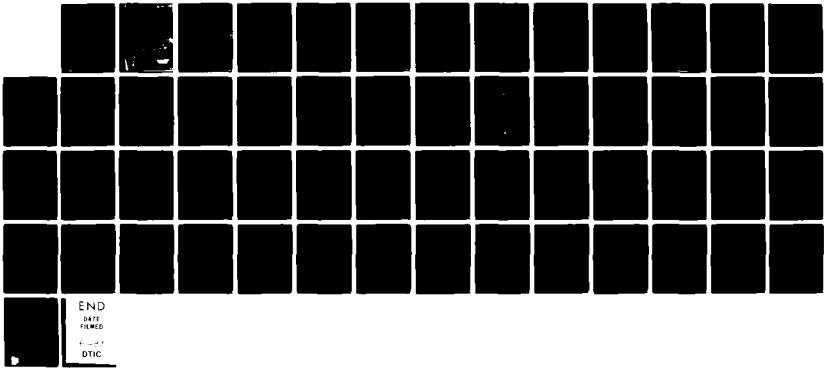
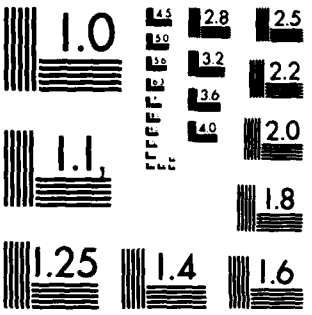
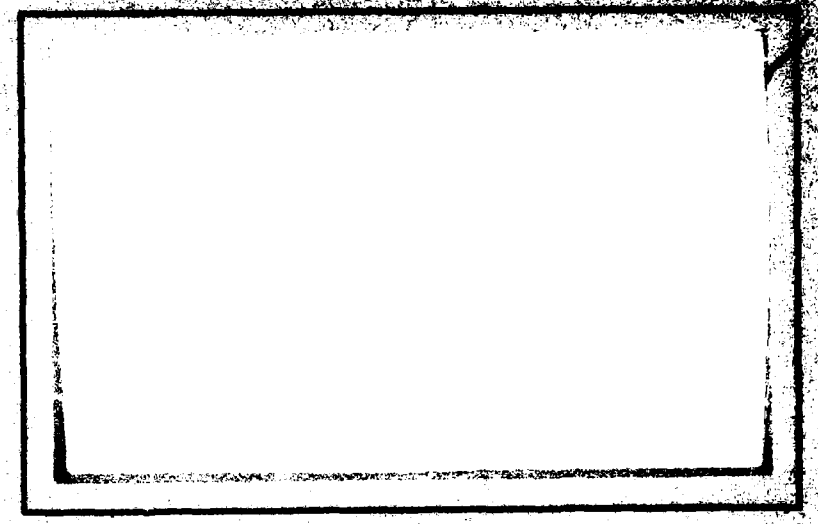
AD-A128 750 TENSOR METHODS FOR NONLINEAR EQUATIONS(U) COLORADO UNIV 1/1
AT BOULDER DEPT OF COMPUTER SCIENCE
R B SCHNABEL ET AL. APR 83 CU-CS-243-83 ARO-18197.3-MA
UNCLASSIFIED DAAG29-81-K-0108 F/G 12/1 NL



END
DATE FILMED
DTIC



UNIVERSITY OF COLORADO



DEPARTMENT OF COMPUTER SCIENCE
CAMPUS BOX 430
UNIVERSITY OF COLORADO, BOULDER
BOULDER, COLORADO 80309

Technical Report

TENSOR METHODS FOR NONLINEAR EQUATIONS

Robert B. Schnabel*
Paul D. Frank

Department of Computer Science
University of Colorado
Boulder, Colorado 80309

CU-CS-243-83 April, 1983

*This research supported by ARO contract DAAG 29-81-K-0108

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Tensor Methods for Nonlinear Equations		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robert B. Schnabel, Paul D. Frank		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Colorado, Computer Science Campus Box 430, Boulder, CO 80309		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office Post Office Box 12211 Research Triangle Park, NC 27709		12. REPORT DATE 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE NA

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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

NA

18. SUPPLEMENTARY NOTES

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

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

Nonlinear equations, tensor, singular Jacobian

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

A new class of methods for solving systems of nonlinear equations, called tensor methods, is introduced. Tensor methods are general purpose methods intended especially for problems where the Jacobian matrix at the solution is singular or ill-conditioned. They base each iteration on a quadratic model of the nonlinear function, the standard linear model augmented by a simple second order term. The second order term is selected so that the model interpolates function values from several previous iterations, as well as the current function value and Jacobian. The tensor method requires

no more function and derivative information per iteration, and hardly more storage or arithmetic per iteration, than a standard method based on Newton's method. In extensive computational tests, a tensor algorithm is significantly more efficient than a similar algorithm based on the standard linear model, both on standard nonsingular test problems and on problems where the Jacobian at the solution is singular.

Abstract

A new class of methods for solving systems of nonlinear equations, called tensor methods, is introduced. Tensor methods are general purpose methods intended especially for problems where the Jacobian matrix at the solution is singular or ill-conditioned. They base each iteration on a quadratic model of the nonlinear function, the standard linear model augmented by a simple second order term. The second order term is selected so that the model interpolates function values from several previous iterations, as well as the current function value and Jacobian. The tensor method requires no more function and derivative information per iteration, and hardly more storage or arithmetic per iteration, than a standard method based on Newton's method. In extensive computational tests, a tensor algorithm is significantly more efficient than a similar algorithm based on the standard linear model, both on standard nonsingular test problems and on problems where the Jacobian at the solution is singular.

1. Introduction

This paper introduces a new class of methods, tensor methods, for solving the nonlinear equations problem

$$\text{given } F: R^n \to R^n, \quad \text{find } x_* \in R^n \text{ such that } F(x_*) = 0 \tag{1.1}$$

where it is assumed that $F(x)$ is at least once continuously differentiable. The novel feature of these methods is that they base each iteration on a quadratic model of $F(x)$ whose second order term has a special, restricted, form. Tensor methods are especially intended to improve upon the performance of standard methods on problems where the Jacobian matrix of F at x_*, $F'(x_*) \in R^{n \times n}$, is singular or ill-conditioned. At the same time, they are intended to be at least as efficient as standard methods on problems where $F'(x_*)$ is nonsingular. Their storage requirements and arithmetic operations per iteration are not significantly higher than the requirements of standard methods.

Standard methods for solving (1.1) base each iteration upon a linear model $M(x)$ of $F(x)$ around the current iterate $x_c \in R^n$,

$$M(x_c + d) = F(x_c) + J_c d \tag{1.2}$$

where $d \in R^n$, $J_c \in R^{n \times n}$. These methods can be divided into two classes, those where J_c is the current Jacobian matrix $F'(x_c)$ or a finite difference approximation to it, and those where J_c is a secant (quasi-Newton) approximation to the Jacobian. In this paper we propose extensions to the first type of methods, those that use analytic or finite difference Jacobians, because this is the most basic setting in which to study the new ideas of this paper. In subsequent papers, we intend to extend tensor methods to secant methods for nonlinear equations, and to unconstrained optimization.

When the analytic Jacobian is available, the linear model (1.2) becomes

$$M(x_c + d) = F(x_c) + F'(x_c)d. \tag{1.3}$$

The most basic method for nonlinear equations, Newton's method, is defined

when $F'(x_c)$ is nonsingular, and consists of setting the next iterate x_+ to the root of (1.3),

$$x_+ = x_c - F'(x_c)^{-1}F(x_c). \tag{1.4}$$

The distinguishing feature of Newton's method is that if $F'(x_c)$ is Lipschitz continuous in a neighborhood containing the root x_* and $F'(x_*)$ is nonsingular, then the sequence of iterates produced by (1.4) converges locally and q-quadratically to x_*. This means that there exist $\delta>0$ and $c\geq 0$ such that the sequence of iterates $\{x_k\}$ produced by Newton's method obeys

$$\|x_{k+1} - x_*\| \leq c\ \|x_k - x_*\|^2$$

if $\|x_0 - x_*\| \leq \delta$. In practice, local q-quadratic convergence means eventual fast convergence.

Newton's method is not usually quickly locally convergent, however, if $F'(x_*)$ is singular. For example when applied to one equation in one unknown ($n=1$) where $f'(x_*)=0$ but $f''(x_*)\neq 0$, Newton's method is locally q-linearly convergent with constant converging to ½, meaning that the sequence of iterates $\{x_k\}$ obeys

$$|x_{k+1} - x_*| = c_k\ |x_k - x_*|\ , \quad \lim_{k\to\infty} c_k = \tfrac{1}{2}$$

if $|x_c - x_*|$ is sufficiently small. For systems of equations, the situation is more complex and has been analyzed by many authors, including Decker and Kelley [1980a, 1980b, 1982], Decker, Keller, and Kelley [1982], Griewank [1980a, 1980b, 1983], Griewank and Osborne [1981], Keller [1970], Kelley and Suresh [1982], Rall [1966], and Reddien [1978, 1980]. In summary, their papers show that from many starting points, Newton's method for systems of equations also is locally q-linearly convergent with constant converging to ½, although from some starting points arbitrarily close to x_*, (1.4) may be repulsive. In practice, Newton's method usually exhibits local linear convergence with constant $\cong$ ½ on singular problems (see Table 6.6), much slower convergence than one would like.

Several of the above mentioned papers, for example Decker, Keller, and Kelley [1982] and Griewank [1980a, 1983], propose methods that are rapidly convergent on some singular problems. Most of these methods are related to the one dimensional acceleration technique of taking j times the Newton step if one has a j^{th} order singularity. This requires deciding whether the problem is singular, which probably makes such methods unsuitable for general purpose use. The major aim of this paper is to provide a general purpose method that has rapid local convergence even when $F'(x_*)$ is singular. In addition, tensor methods usually will not experience any special difficulty when $F'(x_c)$ is singular or ill-conditioned, while methods based on (1.3) must be modified in this situation.

Systems of nonlinear equations with $F'(x_*)$ singular or ill-conditioned occur in a number of important practical situations. For example, conservation laws in stiff systems of ordinary differential equations sometimes cause the Jacobian of the associated system of nonlinear equations to be very nearly singular for all x. In curve tracing problems it also is not uncommon to generate systems of nonlinear equations with nearly singular Jacobians. In unconstrained minimization problems arising from data fitting, the Hessian matrix $\nabla^2 f(x_*)$ usually is singular if the problem is over-parameterized, and often $\nabla^2 f(x_*)$ is ill-conditioned because the data fitting model is far more sensitive to some parameters than to others. In all these cases, it is important to notice that the (near) rank deficiency in the derivative matrix usually is small. This is the case in which our methods are intended to improve upon standard methods. Our methods are not intended for problems where the rank of $F'(x_*)$ is small in comparison to its dimension, although sometimes they are effective in this case in practice.

The other well-known disadvantage of Newton's method is that it may not converge to any root x_* if it is started too far from any root. The main remedies used in practice are augmenting (1.4) by line search or trust region algorithms,

see for example Fletcher [1980], Gill, Murray, and Wright [1981], or Dennis and Schnabel [1983]. Our new methods use the same strategies when the new local step is unsatisfactory.

It is important to consider the costs associated with solving systems of nonlinear equation by standard methods. These can be divided into three types: the arithmetic operations required by the algorithm (excluding function and derivative evaluations), the storage required, and the evaluations of the nonlinear function $F(x)$ and the Jacobian $F'(x)$, if it is provided. For algorithms that use an analytic or finite difference Jacobian, the dominant arithmetic cost is one matrix factorization per iteration, requiring $n^3/3$ (for LU) or $2n^3/3$ (for QR) additions and multiplications per iteration. At least n^2 storage is required, for the Jacobian, and some algorithms store a second $n \times n$ matrix. Finally, $F(x)$ must be evaluated at least once per iteration; in addition, either $F'(x)$ is evaluated once per iteration, or it is approximated by finite differences, requiring up to n additional evaluations of $F(x)$ per iteration. In many practical problems, the evaluations of $F(x)$ and $F'(x)$ are expensive and dominate the other costs. The main efficiency goal of our new method is to decrease the number of function and derivative evaluations required to solve systems of nonlinear equations; however, no substantial increase in the arithmetic cost per iteration, or in the storage requirements, will be permitted.

