. Therefore

$$\beta = \frac{-(1+\hat{\beta}v) \pm \sqrt{(1+\hat{\beta}v)^2 - 2uv}}{v}.$$

Hence, real-valued minimizers of the tensor model (3.8) may exist only if $(1+\hat{\beta}v)^2 \geq 2uv$ and $v \neq 0$. It is straightforward to verify from (3.14) that for θ to be defined $(\hat{\beta}+\beta)v$ cannot equal -1. Now, if $\hat{\beta}+\beta = 0$, then (3.13) reduces to the following cubic equation in β:

$$u + (1+\hat{\beta}v)\beta + (\frac{1}{2}v + \frac{\gamma}{2}w\hat{\beta})\beta^2 + \frac{\gamma}{6}w\beta^3 = 0. \tag{3.16}$$

Once we calculated the expressions for β from (3.16), we substitute them into the following equation for θ obtained from (3.14):

$$\theta = -y - z\hat{\beta}\beta - (\frac{1}{2}z + \frac{\gamma}{2}v\hat{\beta})\beta^2 - \frac{\gamma}{6}v\beta^3.$$

If neither $\hat{\beta}+\beta = 0$ nor $w = 0$, we substitute the expression (3.15) into (3.14) and obtain

$$\begin{aligned}&-(u + 2\hat{\beta}v + \hat{\beta}uv + \hat{\beta}^2v^2 + 1) + (yw + \hat{\beta}^2zw - \hat{\beta}v - v - uv)\beta \\ &+(\hat{\beta}^2zw + \frac{1}{2}\hat{\beta}zw - \frac{1}{2}v - \frac{\gamma}{2}\hat{\beta}w - \frac{1}{2}\hat{\beta}v^2)\beta^2 + (\frac{1}{2}zw - \frac{\gamma}{6}w - \frac{1}{2}v^2)\beta^3 = 0,\end{aligned} \tag{3.17}$$

which is a third-order polynomial in the one unknown β. The roots of (3.17) are then computed analytically. After we determine the values of β, we substitute them into (3.15) to calculate the corresponding values of θ. Then, we simply substitute the values of β and θ into (3.12) to obtain the values of δ. The dominant cost in this whole process is the computation of $\hat{H}^{-1}\hat{g}$, $\hat{H}^{-1}b$, and $\hat{H}^{-1}s$.

Similar to the nonsingular case, a minimizer δ is selected such that there exists a descent path from the current point x_c to $x_c+\delta$, and that δ is closest to x_c in the Euclidean norm sense.

To obtain the tensor step d, we set d to $\hat{d}+\delta$. An appropriate choice of $\hat{d}$ is the step used in the previous iteration simply because it has the right scale.

The above procedure is tailored to handle only the case where the Hessian matrix has rank $n-1$. It has been shown in practice that when $\nabla^2 f(x_*)$ has rank $n-1$ the convergence rate of the tensor method is better than the linear convergence of the standard Newton method [19] (also see §7 for the ratios of the errors of successive iterates on the BRYBND problem with $\text{rank}(\nabla^2 f(x_*)) = n-1$. Tensor methods for nonlinear equations problems have been shown to have 3-step Q-order 1.5 convergence on problems where the Jacobian has rank $n-1$ at the solution [11], whereas Newton's method is linearly convergent with constant 1/2 on such problems. However, no attempt has been made yet to prove the convergence rate of tensor methods for unconstrained optimization problems where the Hessian at the solution has rank $n-1$. On problems where $\text{rank}(\nabla^2 f(x_*)) \leq n-2$, tensor methods do not have enough information to prove faster-than-linear convergence rate, since it usually uses $p = 1$. Consequently, when $\text{rank}(\nabla^2 f(x_*)) \leq n-2$ we simply use the modified Newton step (see §6) as the step direction for the current iteration.

4. Line Search Backtracking Techniques. The line search global strategy we use in conjunction with our tensor method for large, sparse unconstrained optimization is similar to the one used for nonlinear equations [4, 6]. This strategy has shown to be very successful for large, sparse systems of nonlinear equations. We also found that it is superior to the approach used by Schnabel and Chow [19]. The main difference between the two approaches is that ours always tries the full tensor step first. If this provides enough decrease in the objective function, then we terminate; otherwise we find acceptable next iterates in both the Newton and tensor directions and select the one with the lower function value as the next iterate. Schnabel and Chow, on the other hand, always find acceptable next iterates in both the Newton and tensor directions and choose the one with the lower function value as the next iterate. In practice, our approach almost always requires fewer function evaluations while retaining the same efficiency in iteration numbers. The global framework for line search methods for unconstrained minimization is given in Algorithm 4.1.

Algorithm 4.1. Global Framework for Line Search Methods for Unconstrained Minimization

Let x_c be the current iterate,
d_t the tensor step,
d_n is the Newton step,
$g = \nabla f(x_c)$,
$f_c = f(x_c)$,
$slope = g^T d_t$,
and $\alpha = 10^{-4}$.

    $x_+^t = x_c + d_t$
    $f_p = f(x_+^t)$
    if (minimizer of the tensor model was found) **then**
        if $f_p < f_c + \alpha \cdot slope$ **then**
            $x_+ = x_+^t$
        else
            Find an acceptable x_+^n in the Newton direction d_n
            using the line search given by Algorithm A6.3.1 [9, p.325]
            Find an acceptable x_+^t in the tensor direction d_t
            using the line search given by Algorithm A6.3.1 [9, p.325]
            if $f(x_+^n) < f(x_+^t)$ **then**
                $x_+ = x_+^n$

else
$x_+ = x_+^t$
endif
endif
else
Find an acceptable x_+^n in the Newton direction d_n
using the line search given by Algorithm A6.3.1 [9, p.325]
$x_+ = x_+^n$
endif

5. Model Trust Region Techniques. The two computational methods—the locally constrained optimal (or "hook") step and the dogleg step—are generally used for approximately solving the trust region problem based on the standard model,

$$\text{minimize } f(x_c) + \nabla f(x_c) \cdot d + \frac{1}{2}\nabla^2 f(x_c) \cdot d^2 \tag{5.1}$$

$$\text{subjectto } \| d \|_2 \leq \delta_c,$$

where δ_c is the current trust region radius. When δ_c is shorter than the Newton step, the locally constrained optimal step [17] finds a μ_c such that $\| d(\mu_c) \|_2 \approx \delta_c$, where $d(\mu_c) = -(\nabla^2 f(x_c) + \mu I)^{-1}\nabla f(x_c)$. Then it takes $x_+ = x_c + d(\mu_c)$. The dogleg step is a modification of the trust region algorithm introduced by Powell [18]. However, rather than finding a point $x_+ = x_c + d(\mu_c)$ on the curve $d(\mu_c)$ such that $\| x_+ - x_c \| \approx \delta_c$, it approximates this curve by a piecewise linear function in the subspace spanned by the Newton step and the steepest descent direction $-\nabla f(x_c)$, and takes x_+ as the point on this approximation for which $\| x_+ - x_c \| = \delta_c$. (See, e.g., [9] for more details.)

Unfortunately these two methods are hard to extend to the tensor model, which is a fourth-order model. Trust region algorithms based on (5.1) are well defined because it is always possible to find a unique point x_+ on the curve such that $\| x_+ - x_c \| = \delta_c$. Additionally, the value of $f(x_c) + \nabla f(x_c) \cdot d + \frac{1}{2}\nabla^2 f(x_c) \cdot d^2$ along the curve $d(\mu_c)$ is monotonically decreasing from x_c to x_+^n, where $x_+^n = x_c + d_n$, which makes the process reasonable. These properties do not extend to the tensor model, which is a fourth-order model that may not be convex. Furthermore, the analogous curve to $d(\mu_c)$ is more expensive to compute. For these reasons, we consider a different trust region approach for our tensor methods.

The trust region approach that is discussed in this section is a two-dimensional trust region step over the subspace spanned by the steepest descent direction and the tensor (or standard) step. The main reasons that lead us to adopt this approach are that it is easy to construct, and closely related to dogleg type algorithms over the same subspace. This step may be close to optimal trust region step algorithms in practice. Byrd, Schnabel, and Shultz [7] have shown that for unconstrained optimization using a standard quadratic model, the analogous two-dimensional minimization approach produces nearly as much decrease in the quadratic model as the optimal trust region step in almost all cases.

