abstract

We propose an algorithm for implementing Newton's method for a general nonlinear system $f(x) = 0$ where the linear systems that arise at each step of Newton's method are solved by a preconditioned Krylov subspace iterative method. The algorithm requires only function evaluations and does not require the evaluation or storage of the Jacobian matrix. Matrix-vector products involving the Jacobian matrix are approximated by directional differences. We develop a framework for constructing preconditionings for this inner iterative method which do not reference the Jacobian matrix explicitly. We derive a nonlinear SSOR type preconditioning which numerical experiments show to be as effective as the linear SSOR preconditioning that uses the Jacobian explicitly.

Nonlinearly Preconditioned Krylov Subspace Methods for Discrete Newton Algorithms[1]

Tony F. Chan[2]
Kenneth R. Jackson[3]

Report # 259
January 22, 1983

[1]This work was supported in part by Air Force Grant AFOSR-81-0193, by Department of Energy Grant DE-AC02-81ER10996, and by National Sciences and Engineering Research Council of Canada Grant U0133.

[2]Computer Science Dept., Yale Univ., Box 2158, Yale Station, New Haven, CT 06520.

[3]Computer Science Dept., Univ. of Toronto, Toronto, Canada M5S 1A4.

Table of Contents

1. Introduction

One of the most common methods for solving an n by n nonlinear systems of the form

$$f(x) = 0 \tag{1}$$

is **Newton's method:**

Start with an initial guess x_0.

Repeat until convergence:

$$\text{Solve} \quad J(x_i)\, \delta x = - f(x_i) \ , \tag{2}$$

$$\text{Set} \quad x_{i+1} = x_i + \delta x \ , \tag{3}$$

where $J(x)$ is the Jacobian matrix $f_x(x)$.

For many large problems, most of the work is done in solving the linear system (2). This is usually accomplished by evaluating and LU-factoring the Jacobian J and backsolving for δx. However, when J is large and sparse, it is natural to consider iterative methods, which usually require much less storage. Moreover, truncated forms of Newton's method [4, 22], in which the linear system (2) is only solved approximately by an iterative method, can be implemented easily.

Among the most popular and successful iterative methods for large sparse systems are methods based on the Krylov subspace, for example, the Chebychev method [14, 18] and the conjugate gradient method [14, 15]. These methods all have the property that only matrix-vector products involving the coefficient matrix are needed. In the context of applications to Newton's method, the matrix-vector product Jv can be approximated by the directional difference $(f(x+dv)-f(x))/d$, where d is the scalar difference interval. This has the advantage of requiring only function evaluations and avoiding explicit evaluation and storage of the Jacobian matrix. Thus, it is well-suited for large sparse problems, especially if the Jacobian is difficult to evaluate or store.

Often, it is desirable to precondition the Krylov subspace methods as the rate of convergence is often unaccepatably slow otherwise, especially for ill-conditioned problems. However, since most preconditioning techniques require J explicitly [2, 6, 7, 13, 16], it is not obvious how to apply them in the context of directional differencing, as J is not explicitly available. We address this problem in this paper.

After a review of discrete Newton algorithms and Krylov subspace methods in Section 2, we develop in Section 3 a framework for constructing nonlinear preconditionings that do not require J explicitly. We then give in Section 4 a specific preconditioning algorithm that is based on nonlinear SSOR sweeps. The efficiency of this nonlinear preconditioned Krylov subspace Newton method is discussed in Section 5. Numerical experiments in Section 6 show that this nonlinear preconditioning is as effective as the linear SSOR preconditioning that require J explicitly. Some conclusions are given in Section 7.

2. Discrete Newton Algorithms and Krylov Subspace Methods

Newton's method is among the best methods for solving general nonlinear systems. Part of the reason for its success is its quadratic rate of local convergence. Also, it is often more robust than competing methods, for example quasi-Newton methods, especially for optimization problems [12, 22].

These advantages of Newton's method are offset by the requirement that, at each step, the user compute the Jacobian matrix explicitly and solve the linear system (2). For problems for which the Jacobian is difficult to calculate (for example in optimization the Jacobian corresponds to the Hessian matrix which consists of *second* derivatives of the cost function), methods have been proposed which do not require the user to compute the Jacobian explicitly. Among these are Quasi-Newton methods [5], Nonlinear Conjugate Gradient methods [8] and Discrete Newton methods [12, 22, 23].