Our new methods are based on expanding the linear model (1.3) of $F(x)$ around x_c to the quadratic model

$$M_T(x_c+d) = F(x_c) + F'(x_c)d + \tfrac{1}{2}T_c dd \qquad (1.5)$$

where $T_c \in R^{n \times n \times n}$. The three dimensional object T_c often is referred to as a tensor, hence we call (1.5) a ***tensor model***, and methods based upon (1.5) ***tensor methods***. We define the notation $T_c dd$ used in (1.5) before proceeding.

Definition 1.1. Let $T \in R^{n \times n \times n}$. Then T is composed of n ***horizontal faces*** $H_i \in R^{n \times n}$, $i = 1, \cdots, n$, where $H_i[j,k] = T[i,j,k]$. For $v, w, \in R^n$, $Tvw \in R^n$ with

$$Tvw[i] = v^T H_i w = \sum_{j=1}^{n} \sum_{k=1}^{n} T[i,j,k]v[j]w[k].$$

Note that $M_T(x_c + d)$ is simply the n-vector of n quadratic models of the component functions of $F(x)$,

$$(M_T(x_c + d))[i] = f_i + g_i^T d + \tfrac{1}{2} d^T H_i d, \quad i = 1, \cdots, n$$

where $f_i = F(x_c)[i]$, g_i^T = row i of $F'(x_c)$, and H_i is the Hessian matrix of the i^{th} component function of $F(x)$, or an approximation to it.

The obvious choice of T_c in (1.5) is the matrix $F''(x_c)$ of second partial derivatives of F at x_c; this makes (1.5) the first three terms of the Taylor series expansion of F around x_c. Several serious disadvantages, however, make (1.5) with $T_c = F''(x_c)$ unacceptable for algorithmic use. They include:

(1) The n^3 *second partial derivatives* of F at x_c would have to be computed at each iteration.

(2) The model would take more than $n^3/2$ locations to store.

(3) To find a root of the model, at each iteration one would have to solve a system of n quadratic equations in n unknowns.

(4) The model might not have a real root.

To use a model of form (1.5) and avoid these disadvantages, our tensor method uses a very restricted form of T_c. In particular, our tensor model requires no additional derivative or function information; the additional costs of forming and solving our tensor model are small compared to the $O(n^3)$ arithmetic cost per iteration of standard methods; and the additional storage required for our tensor model is small compared to the n^2 storage required for the Jacobian. The key contribution of this paper is showing how one may construct a useful quadratic model that satisfies these criteria. Our tensor model

still does not always have a real root, and we will address this issue.

The remainder of the paper is organized as follows. In section 2, we discuss briefly the specialization of our tensor method to one nonliner equation in one unknown. Of course, when $n=1$, many of the disadvantages stated above for a second order Taylor series model are irrelevant, and indeed, various methods for solving a single nonlinear equation are based on using quadratic models. The material in section 2 is included only to motivate our multi-variable method.

The heart of the paper, our techniques for forming and solving the tensor model for systems of nonlinear equations, is contained in sections 3 and 4, respectively. The full tensor algorithm is presented in section 5 and various implementation considerations are discussed. In section 6 we present test results of our tensor method on the problems of Moré, Garbow, and Hillstrom [1981], and on modifications of these problems constructed so that $F'(x_*)$ is singular. We compare our tensor algorithm to an algorithm that uses the standard linear model (1.3) but is identical in virtually all other respects. We comment briefly on extensions of our tensor methods in section 7.

Notice that we have denoted members of a sequence of n-vectors x by $\{x_k\}$ where each $x_k \in R^n$, and components of a vector $v \in R^n$ by $v[i] \in R$. The convention of using non-numerical subscripts for replications and bracket notation for scalar components is continued throughout the paper. In section 4, integer subscripts are used to denote portions of vectors or matrices that are themselves vectors or matrices; for example, the portions $\hat{d}_1 \in R^{n-p}$ and $\hat{d}_2 \in R^p$ of the vector $\hat{d} \in R^n$, and the portions $\hat{J}_1 \in R^{n \times (n-p)}$ and $\hat{J}_2 \in R^{n \times p}$ of the matrix $\hat{J} \in R^{n \times n}$, are defined in step 2 of Algorithm 4.1.

2. The tensor method for one equation in one unknown

In this section, we discuss briefly the restriction of our tensor method to the case $n = 1$. The use of a quadratic model for solving one nonlinear equation in one unknown is well known and a part of some software libraries; an early reference is Muller [1956]. We make no attempt to compare the one variable version of our tensor algorithm to similar algorithms for solving a single nonlinear equation. Rather, the material in this section is included solely to motivate some features of the tensor algorithm for systems of equations that follows.

The quadratic model (1.5) restricted to $n = 1$ is

$$m_T(x_c + d) = f(x_c) + f'(x_c)d + \tfrac{1}{2}t_c d^2 \tag{2.1}$$

where all quantities now are scalars. We said in section 1 that we would not use second derivatives. Then an obvious way to select t_c is to emulate the secant method by asking the model (2.1) to interpolate the value of $f(x)$ at the previous iterate x_-. This means

$$f(x_-) = f(x_c) + f'(x_c)s_- + \tfrac{1}{2}t_c s_-^2 \tag{2.2a}$$

where we define

$$s_- = x_- - x_c, \tag{2.2b}$$

the step from x_c to x_-. The second derivative approximation t_c is determined uniquely by (2.2).

The roots of (2.1) are found by solving one quadratic equation in one unknown. Usually, (2.1) will have either no real roots or two real roots. If there are two real roots, then a reasonable way to choose between then is to let x_+ be the root that is closer to x_c. This is written in a numerically stable way as

$$x_+ = x_c - \frac{2f(x_c)}{f'(x_c) + \operatorname{sign}(f'(x_c))\sqrt{(f'(x_c))^2 - 2t_c f(x_c)}}. \tag{2.3}$$

If (2.1) has no real roots ($(f'(x_c))^2 < 2t_c f(x_c)$), there are two obvious alternatives: try the Newton step

$$x_+^N = x_c - f(x_c) / f'(x_c), \tag{2.4}$$

or try setting x_+ to the critical point of (2.1),

$$x_+^{min} = x_c - f'(x_c) / t_c. \tag{2.5}$$

where the absolute value of the quadratic model is smallest. The latter strategy definitely is advantageous close to a root x_* where $f'(x_*) = 0 \neq f''(x_*)$. In this case, the iterates produced by (2.2, 2.5) converge to x_* with q-order $(1+\sqrt{5})/2 \cong 1.61$, as this is just a variant of the secant method for minimizing or maximizing $f(x)$. The step (2.2, 2.3) may or may not be defined in this case, if it is defined it is also quite satisfactory as the iterates it produces will converge to x_* with q order $(1+\sqrt{3})/2 \cong 1.37$. (See e.g. Traub [1964] for very similar proofs.) On the other hand, we have already stated that the iterates produced by (2.2, 2.4) are q-linearly convergent to x_* with constant converging to ½ in this case.

When $f'(x_*) \neq 0$, the quadratic model (2.1, 2.2) will have real roots for x_c sufficiently close to x_*. In general, however, (2.1) may not have real roots, and aside from the above-mentioned case, neither x_+^N or x_+^{min} consistently is the better choice. In Frank [1982], we implemented a method for solving one nonlinear equation in one unknown based on (2.2 - 2.5) and a line search. We found that when the quadratic model had no real root, a good strategy for choosing between the steps to x_+^N and x_+^{min} as the initial step in the line search was to choose the longer step. This is equivalent to choosing x_+^{min} if and only if $|m_T(x_+^{min})| < ½ |f(x_c)|$. This strategy always prefers x_+^{min} over x_+^N sufficiently close to a root x_* where $f'(x_*)=0$, guaranteeing fast local convergence in this case.

One case where this strategy is not desirable is when the algorithm cannot locate a root of $f(x)$ and must converge to a stationary point of $f(x)$. In this

case the step to x_+^{min} is shorter but considerably more desirable than the Newton step. The algorithm in Frank [1982] contains a simple strategy to recognize this situation and uses the step to x_+^{min} instead of the Newton step in this case.

Several aspects of the algorithm for one nonlinear equation carry over to systems of equations. There we again use the extra freedom in the quadratic model to interpolate function values at past iterates. The multi-variable quadratic model again may have multiple real roots or no real roots. In the former case we again hope to find the root closest to x_c. In the latter case, we again use either the step to the minimizer (in the l_2 norm) of the quadratic model or the Newton step for our line search direction, choosing between the two steps by criteria similar to those described above.

3. Forming the tensor model

We now show how we select the tensor term $T_c \in R^{n \times n \times n}$ in the model

$$M_T(x_c + d) = F(x_c) + F'(x_c)d + \tfrac{1}{2}T_c dd. \tag{3.1}$$

Our choice of T_c will cause the second order term $T_c dd$ in (3.1) to have a simple, useful, form.

We have already stated that T_c will not contain actual second derivative information. Another way we can use the second order term in (3.1) is to ask the model to interpolate additional values of the function $F(x)$ or the Jacobian $F'(x)$ that have already been computed by the algorithm. In our method, we will select some set of p not necessarily consecutive past iterates $x_{-1}, \cdots, x_{-p}$, and require the model (3.1) to interpolate the function values $F(x_{-k})$ at these points. That is, the model should satisfy

$$F(x_{-k}) = F(x_c) + F'(x_c)s_k + \tfrac{1}{2}T_c s_k s_k, \quad k = 1, \cdots, p \tag{3.2a}$$

where

$$s_k = x_{\nu_k} - x_c, \quad k = 1, \cdots, p \tag{3.2b}$$

First we describe how the past points $x_{\nu_1}, \cdots, x_{\nu_p}$ are selected. Then we show how we choose T_c to satisfy (3.2). Alternative ways to select T_c are mentioned briefly in section 7.

For the equations (3.2) to always be consistent, it is clear that the set of directions $\{s_k\}$ from x_c to the selected past points x_{ν_k} must be linearly independent. In fact, our computational experience with other algorithms that interpolate information from past iterates has shown that the directions $\{s_k\}$ should be *strongly* linear independent, in the sense that each direction s_k should make an angle of at least Θ degrees with the linear subspace spanned by the other directions, values of Θ between 20 and 45 degrees have proven appropriate in practice. At each iteration, therefore, we choose the past points $\{x_{\nu_k}\}$ that we include in (3.2) by the following procedure. We consider the past iterates in order starting with the most recent. We always select the most recent iterate, and then select each preceding past iterate if the step from it to x_c makes an angle of at least Θ degrees with the subspace spanned by the steps to the already selected more recent iterates. This procedure is implemented easily using a modified Gram-Schmidt algorithm.

We also set an upper bound p on the number of past function values interpolated by the model at each iteration. Since the set $\{s_k\}$ must be linearly independent, clearly $p \leq n$, but we enforce a much smaller bound,

$$p < \sqrt{n} \tag{3.3}$$

This bound also was motivated by our computational experience with other algorithms that interpolate information from past iterates; we found that using more than about $\sqrt{n}$ interpolation conditions rarely was beneficial (see e.g. Stordahl [1980]). The bound (3.3) also is crucial to the efficiency in storage and

arithmetic operations of our tensor method. For example, since we also only consider a maximum of $\sqrt{n}$ past iterates in the above mentioned Gram-Schmidt algorithm, it requires about n^2 additions and multiplications.

Now we discuss how we choose T_c to satisfy (3.2). It is convenient to rewrite (3.2) as

$$T_c s_k s_k = z_k , \quad k = 1, \cdots, p \tag{3.4a}$$

where

$$z_k \in R^n, \; z_k = 2 (F(x_{c+k}) - F(x_c) - F'(x_c)s_k). \tag{3.4b}$$

This is a set of $np \leq n^{1.5}$ linear equations in the n^3 unknowns $T_c[i,j,k]$, $1 \leq i,j,k \leq n$. (Actually there are $(n^3+n^2)/2$ unknowns since each horizontal face of T_c must be symmetric, this symmetry is provided automatically by the following derivation.) Since (3.4) is underdetermined, we follow the standard and successful practice in secant methods for nonlinear equations and optimization (see e.g. Dennis and Schnabel [1979]) and choose the T_c that satisfies

$$\underset{T_c \in R^{n \times n \times n}}{\text{minimize}} \quad \|T_c\|_F \tag{3.5}$$

$$\text{subject to} \quad T_c s_k s_k = z_k , \quad k = 1, \cdots, p$$

where $\|T_c\|_F$, the Frobenius norm of T_c, is defined by

$$\|T_c\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} (T_c[i,j,k])^2 .$$

The solution to (3.5) is given by Theorem 3.2. First we define a rank one tensor.