The two-dimensional trust region approach for the tensor model computes an approximate solution to

$$\text{minimize } f(x_c) + \nabla f(x_c) \cdot d + \frac{1}{2}\nabla^2 f(x_c) \cdot d^2 + \frac{1}{2}(b^T d)(s^T d)^2 + \frac{\gamma}{24}(s^T d)^4$$

$$\text{subject to } \| d \|_2 \le \delta_c,$$

by performing a two-dimensional minimization,

$$\text{minimize } f(x_c) + \nabla f(x_c) \cdot d + \frac{1}{2}\nabla^2 f(x_c) \cdot d^2 + \frac{1}{2}(b^T d)(s^T d)^2 + \frac{\gamma}{24}(s^T d)^4 \tag{5.2}$$

$$\text{subject to } \| d \|_2 \le \delta_c, \quad d \in [d_t, g_s],$$

where d_t and g_s are the tensor step and the steepest descent direction, respectively, and δ_c is the trust region radius. This approach will always produce a step that reduces the quadratic model by at least as much as a dogleg-type algorithm, which reduces d to a piecewise linear curve in the same subspace. At each iteration of the tensor algorithm, the trust region method either solves (5.2) or minimizes the standard linear model over the two-dimensional subspace spanned by the standard Newton step and the steepest descent direction. The decision of whether to use the tensor or standard model is made using the following criterion:

if ((no minimizer of the tensor model was found)
  or $(\nabla f(x_c)^T d_t > -10^{-4} \| \nabla f(x_c) \|_2 \| d_t \|_2)$) **then**
  $x_+ = x_c + \alpha d_n - \beta g_s$; α, β selected by trust region algorithm
else
  $x_+ = x_c + \alpha d_t - \beta g_s$; α, β selected by trust region algorithm
endif

Before we define the two-dimensional trust region step for tensor methods, we show how to convert the problem

$$\text{minimize } f(x_c) + \nabla f(x_c) \cdot d + \frac{1}{2}\nabla^2 f(x_c) \cdot d^2 + \frac{1}{2}(b^T d)(s^T d)^2 + \frac{\gamma}{24}(s^T d)^4 \tag{5.3}$$

$$\text{subject to } \| d \|_2 = \delta_c, \quad d \in [\, d_t, g_s \,],$$

to an unconstrained minimization problem.

First, we make g_s orthogonal to d_t by performing the Householder transformation:

$$\hat{g_s} = g_s - d_t \frac{g_s^T d_t}{d_t^T d_t}; \tag{5.4}$$

then, we normalize both $\hat{g}_s$ and d_t to obtain

$$\tilde{d}_t = \frac{d_t}{\| d_t \|_2}, \tag{5.5}$$

$$\tilde{g_s} = \frac{\hat{g_s}}{\| \hat{g_s} \|_2}. \tag{5.6}$$

Since d is in the subspace spanned by the tensor step $\tilde{d}_t$ and the steepest descent direction $\tilde{g}_s$, it can be written as

$$d = \alpha \tilde{d}_t + \beta \tilde{g}_s, \quad \alpha, \beta \in \Re. \tag{5.7}$$

If we square the l_2 norm of this expression for d and set it to δ^2, we obtain the following equation for β as a function of α

$$\beta = \sqrt{\delta^2 - \alpha^2}.$$

Substituting this expression for β into (5.7) and then the resulting d into (5.3) yields the global minimization problem in the one variable α, given by (5.8) below. Thus, problems (5.8) and (5.3) are equivalent. Let $g_{hg} = \tilde{g}_s^T H \tilde{g}_s$, $d_{hd} = \tilde{d}_t^T H \tilde{d}_t$, $d_{hg} = \tilde{d}_t^T H \tilde{g}_s$, $b_t = b^T \tilde{d}_t$, $s_t = s^T \tilde{d}_t$, $b_g = b^T \tilde{g}_s$, and $s_g = s^T \tilde{g}_s$.

$$
\begin{aligned}
\text{minimize} \quad f(x_c) \quad & +\frac{1}{2}\delta_c^2 g_{hg} + \frac{\gamma}{24}\delta_c^4 s_g^4 + (1+\delta_c^2 b_g s_g^2)\sqrt{\delta_c^2-\alpha^2} \\
& +(d_{hg} + \frac{\gamma}{6}\delta_c^2 s_t s_g^3)\alpha\sqrt{\delta_c^2-\alpha^2} + (b_t s_g s_t + b_g s_t^2 + b_t s_t s_g \\
& -b_g s_g^2)\alpha^2\sqrt{\delta_c^2-\alpha^2} + (\delta_c^2 b_g s_g s_t + \delta_c^2 b_t s_g^2 + \delta_c^2 b_g s_t s_g)\alpha \\
& +(\frac{1}{2}d_{hd} - \frac{1}{2}g_{hg} + \frac{1}{2}b_t s_t^2 + \frac{\gamma}{4}\delta_c^2 s_t^2 s_g^2 - \frac{\gamma}{12}\delta_c^2 s_g^4)\alpha^2 \\
& -(b_g s_g s_t + b_t s_g^2 + b_g s_t s_g)\alpha^3 + (\frac{\gamma}{24}s_t^4 - \frac{\gamma}{4}s_t^2 s_g^2 + \frac{\gamma}{24}s_g^4)\alpha^4 \\
& +(\frac{\gamma}{6}s_t^3 s_g - \frac{\gamma}{6}s_t s_g^3)\alpha^3\sqrt{\delta_c^2-\alpha^2},
\end{aligned} \tag{5.8}
$$

where $-\delta_c < \alpha < \delta_c$.

To transform the problem

$$
\text{minimize } f(x_c) + \nabla f(x_c)\cdot d + \frac{1}{2}\nabla^2 f(x_c)\cdot d^2 \tag{5.9}
$$

$$
\text{subject to } \|\, d \,\|_2 = \delta_c, \quad d \in [\, d_n, g \,]
$$

to an unconstrained minimization problem, we use the same procedure described above to show that (5.9) is equivalent to the following global minimization problem in the one variable α:

$$
\begin{aligned}
\text{minimize} \qquad & f(x_c) + \frac{1}{2}\delta_c^2 g_{hg} + \sqrt{\delta_c^2-\alpha^2} \\
& + d_{hg}\alpha\sqrt{\delta_c^2-\alpha^2} + (\frac{1}{2}d_{hd} - \frac{1}{2}g_{hg})\alpha^2,
\end{aligned} \tag{5.10}
$$

where $-\delta_c < \alpha < \delta_c$.

Algorithm 5.1. Two-Dimensional Trust Region for Tensor Methods

Let d_t be the tensor step,
d_n the standard step,
x_c the current iterate,
$f_c = f(x_c)$,
x_+ the next iterate,
$f_+ = f(x_+)$,
$g_s = -\nabla f(x_c)$, the steepest descent direction,
$g_c = \nabla f(x_c)$,
$H_c = \nabla^2 f(x_c)$,
and δ_c the current trust region radius.
$\tilde{d}_t, \tilde{g}_s$ are given by (5.5) and (5.6), respectively.
$\tilde{d}_n$ is obtained in an analogous way to $\tilde{d}_t$; by applying transformations (5.4) and (5.5) to it.

1. **if** *tensor model* selected **then**
   Solve problem (5.8) using the procedure described in Algorithm 3.4 [6]
   else {*standard Newton model* `selected`}
   Solve problem (5.10) using the procedure described in Algorithm 3.4 [6]
   endif
2. **if** *tensor model* selected **then**
   $d = \alpha_* \tilde{d}_t + \tilde{g}_s \sqrt{\delta_c^2 - \alpha_*^2}$
   where α_* is the global minimizer of (5.8)
   else {*standard Newton model* `selected`}
   $d = \alpha_* \tilde{d}_n + \tilde{g}_s \sqrt{\delta_c^2 - \alpha_*^2}$
   where α_* is the global minimizer of (5.10)
   endif
3. { `Check new iterate and update trust region radius.`}
   $x_+ = x_c + d$
   if $\dfrac{f_+ - f_c}{pred} \geq 10^{-4}$ **then**
   the global step d is successful
   else
   decrease trust region
   go to step 1
   endif
   where
   $pred = (f_c + g_c \cdot d + \frac{1}{2} H_c \cdot d^2 + \frac{1}{2}(b^T d)(s^T d)^2 + \frac{\gamma}{24}(s^T d)^4) - f_c$, if *tensor model* selected,
   $pred = (f_c + g_c \cdot d + \frac{1}{2} H_c \cdot d^2) - f_c$, if *standard Newton model* selected.

The methods used for adjusting the trust radius during and between steps are given in Algorithm A6.4.5 [9, p.338]. The initial trust radius can be supplied by the user; if not, it is set to the length of the initial Cauchy step.

