REPORT NO.
UCB/SESM-81/04

STRUCTURAL ENGINEERING AND
STRUCTURAL MECHANICS

A NEWTON-LANCZOS METHOD FOR SOLUTION OF NONLINEAR FINITE ELEMENT EQUATIONS

by
BAHRAM NOUR-OMID

Faculty Investigator: R. L. Taylor

Report to:
Naval Civil Engineering Laboratory

OCTOBER 1981

DEPARTMENT OF CIVIL ENGINEERING
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA

Structural Engineering and Structural Mechanics Division
Report No. UCB/SESM/-81/04

A NEWTON-LANCZOS METHOD FOR SOLUTION OF NONLINEAR FINITE ELEMENT EQUATIONS

by

Bahram Nour-Omid

Department of Civil Engineering
University of California
Berkeley, California 94720

Final Report to

Naval Civil Engineering Laboratory
Port Hueneme
California 93043

Contract No. N62583/81 M R419

October 1981

ACKNOWLEDGEMENTS

This research work was sponsored by Naval Civil Engineering Laboratory under Contract No. N62583/81 M R419.

I would like to express my gratitude to Professor R. L. Taylor who supervised this work and gave me encouragement and freedom to pursue my own related interests. He provided support and much valuable advice on this project. I am deeply indebted to Professor B. N. Parlett for giving me the privilege of working under his supervision. I appreciate his patience, understanding and generosity with his valuable time during the course of this research. I am also grateful to Professors J. L. Sackman and A. Der Kiureghian for their encouragement and many helpful and enjoyable discussions.

TABLE OF CONTENTS

1. INTRODUCTION

During the past few years, considerable effort has been directed towards the solution of nonlinear initial-boundary value problems in structural mechanics. The finite element method is one of the more popular approaches employed to reduce continuum problems to nonlinear algebraic, discrete problems. Finite element methodology is well documented in the literature (e.g., see[23]) and attention here is devoted to procedures which may be employed to solve the resulting nonlinear algebraic problem. The general class of problems of interest include both material and geometric nonlinearities.

A brief review of the application of the finite element to problems in nonlinear continuum mechanics is presented in Section 2. The result is a large set of nonlinear algebraic equations which must be solved for the state variables (e.g., displacements and stresses). Newton's method is commonly employed to solve the nonlinear equations. In Section 3 discussion of Newton's method, including operation count estimates, and advantages and disadvantages, is presented. In addition modified Newton's method is also discussed in this section. In general, Newton's method possesses ideal characteristics of local convergence and stability but is normally too expensive to employ for solving large finite element problems. On the other hand, modified Newton's method has desirable operation count estimates for each iteration but becomes expensive to employ because of its low convergence rate characteristics. Recently, Matthies and Strang [10] have suggested that quasi-Newton methods be used to solve nonlinear finite element problems. In Section 3.3 some of the quasi-Newton's methods which have been used in optimization methods, such as Broyden's method and the Broyden-Fletcer-Goldfarb-Shano (BFGS) method, are discussed and a discussion of the advantages and disadvantages for these methods is presented.

The use of iterative equation solvers as an inner loop for the Newton's method has been suggested before and recently Eisenstat completed a study of the convergence properties of these methods. In section 4 the use of the Lanczos algorithm for iterative solution of linear system of equations is described. The relation of the Lanczos algorithm to the conjugate gradient method and the effect of preconditioning on the algorithm is demonstrated. The preconditioned Lanczos algorithm is used to solve the linear system of equations arising at each iterate of Newton's method. This results in a technique where the rate of convergence of the method can be varied from a linear to a quadratic convergence rate by means of a specified tolerance. In section 5 a detailed description of the Newton-Lanczos method is presented. The above methods are tested on various nonlinear problems in structural mechanics and the results are tabulated.

2. THE FINITE ELEMENT METHOD FOR NONLINEAR PROBLEMS

Since its introduction, the Finite Element Method, FEM, has become one of the most widely used techniques for solving various problems in continuum mechanics. The FEM has established a systematic procedure for discretization of complex structures which results in a large set of nonlinear equations.
The source of nonlinearities are of two types:

(a) material nonlinearity

(b) geometrical nonlinearity

Addressed here are problems associated with both nonlinear constitutive relations and nonlinearities due to large displacements.

In the following section a summary of the formulation of problems in continuum mechanics is presented. The resulting equations are then discretized using the Galerkin method. The usual convention, that implies summation with respect to a repeated index in a term, is adopted.

2.1. Equations of Kinematics

Consider a body with reference configuration at time $t = 0$, denoted by B_0, that is deformed by a motion to a current configuration B. Figure 2.1 illustrates this motion. Using a common rectangular cartesian coordinate system, the Lagrangian description of the motion can be expressed as

$$x = \hat{x}(\mathbf{X},t) = \mathbf{X} + \mathbf{u}(\mathbf{X},t) \tag{2.1.1}$$

where x is the current position at time t of a particle with position vector $\mathbf{X}$ at time $t = 0$ and $\mathbf{u}$ is the displacement the particle undergoes in this motion.
The deformation gradient of the motion is defined by

$$\mathbf{F} = \frac{\partial \hat{\mathbf{x}}}{\partial \mathbf{X}} = \mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}} \qquad (2.1.2)$$

The matrix $\frac{\partial \mathbf{u}}{\partial \mathbf{X}}$ is known as the displacement gradient. Thus, the strain tensor, at a point in the current configuration, referred back to the initial configuration can be written as

$$\mathbf{E} = \tfrac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}) \qquad (2.1.3)$$

Using equations 2.1.2 and 2.1.3 the strain tensor can be written directly in terms of the displacement and the initial position vector

$$\mathbf{E} = \tfrac{1}{2}\left[\left(\frac{\partial \mathbf{u}}{\partial \mathbf{X}}\right)^T + \frac{\partial \mathbf{u}}{\partial \mathbf{X}} + \left(\frac{\partial \mathbf{u}}{\partial \mathbf{X}}\right)^T \frac{\partial \mathbf{u}}{\partial \mathbf{X}}\right] \qquad (2.1.4.a)$$

or in indicial notation

$$E_{IJ} = \tfrac{1}{2}(u_{I,J} + u_{J,I} + u_{K,I}u_{K,J}) \qquad (2.1.4.b)$$

where E_{IJ} are components of the strain tensor E, and u_I are components of displacement vector **u**; $(\,)_{,J}$ denotes partial differentiation with respect to the J-th component of the position vector **X**.

For transient problems a statement of initial conditions is required. That is

$$\mathbf{u}(\mathbf{X},0) = \mathbf{d}_0(\mathbf{X}) \qquad (2.1.5.a)$$

$$\dot{\mathbf{u}}(\mathbf{X},0) = \mathbf{v}_0(\mathbf{X}) \qquad (2.1.5.b)$$

2.2. Momentum Balance

Equilibrium conditions between internal resisting forces and externally applied forces must be satisfied, at least in the weak form, whether the displacements or strains are large or small. The momentum balance equation is given by

$$\mathrm{Div}\mathbf{P} + \mathbf{b} = \rho_0\ddot{\mathbf{u}} \qquad (2.2.1.a)$$

or

$$P_{Ii,I} + b_i = \rho_0\ddot{u}_i \qquad (2.2.1.b)$$

where P_{Ii} are the components of the first Piola-Kirchhoff (unsymmetric) stress tensor $\mathbb{P}$, b_i are components of the body force $\mathbb{b}$, ρ_0 is the density in the initial configuration, and a superposed dot, $(\dot{\ })$, denotes differentiation with respect to time.

A weak form of the momentum balance equations, equivalent to virtual work, may be constructed by multiplying 2.2.1.b by an arbitrary function W_i, integrating over the domain of interest Ω, using integration by parts on the stress term and setting the result equal to zero. The result is

$$\int_{\Omega}(W_{i,I}P_{Ii} - W_i b_i + W_i \rho_0 \ddot{u}_i) d\Omega - \int_{\Gamma_2} W_i \bar{\tau}_i d\Gamma = 0 \tag{2.2.2}$$

where in addition to previously defined quantities, $\bar{\tau}_i$ is a specified traction and Γ_2 is the part of the boundary where traction is specified. For equation 2.1.2 to be valid, W_i must vanish on Γ_1, that part of the boundary where displacements are specified (i.e., $u_i = \bar{u}_i$ on Γ_1). The traction is related to stress through

$$\bar{\tau}_i = N_I P_{Ii} \tag{2.2.3}$$

where N_I are direction cosines of the outward normal to Γ in the initial configuration.

The components of the first Piola-Kirchhoff stress tensor, P_{Ii}, is replaced by S_{IJ}, the components of the symmetric second Piola-Kirchhoff stress tensor, S, using the relation

$$P_{Ii} = S_{IJ} F_{iJ} \tag{2.2.4}$$

where F_{iJ} are the components of the deformation gradient matrix. This results in

$$\int_{\Omega}(W_{i,I}S_{IJ}F_{iJ} - W_i b_i + W_i \rho_0 \ddot{u}_i) d\Omega - \int_{\Gamma_2} W_i \bar{\tau}_i d\Gamma = 0 \tag{2.2.5}$$

2.3. Constitutive Relation

In general the deformation gradient of the motion, the temperature and the temperature gradient determines the stress state. The effects of inelastic material behavior can be modeled through internal variables.

For isothermal deformation the constitutive equation can be written as

$$S_{IJ} = \rho_0 \frac{\partial \hat{\Psi}}{\partial E_{IJ}} \tag{2.3.1}$$

where Ψ is the Helmholz free energy defined at (**X** , t) through the strain tensor and the internal variables

$$\Psi(\mathbf{X},t) = \hat{\Psi}(\mathbf{E},\boldsymbol{\alpha}) \tag{2.3.2}$$

The form of the constitutive relation stated above is currently under debate. A more comprehensive discussion can be found in reference [14]. In addition to the previously defined quantities $\boldsymbol{\alpha}$ is a vector of internal variables. Further, a constitutive relation of the form

$$\dot{\boldsymbol{\alpha}} = \gamma(\mathbf{S},\boldsymbol{\alpha}) \tag{2.3.3}$$

is assumed for the internal variables. It is sometimes more convenient to express these relations with strain rate as the dependent variable

$$\dot{\mathbf{E}} = \mathbf{C}\dot{\mathbf{S}} + \Lambda\dot{\boldsymbol{\alpha}} \tag{2.3.4}$$

where **C** is the instantaneous elastic compliance tensor, and Λ is the inelastic compliance tensor which depends on the stress state, the internal variables and the velocity gradient, **L**, in such a way as to satisfy objectivity

$$\Lambda = \hat{\Lambda}(\mathbf{S},\boldsymbol{\alpha},\mathbf{L}) \tag{2.3.5}$$

This form of the model is most suitable for nonlinear viscoelastic or elastic/viscoplastic materials. The constitutive equations are both nonlinear and rate dependent. Writting 2.3.4 in indicial notation

$$\dot{E}_{IJ} = C_{IJKL}\,\dot{S}_{KL} + \Lambda_{IJ(R)}\dot{\alpha}_{(R)} \tag{2.3.6}$$

where R=1,2, $\cdots$,N and N is the number of internal variables.

The weak form of this equation can similarly be obtained by multiplying 2.3.6 by an arbitrary function V_{IJ}, and integrating over the domain Ω. This leads to

$$\int_{\Omega} V_{IJ}(\dot{E}_{IJ} - C_{IJKL}\dot{S}_{KL} - \Lambda_{IJ(R)}\dot{\alpha}_{(R)})d\Omega = 0 \tag{2.3.7}$$

where $\dot{\alpha}_{(R)}$ is determined form equation 2.3.3 .

2.4. Finite Element Discretization

In a finite element model, the domain Ω is divided into elements Ω_e. By sampling the solution for the field variables, such as displacements or stresses, at a number of points in the elements, known as nodes, an approximate solution can be constructed. When the stresses are known explicitly in terms of the strains it is possible to choose the displacements as the only independent variable and the solution to the nodal parameter can be obtained using the weak form of the equilibrium equation, 2.2.5 . However, in cases when the constitutive relation is more complex it is simpler to use a weak form of this equation, 2.3.7 , and use both stresses and displacements as primary variables.
In each element the approximations for u_i and S_{IJ} which are C^0-continuous and piecewise constant (C^{-1}-continuous), respectively, over Ω_e, are in the form

$$\mathbf{u} = \sum_A N_A(\mathbf{X})\mathbf{u}_A(t) \tag{2.4.1.a}$$

and

$$\mathbf{S} = \sum_B M_B(\mathbf{X})\mathbf{S}_B(t) \tag{2.4.1.b}$$

where $N_A(\mathbf{X})$ and $M_B(\mathbf{X})$ are shape functions over the domain Ω_e satisfying the C^0 and C^{-1} continuity requirements cited above. The arbitrary weighting functions are also approximated using these shape functions, as

$$\mathbf{W} = \sum_A N_A(\mathbf{X})\mathbf{W}_A \tag{2.4.2.a}$$

and

$$\mathbf{V} = \sum_B M_B(\mathbf{X})\mathbf{V}_B \tag{2.4.2.b}$$

where $\mathbf{W}_A$ and $\mathbf{V}_B$ are arbitrary nodal parameter. Using a temporal finite difference scheme, such as Newmark's method [13], the time derivatives in equations 2.2.5 and 2.3.7 can be removed.

The result of the application of the finite element method and the time-stepping procedure is a large set of nonlinear algebraic equations at each time step, t_m, with nodal displacements and stress quantities as the the unknowns. The dependence of these equations on the stress quantities can be removed through static condensation at the element level resulting in nodal displacements as the unknown parameters.
These equations can be written as

$$\mathbf{f}[\mathbf{x}(t_m)] = 0 \tag{2.4.3}$$

where $\mathbf{x}$ is a vector of the state variables, which consists of all the nodal displacements, $\mathbf{u}_I(t_m)$, and $\mathbf{f}$ is a vector function of $\mathbf{x}$ obtained by application of the Galerkin method to equations 2.2.5 and 2.3.7.

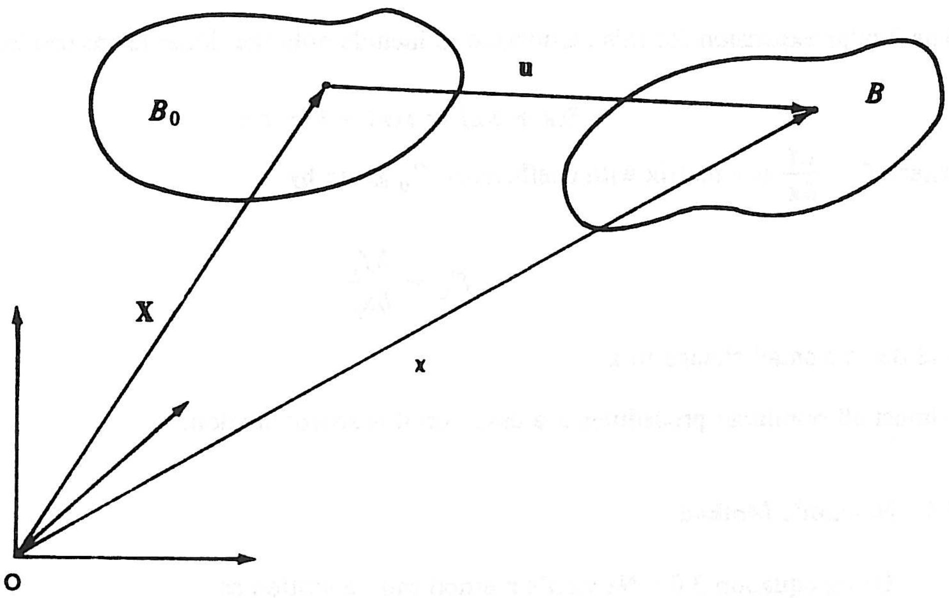


Figure 2.1 Composition of Motion

3. SOLUTION OF THE DISCRETE NONLINEAR EQUATIONS

Employing the finite element method and a time stepping procedure to discretize the nonlinear structural mechanics problem leads to the large set of nonlinear algebraic equation given by 2.4.3 . Presented here, are a number of methods normally employed for the solution of this equation.