Definition 3.1. Let $u, v, w \in R^n$. The tensor $T \in R^{n \times n \times n}$ for which $T[i,j,k] = u[i] * v[j] * w[k]$, $1 \leq i,j,k \leq n$ is called a rank one tensor and denoted $T = uvw$.

Note that the i^{th} horizontal face of the rank one tensor uvw is the rank one matrix $u[i](vw^T)$. Theorem 3.2 shows that the solution to (3.5) is the sum of p rank one tensors whose horizontal faces are symmetric.

Theorem 3.2. Let $p \le n$, let $s_k \in R^n$, $k = 1, \cdots, p$ with $\{s_k\}$ linearly independent, and let $z_k \in R^n$, $k = 1, \cdots, p$. Define $M \in R^{p \times p}$ by $M[i,j] = (s_i^T s_j)^2$, $1 \le i,j \le p$, $Z \in R^{n \times p}$ by column k of $Z = z_k$, $k = 1, \cdots, p$. Then M is positive definite, and the solution to (3.5) is

$$T_c = \sum_{k=1}^{n} a_k \, s_k \, s_k \tag{3.6}$$

where a_k is the k^{th} column of $A \in R^{n \times p}$, A defined by $A = Z * M^{-1}$.

Proof : Problem (3.5) is equivalent to solving independently the n separate problems over the n horizontal faces H_i of T_c

$$\underset{H_i \in R^{n \times n}}{\text{minimize}} \ \|H_i\|_F \tag{3.7}$$

$$\text{subject to} \ \ s_k^T H_i s_k = z_k[i], \ \ k = 1, \cdots, p.$$

because the constraints and the objective function of (3.5) may be decomposed into these n separate constraints and objective functions each involving one of H_i's, and the optimal value of any one H_i clearly does not affect the optimal value of any other H_j. Problem (3.7) simply is an underdetermined system of p linear equations in n^2 unknowns. To express it in standard form, let

$$h_i \in R^{n^2}, \tag{3.8}$$

$$h_i = H_i[1,1], H_i[1,2], \cdots, H_i[1,n], H_i[2,1], \cdots, H_i[2,n], \cdots, H_i[n,1], \cdots, H_i[n,n],$$

$$\bar{S} \in R^{p \times n^2}, \quad \text{row } k \text{ of } \bar{S} = s_k[1] * s_k, \ s_k[2] * s_k, \cdots, s_k[n] * s_k,$$

and $\bar{z}_i$ = row i of Z, that is,

$$\bar{z}_i \in R^p, \quad \bar{z}_i[k] = z_k[i], \ 1 \le i \le n, \ 1 \le k \le p.$$

Then (3.7) is equivalent to

$$\underset{h_i \in R^{n^2}}{\text{minimize}} \ \|h_i\|_2 \quad \text{subject to} \ \ \bar{S} h_i = \bar{z}_i \tag{3.9}$$

and $\bar{S}$ has full row rank because $\{s_k\}$ are linearly independent. Therefore the solution to (3.9) is

$$h_i = \bar{S}^T (\bar{S}\bar{S}^T)^{-1} \bar{z}_i.$$

It is straightforward to verify that $\bar{S}\bar{S}^T = M$, which also shows that M is positive

definite. Since by definition row i of $A = M^{-1} *$(row i of Z),

$$h_i = S^T a_i, \tag{3.10}$$

where $\bar{a}_i$ = row i of A, that is,

$$\bar{a}_i \in R^p, \ \bar{a}_i[k] = a_k[i], \ 1 \le i \le n, \ 1 \le k \le p.$$

Now note that if the transformation (3.8) that transforms $H_i \in R^{n \times n}$ to $h_i \in R^{n^2}$ is applied to $s_k s_k^T \in R^{n \times n}$, the result is row k of $\bar{S}$. Therefore, since (3.10) is equivalent to

$$h_i = \sum_{k=1}^{p} a_i[k] * \text{row } k \text{ of } \bar{S} = \sum_{k=1}^{p} a_k[i] * \text{row } k \text{ of } S, \tag{3.11}$$

transforming (3.11) back to the $n \times n$ matrix H_i yields

$$H_i = \sum_{k=1}^{p} a_k[i] \, s_k \, s_k^T. \tag{3.12}$$

Finally, combining the n horizontal faces H_i given by (3.12) to reform T_c gives (3.6).

Substituting (3.6) into the tensor model (3.1) gives

$$M_T(x_c + d) = F(x_c) + F'(x_c)d + \tfrac{1}{2} \sum_{k=1}^{p} a_k \, (d^T s_k)^2. \tag{3.13}$$

The simple form of the second order term in (3.13) is the key to being able to efficiently form, store, and solve the tensor model. The additional storage required by (3.13) is $2p$ n-vectors, for $\{a_k\}$ and $\{s_k\}$. In addition, the $2p$ n-vectors $\{x_{-k}\}$ and $\{F(x_{-k})\}$ must be stored. Thus the total extra storage required for our tensor model is $4n^{1.5}$ since $p \le \sqrt{n}$. The reader can verify that the entire process described above for forming T_c requires $n^2 p + O(np^2)$ multiplications and additions. The leading term comes from calculating the p matrix-vector products $F'(x_c)s_k$, $k = 1, \cdots, p$; the cost of solving $A = Z * M^{-1}$ is $O(np^2)$. Since $p \le \sqrt{n}$, the leading term in the cost of forming the tensor model is at most $n^{2.5}$ multiplications and additions per iteration, small compared to the at least $n^3/3$ multiplications and additions per iteration required by stan-

dard methods that use analytic derivatives. In the next section, we will see that the extra cost to solve the tensor model also is at most $O(n^{2.5})$.

4. Solving the tensor model

In this section we give an efficient algorithm for finding a root of the tensor model derived in section 3, that is,

$$\text{find } d \in R^n \text{ such that} \tag{4.1}$$

$$M_T(x_c+d) = F(x_c) + F'(x_c)d + \tfrac{1}{2}\sum_{k=1}^{p} a_k\,(d^T s_k)^2 = 0.$$

We show that the solution of (4.1) can be reduced, in $O(n^2 p)$ operations, to the solution of a system of q quadratic equations in p unknowns, plus the solution of a system of $n-q$ linear equations in $n-p$ unknowns. Here $q \geq p$, with $q = p$ whenever $F'(x_c)$ is nonsingular. In addition, if $F'(x_c)$ is singular but has rank at least $n-p$, we show that q still usually equals p and the aforementioned system of linear equations still is well conditioned. We also show that our algorithm efficiently solves the generalization of (4.1),

$$\underset{d \in R^n}{\text{minimize}} \quad \|M_T(x_c+d)\|_2. \tag{4.2}$$

That is, our algorithm will find a minimizer of the tensor model when the model has no real root.

Let us define $S \in R^{n \times p}$ by column k of $S = s_k$.

The basic idea of the algorithm is that since (4.1) is linear on the $n-p$ dimensional subspace $\{d \mid S^T d = 0\}$, (4.1) really only should be quadratic in p variables and linear in the other $n-p$. This is accomplished in steps 1 and 2 of *Algorithm* 4.1 by making a linear transformation of the variable space; an orthogonal transformation is used mainly because it already will be available

from the Gram-Schmidt process used to select $\{s_k\}$. Then a linear transformation of the equations, steps 3 and 4 of Algorithm 4.1, is used to eliminate the $n-p$ transformed linear variables from p of the equations. The result usually is a system of p quadratic equations in p unknowns, (4.5b), that is solved in step 5 of Algorithm 4.1, and a system of $n-p$ equations (4.5a) that are linear in the remaining $n-p$ unknowns and can be used to compute these unknowns once the system of quadratics is solved. The exceptional case $q > p$ arises when this system of linear equations would be singular. In this case, the number of linear equations is decreased by this rank deficiency and the number of quadratic equations is increased correspondingly, but the number of variables in each system is unaffected. Of course in practice, the mathematical notion of rank deficiency is replaced by the numerical notion of adequately small condition number.

Algorithm 4.1 gives the method we use for solving (4.2). Theorem 4.2 verifies that it solves this problem, and gives several other important properties of the algorithm. After the proof of Theorem 4.2 we discuss the efficiency and numerical stability of Algorithm 4.1. We also mention several alternative methods for solving problem (4.2).

We introduce the notation that given $v \in R^m$, $\{v\}^2$ denotes the vector $w \in R^m$ for which $w[i] = v[i]^2$, $i = 1, \cdots, m$. This allows us to denote the second order term of our tensor model below by $\frac{1}{2} A \{S^T d\}^2$.

Algorithm 4.1. Let $p \leq n$, $F \in R^n$, $J \in R^{n \times n}$, A, $S \in R^{n \times p}$, S having full column rank.

Comment: Steps 1-2 transform the system of n equations in n unknowns

$$F + Jd + \tfrac{1}{2} A \{S^T d\}^2 = 0 \tag{4.3}$$

to the system of n equations in the n unknowns $\hat{d}_1 \in R^{n-p}$ and $\hat{d}_2 \in R^p$

$$F + \hat{J}_1\hat{d}_1 + \hat{J}_2\hat{d}_2 + \tfrac{1}{2} A \{\hat{S}_2^T\hat{d}_2\}^2 = 0. \qquad (4.4)$$

1. Find an orthogonal $Q \in R^{n\times n}$ such that $Q^T S = \hat{S}$, where

$$\hat{S} \in R^{n\times p} = \begin{array}{c} \\ \hat{S}_2 \end{array} \begin{array}{|c|} \hline \multicolumn{1}{c}{p} \\ \hline 0 \\ \hline \diagdown\, 0 \\ \hline \end{array} \begin{array}{l} \\ n-p \\ p \end{array}$$

and $\hat{S}_2$ has the triangular shape shown.

2. Calculate $\hat{J} = J * Q$ and let $\hat{J}_1 \in R^{n\times(n-p)}$ and $\hat{J}_2 \in R^{n\times p}$ denote the first $n-p$ and last p columns of $\hat{J}$, respectively.

Also define $\hat{d} = Q^T d$, and let $\hat{d}_1 \in R^{n-p}$ and $\hat{d}_2 \in R^p$ denote the first $n-p$ and last p components of $\hat{d}$, respectively.

Comment : Steps 3-4 transform the system of equations (4.4) to

$$\begin{array}{c} \\ n-q \\ q \end{array} \begin{array}{|c|} \hline \tilde{F}_1 \\ \hline \tilde{F}_2 \\ \hline \end{array} + \begin{array}{|c|c|} \hline \multicolumn{1}{c}{n-p} & \multicolumn{1}{c}{p} \\ \hline \tilde{J}_1 & \tilde{J}_2 \\ \hline 0 & \tilde{J}_3 \\ \hline \end{array} \begin{array}{|c|} \hline \multicolumn{1}{c}{} \\ \hline \hat{d}_1 \\ \hline \hat{d}_2 \\ \hline \end{array} + \tfrac{1}{2} \begin{array}{|c|} \hline \multicolumn{1}{c}{p} \\ \hline \tilde{A}_1 \\ \hline \tilde{A}_2 \\ \hline \end{array} \{\hat{S}_2^T\hat{d}_2\}^2 = 0,$$

that is, to the system of $n-q$ equations in n unknowns

$$\tilde{F}_1 + \tilde{J}_1\hat{d}_1 + \tilde{J}_2\hat{d}_2 + \tfrac{1}{2}\tilde{A}_1\{\hat{S}_2^T\hat{d}_2\}^2 = 0 \qquad (4.5a)$$

and the system of q equations in p unknowns

$$\tilde{F}_2 + \tilde{J}_3\hat{d}_2 + \tfrac{1}{2}\tilde{A}_2\{\hat{S}_2^T\hat{d}_2\}^2 = 0. \qquad (4.5b)$$

3. Find an orthogonal $\hat{Q} \in R^{n\times n}$ and a permutation matrix $P \in R^{(n-p)\times(n-p)}$ such that

$$\hat{Q}\,\hat{J}_1\,P = \begin{array}{c} \\ \tilde{J}_1 \end{array} \begin{array}{|c|} \hline \multicolumn{1}{c}{n-p} \\ \hline \diagdown\, 0 \\ \hline 0 \\ \hline \end{array} \begin{array}{l} \\ n-q \\ q \end{array}$$

where $q \geq p$ and $\tilde{J}_1$ is upper triangular with a non-zero diagonal. Define

$\tilde{d}_1 = P^T \hat{d}_1$, $\tilde{d}_1 \in R^{n-p}$.