6. A High-Level Algorithm for the Tensor Method. In this section, we present the overall algorithm for the tensor method for large, sparse unconstrained optimization. Algorithm 6.1 is a high-level description of an iteration of the tensor method that was described in §§ 3—5. A summary of the test results for this implementation is presented in §7.

Algorithm 6.1. An Iteration of the Tensor Method for Large, Sparse Unconstrained Optimization

Let x_c be the current iterate,
d_t the tensor step,
and d_n the Newton step.

1. Calculate $\nabla f(x_c)$ and decide whether to stop. If not:
2. Calculate $\nabla^2 f(x_c)$.
3. Calculate the terms T_c and V_c in the tensor model, so that the tensor model interpolates $f(x)$ and $\nabla f(x)$ at the past point.
4. Find a potential minimizer d_t of the tensor model (3.1). If d_t cannot be found, then calculate the modified Newton step d_n.

5. Find an acceptable next iterate x_+ using either a line search or a two-dimensional trust region global strategy.
6. $x_c = x_+$,
   $f(x_c) = f(x_+)$,
   go to step 1.

In step 1, the gradient is either computed analytically or approximated by the algorithm A5.6.3 given in Dennis and Schnabel [9]. In step 2, the Hessian matrix is either calculated analytically or approximated by a graph coloring algorithm described in [8]. Note that it is crucial to supply an analytic gradient if the finite difference Hessian matrix requires many gradient evaluations. Otherwise, the methods described in this paper may not be practical, and inexact type of methods may be preferable. The procedures for calculating T_c and V_c in step 3 were discussed in §2. In step 4, the Hessian matrix is factored using MA27 [10], a sparse Cholesky decomposition package. If the Hessian matrix is nonsingular, then the tensor step d_t is calculated as described in §3.1. Otherwise, if the Hessian matrix is singular with rank $n-1$, then d_t is computed as outlined in §3.2. (We comment on the implementation issues related to this case in the next paragraph.) If the rank of the Hessian matrix is less than $n-1$, then the Newton step, d_n, is computed as a by-product of the minimization of the tensor model, and used as the step direction for the current iteration. This Newton step d_n is the modified Newton step $(\nabla^2 f(x_c)+\mu I)^{-1}\nabla f(x_c)$, where $\mu = 0$ if $\nabla^2 f(x_c)$ is safely positive definite, and $\mu > 0$ otherwise. To obtain the perturbation μ, we use a modification of MA27 advocated by Gill, Murray, Ponceleon, and Saunders in [13]. In this method we first compute the LDL^T of the Hessian matrix using the MA27 package, then change the block diagonal matrix D to $D+E$. The modified matrix is block diagonal positive definite. This guarantees that the decomposition $L(D+E)L^T$ is positive definite as well. Note that the Hessian matrix is not modified if it is already positive definite.

Another implementation issue that deserves some attention is how to solve linear systems of the form $\hat{H}x = b$, where $\hat{H} = H + css^T$, $H \in \Re^{n\times n}$ is sparse and rank deficient, and $s \in \Re^n$ is full, (see §3.2). Such systems can be efficiently solved using the augmented matrix defined in Lemma 3.1. That is, we write $(H+css^T)x = b$ as

$$\begin{bmatrix} H & cs \\ cs^T & -c \end{bmatrix} \begin{bmatrix} x \\ w \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}. \tag{6.1}$$

The $(n+1)\times(n+1)$ matrix in (6.1) is sparse and can be factored efficiently as long as the last row and column are not pivoted until the last few iterations. In fact, we can combine the nonsingular and singular cases by factoring H, but we shift to a factorization of the augmented matrix if H is discovered to be singular with rank $n-1$. However, we use a Schur complement method to obtain the solution of the augmented matrix by updating the solution from the system $Hx = b$. This choice was motivated by the fact that the Schur complement method is simpler and more convenient to use than the factorization of the augmented matrix in (6.1). Note that if the Schur complement method shows that the augmented matrix in (6.1) is rank deficient (a case that is very rare in practice), the modified Newton step described above is used as the step direction for the current iteration.

The Schur complement method requires that H must have full rank. Thus, some modifications are necessary in order for this method to work. We have modified the factorization phase of MA27 to be able to detect the row and column indices of the first pivot whose absolute value is less or equal than some given tolerance tol. This stability test is clearly not optimal but appears to work in practice. We also modified the solve phase of MA27 such that whenever a pivot fails the stability criterion above, the corresponding solution component is set to zero. This way the solution of $Hx = b$ is the same as the solution of $H_e y = b$ (where H_e is the matrix H minus the row and column at which singularity occurred. Since y has $n-1$ components, the remaining one, which is also the component corresponding to the pivot failing the stability test, is set to 0). Afterwards, we obtain the solution of an augmented system using a Schur complement method, where the coefficient matrix is the matrix H augmented by two rows and columns; that is, the $(n+1)$-st row and column are the ones at which singularity was detected, and the $(n+2)$-nd row and column are cs^T and cs, respectively. The Schur complement method is implemented by first invoking MA39AD [1] to form the Schur complement $S = D - CH^{-1}B$ of H in the extended matrix, where D is the 2 by 2 lower right submatrix, C is the lower left 2 by n submatrix, and B is the upper right n by 2 submatrix, of the augmented matrix. The Schur complement is then factored into its QR factors. Next, MA39BD [1] solves the extended system (6.1) using the following well-known scheme:

1. Solve $Hu = b$, for u.
2. Solve $Sy = b - Cu$, for y.
3. Solve $Hv = By$, for v.
4. $x = u - v$.

The dominant cost of the above process is the $Hu = b$ and $Hv = By$ solves.

The tensor and Newton algorithms terminate if $\| \nabla f(x_c) \|_2 \leq 10^{-5}$ or $\| d \|_2 < 10^{-9}$.

7. Test Results. We tested our tensor and Newton algorithms on a variety of nonsingular and singular test problems. In the following we present and discuss summary statistics of the test results.

All our computations were performed on a Sun Sparc 10 Model 40 machine using double-precision arithmetic.

First, we tested our program on the set of unconstrained optimization problems from the CUTE [3] and the MINPACK-2 [2] collections. Most of these problems have nonsingular Hessians at the solution. We also created singular test problems as proposed in [4, 20] by modifying the nonsingular test problems from the CUTE collection as follows. Let

$$f(x) = \sum_{i=1}^{m} f_i^2(x)$$

be the function to minimize, where $f_i : \Re^n \to \Re$ and m is the number of element functions, and

$$F^T(x) = (f_1(x), \cdots, f_m(x)). \tag{7.1}$$

In many cases, $F(x) = 0$ at the minimizer x_*, and $F'(x_*)$ is nonsingular. Then according to [4, 20], we can create singular systems of nonlinear equations from (7.1) by forming

$$\hat{F}(x) = F(x) - F'(x_*)A(A^TA)^{-1}A^T(x - x_*), \tag{7.2}$$

where $A \in \Re^{n\times k}$ has full column rank with $1 \le k \le n$. Hence, $\hat{F}(x_*) = 0$ and $\hat{F}'(x_*)$ has rank $n-k$. For unconstrained optimization, we simply need to define the singular function

$$\hat{f}(x) = \frac{1}{2}\hat{F}(x)^T \hat{F}(x). \tag{7.3}$$

From (7.3) and $\hat{F}(x_*) = 0$, we obtain $\nabla \hat{f}(x_*) = 0$. From

$$\hat{F}'(x_*) = F'(x_*)[I - A(A^T A)^{-1} A^T]$$

and

$$\nabla^2 \hat{f}(x_*) = \hat{F}'(x_*)^T \hat{F}'(x_*) + \sum_{i=1}^{m} f_i(x_*)\nabla^2 f_i(x_*) = \hat{F}'(x_*)^T \hat{F}'(x_*),$$

we know that $\nabla^2 \hat{f}(x_*)$ has rank $n-k$.

By using (7.2) and (7.3), we created two sets of singular problems, with $\nabla^2 \hat{f}(x_*)$ having rank $n-1$ and $n-2$, respectively, by using

$$A \in \Re^{n\times 1}, \qquad A^T = (1, 0, \cdots, 0),$$

and

$$A \in \Re^{n\times 2}, \qquad A^T = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & 1 & 0 & 0 & \cdot & \cdot & \cdot & 0 \end{bmatrix},$$

respectively. The reason for choosing unit vectors as columns for the matrix A is mainly to preserve the sparsity of the Hessian during the transformation (7.2).