Discrete Newton methods are perhaps the most natural of the three classes of methods, in that they retain the form of Newton's method but approximate the Jacobian matrix by finite differences of the function values. These differences are often taken along the unit coordinate directions:

$$\partial f \,/\, \partial x_i \approx (\, f(x+de_i) - f(x)\,) \,/\, d \quad , \tag{4}$$

where d is an appropriately chosen difference interval.[4] It is easy to see that it takes n function evaluations to obtain an approximation for J. However, for large and sparse problems, other directions may be more efficient, in terms of both work and storage. At least two approaches using finite differences are possible.

The first approach is to order the columns of J into groups according to the sparsity pattern of

[4]For reliable and automatic techniques for chosing the finite difference interval d, we refer the reader to [11, 12].

J so that the nonzero elements in each group can be evaluated with only *one* function evaluation by finite differencing along a carefully chosen direction. For matrices with highly structured sparsity patterns, this can result in a significant reduction in the number of function evaluations needed to obtain an approximation for J. For example, a tridiagonal Jacobian can be approximated by three function evaluations, independent of the dimension of the matrix. Curtis, Powell and Reid [17] are credited as being the first to make such an observation and they proposed a heuristic algorithm for obtaining a good (but generally not optimal) column ordering for matrices with arbitrary sparsity patterns. Recently, there has been a lot of interest in developing more efficient finite-differencing schemes by solving related graph coloring problems [3, 19, 26, 27]. We call a Newton method employing this approach to approximate the Jacobian the standard discrete Newton algorithm.

The second approach is rather different in that it does not attempt to compute an explicit approximation to the Jacobain matrix at all. Methods in this class use a Krylov subspace iterative method (e.g. the conjugate gradient method) to solve approximately the linear system (2) at each step of Newton's method. In the application of these iterative methods, one does not need to access the Jacobian matrix explicitly. Instead, all that is required is the ability to form matrix-vector products of the form Jv for a given vector v. The central theme of the methods in this class is to approximate matrix-vector products of this form by a directional difference:

$$J(x)\,v \approx (\,f(x + d\,v) - f(x)\,)\,/\,d\,. \tag{5}$$

Note that the direction v is usually unknown *a priori* and depends on the iterates in the Krylov subspace method. Only function evaluations are needed. This method is especially attractive if only a few matrix-vector products are needed at each Newton step, as is often the case in truncated Newton algorithms. For general dense matrices that arise in optimization problems, O'Leary [23] has shown that this method takes fewer operations than the standard discrete Newton method when $n > 39$. Garg and Tapia [9] and Nash [22] have also used this technique in optimization problems. Gear and Saad [10] have employed this idea successfully in solving stiff ordinary differential equations. We shall call this the Directional Differencing Discrete Newton (DDDN) Method.

Traditionally, Krylov subspace methods were developed for matrices with special properties, e.g. symmetric positive definiteness. However, for a general function f, the Jacobian J does not necessarily possess these properties. Even for optimization problems, although J is often symmetric, it need not be positive definite. Recently, the application of Krylov subspace methods to general linear systems has attracted a lot of research interest and significant progress has been achieved. For a survey of recent research in this area the reader is refered to [7, 16].

In this paper, we are concerned with preconditioning the Krylov subspace methods in order to accelerate their convergence. Most preconditionings are based upon a splitting of the form $J = M - N$ and can be implemented by supplying a routine for computing the matrix-vector product $M^{-1}v$ for a given vector v. A good preconditioning is one for which M is a good approximation to J and $M^{-1}v$ is inexpensive to compute.

The matrix M is usually defined in terms of the elements of J. This is true, for example, of the most commonly used incomplete LU-factorizations [13, 20] and the SSOR-type preconditionings [1, 2]. This creates a problem in the context of applications to Newton's method with directional differencing since the Jacobian matrix J is not available explicitly. Computing the elements of J explicitly in order to compute and store the preconditioning M would take away the advantage of the directional differencing. It is thus desirable to be able to apply a preconditioning algorithm that requires function evaluations only and does not reference the Jacobian explicitly. In the next section, we derive a class of nonlinear preconditionings with this property.

