# TOWARDS OPTIMAL SOLUTION TECHNIQUES FOR LARGE EIGENPROBLEMS IN STRUCTURAL MECHANICS

by

SESHADRI RAMASWAMY
B.Sc., Bangalore University, India
(1971)
D.M.I.T. (Aero.), Madras Institute of Technology, India
(1974)
S.M., Massachusetts Institute of Technology
(1975)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF DOCTOR OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1980

© Massachusetts Institute of Technology 1980

Signature of Author Signature redacted
Department of Aeronautics & Astronautics
September 14, 1979

Certified by Signature redacted
Thesis Supervisor

Certified by Signature redacted
Thesis Supervisor

Certified by Signature redacted
Thesis Supervisor

Certified by Signature redacted
Thesis Supervisor

Accepted by
Chairman, Departmental Graduate Committee

ARCHIVES
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEC 5 1979
LIBRARIES

# TOWARDS OPTIMAL SOLUTION TECHNIQUES FOR LARGE EIGENPROBLEMS IN STRUCTURAL MECHANICS

by

SESHADRI RAMASWAMY

Submitted to the Department of
Aeronautics and Astronautics on September 14, 1979
in partial fulfillment of the requirements for
the degree of Doctor of Science

## ABSTRACT

An accelerated subspace iteration method has been developed for the calculation of eigenvalues in an interval and the associated eigenvectors of the eigenproblem $\underline{K}\,\underline{\phi} = \lambda\,\underline{M}\,\underline{\phi}$. The important features of the accelerated method are the following: self-adaptive overrelaxation and shifting strategies are employed to accelerate the convergence in the subspace iterations; the number of vectors that is used in the iteration can be smaller than the number of eigenpairs that has to be calculated; all eigenvalues in a specified interval and the associated eigenvectors can be computed. Also, it has been shown in this study that the vectors derived using the Lanczos method can be used effectively as the starting vectors for the subspace iterations. Numerical examples have been presented to illustrate the effectiveness of the accelerated subspace iteration method.

Thesis Supervisor: Klaus-Jürgen Bathe

Title: Associate Professor of Mechanical Engineering

ACKNOWLEDGEMENTS

I cannot adequately acknowledge my debt of gratitude to my thesis supervisor, Professor K. J. Bathe for his helpful guidance and support throughout my graduate study at M.I.T. I would like to thank Professor J. Dugundji, Professor T. H. H. Pian, Professor G. Strang and Professor E. A. Witmer for their valuable suggestions. I am grateful to the ADINA Users group for supporting financially this research work.

My special thanks are due to I. Buttlar and M. Toscano for typing this thesis fast and accurately. I am grateful to my friends K. S. Aravamudan, S. Bolourchi, M. D. Snyder, Sri Guha and A. V. Durga, who have helped me in several ways during my stay at M.I.T.

Finally, I would like to thank my parents, Seshadri and Kamala, and my wife, Revathi, who with their love and patience have helped me to make my dream come true.

TABLE OF CONTENTS


| Chapter No. | | Page No. |
|---|---|---|
| 1. | INTRODUCTION | 9 |
| 2. | A BRIEF REVIEW OF SOME EXISTING SOLUTION TECHNIQUES | 17 |
| 3. | THE BASIC SUBSPACE ITERATION METHOD | 25 |
| | 3.1 Introduction | 25 |
| | 3.2 A Description of the Basic Method | 26 |
| | 3.3 Some Computational Aspects of the Method | 35 |
| 4. | THE GENERATION OF THE STARTING VECTORS FOR THE ACCELERATED SUBSPACE ITERATION METHOD | 39 |
| | 4.1 Introduction | 39 |
| | 4.2 The "Standard" Starting Subspace | 40 |
| | 4.3 The Lanczos Method | 43 |
| | 4.4 Some Computational Aspects of the Lanczos Method | 51 |
| | 4.5 Summary of the Algorithm for Generating the Starting Subspace | 57 |
| 5. | THE ACCELERATED SUBSPACE ITERATION METHOD | 60 |
| | 5.1 Introduction | 60 |
| | 5.2 The Overrelaxation of Iteration Vectors | 62 |
| | 5.3 The Matrix Shifting Procedure | 72 |
| | 5.4 The Iteration Procedure for the Case q<p | 78 |
| | 5.5 Computer Implementation | 85 |
| | 5.6 Demonstrative Examples | 92 |
| 6. | THE CALCULATION OF EIGENVALUES IN A SPECIFIED INTERVAL | 108 |
| | 6.1 Introduction | 108 |
| | 6.2 A Critical Review of Some Existing Solution Techniques | 109 |
| | 6.3 Modifications to the Accelerated Subspace Iteration Method | 117 |
| 7. | SUMMARY AND CONCLUSIONS | 124 |
| | REFERENCES | 130 |
| | APPENDICES | |
| | A. AN ANALYSIS OF THE LANCZOS METHOD | 134 |
| | B. THE COMPUTER PROGRAM MSUBSP | 154 |
| | C. GLOSSARY OF TERMS | 206 |

# LIST OF FIGURES


| No. | | Page |
|---|---|---|
| 5.1 | An Example to Illustrate the Iteration Procedure for the Case q<p (q=10, p=60)..... | 83 |
| 5.2 | Calculations in Accelerated Subspace Iteration..... | 86 |
| 5.3 | A 16-Degree-of-Freedom Tridiagonal System..... | 93 |
| 5.4 | Geometric View of Piping System..... | 96 |
| 5.5 | A Three-Dimensional Building Frame..... | 100 |
| 5.6 | A Three-Dimensional Marine Probe..... | 104 |

# LIST OF TABLES


| No. | | Page |
|---|---|---|
| 3.1 | Summary of Basic Subspace Iteration Solution..... | 38 |
| 4.1 | Summary of the Lanczos Algorithm for Generating the Subspace Iteration Starting Vectors..... | 59 |
| 5.1 | Analysis of the Tridiagonal System (p=4, q=8)..... | 94 |
| 5.2 | Comparison of Different Solution Strategies in the Analysis of the Piping System..... | 97 |
| 5.3 | Solution Steps in the Analysis of the Piping System (p=60, q=20)..... | 98 |
| 5.4 | Comparison of Different Solution Strategies in the Analysis of the Building Frame..... | 101 |
| 5.5 | Solution Steps in the Analysis of the Building Frame (p=60, q=20)..... | 102 |
| 5.6 | Comparison of Different Solution Strategies in the Analysis of the Marine Probe..... | 105 |
| 5.7 | Frequency Solution Timing Information for the Analysis of the Marine Probe..... | 106 |
| 6.1 | Solution Results for the Calculation of Eigenvalues in the Specified Interval..... | 123 |

## LIST OF SYMBOLS

All symbols are defined in the text, and those not frequently used may have different meaning in different parts of the thesis. Only those which do not change meaning and are often used are listed below.

A bar ( _ ) underneath a symbol denotes that the symbol refers to a matrix or a vector.

The transpose of a matrix $\underline{X}$ is denoted as $\underline{X}^T$.

$\underline{K}$ = stiffness matrix

$\underline{M}$ = mass matrix

$n$ = order of $\underline{K}$ and $\underline{M}$

$m_{\underline{K}}$ = average half bandwidth of $\underline{K}$; bandwidth = $2m_{\underline{K}}+1$

$m_{\underline{M}}$ = average half bandwidth of $\underline{M}$; bandwidth = $2m_{\underline{M}}+1$

(throughout the text it is assumed that

$m_{\underline{M}} = m_{\underline{K}}$, if the matrix $\underline{M}$ is banded

$m_{\underline{M}} = 0$, if the matrix $\underline{M}$ is diagonal)

$\lambda_i$ = eigenvalues of the generalized eigenproblem $\underline{K}\,\underline{\phi} = \underline{\lambda}\,\underline{M}\,\underline{\phi}$

($\lambda_1 \leq \lambda_2 \leq \lambda_3 \ldots \leq \lambda_n$, unless stated otherwise)

$\underline{\phi}_i$ = eigenvector associated with the eigenvalue $\lambda_i$ of the generalized eigenproblem $\underline{K}\,\underline{\phi} = \lambda\,\underline{M}\,\underline{\phi}$

$\underline{\Lambda}$ = diagonal matrix storing the eigenvalues $\lambda_i$

$\underline{\Phi}$ = matrix storing the eigenvectors $\underline{\phi}_i$

$\underline{X}_k$ = matrix storing iteration vectors during the $k^{th}$ iteration

$\underline{K}_k$ = projection of matrix $\underline{K}$, equal to $\underline{X}_k^T\,\underline{K}\,\underline{X}_k$

$\underline{M}_k$ = projection of matrix $\underline{M}$, equal to $\underline{X}_k^T\,\underline{M}\,\underline{X}_k$

$\underline{\Lambda}_k$ = diagonal matrix storing eigenvalues of $\underline{K}_k\underline{Q}_k = \underline{M}_k\,\underline{Q}_k\,\underline{\Lambda}_k$

$\underline{Q}_k$ = matrix storing eigenvectors of $\underline{K}_k\,\underline{Q}_k = \underline{M}_k\,\underline{Q}_k\,\underline{\Lambda}_k$

$\underline{x}_i$ = a vector; also denotes the $i^{th}$ column of a matrix $\underline{X}$

$\mu$ = shift

$\underline{I}$ = identity matrix

$\underline{e}_i$ = $i^{th}$ column of the identity matrix $\underline{I}$

$r_i$ = convergence rate of the $i^{th}$ eigenvalue iterate

$r_i^k$ = estimated convergence rate of the $i^{th}$ eigenvalue iterate at $k^{th}$ iteration

$\lambda_i^k$ = approximation to the $i^{th}$ eigenvalue in the $k^{th}$ iteration

$\bar{\lambda}_i$ = converged value of the $i^{th}$ eigenvalue iterate

tol = the final convergence tolerance on eigenvalue iterates

tolc = accuracy of an eigenvalue approximation at $k^{th}$ iteration

# 1. INTRODUCTION

The objective in this research is to develop optimal solution techniques for large generalized eigenproblems that arise in the analysis of structural systems.

In the field of structural mechanics, the analysis of a number of physical phenomena requires the solution of an eigenproblem [1]. For example, in the dynamic response analysis of an assemblage of structural elements using conventional mode superposition the generalized eigenproblem considered is

$$\underline{K}\,\underline{\Phi} = \omega^2\,\underline{M}\,\underline{\Phi} \tag{1.1}$$

in which $\underline{K}$ and $\underline{M}$ are the stiffness and mass matrices, respectively, and the solutions to the eigenproblem are the free vibration frequencies squared and the corresponding free vibration mode shapes.

Another generalized eigenproblem arises in buckling analysis. The equations governing buckling of an assemblage of structural elements are

$$\underline{K}\,\underline{\Psi} = \lambda\,\underline{K}_G\,\underline{\Psi} \tag{1.2}$$

in which $\underline{K}$ and $\underline{K}_G$ are the linear and nonlinear strain (geometric) stiffness matrices, respectively; the eigenvalues give the buckling loads and the eigenvectors represent the corresponding buckling modes.

A third important eigenproblem arises in dynamic analysis when nonproportional damping is considered. In this case the solution is sought to the quadratic eigenproblem

$$(\underline{K} + \gamma\underline{C} + \gamma^2 \underline{M})\ \underline{\xi} = 0 \qquad (1.3)$$

in which $\underline{K}$, $\underline{M}$ and $\underline{C}$ are the stiffness, mass and damping matrices of the element assemblage. Assuming that the matrices are of order n, there are 2n complex eigenvalues and corresponding eigenvectors.

The rapid development of the digital computers has led to the increased use of computational methods which operate on discrete representation of physical problems. In particular, the use of finite element and finite difference techniques often necessitates the solution of eigenproblems in which the matrices $\underline{K}$, $\underline{M}$, $\underline{C}$ and $\underline{K}_G$ are of order 5000 and more. Fortunately for the analyst, the matrices possess some special properties, like sparseness and symmetry. Also, due to the approximate nature of the finite element and finite difference procedures, the larger eigenvalues of Eqs. (1.1) to (1.3) do not approximate to any accuracy the actual frequencies of the structural systems being analyzed [4, p. 227]. Hence, only the smallest eigenvalues of Eqs. (1.1) to (1.3) are usually calculated.

Wilkinson has summarized in his book [2] most of the solution techniques that were employed in the sixties, for the solution of the generalized eigenproblem. The techniques presented are primarily transformation methods, in which the coefficient matrices are transformed into a simpler form before the eigensolution is carried out. The transformation methods are not suitable for the solution of only a few eigenpairs of large eigenproblems. It should be noted that in the analysis of large finite element systems it is very important to use solution

algorithms that are developed specifically for the calculation of only the required eigenvalues and corresponding eigenvectors and which take maximum advantage of the characteristics of the finite element matrices. Therefore, several approximate eigensolution methods have been developed in the last ten years. A survey of the solution methods currently in use for the structural mechanics eigenproblems of the form in Eqs. (1.1) to (1.3), has been presented in ref. [9, 10, 12]. Detailed descriptions of some of the methods presented in these survey papers are given in ref. [8, 13-18]. The mass lumping techniques that have been employed to reduce the size of the eigenproblem to be solved, i.e. schemes to condense the matrices $\underline{K}$ and $\underline{M}$ in Eq. (1.1), are given in ref. [19-22].

In principle, as stated above, a large number of solution methods is currently available for eigenproblems of the form in Eqs. (1.1) to (1.3). However, when the order of the matrices is large, the central processor and the peripheral processor time and also the storage requirements are excessive for most solution methods. Therefore, in practice, the choice lies only between a few effective solution techniques. If inappropriate methods are used, the cost to solve a large eigenproblem can be several times larger than necessary and, indeed, a solution may not be possible at all. Considering the eigensolution of large structural systems with small bandwidth, the determinant search method developed by Bathe, has been shown to be very effective [1, 23]. When the matrices involved in the eigensolution can be completely kept in-core, the application of the Lanczos method for the

solution can be efficient [30-33]. For the eigensolution of structural systems with large bandwidth, the simultaneous iteration methods, in particular the subspace iteration method developed by Bathe,are considered to be the most effective at present [23-29]. These solution methods have been developed specifically for the structural mechanics eigenproblems and have been applied to a wide variety of problems with considerable success, in a number of general purpose structural analysis programs.

Considering the effectiveness of the three solution methods mentioned above when employed for the solution of large eigenproblems, the efficiency of the determinant search method results from the shifting strategy that is employed to accelerate the convergence in inverse iteration. However, since the number of factorizations that are performed to achieve the favorable convergence rate is relatively large, the method is not suitable for large bandwidth systems, particularly when a large number of eigenpairs are required.

The Lanczos method exhibits several numerical problems when implemented on a computer. Also, the generation of each vector in the Lanczos method requires an inverse iteration. When the matrices are stored out-of-core, the inverse iteration step increases the peripheral processing cost considerably, and hence the use of the Lanczos method in such cases is not effective. On the other hand, in the subspace iteration method, inverse iteration on several iteration vectors is

performed simultaneously, and by clever programming the peripheral processing cost is reduced. Hence the method is very effective when implemented as an out-of-core solver. Also, the parallel processing feature that is being introduced in the new generation computers increases the effectiveness of the method considerably [42]. However, recent experience indicates that the subspace iteration method can still be improved to become a few times more efficient [24]. The specific improvements that are necessary are discussed in detail in Chapter 3. Based on these considerations, it is seen that, for the solution of large eigenproblems, much additional potential lies in further development of the subspace iteration method. The same conclusion was reached by Strang and Fix who recognized the potential of the subspace iteration method and predicted that "the subspace iteration method may well become the accepted algorithm for finite element eigenvalue problems" [4, p. 240].

Scope of the present study

The objective of this research was to develop a solution method that is reliable and yet is cost effective for the solution of large eigenproblems in which the matrices have small or large bandwidths.

In Chapter 2, a brief review of some of the methods that are commonly employed for the solution of structural mechanics eigenproblems is presented. Naturally, the review is by no means completely self-contained and thorough; an interested reader can find an in-depth discussion of the theory of eigenproblems and of the various eigensolution methods in ref. [1, Chapters 10, 11 and 12]. Some of the technical terms that are used frequently in this thesis are defined in Appendix C.

In Chapter 3, we describe the basic subspace iteration method. Also, an analysis of the various steps in the solution method is given. It is pointed out that it is uneconomical to use the basic method to calculate a large number of eigenvalues and the associated eigenvectors of large eigenproblems. It should be noted that when a large number of eigenpairs has to be calculated, the current practice is to use approximate solution methods -- the approximate methods do not employ iterations and often lead to inaccurate solutions. Because of the stringent requirements developing for refined dynamic analysis of structures. it is necessary to develop solution methods that yield accurate solutions at minimum computational expense.

In this study, an attempt has been made to improve the subspace iteration method so that the method can be employed effectively for the calculation of a large number of eigenpairs.

In Chapter 4, some strategies to generate an effective starting subspace for the subspace iterations are presented. In particular, the use of random starting vectors is discussed. Also, a simple theoretical analysis of the Lanczos method is given to show that the method does not always predict good approximations, particularly when almost equal eigenvalues are present in the solution. Based on the analysis, it is further concluded that the vectors generated using the Lanczos method can be used effectively as the subspace iteration starting vectors.

In Chapter 5, overrelaxation and shifting procedures are presented in order to accelerate the convergence in the subspace iterations. The overrelaxation and shifting procedures developed in this study are self-adaptive schemes, i.e., the rate of convergence of the individual vectors in the subspace iterations is measured and the acceleration procedures are applied only when necessary and are economical. In the basic subspace iteration method, the number of iteration vectors that are employed in all iterations is greater than the number of eigenpairs that has to be calculated. In this study, it is shown that considerable amount of computational savings is obtained by not iterating with the converged iteration vectors. Also, the subspace iteration procedure has been modified so that the number of iteration vectors can be less than the number

of eigenpairs required. Thus, the amount of high speed core storage required to perform the subspace iterations can be reduced a great deal and the iteration procedure can be implemented on a mini-computer. Some numerical examples are presented that demonstrate the effectiveness of the above acceleration schemes.

Currently, very few solution methods are available for the calculation of eigenvalues within an interval. In Chapter 6, a critical review of these solution methods is presented. Then, specific modifications to the accelerated subspace iteration method (presented in Chapter 5) are given, in order that the method can be used to compute all eigenvalues within a specified interval. It is shown that this method, which employs simultaneous iteration on several vectors and self-adaptive shifting is reliable, accurate and is more effective than the solution methods that are currently available.

A computer program (in FORTRAN) was written which uses the accelerated subspace iteration procedure developed in this study. The user's manual and the listing of the computer program are given in Appendix B.

Finally, the conclusions of this study and some recommendations for future research are presented in Chapter 7.

## 2. A BRIEF REVIEW OF SOME EXISTING SOLUTION TECHNIQUES

A review of the solution methods that are currently used to calculate the eigenvalues and the eigenvectors of the generalized eigenproblem

$$\underline{K}\,\underline{\phi} = \lambda\,\underline{M}\,\underline{\phi} \tag{2.1}$$

is presented in this chapter. The solution methods under consideration have been classified into the following categories -- the transformation methods, the polynomial iteration methods and the vector iteration methods [1]. The transformation methods reduce the matrices $\underline{K}$ and $\underline{M}$ into a diagonal, tri-diagonal or triangular form by successive pre and post multiplication of the matrices $\underline{K}$ and $\underline{M}$ with certain types of elementary orthogonal matrices. The polynomial iteration techniques operate on the fact that the eigenvalues $\lambda_i$ of Eq. (2.1) are the roots of the $n^{th}$ degree polynomial $p(\lambda)$, where

$$p(\lambda) = |\underline{K} - \lambda\underline{M}| \tag{2.2}$$

In the vector iteration methods, a trial vector $\underline{x}$ is substituted in place of the eigenvector $\underline{\phi}$ in Eq. (2.1) either on the left hand side or on the right hand side; improved approximations to the eigenvector $\underline{\phi}$ are obtained by forcing the trial vector to satisfy Eq. (2.1) in an iterative manner. In addition to the above algorithms, also methods based on the minimization of the Rayleigh quotient are used [14 - 16].

### Transformation methods

In the transformation methods, the matrices $\underline{K}$ and $\underline{M}$ are reduced into a simpler form by means of similarity transformations; i.e, success-

ive pre and post multiplication by matrices $\underline{P}_k^T$ and $\underline{P}_k$, respectively, where k = 1, 2, ..., are performed. Define $\underline{K}_1 = \underline{K}$ and $\underline{M}_1 = \underline{M}$. Then, the matrices $\underline{K}_{k+1}$ and $\underline{M}_{k+1}$ are computed as

$$\underline{K}_{k+1} = \underline{P}_k^T \underline{K}_k \underline{P}_k \tag{2.3}$$

$$\underline{M}_{k+1} = \underline{P}_k^T \underline{M}_k \underline{P}_k \tag{2.4}$$

where the matrices $\underline{P}_k$ are selected to transform the matrices $\underline{K}_k$ and $\underline{M}_k$ closer to a diagonal or tri-diagonal form.

In the generalized Jacobi solution method the matrix $\underline{P}_k$ is a rotation matrix of the following form

$$\underline{P}_k = \begin{array}{c} \\ \\ i^{th} \\ \\ j^{th} \text{ row} \\ \\ \end{array} \overset{\begin{array}{cc} i^{th} & j^{th} \text{ column} \end{array}}{\begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & \alpha & \\ & & \gamma & 1 & \\ & & & & \ddots \\ & & & & & 1 \end{bmatrix}} \tag{2.5}$$

where the constants $\alpha$ and $\gamma$ are selected in such a way as to reduce to zero simultaneously elements (i,j) in $\underline{K}_k$ and $\underline{M}_k$ [1].

In the Householder method [1](applicable only to the standard eigenproblem) the matrix $\underline{P}_k$ is a reflection matrix of the form

$$\underline{P}_k = \underline{I} - \frac{2}{\underline{w}_k^T \underline{w}_k} \underline{w}_k \underline{w}_k^T \tag{2.6}$$

where the vector $\underline{w}_k$ is selected to zero out all the elements in the

columns (k+2) to n of the $k^{th}$ row. It should be noted that the Householder method reduces a general matrix into a tri-diagonal matrix and some other solution methods, like the QR method [1] and inverse iteration, should be employed to calculate the eigenvalues and eigenvectors of the tridiagonal matrix.

The eigensolution employing the generalized Jacobi method requires about $(18n^3 + 36n^2)$ numerical operations and the Householder method for the tridiagonalization requires about $(\frac{2}{3} n^3 + \frac{21}{2} n^2)$ numerical operations [1, p. 454, 472]. Also, when the transformation methods are employed to solve large eigenproblems, a large amount of high speed core storage and peripheral processing are required. On the other hand, the subspace iteration solution for the smallest p eigenvalues employing q iteration vectors, $q > p$, requires about $(nm^2 + 40\ nmq + 20nq^2)$ numerical operations, where m is the average bandwidth of the matrices $\underline{K}$ and $\underline{M}$ [1, p. 507]. It follows that it is not effective to use the transformation methods for finite element eigenproblems.

Polynomial iteration methods

The eigenvalues of the eigenproblem $\underline{K}\ \underline{\phi} = \lambda \underline{M}\ \underline{\phi}$ are the zeroes of the characteristic polynomial $p(\lambda) = |\underline{K} - \lambda \underline{M}|$. Therefore, the eigenvalues can be computed by using any of the several solution methods that are available for the calculation of the zeroes of a polynomial [2]. Since the order of the matrices considered is large, explicit calculation of the polynomial coefficients is very sensitive to round-off errors, particularly when the matrices $\underline{K}$ and $\underline{M}$ are ill-conditioned and

hence explicit polynomial iteration is never used [5]. In the implicit polynomial iteration, the value of the polynomial $p(\lambda)$ is calculated directly at discrete values of $\lambda$ as follows: the matrix $\underline{K}-\lambda\underline{M}$ is factorized

$$\underline{K} - \lambda\underline{M} = \underline{L}\,\underline{D}\,\underline{L}^T \tag{2.7}$$

Then, from the elements of the diagonal matrix $\underline{D}$, the polynomial value $p(\lambda)$ is calculated as

$$p(\lambda) = \prod_{i=1}^{n} d_{ii} \tag{2.8}$$

Thus with a scheme available for an accurate evaluation of $p(\lambda)$, a number of iterative schemes like the secant method or the Newton method, can be used to calculate the roots of the polynomial $p(\lambda)$.

The determinant search method [1, p. 486-494] involves secant iteration on the characteristic polynomial to obtain a good approximation to an eigenvalue. The Sturm sequence property is employed to identify a jump over unknown roots. Then, inverse iteration with the eigenvalue approximation as the shift is carried out to improve this approximation and if necessary further Rayleigh quotient shifting is performed. Gram-Schmidt orthogonalization is employed to orthogonalize the current iteration vector to the previously obtained eigenvectors. The determinant search method has been applied to a wide variety of problems with considerable success, in a number of general purpose structural analysis programs.

The disadvantages of the determinant search method are:

(i) the method is well suited only for small bandwidth systems,

because a relatively large number of triangular factorizations are required in the calculation of each eigenvalue and corresponding eigenvector.

(ii) The selection of effective starting vectors for inverse iteration, in particular in the calculation of almost equal eigenvalues, cannot be well automated.

(iii) The present numerical procedure cannot be used to calculate eigenvalues and corresponding eigenvectors within a prescribed interval, and significant extensions of the algorithm are required in order to obtain such a solution procedure.

(iv) It is important to use a high enough convergence tolerance to prevent premature acceptance of the iteration vector as an approximation to the required eigenvector.

(v) The eigenvalue approximation calculated using the accelerated secant iteration method can lie almost in-between two eigenvalues that have not been calculated as yet, in which case, the convergence in the inverse iteration is very slow. Also, the secant iteration does not always result in the calculation of a desirable value for the shift to be used in the inverse iteration, i.e., depending on the slope of the characteristic polynomial, sometimes several consecutive eigenpairs have to be calculated using only the inverse iteration, without any change in the shift value computed using the secant iteration.

Vector iteration methods

In the vector iteration methods the basic aim is to satisfy Eq. (2.1) by directly operating upon it. Assume a trial vector $\underline{x}_1$ for $\underline{\phi}_1$ and assume $\lambda_1=1$. Substituting the assumed values in the right-hand side of Eq. (2.1), we obtain

$$\underline{K}\,\underline{x}_2 = 1\,\underline{M}\,\underline{x}_1 \tag{2.9}$$

The vector $\underline{x}_2$ is obtained by solving the set of simultaneous equations in Eq. (2.9), and in general $\underline{x}_2 \neq \underline{x}_1$. However, $\underline{x}_2$ is a better approximation to the eigenvector $\underline{\phi}_1$ than was $\underline{x}_1$ and by repeating the cycle an increasingly better approximation to the eigenvector $\underline{\phi}_1$ is obtained. The approximation to the eigenvalue $\lambda_1$ is calculated as the Rayleigh quotient of the iteration vector, equal to $(\underline{x}_k^T\,\underline{K}\,\underline{x}_k/\underline{x}_k^T\,\underline{M}\,\underline{x}_k)$ [11]. The iteration procedure described in Eq. (2.9) forms the basis of inverse iteration. In forward iteration, the iterative cycle is reversed, i.e. Eq. (2.9) is modified as follows

$$\underline{M}\,\underline{x}_2 = \underline{K}\,\underline{x}_1 \tag{2.10}$$

In the inverse iteration, Eq. (2.9), the rate of convergence of the iteration vector to the eigenvector $\underline{\phi}_1$ is linear and is equal to $\lambda_1/\lambda_2$ [1]. The inverse iteration procedure can be used to compute any eigenvector of the eigenproblem in Eq. (2.1). If the matrix $\underline{K}$ in Eq. (2.9) is replaced by the matrix $(\underline{K} - \mu\underline{M})$ the iteration vector converges to the eigenvector $\underline{\phi}_j$ corresponding to the eigenvalue $\lambda_j$ that is nearest to $\mu$, of course, assuming that the starting vector $\underline{x}_1$ is not

orthogonal to the eigenvector $\underline{\phi}_j$. In the inverse iteration employing the matrix $(\underline{K} - \mu\underline{M})$, the convergence rate of the iteration vector to the eigenvector $\underline{\phi}_j$ is the larger value of the two ratios

$$\left| \frac{\lambda_j - \mu}{\lambda_{j-1} - \mu} \right| \text{ or } \left| \frac{\lambda_j - \mu}{\lambda_{j+1} - \mu} \right| \tag{2.11}$$

It is seen from Eq. (2.11) that the rate of convergence of the iteration vector to an eigenvector $\underline{\phi}_j$ is improved considerably by shifting close to the corresponding eigenvalue $\lambda_j$. The inverse iteration scheme can be employed to compute also several eigenpairs of the eigenproblem in Eq. (2.1). After the eigenvector $\underline{\phi}_1$ associated with the smallest eigenvalue $\lambda_1$ has been computed, inverse iteration in Eq. (2.9) with a new starting vector $\underline{x}_1$ is performed. But in the iteration, the iteration vector is orthogonalized to the previously computed eigenvector $\underline{\phi}_1$. It can be shown theoretically that under certain conditions in this iteration with orthogonalization, the iteration vector converges to the next eigenvector $\underline{\phi}_2$, etc.

The inverse iteration with shifting is a widely used eigensolution method. As seen earlier, inverse iteration is employed in the determinant search method to improve the eigenvalue approximation calculated using the secant iteration on the characteristic polynomial. A solution method in which the Sturm sequence property is employed to bracket the eigenvalues and inverse iteration is then used to compute accurately the required eigenpairs, is given in ref. [18]. A major shortcoming of the inverse iteration scheme in Eq. (2.9) is that the convergence can be very slow, particularly in the presence of almost equal eigenvalues. The "quality", i.e. the spectral content, of the starting iteration vector

also governs the number of iterations required for convergence. In the inverse iteration, it is necessary to employ a tight convergence tolerance in order to prevent premature acceptance of the iteration vector as an approximation to the required eigenvector. This observation holds particularly when the Gram-Schmidt orthogonalization of the iteration vectors is performed in the solution. In Chapter 3, the basic subspace iteration method is discussed in which a simultaneous inverse iteration on several iteration vectors is performed. In the simultaneous iteration scheme, the presence of multiple and almost equal eigenvalues does not lead to any solution difficulty and the convergence rates of the iteration vectors are also improved significantly.

# 3. THE BASIC SUBSPACE ITERATION METHOD

## 3.1 Introduction

In the previous chapter it was pointed out that amongst the several iterative methods discussed, the simultaneous vector iteration methods are best suited for the solution of large eigenproblems. The subspace iteration method is one of the eigensolution methods in which simultaneous iteration is used and is considered to be the most effective method at present for the calculation of a few lowest eigenvalues and associated eigenvectors of primarily large banded systems [1, 23]. The solution method developed in this study is based on the subspace iteration method and hence to familiarize the reader first with the basic subspace iteration method, a description and analysis of the various steps in the basic method are presented in Section 3.2. Finally, the shortcomings of the basic subspace iteration method are pointed out in Section 3.3.

## 3.2 A Description of the Basic Method

In essence, the basic subspace iteration method for the solution of generalized eigenproblems consists of the following three steps:

(1) Establish "q" starting vectors, $q > p$, which span the starting subspace $E_1$. Here "p" is the number of eigenpairs to be calculated.