The Taylor expansion for this , truncated to include only the linear terms can be written as

$$\mathbf{f}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{f}(\mathbf{x}) + \mathbf{f}'(\mathbf{x})\Delta\mathbf{x} \qquad (3.0.1)$$

where $\mathbf{f}' = \dfrac{\partial \mathbf{f}}{\partial \mathbf{x}}$ is a matrix with coefficients f'_{ij} given by

$$f'_{ij} = \frac{\partial f_i}{\partial x_j}$$

and $\Delta\mathbf{x}$ is a small change in $\mathbf{x}$.

Almost all nonlinear procedures are based on this approximation.

3.1. Newton's Method

Using equation 3.0.1 Newton's method can be written as

$$\mathbf{f}(\mathbf{x}_{k+1}) \approx \mathbf{f}(\mathbf{x}_k) + \mathbf{A}_k\mathbf{d}_k \qquad (3.1.1)$$

where k is an iteration number, $\mathbf{d}_k$ is a change in $\mathbf{x}_k$ called the step direction, and $\mathbf{A}_k$ is the Jacobian or tangent matrix of $\mathbf{f}$ defined by

$$\mathbf{A}_k = \left[\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right]_{\mathbf{x}_k} \qquad (3.1.2)$$

Newton's method consists of setting 3.1.1 equal to zero, solving for $\mathbf{d}_k$ from

$$\mathbf{A}_k\mathbf{d}_k = -\mathbf{f}(\mathbf{x}_k) \qquad (3.1.2)$$

and setting

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k \qquad (3.1.3)$$

Newton's method requires the initial guess $\mathbf{x}_0$ to be in a domain D, called the domain of attraction, such that convergence to the exact solution, $\mathbf{x}^*$ in D, of equation 2.4.3 does occur. Furthermore, $\mathbf{f}$ must be differentiable in D and the tangent matrix must be non-singular at the solution. In practice it is often desirable to modify 3.1.3 to

$$\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{d}_k \tag{3.1.4}$$

where s_k is a scalar step size which is used to enhance stability of the algorithm. The value of s_k is determined from a line search as described in the following section.

The algorithm for implementing Newton's method consists of choosing a good initial guess $\mathbf{x}_0$ and repeat following steps for $k = 0,1,2, \cdots$ until convergence is achieved:

(1) given $\mathbf{x}_k$ compute $\mathbf{f}(\mathbf{x}_k)$

(2) compute the tangent matrix $\mathbf{A}_k$

(3) solve $\mathbf{A}_k \mathbf{d}_k = -\mathbf{f}(\mathbf{x}_k)$ for $\mathbf{d}_k$

(4) compute s_k from a line search

(5) update $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{d}_k$

(6) test for convergence and terminate if converged

There are several procedures which may be used to terminate the iteration.
These include:

(a) $\|\mathbf{f}(\mathbf{x}_k)\| < \epsilon_1 \max_i \|\mathbf{f}(\mathbf{x}_i)\|$

(b) $\|\mathbf{d}_k\| < \epsilon_2 \max_i \|\mathbf{d}_i\|$

(c) $|\mathbf{d}_k \cdot \mathbf{f}(\mathbf{x}_k)| < \epsilon_3 \max_i |\mathbf{d}_i \cdot \mathbf{f}(\mathbf{x}_i)|$

where ϵ_i are small positive constants. In this work method (a) was used. Method (c) is only applicable for symmetric positive definite Jacobians.

3.1.1. Line Search for Newton's Method

In the above presentation a step size, s_k, has been included to improve the stability of Newton's method. The magnitude of s_k is obtained in such a way as to minimize a norm of the residual, $\mathbf{f}(\mathbf{x}_k)$. A common procedure used, is to solve the following nonlinear scalar equation for s_k

$$G(s_k) = \mathbf{d}_k^T\mathbf{f}(\mathbf{x}_k+s_k\mathbf{d}_k) = 0 \qquad (3.1.5)$$

It is important to note that equation 3.1.5 need not be solved accurately since $\mathbf{x}_{k+1}$ is itself only an approximation and also, in general, the additional function evaluations are costly in finite element analyses.

In fact, when possible, the line search should be omitted when $\mathbf{x}_k$ is close to a solution.

3.1.2. Operation Counts for Newton's Method

For purposes of subsequent cost comparisons, operation counts for Newton's method are estimated to indicate the relative efforts needed in each step. For an n-dimensional $\mathbf{x}$, the operation count estimates are:

(a) computation of $\mathbf{f}(\mathbf{x}_k)$	$O(n)$
(b) computation of $\mathbf{A}_k$	$O(n^2)$
(c) direct solution of equations	
triangular decomposition of $\mathbf{A}_k$	$O(n^3)$
forward reduction/backsubstitution	$O(n^2)$
(d) line search	$O(n)$
(e) update of solution	$O(n)$
(f) convergence test	$O(n)$

While order of magnitude estimates are given, substantial differences in real effort exist between, for example (a) and (d). The majority of effort is associated with steps (a) to (d), and only comparisons between these steps are meaningful. There is an order of magnitude increase in step (b) over step (a) or in step (c) over step (b). It is important to note that these estimates are somewhat problem dependent, for example step (a) can cost as much as step (b). Consequently, considerable efficiency can be achieved by eliminating or reducing the number of expensive steps required.

3.1.3. Convergence of Newton's Method

The rate of convergence of an algorithm will determine how fast the iteration $\mathbf{x}_k$ approaches a solution $\mathbf{x}^*$. An acceptable algorithm must be at least linearly convergent [3]; that is, given a solution $\mathbf{x}^*$, then

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leqslant \alpha \|\mathbf{x}_k - \mathbf{x}^*\| \tag{3.1.6}$$

where α is a positive constant less than unity. Although 3.1.6 ensures that the error norm is reduced by the factor α in each iteration, to be competitive it is generally acknowledged that an algorithm must have better than linear convergence. When Newton's method has continuously differentiable $\mathbf{f}$ and a solution $\mathbf{x}^*$ in D, then the error norm satisfies the stronger condition

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leqslant \alpha \|\mathbf{x}_k - \mathbf{x}^*\|^q \tag{3.1.7}$$

where q is a positive scalar such that $1 < q < 2$. This is called super-linear convergence. In addition, for cases where $\mathbf{f}$ is twice differentiable in the neighborhood of $\mathbf{x}^*$, with non-singular Jacobian, Newton's method is quadratically convergent with $q = 2$.

For finite elment applications, Newton's method will almost always have at least super-linear convergence, and most problems are such that quadratic convergence will be achieved.

3.1.4. Advantages and Disadvantages of Newton's Method

Newton's method has at least two very desirable properties:

(i) Any $\mathbf{x}_k$ in D results in an $\mathbf{x}_{k+1}$ in D; consequently, the method is stable and convergent once an iterate is in D.

(ii) The method possesses at least super-linear convergence and often quadratic convergence [3].

On the negative side, we note that:

(i) If D is small, then the initial approximation may have to be very close to $\mathbf{x}^*$ to get inside D. We do not know when $\mathbf{x}_0$ is in D.

(ii) The triangular decomposition of the Jacobian matrix is very costly in large finite element problems.

The requirement of a good initial guess may be avoided in part by using line searches and, for quasi-static problems, an evolution of the load application [23]. In the sequel we address the possibility of reducing computational factorizations of the tangent matrix.

3.2. Modified Newton's Method

For large systems of equations, the main cost in Newton's method is the triangular decomposition of the Jacobian matrix. The use of a previously factored Jacobian is often advocated in place of the current tangent matrix. The algorithm is given as, for k=0,1,2, $\cdots$ repeat

```
(1) given $\mathbf{x}_k$ compute $\mathbf{f}(\mathbf{x}_k)$
(2) solve $\mathbf{A}_i\mathbf{d}_k = -\mathbf{f}(\mathbf{x}_k)$, where$\mathbf{A}_i$ is the tangent matrix at step $i < k$
(3) compute $s_k$ from a line search
(4) update $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k\mathbf{d}_k$
(5) test for convergence and terminate if converged
```

Comparing this algorithm with that of Newton's method, we observe that the $O(n^2)$ operations to compute the Jacobian matrix and the $O(n^3)$ operations to compute the triangular decomposition are avoided. The algorithm still requires $O(n^2)$ operations to perform a resolution using the triangular factors of $\mathbf{A}_i$. However, these savings are achieved at the expense of the convergence rate since modified Newton's method only converges linearly, as given by equation 3.1.6. There exists some number of iterations p such that computation of a new Jacobian matrix will make the modified Newton's method more efficient. Unfortunately, the value of p depends on the degree of nonlinearity of the problem and cannot be estimated easily.

3.3. Quasi-Newton Methods

Quasi-Newton methods are a generalization of the one-dimensional secant method to the n-dimensional problem (e.g.,see [11] and [12]). In the secant method, an approximation to the tangent matrix, $\mathbf{A}_k$, is used at each iteration. This concept is applied in multi-dimensions and a simple updating is deduced to compute the new $\mathbf{A}_k$ from the previous value of $\mathbf{A}_{k-1}$. The staring matrix $\mathbf{A}_0$ is normally taken as $\mathbf{f}'(\mathbf{x}_0)$, however, other choices are possible such as the finite difference approximations [3]. The convergence rate for all practical quasi-Newton methods is super-linear, and the number of numerical operations for each iteration is $O(n^2)$.

To deduce the equation for the updating of the Jacobian, known as the secant equation, a linear backward Taylor formula at step k is used.

$$\mathbf{f}(\mathbf{x}_k) \approx \mathbf{f}(\mathbf{x}_{k-1}) - \mathbf{f}'(\mathbf{x}_k)(\mathbf{x}_k - \mathbf{x}_{k-1}) \tag{3.3.1}$$

This equation can be rearranged to obtain the approximate Jacobian $\mathbf{A}_k$

$$\mathbf{A}_k \mathbf{d}_{k-1} = \mathbf{b}_{k-1} \tag{3.3.2}$$

where

$$\mathbf{d}_{k-1} = \mathbf{x}_k - \mathbf{x}_{k-1} \tag{3.3.3}$$

and

$$\mathbf{b}_{k-1} = \mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}_{k-1}) \tag{3.3.4}$$

The values in 3.3.3 and 3.3.4 must be computed to perform the step

$$\mathbf{A}_k \mathbf{d}_k = -\mathbf{f}(\mathbf{x}_k) \tag{3.3.5}$$

Equation 3.3.2 is called the quasi-Newton equation, which must be satisfied by $\mathbf{A}_k$. In a one dimensional problem, this equation is sufficient to determine $\mathbf{A}_k$ completely, however, for multi-dimensional problems additional constraints are required to obtain $\mathbf{A}_k$. The construction of the approximate Jacobian depends, to a large extend, on the choices for these constraints. The work of Broyden and Broyden-Fletcher-Goldferb-Shano, BFGS, are but two of the many possibilities for constructing $\mathbf{A}_k$.

3.3.1. Broyden's Method

In 1965 Broyden proposed a method for the approximate specification of the tangent matrix by a simple update of the previous value. Broyden assumed that $\mathbf{A}_k$ does not differ from $\mathbf{A}_{k-1}$ when acting on any vector orthogonal to $\mathbf{d}_{k-1}$. accordingly

$$\mathbf{A}_k \mathbf{z} = \mathbf{A}_{k-1} \mathbf{z} \tag{3.3.6}$$

for

$$\mathbf{z}^T \mathbf{d}_{k-1} = 0$$

These equations provide an update relation for $\mathbf{A}_k$ and, together with equation 3.3.2, allows $\mathbf{A}_k$ to be determined uniquely. Thus, Broyden's method is given by

$$\mathbf{A}_k = \mathbf{A}_{k-1} + \frac{(\mathbf{b}_{k-1} - \mathbf{A}_{k-1}\mathbf{d}_{k-1})\mathbf{d}_{k-1}^T}{\|\mathbf{d}_{k-1}\|} \tag{3.3.7}$$

Post-multiplying 3.3.7 by $\mathbf{d}_{k-1}$ and $\mathbf{z}$ results in equations 3.3.6 and 3.3.2, respectively, thus demonstrating the applicability of 3.3.7 . This equation can be written in the computationally more attractive form, using equation 3.3.4

$$\mathbf{A}_k = \mathbf{A}_{k-1} + \frac{\mathbf{f}(\mathbf{x}_k)\mathbf{d}_{k-1}^T}{\|\mathbf{d}_{k-1}\|} \tag{3.3.8}$$

Broyden's method is super-linearly convergent [3]. Given an initial $\mathbf{x}_0$ and $\mathbf{A}_0$ and using the above equations the quasi-Newton step can be performed. The algorithm is identical to that for Newton's method, except the computation of the Jacobian is replaced by equation 3.3.8 . The factorization of $\mathbf{A}_k$ at each step is still required; consequently, it is best to directly update the inverse of $\mathbf{A}_{k-1}$ to obtain the inverse of $\mathbf{A}_k$. This will eliminate the need to compute the triangular decomposition of $\mathbf{A}_k$ and therefore will result in an algorithm with $O(n^2)$ operations in each step.