3. Nonlinear Preconditioning

In the context of a Krylov subspace method, a preconditioning M is needed only to compute matrix-vector products of the form

$$w = M^{-1}v \, . \tag{6}$$

That is, we want to be able to solve the linear system

$$M\,w = v \, . \tag{7}$$

The optimal preconditioning, at least as far as convergence is concerned, is the choice $M = J$. However, this leads to solving the linear system

$$J\,w = v \tag{8}$$

which is as difficult as the original problem (2). On the other hand, it shows that, for any preconditioning M, $M^{-1}v$ *can be viewed as an algorithm for obtaining an approximate solution of (8).* With this in mind, we can approximate the term Jw in (8) by a directional difference and solve

$$F(w) \equiv (f(x+dw) - f(x)) / d - v = 0 \tag{9}$$

for w. At first glance, this appears to be as difficult as the original nonlinear problem (1). The crucial observation, however, is that *any approximate method for solving (9) will constitute a preconditioning M for J, where M is generally a nonlinear operator.*

We note that (9) does not involve J explicitly. Also, there are many algorithms for solving nonlinear systems that do not involve the Jacobian J explicitly. Therefore, we have a framework for constructing many possible preconditionings that require function evaluations only.

One may argue that nothing is gained by this approach, because any method used to solve (9) approximately can also be applied to the original equation $f(x) = 0$ as well. There is a difference, however. Independent of the method used to solve (9), the outer method is still Newton's method, and thus for example local quadratic convergence is achievable [4]. If the same method applied to (9) is applied to (1) directly, that method will determine the convergence of the outer loop. The resulting method may converge much less quickly and, in particular, quadratic convergence may be lost. See for example Mittelmann [21] who considered an algorithm for nonlinear finite element problems in which nonlinear Gauss-Seidel iterates are accelerated by the conjugate gradient method.

Whether Newton's method is the appropriate choice for the outer algorithm probably depends on the particular problem, but there is some evidence [22] that it is among the best choice for general nonlinear problems. In any case, the general framework in this section allows *any* algorithm for solving (9) to be used as a preconditioning for the Krylov subspace method used to solve the linear systems in Newton's method. If the method used for the nonlinear preconditioning does not require explicit reference to the Jacobian, then the overall directional differencing discrete Newton algorithm can be implemented using only function evaluations. In the next section, we derive one such nonlinear preconditioning.

4. A Nonlinear SSOR Preconditioning

A class of methods for solving a nonlinear system $F(w) = 0$ that does not use the Jacobian matrix explicitly is the class of

Nonlinear Relaxation Methods [24]:

```
Start with some initial guess for w and a relaxation factor $\omega$.
Repeat until convergence:
  Do i = 1, ...., n , n , ...., 1
    (0) Save $(w_i)_{old} \Leftarrow w_i$.
    (1) Solve for $w_i$ in $F_i(w_1,...,w_i,...,w_n) = 0$
        assuming that all other components of w are fixed at
        their current values.
    (2) Set $w_i \Leftarrow (1-\omega)\,(w_i)_{old} + \omega\, w_i$.
  End do
End Repeat
```

The above algorithm can be applied directly to (9), provided we specify the initial guess for w, the relaxation factor ω, and the number of times the outer loop is to be repeated. By looking at the linear SSOR preconditioning for J, we shall show that the initial guess w = 0 and one iteration of the outer loop are appropriate choices.

Let J be written as

$$J = D - L - U$$

where D, L and U are the diagonal, the strictly lower triangular, and strictly upper triangular parts of J, respectively. The linear SSOR preconditioning for J can be written as [1, 2]

$$M = \omega\,(2 - \omega)\,(D - \omega L)\,D^{-1}\,(D - \omega U)\,.$$

Thus, $M^{-1}v$ can be computed by two backsolves with triangular matrices. However, there is another well-known interpretation for $M^{-1}v$. Consider the following two stage iteration:

$$(D - \omega L)\,w_1 = (\,(1 - \omega)\,D + \omega\,U\,)\,w_0 + \omega\,v, \tag{10}$$

$$(D - \omega U)\,w_2 = (\,(1 - \omega)\,D + \omega\,L\,)\,w_1 + \omega\,v. \tag{11}$$