4. Calculate

$$\hat{Q}\hat{J}_2 = \begin{matrix} p \\ \left[\begin{array}{c} \tilde{J}_2 \\ \hline \tilde{J}_3 \end{array}\right] \end{matrix} \begin{matrix} n-q \\ q \end{matrix}$$

Similarly calculate $\tilde{A} = \hat{Q}A$, and let $\tilde{A}_1 \in R^{p \times n-q}$ and $\tilde{A}_2 \in R^{p \times q}$ denote the first $n-q$ and the last q rows of $\tilde{A}$, respectively; also calculate $\tilde{F} = \hat{Q}F$, and let $\tilde{F}_1 \in R^{n-q}$ and $\tilde{F}_2 \in R^q$ denote the first $n-q$ and last q components of $\tilde{F}$, respectively.

5. (Solve (4.5b) in the least squares sense.) Solve

$$\underset{\hat{d}_2 \in R^p}{\text{minimize}} \; \|\tilde{F}_2 + \tilde{J}_3\hat{d}_2 + \tfrac{1}{2}\tilde{A}_2\{\hat{S}_2^T\hat{d}_2\}^2\|_2. \tag{4.6}$$

6. (Backsolve (4.5a) for $\tilde{d}_1$.) Find a $\tilde{d}_1$ that solves

$$\tilde{J}_1\tilde{d}_1 = -\tilde{F}_1 - \tilde{A}_1\{\hat{S}_2^T\hat{d}_2\}^2. \tag{4.7}$$

7. Calculate $\hat{d}_1 = P\tilde{d}_1$, $d = Q\hat{d}$.

Theorem 4.2 shows that Algorithm 4.1 finds a root or minimizer of the tensor model, and gives some properties of some matrices used in Algorithm 4.1 whose relation to the numerical stability of the algorithm is discussed later in this section. Recall that for any $W \in R^{n \times m}$, the rank of W is the dimension of the linear subspace spanned by the rows or columns of W, and the nullity of W, the dimension of the linear subspace $\{y \in R^m \mid Wy = 0\}$, is $(m - \text{rank}(W))$.

Theorem 4.2. Algorithm 4.1 calculates a solution to

$$\underset{d \in R^n}{\text{minimize}} \|F + Jd + A\{S^T d\}^2\|_{l_2} \tag{4.8}$$

Furthermore, define $J_S \in R^{(n+p)\times n} = \begin{bmatrix} J \\ S^T \end{bmatrix}$. Then

$$q = p + (n - \text{rank}(J_S)) \tag{4.9}$$

If $\text{rank}(J_S) = n$, then $q = p$ and

$$K(\tilde{J}_1) \leq K(J_S), \tag{4.10}$$

where $K(W)$ denotes the l_2 condition number of W. Also

$$\text{rank}(\tilde{J}_3) = p - (\text{rank}(J_S) - \text{rank}(J)). \tag{4.11}$$

Proof : Substituting $QQ^T d$ for d in (4.3) and using the definitions in steps 1 and 2 transforms (4.3) to (4.4), and clearly the transformation does not affect the smallest l_2 norm value of the system of equations. Next, it is straightforward to verify that premultiplying (4.4) by $\hat{Q}$, substituting $PP^T\hat{d}_1$ for $\hat{d}_1$ in (4.4), and using the definitions in steps 3 and 4 yields (4.5). The minimum l_2 norm values of these two systems of equations are equivalent because premultiplying a vector by an orthogonal matrix doesn't alter its l_2 norm. Finally, since $\tilde{J}_1$ has full row rank, given any $\hat{d}_2$, a $\tilde{d}_1$ may be found that solves (4.5a). Therefore, the minimum value of (4.6) is the minimum l_2 norm value of (4.5), and by the above, of the original problem (4.8), and the l_2 minimizer of (4.5) is found by (4.6-4.7). Step 7 reverses the transformations in variables made in steps 2 and 3 to obtain the minimizer of (4.8) from the minimizer of (4.5).

Now let $Q_1 \in R^{n\times(n-p)}$ and $Q_2 \in R^{n\times p}$ denote the first $n-p$ and last p columns of the Q used in step 1 of Algorithm 4.1. Then since $S^T Q = \hat{S}^T$, we have $S^T Q_1 = 0$ and $S^T Q_2 = \hat{S}_2^T$. Similarly from step 2 of Algorithm 4.1, $JQ_1 = \hat{J}_1$, $JQ_2 = \hat{J}_2$. Thus

$$J_S Q = \begin{array}{|c|c|} \hline \hat{J}_1 & \hat{J}_2 \\ \hline 0 & \hat{S}_2^T \\ \hline \end{array} ,$$

and since Q and $\hat{S}_2^T$ are nonsingular,

$$\text{nullity}(J_S) = \text{nullity}(J_S Q) = \text{nullity}(\hat{J}_1). \tag{4.12}$$

From step 3, $\text{nullity}(\hat{J}_1) = q - p$, and by definition, $\text{nullity}(J_S) = n - \text{rank}(J_S)$, so (4.12) implies (4.9). Next, if we use the notation that for $v \in R^n$, v_1 denotes the first $n-p$ components of v and v_2 the last p components, and let $\|\cdot\|$ denote the l_2 vector norm, then

$$K(J_S) = K(J_S Q) = \frac{\max_{v \in R^n} \|J_S Q v\| / \|v\|}{\min_{w \in R^n} \|J_S Q w\| / \|w\|} \geq$$

$$\frac{\max_{v \in R^n,\, v_2 = 0} \|J_S Q v\| / \|v\|}{\min_{w \in R^n,\, w_2 = 0} \|J_S Q w\| / \|w\|} = \frac{\max_{v_1 \in R^{n-p}} \|\hat{J}_1 v_1\| / \|v_1\|}{\min_{w_1 \in R^{n-p}} \|\hat{J}_1 w_1\| / \|w_1\|} = K(\hat{J}_1) = K(\tilde{J}_1)$$

with the last equality a direct consequence of step 3 of Algorithm 4.1. Also, since

$$\hat{Q} J Q \hat{P} = \begin{array}{|c|c|} \hline \tilde{J}_1 & \tilde{J}_2 \\ \hline 0 & \tilde{J}_3 \\ \hline \end{array}$$

where $\hat{P} \in R^{n \times n}$ is the permutation matrix with P as the upper $n-p \times n-p$ submatrix and the identity otherwise, and since $\tilde{J}_1$ has full row rank, we have

$$\text{nullity}(J) = \text{nullity}(\hat{Q} J Q \hat{P}) = \text{nullity}(\tilde{J}_1) + \text{nullity}(\tilde{J}_3). \tag{4.13}$$

Substituting again $\text{nullity}(\tilde{J}_1) = q - p$ as well as the definitions $\text{nullity}(J) = n - \text{rank}(J)$ and $\text{nullity}(\tilde{J}_3) = p - \text{rank}(\tilde{J}_3)$ into (4.13) yields (4.11)

The first virtue of Algorithm 4.1 is its efficiency. Let us examine the operation counts for multiplications (and divisions); the counts for additions and subtractions are very similar. While Algorithm 4.1 is valid for any $p \leq n$, here we

reinstate the bound $p \leq \sqrt{n}$ from section 3. Then the dominant cost in Algorithm 4.1 is the QR factorization of $\hat{J}_1$ which requires about $2n^3/3 + n^2p + O(n^2)$ multiplications. The next largest cost is the $2n^2p + O(n^2)$ multiplications for the matrix multiplication $J^* Q$ in step 2. The reader can verify that all other portions of steps 1-4 and 6-7 require at most $O(n^2)$ multiplications. The remaining cost is the solution of the nonlinear least squares problem in step 5. While (4.6) must be solved by an iterative algorithm, the point is that the total cost of solving this problem is negligible. In the usual case $q = p$, each iteration of the nonlinear least squares algorithm requires $O(p^3)$ multiplications, and it is reasonable to expect at most a small multiple of p iterations to suffice. (We use the bound $8p$ in our implementation.) Thus solving (4.6) usually can be expected to cost $O(p^4) \leq O(n^2)$ multiplications. If $q > p$, this cost could rise to $O(qp^3) < O(n^{2.5})$. However in our practical experience, $q = p$ almost all the time and q is hardly greater than p otherwise, some summary statistics are given at the end of Section 6. Thus it is reasonable to expect that the nonlinear least squares problem in Algorithm 4.1 requires at most $O(n^2)$ multiplications.

So the total cost of Algorithm 4.1 is about $2n^3/3 + n^2p$ multiplications, at most $n^{2.5}$ multiplications more than the QR factorization of an $n \times n$ matrix. For small n, these numbers are inconsequential, for moderately large n, the $2n^3/3$ dominates and is the same cost as a standard method for nonlinear equations would have if it used the QR factorization. While some standard algorithms for nonlinear equations do use a QR factorization (see e.g. section 6.5 of Dennis and Schnabel [1983]), others use a PLU factorization and require only $n^3/3$ multiplications per iteration if $F'(x_c)$ is nonsingular. Algorithm 4.1 also could be modified to use $n^3/3 + O(n^2p)$ multiplications per iteration when J is nonsingular by using a PLU factorization of $\hat{J}_1$ at step 3. If the resultant $p \times p$ system of quadratics (4.5b) had a solution, no further modification would be necessary. If

not, the minimum norm solution to this new system of quadratics no longer would correspond to the solution of (4.8), and steps 5-6 would have to be combined into the nonlinear least squares solution of the full system (4.5). This still could be accomplished in $O(n^2p)$ operations when J is nonsingular by using an algorithm that takes advantage of the special structure of this problem. We prefer the more expensive QR-based algorithm, however, because it is simpler and more stable numerically, especially when the tensor model has no real root.

The other virtue of Algorithm 4.1 is its numerical stability, even when the Jacobian J is singular or ill-conditioned. This is reflected in the conditioning of the system of linear equations, (4.7), that is solved by Algorithm 4.1. Equations (4.9) and (4.10) of Theorem 4.2 show that if the $(n+p)\times n$ matrix J_S formed by adding the p rows of S^T beneath J has safely linearly independent columns, then this linear system will be square and at least as well conditioned as J_S. In practice, this means it is likely that the system of quadratics will be $p\times p$ and the linear system will be *square and well conditioned* whenever J has at most p zero or small singular values. Notice that (4.10) also implies that the conditioning of the linear system solved by the tensor algorithm always will be as good or better than the conditioning of the linear system solved by Newton's method.

Of course, if J is singular, this singularity does not magically disappear in Algorithm 4.1. Equation (4.11) shows that if J is rank deficient but J_S has full rank (the likely situation when $\text{rank}(J) \geq n-p$), then $\tilde{J}_3$ will have the same rank *deficiency* as J. However the whole point of the tensor algorithm when J is singular is that the singular submatrix $\tilde{J}_3$ is used in the system of ***quadratic*** equations (4.5b), which also contain a portion of the second order information in the tensor model. This system is not necessarily ill-conditioned even if $\tilde{J}_3$ is; for example, one quadratic equation in one unknown with no linear term is not ill-conditioned. Furthermore, this system of quadratics has maximal dimension

$\sqrt{n}$, and in our practical experience even this small upper bound is not approached as n becomes large. Thus when the Jacobian is singular, the effect of the tensor algorithm usually is to isolate this rank deficiency as the linear term in a small system of quadratic equations that contains useful second order information and may be solved accurately without great cost.

If the tensor algorithm converges to a root x_* where $F'(x_*)$ is singular, the Jacobians of the nonlinear least squares problems (4.6) also will become nearly singular because $\tilde{J}_3$ will be nearly singular and the solution $\tilde{d}_2$ will be small. However the cost of solving these small nearly singular nonlinear least squares problems still will be insignificant. So on nonlinear equations problems where $F'(x_*)$ is singular, it appears the effect of the tensor algorithm is to transfer the slow convergence of Newton's method to a series of small quadratic subproblems where slow convergence does not hurt the overall efficiency of the algorithm. More importantly, no additional function evaluations are used in solving these small subproblems.

A minor deficiency of Algorithm 4.1 is that it relies upon the QRP decomposition to determine rank when $\hat{J}_1$ does not have full rank ($q>p$), a singular value decomposition would be preferable in this case. We have already mentioned that this case is very unusual in practice. Note that when $q>p$, Algorithm 4.1 easily is modified to find the smallest d, in the l_2 norm, that minimizes the l_2 norm of the tensor model; all that is required is to find the smallest $\tilde{d}_1$ that solves the system of linear equations (4.7) that is underdetermined in this case.