(2) Perform subspace iterations, in which simultaneous inverse iteration is used on the q vectors and Ritz analysis is employed to extract optimal eigenvalue and eigenvector approxmations at the end of each simultaneous inverse iteration.

(3) After iteration convergence, use the Sturm sequence check to verify that indeed the required eigenpairs have been calculated.

Considering step (1), the most important and difficult part of the subspace iteration scheme is the selection of the starting subspace, since the number of iterations required for the solution directly depends on the quality of the starting vectors. Various schemes for selecting the starting subspace have been evaluated by Bathe [23]. The scheme that has been found to be very effective and has been widely used is the following: Assuming that the q starting vectors are established in the matrix $\underline{X}_1$, the first column in $\underline{X}_1$ is the diagonal of the mass matrix $\underline{M}$ and the other columns are unit coordinate vectors with their unit entries corresponding to the degrees of freedom with large mass and low stiffness values. Also, an appropriate spacing between the unit entries in the starting iteration vectors is provided in order to ensure that the masses in different coordinate

directions and different regions of the structure are excited.

Considering next step (2), the subspace iterations are performed as follows:

For k = 1, 2, ..., iterate from subspace $E_k$ to subspace $E_{k+1}$

$$\underline{K}\,\overline{\underline{X}}_{k+1} = \underline{M}\,\underline{X}_k \tag{3.1}$$

Calculate the projections of the matrices $\underline{K}$ and $\underline{M}$ onto $E_{k+1}$

$$\underline{K}_{k+1} = \overline{\underline{X}}_{k+1}^T\,\underline{K}\,\overline{\underline{X}}_{k+1} \tag{3.2}$$

$$\underline{M}_{k+1} = \overline{\underline{X}}_{k+1}^T\,\underline{M}\,\overline{\underline{X}}_{k+1} \tag{3.3}$$

Solve for the eigensystem of the projected matrices

$$\underline{K}_{k+1}\,\underline{Q}_{k+1} = \underline{M}_{k+1}\,\underline{Q}_{k+1}\,\underline{\Lambda}_{k+1} \tag{3.4}$$

Calculate an improved approximation to the eigenvectors

$$\underline{X}_{k+1} = \overline{\underline{X}}_{k+1}\,\underline{Q}_{k+1} \tag{3.5}$$

Then, provided that the iteration vectors in $\underline{X}_1$ are not orthogonal to one of the required eigenvectors (and assuming an appropriate ordering of the iterates), the $i^{th}$ diagonal entry in $\underline{\Lambda}_{k+1}$ converges to $\lambda_i$ and the $i^{th}$ vector in $\underline{X}_{k+1}$ converges to $\underline{\phi}_i$. In the subspace iteration convergence is measured on the eigenvalue approximations [1, p. 504],

$$\text{tolc} = \frac{|\lambda_i^{(k+1)} - \lambda_i^{(k)}|}{\lambda_i^{(k+1)}} \tag{3.6}$$

where for convergence tolc must be smaller than tol, the final convergence tolerance; tol is typically equal to $10^{-6}$. The subspace iterations are stopped, if for the first p consecutive eigenvalue iterates the condition tolc $\leq$ tol is satisfied and next a Sturm sequence check is performed.

Considering the Sturm sequence check in step (3) the procedure to apply this check has been described in detail in ref. [1, p. 505]. This check is based on the fact that the number of negative elements measured in $\underline{D}$ is equal to the number of eigenvalues below the shift $\mu$, where

$$\underline{K} - \mu\underline{M} = \underline{L}\,\underline{D}\,\underline{L}^T \quad . \tag{3.7}$$

Hence at a shift $\mu$ just to the right of $\bar{\lambda}_p$ (which is the calculated approximation to the $p^{th}$ eigenvalue $\lambda_p$) the Sturm sequence check is performed to verify whether indeed the p lowest eigenpairs have been calculated. It should be emphasized here that the above check only guarantees that the smallest p eigenvalues have been approximated. However, considering the accuracy of the computed eigenpairs it is necessary that the following error estimates [1, p. 413] be small for all the eigenpairs calculated,

$$\varepsilon_i = \frac{\| \underline{K}\,\underline{\phi}_i - \bar{\lambda}_i\,\underline{M}\,\underline{\phi}_i \|_2}{\| \underline{K}\,\underline{\phi}_i \|_2} \quad . \tag{3.8}$$

The Sturm sequence check and the calculation of the above physical error norms together constitute a robust and reliable method of checking the accuracy of a calculated eigensolution.

An analysis of the basic subspace iteration method

The first step in the subspace iteration, Eq. (3.1), is the simultaneous inverse iteration on all the iteration vectors in $\underline{X}_k$. In the inverse iteration the component of the eigenvector $\underline{\phi}_i$ present in each iteration vector is scaled down by the corresponding eigenvalue, $\lambda_i$. It follows that, by repeating just this inverse iteration step several times, the components of the eigenvectors associated with the larger eigenvalues are reduced much more than the components of the eigenvectors associated with the smaller eigenvalues. But, in the computer, the numerical magnitude of the elements of the iteration vectors decreases (or increases, depending on the magnitude of the eigenvalues) with each inverse iteration, finally leading to the occurrence of underflow (or overflow). Therefore, it is necessary to normalize the iteration vectors in some specific manner. For example, after an inverse iteration, the Euclidean norm of each iteration vector is scaled to unity. However, the normalization of iteration vectors does not prevent the faster reduction of the components of the eigenvectors associated with the larger eigenvalues, that are presumably present in the starting iteration vectors. Hence, due to finite precision arithmetic, the iteration vectors rapidly become more and more parallel and converge to the eigenvector $\underline{\phi}_1$. The iteration vectors should be orthogonalized after each simultaneous inverse iteration in order to ensure that all iteration vectors do not converge to the eigenvector $\underline{\phi}_1$. The orthogonalization can be effected by means of the Gram-Schmidt process. In simultaneous iteration with Gram-Schmidt orthogonalization the $i^{th}$ column in $\underline{X}_{k+1}$ converges linearly to the $i^{th}$ eigenvector with the convergence rate equal to max $\{\lambda_i/\lambda_{i+1}, \lambda_{i-1}/\lambda_i\}$.

When the Gram-Schmidt process is employed for orthogonalization in the iterations and the $i^{th}$ starting iteration vector is particularly a good approximation to the required $j^{th}$ eigenvector ($i \neq j$), no advantage is taken of that fact. This inefficiency is caused by enforcing the orthogonality strictly from left to right. A method that has been devised to overcome this problem and improve considerably the convergence rate in the simultaneous iteration is the Ritz procedure [23, 25]. The Ritz analysis consists of the calculation of the projections of the matrices $\underline{K}$ and $\underline{M}$ onto the subspace spanned by the vectors in $\underline{X}_{k+1}$ and to subsequently compute the optimum eigenvalue and eigenvector approximations corresponding to this subspace. Equations (3.2) to (3.5) illustrate how the Ritz analysis is effected. It should be noted that the Ritz analysis induces a mutual coupling between the iteration vectors - a coupling that cannot be realized by employing Gram-Schmidt orthogonalizations, i.e. each iteration vector is simultaneously orthogonalized to all the other iteration vectors in $\underline{X}_k$. Hence if the fifth starting iteration vector happens to be the eigenvector $\underline{\phi}_2$, in the Ritz analysis this fact is recognized and the fifth iteration vector is left unmodified. Also, since all the other iteration vectors are orthogonalized to the fifth iteration vector, the components of the eigenvector $\underline{\phi}_2$ present in all of them are removed. Of course, the above consideration also implies that the eigenvalue approximations predicted in Eq. (3.4) must be arranged in an appropriate order at the end of each iteration, if meaningful convergence estimates have to be obtained using these approximations.

Some acceleration procedures to improve the convergence rate in the subspace iterations are presented in Chapter 5. The acceleration schemes

are based on the order and ultimate rate of convergence of the iterates in the subspace iteration. Hence in order to render this presentation fairly self-contained, the convergence analysis of the subspace iteration presented in ref. [24] is briefly summarized in this section.

Convergence analysis

As demonstrated in refs. [1, 24, 27], the convergence of an iterative eigensolution scheme is conveniently studied by first expressing the iteration vectors in $\underline{X}_k$ in terms of the eigenvector basis, instead of the reference coordinate basis that was used to establish the matrices $\underline{K}$ and $\underline{M}$, i.e., the vectors in $\underline{X}_k$, Eq. (3.1), are expressed as,

$$\underline{X}_k = \underline{\Phi}\,\underline{Z}_k \tag{3.9}$$

where $\underline{\Phi}$ is the nxn matrix storing all the eigenvectors, $\underline{\Phi} = [\underline{\phi}_1, \underline{\phi}_2, \ldots, \underline{\phi}_n]$. Since $\underline{\Phi}$ is nonsingular, there is a unique $\underline{Z}_k$ for any $\underline{X}_k$ and vice versa. For the sake of simplicity, let us assume stable convergence; i.e., we assume that the iteration vectors in $\underline{X}_k$ are not orthogonal to any of the eigenvectors associated with the lowest q eigenvalues. Then without loss of generality the iteration matrix $\underline{Z}_k$ can always be expressed as follows,

$$\underline{Z}_k = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ 0 & 0 & & 0 \\ 0 & 0 & & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & 1 \\ z^{(k)}_{q+1,1} & z^{(k)}_{q+1,2} & & z^{(k)}_{q+1,q} \\ z^{(k)}_{q+2,1} & z^{(k)}_{q+2,2} & & z^{(k)}_{q+2,q} \\ \vdots & \vdots & & \vdots \\ z^{(k)}_{n,1} & z^{(k)}_{n,2} & \cdots & z^{(k)}_{n,q} \end{bmatrix} \tag{3.10}$$

Introducing the relation of Eq. (3.9) into Eqs. (3.1) to (3.5), it can be seen that, in the inverse iteration step (Eq. (3.1)) the $i^{th}$ row of the matrix $\underline{Z}_k$ is divided by the $i^{th}$ eigenvalue $\lambda_i$ etc. Therefore, the elements of the matrices $\underline{K}_{k+1}$ and $\underline{M}_{k+1}$ in Eqs. (3.2) and (3.3) are, for $i=1, 2, \ldots q$, $j=1, 2, \ldots q$,

$$(K_{k+1})_{i,j} = \delta_{ij}/\lambda_j + \sum_{\ell=q+1}^{n} z^{(k)}_{\ell,i}\, z^{(k)}_{\ell,j}/\lambda_\ell \tag{3.11}$$

$$(M_{k+1})_{i,j} = \delta_{ij}/\lambda_j^2 + \sum_{\ell=q+1}^{n} z^{(k)}_{\ell,i}\, z^{(k)}_{\ell,j}/\lambda_\ell^2 \tag{3.12}$$

where $\delta_{ij}$ is the Kronecker delta.

After a few subspace iterations have been performed and the iteration vectors have settled down, the second terms on the right hand side of the above expressions become negligible compared to the first term,

and the matrix $\underline{Q}_{k+1}$ in Eq. (3.4) tends to a qxq diagonal matrix with the estimates for the eigenvalues $\lambda_i$, i=1, 2, ..., q, along the diagonal. Therefore, the iteration matrix $\underline{Z}_{k+1}$ in Eq. (3.5) is approximately,

$$\underline{Z}_{k+1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & 1 \\ z^{(k)}_{q+1,1}\,\frac{\lambda_1}{\lambda_{q+1}} & z^{(k)}_{q+1,2}\,\frac{\lambda_2}{\lambda_{q+1}} & & z^{(k)}_{q+1,q}\,\frac{\lambda_q}{\lambda_{q+1}} \\ z^{(k)}_{q+2,1}\,\frac{\lambda_1}{\lambda_{q+2}} & z^{(k)}_{q+2,2}\,\frac{\lambda_2}{\lambda_{q+2}} & & z^{(k)}_{q+2,q}\,\frac{\lambda_q}{\lambda_{q+2}} \\ \vdots & \vdots & & \vdots \\ z^{(k)}_{n,1}\,\frac{\lambda_1}{\lambda_n} & z^{(k)}_{n,2}\,\frac{\lambda_2}{\lambda_n} & \cdots & z^{(k)}_{n,q}\,\frac{\lambda_q}{\lambda_n} \end{bmatrix} \tag{3.13}$$

In the iteration, convergence is achieved when the $i^{th}$ column of the matrix $\underline{Z}_{k+1}$ tends to the vector $\underline{e}_i$, indicating that the $i^{th}$ iteration vector has converged to the $i^{th}$ eigenvector. Hence the order and the ultimate rate of convergence of the $i^{th}$ iteration vector to the $i^{th}$ eigenvector is thus obtained by evaluating

$$r_i = \frac{\|\underline{Z}_i^{(k+1)} - \underline{e}_i\|_2}{\|\underline{Z}_i^{(k)} - \underline{e}_i\|_2} = \left\{ \frac{\sum_{\ell=q+1}^{n} \left( z^{(k)}_{\ell,i}\ \lambda_i/\lambda_\ell \right)^2}{\sum_{\ell=q+1}^{n} \left( z^{(k)}_{\ell,i} \right)^2} \right\}^{1/2} \tag{3.14}$$

where $\underline{Z}_i^{(k)}$ is the $i^{th}$ column of the matrix $\underline{Z}_k$. Simplifying the above expression we obtain,

$$r_i = \frac{\lambda_i}{\lambda_{q+1}} \left\{ \frac{\sum_{\ell=q+1}^{n} \left( z_{\ell,i}^{(k)} \, \lambda_{q+1}/\lambda_\ell \right)^2}{\sum_{\ell=q+1}^{n} \left( z_{\ell,i}^{(k)} \right)^2} \right\}^{1/2} \leq \frac{\lambda_i}{\lambda_{q+1}} \qquad (3.15)$$

Hence we conclude that in case of stable convergence, and with appropriate ordering of the columns in $\underline{X}_{k+1}$, the $i^{th}$ column in $\underline{X}_{k+1}$ converges <u>linearly</u> to $\underline{\phi}_i$ with the rate $(\lambda_i/\lambda_{q+1})$. Since the eigenvalues are calculated as the Rayleigh quotients of the vectors in $\underline{X}_{k+1}$, the $i^{th}$ eigenvalue iterate in Eq. (3.4) converges linearly to $\lambda_i$ with the ultimate convergence rate equal to $(\lambda_i/\lambda_{q+1})^2$.

## 3.3 Some Computational Aspects of the Method

Some computational aspects of the basic subspace iteration method are discussed in this section.

An important point of consideration is the number of iteration vectors, q, to be used in the subspace iteration. Since the convergence rate in the iteration is inversely proportional to the magnitude of the $(q+1)^{st}$ eigenvalue, $\lambda_{q+1}$, it follows that the larger q, the smaller is the number of iterations required for convergence. Also as q is increased, the possibility of missing any of the required eigenpairs in the calculated eigensolution becomes smaller. On the other hand, depending upon the spread in the system eigenvalues being considered, the solution efficiency may not increase in direct proportion to the increase in the number of the iteration vectors. This observation holds particularly in the solution of large systems, where the matrices $\underline{K}$ and $\underline{M}$ need to be divided into blocks and stored out-of-core. Hence the number of iteration vectors required to yield an optimal (cost-effective) solution cannot be predicted well without an a priori knowledge of the eigenspectrum; but, based on considerable experience, the value for q that has been suggested in ref. [1] is,

$$q = \text{min. } \{2p,\ p+8\} \tag{3.16}$$

Since in Eq. (3.16) q is always greater than p, the possibility of $\lambda_p$ being a multiple root is taken into account.

Considering the computer implementation of the basic subspace iteration method, the implementation proposed in ref. [23] leads to a very

efficient usage of the available high speed storage, i.e., apart from the storage required for the stiffness and mass matrices, additional storage for only (q+2) vectors of length n is required, where n is the order of the matrices. The number of numerical operations required to perform the various steps of the basic method in the computer is given in Table 3.1. In the solution of large eigenproblems the matrices and the iteration vectors may have to be blocked and stored out-of-core. In such cases, to obtain a more realistic estimate of the cost of performing subspace iterations, the cost of peripheral processing must also be taken into account, i.e., the cost of retrieval and storage of information onto auxiliary storage units must also be included in Table 3.1.

Considering next the solution of eigenproblems in which a relatively large number of eigenpairs (say $p > 20$) is required, experience shows that the cost of solution using the basic subspace iteration method rises rapidly as the number of eigenpairs considered is increased. This rapid increase in cost is due to a number of factors that can be neglected when the solution of only a few eigenpairs is required. An important point is that a relatively large number of subspace iterations may be required, if the default value for q given above is employed. Namely in this case, when p is large, the convergence rate to $\underline{\phi}_p$, equal to $\lambda_p/\lambda_{q+1}$, can be close to one. On the other hand, if q is increased, the numerical operations per subspace iteration are increased significantly. Another shortcoming of the basic subspace iterations with large q is that a relatively large number of iteration vectors is used throughout <u>all</u> subspace iterations in Eqs. (3.1) to (3.5). Namely, convergence to the smallest eigenvalues is generally achieved in only a very few iterations, and the

converged vectors plus the $(p+1)^{st}$ to $q^{th}$ iteration vectors are only included in the additional iterations to provide solution stability and to accelerate the convergence to the larger required eigenvalues. A further important consideration pertains to the high-speed core and low-speed back-up storage requirements. As the number of iteration vectors $q$ increases, the number of matrix blocks that need be used in an out-of-core solution can also increase significantly and the peripheral processing expenditures can be large. Finally, it is noted that the number of numerical operations required in the solution of the reduced eigenproblem in Eq. (3.4) becomes significant when $q$ is large, and must be included in the operation count of the subspace iteration method (see Table 3.1).

The above brief discussion shows that modifications to increase the effectiveness of the basic subspace iteration procedure are very desirable, in particular when the solution of a large number of eigenpairs is considered. Some preliminary results that were obtained in the course of the current research were presented earlier in ref. [24]. As stated in ref. [24], a number of techniques can be employed, such as Aitken acceleration, overrelaxation, the use of Chebyshev polynomials and shifting; however, the difficulty is to provide a reliable and significantly more effective solution method.

In the next three chapters, we present the theory and implementation of the acceleration procedures for the basic subspace iteration method that have been developed in the current study.

TABLE 3.1 SUMMARY OF BASIC SUBSPACE ITERATION SOLUTION

| Operation | Calculation | Number of Operations | |
|---|---|---|---|
| | | $m = m_{\underline{K}} = m_{\underline{M}}$ | $m = m_{\underline{K}}$, $m_{\underline{M}} = 0$ |
| Factorization of $\underline{K}$ | $\underline{K} = \underline{L}\,\underline{D}\,\underline{L}^T$ | $\frac{1}{2}nm^2 + \frac{3}{2}nm$ | $\frac{1}{2}nm^2 + \frac{3}{2}nm$ |
| Subspace iteration | $\underline{K}\,\overline{\underline{X}}_{k+1} = \underline{Y}_k$ | $nq(2m+1)$ | $nq(2m+1)$ |
| | $\underline{K}_{k+1} = \overline{\underline{X}}_{k+1}^T\,\underline{Y}_k$ | $\frac{1}{2}nq(q+1)$ | $\frac{1}{2}nq(q+1)$ |
| | $\overline{\underline{Y}}_{k+1} = \underline{M}\,\overline{\underline{X}}_{k+1}$ | $nq(2m+1)$ | $nq$ |
| | $\underline{M}_{k+1} = \overline{\underline{X}}_{k+1}^T\,\overline{\underline{Y}}_{k+1}$ | $\frac{1}{2}nq(q+1)$ | $\frac{1}{2}nq(q+1)$ |
| | $\underline{K}_{k+1}\,\underline{Q}_{k+1} = \underline{M}_{k+1}\,\underline{Q}_{k+1}\,\underline{\Lambda}^2_{k+1}$ | $18q^3+36q^2$ | $18q^3+36q^2$ |
| | $\underline{Y}_{k+1} = \overline{\underline{Y}}_{k+1}\,\underline{Q}_{k+1}$ | $nq^2$ | $nq^2$ |
| Sturm sequence check | $\overline{\underline{K}} = \underline{K}-\mu\underline{M}$ | $n(m+1)$ | $n$ |
| | $\overline{\underline{K}} = \underline{L}\,\underline{D}\,\underline{L}^T$ | $\frac{1}{2}nm^2+\frac{3}{2}nm$ | $\frac{1}{2}nm^2+\frac{3}{2}nm$ |
| Error estimates for p eigenpairs | $\dfrac{\lVert \underline{K}\,\underline{X}_i^{(\ell+1)} - \lambda_i^{(\ell+1)}\,\underline{M}\,\underline{X}_i^{(\ell+1)} \rVert_2}{\lVert \underline{K}\,\underline{X}_i^{(\ell+1)} \rVert_2}$ | $(4mn+5n)p$ | $(2mn+5n)p$ |

# 4. THE GENERATION OF THE STARTING VECTORS FOR THE ACCELERATED SUBSPACE ITERATION METHOD

## 4.1 Introduction

In the description of the basic subspace iteration method in Section 3.2 it was mentioned that the number of subspace iterations required for convergence depends largely on the starting vectors selected. In some specific applications good approximations to the required eigenvectors may be readily available. For instance, in dynamic optimization, the structure is usually modified in small steps and the eigensystem of the previous structure is in general a good approximation to the eigensystem of the modified structure. Similarly, if some eigenvectors have already been evaluated and in a subsequent analysis more eigenvectors need be calculated, it is effective to use the already calculated eigenvectors along with new iteration vectors to span the starting subspace for the subsequent analysis. However, such good starting vectors based on prior knowledge are not always available. Hence, it is necessary to incorporate into the subspace iteration method, a procedure to generate the starting iteration vectors.

In this chapter, first a modification to the starting subspace described in Section 3.2 is presented. Then, based on a detailed analysis of the Lanczos method [30], it is shown that also the vectors generated using the Lanczos method are effectively used as starting vectors for the subspace iterations.

## 4.2 The "Standard" Starting Subspace

Assuming that the starting iteration vectors are established in matrix $\underline{X}_1$, the starting subspace described in Section 3.2 consists of the diagonal of the mass matrix $\underline{M}$ as the first column in matrix $\underline{X}_1$. The other (q-1) columns of matrix $\underline{X}_1$ are unit coordinate vectors $\underline{e}_i$, where the $i^{th}$ entry in the vector $\underline{e}_i$ is unity and all the other entries are zero. The following procedure is used to locate the degrees of freedom at which the unit entries have to be placed: Compute the ratios $k_{ii}/m_{ii}$, namely the ratios of the $i^{th}$ diagonal elements of the matrices $\underline{K}$ and $\underline{M}$. Arrange the ratios in ascending order, simultaneously keeping track of the numbers of the associated degrees of freedom. From the sequence of degree of freedom numbers thus obtained, the first (q-1) degrees of freedom are chosen as the degrees of freedom at which the unit entries in the starting iteration vectors are placed. When the ratios are sorted, also a special provision is made to ensure that the masses in different regions of the structure are excited [23, p. 54].

To establish the starting subspace described above, it is seen that the relative magnitudes of only the diagonal elements of $\underline{K}$ and $\underline{M}$ are used. In order to improve the effectiveness of the starting subspace, i.e. reduce the number of iterations required for final convergence, the off-diagonal elements of the matrices $\underline{K}$ and $\underline{M}$ should also be considered in the generation of the starting vectors. For example, let us consider the frequency analysis of beam type structures. In the

finite element discretization of such structures, often a diagonal mass matrix $\underline{M}$ is employed. In general, at each node of the beam finite elements, the diagonal elements of the matrix $\underline{M}$ corresponding to both the axial and the translational degrees of freedom are assigned equal values; but at each node, the stiffness terms corresponding to the translational degrees of freedom are substantially smaller than the stiffness term corresponding to the axial degree of freedom. This consideration implies that the ratios $k_{ii}/m_{ii}$ for the translational degrees of freedom are considerably smaller than the similar values for the axial degrees of freedom. Hence in the starting subspace described above, all the unit entries are placed only in the translational degrees of freedom unless q is larger than the total number of translational degrees of freedom.

In the solution of large eigenproblems a direct consideration of all the off-diagonal elements (like the calculation of all the coupling factors $k_{ij}^2/k_{ii}k_{jj}$ [24]) involves significant data handling and hence is not economical. Instead as a possible means of improving the starting subspace, the use of starting vectors with random numbers [27] was studied. In the numerical experiments, the starting subspace employing all random starting iteration vectors was sometimes more effective and sometimes considerably less effective than the starting subspace described above. The superiority of either starting subspace could

not be established from these numerical tests. However, when the last iteration vector in the starting subspace described above was replaced by a random vector, the number of iterations required for convergence was always reduced.

The following procedure, hereafter referred to as the "standard" procedure, has been used to generate the starting iteration vectors for the analysis of the numerical examples presented in later chapters.

- The first starting iteration vector is the diagonal of the matrix $\underline{M}$;
- The next (q-2) vectors each are unit coordinate vectors with the unit entries corresponding to the degrees of freedom with large mass and low stiffness values;
- the $q^{th}$ starting iteration vector is a random vector.

The "standard" starting procedure has been employed for the solution of a large number of eigenproblems. The scheme has proven to be very reliable and even when multiple and close roots are present in the eigenproblem being solved, always accurate solutions have been obtained.

## 4.3 The Lanczos Method

The Lanczos method is a technique originally proposed for the tridiagonalization of matrices. Later it was pointed out that even if the tridiagonalization is not carried out to completion, good approximations to the extremal eigenvalues are obtained from the reduced matrices [7, p. 106]. Recently, several papers on employing the Lanczos method for the calculation of only a few extremal eigenvalues of generalized eigenproblems have been published [30-33]. In this section, first a description of the Lanczos method is presented. Then, it is shown theoretically that, when close eigenvalues are present in the eigenproblem being solved, the Lanczos method does not always predict good approximations to such close eigenvalues. However, the subspace spanned by the sequence of vectors generated using the Lanczos method does tend to be close to the least dominant subspace of the operators $\underline{K}$ and $\underline{M}$. Hence, it is concluded that the vectors generated by the Lanczos method can be effectively used as the starting vectors for the subspace iteration. Since the solution is obtained in an iterative manner in the latter method, by employing a tight convergence tolerance in the subspace iterations, accurate and reliable solutions are obtained.

The Lanczos algorithm for the solution of the generalized eigenproblem

$$\underline{K}\,\underline{\phi} = \lambda\,\underline{M}\,\underline{\phi} \tag{4.1}$$

consists of the following steps:

Choose an arbitrary starting vector $\underline{x}$ and normalize the vector with respect to the matrix $\underline{M}$; i.e. Let

$$\gamma = \{\underline{x}^T \underline{M} \underline{x}\}^{1/2} \qquad \underline{x}_1 = \underline{x}/\gamma \tag{4.2}$$

With $\beta_1 = 0$, define for i = 1, 2, ..., q-1,

$$\underline{K}\, \underline{\bar{x}}_{i+1} = \underline{M}\, \underline{x}_i \tag{4.3}$$

$$\alpha_i = \underline{\bar{x}}_{i+1}^T \underline{M}\, \underline{x}_i \tag{4.4}$$

$$\underline{\tilde{x}}_{i+1} = \underline{\bar{x}}_{i+1} - \alpha_i \underline{x}_i - \beta_i \underline{x}_{i-1} \tag{4.5}$$

$$\beta_{i+1} = \{\underline{\tilde{x}}_{i+1}^T \underline{M}\, \underline{\tilde{x}}_{i+1}\}^{1/2} \tag{4.6}$$

$$\underline{x}_{i+1} = \underline{\tilde{x}}_{i+1}/\beta_{i+1} \tag{4.7}$$

Theoretically, the sequence of vectors, $\underline{x}_j$, j=1, ... q, generated using the above Eqs. (4.2) to (4.7) are $\underline{M}$-orthonormal and the matrix $\underline{X} \equiv [\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_q]$ satisfies the following relation,

$$\underline{X}^T \underline{A}\, \underline{X} = \underline{T}_q \tag{4.8}$$

where $\underline{A} = \underline{M}\, \underline{K}^{-1}\, \underline{M}$ ; $\underline{T}_q$ is a tridiagonal matrix of order q

$$\underline{T}_q = \begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \bullet & \bullet & \bullet & & \\ & & & & \beta_{q-1} & \alpha_{q-1} & \beta_q \\ & & & & & \beta_q & \alpha_q \end{bmatrix} \tag{4.9}$$

where the elements of the matrix $\underline{T}_q$, namely $\alpha_i$ and $\beta_i$, were defined in Eqs. (4.4) and (4.6) respectively. The relation given in Eq. (4.8) can be proven by some algebraic manipulation of the Eqs. (4.2) to (4.7), (see Appendix A). This proof shows that also the three sets of vectors, (i) $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_q$, (ii) $\underline{x}_1, \underline{K}^{-1}\ \underline{M}\ \underline{x}_1, (\underline{K}^{-1}\ \underline{M})^2\ \underline{x}_1, \ldots, (\underline{K}^{-1}\ \underline{M})^{q-1}\ \underline{x}_1$ and (iii) $\underline{x}_1, \underline{K}^{-1}\ \underline{M}\ \underline{x}_1, \underline{K}^{-1}\ \underline{M}\ \underline{x}_2, \ldots, \underline{K}^{-1}\ \underline{M}\ \underline{x}_{q-1}$, span one and the same q-dimensional subspace of the n-dimensional space in which $\underline{K}$ and $\underline{M}$ are defined.

It is seen from Eq. (4.8) that, when q is equal to n, the eigenvalues of the matrices $\underline{A}$ and $\underline{T}_q$ are exactly the same and are equal to the reciprocal of the respective eigenvalues of Eq. (4.1). In Appendix A, it is shown that even when q is much smaller than n, the eigensolution of the matrix $\underline{T}_q$ yields good approximations to some of the smallest eigenvalues of Eq. (4.1). However, the accuracy of the approximations must be verified by performing a Sturm sequence check and by computing the physical error norms (see Section 3.2, Eqs. (3.7) and (3.8)).

Accuracy of the predicted eigenpair approximations

The accuracy of the eigenpair approximations obtained using the Lanczos method, when the eigenvalues of Eq. (4.1) are not discrete and well separated as assumed in Appendix A is considered next. It is shown that the accuracy is largely dependent on the relative spreading in the eigenvalues of Eq. (4.1) and on the number of vectors, q, that are generated using the method. It has been shown in Appendix A

that the set of vectors $\{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_q\}$ obtained using the Lanczos method and the set of vectors $\{\underline{x}_1, \underline{K}^{-1}\,\underline{M}\,\underline{x}_1, (\underline{K}^{-1}\,\underline{M})^2\,\underline{x}_1 \ldots, (\underline{K}^{-1}\,\underline{M})^{q-1}\underline{x}_1\}$ span identically the same subspace of the operators $\underline{K}$ and $\underline{M}$. Therefore, the same predictions for the eigenvalues of Eq. (4.1) are obtained from the Ritz analyses using these two sets of vectors. In the following, employing the second set of vectors, namely $\underline{X} \equiv \{\underline{x}_1, \underline{K}^{-1}\,\underline{M}\,\underline{x}_1, (\underline{K}^{-1}\,\underline{M})^2\,\underline{x}_1, \ldots, (\underline{K}^{-1}\underline{M})^{q-1}\,\underline{x}_1\}$, we derive rough estimates for the accuracy of the eigenpair approximations obtained using the Lanczos method.