For all our test problems we used a standard line search backtracking strategy. All the test problems with the exception of rank $n-1$ and rank $n-2$ problems were run with analytic gradients and Hessians provided by the CUTE and MINPACK-2 collections. For rank $n-1$ and $n-2$ test problems, we have modified the analytic gradients provided by the CUTE collection to take into account the modification (7.2). On the other hand, we used the graph coloring algorithm [8] to evaluate the finite difference approximation of the Hessian matrix.

A summary for the test problems whose Hessians at the solution have ranks n, $n-1$, and $n-2$ is presented in Table 1. The descriptions of the test problems and the detailed results are given in the Appendix. In Table 1 columns "better" and "worse" represent the number of times the tensor method was better and worse, respectively, than Newton's method by more than one gradient evaluation. The "tie" column represents the number of times the tensor and standard methods required within one gradient evaluation of each other. For each set of problems, we summarize the comparative costs of the tensor and standard methods using average ratios of three measures: gradient evaluations, function evaluations, and execution times. The average gradient evaluation ratio (geval) is the total number of gradients evaluations required by the tensor method, divided by the total number of gradients evaluations required by the standard method on these problems. The same measure is used for the average function evaluation (feval) and execution time (time) ratios. These average ratios include only problems that were successfully solved by both methods. We have excluded all cases where the tensor and standard methods converged to a different minimizer. However, the statistics for the "better," "worse," and "tie" columns include the cases where only one of the two methods converges, and exclude the cases where both methods do not converge. We also excluded problems requiring a number of gradient evaluations less or equal than 3 by both methods. Finally, columns "t/s" and "s/t" show the number of problems solved by the tensor method but not by the standard method and the number of problems solved by the standard method but not by the tensor method, respectively.

The improvement by the tensor method over the standard method on problems with rank $n-1$ is dramatic, averaging 48% in function evaluations, 52% in gradient evaluations, and 59% in execution times. This is due in part to the rate of convergence of the tensor method being faster than that of Newton's method, which is known to be only linearly convergent with constant $\frac{2}{3}$. On problems with rank $n-2$, the improvement by the tensor method over the standard method is also substantial, averaging 30% in function evaluations, 37% in gradient evaluations, and 34% in execution times. In the test results obtained for the nonsingular problems, the tensor method is 9% worse than the standard method in function evaluations, but 31% and 33% better in gradient evaluations and in execution times, respectively. The main reason for the tensor method requiring on the average more function evaluations than the standard method is because on some problems, the full tensor step does not provide sufficient decrease in the objective function, and therefore the tensor method has to perform a line search in both the Newton and tensor directions, which causes the number of function evaluations required by the tensor method to be inflated. As a result, we intend to investigate other possible global frameworks for line search methods that could potentially reduce the number of functions evaluations for the tensor method.

To obtain an experimental indication of the local convergence behavior of the tensor and Newton methods on problems where $\text{rank}(\nabla^2 f(x_*)) = n-1$, we examined the sequence of ratios

$$\frac{\| x_k - x_* \|}{\| x_{k-1} - x_* \|} \tag{7.4}$$

produced by the Newton and tensor methods on such problems. These ratios for a typical problem are given in Table 2. In almost all cases the standard method exhibits local linear convergence with constant near $\frac{2}{3}$, which is consistent with the theoretical analysis. The local convergence rate of the tensor method is faster, with a typical final ratio of around 0.01. Whether this is a superlinear convergence remains to be determined. We have done similar experiments for problems with $\text{rank}(\nabla^2 f(x_*)) = n-2$, and the tensor method did not show a faster-than-linear convergence rate, because it did not have enough information since $p = 1$.

The tensor method solved a total of four nonsingular problems, five rank $n-1$ problems, and 7 rank $n-2$ problems that Newton's method failed to solve. The reverse never occurred. These results clearly indicate that the tensor method is most likely to be more robust than Newton's method.

The overall results show that having some extra information about the function and gradient in the past step direction is quite useful in achieving the advantages of tensor methods.

8. Summary and Future Research. In this paper we presented new algorithms for solving large, sparse unconstrained optimization using tensor methods. Implementations using these tensor methods have been shown to be considerably

TABLE 7.1
Summary of the CUTE *and* MINPACK-2 *test problems using line search.*

Rank	Tensor/Standard			Pbs Solved		Average Ratio–Tensor/Standard		
$\nabla^2 f(x_*)$	better	tie	worse	t/s	s/t	feval	geval	time
n	53	38	5	4	0	1.09	0.69	0.67
$n-1$	18	2	0	5	0	0.52	0.48	0.41
$n-2$	18	1	1	7	0	0.70	0.63	0.66

TABLE 7.2
Speed of convergence on the BRYBND problem with rank($\nabla^2 f(x_*)) = n-1$, as modified by (7.2), $n = 5000$, started from x_0. The ratios in second and third columns are defined by (7.4).

Iteration (k)	Standard Method	Tensor Method
1	0.659	0.659
2	0.655	0.033
3	0.650	0.459
4	0.641	0.961
5	0.629	0.850
6	0.612	0.667
7	0.590	0.410
8	0.571	0.323
9	0.600	0.126
10	0.760	0.012
11	0.940	
12	0.988	
13	0.970	
14	0.969	
15	0.956	
16	0.926	
17	0.891	
18	0.909	
19	0.848	
20	0.926	
21	0.939	
22	0.896	
23	0.832	
24	0.871	
25	0.742	
26	0.667	
27	0.667	
28	0.666	
29	0.665	
30	0.666	

more efficient especially on problems where the Hessian matrix has a small rank deficiency at the solution. Typical gains over standard Newton methods range from 40% to 50% in function and gradient evaluations and in computer time. The size and consistency of the efficiency gains indicate that the tensor method may be preferable to Newton's method for solving large, sparse unconstrained optimization problems where analytic gradients and/or Hessians are available. To firmly establish such a conclusion, additional testing is required, including test problems of very large size.

On sparse problems where the function or the gradient is expensive to evaluate, the finite difference approximation of the Hessian matrix by the graph coloring algorithm [8] may be very costly. Hence, quasi-Newton methods may be preferable to use in this case. These methods involve low-rank corrections to a current approximate Hessian matrix. We are currently attempting to extend our tensor methods to quasi-Newton methods for large, sparse unconstrained minimization problems.

We also considered solving large, sparse, structured unconstrained optimization problems using tensor methods. In this variant, we explored the possibility of using exact third- and fourth-order derivative information. The calculation of these derivatives is simplified using the concept of *partial separability*, a structure that has already proven to be useful when building quadratic models for large-scale nonlinear problems [16]. The calculation of the minimizer of this *exact* tensor model is more problematic, however, because we need to solve a sparse system of nonlinear equations. An obvious approach to solve these equations is to use a Newton-like method. Such a method is characterized by the approximation of the Jacobian used in the Newton process. A simple idea is to use a fixed Jacobian at each step. This has the advantage that the Jacobian will have already been obtained in the current tensor iteration. However, potential slow convergence of such a scheme may make the cost of a tensor iteration prohibitive. We are currently investigating other possible approaches, such as a modified Newton's method in which the approximated Jacobian matrix will incorporate more useful information, or an iterative method such as a nonlinear GMRES. This work, a cooperation with Nick Gould [5], will be reported in the near future.

We are almost done with the implementation and testing of the two-dimensional trust region global strategy described in §5. This work will be reported in a forthcoming paper.

We are also implementing the algorithms discussed in this paper in a software package. This package uses one past point in the formation of the tensor terms, which makes the additional cost and storage of the tensor method over the standard method very small. The package will be available soon.

Acknowledgments. The author thanks Professor Bobby Schnabel for his suggestions on how to minimize the tensor model when the Hessian is rank deficient, Nick Gould for discussing a number of implementation issues, Ta-Tung Chow for reviewing the first draft of the paper, and Jacko Koster for his numerous suggestions. He also thanks the referees, and Gail Pieper from the MCS division at Argonne National Laboratory for their suggestions for improvement.

REFERENCES

[1] Anon, *Harwell subroutine library (Release 11)*, Theoretical Studies Department, AEA Industrial Technology, 1993.

[2] B. M. Averick, R. G. Carter, J. J. Moré, and G. L. Xue, *The MINPACK-2 test problem collection*, Technical Report ANL/MCS-P153-0692, Argonne National Laboratory, 1992.

[3] I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint, CUTE*: Constrained and Unconstrained Testing Environment*, ACM Trans. Math. Software, 21(1):123–160, 1995.

[4] A. Bouaricha, *Solving large sparse systems of nonlinear equations and nonlinear least squares problems using tensor methods on sequential and parallel computers*, Ph.D. thesis, Computer Science Department, University of Colorado at Boulder, 1992.