It is interesting to consider when our tensor model has a *complex* root. Even though this information is not of practical value to our tensor algorithm, it turns out to give a succinct explanation of how the Jacobian J, past directions S, and second order information A used in Algorithm 4.1 are interrelated. Clearly a necessary condition for (4.3) to have any root is that F be in the linear

subspace spanned by the columns of J and A, it is easy to show by the techniques of proof of Theorem 4.2 that this is equivalent to $\tilde{F}_2$ being in the linear subspace spanned by the columns of $\tilde{J}_3$ and $\tilde{A}_2$. A sufficient condition for (4.3) to have a complex root, a direct consequence of a result of Garcia and Li [1980], is given in Theorem 4.3.

Theorem 4.3. Let all the notation be the same as in Algorithm 4.1, and let $J_{AS} \in R^{(n+p)\times(n+p)}$,

$$J_{AS} = \begin{array}{|c|c|} \hline J & A \\ \hline S^T & 0 \\ \hline \end{array}$$

If J_{AS} is nonsingular, (4.3) has 2^p (not necessarily distinct) roots in n dimensional complex space.

Proof : From the proof of Theorem 4.2, it is clear that (4.3) has a complex root if and only if (4.5b) has a complex root. It follows easily from Theorem 3.4 of Garcia and Li [1980] that (4.5b) has a complex root if $\tilde{A}_2$ and $\tilde{S}_2^T$ are nonsingular. We show that this is implied by J_{AS} nonsingular.

Let $Q_+ \in R^{(n+p)\times(n+p)}$ be the orthogonal matrix with Q as the upper left $n \times n$ submatrix and the identity elsewhere, and let $\hat{Q}_+ \in R^{(n+p)\times(n+p)}$ be the corresponding augmentation of $\hat{Q}$. Then it is straightforward to verify from Algorithm 4.1 that

$$\hat{Q}_+ J_{AS} Q_+ = \begin{array}{|c|c|c|} \hline \tilde{J}_1 & \tilde{J}_2 & \tilde{A}_1 \\ \hline 0 & \tilde{J}_3 & \tilde{A}_2 \\ \hline 0 & \tilde{S}_2^T & 0 \\ \hline \end{array} = \hat{J}_{AS}$$

If J_{AS} is nonsingular then $\hat{J}_{AS}$ is nonsingular which implies that $\tilde{J}_1$ is square (i.e. $q = p$) and nonsingular and $\tilde{S}_2^T$ is nonsingular. Since $\hat{J}_{AS}$ and $\tilde{J}_1$ are nonsingular,

the bottom right $2p \times 2p$ submatrix of $\hat{J}_{AS}$ composed of $\tilde{J}_3$, $\tilde{A}_2$, and $\hat{S}_2^T$ must be nonsingular, which implies that $\tilde{A}_2$ is nonsingular because $\hat{S}_2^T$ is nonsingular.

The only case in which we can guarantee that our tensor model has a real root is when the tensor algorithm is sufficiently close to a root x_* where $F'(x_*)$ is nonsingular. Here one can show that the second order term in the tensor model becomes small enough that the tensor model must have a real root.

Finally, we mention other approaches for minimizing the l_2 norm of the tensor model (4.1). In Frank [1982], we discuss a method that first tries to find a root of the tensor model and if it is unsuccessful finds a minimizer. It is related to the version of Algorithm 4.1 using the PLU factorization that we discussed above. Another possibility is to use Newton's method, with a line search on the l_2 norm of the tensor model, to solve (4.2). If the Jacobian J is nonsingular, then by using the Sherman-Morrison-Woodbury formula each iteration would cost only $O(p^3)$ operations after a start-up cost of the factorization of J plus $O(n^2p)$. We prefer Algorithm 4.1 because of its greater robustness and numerical stability.

5. Implementation of the tensor method

In the previous two sections we presented the main new features of our nonlinear equations method, namely, how we form our quadratic ("tensor") model of the nonlinear function, and how we calculate a root or minimizer of this model efficiently and stably. Algorithm 5.1 outlines the complete algorithm we have implemented to test these ideas. The remainder of this section clarifies various aspects of this algorithm and its computer implementation. Our test results are presented in section 6.

Algorithm 5.1. An Iteration of the Tensor Method

given x_c, $F(x_c)$

1. Calculate $F'(x_c)$ and decide whether to stop. If not,
2. Select the past points to use in the tensor model from among the $\sqrt{n}$ most recent past points.
3. Calculate the second order term of the tensor model, T_c, so that the tensor model interpolates $F(x)$ at all the points selected in step 2.
4. Find the root of the tensor model, or its minimizer (in the l_2 norm) if it has no real root.
5. Select $x_+ = x_c - \lambda_c d_c$, where d_c either is the step calculated in step 4 or the Newton step, using a line search to choose λ_c.
6. Set $x_c \leftarrow x_+$, $F(x_c) \leftarrow F(x_+)$, go to step 1.

Several standard sections of our implementation of Algorithm 5.1 use algorithms from Appendix 1 of Dennis and Schnabel [1983]. The Jacobian is approximated by their finite difference algorithm A5.4.1. The stopping criteria are their algorithm A7.2.3; the algorithm stops if the relative size of $(x_+ - x_c)$ is less than 10^{-9} or the relative size of $(J^T F)$ is less than 10^{-5}.

Step 2 was described completely in section 3; we set the angle Θ mentioned there to 45 degrees. Step 3 calculates the smallest T_c, in the Frobenius norm, for which the tensor model satisfies the interpolation conditions, using the procedure of Theorem 3.2. Before performing these calculations, the steps to the past points $\{s_k\}$ all are normalized to have l_2 norm one. This assures that the linear systems $A = Z^* M^{-1}$ solved in step 3 are well conditioned. The normalization does not alter the tensor algorithm in exact arithmetic; each a_k simply is multiplied by $\|s_k\|_2^3$ and T_c is unchanged.

Step 4 of Algorithm 5.1, the solution of the tensor model, is accomplished by Algorithm 4.1. The QR decomposition of S in step 1 of Algorithm 4.1 is available from the modified Gram-Schmidt algorithm used to select the past points. The QRP decomposition of $\hat{J}_1$ is the standard QR with column pivoting (see e.g. Dongarra et al. [1979]); a column is considered zero if its l_1 norm is less than $10\sqrt{machineps}\ \|F'(x_c)\|_1$. The subproblem (4.6), finding the minimum l_2 norm value of a system of q quadratic equations in p unknowns, is solved using a standard algorithm and analytic derivatives since they are easily available. The algorithm terminates when a root or minimizer of the system of equations is found, or after $8p$ iterations. When $p=q=1$, the problem is solved analytically. The one difficulty with solving this system of quadratics is that it is likely to have multiple roots or minimizers. Among these, it seems sensible to try to find the root closest to the Newton step. We attempt this by making the starting guess for the small system of quadratics the component of the Newton step in this subspace,

$$\hat{d}_2 = -\tilde{J}_3^{-1}\tilde{F}_3,$$

or the linear least squares solution to this system if $q>p$. When $\tilde{J}_3$ isn't well conditioned, a different procedure is used.

Step 5 of Algorithm 5.1 usually consists of a line search in the direction to the root or minimizer of the tensor model obtained in step 4. We use the line search algorithm A6.3.1 from Dennis and Schnabel, requiring sufficient decrease on $\|F(x)\|_2$. In some cases, a line search in the Newton direction is used instead. (The Newton direction is obtained in $O(n^2)$ operations as a byproduct of the algorithm for solving the tensor model.) These cases are when Algorithm 4.1 finds a root of the tensor model that isn't a descent direction for $\|F(x)\|_2$, a very rare occurrence in practice but not precluded in theory; when Algorithm 4.1 fails to find a root or minimizer of the tensor model; and sometimes when Algorithm 4.1

finds a minimizer of the tensor model that is not a root. The complete strategy for determining the search direction is given in Algorithm 5.2 below. The criterion for choosing between the Newton direction and a minimizer of the tensor model is an extension of our strategy for one dimensional problems discussed in section 2.

Algorithm 5.2. Step Selection, Step 5 of Algorithm 5.1

Let J_c = approximation to $F'(x_c)$,

d_T = root or minimizer of the tensor model,

d_N = Newton step $-J_c^{-1}F(x_c)$ if J_c is sufficiently well conditioned, Levenberg-Marquardt step $-(J_c^TJ_c + \varepsilon I)^{-1}J_c^TF(x_c)$ otherwise, where $\varepsilon = \sqrt{n} * machineps * \|J_c^TJ_c\|_1$ (see Dennis and Schnabel [1983])

IF (no root or minimizer of the tensor model was found) OR ((minimizer of tensor model that is not a root was found) AND ($\|M_T(x_c + d_T)\|_2 > \frac{1}{2}\|F(x_c)\|_2$)

THEN

$x_+ \leftarrow x_c + \lambda_c d_N$, $\lambda_c \in (0,1]$ selected by line search

ELSE

$x_+ \leftarrow x_c + d_T$

IF x_+ is not acceptable THEN

IF d_T is a sufficient descent direction

THEN $x_+ \leftarrow x_c + \lambda_c d_T$, $\lambda_c \in (0,1]$ selected by line search

ELSE $x_+ \leftarrow x_c + \lambda_c d_N$, $\lambda_c \in (0,1]$ selected by line search

6. Test Results

We tested our tensor algorithm on a variety of nonsingular and singular problems. We compared it to an algorithm that is identical except that the second order term T_c always is zero. That is, the comparison algorithm is finite difference Newton's method with a line search, except that the Newton step $-J_c^{-1}F(x_c)$ is modified to the approximation to the pseudo-inverse step $-(J_c^TJ_c + \varepsilon I)^{-1}J_c^TF(x_c)$ given in Algorithm 5.2 when $J_c = F'(x_c)$ is singular or sufficiently ill-conditioned. In this section we summarize our test results. Problem by problem data is provided in the appendix.

All our computation was performed on the DEC VAX 11/780 of the University of Colorado Department of Computer Science, using double precision arithmetic.

First we tested our algorithms on the set of nonlinear equations problems in Moré, Garbow, and Hillstrom [1981]. All these problems have $F'(x_*)$ nonsingular, with the exception of Powell's Singular Function where $n=4$ and $\text{rank}(F'(x_*)) = 2$. Our results are given in Table A.1 in the appendix, and summarized in Tables 6.1 and 6.2 below.

Several comments about the test set are necessary. The $100x_0$ case is excluded for three problems, Brown Almost Linear, Rosenbrock, and Wood, because the objective function overflowed at the starting point or during the first iteration. (The largest real number on the VAX is about 10^{38}.) We ran many of the variable dimension problems for n = 10, 20, 30, 40, 50, and include here the n = 30 results, which are representative. The discrete boundary problem is very easy in all cases, we include n = 10 because it is slightly harder than n = 30. The Chebyquad problems also had overflow problems when the dimension and starting point were large, so we include the cases n = 7 and 9 from x_0 and n = 4 from $10x_0$, which seemed representative. Our Variable Dimension problem is a slight alteration of the problem in Moré, Garbow, and Hillstrom, which has $n+2$

equations in n unknowns, we simply eliminate their $n-1^{st}$ and n^{th} equations. (The first n equations in the standard problem are linear.) Our Wood function is the gradient of the standard Wood function for minimization.

The three columns in Tables 6.1 - 6.5 labelled "Average ratio, Tensor method / Standard method," contain the averages of the ratios for each problem of the number of iterations, Jacobian evaluations, and function evaluations (excluding those used for finite difference Jacobians), respectively, used by the tensor method versus the standard method. For example, if the test set contained two problems for which the tensor method required 3 and 5 iterations, respectively, and the standard method 3 and 10 iterations, respectively, then the average iteration ratio would be $\frac{1}{2}(\frac{3}{3}+\frac{5}{10}) = 0.75$. Thus each problem is given equal weight. Since some of the problems are very easy, a second line in each table contains the same statistics using only those problems where the slower method required at least 10 iterations. The three columns labelled "Tensor better -- Standard better -- Tie" are based on a composite consideration of iterations and function evaluations, there were no problems where one method used more function evaluations but fewer iterations. For both algorithms, the number of Jacobian evaluations per problem always is one more than the number of iterations.