Let the starting vector $\underline{x}_1$ be expressed as

$$\underline{x}_1 = \sum_{i=1}^{n} C_i\,\underline{\phi}_i \tag{4.10}$$

where $\underline{\phi}_i$ are the eigenvectors of Eq. (4.1). Then,

$$(\underline{K}^{-1}\,\underline{M})^j\underline{x}_1 = \sum_{i=1}^{n} C_i\,\frac{1}{\lambda_i^j}\,\underline{\phi}_i \tag{4.11}$$

Hence, the matrix $\underline{X}$ consisting of the vectors $\{\underline{x}_1, \underline{K}^{-1}\,\underline{M}\,\underline{x}_1, (\underline{K}^{-1}\,\underline{M})^2\,\underline{x}_1, \ldots, (\underline{K}^{-1}\,\underline{M})^{q-1}\,\underline{x}_1\}$ can be written as:

$$\underline{X} = \underline{\phi}\,\underline{Z} \tag{4.12}$$

where

$$\underline{Z} = \begin{bmatrix} C_1 & C_1/\lambda_1 & \cdots & C_1/\lambda_1^{q-1} \\ C_2 & C_2/\lambda_2 & & C_2/\lambda_2^{q-1} \\ \vdots & \vdots & & \vdots \\ C_n & C_n/\lambda_n & \cdots & C_n/\lambda_n^{q-1} \end{bmatrix} \tag{4.13}$$

From the Ritz analysis employing the vectors in $\underline{X}$, best approximations to the eigenvectors that can be extracted from the subspace spanned by the vectors in $\underline{X}$ are obtained; in other words, best approximations to the unit coordinate vectors $\underline{e}_i$ are obtained from the Ritz analysis employing the vectors in $\underline{Z}$.

We can eliminate the off-diagonal terms in the first q rows of the matrix $\underline{Z}$, by combining together appropriate multiples of the columns in the matrix $\underline{Z}$, in Eq. (4.13); i.e.,

$$\underline{Z} = \begin{bmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2(\kappa_2-\kappa_1) & & 0 \\ 0 & 0 & & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & C_q \prod_{i=1}^{q-1} (\kappa_q-\kappa_i) \\ C_{q+1}\prod_{i=2}^{q}\frac{(\kappa_i-\kappa_{q+1})}{(\kappa_i-\kappa_1)} & C_{q+1}(\kappa_{q+1}-\kappa_1)\prod_{i=3}^{q}\frac{(\kappa_i-\kappa_{q+1})}{(\kappa_i-\kappa_2)} & \cdots & C_{q+1}\prod_{i=1}^{q-1}(\kappa_{q+1}-\kappa_i) \\ \vdots & \vdots & & \vdots \\ C_n\prod_{i=2}^{q}\frac{(\kappa_i-\kappa_n)}{(\kappa_i-\kappa_1)} & C_n(\kappa_n-\kappa_1)\prod_{i=3}^{q}\frac{(\kappa_i-\kappa_n)}{(\kappa_i-\kappa_2)} & \cdots & C_n\prod_{i=1}^{q-1}(\kappa_n-\kappa_i) \end{bmatrix} \quad (4.14)$$

where $\kappa_i = 1/\lambda_i$ and $\kappa_1 > \kappa_2 > \kappa_3 \ldots > \kappa_n$. The error in the approximation of the $i^{th}$ column of the above transformed $\underline{Z}$ matrix in Eq. (4.14) to the unit coordinate vector $\underline{e}_i$ is

$$\varepsilon_i = \| \underline{Z}_i - \underline{e}_i \| = \left\{ \sum_{j=q+1}^{n} \left[ \frac{C_j}{C_i} \prod_{k=1}^{i-1} \frac{(\kappa_j - \kappa_k)}{(\kappa_i - \kappa_k)} \prod_{\ell=i+1}^{q} \frac{(\kappa_\ell - \kappa_j)}{(\kappa_\ell - \kappa_i)} \right]^2 \right\}^{1/2}$$

$$= \left\{ \sum_{j=q+1}^{n} \left[ \frac{C_j}{C_i} \prod_{k=1}^{i-1} \frac{(1 - \lambda_k/\lambda_j)}{(1-\lambda_k/\lambda_i)} \prod_{\ell=i+1}^{q} \frac{(1 - \lambda_\ell/\lambda_j)}{(\lambda_\ell/\lambda_i - 1)} \right]^2 \right\}^{1/2} \qquad (4.15)$$

It is emphasized again that the terms $\varepsilon_i$ are not error bounds on the actual eigenvector predictions, because the actual eigenpair approximations are obtained from the Ritz analysis employing the vectors in $\underline{X}$. However, the following conclusions governing the accuracy of the predicted eigenpair approximations are deduced from Eq. (4.15).

It follows from Eq. (4.15) that the accuracy of the eigenpair approximation depends to a large extent on the spreading of the eigenvalues, $\lambda_j$, of Eq. (4.1): the more the spreading of the eigenvalues $\lambda_j$, the more accurate are the predictions obtained using the Lanczos method.

The accuracy of the approximation improves as the value of q, namely the number of vectors that are generated using the Lanczos method is increased. If q is equal to n or if the components of the eigenvectors $\underline{\phi}_{q+1}, \ldots, \underline{\phi}_n$, are not present in $\underline{x}_1$, i.e. $C_i = 0$, $q < i \leq n$, the smallest q eigenvalues and associated vectors are predicted exactly.

The accuracy of the predicted eigenpair approximations depends to a great deal on the spectral content of the starting vector $\underline{x}_1$. If the vector $\underline{x}_1$ is orthogonal to some required eigenvector $\underline{\phi}_i$, i.e. $C_i = 0$, the eigenvector $\underline{\phi}_i$ and the corresponding eigenvalue $\lambda_i$ are missed in the predicted eigensolution.

Also, it is seen from Eq. (4.15) that the components of the eigenvectors corresponding to almost equal eigenvalues undergo virtually identical magnifications and hence there is in the generated vectors a poor resolution of the eigenvectors corresponding to almost equal eigenvalues.

Combination of the Lanczos method and the subspace iteration method

The Lanczos method as presented in Section 4.3 is a one-step, approximate eigensolution method for the calculation of the smallest eigenvalues and associated vectors of the generalized eigenproblem. When the solution for the smallest p eigenvalues are required, in order to account for the possibility of some of the required eigenvalues being clustered together, the number of vectors generated, q, must be significantly larger than p. Since the Lanczos method is not an iterative method, the analyst has very little control on the accuracy of the calculated eigensolution other than to choose a large value for q.

On the other hand, the subspace iteration method calculates the eigenpairs to the accuracy specified by the analyst and therefore requires iteration. For rapid convergence in the subspace iteration, the starting vectors need only span a subspace that is close to the subspace spanned by the required eigenvectors; i.e. the individual iteration vectors need not be good approximations to the required eigenvectors. Thus, based on the characteristics of the Lanczos method and the subspace iteration method, we propose that the vectors generated using the Lanczos method be employed as starting vectors for the subspace iteration. The cost-effectiveness of the solution obtained using such a combination of the two schemes depends on the value of q, namely the number of vectors that are generated using the Lanczos method and later used in the subspace iteration.

## 4.4 Some Computational Aspects of the Lanczos Method

The Lanczos method exhibits very appealing characteristics based on theoretical considerations alone, but has some shortcomings when implemented on a computer; for example, round-off errors can cause a loss of orthogonality between the generated vectors and a poor choice for the starting vector leads to a break-down of the recurrence algorithm in Eqs. (4.3) to (4.7). In this section such problems are considered.

In the following, the application of the Lanczos method as a scheme for generating the starting vectors for the subspace iteration is only considered.

In ref. [30], Paige considered several computational variants of the Lanczos method. In essence, different ways of computing the factors $\alpha_i$ and $\beta_i$ in Eqs. (4.3) to (4.7) were outlined. Based on round-off error analyses of these different schemes, the algorithm given in Eqs. (4.3) to (4.7) was recommended by Paige as the best scheme.

### Loss of orthogonality between the generated vectors

Theoretically, the vectors $\underline{x}_i$, $i = 1, \ldots, q$ generated using Eqs. (4.3) to (4.7) form an $\underline{M}$-orthonormal basis. However, in practice the computed vectors are in general not orthogonal because of round-off errors. The loss of orthogonality implies that the generated vectors $\underline{x}_i$ do not satisfy the $\underline{M}$-orthonormality; i.e. $\underline{x}_i^T \underline{M} \underline{x}_j \neq \delta_{ij}$, where $\delta_{ij}$ is the Kronecker delta. Also, the matrix $\underline{T}_q$ in Eq. (4.9) consisting of the elements $\alpha_i$'s and $\beta_i$'s does not satisfy the relation in Eq. (4.8);

i.e. $\underline{T}_q \neq \underline{X}^T \underline{M} \underline{K}^{-1} \underline{M} \underline{X}$. Let us suppose that in spite of the loss of orthogonality, q vectors are generated using the Lanczos algorithm in Eqs. (4.3) to (4.7) and that an eigensolution of the matrix $T_q$ in Eq. (4.9) is performed. In the previous section it was pointed out that the components of the eigenvectors associated with the smallest eigenvalues are eliminated one after the other from the generated vectors. However, round-off errors introduce back the components of these eigenvectors into the generated vectors $\underline{x}_{i+1}$, as i is increased. The inverse iteration in Eq. (4.3) amplifies these components rapidly and hence the set of generated vectors can become linearly dependent. In such case, it is noted that the one-to-one correspondence between the eigenvalues of $\underline{T}_q$ and the eigenvalues of Eq. (4.1) no longer exists, when q is equal to n. Also, the recurrence algorithm in Eqs. (4.3) to (4.7) can be continued for $q > n$.

We have included the following reorthogonalization step in the implementation of the Lanczos method for the generation of the starting vectors for subspace iteration: the vector $\underline{x}_{i+1}$ obtained in Eq. (4.7) is orthogonalized to all previously computed vectors $\underline{x}_j$, $j = 1, \ldots, i$; i.e. for $j = 1, \ldots, i$, compute

$$\delta_j = \underline{x}_j^T \underline{M} \underline{x}_{i+1} \tag{4.16}$$

$$\underline{x}_{i+1} \leftarrow \underline{x}_{i+1} - \delta_j \underline{x}_j \tag{4.17}$$

and the vector $\underline{x}_{i+1}$ finally obtained, is normalized with respect to the matrix $\underline{M}$. Theoretically, the factors $\delta_j$ should be small. Hence

if the value of any $\delta_j$ is greater than $10^{-3}$, we discard the generated vector $\underline{x}_{i+1}$. Instead a new vector $\underline{x}_{i+1}$ is established and the Lanczos algorithm is continued. The procedure used to establish the new vector is described later. Also, in the first subspace iteration, the projection of the matrices $\underline{K}$ and $\underline{M}$ (in Eq. (4.1)) onto the subspace spanned by the vectors generated using the Lanczos method is computed. If the generated vectors become linearly dependent, the projection of the mass matrix will not be positive definite. During the eigensolution of the operator projections, if the projection of the matrix $\underline{M}$ is found to be non-positive definite the solution is terminated.

Generation of a null vector

Another important consideration is that a null vector $\underline{x}_{i+1}$, is theoretically generated by the Lanczos algorithm, if the starting vector $\underline{x}_1$ lies in an i-dimensional subspace of the n-dimensional space formed by the operators $\underline{K}$ and $\underline{M}$. In other words, if $\underline{x}_1$ has components of only i eigenvectors of Eq. (4.1), the generated vector $\underline{x}_{i+1}$ is theoretically a null vector. Therefore it is necessary to test the norm of the generated vector $\underline{\tilde{x}}_{i+1}$, namely $\beta_{i+1}$. We have employed the following simple strategy. Since the vector $\underline{\bar{x}}_{i+1}$ is obtained from the vector $\underline{x}_i$ using inverse iteration (see Eq. (4.3)), the vector $\underline{\bar{x}}_{i+1}$ cannot be a null vector. We compute $\gamma_{i+1}$, the norm of the vector $\underline{\bar{x}}_{i+1}$ equal to $\{\underline{\bar{x}}_{i+1}^T \underline{M} \underline{\bar{x}}_{i+1}\}^{1/2}$, and whenever the ratio $\beta_{i+1}/\gamma_{i+1}$ is smaller than $10^{-4}$, the computed vector $\underline{x}_{i+1}$ is discarded. It should be noted here that from Eqs. (4.3) to (4.7), we have

$$\gamma_{i+1} = \{\alpha_i^2 + \beta_{i+1}^2 + \beta_i^2\}^{1/2} \qquad (4.18)$$

Whenever the ratio $\beta_{i+1}/\gamma_{i+1}$ is small, severe cancellation can have occurred in the generation of the vector $\underline{\tilde{x}}_{i+1}$ from the vector $\underline{\bar{x}}_{i+1}$ in Eq. (4.5).

When the Lanczos method is applied to indefinite systems, for example, if in Eq. (4.1) the matrix $\underline{K}$ is replaced by the matrix $\underline{K} - \mu\underline{M}(\mu \geq \lambda_1)$, another problem arises. If the value of $\mu$ is very close to one of the eigenvalues $\lambda_j$ of Eq. (4.1), then regardless of the vector $\underline{x}_i$, the vectors $\underline{\bar{x}}_{i+1}$, $\underline{\bar{x}}_{i+2}$, etc. in Eq. (4.3) always become the eigenvector $\underline{\phi}_j$. This phenomenon is caused by finite precision arithmetic. The tests on the factors $\delta_j$ in Eq. (4.16) and on the factors $\beta_{i+1}$ in Eq. (4.6), guard against this failure.

The restart vector

In the generation of starting vectors for the subspace iteration using the Lanczos method, it is possible that a generated vector $\underline{x}_{i+1}$ does not satisfy the test on the factors $\delta_j$ in the Eq. (4.16) or the test on the factors $\beta_{i+1}$ in Eq. (4.6); in which case, the generated vector $\underline{x}_{i+1}$ is discarded and it is necessary to restart the recurrence algorithms in Eqs. (4.3) to (4.7) using a new vector $\underline{x}_{i+1}$. For this purpose, we always generate first the "standard" starting subspace; i.e. the degree of freedom numbers with the smallest values of $k_{ii}/m_{ii}$ are always identified (see Section 4.2). In the Lanczos algorithm, when a vector $\underline{x}_{i+1}$ is discarded, the $(i+1)^{st}$ vector generated by the "standard" procedure is used as vector $\underline{x}_{i+1}$. The new vector $\underline{x}_{i+1}$ is orthogonalized

to all vectors $\underline{x}_j$, j=1, ..., i, and finally is normalized with respect to the matrix $\underline{M}$. Then, using $\beta_{i+1}$ equal to zero, the recurrence algorithm in Eqs. (4.3) to (4.7) is continued.

Next, the effect of introducing a new vector $\underline{x}_{i+1}$ on the accuracy of the eigenvalue predictions obtained using the Lanczos method, is considered. Let us assume that the vectors $\underline{x}_j$, j=1, ..., i, have all been obtained from the initial starting vector $\underline{x}_1$. Assume that the subspace spanned by the vectors $\underline{x}_j$ contains the k eigenvectors $\underline{\phi}_1$,..., $\underline{\phi}_k$, $0 \leq k \leq i$. In other words, assume that the Ritz analysis using the i vectors $\underline{x}_j$ yields the k eigenvectors $\underline{\phi}_1$, ..., $\underline{\phi}_k$. When a generated vector $\underline{x}_{i+1}$ is discarded and a new vector $\underline{x}_{i+1}$ is established, the new vector $\underline{x}_{i+1}$ is orthogonalized to all vectors $\underline{x}_j$, j=1, ..., i. It follows that the components of the k eigenvectors $\underline{\phi}_1$, ..., $\underline{\phi}_k$ that may be present in the new vector $\underline{x}_{i+1}$, are removed from the vector $\underline{x}_{i+1}$ by the orthogonalization process. Therefore, the accuracy of the predictions for the eigenvectors $\underline{\phi}_1$, ..., $\underline{\phi}_k$ obtained from the generated vectors, is not affected by the introduction of the new vector. Also, when the starting vector $\underline{x}_1$ happens to be orthogonal to one of the required eigenvectors, the introduction of a new vector $\underline{x}_{i+1}$ in the recurrence algorithm, increases the possibility of obtaining a good approximation corresponding to the particular eigenvector in the calculated eigensolution. However, the favorable distribution of the components of the eigenvectors $\underline{\phi}_{k+1}$, ..., $\underline{\phi}_n$, that is created in the vector $\underline{x}_{i+1}$ generated using the Lanczos method (see Appendix A), is not present in the new vector $\underline{x}_{i+1}$; i.e., the accuracy of the predictions obtained for the eigenvectors $\underline{\phi}_{k+1}$, ..., $\underline{\phi}_n$, depends on the vector $\underline{x}_{i+1}$ chosen and also on the number of vectors

generated from the new vector $\underline{x}_{i+1}$, using Eqs. (4.3) to (4.7).

## 4.5 Summary of the Algorithm for Generating the Starting Subspace

The computer implementation of the algorithm to generate the starting vectors for the subspace iteration is presented in this section. A listing of a computer program that can be used to generate the "standard" or the Lanczos starting subspace is given in Appendix B.

The "standard" starting subspace is always generated, as described in Section 4.2 and consists of the following vectors:

- The first starting iteration vector is the diagonal of the matrix $\underline{M}$;
- The next (q-2) vectors each are unit coordinate vectors with the unit entries corresponding to the degrees of freedom with large mass and low stiffness values;
- the $q^{th}$ starting iteration vector is a random vector

The following algorithm is employed to generate the Lanczos starting subspace:

(i) The matrix $\underline{K}$ is factorized into $\underline{L}\ \underline{D}\ \underline{L}^T$ factors.

(ii) The starting vector $\underline{x}$ is taken as a full unit vector, i.e., $\underline{x} = [1,1,1,\ldots,1]^T$ and $\underline{x}$ is normalized with respect to the matrix $\underline{M}$ (see Ec. (4.2)) to obtain the vector $\underline{x}_1$.

(iii) The recurrence algorithm in Eqs. (4.3) to (4.7) is employed to generate each vector $\underline{x}_{i+1}(i=1,\ldots,q-2)$. Also, after computing each vector $\underline{x}_{i+1}$, the factor $\gamma_{i+1}$ in Eq. (4.18) is calculated and used to test whether the generated vector is a null vector.

(iv) Each generated vector $\underline{x}_{i+1}$ is M-orthogonalized to all vectors $\underline{x}_j$, $j=1,\ldots,i$, (see Eqs. (4.16) and (4.17)). The test on the factors $\delta_j$ in Eq. (4.16) is performed to check for the loss of orthogonality.

(v) If the generated vector is a null vector or does not satisfy the orthogonality test, the generated vector $\underline{x}_{i+1}$ is discarded. Instead, the $(i+1)^{st}$ vector generated using the above "standard" procedure is taken as the vector $\underline{x}_{i+1}$ and is orthogonalized to all the vectors $\underline{x}_j$, $j=1,\ldots,i$.

(vi) The vector $\underline{x}_q$ is a <u>random</u> vector.

The number of numerical operations that have to be performed in order to establish the Lanczos starting subspace is given in Table 4.1. It should be noted that the use of Lanczos starting subspace in the solution of out-of-core systems is not effective, particularly when a large number of eigenpairs need to be calculated. Also, when using the Lanczos procedure, it is efficient to generate 2p vectors, i.e., q equal to 2 p, where p is the number of eigenpairs required. Some numerical examples in which the above starting vectors have been used are given in Section 5.6.

TABLE 4.1 SUMMARY OF THE LANCZOS ALGORITHM FOR GENERATING THE SUBSPACE ITERATION STARTING VECTORS

| Calculation | Number of Operations | |
|---|---|---|
| | $m = m_{\underline{K}} = m_{\underline{M}}$ | $m = m_{\underline{K}}$, $m_{\underline{M}} = 0$ |
| $\underline{K}\,\bar{\underline{x}}_{i+1} = \underline{M}\,\underline{x}_i$ | $n(2m+1)$ | $n(2m+1)$ |
| $\alpha_i = \bar{\underline{x}}_{i+1}^T\,\underline{M}\,\underline{x}_i$ | $n$ | $n$ |
| $\tilde{\underline{x}}_{i+1} = \bar{\underline{x}}_{i+1}^T - \alpha_i\underline{x}_i - \beta_i\underline{x}_{i-1}$ | $2n$ | $2n$ |
| $\beta_{i+1}^2 = \tilde{\underline{x}}_{i+1}^T\,\underline{M}\,\tilde{\underline{x}}_{i+1}$ | $n(2m+2)$ | $2n$ |
| $\underline{x}_{i+1} = \tilde{\underline{x}}_{i+1}/\beta_{i+1}$ | $n$ | $n$ |
| For $j = 1, \ldots, i$<br>$\delta_j = \underline{x}_j^T(\underline{M}\,\underline{x}_{i+1})$<br>$\underline{x}_{i+1} \leftarrow \underline{x}_{i+1} - \delta_j\,\underline{x}_j$ | $2ni$ | $2ni$ |
| $\gamma_{i+1}^2 = \underline{x}_{i+1}^T\,\underline{M}\,\underline{x}_{i+1}$ | $n(2m+2)$ | $2n$ |
| $\underline{x}_{i+1} \leftarrow \underline{x}_{i+1}/\gamma_{i+1}$ | $n$ | $n$ |
| $\underline{M}\,\underline{x}_{i+1} \leftarrow \underline{M}\,\underline{x}_{i+1}/\gamma_{i+1}$ | $n$ | $n$ |
| Total for $i = 1, \ldots, q-2$ | $(6mn + nq + 10n)(q-2)$ | $(2mn + nq + 10n)(q-2)$ |

# 5. AN ACCELERATED SUBSPACE ITERATION METHOD

## 5.1 Introduction

In Chapter 3, a discussion of the basic subspace iteration method was presented. It was pointed out that modifications to improve the effectiveness of the basic scheme are necessary, particularly when a large number of eigenpairs need to be calculated. In this chapter we present an accelerated subspace iteration method that has been developed in this study in which self-adaptive overrelaxation and shifting procedures are employed to accelerate the convergence in the subspace iteration.

In Section 5.2, first we compare some overrelaxation procedures that are commonly employed in iterative processes and then present an effective overrelaxation scheme that has been developed in this study. In Section 5.3, the difficulties encountered in developing shifting procedures are pointed out; then, the shifting strategy developed in this study is presented.

Due to high speed storage limitations, in the eigensolution of large-order matrices, subspace iterations with all the iteration vectors in high speed core may not be possible. In Section 5.4, we address this problem and present modifications to the subspace iteration scheme, so that the solution can be obtained by iterating at a time with a number of iteration vectors, q, that is smaller than the number of eigenpairs required, p.

In Section 5.5, the computer implementation of the accelerated subspace iteration method is presented. Some demonstrative examples are given in Section 5.6 in order to illustrate the effectiveness of the accelerated method.

## 5.2 The Overrelaxation of Iteration Vectors

Consider the solution for the lowest p eigenvalues and associated eigenvectors of the generalized eigenproblem

$$\underline{k}\,\underline{\phi} = \lambda\,\underline{M}\,\underline{\phi} \tag{5.1}$$

using the basic subspace iteration method. The eigenvalues of Eq. (5.1) are assumed to be arranged such that $\lambda_1 \leq \lambda_2 \ldots \leq \lambda_n$. To incorporate overrelaxation into the subspace iteration, the inverse iteration step in Eq. (3.1) is modified as follows

$$\overline{\underline{X}}_{k+1} = \underline{K}^{-1}\,\underline{M}[\underline{X}_k\,\underline{\beta} - \underline{X}_{k-1}\,\underline{\alpha}] \tag{5.2}$$

where $\underline{\alpha}$ and $\underline{\beta}$ are diagonal matrices, with the overrelaxation factors $\alpha_i$ and $\beta_i$ $(i = 1,\ldots,q)$ on the diagonals.

### Overrelaxation using empirical constants

A simple way of selecting the factors $\alpha_i$ and $\beta_i$ is to assign some numerical values to these factors. Equation (5.2) was incorporated into the subspace iteration method and by assigning different values for the factors $\alpha_i$ in the range $0 < \alpha_i < \frac{1}{2}$ and with $\beta_i = 1$, we found that the number of iterations required for convergence was reduced in several cases. Also, we inferred from these analyses that the effectiveness of the overrelaxation scheme depends on the values chosen for the overrelaxation factors; i.e., the optimal values for the overrelaxation factors are very much problem dependent.

### Overrelaxation using Chebyshev iteration

An overrelaxation scheme based on Chebyshev iteration was

first proposed Rutishauser [27] and was later successfully implemented into subspace iteration by Yamamoto and Ohtsubo [34]. In the scheme, the relative spread between the eigenvalues of Eq. (5.1) is implicitly modified so that the components of the eigenvectors associated with the larger eigenvalues are reduced faster in the iteration than using the basic subspace iteration.

In the basic subspace iteration, the reduction of the components of the eigenvectors $\underline{\phi}_i$, $i = q + 1,\ldots,n$, that are present in the starting iteration vectors is realized by simultaneous inverse iteration in Eq. (3.1), i.e.,

$$\underline{\bar{X}}_{k+1} = (\underline{K}^{-1}\underline{M})^k \underline{X}_1 \tag{5.3}$$

The k inverse iterations employed to compute $\underline{\bar{X}}_{k+1}$ from $\underline{X}_1$ in Eq. (5.3) can be considered as a special case of

$$\underline{\bar{X}}_{k+1} = P_k(\underline{K}^{-1}\underline{M})\ \underline{X}_1 \tag{5.4}$$

where $P_k(x)$ represents a $k^{th}$ degree polynomial in $x$. In other words, the component $C_i$ of the eigenvector $\underline{\phi}_i$ that is present in a starting iteration vector is changed to $C_i\lambda_i^{-k}$ in Eq. (5.3), where as in Eq. (5.4) the component $C_i$ is changed to $C_i P_k(\lambda_i^{-1})$. The underlying concept in the Chebyshev iteration is to choose an appropriate polynomial $P_k(\lambda_i^{-1})$ such that the condition $P_k(\lambda_i^{-k}) < \lambda_i^{-k}$, $i = q + 1, \ldots, n$, is satisfied.

The following is a well known theorem on Chebyshev polynomials [6]: "Of all polynomials of degree k with coefficient of $x^k$ equal to 1, the Cheybshev polynomial $T_k(x)$ of degree k (multiplied by

$1/2^{k-1}$) oscillates with minimum maximum amplitude on the interva[-] [-1,1]."

It follows from this theorem that if in Eq. (5.4) the polynomial $P_k$ $(\underline{K}^{-1}\underline{M})$ is chosen as the Chebyshev polynomial $T_k(\underline{K}^{-1}\underline{M})$, the components of the eigenvectors $\underline{\phi}_{q+1},\ldots,\underline{\phi}_n$, present in the starting iteration vectors decay faster than using the inverse iteration in Eq. (5.3). In the following, we first redefine the eigenvalues of Eq. (5.1) such that the interval $[\lambda_{q+1}, \lambda_n]$ is transformed to the interval [1, -1]. Then, using the redefined eigenvalues and also the recurrence relations satisfied by the Chebyshev polynomials, we derive the expressions for the Chebyshev iteration.

Let us define a linear function

$$F(\kappa_i) = \frac{2\kappa_i - \kappa_{q+1} - \kappa_n}{\kappa_{q+1} - \kappa_n} \tag{5.5}$$

where $\kappa_i$ is the reciprocal of the eigenvalue $\lambda_i$ of Eq. (5.1). Assuming that $\lambda_{q+1} < \lambda_n$, it follows from Eq. (5.5) that

$$F(\kappa_1) \geq \ldots \geq 1 = F(\kappa_{q+1}) \geq \ldots > F(\kappa_n) = -1 \tag{5.6}$$

It is seen from Eqs. (5.1) and (5.5) that the functional values $F(\kappa_i)$ are the eigenvalues of the following eigenproblem

$$F(\underline{K}^{-1}\underline{M})\ \underline{\phi} = F(\kappa)\ \underline{\phi} \tag{5.7}$$

where

$$F(\underline{K}^{-1}\underline{M}) = \frac{2}{\kappa_{q+1} - \kappa_n}\ \underline{K}^{-1}\ \underline{M} - \frac{\kappa_{q+1} + \kappa_n}{\kappa_{q+1} - \kappa_n}\ I$$

The Chebyshev polynomials satisfy the following recurrence relations

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \quad T_0(x) = 1 \quad T_1(x) = x \tag{5.8}$$

substituting $F(\underline{K}^{-1}\underline{M})$ for x in Eq. (5.8) and noting that $T_{k-1}(F(\underline{K}^{-1}\underline{M}))\underline{X}_1$

is equal to $\underline{X}_k$, we obtain from Eqs. (5.4) and (5.8)

$$\underline{\bar{X}}_2 = \frac{2}{\kappa_{q+1} - \kappa_n} \underline{K}^{-1} \underline{M} \underline{X}_1 - \frac{\kappa_{q+1} + \kappa_n}{\kappa_{q+1} - \kappa_n} \underline{X}_1 \tag{5.9}$$

$$\underline{\bar{X}}_{k+1} = \frac{4}{\kappa_{q+1} - \kappa_n} \underline{K}^{-1} \underline{M} \underline{X}_k - 2 \frac{\kappa_{q+1} + \kappa_n}{\kappa_{q+1} - \kappa_n} \underline{X}_k - \underline{X}_{k-1} \quad (k > 1)$$

To perform the above iteration it is necessary to obtain reasonable estimates for $\kappa_{q+1}$ and $\kappa_n$. In general, the lower limit $\kappa_n$ (equal to $1/\lambda_n$) is taken as zero. Substituting $1/\lambda_{q+1}$ for $\kappa_{q+1}$ and zero for $\kappa_n$ in Eq. (5.9) we obtain

$$\begin{aligned} \underline{\bar{X}}_2 &= 2 \lambda_{q+1} \underline{K}^{-1} \underline{M} \underline{X}_1 - \underline{X}_1 \\ \underline{\bar{X}}_{k+1} &= 4 \lambda_{q+1} \underline{K}^{-1} \underline{M} \underline{X}_k - 2 \underline{X}_k - \underline{X}_{k-1} \quad (k > 1) \end{aligned} \tag{5.10}$$

It should be noted here that to incorporate the above overrelaxation scheme into the basic subspace iteration scheme, Eqs. (3.2) to (3.5) remain unaltered and Eq. (3.1) is replaced by Eq. (5.10). We need to establish also a reliable estimate for $\lambda_{q+1}$.

Recently in ref. [38], a conceptually similar overrelaxation scheme has been suggested. The modified iteration is

$$\underline{\bar{X}}_{k+1} = \underline{K}^{-1} \underline{M} \underline{X}_k - \beta \underline{X}_k \tag{5.11}$$

It is suggested that in the $k^{th}$ subspace iteration the value of $\beta$ be chosen as the $k^{th}$ Gaussian quadrature point in the interval $0 < \beta < \frac{1}{\lambda_q}$.