It is easily verified that, if $w_0 = 0$, then $w_2 = M^{-1}v$. On the other hand, it is also well-known that this two stage iteration is equivalent to the nonlinear SSOR algorithm applied to the linear system $F(w) \equiv Jw - v$ with one iteration of the outer loop [24]. For nonlinear systems, F(w) in (9) is an approximation to Jw - v. Thus, it follows that, for the nonlinear SSOR preconditioning, $w_0 = 0$ is an appropriate initial guess and the outer loop should be iterated once. It also follows that, *if f is linear, the nonlinear SSOR preconditioning reduces to the linear SSOR preconditioning for J.*

We note that the problem to be solved in Step (1) of the nonlinear SSOR preconditioning algorithm is a scalar one. Any one dimensional nonlinear equation solver can be used. Moreover, the problem in Step (1) does not have to be solved exactly. For example, if one step of Newton's method is used [24], then we have the following algorithm:

```
Algorithm NSSORP: (One Step Nonlinear SSOR - Newton Preconditioning)
  Given $\omega$ and v. Returns preconditioned v in w.
  $w \Leftarrow 0$.
  For $i = 1, .., n, n, ... , 1$
      Compute $(DJ)_i \Leftarrow$ the i-th diagonal elements of $J(x+dw)$.
      Set $w_i \Leftarrow w_i - \omega F_i(w) / (DJ)_i$ .
```

The diagonal elements $(DJ)_i$ of the Jacobian J can either be supplied by the user or be approximated by finite differences. We note that, for efficiency, Algorithm NSSORP requires the function f to be supplied in component form.

5. Efficiency

In this section, we compare the efficiency of the following four discrete Newton algorithms:

(a) Form J using a sparse Jacobian algorithm,
use a Krylov subspace method to solve (2).

(b) Form J using a sparse Jacobian algorithm,
use a linear SSOR preconditioned Krylov subspace method to solve (2).

(c) Use a Krylov subspace method to solve (2), with directional differencing for Jv.

(d) Use a preconditioned Krylov subspace method to solve (2),
with directional differencing for Jv and preconditioned with Algorithm NSSORP.

Note that all four methods do not require the explicit Jacobian. Method (d) is the method that we proposed in Section 4. Method (c) is the same except the Krylov subspace method is not preconditioned. Methods (a) and (b) are the corresponding methods except the Jacobian J is approximated by a sparse Jacobian evaluation algorithm. These are the discrete Newton methods that one might consider using if the Jacobian is not available explicitly.

Concerning the storage requirements, all four methods need storage for the Krylov subspace method, which is usually O(n). Methods (a) and (b), however, require additional storage for the Jacobian J. Thus, methods (c) and (d) might be preferred for problems where the storage of J presents difficulties, e.g. J is too large for the machine, or J is dense with no exploitable structures for storage, or J has a nontrivial sparsity structure which is not convenient to handle.

Next, we compare work. We define one function evaluation of f to be one evaluation of each component of f. First we consider the case that function evaluations are expensive compared to matrix-vector operations. All four methods need one function evaluation to evaluate the right hand side of the linear system (2) in Newton's method. In addition, Method (c) requires one function evaluation per step of the Krylov subspace method (to approximate Jv). Method (d) requires three function evaluations per step (one to approximate Jv and two for the nonlinear preconditioning). The total number of function evaluations of course depends on the number of iterations taken by the Krylov subspace method to achieve the desired accuracy. Let us denote by I_c and I_d the number of iterations taken by method (c) and (d) respectively. For methods (a) and (b), the number of function evaluations needed to evaluate J depends on the sparsity structure of J and the sparse evaluation algorithm. Let us denote this number by s. The total number of function evaluations per Newton step are summarized in Table (5-1). Methods (a) and (b) can easily be modified to reduce the number of function evaluations required at each step by employing the chord variant of Newton's method which uses the same Jacobian approximation for several steps. We note, however, that methods (c) and (d) can be modified in a similar way to reduce the number of function evaluations required at each step by reusing the projection generated from previous iterations [25]. Here, though, for simplicity, we assume these savings are not exploited.

Table 5-1: Total number of function evaluations per Newton step.