The most striking aspect of the test results on nonsingular problems is that the tensor method virtually never is less efficient than the standard method, and almost always is more efficient. In fact, on problems requiring ten or more iterations of the standard method, it is always more efficient. The gains in efficiency are considerable: an average of 21-23% improvement (depending on which measure is used) if all test problems, including some very easy problems where no gains are likely, are considered, and an average of 36-39% improvement on the harder problems. This combination of consistency with reasonable

Table 6.1 -- Summary for Problems with $F'(x_*)$ Nonsingular

Problem Set	Number of Problems	Average Ratio, Tensor Method / Standard Method			Tensor Better	Standard Better	Tie
		Iterations	Jacobian evaluations	Function evaluations			
All problems	28	0.770	0.781	0.793	21	1	6
Harder problems only *	14	0.612	0.636	0.656	14	0	0

Additional problems solved by standard method only: 2
by tensor method only: 1

Table 6.2 -- Summary for Powell's Singular Function

Stopping Tolerance 10^{-3}	3	0.442	0.489	0.510	3	0	0
Stopping Tolerance 10^{-6}	3	0.343	0.385	0.403	3	0	0

Table 6.3 -- Summary for First Singular Test Set with Rank $(F'(x_*)) = n-1$

All Problems	17	0.576	0.609	0.603	15	0	2
Harder Problems Only *	9	0.392	0.429	0.434	9	0	0

Table 6.4 -- Summary for Singular Test Set with Rank $(F'(x_*)) = n-2$

All Problems	13	0.631	0.664	0.729	11	2	0
Harder Problems Only *	7	0.499	0.535	0.542	7	0	0

Additional problems solved by standard method only: 1
by tensor method only: 5

Table 6.5 -- Summary for Second Singular Test Set with Rank $(F'(x_*)) = n-1$

All Problems	16	0.801	0.806	0.849	11	2	3
Harder Problems Only *	11	0.701	0.711	0.787	10	1	0

Additional problems solved by standard method only: 1
by tensor method only: 5

* Problems where slower method required at least 10 iterations

improvement in efficiency indicates that tensor methods may be preferable to standard methods for solving nonsingular systems of nonlinear equations.

Three nonsingular problems were solved by one method and not the other. (We discount the Watson function because the standard method never really found roots and the two methods always went to different regions.) The standard method solved the $10x_0$ Biggs Exp6 problem in 119 iterations while the tensor method didn't solve it in 150, the tensor method solved the Chebyquad $n=9$ problem in 33 iterations while the standard method didn't solve it in 150; the standard method solved the $10x_0$ Wood problem in 60 iterations while the tensor method didn't solve it in 150. This last problem illustrates an occasional difficulty when testing on pathological functions : the tensor method made better progress than the standard method during the first twelve iterations, but reached a point from which neither the tensor method nor the standard method could make reasonable progress. Overall, we noticed no large difference in the success rates of the standard and tensor methods, although the tensor method did have appreciably more successes on two of the three singular test sets.

Table 6.1 also excludes four problems (Box 3D from $10x_0$ and $100x_0$, and Watson from x_0 and $100x_0$) where the two methods converged to different solutions. The tensor method required fewer iterations and function evaluations than the standard method in two of these cases, and the same number in the other two.

On Powell's Singular Function, Table 6.2 shows that the tensor method was 49-56% less expensive, on the average, than the standard method. The stopping tolerance we use, that the relative size of $F'(x)^T F(x)$ must be less than 10^{-5}, is a fairly loose stopping criterion that we believe is typical of tolerances used in practice. Table 6.2 shows that if this stopping tolerance is tightened to 10^{-8} (about the best one can achieve on a VAX using finite difference Jacobians), the

average cost of the tensor method on Powell's Singular Function is 60-66% less than the corresponding cost of the standard method. Presumably this is a reflection of faster local convergence by the tensor method on singular problems, an issue we comment upon later. All our subsequent tests on singular problems use the looser stopping tolerance, 10^{-5}; the improvements by the tensor method over the standard method generally would be more dramatic with a tighter tolerance.

The only other singular nonlinear equations test problems in the literature that we are aware of are the small family proposed by Griewank [1980a]. These four problems all have $n=3$, either a one or two dimensional nullspace for $F'(x_*)$, and either a first or second order singularity. Our results on these problems are given in Table A2 in the appendix, but they don't appear very meaningful, because the standard method failed on 6 of the 12 runs and once converged to a different root than the tensor method. The tensor method was successful in all cases and always was more efficient than the standard method.

We created singular test problems by modifying the nonsingular test problems of Moré, Garbow, and Hillstrom in two different ways. The first is to create problems of the form

$$\hat{F}(x) = F(x) - F'(x_*)\, A(A^TA)^{-1}A^T\,(x-x_*) \tag{6.1}$$

where $F(x)$ is the standard nonsingular test function, x_* is its root, and $A \in R^{n\times k}$ has full column rank with $1<k\le n$. Clearly $\hat{F}(x_*) = 0$ and

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

has rank $n-k$. A disadvantage of this problem class is that $\hat{F}(x)$ may have roots that are not roots of $F(x)$. There is likely to be a manifold of singular Jacobians of $\hat{F}(x)$ in a neighborhood of x_*, but this is not guaranteed. A manifold of singularities is considered desirable because it makes the problems harder and

because we believe it is reflective of most practical singular problems.

We used (6.1) to create two sets of singular problems, with $\hat{F}'(x)$ having rank $n-1$ and $n-2$ respectively, by using

$$A \in R^{n\times 1}, \quad A^T = (1, 1, \ldots, 1)$$

and

$$A \in R^{n\times 2}, \quad A^T = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & -1 & 1 & -1 & \cdots & \pm 1 \end{bmatrix}$$

respectively. We tested our method on the singular versions of all the nonsingular problems except Watson's function, which we excluded because it was quite expensive to run and the two methods never converged to the same root. Our results are given in Tables A3 and A4 in the appendix and summarized in Tables 6.3 and 6.4.

The improvement by the tensor method over the standard method on the problems with $\text{rank}(F'(x_*)) = n-1$ is substantial: an average of 40-43% improvement on all problems and 57-61% on the harder problems. We speculate this is due in part to the tensor method achieving superlinear convergence in this case, and comment further on this at the end of this section. In 9 cases the two methods converged to different roots; in 6 more cases they converged to the same root but not the singular root x_*. These problems are excluded from the summary statistics in Table 6.3; they point out a deficiency of the test set.

On the test set with $\text{rank}(F'(x_*)) = n-2$, the improvement by the tensor method over the standard method was 27-37% on all problems and 46-50% on the harder problems. These are still substantial gains but not as large as when $\text{rank}(F'(x_*)) = n-1$. We speculate in section 7 that our tensor method is not necessarily superlinearly convergent in this case, and mention some modifications that might make it superlinearly convergent when the rank of $F'(x_*)$ is less than $n-1$. The tensor method also solved 5 of the rank $n-2$ prob-

lems that the standard method didn't, the standard method solved one that the tensor method didn't.

Our second method for generating singular test functions from standard nonsingular problems has the desirable property that x_* is a root of the new singular function $\tilde{F}(x)$ if and only if it is a root of the original nonsingular function $F(x)$. This class of functions is described in Theorem 6.1. The functions we generated using this method turned out to be less useful test problems than the singular functions already described, for reasons we will discuss. However they may be a useful class of singular problems for future testing.

Theorem 6.1. Let $F(x)$ $R^n \to R^n$ and $x_* \in R^n$ with $F(x_*) = 0$ and $F'(x_*)$ nonsingular. Define $D_{F2}(x) \in R^{n \times n}$ by

$$D_{F2}(x) = \text{diag}(f_1(x)^2, \ldots, f_n(x)^2)$$

where $f_i(x)$ denotes the i^{th} component function of $F(x)$. Let $A \in R^{n \times k}$, $1 \le k < n$, have full column rank, and let $v \in R^n$ have the property with A that $(Av)[i] = 0$ only if row i of $A = 0$. Define $\tilde{F}(x)$ $R^n \to R^n$ by

$$\tilde{F}(x) = [I - A(A^TA)^{-1}A^T]F(x) + \tfrac{1}{2} D_{F2}(x)\, Av. \tag{6.2}$$

Then $\tilde{F}(x) = 0$ if and only if $F(x) = 0$, and $\tilde{F}'(x_*)$ has rank $n-k$.

Proof : It is obvious that $\tilde{F}(x) = 0$ if $F(x) = 0$. Now suppose $\tilde{F}(x) = 0$, and consider first the case when A has no nonzero rows. Then by definition $w = Av$ has no nonzero components, and

$$0 = v^TA^T\tilde{F}(x) = \tfrac{1}{2} v^TA^T D_{F2}(x)Av = \tfrac{1}{2} \sum_{i=1}^{n} (f_i(x)w[i])^2$$

which implies that $F(x) = 0$. Now consider the case when A has some zero rows, and assume without loss of generality that $A^T = [\, B^T \mid 0 \,]$ where $B \in R^{m \times k}$, $m > k$, has full column rank. Let $G(x)$ $R^n \to R^m$ and $H(x)$ $R^n \to R^{n-m}$ denote the first m

and last $n-m$ components of $F(x)$ respectively, and similarly let $\tilde{G}(x)$ and $\tilde{H}(x)$ denote the first m and last $n-m$ components of $\tilde{F}(x)$. Then

$$\tilde{G}(x) = [I - B(B^T B)^{-1} B^T]\, G(x) + \tfrac{1}{2} D_{G2}(x)\, Dw$$

and

$$\tilde{H}(x) = H(x)$$

where

$$D_{G2}(x) = \mathrm{diag}\{f_1(x)^2, \cdots, f_m(x)^2\}.$$

It follows by the same argument as we used above that $G(x) = 0$ if $\tilde{G}(x) = 0$, and since $\tilde{H}(x)$ and $H(x)$ are identical, $F(x) = 0$ if $\tilde{F}(x) = 0$.

Finally, it is straightforward to verify that

$$\tilde{F}'(x) = [I - A(A^T A)^{-1} A^T] + D_F(x)\, D_w\,]\, F'(x) \tag{6.3}$$

where

$$D_F(x) = \mathrm{diag}\{f_1(x), \cdots, f_n(x)\}, \quad D_w = \mathrm{diag}\{w[1], \cdots, w[n]\}.$$

Thus

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

has rank $n-k$.

$\tilde{F}'(x)$ given by (6.3) almost always has a manifold of singularities around x_*. For example, if A is the first k columns of the identity matrix, then $\tilde{f}_i(x) = w[i]\, f_i(x)^2$, $i = 1, \cdots, k$, so $\tilde{F}'(x)$ is singular whenever any $f_i(x) = 0$, $i = 1, \cdots, k$. More generally, it follows from (6.3) that $\tilde{F}'(x)$ is singular whenever $F(x)$ has $(n+1-k)$ zero components; this usually implies a manifold of singularities around x_* whenever $k > 2$. Finally, it is easy to show that $\tilde{F}'(x)$ is singular whenever

$$z[i] + f_i(x)\, w[i]\, (z^T[i] + (Ay)^T[i]) = 0, \quad i=1, \ldots, m \qquad (6.4)$$

where $z \in R^n$ is any vector in the null space of A^T and y is any vector in R^k. for small z, (6.4) is likely to have solutions.

We used (6.2) to generate two sets of singular test problems with $\text{rank}(F'(x_*)) = n-1$ by setting $k=1$, $v=1$, and $A = (1,1, \ldots,1)$ and $A = (1,0,0, \ldots,0,1)$ respectively. Neither was a very illuminating test set. The problems with $A = (1,1, \ldots,1)$ were too hard for either algorithm, each solved only 30-40% of the problems. In addition, there were numerous overflows due to the squares of the original component functions appearing in the new problems, and the small exponent range of the VAX. The standard method solved 9 problems, overflowed on 7, and failed on 19, the tensor method solved 11, overflowed on 6, and failed on 18. On the few problems solved by both methods, the tensor method was always at least as efficient as the standard method, with improvements ranging from 0 to 62%. The problems with $A = (1,0,0, \ldots,0,1)$ were easier although there still was a considerable number of overflows and failures. The results are given in Table A5 and summarized in Table 6.5. The standard method solved 20 problems, overflowed on 5, and failed on 9. the tensor method solved 24, overflowed on 4, and failed on 6. The average improvement by the tensor method was 15-20%, 25-30% on the problems that required at least ten iterations. These improvements are lower than on the first set of singular problems but do not reflect the higher success rate of the tensor method.