[5] A. Bouaricha and N. I. M. Gould, *Personal communication*, Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique (CERFACS), Toulouse, France, 1994.

[6] A. Bouaricha and R. B. Schnabel, TENSOLVE*: A software package for solving systems of nonlinear equations and nonlinear least squares problems using tensor methods*, Preprint MCS-P463-0894, Mathematics and Computer Science Division, Argonne National Laboratory, 1994.

[7] R. H. Byrd, R. B. Schnabel, and G. A. Shultz, *Approximation solution of the trust region problem by minimization over two-dimensional subspaces*, Math. Programming, 40:247–263, 1988.

[8] T. F. Coleman, B. S. Garbow, and J. J. Moré, *Estimating sparse Hessian matrices*, ACM Trans. Math. Software, 11:363–377, 1985.

[9] J. E. Dennis and R. B. Schnabel *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice-Hall, Englewood Cliffs, N.J., 1983.

[10] I. S. Duff and J. K. Reid, *MA27: A set of Fortran subroutines for solving sparse symmetric sets of linear equations*, Technical Report R-10533, AERE Harwell Laboratory, Harwell, UK, 1983.

[11] D. Feng, P. Frank, and R. B. Schnabel, *Local convergence analysis of tensor methods for nonlinear equations*, Math. Prog., 62:427–459, 1993.

[12] R. Fletcher, *Practical method of optimization*, volume 1, *Unconstrained Optimization*, John Wiley and Sons, New York, 1980.

[13] P. E. Gill, W. Murray, D. B. Ponceleon, and M. A. Saunders, *Preconditioners for indefinite systems arising in optimization and nonlinear least squares problems*, Technical Report SOL 90-8, Department of Operations Research, Stanford University, California, 1990.

[14] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.

[15] A. Griewank and M. R. Osborne, *Analysis of Newton's method at irregular singularities*, SIAM J. Numer. Anal., 18:145–150, 1981.

[16] A. Griewank and Ph. L. Toint, *On the unconstrained optimization of partially separable functions*, In M. J. D. Powell, editor, *Nonlinear Optimization 1981*, pages 301–312. Academic Press, New York, 1982.

[17] J. J. Moré, *The Levenberg-Marquardt algorithm: Implementation and theory*, In G. A. Watson, editor, *Proceedings Dundee 1977,* Lecture Notes in Mathematics 630, pages 105–116. Springer Verlag, Berlin, 1978.

[18] M. J. D. Powell, *A new algorithm for unconstrained optimization*, In J. B. Rosen, O. L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 33–65. Academic Press, New York, 1970.

[19] R. B. Schnabel and T. Chow, *Tensor methods for unconstrained optimization using second derivatives*, SIAM J. Optimization, 1:293–315, 1991.

[20] R. B. Schnabel and P. D. Frank, *Tensor methods for nonlinear equations*, SIAM J. Numer. Anal., 21:815–843, 1984.

Appendix A. The columns in Tables A-3—A-6 have the following meanings:

- *func*: name of the problem.
- n: dimension of the problem.
- x_0: starting point. 1, 10, 100 stand for x_0, $10x_0$, and $100x_0$, respectively.
- *initf*: initial value of the objective function.
- *fcn*: number of function evaluations.
- *grad*: number of gradient evaluations.
- *time*: execution time in seconds.
- *finalf*: final value of the objective function.

IL, NC stand for iteration limit exceeded and convergence to a nonminimizer, respectively. The iteration limit is 300 for the MINPACK-2 collection and 200 for the CUTE collection. All starting points were provided by the MINPACK-2 and CUTE collections.

Remark: For rank $n-1$ and $n-2$ problems *grad* does not include the number of gradients required by Hessian evaluations. On the other hand, *fcn* does include the functions evaluations required by Hessian evaluations.

TABLE A-1
MINPACK-2 *test problems*

Name	Description
DEPT	Elastic-plastic torsion problem
DGL1	Ginzburg-Landau (1-dimensional) superconductivity problem
DGL2	Ginzburg-Landau (2-dimensional) superconductivity problem
DLJ2	2-dimensional Leonard-Jones clusters (molecular conformation) problem
DLJ3	3-dimensional Leonard-Jones clusters (molecular conformation) problem
DMSA	Minimal surface area problem
DODC	Optimal design with composite materials problem
DPJB	Pressure distribution in a journal bearing problem
DSSC	Steady state combustion problem

Table A-2
CUTE *test problems*

Name	Description
ARWHEAD	Quartic problem whose Hessian is an arrow-head (downwards) with diagonal central part and border-width 1
BDQRTIC	Quartic problem whose Hessian is banded with bandwidth 9
BRYBND	Broyden banded system of nonlinear equations, considered in the least square sense
DIXMAANA	Dixon-Maany test problem (version A)
DIXMAANB	Dixon-Maany test problem (version B)
DIXMAANC	Dixon-Maany test problem (version C)
DIXMAANI	Dixon-Maany test problem (version I)
DIXON3DQ	Dixon's tridiagonal quadratic
EDENSCH	Extended Dennis and Schnabel problem, as defined by Li
ENGVAL1	A sum of $2n-2$ groups, $n-1$ of which contain 2 nonlinear elements
FLETCBV2	Boundary Value problem
FREUROTH	Freudenstein and Roth test problem
LIARWHD	A simplified version of the NONDIA problem
MOREBV	Boundary Value problem. This is the nonlinear least-squares version without fixed variables
NONDIA	Shanno's nondiagonal extension of Rosenbrock function
NONDQUAR	A nondiagonal quartic test problem with an arrow-head type Hessian having a tridiagonal central part and a border-width 1. The Hessian is singular at the solution
PENALTY1	A sum of $n+1$ least-squares groups, the first n which have only one linear element
PENALTY2	A nonlinear least-squares problem with $m = 2n$ groups, group 1 is linear, groups 2 to n use 2 nonlinear elements, groups $n+1$ to $m-1$ use 1 nonlinear element, and group m uses n nonlinear elements
POWELLSG	Extended Powell singular problem
QUARTC	A simple quartic function
SINQUAD	A function with nontrivial groups and repetitious elements
SROSENBR	Separable extension of Rosenbrock's function
TQUARTIC	A quartic function with nontrivial groups and repetitious elements
TRIDIA	Shanno's TRIDIA quadratic tridiagonal problem
WOODS	Extended Woods problem
WOODS1	Scaled extended Woods problem