3.3.2. Computation of the Inverse Matrix

The inverse of matrices defined similar to 3.3.8 may be written as

$$(\mathbf{A} + \mathbf{v}\mathbf{w}^T)^{-1} = \mathbf{A}^{-1} - \frac{1}{\sigma}\mathbf{A}^{-1}\mathbf{v}\mathbf{w}^T\mathbf{A}^{-1} \tag{3.3.9}$$

where

$$\sigma = 1 + \mathbf{w}^T\mathbf{A}^{-1}\mathbf{v}$$

If we let $\mathbf{H}_k$ be the inverse of $\mathbf{A}_k$, Broyden's method for the update of the inverse becomes

$$\mathbf{H}_k = \mathbf{H}_{k-1} + \frac{(\mathbf{d}_{k-1} - \mathbf{H}_{k-1}\mathbf{b}_{k-1})\mathbf{d}_{k-1}^T\mathbf{H}_{k-1}}{\mathbf{d}_{k-1}^T\mathbf{H}_{k-1}\mathbf{b}_{k-1}} \tag{3.3.10}$$

provided $\mathbf{d}_{k-1}^T\mathbf{H}_{k-1}\mathbf{b}_{k-1}$ is nonzero. Broyden's method may be implemented as

$$\mathbf{d}_k = -\mathbf{H}_k\mathbf{f}(\mathbf{x}_k) \tag{3.3.11}$$

3.3.3. Convergence of Quasi-Newton Method

Convergence properties of quasi-Newton method are discussed by Dennis and More in [3]. In this study they restate an earlier result that an iterative method is super-linearly convergent if

$$\lim_{k\to\infty} \frac{\|[\mathbf{A}_k - \mathbf{f}'(\mathbf{x}^*)](\mathbf{x}_{k+1} - \mathbf{x}_k)\|}{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|} = 0 \qquad (3.3.12)$$

If $\mathbf{A}_k$ converges to $\mathbf{f}'(\mathbf{x}^*)$, as for Newton's method where $\mathbf{A}_k$ equals $\mathbf{f}'(\mathbf{x}_k)$, then convergence is always super-linear. However, the important result of 3.3.12 is that when $\mathbf{A}_k$ only converges to $\mathbf{f}'(\mathbf{x}^*)$ in the direction $\mathbf{d}_k$, convergence is also super-linear. Both the Broyden and the BFGS quasi-Newton method satisfy 3.3.12 for continuously differentiable $\mathbf{f}$ and thus are super-linearly convergent.

3.3.4. Methods for Symmetric Positive Definite Jacobians

Brodlie, Gourlay and Greenstadt have shown that certain rank-one and rank-two corrections to symmetric positive definite matrices may be conveniently expressed in the product form [1]

$$\mathbf{H}_k = (\mathbf{I} + \mathbf{w}_k\mathbf{v}_k^T)\mathbf{H}_{k-1}(\mathbf{I} + \mathbf{v}_k\mathbf{w}_k^T) \qquad (3.3.13)$$

This form has the important property that successive inverse tangent matrices remain symmetric. In [10] the algorithm is related to the BFGS algorithm, which gives

$$\mathbf{w}_k = \frac{1}{\mathbf{d}_{k-1}^T\mathbf{b}_k}\,\mathbf{d}_{k-1} \qquad (3.3.14)$$

and

$$\mathbf{v}_k = \mathbf{f}(\mathbf{x}_{k-1})\left[1 - \left(\frac{\mathbf{d}_{k-1}^T\mathbf{b}_{k-1}}{\mathbf{d}_{k-1}^T\mathbf{f}(\mathbf{y}_k)}\right)^{1/2}\right] - \mathbf{f}(\mathbf{x}_k) \qquad (3.3.15)$$

The computational steps for implementing the BFGS method are very straight- forward and consist of solving 3.3.11 with following steps:

(1) for each k compute $\mathbf{f}(\mathbf{x}_k)$ and $\mathbf{z}_k = -\,(\mathbf{I} + \mathbf{v}_k\mathbf{w}_k^T)\mathbf{f}(\mathbf{x}_k)$

(2) solve $\mathbf{H}_{k-1}^{-1}\mathbf{u}_k = \mathbf{z}_k$

(3) compute $\mathbf{d}_k = (\mathbf{I} + \mathbf{w}_k\mathbf{v}_k^T)\mathbf{u}_k$

(4) if required, compute a line search for s_k

(5) update $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k \mathbf{d}_k$

(6) check convergence

It should be noted that $\mathbf{H}_{k-1}$ may result from a previous BFGS update. Thus, step (1) and (3) may require several updates before the resolve step is performed. Furthermore, this form of the algorithm is strictly limited to positive definite tangent matrices. Indefinite matrices resulting from Lagrange multiplier constraint, such as contact problems, can not be considered by present implementation of BFGS and therefore an alternative implementation is required.

4. LANCZOS ALGORITHM FOR SOLUTION OF THE LINEARIZED PROBLEM

The discretization of nonlinear structural mechanics problems and linearization of the resulting nonlinear algebraic equations for application of a Newton type methods normally will lead to a symmetric system of linear equations. These equations may be written as

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.0.1}$$

Presently, most nonlinear procedures employ a direct method such as Gaussian elimination for the solution of this system of equations. In this section an alternative solution technique is presented.

In most applications A will be sparse, and an elegant way to make use of sparsity is to employ A solely as a linear operator which computes the product $\mathbf{Av}$ for a given vector $\mathbf{v}$. This has the added advantage that A need not be known explicitly but only a means of computing the matrix vector product is required.

For years, iterative methods have been used for the solution of large systems of equations. The Conjugate Gradient method is one such technique introduced by Hestenes and Stiefel [5]. This method, in theory, requires at most n steps to obtain the exact solution of 4.0.1 and was accepted because of this termination property. In the same year Lanczos published his method of Minimized Iteration which was initially introduced for computing the eigen pairs of a large symmetric matrix. Lanczos and Householder [6] pointed out the intimate connection between the two approaches.

These methods have several attractive features in common. There is no need for A to have further special properties, such as banded form, no acceleration parameters have to be estimated, and the storage requirements are only a few n-vectors in addition to the storage needs of A.

In finite element applications the matrix vector multiplication can be performed at the element level. This will result in considerable saving in storage at the expense of some additional multiplications.

4.1. The Lanczos Algorithm

For certain applications of the finite element method, especially in nonlinear problems, it is usual to have on hand an initial approximation $\mathbf{x}^a$ to the true solution of 4.0.1 . The problem is now to find a correction $\mathbf{x}^c$ to be added to $\mathbf{x}^a$. Then

$$\mathbf{A}\mathbf{x}^c = \mathbf{r}_0 \tag{4.1.1}$$

where

$$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}^a \tag{4.1.2}$$

Lanczos algorithm may be described very simply as a process of constructing the weak form of equation 4.0.1 from a very special subspace. The subspace under consideration is generated from the set of j vectors ($\mathbf{r}_0$, $\mathbf{A}\mathbf{r}_0$, $\cdots$, $\mathbf{A}^{j-1}\mathbf{r}_0$), known to mathematicians as the Krylov subspace [18]. To construct the weak form it would be simpler if an orthonormal set of vectors, say ($\mathbf{q}_1$,$\mathbf{q}_2$, $\cdots$,$\mathbf{q}_j$), were available. This can be achieved by applying Gram-Schmidt orthogonalization to the Krylov vectors. Initially, this appears to be an expensive way of obtaining an orthonormal base vectors, however this process can be simplified when the following two facts are used:

(i) The use of $\mathbf{A}\mathbf{q}_j$ and $\mathbf{A}^j\mathbf{r}_0$, for orthogonalization against the previous $\mathbf{q}$ vectors and normalization of the resulting vector, leads to the same vector $\mathbf{q}_{j+1}$.

(ii) The vector $\mathbf{A}\mathbf{q}_j$ is orthogonal to $\mathbf{q}_1$,$\mathbf{q}_2$, $\cdots$,$\mathbf{q}_{j-2}$.

These results are demonstrated in Appendix A. A more complete explanation is presented in [18]. Therefor it is sufficient to orthogonalize $\mathbf{A}\mathbf{q}_j$ against $\mathbf{q}_{j-1}$ and $\mathbf{q}_j$ to obtain the next orthogonal vector.

$$\mathbf{r}_j \equiv \beta_{j+1}\mathbf{q}_{j+1} = \mathbf{A}\mathbf{q}_j - \alpha_j\mathbf{q}_j - \beta_j\mathbf{q}_{j-1} \tag{4.1.3}$$

where $\alpha_j = \mathbf{q}_j^T \mathbf{A} \mathbf{q}_j$ and $\beta_j = \mathbf{q}_{j-1}^T \mathbf{A} \mathbf{q}_j$. It is important to note that the vectors $(\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_{j-2})$ are not needed in equation 4.1.3 . This defines one step of the Lanczos algorithm. The normalization of $\mathbf{r}_j$ results in $\mathbf{q}_{j+1}$. It is easy to show, by looking at $\mathbf{q}_{j+1}^T \mathbf{r}_j$, that $\beta_{j+1} = \|\mathbf{r}_j\|$.

The special choice of the base vectors for the subspace has an additional advantage. The projection of A onto this subspace is a tridiagonal matrix, $\mathbf{T}_j$.

$$\mathbf{T}_j = \mathbf{Q}_j^T \mathbf{A} \mathbf{Q}_j = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \cdot & & \\ & & & \cdot & \\ & & & \alpha_{j-1} & \beta_j \\ & & & \beta_j & \alpha_j \end{bmatrix} \tag{4.1.4}$$

where the $\mathbf{q}$ vectors form the columns of the matrix $\mathbf{Q}_j$, $\mathbf{Q}_j \equiv (\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_j)$.
This fact was realized soon after Lanczos introduced his method and the algorithm was put to use as a process for the orthogonal transformation of a matrix to tridiagonal form. Despite its additional attractions, the Lanczos process gave way to Givens' method in 1954 and later to Householder's method in 1958.

The relationships that define the Lanczos algorithm can now be summarized in the following three equations.

$$\mathbf{Q}_j^T \mathbf{Q}_j = \mathbf{I}_j \tag{4.1.5.a}$$

$$\mathbf{A}\mathbf{Q}_j - \mathbf{Q}_j \mathbf{T}_j = \mathbf{r}_j \mathbf{e}_j^T \tag{4.1.5.b}$$

$$\mathbf{Q}_j^T \mathbf{r}_j = 0 \tag{4.1.5.c}$$

where $\mathbf{e}_j$ is the j-th column of the $j \times j$ identity matrix $\mathbf{I}_j$.
Setting $\mathbf{q}_0 = 0$ and using $\mathbf{r}_0$ as the starting vector, the Lanczos algorithm can then be described as

Given $\mathbf{r}_0$, set $\beta_1 = \|\mathbf{r}_0\|$

```
For $j = 1,2, \cdots$ repeat
    (1) $\mathbf{q}_j \leftarrow \frac{\mathbf{r}_{j-1}}{\beta_j}$
    (2) $\mathbf{u}_j \leftarrow \mathbf{A}\mathbf{q}_j$
    (3) $\mathbf{r}_j \leftarrow \mathbf{u}_j - \beta_j \mathbf{q}_{j-1}$
    (4) $\alpha_j \leftarrow \mathbf{q}_j^T \mathbf{r}_j$
    (5) $\mathbf{r}_j \leftarrow \mathbf{r}_j - \alpha_j \mathbf{q}_j$
    (6) $\beta_{j+1} \leftarrow \|\mathbf{r}_j\|$
```

At the end of a simple Lanczos step the newly computed Lanczos vector is written into secondary storage.

4.2. The Weak Form

A weak form of equation 4.0.1 can now be constructed using equation 4.1.5 . The approximation to the solution from the subspace can be written as

$$\mathbf{x}_j = \mathbf{Q}_j \mathbf{f}_j \tag{4.2.1}$$

where $\mathbf{f}_j$ is the vector in the subspace defining the approximate solution. The i-th coefficient ϕ_i of $\mathbf{f}_j$ and its corresponding $\mathbf{q}_i$ are much like the nodal parameter and its associated shape function in the finite element discretization.

Equations 4.1.1 together with equation 4.2.1 results in

$$\mathbf{A}\mathbf{Q}_j \mathbf{f}_j = \mathbf{r}_0 \tag{4.2.2}$$

This is an overdetermined system of equation. However a solution can be obtained using a procedure similar to the method of weighted residuals, and an approximation can be obtained by premultiplying this with $\mathbf{W}_j^T$. The weighting vectors W_j can be chosen in such manners as to reduce the residual of the quantities of interest.

$$\mathbf{W}_j^T\mathbf{A}\mathbf{Q}_j\mathbf{f}_j = \mathbf{W}_j\mathbf{r}_0$$

A Galarkin type method results when the Lanczos vectors are chosen as the weighting vectors.

$$\mathbf{Q}_j^T\mathbf{A}\mathbf{Q}_j\mathbf{f}_j = \mathbf{Q}_j^T\mathbf{r}_0$$

$$\mathbf{T}_j\mathbf{f}_j = \beta_1\mathbf{e}_1^{(j)} \tag{4.2.3}$$

Here $\mathbf{e}_1^{(j)}$ is the first column of the identity matrix and the above is derived using the fact that $\mathbf{r}_0 = \mathbf{q}_1\beta_1$. The above equations possess the orthogonal characteristics of the residual vector peculiar to Galarkin methods, that is the residual vector is kept orthogonal to the trial vectors and therefore is "minimized". In appendix B equation 4.2.3 is also derived from an energy consideration. This orthogonality property can be seen in the form of a monotonic reduction in the potential energy of the system which is a characteristic of both the Lanczos method and conjugate gradient algorithm. In Figure 4.1 this is shown in terms of a simple beam problem.

4.3. Termination Criteria

In the previous section the Lanczos algorithm was viewed as a method for constructing the weak form of the linear set of equations. The form of the algorithm suggests that after every Lanczos step the number of $\mathbf{q}$ vectors increase by one. In exact arithmetic the algorithm can be terminated after n steps and the tridiagonal matrix contains all the information needed to solve the problem exactly. However, it is often sufficient to stop the iteration at some step $m < n$, when the solution is close enough to the exact solution. Using the weak form, equation 4.2.3, the solution vector

$$\mathbf{f}_m = [\phi_1, \phi_2, \cdots, \phi_m]^T \tag{4.3.1}$$

can be obtained by solving the equations. The approximate solution $\mathbf{x}_m$ can then be constructed directly from equation 4.2.1 . The residual vector is

$$\begin{aligned}\hat{\mathbf{r}}_m &\equiv \mathbf{A}\mathbf{x}_m - \mathbf{r}_0 \\ &= \mathbf{A}\mathbf{Q}_m\mathbf{f}_m - \mathbf{r}_0 \end{aligned} \tag{4.3.2}$$

Setting $j = 1$ in equation 4.1.3 the following can be derived

$$\begin{aligned}\mathbf{r}_0 &= \beta_1\mathbf{q}_1 \\ &= \mathbf{Q}_m\ (\mathbf{e}_1^{(m)}\beta^1) \end{aligned} \tag{4.3.3}$$

This together with 4.3.2 and 4.2.3 results in

$$\begin{aligned}\hat{\mathbf{r}}_m &= \mathbf{A}\mathbf{Q}_m\mathbf{f}_m - \mathbf{Q}_m\mathbf{T}_m\mathbf{f}_m \\ &= (\mathbf{A}\mathbf{Q}_m - \mathbf{Q}_m\mathbf{T}_m)\mathbf{f}_m \\ &= (\mathbf{r}_m\mathbf{e}_m^T)\ \mathbf{f}_m \\ &= \mathbf{r}_m\phi_m \end{aligned} \tag{4.3.4}$$

Therefore the residual vector is a scalar multiple of the $\mathbf{r}_m$ vector computed at the m-th Lanczos step. Moreover the residual norm ρ_m is given by

$$\begin{aligned}\rho_m &\equiv \|\hat{\mathbf{r}}_m\| \\ &= |\phi_m|\ \|\mathbf{r}_m\| \\ &= \beta_{m+1}|\phi_m| \end{aligned} \tag{4.3.5}$$