To implement the overrelaxation schemes Eq. (5.10) and Eq. (5.11),

it is necessary to modify the computational procedure that is used in the basic subspace iteration method to calculate the projection of the stiffness matrix. The required modification involves significant additional numerical operations and also additional storage requirement. Also, the two overrelaxation schemes have been derived with the intention of modifying the inverse iteration scheme in order to reduce the number of iterations required for solution. But the convergence analysis of the subspace iteration method indicates that the different iteration vectors in $\underline{X}_{k+1}$ have different convergence rates. Hence, it seems appropriate to develop a scheme in which the convergence behavior of each iteration vector is studied individually and the overrelaxation is applied only when necessary. The overrelaxation scheme we have developed is based on this concept of individual vector overrelaxation and is presented next.

The overrelaxation method used

In Section 3.3 it was pointed out that in the subspace iteration the $i^{th}$ iteration vector converges linearly to the $i^{th}$ eigenvector, with an ultimate rate of convergence equal to $\lambda_i/\lambda_{q+1}$. Similarly the $i^{th}$ eigenvalue iterate converges linearly to $\lambda_i$. For such a linearly convergent process the successive asymptotic errors in the approximations form a geometric progression; i.e., if we define $\lambda_i^{(k)}$ as the value of the $i^{th}$ eigenvalue iterate in the $k^{th}$ subspace iteration, the linear convergence of the iterates implies

$$\frac{\lambda_i^{(k+1)} - \lambda_i}{\lambda_i^{(k)} - \lambda_i} \doteq \frac{\lambda_i^{(k)} - \lambda_i}{\lambda_i^{(k-1)} - \lambda_i} \doteq \quad \ldots \quad \doteq r_i \tag{5.12}$$

where $r_i$ is the rate of convergence of the $i^{th}$ eigenvalue iterate to the $i^{th}$ eigenvalue $\lambda_i$.

Consider the first two terms in Eq. (5.12)

$$\frac{\lambda_i^{(k+1)} - \lambda_i}{\lambda_i^{(k)} - \lambda_i} \doteq \frac{\lambda_i^{(k)} - \lambda_i}{\lambda_i^{(k-1)} - \lambda_i} \quad .$$

Simplifying the above terms the following approximate relation is obtained

$$\lambda_i \doteq \frac{\lambda_i^{(k+1)}\,\lambda_i^{(k-1)} - \lambda_i^{(k)}\,\lambda_i^{(k)}}{\lambda_i^{(k+1)} - 2\lambda_i^{(k)} + \lambda_i^{(k-1)}} \tag{5.13}$$

Equation (5.13) is the well-known Aitken extrapolation formula [35] and applying it to successive eigenvector approximations, we obtain

$$x_{ij} \doteq \frac{x_{ij}^{(k+1)} x_{ij}^{(k-1)} - x_{ij}^{(k)} x_{ij}^{(k)}}{x_{ij}^{(k+1)} - 2x_{ij}^{(k)} + x_{ij}^{(k-1)}} \tag{5.14}$$

where $x_{ij}^{(k)}$ is the element (j,i) of $\underline{X}_k$. To incorporate the Aitken extrapolation into subspace iteration, Eq. (3.5) is followed by Eq. (5.14) and the iteration is continued with the elements $x_{ij}$ in place of $x_{ij}^{k+1}$ [24]. A major drawback in applying Aitken's extrapolation is that apart from the current iteration matrix $\underline{X}_{k+1}$, the two preceding iteration matrices $\underline{X}_k$ and $\underline{X}_{k-1}$ are also required. Also, in Eq. (5.14) almost equal terms are subtracted in the denominator and the scheme is therefore numerically very sensitive.

Consider the following terms in Eq. (5.12)

$$\frac{\lambda_i^{(k+1)} - \lambda_i}{\lambda_i^{(k)} - \lambda_i} \doteq r_i$$

Simplifying the above expression, we obtain an improved approximation to the eigenvalue $\lambda_i$ in terms of the two successive eigenvalue iterates $\lambda_i^{(k+1)}$, $\lambda_i^{(k)}$ and the theoretical convergence rate $r_i$ of the eigenvalue iterates

$$\lambda_i \doteq \frac{1}{(1 - r_i)} \left[\lambda_i^{(k+1)} - r_i \lambda_i^{(k)}\right] \tag{5.15}$$

A similar expression for an improved eigenvector approximation in terms of two successive eigenvector iterates and the theoretical vector convergence rate (equal to $\lambda_i/\lambda_{q+1}$) is

$$x_{ij} \doteq \frac{1}{(1 - \lambda_i/\lambda_{q+1})} \left[x_{ij}^{(k+1)} - \frac{\lambda_i}{\lambda_{q+1}} x_{ij}^{(k)}\right]$$

For computational purposes the above expression is modified as follows:

$$\underline{X}_{k+1} \leftarrow \underline{X}_k + (\underline{X}_{k+1} - \underline{X}_k)\ \underline{\omega} \qquad (5.16)$$

where $\underline{\omega}$ is a diagonal matrix, with the individual overrelaxation factors $\omega_i$ equal to $1/(1 - \lambda_i/\lambda_{q+1})$, along the diagonal. It should be noted that the overrelaxation scheme that we have developed in this study, given by Eq. (5.16), is a computationally more efficient form of the Aitken extrapolation in Eq. (5.14) and is applicable to any iterative process with linear convergence like inverse iteration. To employ the overrelaxation scheme given in Eq. (5.16) it is necessary to have a reliable scheme for the calculation of the vector convergence rate, $\lambda_i/\lambda_{q+1}$. The scheme that we have employed is described next.

Assuming that some of the iteration vectors have reached their asymptotic rate of convergence and we have a reasonable approximation to the corresponding eigenvalues, our objective is to calculate an approximation to $\lambda_{q+1}$, so that Eq. (5.16) can be employed to evaluate improved eigenvector approximations. The approximation to $\lambda_{q+1}$ is calculated effectively using the successive eigenvalue predictions obtained during the subspace iterations.

Considering the convergence to $\lambda_i$, let

$$r_i^{(k+1)} = \frac{\lambda_i^{(k+1)} - \lambda_i}{\lambda_i^{(k)} - \lambda_i} \tag{5.17}$$

where thus

$$\lim_{k\to\infty} r_i^{(k+1)} = \left(\frac{\lambda_i}{\lambda_{q+1}}\right)^2 \tag{5.18}$$

hen, the following expression for $r_i^{(k+1)}$ is obtained from Eqs. (5.17) and (5.12)

$$r_i^{(k+1)} = \frac{|\lambda_i^{(k+1)} - \lambda_i^{(k)}|}{|\lambda_i^{(k)} - \lambda_i^{(k-1)}|}; \quad i = 1, \ldots, p. \tag{5.19}$$

where it is assumed that $r_i^{(k+1)}$ is approximately equal to $r_i^{(k)}$. Depending on the iteration number the convergence rate estimates in Eq. (5.19) can be grossly in error, particularly due to finite precision arithmetic at later iterations. However, the estimates are fairly reliable if the following two conditions are satisfied,

$$\left|\frac{r_i^{(k+1)} - r_i^{(k)}}{r_i^{(k+1)}}\right| \leq \text{tolr} \qquad 10^{-3} \leq \text{tolc} \leq 10^{-10} \tag{5.20}$$

where tolr is typically 0.2 to 0.35, and tolc is defined in Eq. (3.6).

Assume that in iteration "$\ell$", there are some eigenvalue estimates (of the p eigenvalues to be calculated), for which the tests in Eq. (5.20) are passed, then we can calculate using each of these eigenvalue estimates an approximation to $\lambda_{q+1}$,

$$\lambda_{q+1} = \frac{\lambda_i^{(\ell+1)}}{\sqrt{r_i^{(\ell+1)}}} \tag{5.21}$$

and use as the best estimate for $\lambda_{q+1}$ the average, $\tilde{\lambda}_{q+1}$, of all estimates ever calculated in the iterations.

The values $\lambda_i^{(k+1)}$ and $\tilde{\lambda}_{q+1}$ are employed instead of $\lambda_i$ and $\lambda_{q+1}$, respectively in Eq. (5.16). The computer implementation of the overrelaxation scheme, Eq. (5.16), is presented in Section 5.5. It should be noted that the magnitude of the overrelaxation factor $\omega_i$ in Eq. (5.16) (equal to $1/(1 - \lambda_i^{(k+1)}/\tilde{\lambda}_{q+1})$) is large when the estimated vector convergence rate $\lambda_i^{(k+1)}/\tilde{\lambda}_{q+1}$ is close to unity. In such case the magnitude of the overrelaxation factor $\omega_i$ is sensitive to the accuracy of the eigenvalue estimate $\tilde{\lambda}_{q+1}$.

## 5.3 The Matrix Shifting Procedure

The basic premise of using shifting procedures in the subspace iterations is that the rates of convergence to the required eigenpairs can be increased significantly, and to such an extent as to more than compensate the added computational expense that is due to the additional triangular factorizations. It was shown earlier that, when considering the solution of only a few eigenpairs, for small-banded systems the determinant search algorithm is more effective than the basic subspace iteration method [1], and that for large-banded systems the subspace iteration method is more efficient. It can, therefore, be conjectured that depending on the bandwidth of the system matrices and the number of eigenpairs to be calculated, some shifting procedure that is similar to the one employed in the determinant search method should be effective in the subspace iteration method.

Considering shifting in the subspace iterations it is most important to develop a stable and reliable solution scheme. A major difficulty in the use of shifting lies in that if a shift is on or very close to an eigenvalue, all iteration vectors immediately converge to the eigenvector corresponding to that eigenvalue. The vectors can then not be orthogonalized any more and the iteration is unstable. If the shift $\mu$ is very close to an eigenvalue, the last pivot element in the $\underline{L}\,\underline{D}\,\underline{L}^T$ factorization of the coefficient matrix $\underline{K} - \mu\,\underline{M}$ is small compared to its original value and the shift must be changed (see Section 6.3), but two serious situations can arise:

(1) If a shift is close but not very close to an eigenvalue

(which is a situation in-between the case of a shift "exactly" on an eigenvalue and the case of a shift far away from an eigenvalue), the attraction of an iteration vector to the shift may just be counterbalanced by the vector orthogonalization process. In such a case, if the convergence tolerance employed is not high enough, an iteration vector is erroneously considered to have converged to an eigenvector.

(2) Although an iteration vector may have converged already to a required eigenvector, if a shift is imposed, and this iteration vector is still included in the subspace iterations, it is possible that this iteration vector may deteriorate and suddenly converge to another eigenvector.

With due regard to these difficulties, the following shifting procedure developed in this study is a simple and stable algorithm to accelerate the convergence of the subspace iterations.

The shifting procedure used

Assume that the smallest r eigenvalue iterates have already converged; i.e., we have tolc $\leq$ tol (using Eq. (3.6)) for the approximations to the r smallest consecutive eigenvalues. The calculated r eigenvalue approximations, the estimate for $\lambda_{q+1}$ defined in Section 5.2 as $\tilde{\lambda}_{i,r1}$, and the eigenvalue iterates that are converging to the higher eigenvalues (i>r) and satisfy Eq. (3.6) with tolc $\leq 10^{-2}$ are employed to establish an appropriate algorithm for shifting in the subspace iterations.

In order that the iteration vectors continue to converge monotonically to the required p eigenvectors the shift $\mu_s$ must satisfy the following condition,

$$\mu_s - \lambda_1 < \lambda_{q+1} - \mu_s \tag{5.22}$$

which means that $\mu_s$ is in the left half of the eigenvalue spectrum $\lambda_1$ to $\lambda_{q+1}$. To satisfy Eq. (5.22) in a conservative manner, we use

$$\mu_s \leq \bar{\lambda}_1 + \frac{1}{3}(\tilde{\lambda}_{q+1} - \bar{\lambda}_1) \tag{5.23}$$

where $\bar{\lambda}_1$ is the calculated approximation to $\lambda_1$ at the current iteration.

A second requirement for continued stability of the subspace iterations is that $\mu_s$ must be relatively far away from an eigenvalue. The shift $\mu_s$ is therefore chosen to lie midway between two well-spaced eigenvalue approximations, and these iterates must have converged to a toler-

ance (tolc) equal to $10^{-10}$ in Eq. (3.6).

The criterion for shifting in the iterations is that the improvement in the convergence to the higher eigenvalues must outweigh the computational expense of performing a shift. Therefore, a shift is performed if the following condition is met:

$$\begin{bmatrix}\text{number of operations to}\\ \text{perform the shift}\end{bmatrix} + \begin{bmatrix}\text{number of operations to}\\ \text{obtain convergence}\\ \text{after shifting}\end{bmatrix} < \begin{bmatrix}\text{number of operations}\\ \text{to obtain convergence}\\ \text{without shifting}\end{bmatrix} \tag{5.24}$$

Let $\lambda_i^{(k+1)}$ be the latest estimate of $\lambda_i$ with tolc $\leq 10^{-2}$ in Eq. (3.6), where $r < i \leq p$, and let $\tilde{\lambda}_{q+1}$ be the latest estimate for $\lambda_{q+1}$; then we can estimate the convergence rate to $\lambda_i$ as $d = (\lambda_i^{(k+1)}/\tilde{\lambda}_{q+1})^2$. Also, let t be the number of additional iterations to reach convergence without shifting. We can estimate t using

$$d^t = \text{toli} \tag{5.25}$$

where toli is the increase in accuracy still to be gained in the subspace iterations; here we have toli = tol/tolc. Hence

$$t = \frac{\log\,(\text{toli})}{\log\,(d)} \tag{5.26}$$

Using the information in Eqs. (5.24) to (5.26) we can evaluate whether it is efficient to shift. Assume that a stable shift is given by $\mu_s$ (i.e. $\mu_s$ is chosen using Eq. (5.23), then the convergence rate $\bar{d}$ to $\lambda_i$ after shifting would be approximately

$$\bar{d} = \frac{(\lambda_i^{(k+1)} - \mu_s)^2}{(\tilde{\lambda}_{q+1} - \mu_s)^2} \tag{5.27}$$

Hence the number of subspace iterations required for convergence after shifting are approximately

$$\bar{t} = \frac{\log (toli)}{\log (\bar{d})} \tag{5.28}$$

The above values for t and $\bar{t}$ are calculated for all eigenvalue iterates $\lambda_i^{(k+1)}$ where $r < i \leq p$. The maximum difference between t and $\bar{t}$, given by $(t-\bar{t})_{max}$, is then used in Eq. (5.24). Thus, a shift is efficient if [see Table 3.1],

$$\begin{aligned} &\tfrac{1}{2}\, n(m^2+3m) < \{n\,(2qm + 2q^2+3q) + 18\, q^3\}\,(t-\bar{t})_{max} \text{ (lumped mass matrix)} \\ &\tfrac{1}{2}\, n(m^2+5m) < \{n\,(4qm + 2q^2+3q) + 18q^3\}\,(t-\bar{t})_{max} \text{ (banded mass matrix)} \end{aligned} \tag{5.29}$$

where m is the average bandwidth of the stiffness matrix $\underline{K}$ in Eq. (5.1), $m_{\underline{M}} = 0$ for a lumped mass idealization and $m_{\underline{M}} = m$ for a consistent mass idealization. It is seen from Eq. (5.29) that when m is small and q is large, the number of numerical operations required to perform a shift is considerably smaller than the number of numerical operations required in the subspace iterations. Hence for systems with small bandwidth, it is advantageous to perform several matrix shifts as in the case of a determinant search solution. But when m is large and q is small, the operations required in a triangular factorization are more than the

operations required in a subspace iteration (see Fig. 1, ref. [8]), particularly when the matrices $\underline{K}$ and $\underline{M}$ are stored out-of-core.

Computational savings in the subspace iterations

In the operation count to arrive at Eq. (5.29) it is assumed that all $q$ iteration vectors are included in the complete subspace iterations. Equation (5.29) is modified if some of the iterates, say the first $r$ of the $q$ iterates, are accurately approximating the respective eigenpairs; i.e., in Eq. (3.6) $tolc \leq 10^{-10}$, for $i = 1, \ldots, r$. In such case we accept the iteration values as fully converged and corresponding to these converged iteration vectors, we dispense with the additional inverse iterations in Eq. (3.1) and the calculation of the new iteration vectros in Eq. (3.5) [23]. Also the rxr submatrices of the matrices $\underline{K}_{k+1}$ and $\underline{M}_{k+1}$ in Eqs. (3.2) and (3.3) corresponding to the $r$ fully converged iteration vectors, namely $\overline{X}_i^T \underline{K} \overline{X}_j$, $\overline{X}_i^T \underline{M} \overline{X}_j$, $i = 1,\ldots,r$ and $j = 1,\ldots,i$, are not calculated. Instead the submatrix of the stiffness matrix projection is set to a diagonal matrix with the shifted eigenvalues, $\lambda_i - \mu_s$, $(i = 1,\ldots,r)$, along the diagonal (where $\mu_s$ is the value of the current shift) and the submatrix of the mass matrix projection is set to the identity matrix of order $r$. Hence, the fully converged iteration vectors do not undergo any change in the following subspace iterations and the vectors are retained in the iterations only for orthogonalizing the $(r+1)^{st}$ to $q^{th}$ iteration vectors to these converged vectors. The above procedure leads to a considerable amount of computational savings, particularly when $q$ is large and also safeguards against a possible loss of accuracy in the fully converged iteration vectors during the following subspace iterations at new shifts.

## 5.4 The Iteration Procedure for the Case $q < p$

In the previous sections we dealt with the acceleration of the basic subspace iteration scheme by shifting and overrelaxation when throughout the solution the number of iteration vectors is larger than p. However, in some cases, when p is large, the effectiveness of the solution is increased if the required p eigenpairs are calculated with a number of iteration vectors, q, smaller than p. Basically, two advantages can arise in such solution. Firstly, the required high speed core storage decreases and, secondly, depending on the spread in the eigenspectrum, unnecessary orthogonalizations of iteration vectors to already calculated eigenvectors are automatically avoided. For the case $q < p$, the implementation of the following additional features in the subspace iteration is required - remove converged iteration vectors, replace them with new iteration vectors, orthogonalize current iteration vectors to the converged vectors on the back-up storage and perform further matrix shifting.

In the iteration with $q < p$, orthogonalization of the current iteration vectors to some of the converged iteration vectors that have already been stored on back-up storage, is performed. It follows that the accuracy of the eigenpair approximations obtained form the current iterates depends on the accuracy of the iteration vectors that have been accepted and removed to back-up storage. Hence we remove only the fully converged iteration vectors, for which the corresponding eigenvalue iterates, satisfy Eq. (3.6) to a tolerance of $10^{-10}$. Also, in case $q < p$, the required eigenpairs cannot be calculated effectively without shifting, and since a shift should lie in-between eigenvalue iterates that have converged to

a tolerance of $10^{-10}$ (using Eq. (3.6), the decision on whether to shift is based on the eigenvalue iterates that have not yet converged to the $10^{-10}$ tolerance. Hence, in the analysis of Section 5.3, the variable $r$ is equal to the number of consecutive smallest eigenvalue iterates that all have converged to tolc $\leq 10^{-10}$.

In the iteration with $q < p$, a factor of major concern is the number of converged iteration vectors that need be removed at a time. We evaluate next different strategies for selecting this number.

(i) After all the $q$ iteration vectors have fully converged, all the $q$ iteration vectors can be simultaneously removed and subspace iterations with a set of $q$ new iteration vectors can be continued.

- Due to convergence difficulty, it is seldom efficient to compute $q$ eigenpairs with only $q$ iterates and hence the scheme of removing all $q$ iteration vectors together is not always effective.

(ii) Each iteration vector can be removed as soon as the corresponding eigenvalue iterate converges to the $10^{-10}$ tolerance.

- Whenever an iteration vector is removed and replaced by a new vector, the rate of convergence of the other iteration vectors that have been employed in the current subspace iteration is altered. But, at least four subspace iterations with one particular set of $q$ iteration vectors must be performed before the effectiveness of applying the overrelaxation and shifting procedures can be determined. Hence, the procedure of removing and replacing each iteration vector on convergence may lead to an inefficient solution in which no overrelaxation and matrix shifting could be performed.

(iii) Based on the above reasons, we conclude that an optimal strategy

is to remove the iteration vectors in bundles, say $r$ at a time where $1 \leq r \leq q$, and the several factors that need be considered in estimating the optimal value for $r$, are discussed in the following.

The advantage in assigning a large value for $r$ is that the cost per subspace iteration is reduced, when some of the iteration vectors have already fully converged. On the other hand, a small value for $r$, i.e., removal of iteration vectors in small bundles, implies that the rate of convergence of the iteration vectors that have not converged yet is improved; also the inclusion of new iteration vectors yields in the following subspace iterations, approximations to some of the next required eigenvectors. However, due to the lack of prior knowledge of the larger eigenvalues, an a priori prediction of the improvement in the rate of convergence of current iteration vectors due to the introduction of new vectors, cannot be made. Further, the replacement of iteration vectors necessitates orthogonalization of the current iteration vectors to some of the converged iteration vectors on back-up storage, in addition to the subspace iteration calculations. Such orthogonalization procedure in general involves, apart from considerable numerical operations, also significant peripheral processing. An empirical relation that accounts for the above factors (like Eq. (5.29) for matrix shifting), in order to estimate the number of vectors to be removed at a time, could not be derived and implemented in the computer.

The iteration procedure used

In the iteration procedure for the case $q < p$ described next, we

remove all the fully converged vectors from the current set of iteration vectors after the largest possible shift given by Eq. (5.23) has been applied. Assume that tolc $\leq 10^{-10}$ in Eq. (3.6) is satisfied for the r smallest consecutive eigenvalue iterates. Assume further that a shift chosen in-between the eigenvalue estimates $\bar{\lambda}_r$ and $\bar{\lambda}_{r-1}$ does not satisfy Eq. (5.23) and any shift smaller than $\bar{\lambda}_{r-1}$ is either less than or equal to the current shift or is discarded based on Eq. (5.29). In this case a shift is not performed, but the r iteration vectors that correspond to the smallest eigenvalue iterates and that consecutively all satisfy Eq. (3.6) to a tolerance of $10^{-10}$ are transferred to back-up storage and replaced with new starting iteration vectors. The effect of this procedure is to increase $\tilde{\lambda}_{q+1}$, by which the rate of convergence of the iteration vectors is improved, and also allow further shifting. Considering the next subspace iterations, it is important to assure that the iteration vectors do not converge to already calculated eigenvectors and it is effective to employ Gram-Schmidt orthogonalization. All q iteration vectors are orthogonalized prior to a subspace iteration to the eigenvectors $\underline{\phi}_i$ that have already been stored on a back-up storage and correspond to eigenvalues $\lambda_i$ with

$$| \bar{\lambda}_i - \mu_s | \leq | \lambda^* - \mu_s | \tag{5.30}$$

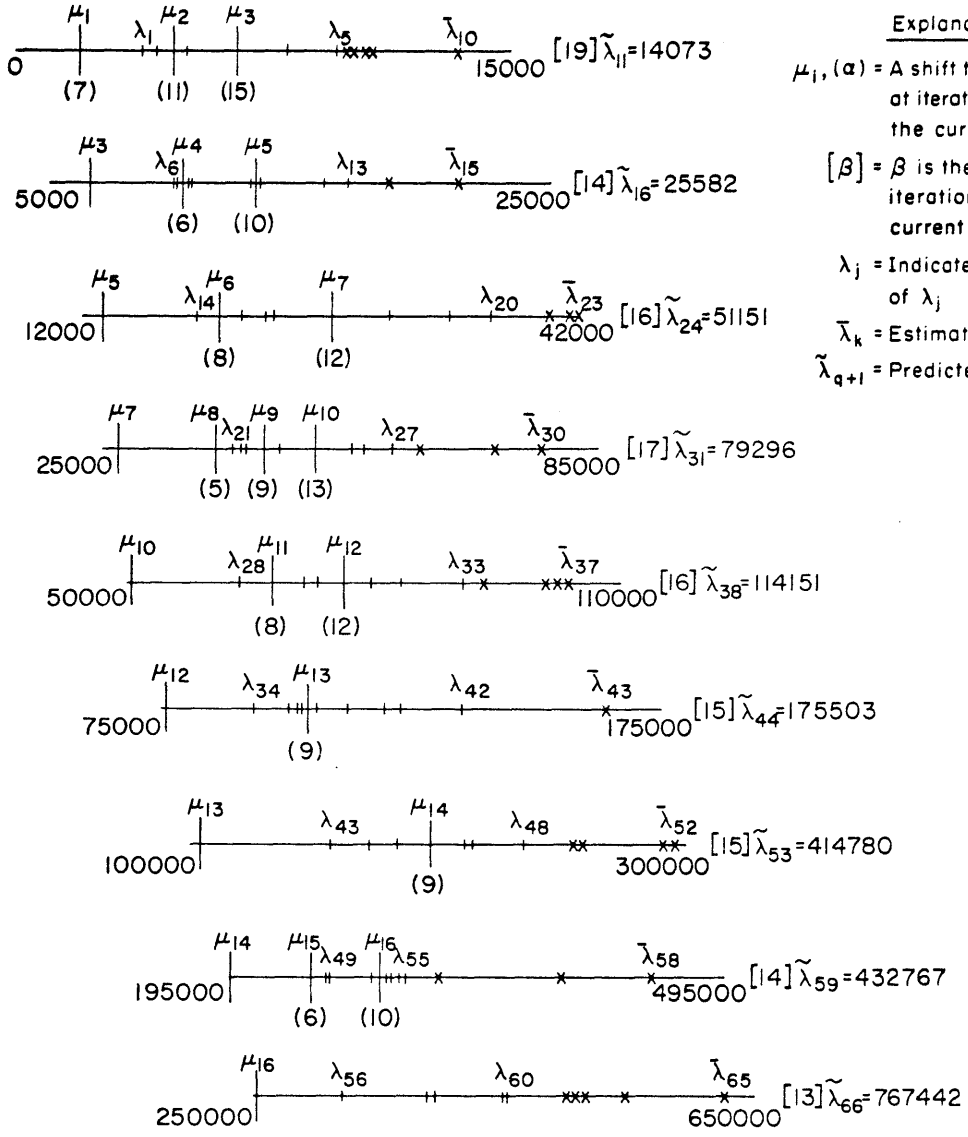
where $\lambda^*$ is the largest eigenvalue estimate calculated in the current subspace iteration and $\bar{\lambda}_i$ represents the calculated approximation to $\lambda_i$. The number of Gram-Schmidt orthogonalizations thus performed is conservative.

Considering the selection of additional shifting, we note that once accurate approximations to the eigenvectors $\underline{\phi}_1, \ldots, \underline{\phi}_j$ have been calculated and stored on back-up storage, the smallest eigenvalue to be calculated next is $\lambda_{j+1}$, and hence Eq. (5.23) is modified to

$$\mu_s \leq \bar{\lambda}_{j+1} + \frac{1}{3}(\tilde{\lambda}_{q+j+1} - \bar{\lambda}_{j+1}) \tag{5.31}$$

where $\bar{\lambda}_{j+1}$ is the calculated approximation to $\lambda_{j+1}$.

In Fig. 5.1 an example to illustrate the iteration procedure for the case $q < p$ is presented. In the example the number of eigenpairs sought, p, is equal to sixty and ten iteration vectors (q equal to ten) are used. Using the first set of ten iteration vectors, nineteen subspace iterations with three matrix shifts in-between ($\mu_1$, $\mu_2$ and $\mu_3$ in figure), are performed. As seen from the figure, the matrix shifts are performed at iterations 7, 11 and 15 respectively. After performing four subspace iterations at shift $\mu_3$, i.e. at iteration 19, it is recognized that the iterates for the eigenvalues $\lambda_1$ to $\lambda_5$ have converged to tolc $\leq 10^{-10}$ and that a shift in-between $\lambda_5$ and $\lambda_4$ violates Eq. (5.23) based on the current estimate of the eleventh eigenvalue, $\tilde{\lambda}_{11}$. Therefore, with this first set of ten iteration vectors, further matrix shifting is not allowed and instead the first five iteration vectors are accepted as fully converged and removed to back-up storage. Five new iteration vectors are generated using the "standard" procedure and along with the five iteration vectors from the first set that are still retained in-core, the next 14 subspace iterations are performed and so on. From the Fig. 5.1, it is seen that




Explanation of Symbols

$\mu_i, (\alpha)$ = A shift to $\mu_i$ was performed at iteration number $\alpha$ with the current set of vectors

$[\beta]$ = $\beta$ is the total number of iterations performed with the current set of vectors

$\lambda_j$ = Indicates accepted value of $\lambda_j$

$\bar{\lambda}_k$ = Estimate of eigenvalue $\lambda_k$

$\tilde{\lambda}_{q+1}$ = Predicted value of $\lambda_{q+1}$

FIGURE 5.1 AN EXAMPLE TO ILLUSTRATE THE ITERATION PROCEDURE FOR THE CASE $q < p$ ($q = 10$, $p = 60$)

the required sixty eigenpairs are calculated in bundles of 5, 8, 7, 7, 6, 9, 6 and 7 each from nine iteration vector sets and sixteen matrix shifts are performed in the solution.

The iteration procedure presented in this section is used effectively if $q < p$, but to avoid convergence difficulties we recommend the use of this iteration procedure also if $q$ is only slightly larger than $p$, i.e. when $q < \min. \{2p, p+8\}$.

## 5.5 Computer Implementation

The computer Program SSPACE given in ref. [1, p. 509] is an implementation of the basic subspace iteration method (see Chapter 3). The acceleration procedures to the basic subspace iteration method that have been developed in this study were implemented into the above computer program. The purpose of this section is to summarize how the solution procedures have been implemented in the program, see Fig. 5.2.

The user's manual and the listing of the computer program are given in Appendix B.

### Overrelaxation

Having calculated $\underline{X}_{k+1}$ in Eq. (3.5) an overrelaxation is performed for an iteration vector if Eq. (5.20) is satisfied. In the overrelaxation the iteration vector from the previous iteration is used from back-up storage, and we calculate

$$\underline{X}^i_{k+1} \leftarrow \underline{X}^i_k + (\underline{X}^i_{k+1} - \underline{X}^i_k)\ \omega_i \qquad (5.32)$$

where $\omega_i = 1/(1-\lambda_i^{(k+1)}/\tilde{\lambda}_{q+1})$ and the vectors in $\underline{X}_{k+1}$, for which the corresponding eigenvalue iterates have not yet converged to the $10^{-10}$ accuracy using Eq. (3.6), are written onto back-up storage.