Table A-3
Results of the MINPACK-2 *test problems*

				Standard				Tensor			
func	*n*	x_0	*initf*	*fcn*	*grad*	*finalf*	*time*	*fcn*	*grad*	*finalf*	*time*
DEPT	100	1	-0.36364D+01	2	2	-0.10694D+02	0.410D-01	2	2	-0.10694D+02	0.391D-01
DEPT	400	1	-0.36584D+01	2	2	-0.10902D+02	0.180D+00	2	2	-0.10902D+02	0.182D+00
DEPT	900	1	-0.36629D+01	2	2	-0.10946D+02	0.449D+00	2	2	-0.10946D+02	0.471D+00
DEPT	1600	1	-0.36645D+01	2	2	-0.10961D+02	0.900D+00	2	2	-0.10961D+02	0.900D+00
DEPT	2500	1	-0.36653D+01	2	2	-0.10969D+02	0.153D+01	2	2	-0.10969D+02	0.151D+01
DEPT	3600	1	-0.36657D+01	2	2	-0.10973D+02	0.239D+01	2	2	-0.10973D+02	0.236D+01
DEPT	4900	1	-0.36659D+01	2	2	-0.10976D+02	0.348D+01	2	2	-0.10976D+02	0.349D+01
DEPT	6400	1	-0.36661D+01	2	2	-0.10977D+02	0.478D+01	2	2	-0.10977D+02	0.483D+01
DEPT	8100	1	-0.36662D+01	2	2	-0.10978D+02	0.746D+01	2	2	-0.10978D+02	0.713D+01
DEPT	10000	1	-0.36663D+01	2	2	-0.10979D+02	0.833D+01	2	2	-0.10979D+02	0.831D+01
DGL1	100	1	-0.16619D-03	18	18	-0.84562D+04	0.410D+00	5	5	-0.84562D+04	0.110D+00
DGL1	400	1	-0.16619D-03	18	18	-0.84562D+04	0.173D+01	9	6	-0.84562D+04	0.620D+00
DGL1	900	1	-0.16619D-03	18	18	-0.84562D+04	0.397D+01	6	6	-0.84562D+04	0.129D+01
DGL1	1600	1	-0.16619D-03	18	18	-0.84562D+04	0.706D+01	7	7	-0.84562D+04	0.282D+01
DGL1	2500	1	-0.16619D-03	18	18	-0.84562D+04	0.110D+02	8	8	-0.84562D+04	0.512D+01
DGL1	3600	1	-0.16619D-03	19	19	-0.84562D+04	0.169D+02	9	9	-0.84562D+04	0.847D+01
DGL1	4900	1	-0.16619D-03	19	19	-0.84562D+04	0.230D+02	7	7	-0.84562D+04	0.860D+01
DGL1	6400	1	-0.16619D-03	17	17	-0.84413D+04	0.270D+02	7	7	-0.84562D+04	0.115D+02
DGL1	8100	1	-0.16619D-03	–	NC	–	–	7	7	-0.84562D+04	0.149D+02
DGL1	10000	1	-0.16619D-03	–	NC	–	–	9	9	-0.84562D+04	0.236D+02
DGL2	100	1	0.18190D+02	231	84	0.16228D+02	0.113D+02	150	38	0.16228D+02	0.531D+01
DGL2	400	1	0.20131D+02	159	67	0.16231D+02	0.450D+02	210	43	0.16231D+02	0.307D+02
DGL2	900	1	0.22015D+02	265	96	0.16232D+02	0.202D+03	418	76	0.16232D+02	0.169D+03
DGL2	1600	1	0.23884D+02	306	111	0.16232D+02	0.584D+03	455	81	0.16232D+02	0.444D+03
DGL2	2500	1	0.25748D+02	354	122	0.16232D+02	0.133D+04	607	102	0.16232D+02	0.117D+04
DGL2	3600	1	0.27609D+02	503	165	0.16232D+02	0.314D+04	751	137	0.16232D+02	0.219D+04
DGL2	4900	1	0.29469D+02	686	223	0.16232D+02	0.128D+05	849	144	0.16232D+02	0.644D+04
DLJ2	100	1	-0.10698D+03	252	107	-0.13375D+03	0.113D+03	176	51	-0.13396D+03	0.544D+02
DLJ2	200	1	-0.22945D+03	405	132	-0.28056D+03	0.103D+04	475	89	-0.28140D+03	0.698D+03
DLJ2	300	1	-0.35261D+03	544	145	-0.44216D+03	0.372D+04	631	118	-0.44025D+03	0.305D+04
DLJ3	120	1	-0.11782D+03	375	112	-0.17954D+03	0.137D+03	348	65	-0.17073D+03	0.805D+02
DLJ3	210	1	-0.23253D+03	485	139	-0.34073D+03	0.838D+03	608	113	-0.34522D+03	0.687D+03
DLJ3	360	1	-0.42908D+03	1031	281	-0.63744D+03	0.826D+04	963	173	-0.63311D+03	0.466D+04
DMSA	100	1	0.14608D+01	4	4	0.14185D+01	0.150D+00	4	4	0.14185D+01	0.160D+00
DMSA	400	1	0.14891D+01	4	4	0.14206D+01	0.640D+00	10	4	0.14206D+01	0.710D+00
DMSA	900	1	0.15035D+01	5	5	0.14210D+01	0.212D+01	4	4	0.14210D+01	0.172D+01
DMSA	1600	1	0.15123D+01	5	5	0.14212D+01	0.396D+01	10	5	0.14212D+01	0.446D+01
DMSA	2500	1	0.15183D+01	6	6	0.14212D+01	0.833D+01	14	5	0.14212D+01	0.761D+01
DMSA	3600	1	0.15227D+01	6	6	0.14213D+01	0.130D+02	10	6	0.14213D+01	0.146D+02
DMSA	4900	1	0.15260D+01	6	6	0.14213D+01	0.190D+02	11	6	0.14213D+01	0.210D+02
DMSA	6400	1	0.15286D+01	7	7	0.14213D+01	0.308D+02	9	7	0.14213D+01	0.342D+02
DMSA	8100	1	0.15307D+01	17	12	0.14213D+01	0.846D+02	16	8	0.14213D+01	0.595D+02
DMSA	10000	1	0.15324D+01	21	14	0.14213D+01	0.117D+03	17	7	0.14213D+01	0.601D+02
DODC	100	1	0.44626D-01	14	8	-0.10980D-01	0.420D+00	16	8	-0.10980D-01	0.487D+00
DODC	400	1	0.47194D-01	13	10	-0.11248D-01	0.234D+01	19	10	-0.11248D-01	0.272D+01
DODC	900	1	0.47771D-01	23	13	-0.11329D-01	0.744D+01	41	14	-0.11329D-01	0.943D+01
DODC	1600	1	0.47974D-01	55	23	-0.11351D-01	0.256D+02	56	21	-0.11351D-01	0.267D+02
DODC	2500	1	0.48082D-01	70	33	-0.11359D-01	0.617D+02	117	28	-0.11359D-01	0.623D+02
DODC	3600	1	0.48139D-01	129	49	-0.11368D-01	0.148D+03	194	42	-0.11368D-01	0.144D+03
DODC	4900	1	0.48178D-01	565	163	-0.11372D-01	0.713D+03	406	76	-0.11372D-01	0.380D+03
DODC	6400	1	0.48202D-01	597	168	-0.11374D-01	0.999D+03	526	94	-0.11374D-01	0.640D+03
DODC	8100	1	0.48221D-01	–	IL	–	–	–	IL	–	–
DODC	10000	1	0.48234D-01	–	IL	–	–	–	IL	–	–
DPJB	100	1	0.11274D+02	2	2	-0.27881D+00	0.488D-01	2	2	-0.27881D+00	0.508D-01
DPJB	400	1	0.13331D+02	2	2	-0.28144D+00	0.209D+00	2	2	-0.28144D+00	0.201D+00
DPJB	900	1	0.14544D+02	2	2	-0.28219D+00	0.500D+00	2	2	-0.28219D+00	0.490D+00
DPJB	1600	1	0.15545D+02	2	2	-0.28249D+00	0.939D+00	2	2	-0.28249D+00	0.959D+00
DPJB	2500	1	0.16462D+02	2	2	-0.28264D+00	0.150D+01	2	2	-0.28264D+00	0.160D+01
DPJB	3600	1	0.17336D+02	2	2	-0.28272D+00	0.243D+01	2	2	-0.28272D+00	0.256D+01
DPJB	4900	1	0.18186D+02	2	2	-0.28277D+00	0.374D+01	2	2	-0.28277D+00	0.362D+01
DPJB	6400	1	0.19022D+02	2	2	-0.28280D+00	0.496D+01	2	2	-0.28280D+00	0.489D+01
DPJB	8100	1	0.19848D+02	2	2	-0.28282D+00	0.733D+01	2	2	-0.28282D+00	0.741D+01
DPJB	10000	1	0.20666D+02	2	2	-0.28284D+00	0.878D+01	2	2	-0.28284D+00	0.862D+01
DSSC	100	1	-0.52548D+01	3	3	-0.55979D+01	0.110D+00	3	3	-0.55979D+01	0.120D+00
DSSC	400	1	-0.50507D+01	3	3	-0.56077D+01	0.510D+00	3	3	-0.56077D+01	0.540D+00
DSSC	900	1	-0.49189D+01	3	3	-0.56098D+01	0.120D+01	3	3	-0.56098D+01	0.131D+01
DSSC	1600	1	-0.48224D+01	3	3	-0.56105D+01	0.229D+01	3	3	-0.56105D+01	0.246D+01
DSSC	2500	1	-0.47466D+01	3	3	-0.56108D+01	0.382D+01	3	3	-0.56108D+01	0.413D+01
DSSC	3600	1	-0.46842D+01	3	3	-0.56110D+01	0.595D+01	3	3	-0.56110D+01	0.624D+01
DSSC	4900	1	-0.46312D+01	3	3	-0.56112D+01	0.880D+01	3	3	-0.56112D+01	0.913D+01
DSSC	6400	1	-0.45852D+01	3	3	-0.56112D+01	0.115D+02	3	3	-0.56112D+01	0.122D+02
DSSC	8100	1	-0.45445D+01	3	3	-0.56113D+01	0.173D+02	3	3	-0.56113D+01	0.179D+02
DSSC	10000	1	-0.45080D+01	2	2	-0.56113D+01	0.102D+02	2	2	-0.56113D+01	0.102D+02