Method	a	b	c	d
func. eval.	s+1	s+1	I_c+1	$3I_d$+1

s	:	number of function evaluations needed to evaluate J.
I_c	:	number of iterations taken by Krylov subspace method for method (c).
I_d	:	number of iterations taken by Krylov subspace method for method (d).

We can make the following observations from the table. If the number of Krylov subspace iterations needed is small compared to s, then it is more efficient to use directional differencing (methods (c) and (d)). Such situations arise, for example, in the early stages of a truncated Newton algorithm [4] or if J has an advantageous distribution of eigenvalues (e.g. stiff ODE's with only a few large eigenvalues [10]). On the other hand, if the number of iterations can be reduced by a factor of more than about three by the nonlinear SSOR preconditioning, then it pays to use method (d) instead of method (c). Whether the preconditioning can achieve this depends on the eigenvalue distribution of J and on the accuracy desired. We note that for model partial differential equations, the linear SSOR preconditioning can be shown to reduce the number of iterations by a factor of $O(h^{-1/2})$, where h is the mesh size [2]. Thus, for this class of problems, the benefits of preconditioning is more pronounced for larger problems.

Next, we consider the case that function evaluations are inexpensive compared to matrix-vector operations. For example, if one function evaluation costs approximately the same as forming the matrix-vector product Jv (which is often the case for difference methods for PDEs with simple coefficients), then the costs of methods (c) and (d) would be about the same as that of methods (a) and (b), respectively.

6. Numerical Experiments

We have performed some numerical experiments with the following model nonlinear partial differential equation

$$- u_{xx} + 2\,b\,(e^{u})_{x} + c\,e^{u} = R(x), \quad 0 < x < 1 ,$$

with homogeneous Dirichlet boundary conditions. This problem is discretized on a uniform grid with n intervals using standard centered finite differencing. The function R(x) is constructed so that the discrete solution has each component equal to 1. The resulting n by n nonlinear algebraic system is solved by the methods discussed in earlier sections. The coefficients b and c allow us to control the asymmetry and the diagonal dominance of the Jacobian matrix respectively. We note that in practice it would not be cost effective to solve a one dimensional problem like this one by an iterative method. However, the distribution of the eigenvalues of J for this problem is similar to the distribution of eigenvalues for problems in higher dimensions. Thus, this problem is quite suitable for testing purposes.

For the Krylov subspace method, we used the Direct Incomplete Orthogonalization Method (DIOM) of Saad [28]. This method is related to Arnoldi's method for computing eigenvalues of general nonsymmetric matrices [29]. At the i-th step, an orthogonal basis for the i-th Krylov subspace is generated and the new solution is constructed so that the corresponding residual is orthogonal to this subspace. Saad has improved the basic method so that the same code can be adapted to symmetric positive definite, symmetric indefinite and general nonsymmetric problems. For the preconditioning, we fix $\omega = 1$ (Symmetric Gauss Seidel).

All computations were carried out on a DEC 20, with a machine precision of about 10^{-8} (27 bit mantissa). For the finite difference interval used in the directional differencing, we use $d = 10^{-4}$.

For the stopping criteria of the inner loop, we use a simple form of truncated Newton strategy:

$$\| J(x_i)\,\delta x + f(x_i) \| \,/\, \| f(x_i) \| < 10^{-i-1}.$$

For the convergence of the outer Newton iteration, we use

$$\| f(x_i) \| < 10^{-4} \text{ and } \| \delta x \| < 10^{-4} + 10^{-3} \|x_i\|.$$

All norms used are infinity norms.

The main objective of the experiments is to compare the performance of directional differencing methods to that of the corresponding methods using the explicit Jacobian. For this purpose, we also tested the following two methods:

(e) Exact Jacobian used in Krylov subspace method.

(f) Exact Jacobian used in Krylov subspace method,
linear SSOR preconditioning based on the exact Jacobian.

Another objective is to show the effectiveness of the nonlinear preconditioning. Experiments were carried out with methods (c), (d), (e) and (f) for various values of n, b and c. The results are summarized in Table (6-1). In the table, the columns labelled $||f||$ and $||error||$ denote the values of $||f(x)||$ and the error in the solution x at termination of the Newton iteration.