### Shifting

Considering the matrix shifting strategy, the initial shift is zero and then $\mu_s$ is increased with each shift. Based on the consideration in Section 5.3 we assess after 4 subspace iterations at a shift whether it is effective to shift to a new position. Namely, about 4 subspace iterations are required for the iteration vectors to settle down after a shift, so that the analysis presented above is approximately applicable. Also,

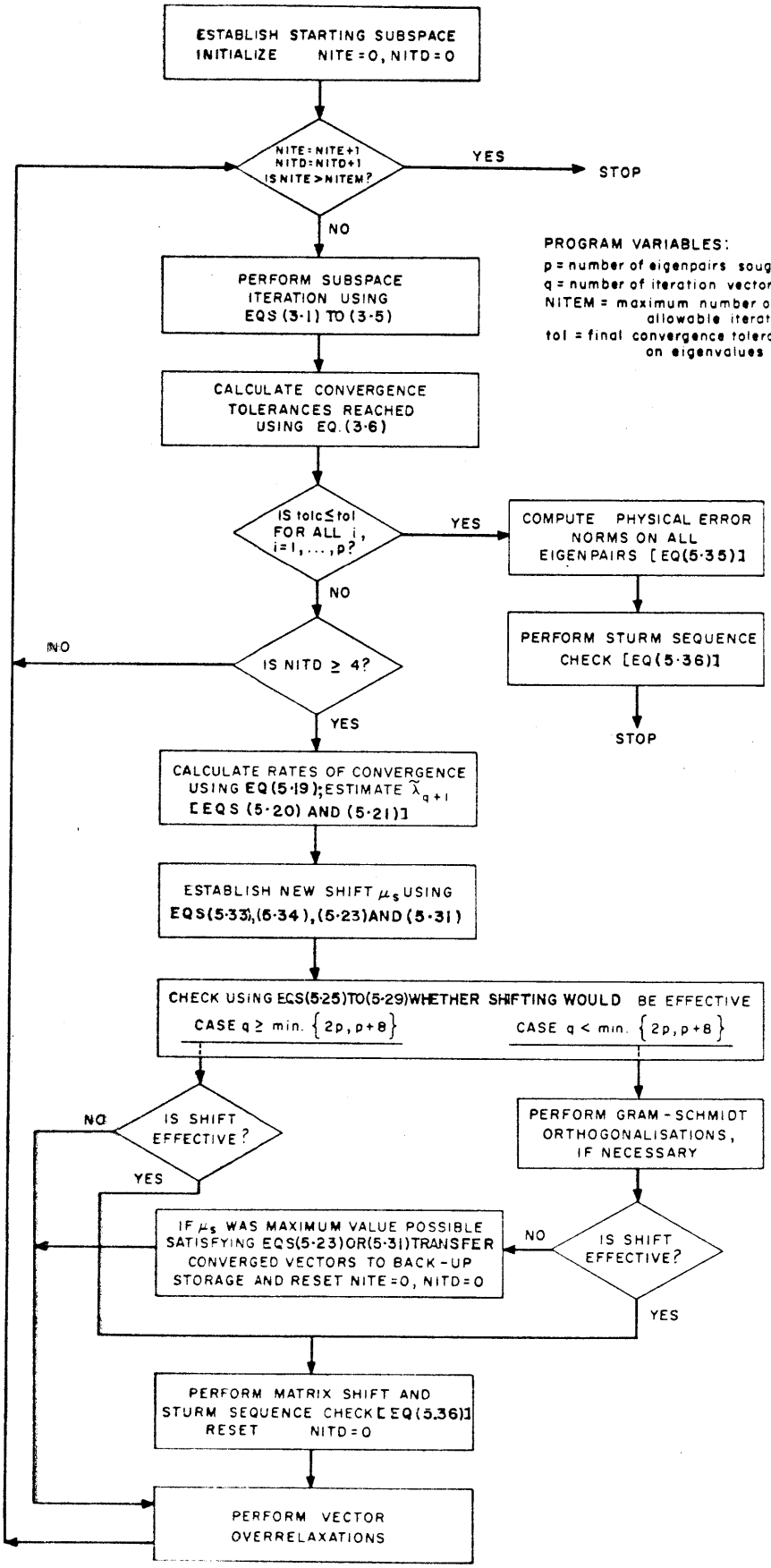



FIGURE 5.2 CALCULATIONS IN ACCELERATED SUBSPACE ITERATION

a shift is only performed if there is a saving of at least 3 subspace iterations, i.e., in addition to satisfying Eq. (5.29), we must also have $(t-\bar{t})_{max} \geq 3$.

After 4 subspace iterations have been performed at the current shift, the following procedure is employed to establish a <u>new</u> shift, $\mu_s$.

(a) Calculate the convergence rates $r_i$ of the eigenvalue iterates using Eq. (5.19) at the end of each iteration, and calculate $\tilde{\lambda}_{q+1}$ using Eqs. (5.19) to (5.21).

(b) Establish the largest allowable shift, $\mu_s$. This shift is calculated as

$$\mu_s = \frac{\bar{\lambda}_s + \bar{\lambda}_{s-1}}{2} \tag{5.33}$$

where $\bar{\lambda}_s$ is the calculated approximation to $\lambda_s$ and $\lambda_s$ is the largest eigenvalue for which all eigenvalue iterates, below and including $\lambda_s$, have converged to a tolerance of $10^{-10}$ using Eq. (3.6). Check whether this shift satisfies Eq. (5.23) (or Eq. (5.31)), and also the following condition

$$1.01\ \bar{\lambda}_{s-1} \leq \mu_s \leq 0.99\ \bar{\lambda}_s. \tag{5.34}$$

If either Eq. (5.23) (or Eq. (5.31)) or Eq. (5.34) is not satisfied, decrease s (using $s \leftarrow s-1$) until both conditions are met. It is next assessed whether shifting to $\mu_s$ is effective, if the value of $\mu_s$ thus obtained is still larger than the current shift.

(c) If only a few subspace iterations have been performed, reasonably accurate estimates for all $\lambda_m$, $s < m \leq p$ may not yet be attainable. Hence, to evaluate Eqs. (5.25) to (5.29), we use

only the eigenvalue iterates $\bar{\lambda}_m$ for which tolc $\leq 10^{-2}$. In order that shifting to $\mu_s$ be performed, Eq. (5.29) must be satisfied.

(d) If a shift is performed, use the Sturm sequence information and error estimates on the calculated eigenpair approximations to establish whether all eigenvalues between the previous shift and the new shift have been obtained [1, p. 505]. Assume that j eigenvalues have been calculated between the previous and the current shift, then the following error estimates [1, p. 413] should be small for all eigenpairs calculated

$$\varepsilon_i = \frac{\| \underline{K}\, \underline{\phi}_i^{(\ell+1)} - \lambda_i^{(\ell+1)}\, \underline{M}\, \underline{\phi}_i^{(\ell+1)} \|_2}{\| \underline{K}\, \underline{\phi}_i^{(\ell+1)} \|_2}$$

and j additional negative elements must be measured in $\underline{D}$, where

$$\underline{K} - \mu_s \underline{M} = \underline{L}\, \underline{D}\, \underline{L}^T \tag{5.36}$$

In theory, it could happen that an eigenpair has been missed. However, in practice such situation is very rare. If an eigenpair has been missed, the solution is stopped and should be repeated with a larger number of iteration vectors, q, and tighter convergence tolerance, tol.

Case q < p

Considering the case q < min {2p, p+8}, the matrix shifting strate-

gy is as described above with one additional calculation procedure. Assume that the candidate shift is discarded based on Eq. (5.29) and is the maximum value possible satisfying Eq. (5.23) (or Eq. (5.31)). In this case, all iteration vectors that correspond to the smallest eigenvalue iterates and that consecutively all satisfy Eq. (3.6) to a tolerance of $10^{-10}$ are written on back-up storage and replaced by new starting iteration vectors. Further checking on whether additional matrix shifting is effective is then performed after four more subspace iterations.

On the computer implementation of an out-of-core solution scheme

In Section 3.3, the high-speed storage requirement for the computer implementation of the basic subspace iteration method was discussed briefly. It was mentioned that apart from the storage required for the matrices $\underline{K}$ and $\underline{M}$, storage for (q+2) vectors of length n (where n is the order of the matrices) is required. When n is large the above storage requirement can be enormous, particularly when a banded matrix $\underline{M}$ is used and the maximum high-speed storage available on a particular computer system may be insufficient. In such case, it is necessary to divide the matrices $\underline{K}$ and $\underline{M}$ and the iteration matrix $\underline{X}_k$ into smaller blocks and store the matrix blocks on auxiliary storage devices. In the present day computer systems, the cost of performing input/output operations onto auxiliary storage devices is significantly more expensive compared to the cost of performing arithmetic operations on the high speed core. Hence the strategy for dividing the iteration vectors in $\underline{X}_k$ into blocks must be chosen with care in order to minimize the inevitable input/output

operations onto the peripheral devices. In the following, different schemes for dividing the iteration matrix into blocks are considered.

In the accelerated subspace iteration method, overrelaxation of the individual iteration vectors is performed and when $q < p$ also Gram-Schmidt orthogonalization of the current iteration vectors to some of the converged iteration vectors on back-up storage is performed. To perform the overrelaxation or orthogonalization of an iteration vector, it is effective if all the elements of the iteration vector are available in high speed core simultaneously. In other words, considering the iteration matrix $\underline{X}_k$ as an n x q array with each iteration vector as one column, it is effective to block the matrix $\underline{X}_k$ columnwise.

In this study, an out-of-core solution scheme in which only the matrices $\underline{K}$ and $\underline{M}$ are divided into blocks and all the iteration vectors are kept in high speed core, has been implemented [37, 41]. In this scheme, the size of the blocks of the matrix $\underline{K}$ and $\underline{M}$ is small implying that the number of blocks of the matrices is large. However, since enough storage is available to retain all the iteration vectors in high speed core, the inverse iteration in Eq. (3.1) is performed simultaneously on all the iteration vectors. It is necessary to write each iteration vector in $\underline{X}_k$ onto a scratch tape and read back the iteration vectors one after the other onto highspeed core during the calculation of the projection matrix $\underline{K}_{k+1}$ in Eq. (3.2). Since the fully converged iteration vectors are no longer included in the subspace iteration (see Section 5.3), the converged iteration vectors are not

written onto the scratch tape. Thus in this scheme, when a diagonal matrix $\underline{M}$ is used, no information is read from the auxiliary storage more than once during a subspace iteration. But when a banded matrix $\underline{M}$ is employed and the matrix $\underline{M}$ is stored out-of-core, all the blocks of the matrix $\underline{M}$ have to be read into core once in order to compute one column of the projection matrix $\underline{M}_{k+1}$.

We consider next a scheme in which the iteration matrix $\underline{X}_k$ is blocked. Assume that the matrices $\underline{K}$ and $\underline{M}$ are divided into $x$ blocks each and the matrix $\underline{X}_k$ is divided into $y$ blocks. Then, during the inverse iteration in Eq. (3.1) all the $y$ blocks of the matrix $\underline{X}_k$ have to be read (and overwritten) $x$ times. The projection calculations in Eq. (3.2) and (3.3) also involve significant peripheral processing. Hence the blocks of the iteration matrix $\underline{X}_k$ have to be read into core and written onto a scratch tape several times during each subspace iteration. Also, in this scheme all the blocks of the iteration matrix $\underline{X}_k$ have to be read into core once in order to perform the overrelaxation for a few iteration vectors which satisfy the necessary condition in Eq. (5.20). However, when a banded matrix $\underline{M}$ is employed, this scheme of dividing the iteration matrix $\underline{X}_k$ into blocks may be more effective than the scheme discussed earlier, depending on the number of iteration vectors used and on the order and bandwidth of the matrices $\underline{K}$ and $\underline{M}$.

## 5.6 Demonstrative Examples

We present in this section the analysis results of some solutions that demonstrate typical features of the accelerated subspace iteration method. Some of the solutions were obtained using the computer program given in Appendix B and some solutions, particularly all the out-of-core solutions, were obtained using the out-of-core version of the accelerated subspace iteration method that has been implemented in the finite element computer program ADINA [41].

The final convergence tolerance, tol, is equal to $10^{-6}$ for all the analyses.

For some analyses, a value larger than the default value had to be used for the variable NITEM, which determines the maximum number of subspace iterations allowed (see Appendix B).

### Solution of a 16 degree-of-freedom tridiagonal system

The eigenvalue problem summarized in Fig. 5.3 was solved with $p = 4$ and $q = 8$ using the basic subspace iteration scheme and the accelerated method. Using the basic subspace iteration method, the solution of this problem requires, theoretically, about 55 iterations to calculate the 4 smallest eigenvalues with tol = $10^{-6}$. Thus, the eigensolution requires a relatively large number of iterations and provides a good testing ground for the accelerated iteration scheme. Table 5.1 gives the actual number of iterations employed in the solutions using the basic subspace iteration method and the accelerated procedure. The table shows that the accelerated method provides a significantly more effective solution to this

$$\underline{K} = \begin{bmatrix} 110 & -10 & & & & & & & & & & & & & & \\ & 122 & -10 & & & & & & & & & & & & & \\ & & 123 & -10 & & & & & & & & & & & & \\ & & & 124 & -10 & & & & & & & & & & & \\ & & & & 125 & -10 & & & & & & & & & & \\ & & & & & 126 & -10 & & & & & & & & & \\ & & & & & & 127 & -10 & & & & & & & & \\ & & & & & & & 128 & -10 & & & & & & & \\ & & & & & & & & 129 & -10 & & & & & & \\ & & & & & & & & & 130 & -10 & & & & & \\ & & & & & & & & & & 131 & -10 & & & & \\ & & & & & & & & & & & 132 & -10 & & & \\ & & & & & & & & & & & & 133 & -10 & & \\ & & & & & & & & & & & & & 134 & -10 & \\ & & & & & & & & & & & & & & 135 & -10 \\ & & & & & & & & & & & & & & & 126 \end{bmatrix}$$

SYMMETRIC

ELEMENTS NOT SHOWN ARE ZEROES

$$\underline{M} = \underline{I}$$

EIGENVALUES:

$\lambda_1 = 102.2$

$\lambda_2 = 107.3$

$\lambda_3 = 110.6$

$\lambda_4 = 113.4$

$\lambda_5 = 115.6$

$\lambda_6 = 117.9$

$\lambda_7 = 121.1$

$\lambda_8 = 124.7$

$\lambda_9 = 128.5$

$\lambda_{10} = 132.3$

$\lambda_{11} = 135.9$

$\lambda_{12} = 139.2$

$\lambda_{13} = 142.1$

$\lambda_{14} = 144.8$

$\lambda_{15} = 147.8$

$\lambda_{16} = 151.5$

FIGURE 5.3 A 16 DEGREE-OF-FREEDOM TRIDIAGONAL SYSTEM

TABLE 5.1 ANALYSIS OF THE TRIDIAGONAL SYSTEM ($p = 4, q = 8$)

| | BASIC SCHEME | ACCELERATED SCHEME | | | |
|---|---|---|---|---|---|
| | | Standard Starting Subspace | | | Lanczos Starting Subspace |
| | | Overrelaxation Only | Shifting Only | Overrelaxation and Shifting | Overrelaxation and Shifting |
| Number of Iterations | 49 | 35 | 31 | 23 | 20 |

problem.

Analysis of a piping system

Figure 5.4 shows a computer plot of the piping system considered in this study. For this system the order (n) of the stiffness and mass matrices is 330, the mean half-bandwidth ($m_K$) of the stiffness matrix is 26 and a diagonal mass matrix was employed. The sixty smallest eigenvalues and corresponding eigenvectors were required.

Table 5.2 summarizes some relevant solution data corresponding to the various analyses performed.

Considering the solution with 68 iteration vectors, i.e., q = 68, the rate of convergence for the sixtieth eigenvalue iterate, $(\lambda_{60}/\lambda_{69})^2$, is equal to 0.3 resulting in rapid convergence using the basic scheme. Hence for the case q = 68, there is no reduction in the number of subspace iterations and in the required high speed storage using the accelerated method. However, considering the solution with q = 20 and q = 4 significantly less high speed storage is needed at no increase in central processor time. Since the average bandwidth of the stiffness matrix is small, the determinant search method is equally effective for this problem [1].

It is interesting to compare the solutions using the "standard" and the Lanczos starting subspaces. For the case q = 68, more iterations are required using the Lanczos starting vectors. However, after two iterations the first 34 eigenvalue iterates and after a total of only five iterations the smallest 45 eigenvalue iterates had converged to tolc = $10^{-10}$. Since

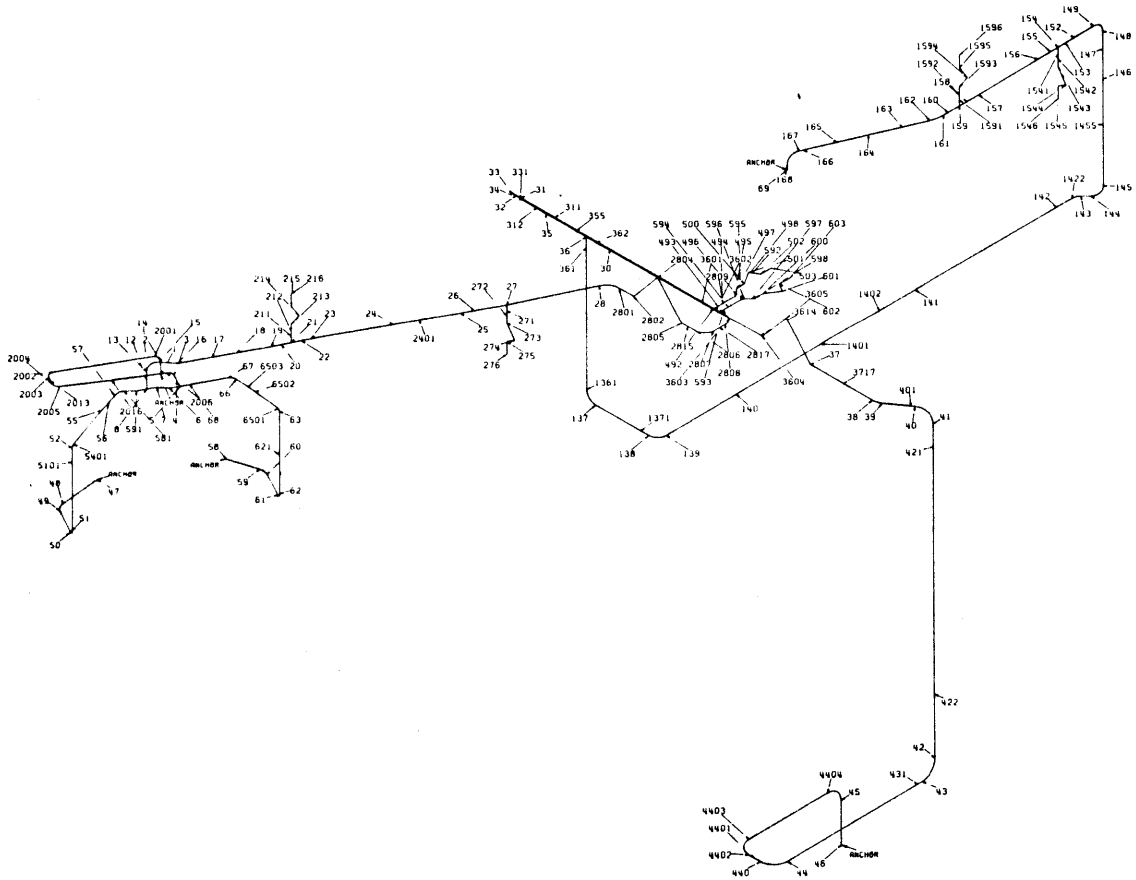



FIGURE 5.4 GEOMETRIC VIEW OF PIPING SYSTEM

Order of matrices (n) = 330; mean half bandwidth ($m_K$) = 26

TABLE 5.2 COMPARISON OF DIFFERENT SOLUTION STRATEGIES IN THE ANALYSIS OF THE PIPING SYSTEM

$n = 330$, $m_{\underline{K}} = 26$, Diagonal mass matrix was used (Computer used was CDC CYBER 175; tol = $10^{-6}$)

| | BASIC SCHEME | ACCELERATED SCHEME | | | | | DETERM. SEARCH |
|---|---|---|---|---|---|---|---|
| | | Standard Starting Subspace | | | Lanczos Starting Subspace | | |
| p/q | 60/68 | 60/68 | 60/20 | 60/4 | 60/68 | 60/100 | 60/1 |
| Total Number of Subspace Iterations | 13 | 13 | 74 | 408 | 23 | 2 | (5+6)* |
| Total High Speed Core Storage Used | 42,494 | 42,494 | 17,870+ | 11,700+ | 42,494 | 64,094 | 12,490+ |
| Solution Time, CPU (sec) | 90 | 74 | 65 | 63 | 77 | 71 | 63 |

Stiffness matrix was stored in-core for all solutions.

+ Additional secondary storage was required for storing converged iteration vectors.

* Average number of factorizations and inverse iterations per eigenpair.

TABLE 5.3 SOLUTION STEPS IN THE ANALYSIS OF THE PIPING SYSTEM (p = 60, q = 20)

| Iteration Vector Set Numbers | Eigenpairs Sought | Iterations Performed With These Vectors | Converged Trial Vectors Simultaneously Removed to Back-up Storage | Eigenpair Approximations Carried Over to Next Set | Matrix Shifts Applied at $(\frac{\lambda_i+\lambda_{i-1}}{2})$ Based on Convergence of $\lambda_k$ | | | Calculation of $\lambda_{q+1}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Iteration No. | i | k | q + 1 | Estimated Value, $\tilde{\lambda}_{q+1}$ | $\lambda_{q+1}$ |
| 1 | 1 to 20 | 1 to 17 | 1 to 15 at iteration 17 | 16 to 20 | 5 | 1 | 14 | | 24501 | |
| | | | | | 9 | 4 | 17 | 21 | 29738 | |
| | | | | | 13 | 11 | 17 | | 32847 | 40373 |
| 2 | 16 to 35 | 18 to 36 | 16 to 28 at iteration 36 | 29 to 35 | 24 | 18 | 35 | | 125604 | |
| | | | | | 28 | 22 | 35 | 36 | 118106 | |
| | | | | | 32 | 25 | 35 | | 113883 | 102360 |
| 3 | 29 to 48 | 37 to 52 | 29 to 43 at iteration 52 | 44 to 48 | 40 | 31 | 44 | | 182623 | |
| | | | | | 44 | 34 | 46 | 49 | 227682 | |
| | | | | | 48 | 41 | 46 | | 232584 | 254685 |
| 4 | 44 to 63 | 53 to 66 | 44 to 48 at iteration 66 | 49 to 63 | 61 | 49 | 60 | 64 | 658386 | 548048 |
| 5 | 49 to 68 | 67 to 73 | 49 to 60 at iteration 74 | | | | | | | |

the converged iteration vectors are no longer included in the iterations (see Section 5.3) about the same total solution times are required using the "standard" and the Lanczos starting subspaces.

There is very little difference between the total solution costs for the various analyses summarized in Table 5.2.

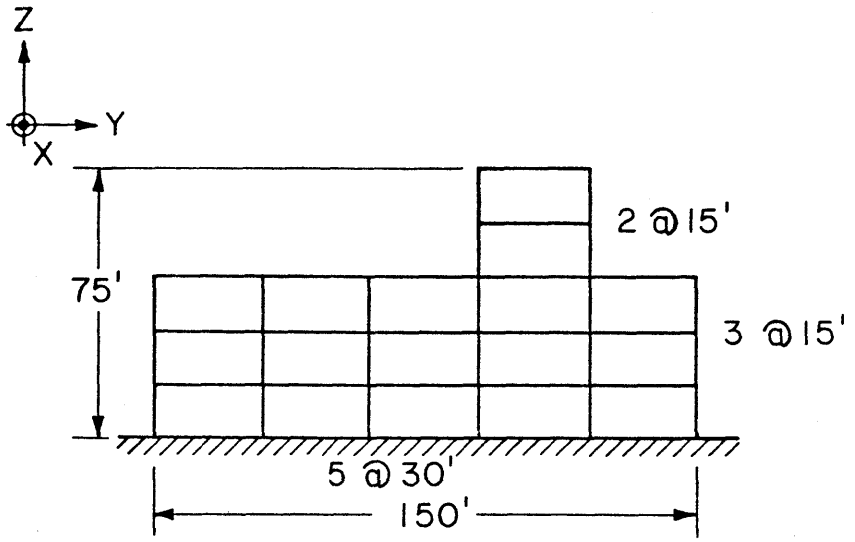
Finally, Table 5.3 summarizes the complete solution steps for the case q = 20. As seen from this table, a total of ten matrix shifts were performed in the solution and the required sixty eigenpairs were calculated in bundles of 15, 13, 15, 5 and 12, each from five iteration vector sets.

Analysis of a building frame

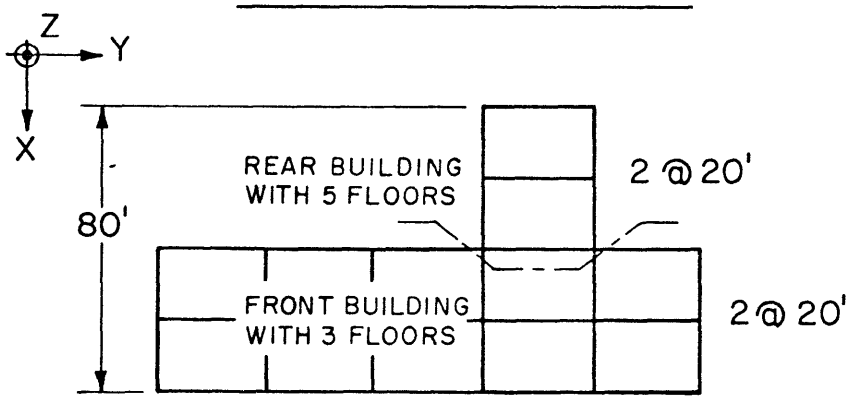
The building frame shown in Fig. 5.5 was earlier analyzed in ref. [23]. We analyzed the same structure in this study to demonstrate some important features of the accelerated subspace iteration method. For this system n = 468 and $m_{\underline{K}}$ = 91 and a diagonal mass matrix was employed. The sixty smallest eigenvalues and corresponding eigenvectors were required.

Since the stiffness matrix has a relatively large bandwidth a determinant search solution is not effective, and only subspace iteration solutions have been calculated.

Table 5.4 gives the characteristics of the solutions obtained. It is seen that the accelerated subspace iteration method yields a significantly more effective solution than the basic scheme, with the stiffness matrix in-core or out-of-core. Table 5.5 summarizes the solution steps for the




(a) ELEVATION OF BUILDING




(b) PLAN OF BUILDING

DATA:

YOUNG'S MODULUS=432000, MASS DENSITY=1.0
COLUMNS IN FRONT BUILDING $A_1=3.0, I_1=I_2=I_3=1.0$
COLUMNS IN REAR BUILDING $A_1=4.0, I_1=I_2=I_3=1.25$
ALL BEAMS INTO X-DIRECTION $A_1=2.0, I_1=I_2=I_3=0.75$
ALL BEAMS INTO Y-DIRECTION $A_1=3.0, I_1=I_2=I_3=1.0$
UNITS : FT, KIP

FIGURE 5.5 A THREE-DIMENSIONAL BUILDING FRAME
Order of matrices (n) = 468; mean half bandwidth ($m_K$) = 91

TABLE 5.4 COMPARISON OF DIFFERENT SOLUTION STRATEGIES IN THE ANALYSIS OF THE BUILDING FRAME

$n = 468$, $m_{\underline{K}} = 91$, Diagonal mass matrix was used (Computer used was CDC CYBER 175; $tol = 10^{-6}$)

| | BASIC SCHEME | | ACCELERATED SCHEME | | | | |
|---|---|---|---|---|---|---|---|
| | | | Standard Starting Subspace | | | Lanczos Starting Subspace | |
| p/q | 60/68 | 60/68 | 60/68 | 60/68 | 60/20 | 60/68 | 60/68 |
| Total Number of Subspace Iterations | 46 | 46 | 36 | 36 | 80 | 37 | 37 |
| Total High Speed Core Storage Used | 66850* | 86056+ | 66850* | 86056+ | 54808+ | 66850* | 86056+ |
| Solution Time, CPU (sec.) | 570 | 538 | 279 | 272 | 204 | 218 | 210 |
| Relative Total Solution Cost | 3.47 | 2.35 | 2.01 | 1.33 | 1.00 | 2.80 | 1.10 |

* Stiffness matrix was stored out-of-core (4 blocks).

\+ Stiffness matrix was stored in-core.

TABLE 5.5 SOLUTION STEPS IN THE ANALYSIS OF THE BUILDING FRAME ($p = 60$, $q = 20$)

| Iteration Vector Set Numbers | Eigenpairs Sought | Iterations Performed With These Vectors | Converged Trial Vectors Simultaneously Removed to Back-up Storage | Eigenpair Approximations Carried Over to Next Set | Matrix Shifts Applied at $(\frac{\lambda_i + \lambda_{i-1}}{2})$ Based on Convergence of $\lambda_k$ | | | Calculation of $\lambda_{q+1}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Iteration No. | i | k | q + 1 | Estimated Value, $\tilde{\lambda}_{q+1}$ | $\lambda_{q+1}$ |
| 1 | 1 to 20 | 1 to 17 | 1 to 12 at iteration 17 | 13 to 20 | 9 | 6 | 15 | 21 | 9.59 | |
| | | | | | 13 | 9 | 16 | | 10.01 | 11.58 |
| 2 | 13 to 32 | 18 to 37 | 13 to 27 at iteration 37 | 28 to 32 | 21 | 13 | 24 | 33 | 17.71 | |
| | | | | | 25 | 17 | 30 | | 28.38 | |
| | | | | | 29 | 24 | 32 | | 29.64 | |
| | | | | | 33 | 25 | 32 | | 30.31 | 29.44 |
| 3 | 28 to 47 | 38 to 54 | 28 to 42 at iteration 54 | 43 to 47 | 42 | 29 | 34 | 48 | 41.51 | |
| | | | | | 46 | 34 | 42 | | 55.03 | |
| | | | | | 50 | 41 | 43 | | 65.23 | 86.12 |
| 4 | 43 to 62 | 55 to 72 | 43 to 48 at iteration 72 | 49 to 62 | 58 | 43 | 48 | 63 | 87.64 | |
| | | | | | 64 | 46 | 59 | | 122.75 | |
| | | | | | 68 | 47 | 60 | | 122.81 | 122.63 |
| 5 | 49 to 60 | 73 to 80 | 49 to 60 at iteration 80 | | | | | | | |

case q = 20.

The generation of each starting vector using the Lanczos method involves one inverse iteration (see Chapter 4). When the stiffness matrix is stored out-of-core, to perform each inverse iteration the stiffness matrix blocks have to be read into high-speed core and the read operations increase the peripheral processing costs considerably. Hence, as is evident from Table 5.4, the generation of starting vectors using the Lanczos method for out-of-core solutions is not effective when a large number of vectors need be calculated in the solution. Also, from Table 5.4 it is seen that when the stiffness matrix is kept in-core, the use of accelerated subspace iteration method employing the "standard" starting subspace, with q = 20, results in the most effective solution for this eigenproblem.

Analysis of a marine probe

A finite element model that was employed to calculate the natural frequencies of a three dimensional marine probe is shown in Fig. 5.6. For this system n = 1278 and $m_K$ = 16 and a diagonal mass matrix was employed. The forty smallest eigenvalues and corresponding eigenvectors were required.

Tables 5.6 and 5.7 summarize some relevant solution data corresponding to the various analyses performed.