TABLE A-4

Results of the CUTE *test problems*

				Standard				Tensor			
func	*n*	x_0	*initf*	*fcn*	*grad*	*finalf*	*time*	*fcn*	*grad*	*finalf*	*time*
ARWHEAD	5000	1	0.14997D+05	7	7	0.00000D+00	0.496D+02	3	3	0.00000D+00	0.168D+02
		10	0.19978D+09	12	12	0.00000D+00	0.909D+02	18	14	0.00000D+00	0.110D+03
		100	0.19996D+13	18	18	0.00000D+00	0.140D+03	33	20	0.00000D+00	0.160D+03
BDQRTIC	1000	1	0.22510D+06	10	10	0.39838D+04	0.992D+01	24	12	0.39838D+04	0.127D+02
		10	0.22424D+10	16	16	0.39838D+04	0.165D+02	38	17	0.39838D+04	0.185D+02
		100	0.22410D+14	22	22	0.39838D+04	0.231D+02	51	23	0.39838D+04	0.254D+02
BRYBND	5000	1	0.12490D+06	24	17	0.13587D-19	0.327D+02	49	16	0.12928D-16	0.381D+02
		10	0.10765D+12	37	26	0.14231D-19	0.510D+02	50	24	0.98532D-17	0.551D+02
		100	0.12303D+18	–	IL	–	–	810	189	0.35466D-16	0.473D+03
DIXON3DQ	5000	1	0.80000D+01	2	2	0.11414D-24	0.600D+00	2	2	0.11414D-24	0.560D+00
		10	0.24200D+03	2	2	0.34514D-23	0.570D+00	2	2	0.34514D-23	0.570D+00
		100	0.20402D+05	2	2	0.29050D-21	0.560D+00	2	2	0.29050D-21	0.560D+00
DIXMAANA	3000	1	0.20501D+05	6	6	0.10000D+01	0.165D+01	8	6	0.10000D+01	0.205D+01
		10	0.80013D+10	18	12	0.10000D+01	0.366D+01	19	12	0.10000D+01	0.455D+01
		100	0.80000D+16	29	21	0.10000D+01	0.654D+01	19	19	0.10000D+01	0.724D+01
DIXMAANB	3000	1	0.43242D+05	6	6	0.10000D+01	0.162D+01	15	6	0.10000D+01	0.218D+01
		10	0.17227D+11	–	IL	–	–	–	IL	–	–
		100	0.16116D+17	–	IL	–	–	–	IL	–	–
DIXMAANC	3000	1	0.74483D+05	15	15	0.10000D+01	0.450D+01	15	13	0.10000D+01	0.506D+01
		10	0.34452D+11	–	IL	–	–	–	IL	–	–
		100	0.32233D+17	–	IL	–	–	–	IL	–	–
DIXMAANI	3000	1	0.12022D+05	100	33	0.10000D+01	0.119D+02	108	18	0.10000D+01	0.907D+01
		10	0.80004D+10	184	58	0.10000D+01	0.217D+02	152	32	0.10000D+01	0.157D+02
		100	0.80000D+16	263	77	0.10000D+01	0.287D+02	247	41	0.10000D+01	0.209D+02
EDENSCH	2000	1	0.73583D+07	13	13	0.12003D+05	0.442D+01	31	16	0.12003D+05	0.666D+01
		10	0.15184D+12	19	19	0.12003D+05	0.666D+01	53	20	0.12003D+05	0.877D+01
		100	0.16253D+16	24	24	0.12003D+05	0.848D+01	48	25	0.12003D+05	0.106D+02
ENGVAL1	5000	1	0.29494D+06	8	8	0.55487D+04	0.536D+01	7	7	0.55487D+04	0.548D+01
		10	0.31990D+10	14	14	0.55487D+04	0.983D+01	27	14	0.55487D+04	0.124D+02
		100	0.31994D+14	20	20	0.55487D+04	0.143D+02	49	20	0.55487D+04	0.186D+02
FLETCBV2	10000	1	-0.50013D+00	1	1	0.00000D+00	0.460D+00	1	1	0.00000D+00	0.380D+00
		10	0.39995D+02	2	2	-0.50013D+00	0.207D+01	2	2	-0.50013D+00	0.215D+01
		100	0.48995D+04	2	2	-0.50013D+00	0.212D+01	2	2	-0.50013D+00	0.212D+01
FREUROTH	5000	1	0.50486D+07	461	83	0.60793D+06	0.956D+02	424	53	0.60821D+06	0.785D+02
		10	0.15963D+09	444	77	0.60726D+06	0.894D+02	200	30	0.35200D+07	0.414D+02
		100	0.13056D+15	92	45	0.42206D+06	0.426D+02	155	51	0.53488D+06	0.605D+02
LIARWHD	10000	1	0.58500D+07	13	13	0.81983D-21	0.217D+03	13	9	0.49397D-27	0.148D+03
		10	0.97359D+11	22	21	0.63218D-17	0.363D+03	24	12	0.11125D-16	0.205D+03
		100	0.10189D+16	26	26	0.16259D-16	0.463D+03	48	18	0.31712D-21	0.319D+03
MOREBV	5000	1	0.15969D-06	2	2	0.58271D-14	0.100D+01	2	2	0.58271D-14	0.940D+00
		10	0.15983D-04	2	2	0.22833D-09	0.950D+00	2	2	0.22833D-09	0.960D+00
		100	0.17190D-02	2	2	0.32151D-04	0.910D+00	2	2	0.32151D-04	0.910D+00
NONDIA	10000	1	0.39996D+07	6	6	0.47632D-24	0.909D+02	10	5	0.11200D-20	0.737D+02
		10	0.12099D+11	34	34	0.53482D-25	0.595D+03	20	16	0.19919D-28	0.274D+03
		100	0.10200D+15	39	39	0.22382D-20	0.681D+03	52	21	0.65733D-17	0.367D+03
NONDQUAR	10000	1	0.10006D+05	20	20	0.41398D-09	0.965D+03	20	20	0.41413D-09	0.970D+03
		10	0.99981D+08	25	25	0.12450D-08	0.122D+04	25	25	0.12538D-08	0.123D+04
		100	0.99980D+12	31	31	0.73954D-09	0.152D+04	31	31	0.87210D-09	0.153D+04
PENALTY1	100	1	0.11448D+12	47	38	0.90255D-03	0.493D+01	10	7	0.90249D-03	0.780D+00
		10	0.11448D+16	51	43	0.90255D-03	0.557D+01	7	7	0.90249D-03	0.850D+00
		100	0.11448D+20	55	48	0.90257D-03	0.625D+01	30	16	0.90252D-03	0.213D+01
PENALTY2	100	1	0.16885D+07	24	21	0.97096D+05	0.296D+01	26	20	0.97096D+05	0.300D+01
		10	0.15939D+11	27	26	0.97096D+05	0.369D+01	47	27	0.97096D+05	0.411D+01
		100	0.15939D+15	31	31	0.97096D+05	0.444D+01	70	31	0.97096D+05	0.481D+01
POWELLSG	10000	1	0.53750D+06	16	16	0.10947D-04	0.143D+02	33	15	0.83906D-05	0.179D+02
		10	0.40385D+10	21	21	0.32920D-04	0.190D+02	28	22	0.11695D-04	0.257D+02
		100	0.40251D+14	27	27	0.19556D-04	0.247D+02	31	27	0.54051D-05	0.316D+02
QUARTC	1000	1	0.19850D+15	35	35	0.22354D-09	0.231D+01	35	35	0.22354D-09	0.287D+01
		10	0.18125D+15	35	35	0.20411D-09	0.229D+01	35	35	0.20411D-09	0.285D+01
		100	0.65804D+14	34	34	0.37515D-09	0.223D+01	35	34	0.37515D-09	0.278D+01