Thus by computing the bottom element of the solution vector to the tridiagonal system the residual norm can be computed using a simple multiplication, noting that β_{m+1} is computed at the m-th Lanczos step. In fact it is not even necessary to compute the solution to the tridiagonal system, $\mathbf{f}_m$ to obtain ϕ_m. A recurrence for updating ϕ_j from information at the previous step can be derived by considering a QR factorization of $\mathbf{T}_m$. Consider first a factorized form of the tridiagonal matrix at step $j - 1$

$$\mathbf{T}_{j-1} = \mathbf{R}_{j-1}\mathbf{U}_{j-1} \tag{4.3.6}$$

where $\mathbf{R}_{j-1}$ is the product of all the rotation matrices up to and including step $j-1$ that transforms $\mathbf{T}_{j-1}$ to the upper triangular matrix $\mathbf{U}_{j-1}$. Then the system of equations 4.2.3, for step $j-1$, can now be written as

$$\mathbf{U}_{j-1}\mathbf{f}_{j-1} = \beta_1 \mathbf{R}_{j-1}^T \mathbf{e}_1^{(j-1)} \tag{4.3.7}$$

The last equation of the above can be written as

$$\tilde{\alpha}_{j-1}\phi_{j-1} = \gamma_{j-1} \tag{4.3.8}$$

Here tilde denotes all quantities that were modified by the all the previous rotations and γ_{j-1} is the bottom element that is created by these rotations. After an additional Lanczos step the last two equations of the new system is

$$\begin{bmatrix} \tilde{\alpha}_{j-1} & \tilde{\beta}_j \\ \beta_j & \alpha_j \end{bmatrix} \begin{Bmatrix} \phi_{j-1} \\ \phi_j \end{Bmatrix} = \begin{Bmatrix} \gamma_{j-1} \\ 0 \end{Bmatrix} \tag{4.3.9}$$

these equations are transformed again into an upper triangular form by premultiplication of equation 4.3.9 by the plane rotation matrix

$$\begin{bmatrix} c_j & -s_j \\ s_j & c_j \end{bmatrix}$$

where $c_j = \cos\psi_j$ and $s_j = \sin\psi_j$. Here ψ_j is the rotation angle. The result of this matrix multiplication is

$$\begin{bmatrix} c_j\tilde{\alpha}_{j-1} - s_j\beta_j & c_j\tilde{\beta}_j - s_j\alpha_j \\ s_j\tilde{\alpha}_{j-1} + c_j\beta_j & s_j\tilde{\beta}_j + c_j\alpha_j \end{bmatrix} \begin{Bmatrix} \phi_{j-1} \\ \phi_j \end{Bmatrix} = \begin{Bmatrix} c_j\gamma_{j-1} \\ s_j\gamma_{j-1} \end{Bmatrix} \tag{4.3.10}$$

The value of the rotation angle is determined by setting the term in the lower triangular part equal to zero. Thus

$$\tan\psi_j = \frac{\beta_j}{\tilde{\alpha}_{j-1}} \tag{4.3.11}$$

and finally ϕ_j can be obtained by performing the first step of the backsubstitution process.

$$\phi_j = \frac{s_j \gamma_{j-1}}{(s_j \tilde{\beta}_j + c_j \alpha_j)} \tag{4.3.12}$$

It should be pointed out that the rotation at this step effects the next off-diagonal term in the tridiagonal matrix and the effect is only on the super-diagonal term which explains the $\tilde{\beta}_j$ term in equation 4.3.9 . The final form of the updating algorithm can be arranged as follows

$$\tan\psi_j = \frac{\beta_j}{\tilde{\alpha}_{j-1}}$$

$$\tilde{\alpha}_j = \sin\psi_j \tilde{\beta}_j + \cos\psi_j \alpha_j$$

$$\tilde{\beta}_{j+1} = \cos\psi_j \beta_{j+1}$$

$$\tilde{\gamma}_j = \sin\psi_j \tilde{\gamma}_{j-1}$$

$$\phi_j = \frac{\tilde{\gamma}_j}{\tilde{\alpha}_j}$$

This is a very inexpensive step and therefore an attractive form of updating the bottom element of the solution vector. The residual norm, ρ_j, can then be computed from equation 4.3.5 .

Contrary to the Conjugate Gradient method it is not necessary to update an approximate solution vector at each step, but only the residual norm has to be monitored. Once ρ_j falls bellow the specified tolerance the Lanczos algorithm is terminated and the small m degree tridiagonal system is solved. It is very important to note that if $\mathbf{A}$ is indefinite then $\mathbf{T}_j$ may also be indefinite and care must be taken in solving the small system. For this reason a stable equation solver such as the one used in the updating procedure for the residual norm should be used.

Finally the approximate solution $\mathbf{x}_m$ has to be computed. It is here that the Lanczos vectors are needed again to construct $\mathbf{x}_m$. In the present implementation the Lanczos vectors are put onto secondary storage as they are computed and at the end of the algorithm they are brought back one by one to form $\mathbf{x}_m$.

4.4. Derivation of Conjugate Gradients Method from Lanczos Algorithm

The conjugate gradients method may be developed directly from the Lanczos process. This will demonstrate the limitation of this method. The aim is to obtain a procedure for updating of the solution vector $\mathbf{x}_j$ without retaining all the Lanczos vectors.

Consider first the case where A is positive definite, then the resulting tridiagonal matrix will also be positive definite and hence its cholesky factorization exists.

$$\mathbf{T}_j = \mathbf{L}_j\mathbf{L}_j^T \tag{4.4.1}$$

Here $\mathbf{L}_j$ is a lower bidiagonal matrix with positive diagonal terms.

$$\mathbf{L}_j = \begin{bmatrix} \delta_1 & & & & \\ \mu_2 & \delta_2 & & & \\ & \mu_3 & \ddots & & \\ & & \ddots & \delta_{j-1} & \\ & & & \mu_j & \delta_j \end{bmatrix} \tag{4.4.2}$$

The main obstacle to updating $\mathbf{x}_j$ is the fact that the solution vector $\mathbf{f}_j$ to equation 4.2.3 changes fully after each Lanczos step and therefor $\mathbf{Q}_j\mathbf{f}_j$ can not be simply accumulated recursively. This difficulty can be overcome by defining

$$\mathbf{g}_j = \mathbf{L}_j^T\mathbf{f}_j \tag{4.4.3}$$

and

$$\mathbf{P}_j = \mathbf{Q}_j\mathbf{L}_j^{-T} \tag{4.4.4}$$

The vectors in $\mathbf{P}_j$ are the conjugate vectors. Then the approximation $\mathbf{x}_j$ becomes

$$\mathbf{x}_j = \mathbf{P}_j\mathbf{g}_j \tag{4.4.6}$$

Further, the weak form equation now becomes

$$\mathbf{L}_j \mathbf{g}_j = \mathbf{e}_1^{(j)} \beta_1 \tag{4.4.7}$$

It is apparent from the bidiagonal structure of $\mathbf{L}_j$, equation 4.4.2, the g_j can be updated very easily since the solution of equation 4.4.7 is obtained by a simple forward reduction. Therefor $\mathbf{g}_{j+1}$ can be written in terms of $\mathbf{g}_j$ as

$$\mathbf{g}_{j+1} = \begin{pmatrix} \mathbf{g}_j \\ \eta_{j+1} \end{pmatrix} \tag{4.4.8}$$

where

$$\eta_{j+1} = -\mu_{j+1} \eta_j / \delta_{j+1} \tag{4.4.9}$$

Thus the approximation $\mathbf{x}_{j+1}$ can obtained from $\mathbf{x}_j$ using equation 4.4.6 in arranged in the following form

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \eta_{j+1} \mathbf{p}_{j+1} \tag{4.4.10}$$

and finally the $\mathbf{p}$-vectors can be obtained from their definition. Thus

$$\mathbf{p}_{j+1} = \frac{1}{\delta_{j+1}} (\mathbf{q}_{j+1} - \mu_{j+1} \mathbf{p}_j) \tag{4.4.11}$$

This is equivalent to the method of conjugate gradients. The approach here is computationally a little different, but the role of the cholesky decomposition becomes apparent, with the necessary requirement that A be positive definite to ensure numerical stability. If A is an indefinite symmetric matrix, the algorithm may still be carried out, with some success, but due to instability it can no longer be numerically reliable.

4.5. Selective Orthogonalization

A detailed account of the behaviour of the Lanczos algorithm in the presence of roundoff and of selective orthogonalization is available in Parlett [18]. Therefore in this section only the basic facts about selective orthogonalization will be presented.

Let ϵ be the smallest number in the computer such that $1 + \epsilon > 1$. This is known as the unit roundoff error. The basic equations 4.1.5.a and 4.1.5.c are now perturbed by roundoff. Although for each j 4.1.5.b is only slightly perturbed, the relations 4.1.5.a and 4.1.5.c completely fail after a certain number of steps depending on ϵ and on A. The Lanczos vectors, which are orthogonal in exact arithmetic, not only loose their orthogonality, but even become linearly dependent.

For a long time, as a remedy against this loss of orthogonality, it was suggested to reorthogonalize each new q_j against all previous Lanczos vectors, which is of course very expensive. The results of Paige [15] gave some better insight in the way orthogonality is lost, and provide the theoretical basis for selective orthogonalization.

In order to explain Paige's results it is necessary to introduce certain quantities which are not of direct interest when solving a linear system of equations. Let $\theta_i^{(j)}$ be the eigenvalues and $\mathbf{s}_i^{(j)}$ be the corresponding normalized eigenvectors of $\mathbf{T}_j$

$$\mathbf{T}_j\mathbf{s}_i^{(j)} = \mathbf{s}_i^{(j)}\theta_i^{(j)} \qquad i = 1,\ldots,j \ . \tag{4.5.1}$$

From these one can compute the Ritz vectors $\mathbf{y}_i^{(j)}$ by

$$\mathbf{y}_i^{(j)} = \mathbf{Q}_j\mathbf{s}_i^{(j)} \tag{4.5.2}$$

These quantities change at each Lanczos step. In subsequent discussions where there is no confusion possible the superscripts will be dropped. The pairs $(\theta_i^{(j)},\mathbf{y}_i^{(j)})$ $i=1,\ldots,j$ are approximate eigenpairs of A. The quality of this approximation can be determined by considering the residual $\|\mathbf{A}\mathbf{y}_i - \mathbf{y}_i\theta_i\|$. Applying (4.5.2) one finds that, to within roundoff,

$$\|\mathbf{A}\mathbf{y}_i - \mathbf{y}_i\theta_i\| \leqslant \beta_{ji} + \|\mathbf{E}_j\| \tag{4.5.3}$$

where $\mathbf{E}_j$ is an error matrix and $\beta_{ji} \equiv \beta_{j+1}|s_{ji}|$ and $s_{ji} \equiv \mathbf{e}_j^T\mathbf{s}_i$, i.e., the bottom element of the corresponding eigenvector of $\mathbf{T}_j$. The β_{ji} play an important role in the process of selective orthogonalization.

Now everything is ready to state one of the most important consequences of Paige's work, which can be summarized as follows:

Loss of orthogonality among the Lanczos vectors is equivalent to the convergence of a Ritz pair.

In other words, if one of the β_{ji} becomes small, the corresponding Ritz pair converges to an eigenpair of A and the Lanczos vector $\mathbf{q}_{j+1}$ loses its orthogonality to $\mathbf{q}_1, \cdots, \mathbf{q}_{j-2}$. But more is known about the way $\mathbf{q}_{j+1}$ behaves, it is tilted towards $\mathbf{y}_i^{(j)}$ while it is retaining its previous level of orthogonality to all the other Ritz vectors, $\mathbf{y}_k^{(j)}, k \neq i$. In order to maintain a certain level of orthogonality it is therefore only necessary to orthogonalize the new $\mathbf{q}_{j+1}$, or equivalently the unnormalized $\mathbf{r}_j$ against $\mathbf{y}_i^{(j)}$ when the β_{ji} become smaller than some threshold. So first compute $\mathbf{r}'_j$ (compare 4.1.3) by

$$\mathbf{r}'_j \equiv A\mathbf{q}_j - \mathbf{q}_j\alpha_j - \mathbf{q}_{j-1}\beta_j \tag{4.5.4}$$

as usual, then check if any β_{ji} is small. If so then compute the corresponding $\mathbf{y}_i^{(j)}$ and orthogonalize $\mathbf{r}'_j$ against it obtaining

$$\mathbf{r}_j \equiv \mathbf{r}'_j - \mathbf{y}_i^{(j)}\xi_i^{(j)} \tag{4.5.5}$$

where

$$\xi_i^{(j)} \equiv \frac{\mathbf{y}_i^{(j)T}\mathbf{r}'_j}{\|y_i^{(j)}\|^2} \tag{4.5.6}$$

This requires the formation of $\mathbf{y}_i = \mathbf{Q}_j\mathbf{s}_i$. The payoff for the expensive calculation comes later when subsequent Lanczos vectors, say $\mathbf{q}_{j+15}$ and $\mathbf{q}_{j+30}$ need to be orthogonalized against $\mathbf{y}_i$, since a certain number p of Ritz vectors are also put in secondary storage and need not be formed again. The question of how many Ritz vectors to keep, and whether to keep them in core in case the optimal number is small is still under investigation.

Finally it should be mentioned that the threshold for which a β_{ji} is considered to be small is set to be $\sqrt{\epsilon}\|\mathbf{T}_j\|$ in the present implementation. This choice was based on the computational experience for the eigenvalue problem [20]. An analysis in [20] shows that this guarantees a certain level of orthogonality among the Lanczos vectors.

To demonstrate the loss of orthogonality, in Figure 4.2, the variation in the residual norm is plotted for the simple Lanczos process when no action is taken to maintain orthogonality and for the case when selective orthogonalization is performed to maintain a level of orthogonality among the Lanczos vectors. The conjugate gradient method and other methods based on the Krylov vectors also suffer this effect. It should be noted that the loss of orthogonality does not stop convergence, it only delays it.

4.6. Lanczos with Selective Orthogonalization

The final form of the Lanczos algorithm which maintains a certain level of orthogonality is summarized in the following table.

```
1.    Loop: for $j = 1,2, \cdots$
      1.1  Take a simple Lanczos step
      1.2  Update residual norm $\rho_j$
      1.3  Check: if $\rho_i \leqslant$ tolerance end the loop
      1.4  Update the eigenvalues of $\mathbf{T}_j$
      1.5  Compute $\beta_{ji} = \beta_{j+1} s_{ji}$
      1.6  Check: if $\beta_{ji} \leqslant \sqrt{\epsilon}\|\mathbf{T}_j\|$
               then compute the $i$-th eigen vector and orthogonalize
2.    Solve $\mathbf{T}_j \mathbf{f}_j = \mathbf{e}_1 \beta_1$
3.    Assemble solution: $\mathbf{x}_j = \mathbf{Q}_j \mathbf{f}_j$
```

The above formulation requires a total of 5 n-vectors and 10 j-vectors workspace. This includes two n-vectors containing the right hand side and the initial guess, two n-vectors for $\mathbf{r}_j$ and $\mathbf{q}_j$ and a n-vector for use in construction of Ritz vectors.