Taken together, our test results seem to indicate that the tensor method is consistently more efficient than standard methods in solving problems where $F'(x_*)$ has a small rank deficiency. We speculate that when $F'(x_*)$ has rank $n-1$, the tensor method is superlinearly convergent in most cases. To check whether this is a reasonable possibility, we examined the sequence of ratios

$$\|x_k - x_*\| \,/\, \|x_{k-1} - x_*\|$$

produced by the standard and tensor methods on problems with $\text{rank}(F'(x_*)) =$

$n-1$. Ratios for a typical problem are given in Table 6.6. In almost all cases, the standard method exhibited local linear convergence with constant very nearly 0.5, as the analysis of various authors mentioned in Section 1 would predict. The local convergence of the tensor method clearly is faster; the final ratio of about 0.01 is typical and might be smaller if analytic Jacobians or tighter stopping tolerances were used. Whether this is superlinear convergence remains to be determined.

Table 6.6 -- Speed of convergence on a typical problem with rank $\hat{F}'(x_*) = n-1$
(Broyden Banded, n = 30, as modified by (6.1), started from $10x_0$)

Iteration (k)	$\|x_k - x_*\| / \|x_{k-1} - x_*\|^2$	
	Tensor Method	Standard Method
1	0.638	0.638
2	0.511	0.626
3	0.502	0.610
4	0.426	0.591
5	0.330	0.570
6	0.204	0.549
7	0.0916	0.532
8	0.0106	0.520
9		0.511
10		0.506
11		0.503
12		0.501
13		0.5007
14		0.5003
15		0.5002
16		0.50009
17		0.50005

It is important to comment that on the test problems where either method had difficulty, it appeared to us that a trust region method that biased short steps towards the steepest descent direction often would have helped. We used a line search algorithm in our tests because we did not want to introduce the unresolved questions about trust region strategies for nonlinear equations into our comparison of the standard and tensor models. It will be important to investigate whether the computational comparison between methods using these two models is similar in a trust region setting.

Finally, we make some comments on details of the tensor algorithm we observed in our testing. The linear independence requirement usually limited the number of past points interpolated at each iteration to a smaller number than the upper bound $\sqrt{n}$. For example on the $100x_0$ Broyden banded problem where $n = 30$, the algorithm used one past point in 83% of the iterations and two past points in the remaining 17%, although it could have used up to six past points, similarly on the x_0 Trigonometric problem where n also is 30, it used one, two, and three past points on 20%, 60%, and 20% of the iterations, respectively. Thus the tensor method seems to obtain surprisingly large improvement from a comparatively small amount of additional information. We tested the algorithm on the nonsingular problems with the linear independence angle reduced to 22.5° (from 45°), there was some fluctuation in the results on individual problems but no overall improvement or deterioration in efficiency, and the number of past points interpolated at each iteration increased somewhat but not dramatically. From past experience, a very small angle, say less than 10°, would give inferior results. The system of linear equations that is solved as part of solving the system of quadratics at each iteration was square and reasonably well conditioned (i.e. $q = p$) almost all of the time; q was greater than p at about 71% of the iterations on the singular and nonsingular Biggs functions, and at about 3% of the iterations on all the other test problems. While the step to the

root of the tensor model is not guaranteed to be in a descent direction, on the nonsingular problems this only occurred on 5 of the 44 problems, and there only about 25% of the time, mostly when the method got stuck in one place.

7. Future research on tensor models

The computational results in section 6 indicate that tensor methods may be preferable to standard methods for solving general systems of nonlinear equations where analytic or finite difference Jacobians are available, and that they may have a substantial advantage on problems where the Jacobian at the solution has a small rank deficiency. To firmly establish such a conclusion, additional testing is required, including tests comparing trust region versions of standard and tensor methods for nonlinear equations. Our inclination is to use dogleg-like methods in these trust region tests.

It would be very helpful to obtain local convergence results for our tensor algorithm applied to singular problems. Hopefully, the algorithm can be shown to converge faster than linearly to a root x_* where $F'(x_*)$ has rank $n-1$ and $F''(x_*)$ obeys appropriate conditions. Related results of this type recently have been obtained by Griewank [1983]. Griewank shows that an algorithm that also bases its iteration on a quadratic model with a simple second order term is locally 2-step q-superlinearly convergent in the above case. His algorithm, however, forms the second order term in the quadratic model using information about the singularity in $F'(x_*)$ that would not be available to general purpose nonlinear equations solvers.

We believe that the tensor method presented in this paper may not always achieve faster than linear convergence on problems where the rank of $F'(x_*)$ is

$n-2$ or less. To justify this remark, suppose $z_1, \ldots, z_m$ is a basis for the null space Z of $F'(x_*)$, where $m>1$. For a method based on a quadratic model to achieve fast convergence, it seems necessary that the second order term in the model be a good approximation to $F''(x_*)$ acting on Z. This seems to require $((m^2+m)/2)$ n-vectors of information, to characterize $F''(x_*)z_iz_j$, $1 \le i \le j \le m$. Our method, however, may not contain this much information even if the past points are in the desired directions, for example, if all the past directions were in Z, our method would interpolate at most m function values. Thus our method does not seem to interpolate enough information to always achieve fast convergence on problems where the dimension of the null space of $F'(x_*)$ is greater than one. This speculation is not well supported by our computational results, however, our tensor method seems to perform almost as well on problems where $F'(x_*)$ has rank $n-2$ as where $F'(x_*)$ has rank $n-1$.

There are several ways to incorporate more information into our tensor model and eliminate the objection raised in the previous paragraph. One is to interpolate values of $F(x_c+s_k)$ at a set of points for which the steps $\{s_k\}$ may not meet the linear independence criterion of section 3, requiring instead that the matrix $S \in R^{n^2 \times p}$ used in Theorem 3.1 meets this criterion. It is easy to show that this procedure would allow choosing up to $((m^2+m)/2)$ directions from an m dimensional subspace while leaving the calculation of the second order term T_c well conditioned. A second alternative is to choose T_c using information from Jacobians at past iterations. We intend to investigate these alternatives.

The methods proposed in this paper can be adapted easily to remain efficient on large, sparse systems of nonlinear equations. In particular, the main additional computational costs of our method are Jacobian-vector products, and presumably these can be performed efficiently when the Jacobian is sparse. Two modifications required would be to select the maximum number of past points

small enough that the cost of solving $p \times p$ linear and quadratic subproblems remains acceptably small, and to use an efficient sparse factorization in the algorithm for solving the tensor model.

The methods discussed in this paper also could be applied with very little modification to nonlinear least squares problems. Nonlinear least squares algorithms virtually always use analytic or finite difference Jacobians so the requirements of the tensor methods presented in this paper are no restriction in this case. The augmentation of the linear model by a second order term would lead to natural extensions of Gauss-Newton or Levenberg-Marquardt methods, and tensor methods might require fewer iterations and function valuations than these methods, especially on problems where the Jacobian at the solution is rank deficient. It is not clear how tensor methods for nonlinear least squares would perform on large residual problems, and whether there is any reason to prefer them to quasi-Newton methods like those of Dennis, Gay, and Welsch [1981] in this case.

We are currently developing extensions of our tensor methods to secant methods for nonlinear equations, and to unconstrained minimization. Neither extension is straightforward. In secant methods for nonlinear equations, analytic or finite difference Jacobians are not available, but it is possible to interpolate all the function values $F'(x_c + s_k)$ used in section 3 with a linear model (see e.g. Gay and Schnabel [1978]). To create a useful second order term it is necessary to interpolate function values in (nearly) dependent directions. The primary difficulty in extending tensor methods to unconstrained minimization is that for problems where the Hessian matrix at the minimizer is singular, an approximation not to the third but to the fourth derivative matrix is necessary to speed convergence. This is because the projection of the third derivative onto the null space of the Hessian must be zero at such a minimizer. In addition, all derivative approximations for minimization must be symmetric. Our

solutions to these difficulties will be reported in future papers

Acknowledgements

We thank Richard Byrd and Andreas Griewank for many helpful discussions related to this paper

8. References

D. W. Decker, H. B. Keller, and C. T. Kelley [1982], "Convergence rates for Newton's method at singular points", preprint, Department of Mathematics, North Carolina State University.

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

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

D. W. Decker and C. T. Kelley [1982], "Convergence acceleration for Newton's method at singular points", *SIAM Journal on Numerical Analysis* 19, pp. 219-229.

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

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

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

J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart [1979], *LINPACK Users Guide*, SIAM Publications, Philadelphia.

R. Fletcher [1980], *Practical Method of Optimization, Vol. 1, Unconstrained Optimization*, John Wiley and Sons, New York.

P. Frank [1982], "A second-order local model for solution of systems of nonlinear equations", M.S. Thesis, Department of Computer Science, University of Colorado at Boulder.

C. B. Garcia and T. Y. Li [1980], "On the number of solutions to polynomial systems of equations", *SIAM Journal on Numerical Analysis* 17, pp. 540-546.

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

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

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

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

A. O. Griewank [1983], "On solving nonlinear equations with simple singularities or nearly singular solutions", preprint, Department of Mathematics, Southern Methodist University.

A. O. Griewank and M. R. Osborne [1981], "Newton's method for singular problems when the dimension of the null space is >1", *SIAM Journal on Numerical Analysis* 18, pp. 145-150.

H. B. Keller [1970], "Newton's method under mild differentiability conditions", *Journal of Computing and Systems Sciences* 4, pp. 15-28.

C. T. Kelley and R. Suresh [1982], "A new acceleration method for Newton's method at singular points", preprint, Department of Mathematics, North Carolina State University.

J. J. Moré B. S. Garbow, and K. E. Hillstrom [1981], "Testing unconstrained optimization software", *ACM Transactions on Mathematical Software* 7, pp. 17-41.

D. E. Muller [1956], "A method for solving algebraic equations using an automatic computer", *Mathematical Tables and Other Aids to Computation* 10, pp. 208-215.

L. B. Rall [1966], "Convergence of the Newton process to multiple solutions", *Numerische Mathematik* 9, pp. 23-37.

G. W. Reddien [1978], "On Newton's method for singular problems", *SIAM Journal on Numerical Analysis* 15, pp. 993-996.

G. W. Reddien [1980], "Newton's method and high order singularities", *Comput Math Appl.* 5, 79-86.

K. A. Stordahl [1980], "Unconstrained minimization using conic models and exact second derivatives", M.S. thesis, Department of Computer Science, University of Colorado at Boulder.

J. E. Traub [1964], *Iterative Methods for the Solution of Equations*, Prentice-Hall, Englewood Cliffs, New Jersey.

Appendix

Tables A1 - A5 contain the results of the tests described in Section 6. The problems in Table A1 are from Moré, Garbow, and Hillstrom [1981]. The problems in Tables A3 - A5 are singular modifications, described in Section 6, of the problems in Moré, Garbow, and Hillstrom. The starting points used for all these problems are the ones suggested by Moré, Garbow, and Hillstrom, the third column of each table designates whether the starting point is x_0, $10x_0$, or $100x_0$, where x_0 is the point listed in Moré, Garbow, and Hillstrom. The problems in Table A2 are taken from Griewank [1980a].

The two columns in each table labelled "$\frac{1}{2}\|F(x_*)\|_2^2$" give half the sum of squares of the component functions values at the final iterate for the standard method and tensor method, respectively, using abbreviated notation : e.g. .43-12 means 0.43×10^{-12}. If the method failed on a problem, this column instead contains one of the following alphabetic codes :

OF -- method overflowed

D -- divergence detected (5 consecutive very long steps)

F -- method failed to find a root in 150 iterations

S -- method stalled at non-root

The rightmost column in each table, labelled "same x_* ?", contains a Y (yes) if the two methods converged to the same point on this problem, a N (no) otherwise. Only problems that converged to the same root are included in the statistics in Tables 6.1 - 6.5.

In Tables A3 - A5, the two columns labelled "n.s. x_* ?" (short for "nonsingular x_*?"), contain a Y (yes) if the method converged to the same root as the corresponding nonsingular problem in Table A1, a N (no) otherwise. Only problems were both methods converged to the same root and the same root as in the nonsingular case are included in the statistics in Tables 6.3 - 6.5.