Considering the solution with 48 iteration vectors, i.e., q = 48, only 20 subspace iterations are required using the basic scheme. Hence for the case q = 48, there is no significant reduction in the number of subspace iterations performed using the accelerated method with the

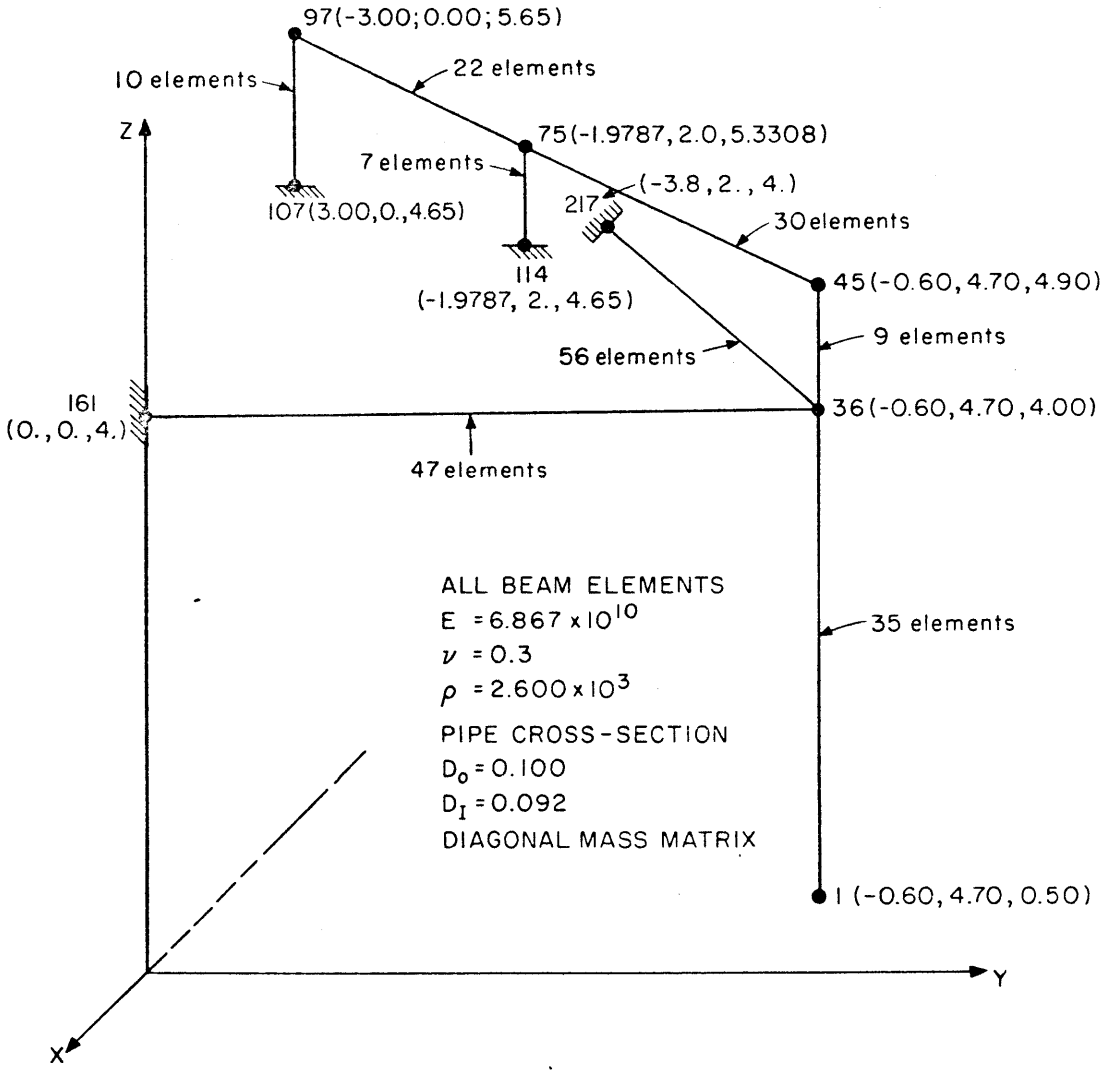



FIGURE 5.6 A THREE-DIMENSIONAL MARINE PROBE
Order of matrices (n) = 1278; mean
half bandwidth ($m_K$) = 16

TABLE 5.6 COMPARISON OF DIFFERENT SOLUTION STRATEGIES IN THE ANALYSIS OF THE MARINE PROBE

n = 1278, $m_{\underline{K}}$ = 16, Diagonal mass matrix was used (Computer used was CDC CYBER 175)

| | BASIC SCHEME | ACCELERATED SCHEME | | | | DETERM. SEARCH |
|---|---|---|---|---|---|---|
| | | Standard Starting Subspace | | | Lanczos Starting Subspace | |
| p/q | 40/48 | 40/48 | 40/20 | 40/4 | 40/48 | 40/1 |
| Total Number of Subspace Iterations | 20 | 17 | 42 | 250 | 12 | 4/5 * |
| Total High Speed Core Storage Used | 94000 | 94000 | 52059 | 30721 | 94000 | 34631 |
| Solution Time, CPU(sec) | 195 | 132 | 93 | 88 | 77 | 78 |
| Relative Total Solution Cost | 2.22 | 1.84 | 1.33 | 1.46 | 1.00 | 1.14 |

Stiffness matrix was stored in-core for all solutions.

* Average number of factorizations and inverse iterations per eigenpair.

TABLE 5.7 FREQUENCY SOLUTION TIMING INFORMATION FOR THE ANALYSIS OF THE MARINE PROBE

(Computer used was CDC Cyber 175)

| Calculation | BASIC SCHEME | ACCELERATED SCHEME | | | |
|---|---|---|---|---|---|
| | | Standard Starting Subspace | | | Lanczos Starting Subspace |
| p/q | 40/48 | 40/48 | 40/20 | 40/4 | 40/48 |
| Calculation of Starting Subspace | 0.1 | 0.1 | 0.1 | 0.1 | 7.3 |
| Initial Factorization of Stiffness Matrix | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 |
| Calculation of Projections of $\underline{K}$ and $\underline{M}$ | 97.5 | 61.6 | 46.6 | 39.8 | 36.3 |
| Eigensolution of Subspace Operators | 27.7 | 27.3 | 3.5 | 0.5 | 7.9 |
| Calculation of Eigenvector Approximations | 66.8 | 38.8 | 15.4 | 7.0 | 21.8 |
| Overrelaxation, Shifting, Gram-Schmidt Orthogonalization and Replacement of Iteration Vectors | 0. | 1.5 | 24.4 | 37.8 | 0.8 |
| Total Central Processor Time in Seconds | 195 | 132 | 93 | 88 | 77 |

standard starting subspace. However, considering the solution with q = 20 and q = 4, it is seen from Table 5.6 that the required high speed storage and the total solution cost are reduced considerably. Since the stiffness matrix has a very small average bandwidth, the determinant search method also yields an effective solution for this problem.

The use of accelerated subspace iteration method employing the Lanczos starting subspace, with q = 48, results in the most effective solution for this problem. Since the converged iteration vectors are no longer included in the accelerated subspace iteration (see Section 5.3), for the cases q = 48 the computational time required to perform the subspace iterations is considerably reduced (see Table 5.7). The maximum reduction is achieved in the solution using the Lanczos starting subspace, since 30 out of the 48 eigenpair iterates had converged to tolc = $10^{-10}$ after only three iterations.

# 6. THE CALCULATION OF EIGENVALUES IN A SPECIFIED INTERVAL

## 6.1 Introduction

In the previous chapters, the solution for the p smallest eigenvalues and associated vectors of the generalized eigenproblem

$$\underline{K}\,\underline{\phi} = \lambda\,\underline{M}\,\underline{\phi} \tag{6.1}$$

was considered. In the earthquake response analysis of structures using mode superposition method, generally only the smallest few eigenvalues and associated eigenvectors need be considered. However, experience has shown that in the dynamic response analysis of many structures subjected to loads that vary with time, it is often necessary to consider the response in some modes which are not necessarily the modes associated with the smallest eigenvalues. In other words, the solution for all the eigenvalues that lie in a specified interval and the associated eigenvectors, of the generalized eigenproblem in Eq. (6.1) is required. In Section 6.2, a critical review of some solution techniques that are currently used for such analyses is presented . Then in Section 6.3, we present modifications to the accelerated subspace iteration method (see Chapter 5) in order to obtain an algorithm for the calculation of within a specified interval. The computer implementation of the modifications and the results obtained in the solution of some eigenproblems using the accelerated subspace iteration method are also given in Section 6.3.

## 6.2 A Critical Review of Some Existing Solution Techniques

We point out first a major difficulty that is encountered in the development of an eigensolution scheme for calculating only the eigenvalues in an interval $(\alpha,\beta)$, $\alpha < \beta$, without computing all the eigenvalues that are smaller than $\alpha$ (or all the eigenvalues larger than $\beta$):

In order to compute an eigenpair in the interval $(\alpha,\beta)$ let us suppose that the following inverse iteration scheme is employed

$$(\underline{K} - \mu\underline{M})\ \underline{x}_{k+1} = \underline{M}\ \underline{x}_k \tag{6.2}$$

where $\alpha < \mu < \beta$ and $\underline{x}_k$ is the iteration vector at the $k^{th}$ iteration. Consider the case, where the starting iteration vector $\underline{x}_1$ is orthogonal to all the eigenvectors associated with the required eigenvalues. Also, let

$$\underline{x}_i = \underline{\phi}_i + \underline{\phi}_j \tag{6.3}$$

where $\underline{\phi}_i$ is the eigenvector associated with an eigenvalue $\lambda_i < \alpha$ and $\underline{\phi}_j$ is the eigenvector associated with an eigenvalue $\lambda_j > \beta$. When the eigenvalues $\lambda_i$ and $\lambda_j$ satisfy the relation

$$\mu - \lambda_i = \lambda_j - \mu \tag{6.4}$$

it is seen that the Rayleigh quotient of the iteration vector $\underline{x}_1$ is

$$\rho(\underline{x}_1) = \frac{\underline{x}_1^T\ (\underline{K}-\mu\underline{M})\underline{x}_1}{\underline{x}_1^T\ \underline{M}\ \underline{x}_1} = 0 \tag{6.5}$$

Hence, in two successive inverse iterations the same value for the

Rayleigh quotient (the corresponding eigenvalue approximation is $\mu$) is computed and the iteration values are accepted, rather erroneously!

In the above illustrative example, a very pathological case was considered. However, the example shows that "an iteration vector that is orthogonal to the subspace spanned by the required eigenvectors can give a Rayleigh quotient which lies in the specified interval." Of course, the error in the accepted eigenpair approximation is detected when the physical error bounds are computed using Eq. (3.8). The above analysis shows that also a tight convergence tolerance must be used in the iteration. Even if the components of the two eigenvectors $\underline{\phi}_i$ and $\underline{\phi}_j$ present in $\underline{x}_1$ are not exactly equal as assumed in Eq. (6.3), still the Rayleigh quotient computed in Eq. (6.5) may not change appreciably in the inverse iterations and hence the need for using a tight convergence tolerance. It should be noted that this situation does not arise when $\mu < \lambda_1$ and the matrix $\underline{K}$ is positive definite or when $\mu > \lambda_1$ but the eigenvectors corresponding to all the eigenvalues smaller than $\mu$ have been computed and are included in the iteration.

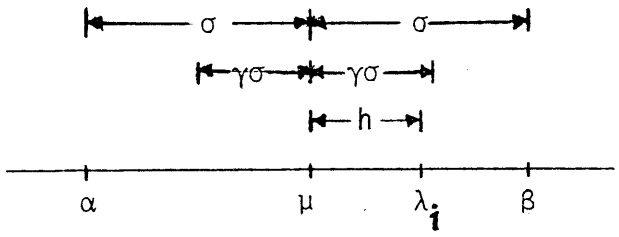
There are only a few solution techniques that are currently used for the calculation of eigenvalues in an interval [39, 40, 23], and all the solution schemes available are basically inverse iteration schemes.

The method of sectioning

The method of sectioning proposed by Jensen [39] consists of three steps. First, a basis for the subspace spanned by the required eigenvectors is established using inverse iteration with shifts. Next, the

projections of the operators $\underline{K}$ and $\underline{M}$ onto the subspace are calculated. Finally the small eigenvalue problem corresponding to the operator projections is solved using the HQRI method [1, p. 460]. In essence, one subspace iteration is performed with a set of starting vectors, which presumably span the same subspace that is spanned by the required eigenvectors.

In thisscheme, inverse iterations with shifts are employed to generate the starting iteration vectors. In the following figure $\alpha$ and $\beta$ represent the end points of the eigenvalue interval under consideration. Using an arbitrary starting iteration vector, inverse iterations with a shifted matrix (see Eq. (6.2)) are performed. The initial shift value is chosen as the mid-point of the specified interval, i.e. $\mu = (\alpha+\beta)/2$. An input constant $\gamma$, where $0<\gamma<1$, together with the values of the Rayleigh quotients computed in the inverse iteration (h in figure), determines the shifting strategy, as described below.




(i) If the estimates of $|h|$ converge to a value greater than $\sigma$, where $\sigma = \min.(\beta-\mu, \mu-\alpha)$ (equal to $(\beta-\alpha)/2$ in figure), it is assumed that there are no eigenvalues in the interval $(\mu-\sigma,\mu+\sigma)$.

(ii) If the estimates of $|h|$ converge to a value less than $\gamma\sigma$, the iteration is continued until a specified accuracy on $|h|$ is reached. Upon convergence, the iteration vector is accepted to be a "basis" vector for the subspace spanned by the required eigenvectors. In-

verse iterations with a new vector are then performed and the value of the new Rayleigh quotients are tested as before. Also, during the inverse iterations, the current iteration vector is orthogonalized to all the previously computed "basis" vectors.

(iii) If the estimates of $|h|$ converge to a value $\gamma\sigma < |h| < \sigma$, it is concluded that there are no eigenvalues, other than any previously computed, in the range $\mu-\gamma\sigma < \lambda < \mu+\gamma\sigma$. In this case the two intervals $(\alpha,\mu-\gamma\sigma)$ and $(\mu+\gamma\sigma,\beta)$ are separately analyzed, using essentially the same algorithm, but with new shift points. In other words, a new shift value in the interval $(\alpha,\mu-\gamma\sigma)$ is chosen and inverse iterations with the current iteration vector are performed, etc.

The efficiency of the scheme very much depends on the value of $\gamma$.

- If $\gamma$ is small, several factorizations of the matrix $\underline{K} - \mu\underline{M}$ are performed, but the subspace spanned by the accepted iteration vectors tends to be close to the subspace spanned by the required eigenvectors. This conclusion is based on the fact that due to frequent shifting, the possibility of the shift points being close to the required eigenvalues is increased, thus improving the accuracy obtained in the inverse iterations.
- If $\gamma$ is large, fewer factorizations are performed. However, the rate of convergence in the inverse iteration, i.e., the rate at which $|h|$ converges to a particular value, is considerably reduced, thus, increasing the solution cost. In this case, the pole of attraction in the inverse iteration, namely the shift point is held

stationary during most part of the solution. Therefore, the quality of the finally accepted iteration vectors can be poor, as illustrated by the example presented in the beginning of this section.

The solution accuracy in thisscheme can be significantly improved by performing a subspace iteration solution, with the vectors generated using inverse iterations as the starting iteration vectors.

The most severe drawback of the scheme is that several shifts are performed and no consideration is given to any of the restrictions on matrix shifting that we presented in Section 5.3.

A modified determinant search method

In the determinant search method, secant iteration is employed to calculate a shift that is close to one of the required eigenvalues. Then, inverse iteration is employed to compute accurately the corresponding eigenvector and to improve the accuracy of the eigenvalue approximation. In the scheme presented in ref. [1, 23] the smallest eigenvalue is computed first. In order to compute the larger eigenvalues subsequent secant iterations are performed on the deflated polynomial. Kaul presented a modification of the secant iteration step so that the determinant search method can be employed to calculate eigenvalues in a specified interval $(\alpha,\beta)$ [40].

In the modified determinant search method, the number of eigenvalues that are smaller than $\alpha$ is calculated using the Sturm sequence property. The only other modification to the original scheme [see Chapter 2] is that in the secant iteration, the polynomial values at all

the shift points are deflated using several constants chosen in the interval $(0,\alpha)$; i.e., let $\mu_1$ and $\mu_2$ be the estimates of the smallest eigenvalue $\lambda_1$ and let $P(\mu_1)$ and $P(\mu_2)$ be the corresponding values of the matrix polynomial $(\underline{K} - \mu\underline{M})$. Then the new estimate $\mu_3$ to the eigenvalue $\lambda_1$ predicted using secant iteration is

$$\mu_3 = \mu_1 - \frac{\mu_2 - \mu_1}{P(\mu_2) - P(\mu_1)} P(\mu_1) \tag{6.6}$$

It is suggested in ref. [40] that the polynomial $P(\mu)$ in Eq. (6.6) be deflated by the following polynomial $F(\mu)$, in which case the iterate $\mu_3$ in secant iteration converges to the smallest eigenvalue that is to the immediate right of the initial shift $\alpha$,

$$F(\mu) = \prod_{i=1}^{r} (\mu - C_i)^{n_i - n_{i-1}} \tag{6.7}$$

where $C_i$, $i=1,\ldots, r$, are some r arbitrarily chosen constants in the interval $(0,\alpha)$; $n_i$ is the number of eigenvalues smaller than $C_i$, established using the Sturm sequence property and $\underline{L}\,\underline{D}\,\underline{L}^T$ decomposition of the matrix $(\underline{K} - C_i\underline{M})$; $\Pi$ is the product symbol. In ref. [40], it is proven that the secant iteration on the polynomial $P(\mu)/F(\mu)$ yields monotonic convergence to the eigenvalue that is to the immediate right of the shift $\alpha$.

The scheme is effective only when the bandwidth of the matrices involved in the eigensolution is small because several triangular factorizations are employed. The number and the values of the constants $C_i$ in Eq. (6.7) should be chosen with care, since the slope of the deflated polynomial (which governs the rate of convergence in secant iteration) is affected by these parameters.

Inverse iteration is employed in the scheme to compute the final eigenpair approximations. When close or multiple eigenvalues are present in the eigenproblem being solved, convergence difficulties are encountered in the inverse iteration.

The Sturm sequence method

The Sturm sequence method for the calculation of all eigenvalues in the interval $(\alpha,\beta)$ consists of the following steps [18]:

(i) The matrix $\underline{K}-\alpha\,\underline{M}$ is factorized into $\underline{L}\,\underline{D}\,\underline{L}^T$ factors and the number of eigenvalues that are smaller than $\alpha$ is calculated using the Sturm sequence property -- the number of eigenvalues smaller than $\alpha$ is equal to the number of negative diagonal elements in the matrix $\underline{D}$.

(ii) Similarly, the number of eigenvalues smaller than $\beta$ is computed by factorizing the matrix $\underline{K}-\beta\,\underline{M}$.

(iii) The upper and lower limits on all eigenvalues in the interval $(\alpha,\beta)$ are obtained by repeated bisection [1, p. 479].

(iv) Having computed rough bounds for each eigenvalue in the interval, successive linear interpolation (which requires additional factorizations) is employed to calculate the eigenvalues to the accuracy specified by the analyst. Then, the corresponding eigenvector is obtained using the inverse iteration with shifting, where the calculated eigenvalue approximation is employed as the shift.

In this scheme, several factorizations are performed; hence, it is not effective to employ this scheme for the solution of large eigenproblems, particularly when the matrices, $\underline{K}$ and $\underline{M}$ are stored out-of-core. A relatively large number of factorizations may be required in step (iii) in order to isolate almost equal eigenvalues. The convergence rate in the inverse iteration is very much dependent on the accuracy of the eigenvalue approximations calculated in step (iv).

## 6.3 Modifications to the Accelerated Subspace Iteration Method

In Chapter 5, an accelerated subspace iteration method was presented. In the accelerated method, self-adaptive shifting strategies have been incorporated; i.e. at the end of each subspace iteration, the rate of convergence of the eigenpair iterates are estimated and if necessary matrix shifts are performed in order to accelerate the convergence. The shifts are applied only in-between well converged and well separated eigenvalues and the magnitude of the applied shifts increase monotonically. Also, after an eigenvalue iterate converges to an accuracy of tolc equal to $10^{-10}$ using Eq. (3.6), the corresponding eigenvector iterate is no longer included in the subspace iteration (see Section 5.3). Thus, the accuracy of the converged iteration vectors is preserved during further subspace iterations with new shifts. The scheme can be employed with q, the number of iteration vectors, smaller than p, the number of eigenpairs sought. In the iteration with $q < p$, Gram-Schmidt orthogonalizations of the current iteration vectors to the converged iteration vectors that have been removed to back-up storage, are performed in a conservative manner. The above solution strategies are well suited for the calculation of the eigenvalues in a specified interval and the associated eigenvectors. However, to improve the solution effectiveness, a few modifications to the accelerated subspace iteration method are necessary.

A summary of the solution steps in the accelerated subspace iteration method was presented in Section 5.5 (see Fig. 5.2). In the following, only the necessary modifications to the procedure described in Section 5.5 are outlined.

In the accelerated subspace iteration method for the calculation of eigenvalues in an interval $(\alpha,\beta)$, the initial shift value, $\mu_1$, is equal to $\alpha$, i.e., the matrix $(\underline{K}-\mu_1\underline{M})$ is factorized and employed instead of the matrix $\underline{K}$ in Eq. (3.1). Also, the number of eigenvalues of Eq. (6.1) that are smaller than the shift $\alpha$ is noted. In the subspace iterations, Eqs. (3.1) to (3.5), q iteration vectors are employed, where q is input by the analyst.

(a) It should be noted that the initial shift $\mu_1$ can be on or very close to an eigenvalue, in which case all iteration vectors immediately converge to the eigenvector corresponding to that eigenvalue and hence, theshift must be changed. If the shift $\mu_1$ is very close to an eigenvalue, the last pivot element in the $\underline{L}\,\underline{D}\,\underline{L}^T$ factorization of the matrix $\underline{K}-\mu_1\underline{M}$ becomes small compared to its original value. In such case, the shift $\mu_1$ is changed to $(1-10^{-5})\mu_1$ and the factorization is repeated. If the last pivot element is still small, the shift is changed to $(1-10^{-4})\mu_1$ and the shifted matrix is refactorized. Also, in case a shift $\mu$ is

very close to one of the eigenvalues of an associated constraint eigenproblem [1, p. 271], the factorization breaks down; i.e., it is possible that during the factorization of the matrix $\underline{K}-\mu\underline{M}$ any pivot element can become small and in this case also the value of the shift $\mu$ is changed as described above and the factorization is repeated. However, such breakdown of the factorization has not been observed in the numerical examples analyzed in this study.

(b) The rate of convergence calculations, overrelaxation and shifting considerations are all based only on the iteration pairs which yield eigenvalue estimates larger than $\mu_1$. In other words, the iteration vectors for which the corresponding eigenvalue estimates are smaller than $\mu_1$ are included in the subspace iterations only for stability considerations.

(c) In the criterion for the maximum allowable shift, Eq. (5.23) (and Eq. (5.31)), the eigenvalue approximation $\overline{\lambda}_1$ is interpreted in this case, as the smallest eigenvalue estimate to the right of the shift $\mu_1$. Hence, it is seen that the (modified) shifting criterion in Eq. (5.23) is biased towards shifting to the right in the interval $(\alpha,\beta)$. Whenever a shift is applied, the Sturm sequence check is performed as described in Section 5.3 (see

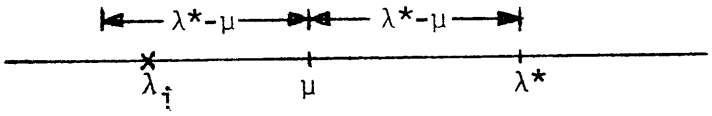
Eq. (5.36)).

(d) Next, assume that the largest possible shift that satisfies the (modified) shifting criterion in Eq. (5.23) has been applied or has been discarded based on Eq. (5.29). Then, all the iteration vectors that correspond to the eigenvalue iterates which are larger than $\alpha$ and consecutively all satisfy Eq. (3.6) to a tolerance of $10^{-10}$, are transferred to back-up storage and replaced with new starting iteration vectors. Also, if there is any iteration vector in the current set of iteration vectors for which the corresponding eigenvalue estimate is smaller than $\alpha$, a matrix shift is performed simultaneously. Assume that $\bar{\lambda}_j$ is the largest converged eigenvalue estimate in the current iteration and that all eigenvalues in the range $\alpha \leq \lambda_i \leq \bar{\lambda}_j$ have been computed to an accuracy of $10^{-10}$ using Eq. (3.6). The new shift value is chosen to be $\mu_s = (\bar{\lambda}_j+\bar{\lambda}_{j-1})/2$. The condition given in Eq. (5.34) is also taken into account, before accepting a candidate shift $\mu_s$.

(e) Let the value of the current shift be $\mu$ and the value of the largest eigenvalue estimate obtained in the current

subspace iteration be $\lambda^*$. Then, the current set of iteration vectors always either include or are orthogonalized to all the vectors which approximate the eigenvectors $\phi_i$ that correspond to the eigenvalues $\lambda_i$ with (see Eq. (5.30))

$$\lambda_i \geq \mu - (\lambda^* - \mu) \tag{6.8}$$




(f) In most solutions for the calculation of eigenvalues in an interval, it is necessary to perform matrix shifts and to remove some of the converged iteration vectors to back-up storage. Also, since the number of eigenvalues in the specified interval is not known until the final solution is obtained, the iteration procedure is similar to the case $q < p$. Hence, the eigenvalues are <u>always</u> computed to an accuracy of tolc $\leq 10^{-10}$ using Eq. (3.6).

(g) The subspace iterations are stopped when the magnitude of an eigenvalue iterate in the current set exceeds the limit point $\beta$ and the magnitudes of all eigenvalue estimates which lie in the interval $(\alpha,\beta)$ have been computed to an accuracy of tolc $\leq 10^{-10}$ using Eq. (3.6).

The above modifications to the accelerated subspace iteration method have been implemented in the computer program given in Appendix B. The results obtained in the solution of some eigenproblems are given in Table 6.1. It is seen from the table that the solution procedure is similar to the iteration procedure for $q < p$, presented in Section 5.4; i.e., the computed eigenvectors are removed in bundles and stable matrix shifts are employed in the solution to improve the convergence rates to the required eigenpairs.

Based on the theory employed and on the numerical experiments, we conclude that the iteration procedure presented in this section is a reliable and effective solution scheme for the calculation of eigenvalues in a specified interval and the associated vectors. Comparing the accelerated subspace iteration method, in particular employing the Lanczos starting subspace (see Section 4.5), to the method of sectioning [39], it is seen that the starting vectors in the subspace iteration scheme are obtained in a more systematic manner and at a considerably reduced cost. Also, in the accelerated subspace iteration method, fewer factorizations are employed than in the modified determinant search method [40] or the Sturm sequence method [18]; hence, the accelerated scheme developed in this study is well suited for the calculation of eigenvalues in an interval for large eigenproblems.

TABLE 6.1 SOLUTION RESULTS FOR THE CALCULATION OF EIGENVALUES IN THE SPECIFIED INTERVAL

| | 16 Degree of Freedom Tridiagonal System | | Piping System | |
|---|---|---|---|---|
| $n, m_K$ | 16, 0 | 16, 0 | 330, 26 | |
| Number of Eigenpairs in the Interval | 4 | 5 | 9 | |
| Number of Iteration Vectors Used | 8 | 8 | 12 | |
| Total Number of Subspace Iterations Performed | 12 (8)* | 39 (31) | 64 (54) | |
| Total Number of Factorizations Performed | 2 | 5 (4) | 6 (4) | |
| Iteration Vectors Removal Strategy | None | 3+2 (2+3) | 3+3+3 (4+3+2) | |

* Numbers in parentheses refer to the solution obtained using the Lanczos starting subspace, if the values were different from the values obtained using the Standard starting subspace.

## 7. SUMMARY AND CONCLUSIONS

A solution method for the calculation of all eigenvalues in an interval $(\alpha,\beta)$ and the associated vectors of the generalized eigenproblem $\underline{K}\,\underline{\phi} = \lambda\,\underline{M}\,\underline{\phi}$ has been presented. In case the smallest p eigenvalues and the associated eigenvectors are required the end points $\alpha$ and $\beta$ of the interval should be specified as $\alpha$ equal to zero and $\beta$ equal to a very large value. The method is well suited for the finite element eigenproblems, in which the matrices $\underline{K}$ and $\underline{M}$ are large, sparse, symmetric and positive or semi-positive definite.

In principle, a large number of solution methods are currently available for the solution of the generalized eigenproblems that arise in structural analysis. However, considering the solution for a few eigenpairs of large eigenproblems, only a few solution techniques -- in particular the determinant search method, the subspace iteration method, and the Lanczos method, are effective because the methods have been specifically developed to exploit the sparse and symmetric structure of the finite element matrices. During the course of the current research, several eigenproblems were solved using the above three methods in order to recognize the advantages and shortcomings of these methods. The specific conclusions drawn from the numerical experiments are briefly summarized next.

The important aspect of the determinant search method (see Chapter 2) is the improvement of the convergence rate in the inverse iterations due to shifting; a large number of factorizations is performed in the solution which increases the solution cost considerably for large eigenproblems, particularly when the matrices are stored out-of-core; also, sometimes the values of the shifts for the inverse iteration calculated using the secant iteration on the characteristic polynomial are not good approximations to the required eigenvalues and, therefore, convergence difficulties occur in the inverse iterations. The Lanczos method (see Chapter 4) was found to be effective in many cases. However, this method exhibits some numerical problems when implemented on a computer; the accuracy of the solution can be poor when multiple or almost equal eigenvalues are present in the eigenproblem being solved; the selection of the starting vector is very crucial because the solution accuracy is largely dependent on the spectral content of the starting vector. The subspace iteration method (see Chapter 3) was found to be reliable in all cases; the simultaneous iteration concept that is employed in this method is ideally suited for the solution of large eigenproblems. However, when a large number of eigenpairs was calculated using the subspace iteration method, the solution cost was often found to be excessive.

The accelerated subspace iteration procedure developed in this study is a significant extension of the basic subspace iteration method and is more general and more effective than the basic method:

(i) The number of numerical operations per subspace iteration has been reduced by avoiding iterations with the converged iteration vectors. This modification leads to a considerable savings in the solution cost when a large number of eigenpairs is calculated; also, the accuracy of the converged iteration vectors is thus preserved during the subsequent iterations at new shifts.

(ii) An overrelaxation procedure has been developed to accelerate the convergence rate in the subspace iterations. The overrelaxation procedure is applicable to any linearly convergent iterative process. The overrelaxation scheme has been found to improve considerably the effectiveness of the subspace iteration method.

(iii) A stable shifting procedure has been developed. The effectiveness of the subspace iteration method for the solution of structural systems with small bandwidth has been improved by incorporating the shifting procedure. Also, when close eigenvalues are present in the eigenproblem being solved, the shifting accelerates the convergence rate considerably. It should be noted that the shifting procedure developed in this study is based on conservative notions; i.e., as much importance has been

given to the solution stability as to the acceleration of the convergence in the iteration. Also, the shifting procedure is applied in a self-adaptive manner with due consideration to the number of numerical operations involved in the shifting.

(iv) The effectiveness of the starting iteration vectors to the subspace iterations has been improved. A random starting vector has been included to the set of starting vectors that are usually generated in the basic subspace iteration method. Also, an algorithm to generate the subspace iteration starting vectors using the Lanczos method has been developed. The Lanczos starting subspace has been found to be effective only when the matrices are stored in high speed core.