Tests 1-4 and 5-8 show that the directional differencing algorithms behave almost exactly like the corresponding algorithm with the explicit Jacobian. They show that the nonlinear preconditioning (Algorithm NSSORP) is as effective as the linear SSOR preconditioning based on the explicit Jacobian.

Tests 9-10 show the efffectiveness of the preconditioning for a more nonsymmetric problem (larger b). The reduction of the number of DIOM iterations caused by the preconditioning is more pronounced than for more symmetric problems (e.g. Test 5 and 7 and Tests 13-14), especially near convergence of the Newton iteration where higher accuracy is needed in the inner loop.

Tests 11-12, 15-16 are similar to Tests 1 and 3 (b and c are the same), except that the size n of the system is larger. They show that the unpreconditioned algorithms converge rather slowly and take a lot of DIOM iterations to achieve the desired accuracy. On the other hand, the preconditioned algorithms converged rather fast and took much fewer iterations. These results confirm that, at least for PDE type problems, the benefit of preconditioning is more pronounced for larger problems.

We have also performed tests on some other non-PDE type problems (e.g. the PEN1 Problem

Table 6-1: Numerical Results for Model Problem.

Test	n	b	c	Method	No. of DIOM iters. at each step of Newton's method 1	2	3	4	5	6	||f||	||error||
1	20	1	1	c	20	45	61				.3 (-5)	.4 (-5)
2				e	20	45	64				.2 (-6)	.1 (-5)
3				d	8	10	10				.3 (-5)	.3 (-5)
4				f	8	10	10				.2 (-6)	.1 (-5)
5	20	1	10	c	20	25	30	40			.1 (-5)	.3 (-5)
6				e	20	25	34	35			.3 (-6)	.2 (-5)
7				d	7	7	8	9			.3 (-5)	.3 (-5)
8				f	7	7	8	9			.2 (-6)	.1 (-5)
9	20	10	1	c	20	30	50	55			.5 (-6)	.2 (-6)
10				d	7	5	7	6	7	9	.2 (-5)	1.0 (-5)
11	40	1	1	c	40	*120	108	*120			.6 (-4)	.4 (-3)
12				d	15	24	26				.6 (-5)	.5 (-4)
13	60	0	1	c	30	71	74				.8 (-5)	.9 (-4)
14				d	14	28	31				.3 (-5)	.6 (-4)
15	60	1	1	c	60	110	*140	*140	*140	*140	.5 (-4)	.2 (-3)
16				d	22	55	78				.5 (-5)	.3 (-4)

Notation: a (b) means a x 10^b.
*I means that the desired accuracy was not achieved after I iterations.

tested in [22, 23]). The trends are similar to the ones reported here, namely, that the directional differencing algorithms behave almost exactly like the corresponding algorithms with the exact Jacobian.

7. Conclusion

In this paper, we propose an algorithm for implementing Newton's method that is particularly attractive for certain large sparse nonlinear systems. A preconditioned Krylov subspace method is used to solve the linear systems that arise at each step of Newton's method. However, the Jacobian matrix is neither stored nor referenced explicitly. The user supplies the nonlinear function only. The main contribution of the paper is a framework in which various nonlinear preconditionings can be derived which require function evaluations only. We derived and tested a nonlinear SSOR preconditioning. Numerical results show that it is as effective as the linear SSOR preconditioning that uses the exact Jacobian explicitly. These tests also show that the nonlinerarly preconditioned discrete Newton algorithm is as efficient and as robust as the standard discrete Newton algorithm.

References

[1] O. Axelsson.
A Generalized SSOR Method.
BIT 13:443-467, 1972.

[2] R. Chandra.
Conjugate gradient methods for partial differential equations.
PhD thesis, Dept. of Computer Science, Yale U., 1978.

[3] T.F. Coleman and J.J. More.
Estimation of Sparse Jacobian Matrices and Graph Coloring Problems.
Technical Report ANL-81-39, Argonne National Laboratory, Argonne, Illinois, 1981.

[4] R.S. Dembo, S. Eisenstat and T. Steihaug.
Inexact Newton Methods.
SIAM J. Numer. Anal. 18(2):400-408, 1982.

[5] J.E. Dennis and J.J. More.
Quasi-Newton Methods, Motivation and Theory.
SIAM Review 19:46 - 89, 1977.