Table A1 -- Results on More', Garbow, and Hillstrom Test Set

Function	n	x_0	Standard Method			Tensor Method			
			Fcn vals	Itns	$\frac{1}{2}\|F(x_*)\|_2^2$	Fcn vals	Itns	$\frac{1}{2}\|F(x_*)\|_2^2$	same x_*?
Biggs Exp6	6	1	209	104	.97-18	157	70	.13-12	Y
		10	228	119	22-5	200	150	F	-
		100	306	150	F	305	150	F	-
Box 3D	3	1	5	4	.10-16	4	3	.10-11	Y
		10	20	14	.56-19	12	8	18-12	N
		100	4	3	.16-8	4	3	.48-10	N
Brown Alm. Linear	10	1	15	11	.57-12	10	7	.38-11	Y
		10	11	9	.25-15	10	7	.40-11	Y
Broyden Banded	30	1	6	5	.12-15	5	4	.12-11	Y
		10	12	11	.16-22	9	8	92-13	Y
		100	17	16	.95-14	14	13	.71-20	Y
Broyden Tridiag.	30	1	5	4	.56-18	5	4	20-12	Y
		10	8	7	.47-13	6	5	93-20	Y
		100	11	10	.64-10	6	5	.87-16	Y
Chebyquad	7	1	11	7	.38-16	8	6	.63-18	Y
	9	1	332	150	F	63	33	68-15	-
	4	10	51	35	.90-20	44	24	.11-11	Y
Discrete Boundary	10	1	3	2	.48-15	3	2	23-16	Y
		10	4	3	.15-17	4	3	13-18	Y
		100	9	8	.10-13	8	7	13-10	Y
Discrete Integral	30	1	3	2	.18-12	3	2	21-14	Y
		10	4	3	.49-15	4	3	27-16	Y
		100	9	8	.40-11	9	8	.70-15	Y
Helical Valley	3	1	12	9	.17-20	11	8	.78-18	Y
		10	17	13	.23-21	16	11	16-18	Y
		100	21	16	.22-14	16	11	12-22	Y
Powell Singular	4	1	9	8	19-7	4	3	25-15	Y
		10	13	12	.43-8	6	5	13-16	Y
		100	16	15	.10-7	10	8	26-16	Y
Rosen-brock	2	1	27	14	.10-7	12	7	.14-20	Y
		10	5	3	0	6	4	.17-14	Y
Trigono-metric	30	1	31	16	.19-12	21	11	.14-4	Y
		10	326	150	F	145	64	S	-
		100	334	150	F	132	62	S	-
Variable Dimension (Altered)	10	1	14	13	.24-9	6	5	.20-13	Y
		10	15	14	10-8	5	4	.55-18	Y
		100	19	18	10-8	14	11	.22-12	Y
Watson	31	1	5	4	.10-6	5	4	.48-15	N
		10	41	25	.25-6	378	150	F	-
		100	45	28	.35-6	27	17	.80-11	N
Wood Gradient	4	1	24	17	.23-26	11	9	.27-17	Y
		10	105	61	.30-24	366	150	F	-

Table A2 -- Results on Griewank's Singular Functions

Function	n	x_0	Standard Method			Tensor Method			
			Fcn vals	Itns	$\frac{1}{2}\|F(x_*)\|_2^2$	Fcn vals	Itns	$\frac{1}{2}\|F(x_*)\|_2^2$	same x_*?
Dimension of Nullspace = 1 Order of Singularity = 1	3	1	9	8	.2-10	6	5	.1-10	Y
		10	15	14	1-10	7	6	.9-16	N
		100	21	19	.1-10	18	14	.7-10	Y
Dimension of Nullspace = 1 Order of Singularity = 2	3	1	11	9	.3-10	7	6	.5-11	Y
		10	332	150	F	11	9	.3-11	-
		100	297	150	F	17	13	.1-11	-
Dimension of Nullspace = 2 Order of Singularity = 1	3	1	11	9	.3-10	7	6	.5-11	Y
		10	332	150	F	11	9	.3-11	-
		100	297	150	F	17	13	.1-11	-
Dimension of Nullspace = 2 Order of Singularity = 2	3	1	11	9	.3-10	7	6	.5-11	Y
		10	332	150	F	11	9	.3-11	-
		100	297	150	F	17	13	.1-11	-

Table A3 -- Results on First Singular Test Set with Rank $(F'(x_*)) = n-1$

Function	n	x_0	Standard Method				Tensor Method				
			Fcn vals	Itns	$\frac{1}{2}\|F(x_*)\|_2^2$	n.s. x_*?	Fcn vals	Itns	$\frac{1}{2}\|F(x_*)\|_2^2$	n.s. x_*?	same x_*?
Biggs Exp6	6	1	74	52	.12-10	N	411	150	F	N	N
		10	33	19	.56-17	N	69	38	.14-6	N	N
		100	6	5	0	N	305	150	F	N	N
Box 3D	3	1	10	9	.96-13	Y	6	5	.57-15	Y	Y
		10			OF	-	11	10	.42-7	N	-
		100	4	3	.46-15	N			OF	-	-
Brown Alm. Linear	10	1	7	6	.33-7	Y	5	4	.41-7	Y	Y
		10	22	21	.51-7	Y	9	7	.39-9	Y	Y
Broyden Banded	30	1	12	11	.88-11	Y	6	5	.65-14	Y	Y
		10	18	17	.11-10	Y	10	9	.54-15	Y	Y
		100	24	23	.37-11	Y	14	13	.48-11	Y	Y
Broyden Tridiag	30	1	9	8	.15-9	Y	6	5	.80-12	Y	Y
		10	14	13	.10-9	Y	6	5	.15-13	Y	Y
		100	17	16	.32-9	Y	5	4	.16-12	Y	Y
Chebyquad	7	1	11	7	.38-16	Y	8	6	.98-14	Y	Y
	9	1	66	33	.29-13	N	294	103	.60-13	N	N
	9	10	300	106	.90-16	N	78	42	.19-17	N	N
Discrete Boundary	10	1	3	2	.16-9	Y	3	2	.14-9	Y	Y
		10	5	4	.36-10	Y	5	4	.20-13	Y	Y
		100	9	8	.13-10	N	7	6	.10-13	N	Y
Discrete Integral	30	1	5	4	.22-9	Y	4	3	.20-11	Y	Y
		10	6	5	.82-9	Y	5	4	.68-12	Y	Y
		100	10	9	.65-18	N	9	8	.32-23	N	Y
Helical Valley	3	1	8	7	.17-16	N	7	6	.64-14	N	Y
		10	7	6	.38-14	N	7	6	.27-13	N	Y
		100	7	6	.10-14	N	7	6	.70-22	N	Y
Rosen-brock	2	1	15	14	.16-13	Y	4	3	.47-14	Y	Y
		10	17	16	.77-13	Y	5	4	.22-28	Y	Y
Trigono-metric	30	1	19	11	.10-10	N	24	13	.62-10	N	N
Variable Dimension (Altered)	10	1	14	13	.24-9	Y	10	9	.46-15	N	N
		10	15	14	.63-12	N	12	11	.18-13	N	N
		100	18	17	.14-15	N	11	9	.18-15	N	Y
Wood Gradient	4	1	21	20	.22-15	Y	12	11	.41-18	Y	Y
		10	26	25	.94-15	Y	15	14	.15-20	N	N

Table A4 -- Results on Singular Test Set with Rank $(F'(x_*)) = n-2$

Function	n	x_0	Standard Method				Tensor Method				
			Fcn vals	Itns	$\frac{1}{2}\|F(x_*)\|_2^2$	n.s. x_*?	Fcn vals	Itns	$\frac{1}{2}\|F(x_*)\|_2^2$	n.s. x_*?	same x_*?
Biggs Exp6	6	1	27	17	.5-6	N	32	18	.4-10	N	N
		10	306	150	F	-	390	150	F	-	-
		100	6	5	D	-	6	5	D	-	-
Box 3D	3	1	8	7	.1-10	Y	12	7	.7-12	Y	Y
		10	11	10	.1-12	Y	12	11	.2-12	Y	N
		100			OF	-			OF	-	-
Brown Alm Linear	10	1	7	6	.1-10	N	6	4	.9-13	N	N
		10	19	16	.2-15	N	37	17	.8-17	N	N
Broyden Banded	30	1	91	99	.8-12	Y	8	7	.5-11	N	N
		10	304	150	F	-	21	18	.1-12	Y	-
		100	25	21	.9-19	N	68	35	.1-13	Y	N
Broyden Tridiag	30	1	10	9	.9-12	Y	14	10	.1-13	Y	Y
		10	298	150	F	-	7	6	.7-13	N	-
		100	19	18	.3-11	Y	8	7	.2-19	N	N
Chebyquad	7	1	10	9	.9-11	Y	8	7	.4-13	Y	Y
	9	1	335	150	F	-	12	9	.1-19	N	-
	4	10	301	150	F	-	53	32	.4-20	N	-
Discrete Boundary	10	1	5	4	.2-10	Y	4	3	.6-11	Y	Y
		10	9	8	.3-10	Y	5	4	.8-10	Y	Y
		100	12	11	.1-13	N	306	150	F	-	-
Discrete Integral	30	1	8	7	.3-10	Y	5	4	.6-10	Y	Y
		10	11	10	.6-10	Y	8	7	.2-9	Y	Y
		100	9	8	.6-18	N	16	14	.3-10	Y	N
Helical Valley	3	1	15	14	.1-12	Y	8	7	.6-21	Y	Y
		10	15	14	6-13	Y	7	6	.1-13	Y	Y
		100	15	14	6-13	Y	7	6	.6-13	Y	Y
Rosen-brock	2	1	11	10	.1-8	Y	4	3	.4-9	Y	Y
		10	13	12	.5-8	N	5	4	.1-11	N	Y
Trigono-metric	30	1	318	150	F	-	12	7	.3-14	N	-
Variable Dimension (Altered)	10	1	14	13	.2-9	Y	8	7	.6-15	Y	Y
		10	16	15	.6-10	N	12	8	.2-9	N	Y
		100	20	19	.4-10	N	11	8	.4-13	N	Y
Wood Gradient	4	1	21	20	.1-16	Y	14	12	.4-16	Y	Y
		10	27	26	.2-16	Y	49	28	.4-5	N	N

Table A5 -- Results on Second Singular Test Set with Rank $(F'(x_*)) = n-1$

Function	n	x_0	Standard Method				Tensor Method				
			Fcn vals	Itns	$\frac{1}{2}\|F(x_*)\|_2^2$	n.s. x_*?	Fcn vals	Itns	$\frac{1}{2}\|F(x_*)\|_2^2$	n.s. x_*?	same x_*?
Biggs Exp6	6	1			OF	-	351	150	F	-	-
		10	292	150	F	-			OF	-	-
		100	304	150	F	-	303	150	F	-	-
Box 3D	3	1			OF	-			OF	-	-
		10			OF	-			OF	-	-
		100			OF	-	5	4	2-10	Y	-
Brown Alm. Linear	10	1			OF	-			OF	-	-
		10	40	39	9-10	Y	44	33	2-10	N	N
Broyden Banded	30	1	13	12	2-10	Y	10	9	2-11	Y	Y
		10	303	150	F	-	15	14	3-12	Y	-
		100	302	150	F	-	37	28	1-15	Y	-
Broyden Tridiag.	30	1	12	11	4-9	Y	10	9	1-15	Y	Y
		10	28	21	8-9	Y	12	11	4-18	Y	Y
		100	25	24	5-8	Y			OF	-	-
Chebyquad	7	1	11	7	4-16	Y	8	6	6-18	Y	Y
	9	1	305	150	F	-	192	71	5-12	Y	-
	4	10	293	150	F	-			OF	-	-
Discrete Boundary	10	1	4	3	7-9	Y	4	3	2-10	Y	Y
		10	5	4	2-8	Y	5	4	4-11	Y	Y
		100	12	11	6-8	Y	22	15	1-15	Y	Y
Discrete Integral	30	1	3	2	5-10	Y	3	2	5-10	Y	Y
		10	5	4	3-7	Y	6	5	4-15	Y	Y
		100	15	14	1-17	Y	12	11	4-14	Y	Y
Helical Valley	3	1	239	123	7-13	Y	15	11	3-13	Y	Y
		10	20	18	1-12	Y	16	12	4-15	Y	Y
		100	22	21	1-12	Y	21	16	4-17	Y	Y
Rosen-brock	2	1	313	150	F	-	45	23	1-15	Y	-
		10	322	150	F	-	31	18	1-17	Y	-
Trigono-metric	30	1	49	25	1-9	N	23	12	3-10	N	N
Variable Dimension (Altered)	10	1	28	27	6-10	Y	17	16	1-10	Y	Y
		10	32	31	4-10	Y	20	19	2-9	Y	Y
		100	37	36	3-10	Y	37	27	6-9	Y	Y
Wood Gradient	4	1	292	150	F	-	329	150	F	-	-
		10	40	38	2-12	N	452	150	F	-	-

END

DATE
FILMED

6-83

DTIC