4.7. Selective Reorthogonalization

The procedure described in the previous section is not the only way of maintaining orthogonality among the Lanczos vectors. Recently H. Simon [22] has proposed an alternative technique based on a simple recursion formula. This recursion is derived as follows. Define

$$\omega_{i,j} = \mathbf{q}_i^T \mathbf{q}_j \qquad (4.7.1)$$

and multiplying equation 4.1.3 by $\mathbf{q}_i^T$ and using the above definition we get

$$\mathbf{q}_i^T \mathbf{A} \mathbf{q}_j = \alpha_j \omega_{i,j} + \beta_j \omega_{i,j-1} + \beta_{j+1} \omega_{i,j+1} \qquad (4.7.1.a)$$

a similar equation can be obtained when the same steps are applied to the i-th Lanczos step.

$$\mathbf{q}_j^T \mathbf{A} \mathbf{q}_i = \alpha_i \omega_{j,i} + \beta_i \omega_{j,i-1} + \beta_{i+1} \omega_{j,i+1} \qquad (4.7.1.b)$$

subtracting the above two relations to eliminate the terms involving A results in the recursion

$$\beta_{j+1} \omega_{j+1,i} = \beta_{i+1} \omega_{j,i+1} + (\alpha_i - \alpha_j) \omega_{j,i} + \beta_i \omega_{j,i-1} - \beta_j \omega_{j,i-1} \qquad (4.7.3)$$

This recursion holds for $j = 2,3, \cdots$ and $1 \leqslant i \leqslant j-1$. To start the three step recursion it is necessary to assume that

$$\omega_{j,j} = 1 \qquad j = 1,2, \cdots \qquad (4.7.4.a)$$

and

$$\omega_{j,j-1} = \epsilon \qquad j = 2,3, \cdots \qquad (4.7.4.b)$$

Using the above recursion the loss of orthogonality can be monitored during the Lanczos process simply by looking at the magnitude of $\omega_{j,i}$. Whenever $|\omega_{j,i}|$ is larger than the threshold value, $\sqrt{\epsilon}\,\|\mathbf{T}_j\|$, orthogonality is considered to be lost against $\mathbf{q}_i$ and reorthogonalization must be performed. Lanczos algorithm with selective reorthogonalization then becomes:

```
1.  Loop: for $j = 1,2, \cdots$
    1.1  Take a simple Lanczos step
    1.2  Update residual norm $\rho_j$
    1.3  Check: if $\rho_i \leqslant$ tolerance end the loop
    1.4  Update $\omega_{i,j}$
    1.5  Check: if $\omega_{i,j} \geqslant \sqrt{\epsilon}\|\mathbf{T}_j\|$
             then orthogonalize $\mathbf{q}_{j+1}$ against previous Lanczos vectors.
2.  Solve $\mathbf{T}_j\mathbf{f}_j = \mathbf{e}_1\beta_1$
3.  Assemble solution: $\mathbf{x}_j = \mathbf{Q}_j\mathbf{f}_j$
```

This algorithm has a number of desirable advantages over the previous approach.

(i) The implementation of this recursion is very easy and the resulting code is considerably simpler.

(ii) This approach requires less storage space.

(iii) There is no need for computing the eigenvalues and eigenvectors of the tridiagonal system.

Initial experiments with the two approaches indicate that both methods flag the loss of orthogonality at the same step. Therefore due to the above advantages the selective reorthogonalization technique is a better way of maintaining orthogonality for purpose of equation solving.

4.8. Starting Vector

The choice of a good starting vector can reduce the number of Lanczos steps the algorithm takes to converge to the solution. In fact if the true solution is on hand the Lanczos algorithm terminates after one step. However it is not often that one has the exact solution or even a good approximation. Nevertheless the question of starting vectors still remains. Consider the extreme case where the right hand side is an eigenvector. Then any starting vector other than zero or the right hand side will take more than one step to converge, whereas, a zero starting vector will take exactly one step to obtain the true solution. This can be seen by performing the algorithm on the equation $\mathbf{Ax} = \mathbf{z}_i$, where $\mathbf{z}_i$ is the i-th eigenvector with corresponding eigenvalue λ_i. Then

Setting $\mathbf{q}_0 = 0$

$\mathbf{r}_0 = \mathbf{z}_i$ and $\beta_1 = 1$

(1) $\mathbf{q}_1 \leftarrow \mathbf{r}_0/\beta_1 = \mathbf{z}_i$

(2) $\mathbf{u}_1 \leftarrow \mathbf{A}\mathbf{q}_1 = \lambda_1 \mathbf{z}_1$

(3) $\mathbf{r}_1 \leftarrow \mathbf{u}_1 - \beta_1 \mathbf{q}_0 = \lambda_1 \mathbf{z}_1$

(4) $\alpha_1 \leftarrow \mathbf{q}_1^T \mathbf{r}_1 = \lambda_1$

(5) $\mathbf{r}_1 \leftarrow \mathbf{r}_1 - \alpha_1 \mathbf{q}_1 = 0$

(6) $\beta_2 \leftarrow 0$

and the residual norm will be zero.

In general the right hand side has components along a certain number of eigenvectors and introduction of any starting vector that has components along any additional eigenvectors will delay convergence. Therefore, it can be said that unless special knowledge about the solution is available, the best starting vector is the zero vector. The influence of a random staring vector on the residual norm during a Lanczos run can be seen in Figure 4.3 for the simple beam problem.

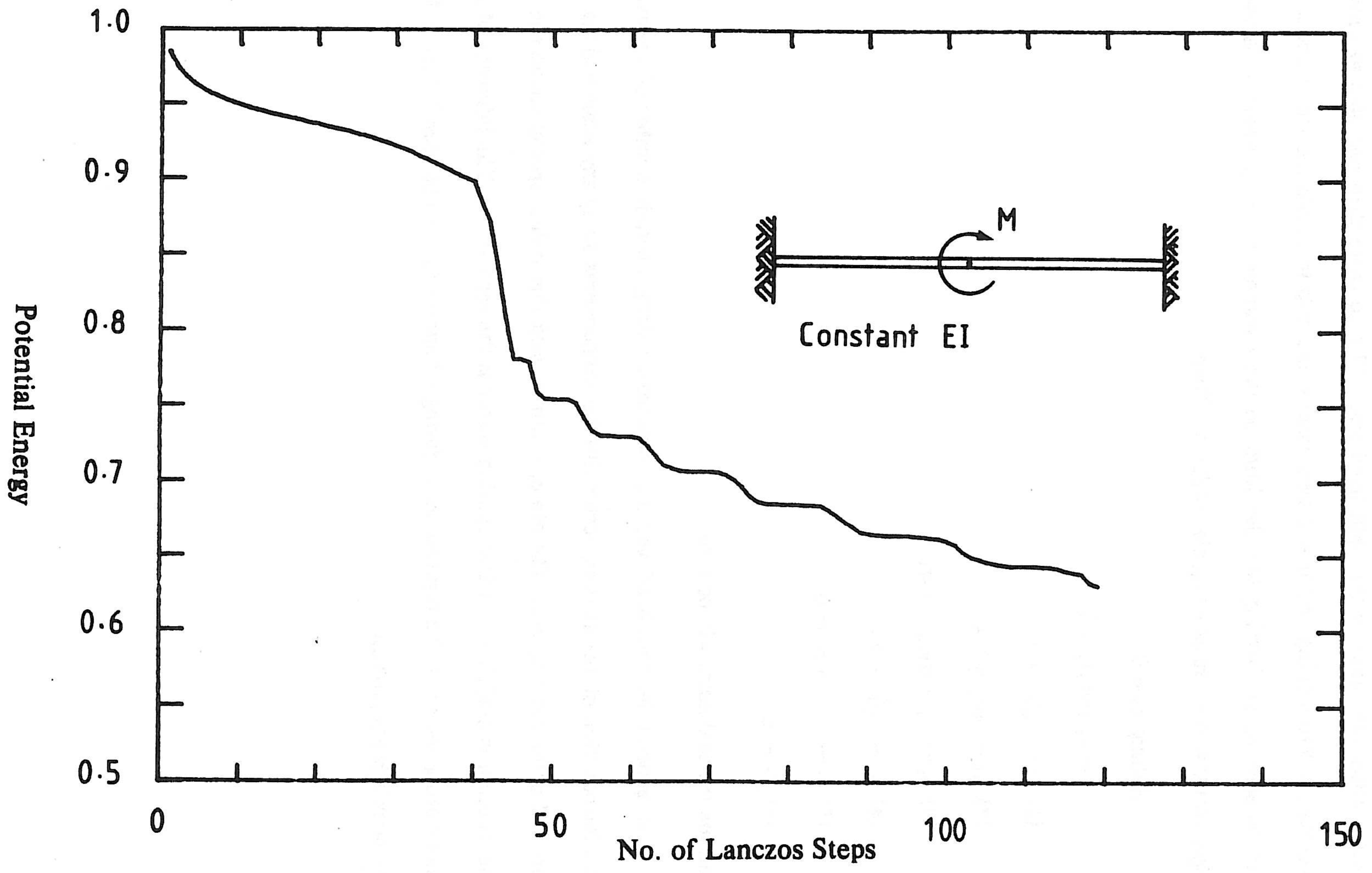


Figure 4.1 Monotonic Reduction of the Potential Energy of the System in the Lanczos Algorithm.

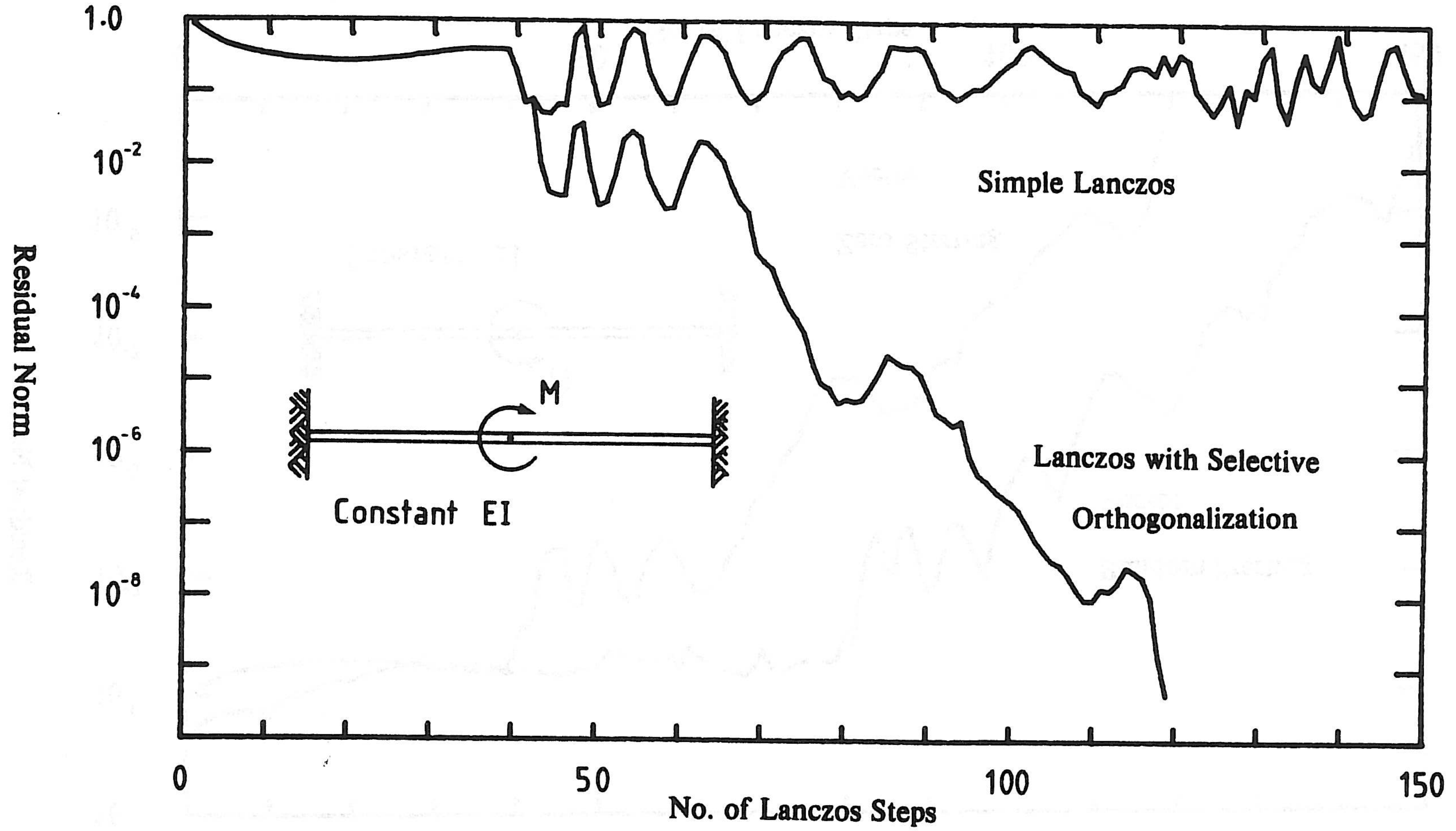


Figure 4.2 Effect of Selective Orthogonalization on the Reduction of the Residual Norm in the Lanczos Algorithm.