(v) The subspace iteration procedure has been modified such that the number of iteration vectors can be chosen idependently of the number of eigenpairs required. This modified procedure leads to two advantages: (1) the total high-speed storage required during the subspace iterations is significantly reduced; also, the large peripheral processing cost associated with the out-of-core solutions which otherwise may be necessary, is avoided; (2) unnecessary orthogonalizations of the current iteration vectors to the converged iteration vectors is avoided.

(vi) A computer implementation of the out-of-core solution procedure has been presented, in which only the matrices $\underline{K}$ and $\underline{M}$ are blocked and stored out-of-core, whereas all the iteration vectors are kept in core. This implementation has been found to be effective, particularly when a diagonal matrix $\underline{M}$ is employed.

(vii) The accelerated subspace iteration method can be employed also to compute all eigenvalues within an interval and the associated eigenvectors. This method has been found to be more reliable and effective than the solution methods that are currently employed for the calculation of eigenvalues in an interval.

Suggestions for future research

The efficiency of the solution method developed in this study depends to some extent on the value of q, namely the number of iteration vectors used in the solution. The solution algorithm can be modified further in order that the optimal value for q is calculated by the algorithm itself. Currently the analyst has to specify the value of q. It should be noted that the optimal value of q depends on several factors like, the number of eigenpairs required, the order and bandwidth of the matrices, the spreading in the eigenvalues of the eigenproblem being solved, the total high speed core storage available, the cost difference between the central processor unit

time and the input/output time.

The eigenproblem of the operator projections in the subspace iteration method is solved using a generalized Jacobi method. When q is large, it was found that the Jacobi method requires a considerable amount of computational time. Hence, alternate means of solving the eigenproblem of the operator projections should be explored.

## 8. REFERENCES


1. Bathe, K.J. and Wilson, E.L., Numerical Methods in Finite Element Analysis, Prentice-Hall, Inc., 1976.

2. Wilkinson, J.H., The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

3. Gourlay, A.R. and Watson, G.A., Computational Methods for Matrix Eigenproblems, John Wiley & Sons, 1973.

4. Strang, G. and Fix, G.J., An Analysis of the Finite Element Method, Prentice-Hall, Inc., 1973.

5. Wilkinson, J.H., Rounding Errors in Algebraic Processes, Prentice-Hall, Inc., 1963.

6. Ralston, A., A First Course in Numerical Analysis, McGraw-Hill Book Company, 1965.

7. Crandall, S.H., Engineering Analysis, McGraw-Hill Book Company, New York, 1956.

8. Bathe, K.J. and Wilson, E.L., "Large Eigenvalue Problems in Dynamic Analysis," Proc. of the ASCE, Vol. 98, No. EM6, Dec. 1972, pp. 1471-1485.

9. Bathe, K.J. and Wilson, E.L., "Solution Methods for Eigenvalue Problems in Structural Mechanics," Int. Journal for Num. Methods in Eng., Vol. 6, 1973, pp. 213-226.

10. Peters, G. and Wilkinson, J.H., "$Ax = \lambda Bx$ and the Generalized Eigenproblem," SIAM Journal of Numerical Analysis, Vol. 7, No. 4, Dec. 1970, pp. 479-492.

11. Ostrowski, A.M., "On the Convergence of the Rayleigh Quotient Iteration for the Computation of the Characteristic Roots and Vectors, Parts I-VI," Archives for Rational Mechanics and Analysis, Vols. 1-4, 1958-1959.

12. Brönlund, O.E., "Eigenvalues of Large Matrices," Symposium on Finite Element Techniques at the Institut für Statik und Dynamik der Luft und Raumfahrtskonstruktionen, Univ. of Stuttgart, Germany, 1969.

13. Parlett, B.N. and Kahan, W., "On the Convergence of a Practical QR Algorithm," Proceedings, I.F.I.P. Congress, 1968.

14. Fried, I., "Optimal Gradient Minimization Scheme for Finite Element Eigenproblems," Journal of Sound and Vibration, Vol. 20, No. 3, 1972, pp. 333-342.

15. Fox, R.L. and Kapoor, M., "A Minimization Method for the Solution of the Eigenproblem Arising in Structural Dynamics," Proceedings, 2nd Conference on Matrix Methods in Structural Mechanics, Wright-Patterson A.F.B., Ohio, 1968.

16. Schwarz, H.R., "The Method of Coordinate Overrelaxation for $(A-\lambda B)X = 0$," Numerische Mathematik, Vol. 23, 1974, pp. 135-151.

17. Gupta, K.K., "Vibration of Frames and Other Structures with Banded Stiffness Matrix," Int. Journal for Num. Methods in Eng., Vol. 2, 1970, pp. 221-228.

18. Gupta, K.K., "Eigenproblem Solution by a Combined Sturm Sequence and Inverse Iteration Technique," Int. Journal for Num. Methods in Eng., Vol. 7, 1973, pp. 17-42.

19. Guyan, R.J., "Reduction of Stiffness and Mass Matrices," A.I.A.A. Journal, Vol. 3, No. 2, 1965, pp. 380.

20. Hurty, W.C., "Dynamic Analysis of Structural Systems Using Component Modes," A.I.A.A. Journal, Vol. 3, 1965, pp. 678-685.

21. MacNeal, R.H., "A Hybrid Method of Component Mode Synthesis," Journal of Computers and Structures, Vol. 1, No. 4, Dec. 1971, pp. 581-601.

22. Craig, R.R., Jr. and Chang, C-J., "A Review of Substructure Coupling Methods for Dynamic Analysis," Advances in Engineering Science, V. 2, NASA CP-2001, Nov. 1976, pp. 393-408.

23. Bathe, K.J., "Solution Methods for Large Generalized Eigenvalue Problems in Structural Engineering," Report UC SESM 71-20, University of California, Berkeley, Nov. 1971.

24. Bathe, K.J., "Convergence of Subspace Iteration," in Formulations and Numerical Algorithms in Finite Element Analysis, Bathe, K.J., Oden, J.T. and Wunderlich, W. (eds.), M.I.T. Press, 1977, pp. 575-598.

25. Laasonen, P., "A Ritz Method for Simultaneous Determination of Several Eigenvalues and Eigenvectors of a Big Matrix," Annales Academiae Scientiarum Fennicae, Ser. A, Sec. 1, Math. 265, 1959, pp. 1-16.

26. Jennings, A., "A Direct Iteration Method of Obtaining Latent Roots and Vectors of a Symmetric Matrix," Proc. of the Cambridge Phil. Society, Vol. 63, 1967, pp. 755-765.

27. Rutishauser, H., "Computational Aspects of F.L. Bauer's Simultaneous Iteration Method," Numerische Mathematik, Vol. 13, 1969, pp. 4-13.

28. Rutishauser, H., "Simultaneous Iteration Method for Symmetric Matrices," Numerische Mathematik, Vol. 16, 1970, pp. 205-223.

29. Jennings, A. and Orr, D.R.L., "Application of the Simultaneous Iteration Method to Undamped Vibration Problems," Int. Journal for Num. Methods in Eng., Vol. 3, 1971, pp. 13-24.

30. Paige, C.C., "Computational Variants of the Lanczos Method for the Eigenproblem," Journal of the Institute of Mathematics and its Applications, Vol. 10, 1972, pp. 373-381.

31. Weaver, W. and Yoshida, D.M., "The Eigenvalue Problem for Banded Matrices," Computers and Structures, Vol. 1, 1971, pp. 651-664.

32. Newman, M. and Flanagan, P.F., "Eigenvalue Extraction in NASTRAN by the Tridiagonal Reduction (FEER) Method - Real Eigenvalue Analysis," Report, NASA CR-2731, Aug. 1976.

33. Golub, G.H. and Underwood, R., "The Block Lanczos Method for Computing Eigenvalues," Mathematical Software III, Ed. Rice, J.R., Academic Press, 1977, pp. 361-379.

34. Yamamoto, Y. and Ohtsubo, H., "Subspace Iteration Accelerated by Using Chebyshev Polynomials for Eigenvalue Problems with Symmetric Matrices," Int. Journal for Num. Methods in Eng., Vol. 10, 1976, pp. 935-944.

35. Jennings, A., "Accelerating the Convergence of Matrix Iterative Processes," Journal of the Institute of Mathematics and its Applications, Vol. 8, 1971, pp. 99-110.

36. Bathe, K.J. and Ramaswamy, S., "Subspace Iteration with Shifting for the Solution of Large Eigensystems," Report, AVL 82448-7, Dept. of Mech. Eng., Massachusetts Institute of Technology, Dec. 1977.

37. Bathe, K.J. and Ramaswamy, S., "An Accelerated Subspace Iteration Method," Journal Computer Methods in Applied Mechanics and Engineering, submitted for publication.

38. Akl, F., Dilger, W.H. and Irons, B.M., "Over-relaxation and Subspace Iteration," Int. Journal for Num. Methods in Eng., Vol. 14, No. 4, 1979, pp. 629-630.

39. Jensen, P.S., "The Solution of Large Symmetric Eigenproblems by Sectioning," SIAM Journal of Num. Analysis, Vol. 9, No. 4, Dec. 1972, pp. 534-545.

40. Kaul, M.K., "On Determination of Eigenvalues of Large Structural Systems in an Arbitrarily Specified Range," Int. Journal for Num. Methods in Eng., Vol. 11, No. 5, 1977, pp. 867-874.

41. Bathe, K. J., "ADINA - A Finite Element Program for Automatic Dynamic Incremental Nonlinear Analysis," Report AVL 82448-1, Massachusetts Institute of Technology, May 1975 (rev. Dec. 1978).

42. Rivers, A., "ADINA on the STAR 100", in Proceedings of the ADINA Conference August 1979, Bathe, K. J., (Ed.), Report AVL 82448-9, Massachusetts Institute of Technology, August, 1979.

## APPENDIX A - AN ANALYSIS OF THE LANCZOS METHOD

In Section 4.3, the Lanczos method for the solution of a few smallest eigenvalues and the associated eigenvectors of the generalized eigenproblem in Eq. (4.1) was described. In this Appendix, first the relation given in Eq. (4.8) is proved. Then, it is pointed out that a Ritz analysis employing the set of vectors generated using the Lanczos method yields under certain conditions good approximations to the smallest eigenvalues and the associated eigenvectors. Finally, the solutions of three small eigenproblems using the Lanczos method are presented.

### Orthogonality of the generated vectors

In the Lanczos method, a sequence of vectors is generated from a starting vector $\underline{x}_1$ using the following recurrence algorithm (see Eqs. (4.2) to (4.7))

$$\beta_{i+1}\,\underline{x}_{i+1} = \underline{K}^{-1}\,\underline{M}\,\underline{x}_i - \alpha_i\,\underline{x}_i - \beta_i\,\underline{x}_{i-1} \qquad (A.1)$$

It is next proven using the principle of mathematical induction that the vector $\underline{x}_{i+1}$ is theoretically $\underline{M}$-orthogonal to all the vectors $\underline{x}_j, j=1,\ldots,i$.

It is seen from Eq. (A.1) that

$$\beta_2 \underline{x}_1^T \underline{M} \underline{x}_2 = \underline{x}_1^T \underline{M} \underline{K}^{-1} \underline{M} \underline{x}_1 - \alpha_1 \underline{x}_1^T \underline{M} \underline{x}_1 = 0 \qquad (A.2)$$

because $\underline{x}_1^T \underline{M} \underline{x}_1$ equals 1 from Eq. (4.2) and $\underline{x}_1^T \underline{M} \underline{K}^{-1} \underline{M} \underline{x}_1$ equals $\alpha_1$ from Eq. (4.4). Therefore, the vector $\underline{x}_2$ is $\underline{M}$ orthogonal to the vector $\underline{x}_1$.

Assume that the vectors $\underline{x}_\ell$, $\ell = 1,\ldots,i$, are $\underline{M}$-orthogonal. Then, for $j = 1,\ldots,i-2$, we have from Eq. (A.1)

$$\beta_{i+1} \underline{x}_j^T \underline{M} \underline{x}_{i+1} = \underline{x}_j^T \underline{M} \underline{K}^{-1} \underline{M} \underline{x}_i - \alpha_i \underline{x}_j^T \underline{M} \underline{x}_i - \beta_i \underline{x}_j^T \underline{M} \underline{x}_{i-1} \qquad (A.3)$$

But, again from Eq. (A.1)

$$\underline{x}_i^T \underline{M} \underline{K}^{-1} \underline{M} \underline{x}_j = \underline{x}_i^T \underline{M} (\beta_{j+1} \underline{x}_{j+1} + \alpha_j \underline{x}_j + \beta_j \underline{x}_{j-1}) \qquad (A.4)$$

Since the vectors $\underline{x}_\ell$, $\ell=1,\ldots,i$, have been assumed to be $\underline{M}$-orthogonal, each term on the right-hand side of the equal sign in Eqs. (A.4) and (A.3) are zero. Therefore, the vector $\underline{x}_{i+1}$ is $\underline{M}$-orthogonal to all the vectors $\underline{x}_j$, $j=1,\ldots,i-2$. Also, from Eq. (A.1)

$$\beta_{i+1} \underline{x}_i^T \underline{M} \underline{x}_{i+1} = \underline{x}_i^T \underline{M} \underline{K}^{-1} \underline{M} \underline{x}_i - \alpha_i \underline{x}_i^T \underline{M} \underline{x}_i - \beta_i \underline{x}_i^T \underline{M} \underline{x}_{i-1} = 0 \qquad (A.5)$$

$$\beta_{i+1} \underline{x}_{i-1}^T \underline{M} \underline{x}_{i+1} = \underline{x}_{i-1}^T \underline{M} \underline{K}^{-1} \underline{M} \underline{x}_i - \alpha_i \underline{x}_{i-1}^T \underline{M} \underline{x}_i -$$

$$\beta_i \underline{x}_{i-1}^T \underline{M} \underline{x}_{i-1} = 0 \qquad (A.6)$$

In simplifying the above equations, the facts that $\underline{x}_i^T \underline{M} \underline{x}_i = \underline{x}_{i-1}^T \underline{M} \underline{x}_{i-1} = 1$ (Eq. (4.7)), $\underline{x}_{i-1}^T \underline{M} \underline{x}_i = 0$, $\underline{x}_i^T \underline{M} \underline{K}^{-1} \underline{M} \underline{x}_i = \alpha_i$ (Eq. (4.4)), and $\underline{x}_i^T \underline{M} \underline{K}^{-1} \underline{M} \underline{x}_{i-1} = \beta_i$ (Eqs. (4.3) to (4.7)) have been used. Hence, the generated vector $\underline{x}_{i+1}$ is theoretically $\underline{M}$-orthogonal to all the computed vectors $\underline{x}_\ell$, $\ell=1,\ldots,i$.

Proof of the Relation in Eq. (4.8)

Consider next the relation in Eq. (4.8), namely, that $\underline{X}^T \underline{A} \underline{X} = \underline{T}_q$ where $\underline{A} = \underline{M} \underline{K}^{-1} \underline{M}$ and $\underline{X}$ is the matrix storing all the generated vectors. It should be noted that since the matrix $\underline{A}$ is symmetric, it is sufficient if we consider only the lower triangular part of the matrix $\underline{T}_q$. Based on the theoretical $\underline{M}$-orthogonality of the generated vectors, it follows from Eq. (A.4) that

$$\underline{x}_j^T \underline{M} \underline{K}^{-1} \underline{M} \underline{x}_i = 0 \qquad j=1,\ldots,i-2 \tag{A.7}$$

Hence, the matrix $\underline{T}_q$ is tridiagonal. Also, we have from Eq. (A.4) corresponding to j= i-1,i,i+1,

$$\begin{aligned} \underline{x}_i^T \underline{M} \underline{K}^{-1} \underline{M} \underline{x}_{i-1} &= \beta_i \\ \underline{x}_i^T \underline{M} \underline{K}^{-1} \underline{M} \underline{x}_i &= \alpha_i \\ \underline{x}_i^T \underline{M} \underline{K}^{-1} \underline{M} \underline{x}_{i+1} &= \beta_{i+1} \end{aligned} \tag{A.8}$$

Thus, the relation given in Eq. (A.8) has been proved.

Consider the following relations derived from Eq. (A.1):

$$\beta_2 \underline{x}_2 = \underline{K}^{-1}\underline{M}\,\underline{x}_1 - \alpha_1 \underline{x}_1$$

$$\beta_3 \underline{x}_3 = \underline{K}^{-1}\underline{M}\,\underline{x}_2 - \alpha_2 \underline{x}_2 - \beta_2 \underline{x}_1$$

$$= \underline{K}^{-1}\underline{M}\,\underline{x}_2 - (\alpha_2/\beta_2)\underline{K}^{-1}\underline{M}\,\underline{x}_1 + (\alpha_1/\beta_1 - \beta_2)\underline{x}_1$$

$$= \frac{1}{\beta_2}(\underline{K}^{-1}\underline{M})^2 \underline{x}_1 - \frac{\alpha_1+\alpha_2}{\beta_2}\,\underline{K}^{-1}\underline{M}\,\underline{x}_1 + \frac{\alpha_1\alpha_2-\beta_2^2}{\beta_2}\,\underline{x}_1$$

$$\beta_4 \underline{x}_4 = \underline{K}^{-1}\underline{M}\,\underline{x}_3 - \alpha_3 \underline{x}_3 - \beta_3 \underline{x}_2$$

$$\vdots$$

From the above relations it is seen that the set of vectors $\underline{x}_1, \underline{x}_2, \underline{x}_3, \ldots$, form on $\underline{M}$-orthonormal basis to the space spanned by the vectors $\underline{x}_1, \underline{K}^{-1}\underline{M}\,\underline{x}_1$, $(\underline{K}^{-1}\underline{M})\underline{x}_2, \ldots$, and that the vectors $\underline{x}_1, \underline{K}^{-1}\underline{M}\,\underline{x}_1$, $(\underline{K}^{-1}\underline{M})^2\,\underline{x}_1, \ldots$, also span identically the same space. This fact has been made use of in the analysis presented in Section 4.3.

Characteristics of the Vectors Generated Using the Lanczos Method

In the description of the Lanczos method in Section 4.3, the starting vector $\underline{x}_1$ was normalized with respect to the matrix $\underline{M}$. The vector $\underline{x}_1$ can always be expressed in terms of the eigenvectors $\underline{\phi}_i$, $i=1,\ldots,n$, of Eq. (4.1), i.e.,

$$\underline{x}_1 = \sum_{i=1}^{n} C_i \underline{\phi}_i \qquad (A.9)$$

where $C_i$ is the component of the eigenvector $\underline{\phi}_i$ present in $\underline{x}_1$. Also, since the eigenvectors $\underline{\phi}_i$ are orthonormal with respect to the matrix $\underline{M}$, it follows that $\sum_{i=1}^{n} C_i^2 = 1$. In the following, it is shown that the vectors generated using Eqs. (4.2) to (4.7) become successively void in the components of the eigenvectors $\underline{\phi}_1$, $\underline{\phi}_2$, ..., etc.

Substituting Eq. (A.9) into the recurrence algorithm in Eqs. (4.2) to (4.7), we obtain

$$\bar{\underline{x}}_2 = \sum_{i=1}^{n} C_i \frac{1}{\lambda_i} \underline{\phi}_i \qquad (a)$$

$$\alpha_1 = \sum_{i=1}^{n} C_i^2 \frac{1}{\lambda_i} \qquad (b)$$

$$\tilde{\underline{x}}_2 = \sum_{i=1}^{n} \left(\frac{1}{\lambda_i} - \alpha_1\right) C_i \underline{\phi}_i \qquad (c)$$

$$\beta_2 = \left\{\left( \sum_{i=1}^{n} C_i^2 \frac{1}{\lambda_i^2} \right) - \alpha_1^2 \right\}^{1/2} \qquad (d)$$

$$\underline{x}_2 = \frac{1}{\beta_2} \sum_{i=1}^{n} \left(\frac{1}{\lambda_i} - \alpha_1\right) C_i \underline{\phi}_i \qquad (e)$$

Assuming that the eigenvalues of Eq. (4.1) are all discrete and well separated, i.e. $\lambda_1 < \lambda_2 < \lambda_3 \ldots < \lambda_n$, we obtain approximately,

$$\alpha_1 \doteq C_1^2/\lambda_1 \qquad (f)$$

$$\beta_2 \doteq C_1 (1-C_1^2)^{1/2} / \lambda_1 \qquad (g)$$

and the component $\bar{C}_i$ of eigenvector $\underline{\phi}_i$ present in $\underline{x}_2$ is approximately,

$$\bar{C}_i \doteq \frac{\lambda_1}{C_1(1-C_1^2)^{1/2}} \left(\frac{1}{\lambda_i} - \frac{C_1^2}{\lambda_1}\right) C_i \qquad (h)$$

Assuming further that the starting vector $\underline{x}_1$ consists of equal components of all the eigenvectors $\underline{\phi}_i$, i.e. $C_1 = C_2 = \ldots = C_n = \frac{1}{\sqrt{n}}$, we obtain from Eq. (h)

$$\begin{aligned} \bar{C}_1 &\doteq (1 - 1/n)^{1/2} \\ \bar{C}_i &\doteq -\frac{1}{n}(1 - \frac{1}{n})^{-1/2} \qquad i=2, \ldots, n \end{aligned} \qquad (k)$$

For large systems, $n >> 1$ and it follows from Eq. (k) that $\bar{C}_1 \doteq 1$ and $\bar{C}_i \doteq -\frac{1}{n}$, $i=2,\ldots n$. Hence the <u>dominant contribution</u> to the vector $\underline{x}_2$ is from the eigenvector $\underline{\phi}_1$.

Substituting the expressions for the vectors $\underline{x}_1$ and $\underline{x}_2$ from Eqs. (A.9) and (e) respectively into Eqs. (4.3) to (4.5), we obtain

$$\bar{\underline{x}}_3 = \frac{1}{\beta_2} \sum_{i=1}^{n} \left(\frac{1}{\lambda_i} - \alpha_1\right) \frac{C_i}{\lambda_i} \underline{\phi}_i \qquad (l)$$

$$\alpha_2 = \frac{1}{\beta_2^2} \sum_{i=1}^{n} \left(\frac{C_i^2}{\lambda_i^3} - \alpha_1^3\right) - 2\alpha_1 \qquad (m)$$

$$\tilde{\underline{x}}_3 = \frac{1}{\beta_2^3} \sum_{i=1}^{n} \left\{ \frac{\beta_2^2}{\lambda_i^2} + \frac{\alpha_1 \beta_2^2}{\lambda_i} - 2\beta_2^2 \alpha_1^2 - \beta_2^4 \right.$$

$$\left. - \sum_{i=1}^{n} \left(\frac{C_i^2}{\lambda_i^3} - \alpha_1^3\right)\left(\alpha_1 - \frac{1}{\lambda_i}\right) \right\} C_i \underline{\phi}_i \quad . \qquad (n)$$

Using the approximations for $\alpha_1$ and $\beta_2$ from Eqs. (f) and (g) and $C_i = \frac{1}{\sqrt{n}}$ in Eq. (n), we obtain

$$\tilde{\underline{x}}_3 \doteq \sum_{i=1}^{n} \left\{ \frac{1}{\lambda_i} \left[ \frac{\lambda_1}{\lambda_i} - 1 + \frac{2}{n} \right] \underline{\phi}_i \right. \qquad \text{(p)}$$

It is seen from Eq. (p) that the component of $\underline{\phi}_1$ in the vector $\tilde{\underline{x}}_3$ is negligible and that the dominant contribution to the vector $\tilde{\underline{x}}_3$ is from the eigenvector $\underline{\phi}_2$, when the eigenvalues are well separated. Therefore, we conclude that (1) the dominant contribution to the vector $\underline{x}_{i+1}$ is from the eigenvector $\underline{\phi}_i$ ; (2) as i is increased, the components of the eigenvectors $\underline{\phi}_1$, $\underline{\phi}_2$, ..., are removed one after the other from the generated vectors $\underline{x}_{i+1}$. Hence, the subspace spanned by the vectors $\underline{x}_i$, i = 1, ..., q, tends to be close to the least dominant q-dimensional subspace of the operators $\underline{K}$ and $\underline{M}$. Some numerical examples to illustrate the above discussion are given later.

Consider the following generalized eigenproblem

$$\underline{M}\,\underline{K}^{-1}\,\underline{M}\,\underline{\phi} = \kappa\,\underline{M}\,\underline{\phi} \qquad \text{(A.10)}$$

The eigenvectors of the above eigenproblem are the same as the eigenvectors of Eq. (4.1); but, the eigenvalue $\kappa_i$ of Eq. (A.10) is equal to $1/\lambda_i$, where $\lambda_i$ is an eigenvalue of Eq. (4.1). The matrix $\underline{T}_q$

in Eq. (4.8) is the projection of the matrix $\underline{M}\,\underline{K}^{-1}\,\underline{M}$ onto the subspace spanned by the generated vectors $\underline{x}_i$, i=1,...q, where theoretically the vectors $\underline{x}_i$ are $\underline{M}$-orthonormal. Consider next the following standard eigenproblem

$$\underline{T}_q\,\underline{Z} = \bar{\kappa}\,\underline{Z} \qquad . \qquad (A.11)$$

When q is equal to n (the order of the matrices $\underline{K}$ and $\underline{M}$), it follows that the eigenvalues $\bar{\kappa}_i$ of Eq. (A.11) are the same as the eigenvalues $\kappa_i$ of Eq. (A.10) and the elements of the matrix $\underline{Z}$ relate the eigenvectors $\underline{\phi}_i$ of Eq. (A.10) to the vectors $\underline{x}_j$ generated using the Lanczos method, i.e.,

$$\underline{\Phi} = \underline{X}\,\underline{Z} \qquad (A.12)$$

where $\underline{X} \equiv \{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n\}$. When q is much smaller than n, from the analysis presented above (Eqs. (a) to (p)) we conclude that the eigensolution of the matrix $\underline{T}_q$ in Eq. (A.11) yields good approximations to the smallest eigenvalues of Eq. (4.1). The approximations to the corresponding eigenvectors of Eq. (4.1) are obtained from Eq. (A.12). It should be noted here that the elements of the matrix $\underline{T}_q$ are obtained directly from Eqs. (4.4) and (4.6), during the generation of the vectors $\underline{x}_{i+1}$. Also, when q<n, the solution of the standard eigenproblem in Eq. (A.11), is relatively inexpensive, particularly since $\underline{T}_q$ is a tridiagonal matrix. Hence, the eigensolution using the Lanczos method is considerably faster than the solution using

other methods, particularly when the matrices $\underline{K}$ and $\underline{M}$ can be kept in-core.

Solution of Small Eigenproblems Using the Lanczos Method

The eigensolutions of three small eigenproblems using the Lanczos method are presented next.

The starting vector $\underline{x}_1$ is a full unit vector, i.e. $\underline{x}_1 = [1, 1, \ldots, 1]^T$ and the vectors $\underline{x}_2, \ldots, \underline{x}_q$ are obtained using the recurrence algorithm in Eqs. (4.2) to (4.7) with the reorthogonalization given in Eqs. (4.16) and (4.17). The matrix $\underline{T}_q$ given in Eq. (4.9) was formed and the eigenvalues and eigenvectors of the matrix $\underline{T}_q$ were computed using the Jacobi method.

A diagonal matrix of order 5

$$\underline{K} = \begin{bmatrix} 1 & & & & 0 \\ & 3 & & & \\ & & 5 & & \\ & & & 4 & \\ 0 & & & & 2 \end{bmatrix} \qquad \underline{M} = \underline{I} \qquad \text{(i)}$$

The eigenvalues $\lambda_i$, $i = 1, \ldots, 5$, are equal to 1, 2, 3, 4, 5 respectively and the corresponding eigenvectors are the unit coordinate vectors $\underline{e}_j$, $j = 1, 5, 2, 4, 3$, respectively.

Let

$$\underline{x}_1 = [\, 1/\sqrt{5},\ 1/\sqrt{5},\ 1/\sqrt{5},\ 1/\sqrt{5},\ 1/\sqrt{5}]^T \quad . \qquad \text{(ii)}$$

It should be noted that the vector $\underline{x}_1$ has equal components of all the eigenvectors of Eq. (i) and that $\underline{x}_1^T \underline{M}\, \underline{x}_1 = 1$. The vectors $\underline{x}_{i+1}$, i=1, ..,4, generated using the recurrence algorithm in Eqs. (4.2) to (4.7) are,

$$\underline{X} \equiv \begin{bmatrix} \underline{x}_1 & \underline{x}_2 & \underline{x}_3 & \underline{x}_4 & \underline{x}_5 \\ 1/\sqrt{5} & \underline{0.837} & 0.311 & 0.046 & 0.003 \\ 1/\sqrt{5} & -0.190 & -0.236 & \underline{0.683} & 0.492 \\ 1/\sqrt{5} & -0.396 & 0.493 & -0.506 & 0.380 \\ 1/\sqrt{5} & -0.319 & 0.187 & 0.243 & \underline{-0.778} \\ 1/\sqrt{5} & 0.067 & \underline{-0.755} & -0.465 & -0.097 \end{bmatrix} \qquad \text{(iii)}$$

The matrix $\underline{T}_q$, for q equal to 5, is

$$\underline{T}_q \equiv \begin{bmatrix} 0.4567 & 0.2901 & & & \\ 0.2901 & 0.7724 & 0.1960 & & \\ & 0.1960 & 0.4573 & 0.0975 & \\ & & 0.0975 & 0.3317 & 0.0490 \\ & & & 0.0490 & 0.2654 \end{bmatrix} \qquad \text{(iv)}$$

In the analysis of the Lanczos method presented earlier, we pointed out that when the eigenvalues of the eigenproblem being solved are discrete and well spread out, the dominant contribution to the generated vector $\underline{x}_{i+1}$ is from the eigenvector $\underline{\phi}_i$. For this example problem the eigenvectors are the unit coordinate vectors $\underline{e}_j$ and the component of the vector $\underline{e}_j$ present in the generated vector $\underline{x}_{i+1}$ is simply the corresponding $j^{th}$ element of the vector $\underline{x}_{i+1}$. The largest component of each vector $\underline{x}_{i+1}$ has been underlined in Eq. (iii) and it is seen that the above conclusion holds good in this case.

The eigensolutions of the matrices $\underline{T}_q$ for q = 1, 2 and 3 yield the following eigenvalue approximations:

For q = 1, $\kappa_1 = 0.4567$ and $\bar{\lambda}_1 = 2.19$, where $\kappa_1$ is the eigenvalue of the 1 x 1 matrix $\underline{T}_q$ and $\bar{\lambda}_1$ is the corresponding approximation to the first eigenvalue of Eq. (i), equal to $1/\kappa_1$.

Similarly for q = 2,

$$\kappa_1 = 0.9448 \; ; \quad \bar{\lambda}_1 = 1.058$$

$$\kappa_2 = 0.2842 \; ; \quad \bar{\lambda}_2 = 3.518$$

Also, computing the eigenvector of $\underline{T}_q$ corresponding to $\kappa_1$ in this case, we find that the approximation to the eigenvector $\underline{\phi}_1$ of Eq. (i) is $(0.51\ \underline{x}_1 + 0.86\ \underline{x}_2)$.