[6] T.Dupont, R.P. Kendall and H.H. Rachford.
An Approximate Factorization Procedure for Solving Self-Adjoint Elliptic Difference Equations.
SIAM Journal on Numerical Analysis 6 :753-782, 1968.

[7] Howard C. Elman.
Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations.
PhD thesis, Yale University, 1982.
Techreport # 229.

[8] R. Fletcher and C.M. Reeves.
Function Minimization by Conjugate Gradients.
Comput. J. 7:149-154, 1964.

[9] N.K. Garg and R.A. Tapia.
QDN: A Variable Storage Algorithm for Unconstrained Optimization.
Technical Report , Dept. of Math. Sci., Rice Univ., Houston, TX., 1980.

[10] W. C. Gear and Y. Saad.
Iterative Solution of Linear Equations in ODE Codes.
Technical Report UIUCDCS-R-81-1054, Dept. of Computer Science, Univ. of Illinois, Urbana, Illinois., 1981.
To appear in Siam J. of Sci. and Stat. Comp.

[11] P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright.
A Procedure for Computing Forward-Difference Intervals for Numerical Optimization.
Technical Report SOL 81-25, Systems Optimization Lab., Dept. of Operations Research, Stanford University, Stanford, 1981.

[12] P.E. Gill, W. Murray and M. Wright.
Practical Optimization.
Academic Press, New York, 1981.

[13] Ivar Gustafsson.
A Class of First Order Factorization.
BIT 18:142-156, 1978.

[14] Louis A. Hageman and David M. Young.
Applied Iterative Methods.
Academic Press, New York, 1981.

[15] M.R. Hestenes and E. Stiefel.
Methods of Conjugate Gradient for solving Linear Systems.
Journal of Research of the National Bureau of Standards 49:409-436, 1952.

[16] Kang Chang Jea.
Generalized Conjugate Gradient Acceleration of Iterative Methods.
PhD thesis, University of Texas at Austin, 1982.

[17] A.R. Curtis, M.J.D. Powell and J.K. Reid.
On the Estimation of Sparse Jacobian Matrices.
J. Inst. Maths. Applics. 13:117-119, 1974.

[18] Thomas A. Manteuffel.
The Tchebychev Iteration for Nonsymmetric Linear Systems.
Numer. Math. 28:307-327, 1977.

[19] S.T. McCormick.
Optimal Approximation of Sparse Hessians and Its Equivalence to a Graph Coloring Problem.
Technical Report SOL 81-22, Dept. of Oper. Res., Stanford Univ., Palo Alto, Tx., 1981.

[20] J.A. Meijerink and H.A. van der Vorst.
An Iterative Solution Method for Linear Systems of which the Coefficient Matrix is a Symmetric M-Matrix.
Mathematics of Computation 31:148-162, 1977.

[21] Hans D. Mittelmann.
On the Efficient Solution of Nonlinear Finite Element Equations I.
Numer. Math. 35:277-291, 1980.

[22] Stephen G. Nash.
Truncated Newton Method.
PhD thesis, Stanford University, 1982.

[23] Dianne P. O'Leary.
A Discrete Newton Algorithm for Minimizing a Function of Many Variables.
Math. Progr. 23:20 - 33, 1982.

[24] J.M. Ortega and W.C. Rheinboldt.
Iterative Solution of Nonlinear Equations in Several Variables.
Academic Press, New York, 1970.

[25] B.N.Parlett.
A new look a the Lanczos algorithm for solving symmetric systems of linear equations.
Lin. Alg. and its Appl. 29:323-346, 1980 .

[26] M.J.D. Powell and Ph. L. Toint.
On the Estimation of Sparse Hessian Matrices.
SIAM J. Numer. Anal. 16:1060-1073, 1979.

[27] John D. Ramsdell.
Structural Analysis of Large Sparse Systems of Nonlinear Equations with Applications to Fire Modeling.
PhD thesis, Harvard University, 1982.

[28] Y. Saad.
Practical use of some Krylov subspace methods for solving indefinite and unsymmetric linear systems.
Technical Report 214, Yale University, New Haven, Connecticut, 1982.
To appear in SIAM J. on Sci. and Stat. Comp.

[29] J.H. Wilkinson.
The Algebraic Eigenvalue Problem.
Oxford University Press, London, 1965.