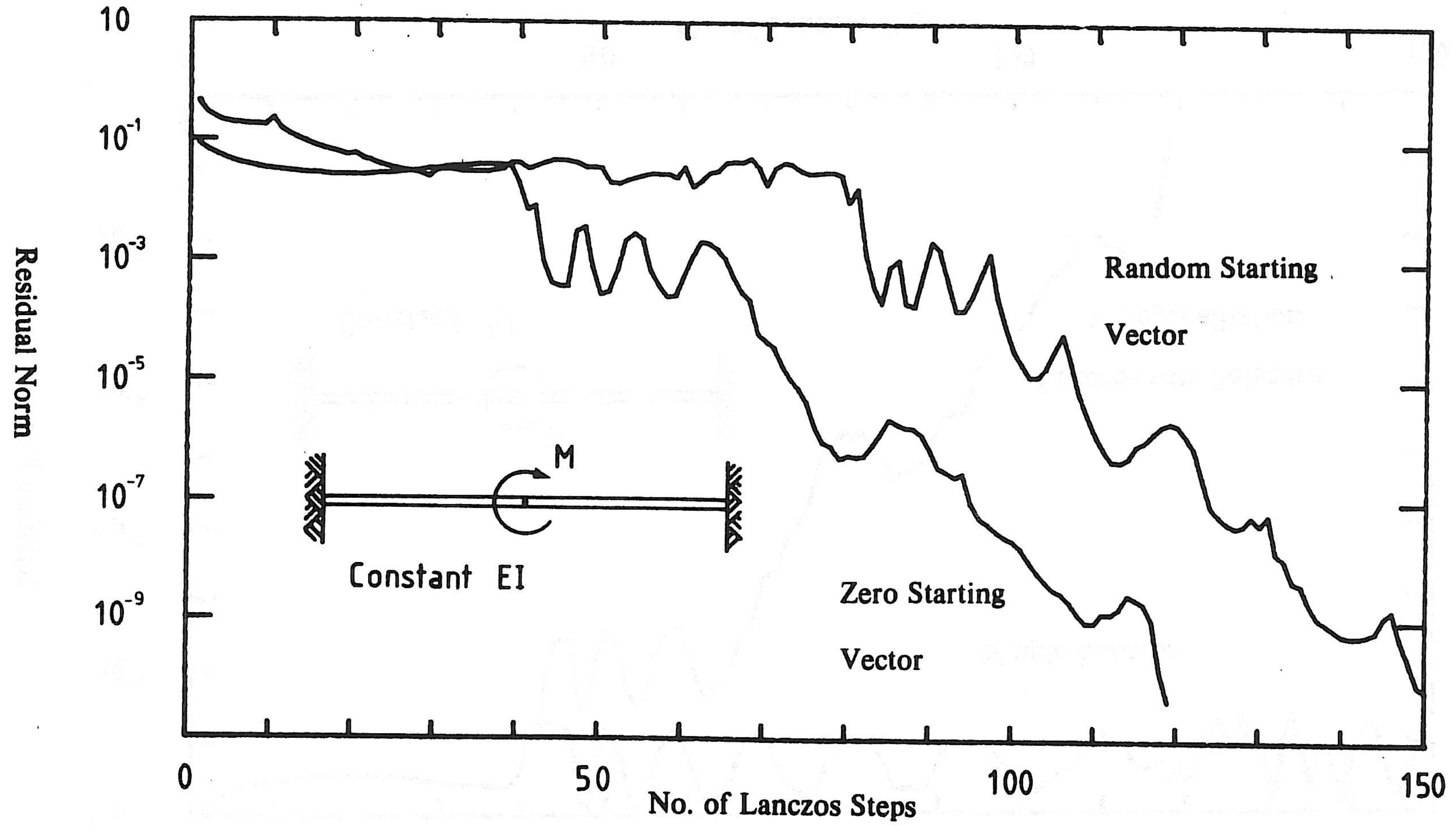


Figure 4.3 Effect of Different Starting Vectors on the Reduction of the Residual Norm in the Lanczos Algorithm.

5. NEWTON-LANCZOS METHOD

Generally in nonlinear structural analysis, the solution to the nonlinear equations is required at a number of load levels to generate a complete equilibrium history. The BFGS method described in section 3.3 attempts to make use of the information at a previous iterate, by updating the factorization of the tangent matrix, to find the next solution. This updating procedure imposes a limitation on the solution algorithm. That is:

> The BFGS can not be applied to nonlinear problems where the tangent matrix becomes indefinite during the iteration process.

The object of the BFGS algorithm was to reduce the overall number of matrix factorizations but this could only be achieved at the cost of limiting the class of problems for which the method can be useful.

In addition to the short comings of Newton type methods described in section 3, all these methods including Newton's method have the following limitations.

(1) These methods are not globally convergent, i.e not all starting vectors converge to the solution.

(2) The algorithm can not be applied to the cases when the tangent matrix becomes singular at some $\mathbf{x}_k$.

Here global convergence means that if $\mathbf{f}(\mathbf{x}) = 0$ has a unique solution then all starting vectors converge to this solution.

The object of the proposed algorithm is

(i) Remove the above two restrictions.

(ii) Reduce the number of matrix factorization.

5.1. Description of the Algorithm

The Newton-Lanczos algorithm is based on the simple idea that so long as the approximate solution is far from the exact solution, it is not necessary to solve the linearized problem exactly, but only a modest level of accuracy is sufficient.

The algorithm consists of choosing an initial guess $\mathbf{x}_0$ and repeating the following steps for $k = 0,1,2, \cdots$, until convergence is achieved:

(1) given $\mathbf{x}_k$ compute $\mathbf{f}(\mathbf{x}_k)$

(2) compute the tangent matrix $\mathbf{A}_k$

(3) find some $\mathbf{d}_k$ such that $\dfrac{\|\mathbf{A}_k\mathbf{d}_k + \mathbf{f}(\mathbf{x}_k)\|}{\|\mathbf{f}(\mathbf{x}_k)\|} \leqslant \eta_k$

(4) compute s_k from a line search

(5) update $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k\mathbf{d}_k$

(6) test for convergence and terminate if converged

Here η_k is some parameter such that $\eta_k < 1$. The only difference between this algorithm and Newton's method is in step 3. At this step the Lanczos algorithm as described in section 4 is used to obtain $\mathbf{d}_k$ and η_k is the specified solution tolerance for the Lanczos algorithm.

It is evident from the description of the algorithm that if $\mathbf{A}_k$ is singular at some step k then the process need not be terminated as long as a $\mathbf{d}_k$ can be found that can achieve the required tolerance. Under this circumstance it is desirable to have small components of the eigenvectors corresponding to the zero eigenvalues along $\mathbf{d}_k$. The behavior of the Lanczos algorithm is such that the process takes close to n steps, where n is the number of unknowns, before it introduces any component of these eigenvectors into the solution vector. The approximate solution is obtained from a linear combinations of the Krylov vectors and it is the matrix-vector multiplications that annihilates components of the eigenvectors corresponding to zero eigenvalue. Therefore, it is possible to obtain an approximate solution without introducing large components of the undesirable eigenvectors.

The cost of step 3 can be reduced by obtaining a good initial guess to the solution of the new system. This can be done by solving the weak form of the linearized equation at a previous step i,

$$\mathbf{T}_j^{(i)}\mathbf{g}_j = \mathbf{h}_j$$

with the appropriate right hand side. The projection of the residual vector, $\mathbf{f}(\mathbf{x}_k)$, on the Lanczos vectors of the previous step is identically this right hand side, That is

$$\mathbf{h}_j = \mathbf{Q}_j^{(i)T}\mathbf{f}(\mathbf{x}_k)$$

5.2. Convergence of Newton-Lanczos

Newton-Lanczos algorithm has very attractive convergence properties. The solution part of the algorithm can be rearranged to get

$$\|\mathbf{A}_k\mathbf{d}_k + \mathbf{f}(\mathbf{x}_k)\| \leqslant \eta_k\|\mathbf{f}(\mathbf{x}_k)\| \tag{5.2.1}$$

The convergence of Newton-Lanczos method can be proved provided the following assumptions hold.

(a) The solution $\mathbf{x}^*$ to $\mathbf{f}(\mathbf{x}^*)$ is unique.

(b) The Jacobian matrix exists.

(c) $\mathbf{f}'(\mathbf{x}^*)$ is nonsingular.

The Taylor expansion about $\mathbf{x}_k$ can be written as

$$\mathbf{f}(\mathbf{x}_{k+1}) = \mathbf{f}(\mathbf{x}_k) + \mathbf{A}_k\mathbf{d}_k + \mathbf{e}(\mathbf{d}_k) \tag{5.2.2}$$

where $\mathbf{e}(\mathbf{d}_k)$ includes all the higher order terms in $\mathbf{d}_k$. Hence

$$\|\mathbf{f}(\mathbf{x}_{k+1})\| \leqslant \|\mathbf{f}(\mathbf{x}_k) + \mathbf{A}_k\mathbf{d}_k\| + \|\mathbf{e}(\mathbf{d}_k)\| \tag{5.2.3}$$

Using equation 5.2.1 this can be modified to

$$\|\mathbf{f}(\mathbf{x}_{k+1})\| \leqslant \eta_k\|\mathbf{f}(\mathbf{x}_k)\| + \|\mathbf{e}(\mathbf{d}_k)\| \tag{5.2.4}$$

At this stage there are two possible cases that need be considered:

(1) $\eta_k\|\mathbf{f}(\mathbf{x}_k)\| \leqslant \|\mathbf{e}(\mathbf{d}_k\|$

This implies that $\|\mathbf{f}(\mathbf{x}_{k+1})\| < 2\|\mathbf{e}(\mathbf{d}_k)\| = O(\|\mathbf{d}_k\|^2)$ which in turn implies at least superlinear convergence.

(2) $\|e(d_k)\| < \eta_k \|f(x_k)\|$

This inequality can be stated as an equality of the form

$$\|e(d_k)\| = t\eta_k \|f(x_k)\| \tag{5.2.5}$$

where t is a positive scalar such that $t < 1$, then equation 5.2.4 can be written as

$$\|f(x_{k+1})\| < \gamma \|f(x_k)\| \tag{5.2.6}$$

where $\gamma = (t + 1)\eta_{\max}$ is a scalar. Choosing $\eta_{\max}$ such that $\eta_k < \eta_{\max} < \gamma < 1$ for all values of k Then

$$\lim_{k \to \infty} f(x_k) = 0 \tag{5.2.7}$$

These results show that the algorithm is at least linearly convergent,i.e

$$\|x_{k+1} - x^*\| \leqslant \alpha \|x_k - x^*\| \qquad \alpha < 1 \tag{5.2.8}$$

It can also be observed directly from the algorithm that by setting $\eta_k = \epsilon$, the computer precision, the algorithm will achieve a quadratic rate of convergence.

In general the rate of convergence is superlinear.

$$\|x_{k+1} - x^*\| \leqslant \alpha \|x_k - x^*\|^{q(\eta_k)} \tag{5.2.9}$$

where $1 \leqslant q(\eta_k) \leqslant 2$ and $\lim_{\eta \to 0} q(\eta) = 2$. The superlinear rate of convergence can be guaranteed by choosing η_k such that $\eta_k \to 0$ as $x_k \to x^*$. This can be achieved very simply by making η_k dependent on $\frac{\|f(x_k)\|}{\|f(x_0)\|}$. In the present implementation this dependence was chosen as

$$\eta_k = \eta_0 \left[\frac{\|f(x_k)\|}{\|f(x_0)\|} \right]^{1.5}$$

based on some computational experiments.

5.3. Preconditioning of The Linear Equations

The use of preconditioning of the linear system of equations can often speed up convergence of the Lanczos algorithm considerably. The number of Lanczos steps required to reduce the residual norm by a factor η depends, in general, on the distribution of the eigenvalues of A. An upper bound, p, to this number can be obtained from following relation

$$\eta = \left[\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right]^p \tag{5.3.1}$$

where κ is the condition number of A, $\kappa = \lambda_n/\lambda_1$. This upper bound is too crude to have any practical use for estimating the number of Lanczos steps but it can be seen that when the condition number is unity Lanczos algorithm requires only one step to obtain the exact solution. The equation 4.1.1 can now be modified to

$$\mathbf{C}^{-1}\mathbf{A}\mathbf{C}^{-T}\hat{\mathbf{x}} = \hat{\mathbf{r}} \tag{5.3.2}$$

where

$$\hat{\mathbf{x}} = \mathbf{C}^T\mathbf{x}^c \tag{5.3.3}$$

and

$$\hat{\mathbf{r}} = \mathbf{C}^{-1}\mathbf{r}_0 \tag{5.3.4}$$

where C is any matrix with an inverse that can be computed easily such as a triangular matrix. The object now is to solve 5.3.2 . The number of Lanczos steps can be reduced substantially by choosing C in such a way as to reduce the condition number of the product matrix of equation 5.3.2 .

The present implementation of the Newton-Lanczos algorithm uses the Cholesky factor of the tangent matrix at some previous step for C. In this way the information is passed on from on iterate to the next and therefore reduces the number of factorizations. However, factorizations can be eliminated completely if a good approximation to the preconditioning matrix C can be obtained. Moreover, the preconditioning can be updated from one step to the next using an updating procedure similer to the one used for the BFGS algorithm.

5.4. Numerical Results

Example I

This example, figure 5.1, is a simple truss structure with a peculiar property. The initial tangent matrix is singular. The purpose of this example is to demonstrate the global convergence of the Newton-Lanczos algorithm. A linearly elastic, large displacement truss element was used to model the structure which has a total of 8 degrees of freedom.

Using a zero starting solution, all the methods described in section 3, including Newton's method, diverged immediately after the initial iteration when applied to this problem. This is due to the fact that a direct equation solver is used to solve the linearized problem and since the tangent matrix is singular the solution vector will be an eigenvector corresponding to a zero eigenvalue with a large magnitude. This is like performing one step of inverse iteration method with a singular matrix which always converge to the eigenvector with smallest eigenvalue. No line search was used for Newton's method or Modified Newton method, however the BFGS algorithm which has line search also failed to converge.

Newton-Lanczos algorithm, with no preconditioning, when applied to this problem converged to the correct solution. However the convergence was only linear because of the singularity. The convergence rate improved as the approximate solution moved away from the singularity. Despite the low rate of convergence, the fact that the correct solution was obtained shows that the domain of attraction of Newton-Lanczos is larger than that of Newton's method. The load-displacement curve of figure 5.2 gives an indication of the stiffening behavior of the structure.

Example II

For a second example, we consider the shear loading of a block of rubber, as shown in Figure 5.3. The nonlinear elastic behavior of the rubber was modeled using a simplification of the general Rivlin model, originally proposed by Mooney [4]. The nonlinearities are present throughout the loading process. A nine node plane stress Lagrangian element, with penalty formulation [21], was used to discretize the problem to a total of 60 degrees of freedom.

The four basic methods were tested on this problem and the results are tabulated in Table 5.1. The convergence of the BFGS quasi-Newton method was disappointing since it required 11 iterations to converge at each step while the modified Newton which has only a linear convergence rate, required 14 iteration to converge.

The total number of factorizations required by each method is listed below.

Newton	9
BFGS	3
Modified Newton	3
Newton-Lanczos	1

The control of the rate of convergence for Newton-Lnaczos can be seen, Figure 5.4, when the method was used, without any preconditioning, to solve this problem with different η_0 values.

Example III

As a final example a parabolic beam illustrated in Figure 5.5 was modeled using 4 node, plane stress, quadrilateral elements. The nonlinearities enter the problem when the beam comes into contact with a rigid boundary. The contact condition is enforced by Lagrange multiplier constraints on those nodes indicating penetration and/or compressive contact force. The contact problem is of the same type as the problem of constrained optimization.

Both The BFGS and the Modified Newton method failed to converge when applied to this problem. This is because the tangent matrix becomes indefinite during the iteration process and the BFGS updates are incapable of accounting for the indefiniteness introduced by the contact elements.

The structure has stiffening characteristic during the loading process and a softening characteristic during the unloading phase. The total number of degrees of freedom is 870. The cost comparisons for Newton's method and Newton-Lanczos method are tabulated in Table 5.2. The total number of factorization required by each method is as follows:

Newton	13
Newton-Lanczos	2

5.5. Closure

The motivation for the development of quasi-Newton methods was reduction in the overall cost of the analysis, using fewer factorization steps. However, the use of quasi-Newton methods to solve nonlinear finite element equations has been restricted to problems with positive definite Jacobians.