TABLE A-4

Results of the CUTE *test problems (continued)*

				Standard				Tensor			
func	*n*	x_0	*initf*	*fcn*	*grad*	*finalf*	*time*	*fcn*	*grad*	*finalf*	*time*
SINQUAD	10000	1	0.65610D+00	25	20	0.39609D-10	0.975D+03	66	21	0.35876D-15	0.103D+04
		10	0.00000D+00	1	1	0.35876D-15	0.290D+00	1	1	0.35876D-15	0.300D+00
		100	0.65610D+04	18	18	0.69625D-08	0.881D+03	47	19	0.42524D-15	0.966D+03
SROSENBR	5000	1	0.48500D+05	9	8	0.93253D-11	0.297D+01	16	7	0.10927D-17	0.332D+01
		10	0.44893D+10	97	66	0.38588D-18	0.279D+02	65	33	0.22535D-15	0.179D+02
		100	0.51123D+14	–	IL	–	–	204	97	0.26051D-08	0.547D+02
TQUARTIC	1000	1	0.81000D+00	2	2	0.39936D-27	0.270D+00	2	2	0.39936D-27	0.260D+00
		10	0.00000D+00	1	1	0.39936D-27	0.200D-01	1	1	0.39936D-27	0.200D-01
		100	0.81000D+02	2	2	0.12622D-24	0.260D+00	2	2	0.12622D-24	0.260D+00
TRIDIA	10000	1	0.50005D+08	2	2	0.41242D-24	0.119D+01	2	2	0.41242D-24	0.117D+01
		10	0.50005D+10	2	2	0.13131D-22	0.117D+01	2	2	0.13131D-22	0.117D+01
		100	0.50005D+12	2	2	0.33835D-20	0.117D+01	2	2	0.33835D-20	0.117D+01
WOODS	10000	1	0.27296D+08	28	23	0.31973D-14	0.259D+02	49	21	0.33996D-17	0.305D+02
		10	0.22566D+12	51	42	0.42521D-12	0.484D+02	72	34	0.42039D-09	0.503D+02
		100	0.22122D+16	73	60	0.27578D-10	0.698D+02	100	49	0.16526D-16	0.730D+02
WOODS1	10000	1	0.55500D+06	9	9	0.17486D-11	0.949D+01	12	8	0.25903D-20	0.103D+02
		10	0.41460D+10	15	15	0.38193D-13	0.165D+02	22	14	0.26198D-19	0.196D+02
		100	0.40591D+14	21	21	0.61171D-14	0.236D+02	33	20	0.17403D-17	0.285D+02

TABLE A-5
Results of the rank $n-1$ test problems from the CUTE *collection*

func	*n*	x_0	*initf*	Standard *fcn*	Standard *grad*	Standard *finalf*	Standard *time*	Tensor *fcn*	Tensor *grad*	Tensor *finalf*	Tensor *time*
BRYBND	5000	1	0.12488D+06	488	30	0.17586D-10	0.376D+03	176	10	0.13179D-10	0.130D+03
		10	0.10765D+12	–	IL	–	–	1088	60	0.85644D-10	0.785D+03
		100	0.12303D+18	3396	201	0.97750D-21	0.263D+04	1560	84	0.16631D-11	0.111D+04
DIXON3DQ	5000	1	0.40000D+01	6	2	0.62536D-17	0.712D+01	6	2	0.62536D-17	0.718D+01
		10	0.12100D+03	6	2	0.18917D-15	0.713D+01	6	2	0.18917D-15	0.713D+01
		100	0.10201D+05	6	2	0.15948D-13	0.713D+01	6	2	0.15948D-13	0.713D+01
NONDQUAR	10000	1	0.10003D+05	–	IL	–	–	182	24	0.57721D-07	0.635D+03
		10	0.99981D+08	–	IL	–	–	4414	187	0.17004D-07	0.608D+04
		100	0.99980D+12	–	IL	–	–	3820	194	0.62846D-07	0.560D+04
QUARTC	1000	1	0.45000D+05	57	15	0.61708D-05	0.631D+01	13	4	0.24654D-07	0.144D+01
		10	0.45000D+09	81	21	0.36635D-05	0.905D+01	29	5	0.53107D-07	0.240D+01
		100	0.45000D+13	101	26	0.11038D-04	0.113D+02	130	22	0.50906D-06	0.107D+02
SROSENBR	5000	1	0.48481D+05	30	8	0.11403D-09	0.477D+02	44	7	0.45822D-12	0.422D+02
		10	0.44888D+10	286	65	0.23622D-12	0.440D+03	121	21	0.16587D-10	0.146D+03
		100	0.51122D+14	–	IL	–	–	242	49	0.35217D-11	0.344D+03
TQUARTIC	1000	1	0.32368D+04	38	12	0.38436D-15	0.433D+01	17	4	0.98215D-17	0.155D+01
		10	0.15962D-23	1	1	0.98215D-17	0.200D-01	1	1	0.98215D-17	0.200D-01
		100	0.32368D+06	23	8	0.20695D-15	0.275D+01	28	9	0.14036D-15	0.335D+01
TRIDIA	10000	1	0.50005D+08	6	2	0.41155D-14	0.267D+02	6	2	0.41155D-14	0.266D+02
		10	0.50005D+10	6	2	0.44999D-12	0.266D+02	6	2	0.44999D-12	0.266D+02
		100	0.50005D+12	11	3	0.14577D-13	0.531D+02	11	3	0.14914D-13	0.535D+02
WOODS	1000	1	0.27296D+07	248	49	0.52712D-11	0.236D+02	224	32	0.41898D-10	0.168D+02
		10	0.22566D+11	342	67	0.63594D-11	0.324D+02	245	38	0.20790D-11	0.199D+02
		100	0.22122D+15	446	87	0.44137D-11	0.423D+02	308	47	0.22064D-10	0.247D+02
WOODS1	1000	1	0.55491D+05	86	18	0.25201D-09	0.816D+01	50	10	0.21981D-08	0.463D+01
		10	0.41460D+09	116	24	0.21634D-09	0.111D+02	84	16	0.40452D-08	0.765D+01
		100	0.40591D+13	146	30	0.19591D-09	0.139D+02	125	22	0.50008D-08	0.108D+02

TABLE A-6
Results of the rank $n-2$ test problems from the CUTE *collection*

func	*n*	x_0	*initf*	Standard *fcn*	Standard *grad*	Standard *finalf*	Standard *time*	Tensor *fcn*	Tensor *grad*	Tensor *finalf*	Tensor *time*
BRYBND	5000	1	0.12487D+06	527	29	0.42357D-09	0.454D+03	268	14	0.30203D-08	0.219D+03
		10	0.10765D+12	824	46	0.16732D-15	0.724D+03	670	32	0.34308D-10	0.519D+03
		100	0.12303D+18	–	IL	–	–	1401	68	0.26897D-12	0.110D+04
DIXON3DQ	5000	1	0.80000D+01	7	2	0.62564D-17	0.938D+01	7	2	0.62564D-17	0.938D+01
		10	0.24200D+03	7	2	0.18928D-15	0.934D+01	7	2	0.18928D-15	0.934D+01
		100	0.20402D+05	7	2	0.15948D-13	0.933D+01	7	2	0.15948D-13	0.936D+01
NONDQUAR	10000	1	0.10002D+05	–	IL	–	–	1109	70	0.14468D-06	0.271D+04
		10	0.99980D+08	–	IL	–	–	1674	86	0.96220D-07	0.332D+04
		100	0.99980D+12	–	IL	–	–	1923	101	0.40263D-07	0.382D+04
QUARTC	1000	1	0.45000D+05	57	15	0.61708D-05	0.646D+01	13	4	0.24654D-07	0.145D+01
		10	0.45000D+09	81	21	0.36635D-05	0.921D+01	101	17	0.53107D-07	0.819D+01
		100	0.45000D+13	101	26	0.11038D-04	0.115D+02	130	22	0.50906D-06	0.107D+02
SROSENBR	5000	1	0.48481D+05	72	13	0.82242D-14	0.108D+03	91	15	0.23908D-16	0.128D+03
		10	0.44890D+10	429	77	0.69440D-04	0.683D+03	465	68	0.14337D-16	0.615D+03
		100	0.51122D+14	–	IL	–	–	1294	201	0.80433D+06	0.183D+04
TQUARTIC	1000	1	0.32335D+04	48	12	0.94635D-16	0.565D+01	30	6	0.65443D-18	0.305D+01
		10	0.15946D-23	1	1	0.15946D-23	0.200D-01	1	1	0.15946D-23	0.200D-01
		100	0.32335D+06	49	12	0.18893D-15	0.564D+01	54	12	0.56162D-18	0.636D+01
TRIDIA	10000	1	0.50005D+08	8	2	0.41344D-14	0.349D+02	8	2	0.41344D-14	0.349D+02
		10	0.50005D+10	8	2	0.45002D-12	0.350D+02	8	2	0.45002D-12	0.349D+02
		100	0.50005D+12	15	3	0.25973D-12	0.703D+02	15	3	0.25973D-12	0.709D+02
WOODS	1000	1	0.27277D+07	196	31	0.77284D-13	0.189D+02	168	26	0.18453D-12	0.165D+02
		10	0.22564D+11	325	51	0.68702D-06	0.316D+02	289	41	0.10869D-12	0.268D+02
		100	0.22121D+15	434	68	0.56038D-05	0.423D+02	89	11	0.11251D-08	0.684D+01
WOODS1	1000	1	0.55470D+05	118	18	0.18927D-09	0.107D+02	91	16	0.10966D-07	0.975D+01
		10	0.41458D+09	–	NC	–	–	127	22	0.30436D-08	0.136D+02
		100	0.40590D+13	–	NC	–	–	31	6	0.19654D-08	0.324D+01