The Newton-Lanczos algorithm described herein not only requires less factorization steps than either the quasi-Newton method or the modified Newton method but also has the following advantages over its rivals:

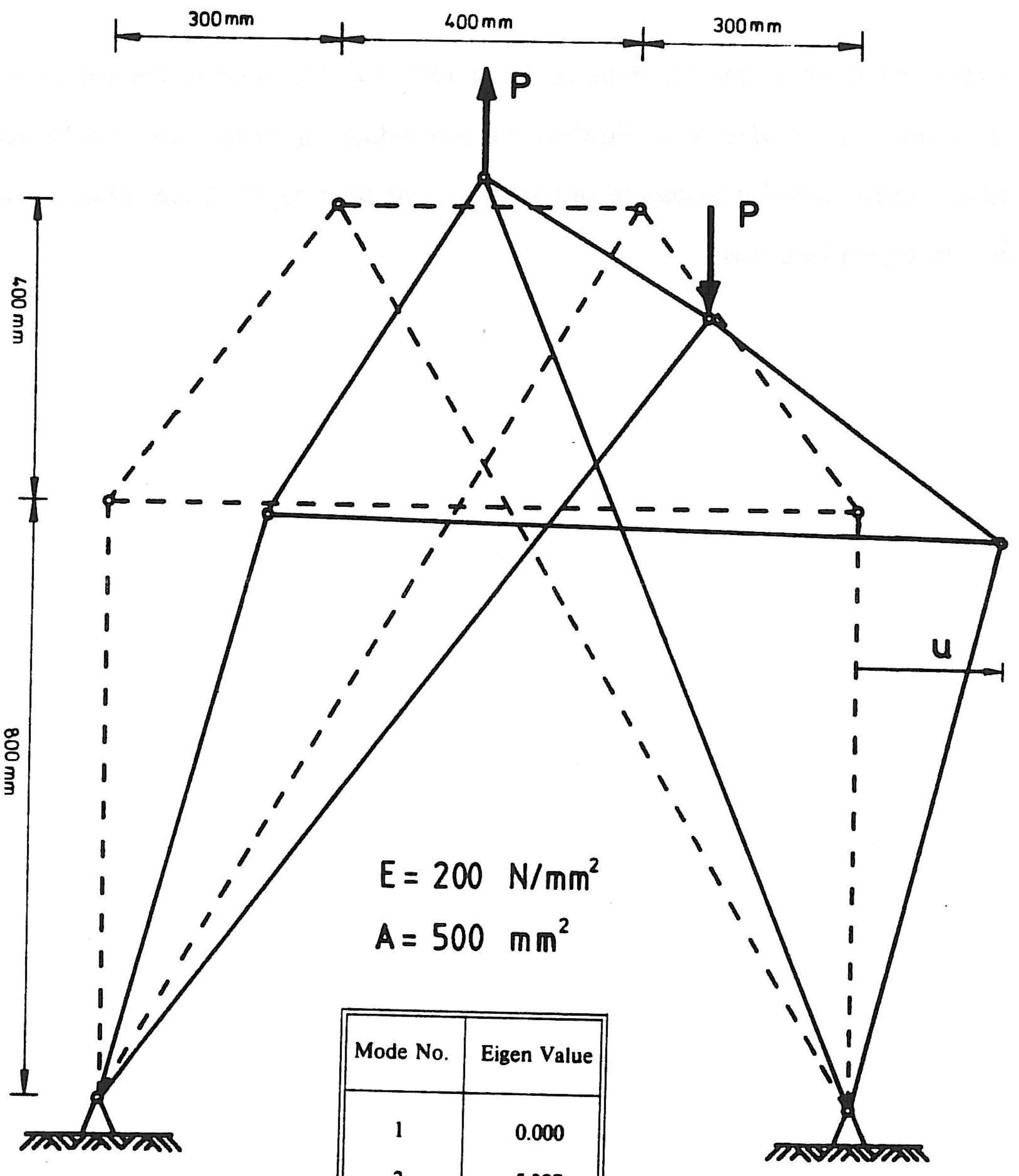
(1) The method is globally convergent.

(2) The rate of convergence can be controlled through a specified tolerance.

(3) The algorithm does not terminate in the case of a near singular Jacobian matrix.

(4) The algorithm can be applied to indefinite systems.

In the present implementation of Newton-Lanczos, the Jacobian matrix is decomposed into its Cholesky factors in order to get a preconditioning matrix for the Lanczos algorithm. This factorization is avoided if an effective preconditioning matrix can be constructed by other means, resulting in an additional reduction in the overall storage requirement of the algorithm. Further, the preconditioning matrix can then be updated using a sparse updating procedure similar to the one used by BFGS algorithm, for use in the subsequent iterations.



Mode No.	Eigen Value
1	0.000
2	5.397
3	8.157
4	8.873
5	17.292
6	36.912
7	45.428
8	67.337

Figure 5.1 Properties of the Truss Structure used in Example I.

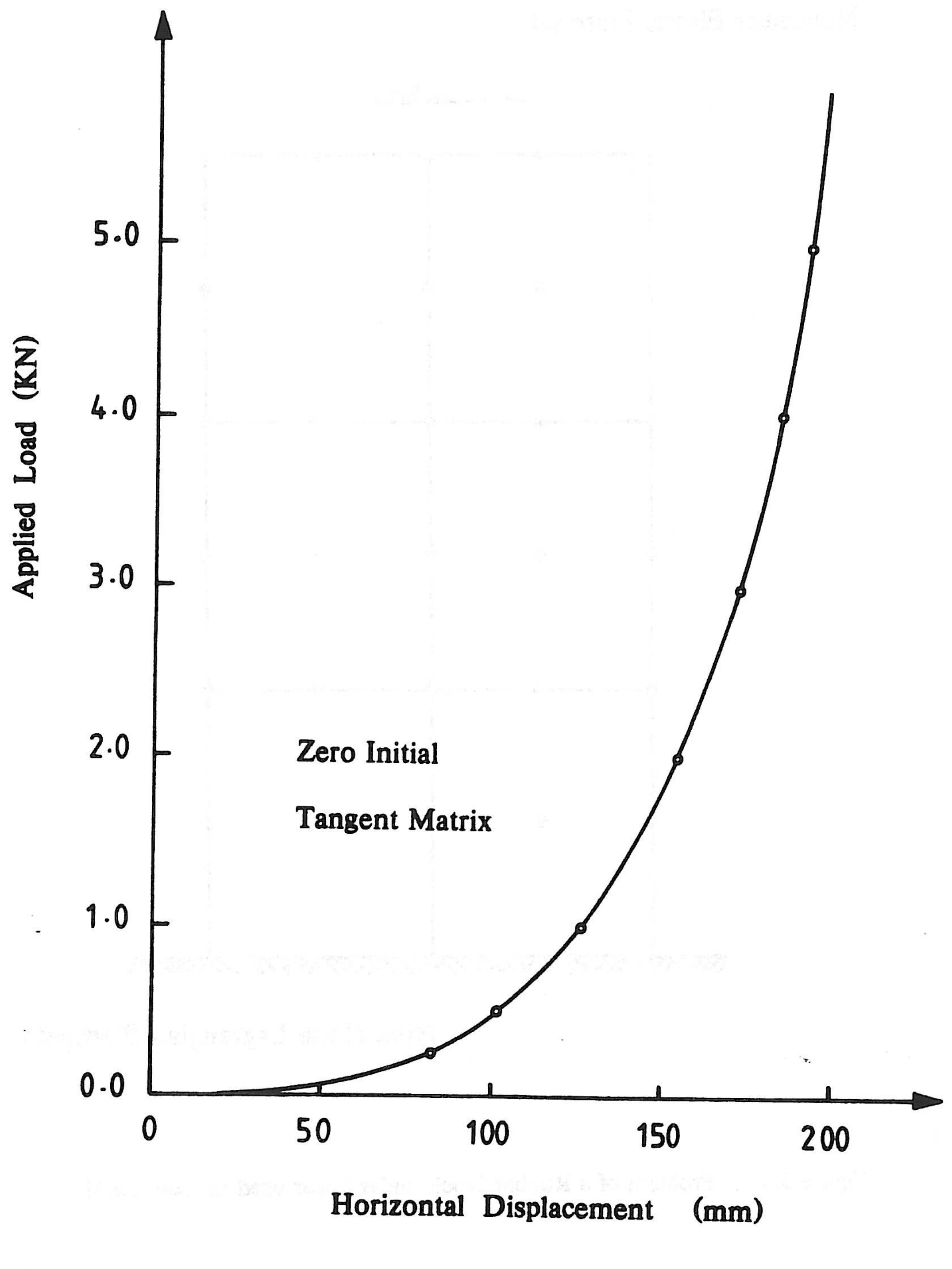


Figure 5.2 Load-Displacement Characteristic of Example I.

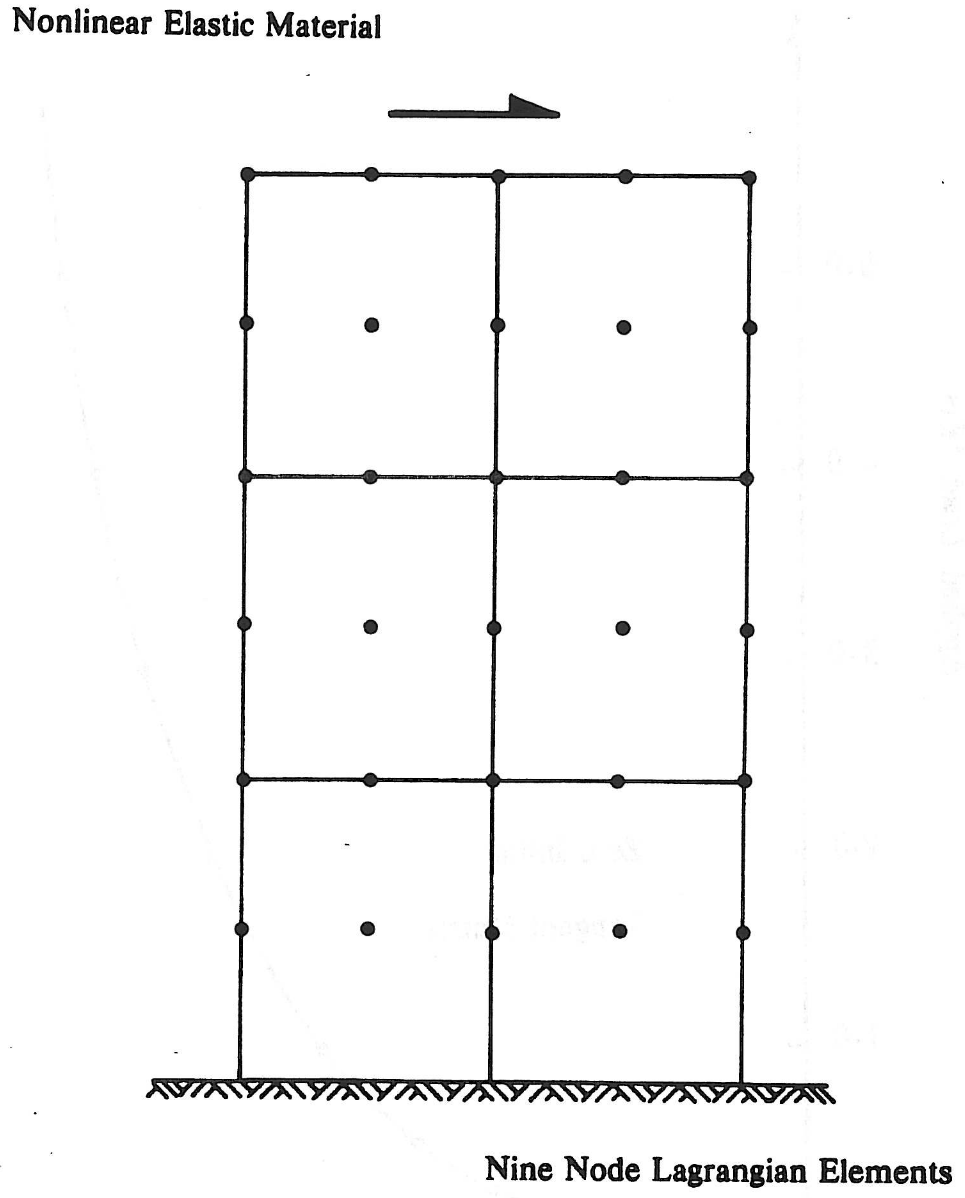


Figure 5.3 Problem of a Rubber Block under Shear used in Example II.

Load level	Method	No. of Iteration	No. of LU Factor.	No. of Function Evaluations	No. of Matrix-Vector ops.
1	Newton	3	3	4	3
	Mod. Newton	14	1	15	14
	BFGS	11	1	12	11
	N-L $\eta_0 = 10^{-3}$	4	1	5	25
	N-L $\eta_0 = 10^{-5}$	3	1	4	15
2	Newton	3	3	4	3
	Mod. Newton	14	1	15	14
	BFGS	11	1	12	11
	N-L $\eta_0 = 10^{-3}$	3	0	4	18
	N-L $\eta_0 = 10^{-5}$	3	0	4	24
3	Newton	3	3	4	3
	Mod. Newton	14	1	15	14
	BFGS	11	1	12	11
	N-L $\eta_0 = 10^{-3}$	3	0	4	18
	N-L $\eta_0 = 10^{-5}$	3	0	4	32

Table 5.1 Cost comparisons of different methods using

Example II (Rubber Block).

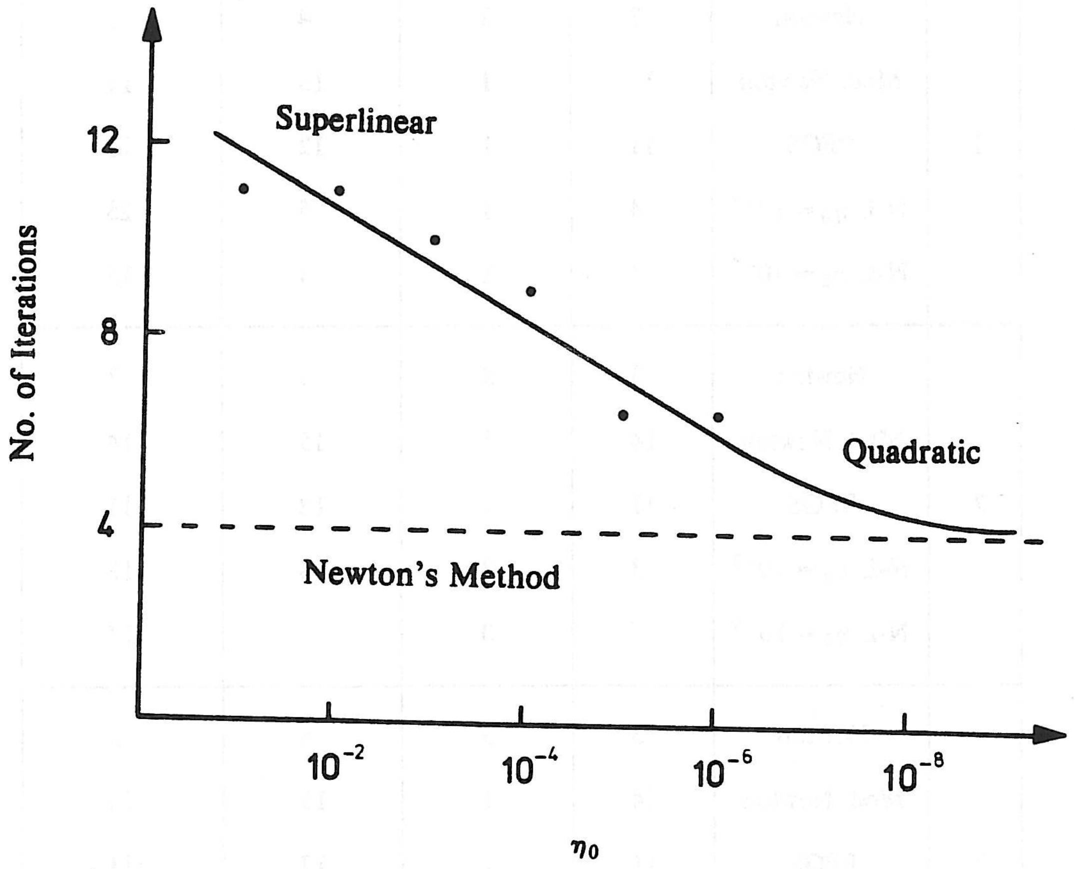


Figure 5.4 Effect of η_0 on the Rate of Convergence of the Newton-Lanczos algorithm.

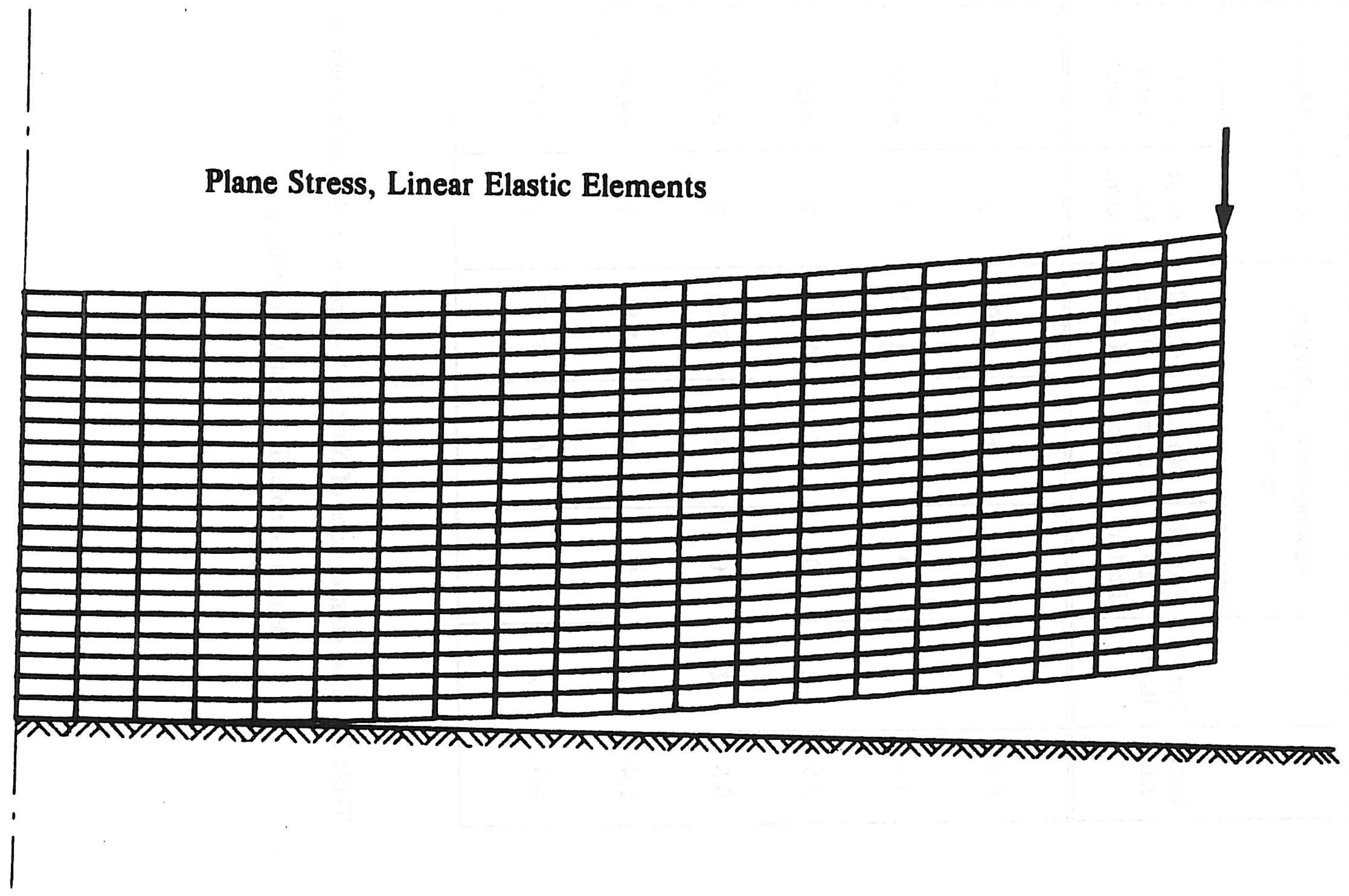


Figure 5.5 Contact Problem of a Parabolic Beam used in Example III.

Load (KN)	No. of Iterat.	Newton-Lanczos Method $\eta_0 = 10^{-7}$			Newton's Method	
		No. of LU Factor.	No. of Lanczos steps	C.P.U. time (sec.)	No. of LU Factor.	C.P.U. time (sec.)
0.4	3	2	6	64.01	3	67.22
0.6	2	0	3	18.11	2	44.43
0.9	2	0	7	28.29	2	43.69
0.6	2	0	8	33.18	2	44.45
0.4	2	0	4	22.16	2	44.45
0.0	2	0	6	31.09	2	44.32

Table 5.2 Cost comparisons of Newton-Lanczos and Newton's Method using Example III (contact problem).

APPENDIX A

Orthogonalization of the Krylov Vectors

Consider the set of vectors $(\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \cdots, \mathbf{A}^{j-1}\mathbf{r}_0)$. Assume for the moment that a set of orthonormal vectors, $(\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_j)$, are produced by successivly orthogonalizing $\mathbf{A}^i\mathbf{r}_0$ to all the previous vectors, all vectors $\mathbf{A}^k\mathbf{r}_0$ for $k < i$, and then normalizing the resulting vector to obtain the next orthonormal vector $\mathbf{q}_{j+1}$. Repeating this procedure for $i = 1, 2, \cdots, j-1$ defines the Gram-Schmidt method.

At some step $i \leqslant j-1$ we have

$$\tilde{\mathbf{r}}_i = \mathbf{A}^i\mathbf{r}_0 - \sum_{k=1}^{i} \gamma_k \mathbf{q}_k \tag{A1}$$

where $\gamma_k = \mathbf{q}_i^T\mathbf{A}^i\mathbf{r}_0$ and $\mathbf{q}_{i+1} = \dfrac{\tilde{\mathbf{r}}_i}{\|\tilde{\mathbf{r}}_i\|}$ but $\mathbf{A}^{i-1}\mathbf{r}_0 \in span(\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_i)$. This can be rewritten as

$$\mathbf{A}^{i-1}\mathbf{r}_0 = \sum_{k=1}^{i} \nu_k \mathbf{q}_k \tag{A2}$$

where $\nu_k = \mathbf{q}_k\mathbf{A}^{i-1}\mathbf{r}_0$. Equations A1 and A2 can now be combined to give

$$\tilde{\mathbf{r}}_i = \sum_{k=1}^{i} \nu_k \mathbf{A}\mathbf{q}_k - \sum_{k=1}^{i} \gamma_k \mathbf{q}_k \tag{A3}$$

and in turn $\mathbf{A}\mathbf{q}_{i-1} \in span(\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_i)$. Therefore

$$\bar{\mathbf{r}}_i = \mathbf{A}\mathbf{q}_i - \sum_{k=1}^{i} \bar{\gamma}_k \mathbf{q}_k \tag{A4}$$

where $\bar{\gamma}_k = \mathbf{q}_k^T\mathbf{A}\mathbf{q}_i$ and now $\mathbf{q}_{i+1} = \dfrac{\bar{\mathbf{r}}_i}{\|\bar{\mathbf{r}}_i\|}$.

At some step j, the orthonormalization proccess described by equation A4 must be performed for the vector $\mathbf{A}\mathbf{q}_j$. However, consider the case when we orthogonalize this vector only to the previous two vectors, $\mathbf{q}_j$ and $\mathbf{q}_{j-1}$. Then we get

$$\mathbf{r}_j = \mathbf{A}\mathbf{q}_j - \alpha_j\mathbf{q}_j - \beta_j\mathbf{q}_{j-1} \tag{A5}$$

where $\alpha_j = \mathbf{q}_j^T\mathbf{A}\mathbf{q}_j$ and $\beta_j = \mathbf{q}_{j-1}^T\mathbf{A}\mathbf{q}_j$. The orthogonality of $\mathbf{r}_j$ to the remaining vectors can now be checked by forming $\mathbf{q}_i^T\mathbf{r}_j$ for $i < j - 1$

$$\begin{aligned} \mathbf{q}_i\mathbf{r}_j &= \mathbf{q}_i^T(\mathbf{A}\mathbf{q}_j - \alpha_j\mathbf{q}_j - \beta_j\mathbf{q}_{j-1}) \\ &= \mathbf{q}_i^T\mathbf{A}\mathbf{q}_j \\ &= \mathbf{q}_j^T\mathbf{A}\mathbf{q}_i \end{aligned} \tag{A6}$$

This is derived using the orthogonality condition of the $\mathbf{q}$ vectors and the symmetry property of A. This eqution together with equation A4 leads to the following:

$$\mathbf{q}_i\mathbf{r}_j = \mathbf{q}_j^T(\|\overline{\mathbf{r}}_i\|\mathbf{q}_{i+1} + \sum_{k=1}^{i}\overline{\gamma}_k\mathbf{q}_k) = 0$$

This results shows that $\mathbf{r}_j$ is orthogonal to all the previous vectors $\mathbf{q}_i$ $i = 1,2, \cdots ,j$. Therefore to generate the orthonormal bases, at each step $\mathbf{A}\mathbf{q}_j$ need only be orthogonalized aqainst $\mathbf{q}_j$ and $\mathbf{q}_{j-1}$.

APPENDIX B

Mininum Potantial Energy over Krylov Space

The potential energy of a linear system is given by

$$E^n(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b} \tag{B1}$$

where the E^n denotes the energy in the n-dimentional space. The minimum of this scalar function occurs when

$$\nabla E^n(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} = 0 \tag{B2}$$

When the variables are restricted to the space of Lanczos vectors they are redefined as

$$\mathbf{x} = \mathbf{Q}_j\mathbf{f}_j$$

and

$$\mathbf{b} = \beta_1\mathbf{q}_1 = \beta_1\mathbf{Q}_j\mathbf{e}_1$$

The potential energy now becomes

$$\begin{aligned} E^j(\mathbf{f}_j) &= \tfrac{1}{2}\mathbf{f}_j^T\mathbf{Q}_j^T\mathbf{A}\mathbf{Q}_j\mathbf{f}_j - \mathbf{f}_j^T\mathbf{Q}_j\mathbf{e}_j\beta_1 \\ &= \mathbf{f}_j^T\mathbf{T}_j\mathbf{f}_j - \mathbf{f}_j^T(\beta_1\mathbf{e}_1) \end{aligned} \tag{B3}$$

The minimum of the potential energy, restricted to the Krylov space, now occurs when

$$\nabla E^j(\mathbf{f}_j) = \mathbf{T}_j\mathbf{f}_j - \beta_1\mathbf{e}_1 = 0 \tag{B4}$$

References

[1] K. W. Brodlie, A. R. Gourlay and J. Greenstadt, "Rank-One and Rank-Two Corrections to Positive Definite Matrices Expressed in Product Form," *J. Inst. Math. Appl.*, v. 11, pp. 73-82, 1973.

[2] M. A. Crisfield, Letter to the Editor, *Int. J. Num. Meth. in Engr.*, v. 15, pp. 1419-1420, 1980.

[3] J. R. Dennis, Jr. and J. J. More, "Quasi-Newton Methods, Motivation and Theory," *SIAM Review*, v. 19, pp. 46-86, 1977.

[4] M. E. Gurtin, *Topics in Finite Elasticity*, CBMS-NSF Regional Conference series in Applied Mathematics, 35, SIAM 1981.

[5] M. R. Hestenes and E. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," *J. Res. Bur. Standards*, v. 49, pp. 409-436, 1952.

[6] A. S. Householder, *The Theory of Matrices in Numerical Analysis*, Blaisdell, 1964.

[7] S. Kaniel, "Estimates for some Computational Techniques in Linear Algebra," *Math. Comp.*, v. 20, pp. 369-378, 1966.

[8] C. Lanczos, "An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators," *J. Computational Phys.*, v. 26, pp. 43-65, 1977.

[9] C. Lanczos, "Solution of Systems of Linear Equations by Minimized Iterations," *J. Res. Nat. Bur. Standards*, v. 49, pp. 33-53, 1952.

[10] H. Matthis and G. Strang, "The Solution of Nonlinear Finite Element Equations," *Int. J. Num. Meth. Engr.*, v. 14, pp. 1613-1626, 1979.

[11] L. Nazareth, "On the BFGS Method Part I: Properties of The Standard Algorithm," submitted for publications.

[12] L. Nazareth, "On the BFGS Method Part II: Extended and Limited Memory Versions," submitted for publications.

[13] N. M. Newmark, "A Method of Computation for Structural Dynamics," *J. Eng. Mech. Div., ASCE*, v. 85, EM3, pp. 67-94,1959.

[14] M. Ortiz, *Topics in Constitutive Theory for Inelastic Solids*, Ph.D. Thesis, University of California, Berkeley, 1981.

[15] C. C. Paige, *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*, Ph.D. Thesis, University of London, 1971.

[16] C. C. Paige, "Computational Variants of the Lanczos Method for the Eigenproblem," *J. Inst. Math. Appl.*, v. 10, pp. 373-381, 1972.

[17] C. C. Paige and M. A. Saunders, "Solution of sparse indefinite systems of linear equations," *SIAM J. Num. Anal.*, v. 12, pp. 617-629, 1975.

[18] B. N. Parlett, *The Symmetric Eigenvalue Problem,* Prentice-Hall, Englewood Cliffs (1980).

[19] B. N. Parlett, "A New Look at the Lanczos Algorithm for Solving Symmetric Systems of Linear Equations," *Lin. Alg. Appl.*, v. 29, pp. 323-346, 1980.

[20] B. N. Parlett and D. Scott, "The Lanczos Algorithm with Selective Orthogonalization," *Math. Comp.* v. 33, pp. 217-238, 1979.

[21] J. C. Simo and R. L. Taylor, "Penalty Formulations for nonlinear incompressible elastostatics," to be published.

[22] H. Simon, *The Lanczos Algorithm with Selective Reorthogonalization,* Ph.D. Thesis, University of California, Berkeley, 1982.

[23] O. C. Zienkiewicz, *The Finite Element Method,* 3rd Edition, McGraw-Hill Book Co., London, 1977.