For the case q = 3,

$$\kappa_1 = 0.9986 \; ; \quad \bar{\lambda}_1 = 1.0014$$

$$\kappa_2 = 0.4571 \; ; \quad \bar{\lambda}_2 = 2.1877$$

$$\kappa_3 = 0.2306 \; ; \quad \bar{\lambda}_3 = 4.3374$$

and the approximation to the eigenvector $\underline{\phi}_1$ is $(0.45\ \underline{x}_1 + 0.84\ \underline{x}_2 + 0.3\underline{x}_3)$. Also, the approximation to the eigenvector $\underline{\phi}_2$ is $(0.56\ \underline{x}_1 + 0.0008\ \underline{x}_2 + 0.83\ \underline{x}_3)$.

Comparing the cases q = 2 and q = 3, we find that the improvement in the accuracy of the approximation to the eigenpair $\lambda_1$ and $\underline{\phi}_1$ is already small when q is increased from 2 to 3.

Analysis of a cantilever beam

The stiffness matrix and the diagonal mass matrix of the cantilever beam shown in Fig. A.1, were generated using the computer program ADINA [41]. The smallest eight eigenvalues were computed using the Lanczos method, with q = 16, 20 and 24. The results obtained are shown in Table A.1. The computed values of the factors $\alpha_i$ (Eq. (4.4)), $\beta_{i+1}$ (Eq. (4.6)), $\gamma_{i+1}$ (Eq. (4.18)) and the Rayleigh quotient $\rho_{i+1}$ of the vectors $\bar{\underline{x}}_{i+1}$ (equal to $\alpha_i/\gamma_{i+1}^2$), for i = 1, ..., 23, are given in Table A.2.

From Table A.2, it is seen that the Rayleigh quotient of the vectors $\bar{\underline{x}}_{i+1}$ increases as $i$ is increased implying that the components of the eigenvectors associated with the smallest eigenvalues are successively removed from the generated vectors $\underline{x}_{i+1}$.

The diagonal mass matrix of this beam contains non-zero elements corresponding to only the translational degrees of freedom. Hence, there are only 24 non-zero entries in the mass matrix and there are only 24 finite eigenvalues; 16 of them correspond to bending modes and 8 correspond to axial modes. The starting vector for the Lanczos method is chosen as a full unit vector. It is seen from the Fig. A.1 that the starting vector contains only 8 of the 16 bending modes and the 8 axial modes. In Table A.2 this observation is substantiated by the fact that $\beta_{17}$ is extremely small (equal to $\gamma_{17} * 10^{-10}$), implying that the generated vector $\underline{x}_{17}$ is a null vector. Hence the vectors $\underline{x}_{18}$ to $\underline{x}_{24}$ are generated from a new vector $\underline{x}_{17}$. For the case $q = 24$, all the 24 finite eigenvalues and the corresponding eigenvectors were predicted correctly using the Lanczos method.

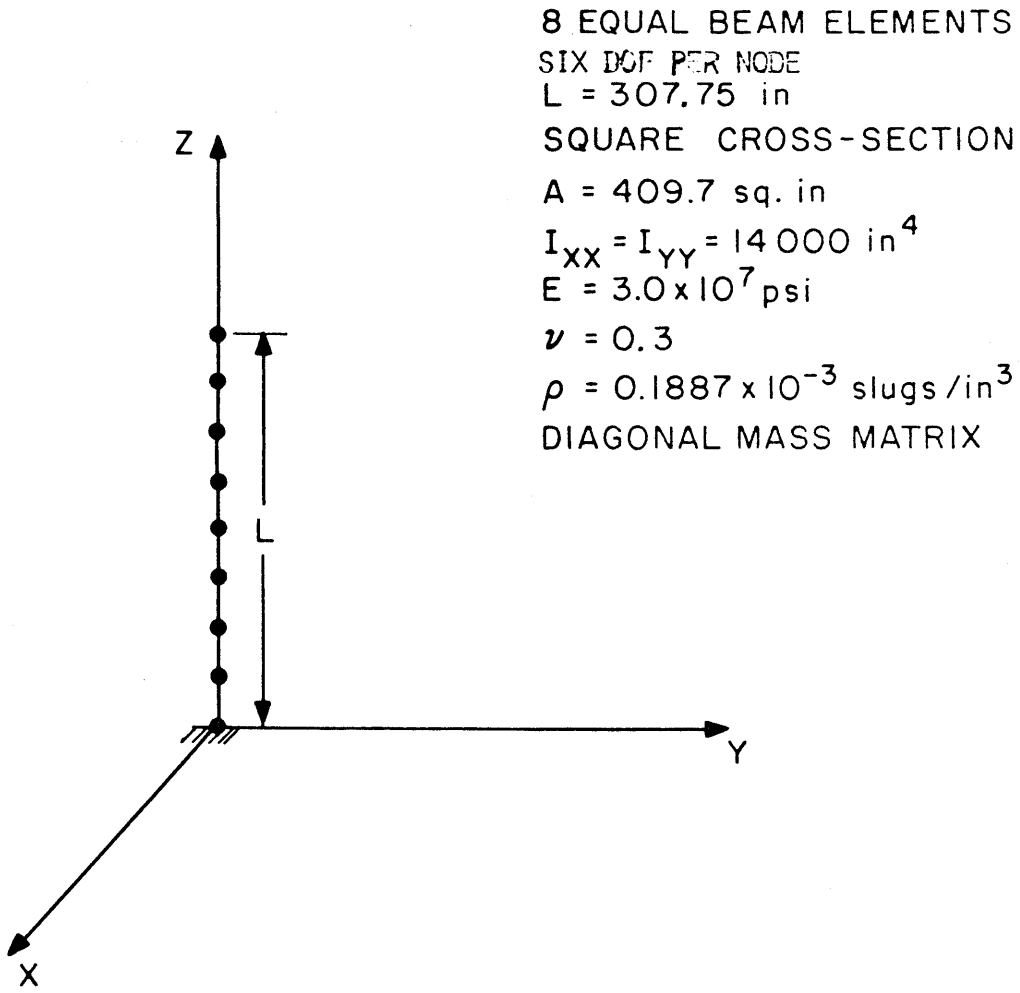



FIGURE A.1 A CANTILEVER BEAM

Order of the matrices (n) = 48; mean half bandwidth ($m_{\underline{K}}$) = 9

TABLE A.1 THE SMALLEST EIGHT EIGENVALUE APPROXIMATIONS CALCULATED USING THE LANCZOS METHOD IN THE ANALYSIS OF THE CANTILEVER BEAM

| q = 16 | q = 20 | q = 24 |
|---|---|---|
| $0.738101 \times 10^4$ | $0.738101 \times 10^4$ | $0.738101 \times 10^4$ |
| $0.280001 \times 10^6$ | $0.738101 \times 10^4$ | $0.738101 \times 10^4$ |
| $0.212824 \times 10^7$ | $0.280001 \times 10^6$ | $0.280001 \times 10^6$ |
| $0.412855 \times 10^7$ | $0.280033 \times 10^6$ | $0.280001 \times 10^6$ |
| $0.791162 \times 10^7$ | $0.212824 \times 10^7$ | $0.212824 \times 10^7$ |
| $0.208164 \times 10^8$ | $0.221162 \times 10^7$ | $0.212824 \times 10^7$ |
| $0.362111 \times 10^8$ | $0.412855 \times 10^7$ | $0.412855 \times 10^7$ |
| $0.437700 \times 10^8$ | $0.791162 \times 10^7$ | $0.791162 \times 10^7$ |

The $7^{th}$, $12^{th}$, $17^{th}$ and $20^{th}$ to $24^{th}$, eigenvalues correspond to the axial vibration modes.

TABLE A.2 SOLUTION STEPS IN THE ANALYSIS OF THE CANTILEVER BEAM USING THE LANCZOS METHOD

| $i+1$ | $\alpha_i$ | $\beta_{i+1}$ | $\gamma_{i+1} = (\alpha_i+\beta_{i+1}^2+\beta_i^2)^{1/2}$ | $\rho_{i+1} = (\alpha_i/\gamma_{i+1}^2)$ |
|---|---|---|---|---|
| 2 | .59288E-04 | .66643E-04 | .89199E-04 | .74517E+04 |
| 3 | .77159E-04 | .21400E-05 | .10198E-03 | .74196E+04 |
| 4 | .28051E-05 | .24724E-06 | .35369E-05 | .22424E+06 |
| 5 | .23953E-06 | .18043E-06 | .38866E-06 | .15857E+07 |
| 6 | .29197E-06 | .50846E-07 | .34697E-06 | .24252E+07 |
| 7 | .10488E-06 | .43242E-07 | .12432E-06 | .67862E+07 |
| 8 | .49558E-07 | .16782E-07 | .67879E-07 | .10756E+08 |
| 9 | .32432E-07 | .93439E-08 | .37693E-07 | .22827E+08 |
| 10 | .13080E-07 | .55114E-08 | .16993E-07 | .45296E+08 |
| 11 | .18228E-07 | .64011E-08 | .20090E-07 | .45163E+08 |
| 12 | .10358E-07 | .37891E-08 | .12752E-07 | .63695E+08 |
| 13 | .96111E-08 | .20790E-08 | .10538E-07 | .86546E+08 |
| 14 | .49910E-08 | .19608E-08 | .57513E-08 | .15089E+09 |
| 15 | .84485E-08 | .14972E-08 | .88014E-08 | .10906E+09 |
| 16 | .33091E-08 | .15603E-09 | .36354E-08 | .25039E+09 |
| 17 | .24262E-08 | .10589E-18 | .24312E-08 | .41047E+09 |
| 17 | | .12194E+01 | .12194E+01 | |
| 18 | .29024E-04 | .55400E-04 | .62543E-04 | .74200E+04 |
| 19 | .10665E-03 | .99771E-06 | .12018E-03 | .73837E+04 |
| 20 | .30077E-05 | .11717E-05 | .33786E-05 | .26349E+06 |
| 21 | .86003E-06 | .95047E-07 | .14566E-05 | .40536E+06 |
| 22 | .12670E-06 | .12957E-07 | .15892E-06 | .50169E+07 |
| 23 | .21073E-07 | .62478E-08 | .25515E-07 | .32371E+08 |
| 24 | .34765E-07 | .16738E-07 | .39087E-07 | .22755E+08 |

For this example, the choice of the full unit vector for the starting vector $\underline{x}_1$, leads to an inaccurate eigensolution using the Lanczos method when q<24. Using the full unit vector, only the bending modes in which the X and Y translational displacements have equal magnitudes and same sign are obtained because the starting vector is orthogonal to all the bending modes in which the X and Y translational displacements have equal magnitudes but opposite signs. For symmetric structures with multiple frequencies, the choice of a random vector for the vector $\underline{x}_1$ alleviates the above problem. However, this example clearly illustrates a shortcoming of the Lanczos method, namely, that the solution results are dependent to a large extent on the spectral content of the starting vector.

Analysis of a structure with a rigid body mode

A finite element model of a structure that is allowed to move as a rigid body along the X-direction is shown in Fig. A.2. The stiffness matrix and the mass matrix were computed using the program ADINA [41]. The value of the rigid body mode shift applied is $-0.125 * 10^7$. The calculated eigenvalues of this eigenproblem are clustered together, as seen from Table A.3.

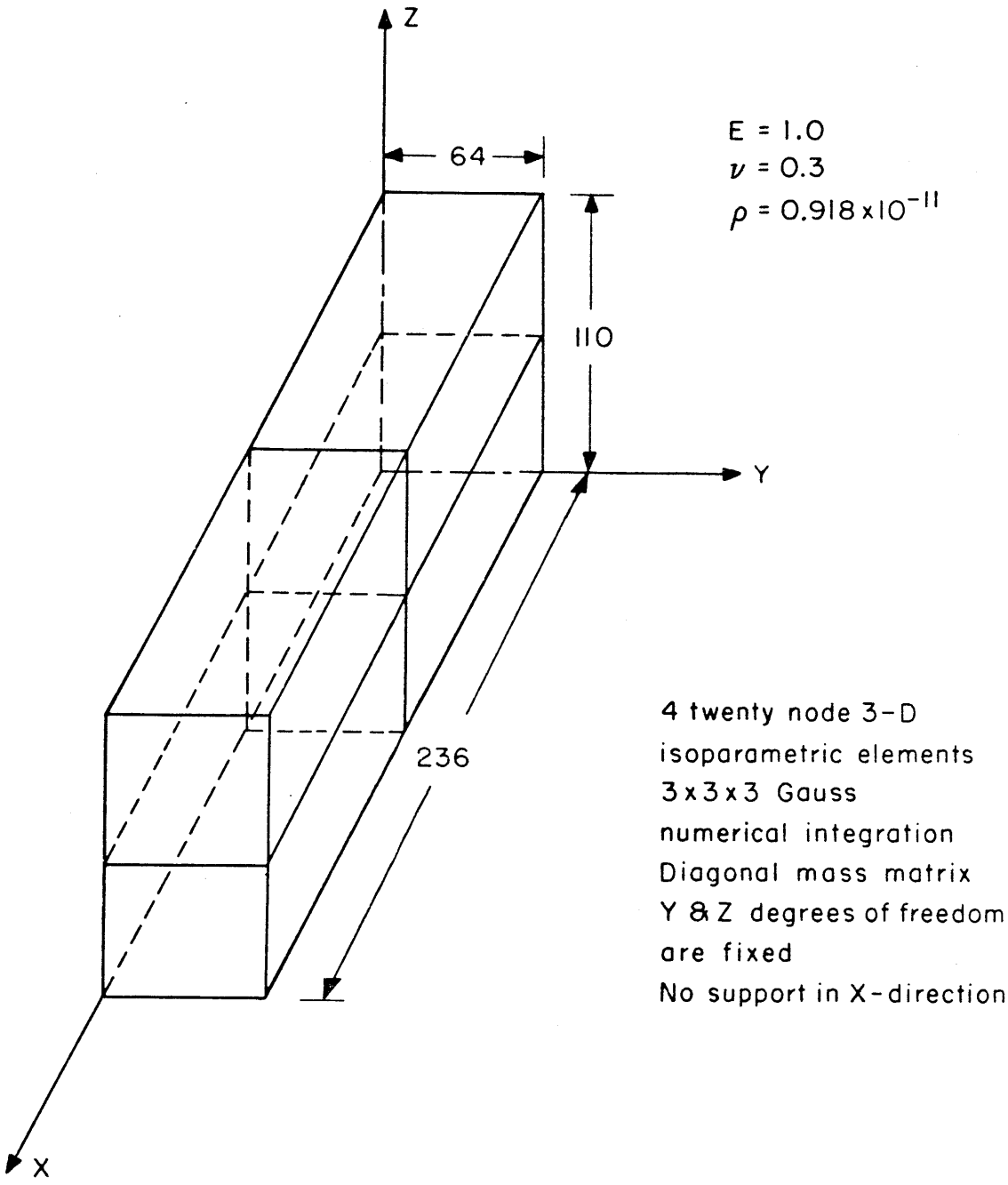



FIGURE A.2 A STRUCTURE WITH A RIGID BODY MODE

Order of matrices (n)=51; mean half bandwidth ($m_K$)=22

TABLE A.3 THE SMALLEST SIX EIGENVALUE APPROXIMATIONS CALCULATED USING THE LANCZOS METHOD IN THE ANALYSIS OF THE STRUCTURE WITH A RIGID BODY MODE

| q = 12 | q = 16 | q = 22 | q = 30 |
|---|---|---|---|
| $0.125000 \times 10^{7}$ | $0.125000 \times 10^{7}$ | $0.125000 \times 10^{7}$ | $0.125000 \times 10^{7}$ |
| $0.247458 \times 10^{8}$ | $0.247458 \times 10^{8}$ | $0.247458 \times 10^{8}$ | $0.205598 \times 10^{8}$ |
| $0.322807 \times 10^{8}$ | $0.322244 \times 10^{8}$ | $0.304545 \times 10^{8}$ | $0.247458 \times 10^{8}$ |
| $0.327948 \times 10^{8}$ | $0.325269 \times 10^{8}$ | $0.322049 \times 10^{8}$ | $0.321924 \times 10^{8}$ |
| $0.344979 \times 10^{8}$ | $0.341909 \times 10^{8}$ | $0.323329 \times 10^{8}$ | $0.322249 \times 10^{8}$ |
| $0.362185 \times 10^{8}$ | $0.344780 \times 10^{8}$ | $0.325610 \times 10^{8}$ | $0.323446 \times 10^{8}$ |

The smallest six eigenvalues were computed using the Lanczos method, with $q = 12, 16, 22$ and 30, and the results obtained are given in Table A.3. Based on Sturm sequence check and "physical" error bound calculation (see Eq. (3.7) and (3.8)), it was found that only in the solution using $q = 30$, the smallest six eigenvalues are predicted correctly. The solution obtained with a random vector as the vector $\underline{x}_1$ and $q = 22$, was also inaccurate. It is seen from this example that the value of $q$, namely, the number of vectors generated using the Lanczos method, must be considerably larger than $p$, namely, the number of eigenpairs required, particularly when the required eigenvalues are clustered together.

## APPENDIX B - THE COMPUTER PROGRAM MSUBSP

The computer program presented in this appendix uses the accelerated subspace iteration method described in Chapters 5 and 6 to evaluate the eigenvalues in any specified interval and the associated eigenvectors of the generalized eigenproblem $\underline{K}\,\underline{\phi} = \lambda \underline{M}\,\underline{\phi}$. The matrix $\underline{K}$ can be positive definite or positive semidefinite and the matrix $\underline{M}$ can be banded positive definite or diagonal non-negative definite.

The program organization, input, output, a detailed user's manual and a listing of the computer program are presented.

### Program organization

In the program the entire stiffness and mass matrices and q iteration vectors are stored in the blank common; i.e., in high-speed core storage. A list of all the arrays used in the program and their storage locations in the blank common are given in Fig. B.1. The stiffness matrix and the banded mass matrix (if used) are assumed to be stored as single column vectors, employing the skyline storage scheme [1, p. 211].

The size of the blank common area is given by the value of the variable MTOT in the main program. If the storage required in the blank common area exceeds the value of MTOT for a data case, the program is not executed and an error message (indicating the total storage required) is printed.

Five sequential access read/write units are required during the

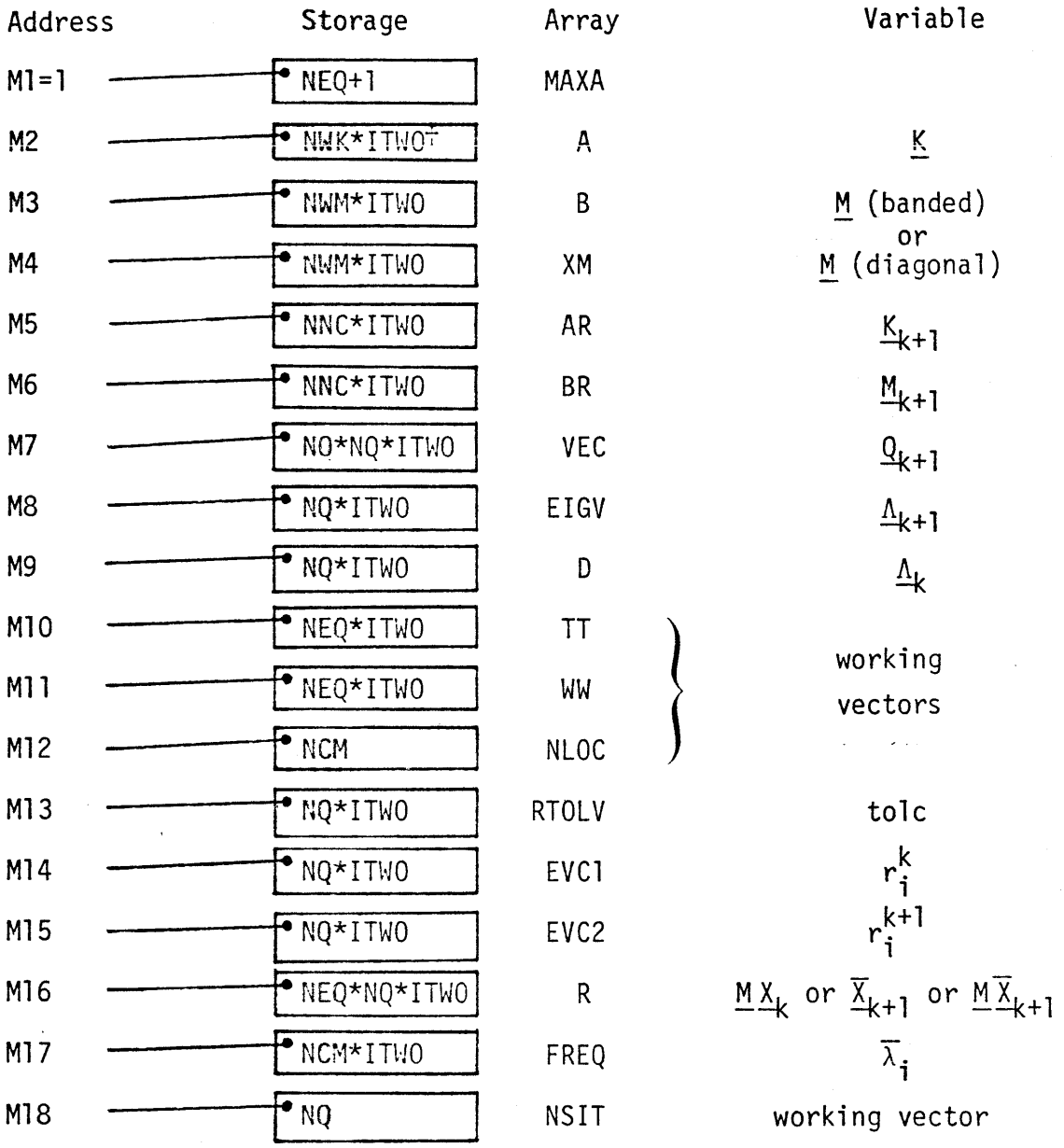



FIGURE B.1 STORAGE ALLOCATION IN PROGRAM MSUBSP.

NEQ, NWK, NWM, NFREQ and NQ are input variables; NNC=NQ*(NQ+1)/2 and NCM = larger of {NQ, min. (2*NFREQ, NFREQ+8)}.

† ITWO=1 in single precision arithmetic and ITWO=2 in double precision arithmetic.

execution of the program (see Table B.1).

Program input

For each data case the following user provided information is read from Unit 3 (NREAD), using four unformatted read statements

```
READ (NREAD) NEQ, NWK, NWM
READ (NREAD) (MAXA(I), I=1, NEQ1)
READ (NREAD) (A(I), I=1, NWK)
READ (NREAD) (B(I), I=1, NWM)
```

where

NEQ = Order of the stiffness and mass matrices

NWK = Number of elements below skyline of stiffness matrix
(see ref. [1, p. 211])

NWM = Number of elements below skyline of mass matrix
i.e. NWM=NWK for banded mass matrix
NWM=NEQ for diagonal mass matrix

NEQ1 = NEQ+1, Length of MAXA array

A = Stiffness matrix

B = Mass matrix

In addition, the card input as described in the user's manual is also required.

Program output

At completion of solution, "physical" error bounds for each eigenvalue and eigenvector are calculated using $\|(\underline{K}-\omega^2\underline{M})\underline{\phi}\|_2 / \|\underline{K}\ \underline{\phi}\|_2$, where

| Unit Number | File Name | Usage |
|---|---|---|
| 3 | NREAD | Input file containing header information, MAXA array, stiffness and mass matrices |
| 4 | NSTIF | Stores the stiffness matrix during the execution |
| 7 | NOVER | Stores iteration vectors for overrelaxation |
| 8 | NSHIFT | Stores user input starting iteration vectors; required also during the generation of starting iteration vectors using the Lanczos method |
| 9 | NT | Stores the converged iteration vectors and eigenvalue approximations |
| 5 | IIN | Card reader |
| 6 | IOUT | Line printer |

TABLE B.1 READ/WRITE UNITS REQUIRED DURING THE EXECUTION OF THE COMPUTER PROGRAM

$(\omega, \underline{\phi})$ is a calculated eigenpair, and are printed.

After solution the eigenvalues and eigenvectors are stored on UNIT 9 and can be retrieved with the following format

```
   REWIND 9
   DØ XX L = 1, NFREQ
XX Read (9) (PHI(I,L), I = 1, NEQ)
   Read (9) (ROOT(I), I = 1, NFREQ)
```

User's manual for the computer program MSUBSP

Card 1 (12A6)

| Note | Columns | Variable | Entry |
|---|---|---|---|
| (1) | 1 - 72 | HED(12) | Enter the master heading information for use in labeling the output |

NOTES/

(1) Several data cases can be run back-to-back. Begin each new data case with a new heading card. The characters STOP in the first four columns of this card stops the execution.

Card 2 (3I5,F10.0,I5)

| Note | Columns | Variable | Entry |
|---|---|---|---|
| | 1 - 5 | NFREQ | Number of eigenpairs required;<br>EQ.0, set to 1 |
| (1) | 6 - 10 | NQ | Number of iteration vectors to be used;<br>EQ.0, set to min. {2*NFREQ,NFREQ+8} |
| (2) | 11 - 15 | IRBM | Flag indicating presence of rigid body modes;<br>EQ.0, no rigid body modes present |
| (2) | 16 - 25 | RBMSH | Rigid body mode shift to be applied; must be less than or equal to zero;<br>EQ.0, program calculates a value |
| | 26 - 30 | IFPR | Intermediate print-out control parameter;<br>EQ.0, no printing<br>EQ.1, print details of solution behavior<br>EQ.2, print extensive debug information |

NOTES/

(1) The number of vectors to be employed in a subspace iteration solution can be smaller or larger than the number of required eigenvalues and vectors. During the iteration all NQ iteration vectors are kept in high speed core; hence a smaller number of vectors results in less high speed core requirements. But if NQ.LT.(min {2*NFREQ,NFREQ+8}) the acceleration schemes of the iteration are always used.

(2) The program can be used to compute frequencies of zero magnitude and associated mode shapes. If in the eigenproblem being solved, zero frequencies are present, the flag IRBM must be set to 1. Otherwise, the solution is terminated with an error message.

NOTES/

(2) (Continued)

If IRBM.GT.0, the number of non-zero frequencies calculated by the program is equal to the total number of frequencies calculated (NFREQ) minus the number of zero frequencies that are actually calculated.

If IRBM.GT.0 and RBMSH.EQ.0, the program computes the following value for the rigid body mode shift to be applied

$$\text{RBMSH} = \left( \min_{i=1,\ldots,\text{NEQ}} \left( \frac{k_{ii}}{m_{ii}} \right) \right) * 10^{-3}$$

where $k_{ii}$ and $m_{ii}$ are the $i^{th}$ diagonal entries in the stiffness and mass matrices and NEQ is the order of the matrices.

Card 3 (3I5,F10.0,3I5,2F10.0)

| Note | Columns | Variable | Entry |
|---|---|---|---|
| (1) | 1 - 5 | NITEM | Number of subspace iterations allowed;<br>EQ.0, set to 24 |
| (2) | 6 - 10 | IFSS | Sturm sequence check control parameter;<br>EQ.0, check not performed<br>EQ.1, check performed |
| (3) | 11 - 15 | IACCN | Flag for applying acceleration procedures;<br>EQ.0, basic subspace iteration method<br>EQ.1, accelerated subspace iteration method |
| | 16 - 25 | RTOL | Converge tolerance on eigenvalues;<br>EQ.0, set to 1.E-06 |

Card 3 (Continued)

| Note | Columns | Variable | Entry |
|---|---|---|---|
| (4) | 26 - 30 | ISVTYP | Flag to generate starting iteration vectors;<br>EQ.0, standard procedure used<br>EQ.1, Lanczos method used |
| (5) | 31 - 35 | NSTV | Number of user provided starting vectors;<br>GT.0, NSTV starting vectors are read from tape 8 |
| (6) | 36 - 40 | IINTER | Flag for intermediate eigenpairs calculation;<br>EQ.0, NFREQ lowest eigenvalues are calculated<br>GT.0, calculate all eigenvalues between shifts 1 and 2 |
| | 41 - 50 | SHIFT1 | Lower limit on frequencies to be calculated, in cycles/sec.; GE.0. |
| | 51 - 60 | SHIFT2 | Upper limit on frequencies to be calculated, in cycles/sec.; GE.0;<br>EQ.0, set to 1.E08 |

NOTES/

(1) The number of subspace iterations required for convergence depends on the convergence tolerance specified, and must be greater than the default value (24) if RTOL is smaller than its default value (1.E-06). If NQ is smaller than the default value in Card 2, during the solution some of the converged iteration vectors may be removed to back-up storage. In such case, NITEM refers to the number of subspace iterations allowed with each set of NQ iteration vectors.

NOTES/ (Continued)

(2) The Sturm sequence check can be used to verify that indeed the smallest required eigenvalues and corresponding eigenvectors have been calculated. Normally, the required eigenpairs are computed provided the number of iteration vectors is large enough (the variable NQ on Card 2) and the tolerance for convergence, RTOL, is small enough, thus enforcing a large enough number of vector iterations. However, in theory, an eigenpair may be missed and whether an eigenpair has been missed is detected by the Sturm sequence check. The cost of the Sturm sequence check is approximately equal to the cost of the triangular factorization of the stiffness matrix (the cost of a static solution), and, in practice, the check may be dispensed with if by experience the computed eigenpairs are known to correspond to the required values.

(3) If IACCN.EQ.1, the accelerated subspace iteration method (with shifting and overrelaxation) is employed. The acceleration schemes are generally effective, if the ordinary subspace iteration solution shows slow convergence, the eigenpairs need be calculated to high precision, or a small number of iteration vectors are used.

If NQ.LT. (min. {2*NFREQ, NFREQ+8}, the program always uses the acceleration schemes, i.e. IACCN is set to "1".

(4) Usually, the use of the standard starting subspace gives an effective solution of the eigenproblem. However, in some cases using the Lanczos technique can be efficient in establishing the starting vectors. In this case, the number of iteration vectors

used should be considerably larger than the eigenpairs required (say NQ should be about 2*NFREQ on card 2).

(5) This option is employed effectively when an eigensolution was already performed on a structure identical or similar to the one that is considered in this analysis.

The effectiveness of the subspace iteration solution can then usually be increased by providing the already calculated eigenvectors as the starting iteration vectors. The NSTV vectors are read from UNIT 8 with the following format

```
      REWIND 8
      DØ XX L=1,NSTV
   XX READ (8) (V(I,L),I=1,NEQ)
```

where NEQ is the order of the matrices. If NSTV is smaller than the default value for NQ (see card 2), the vectors read from UNIT 8 replace the first NSTV starting iteration vectors that are usually generated and used by the program.

(6) If IINTER.EQ.0, the lowest NFREQ vibration mode shapes and frequencies are evaluated. Frequencies are extracted in numerically ascending sequence, and unless the "N-th" frequency is > SHIFT2, the program will continue the eigensolution until all "NFREQ" modes have been determined. If the N'th frequency is > SHIFT2, where $1 \leq N \leq$ NFREQ, the eigensolution is discontinued, and only the lowest "N" frequencies are printed.

If IINTER.GT.0, the variable NFREQ does not determine the number

NOTES/

(Continued)

of frequencies to be calculated. Instead, frequencies greater than SHIFT1 are extracted in numerically ascending sequence until the "N-th" frequency calculated is > SHIFT 2. If the N-th frequency is < SHIFT2, but N > 50 storage overflow occurs and the eigensolution is discontinued. In such case, the 50 frequencies in the specified interval (SHIFT1, SHIFT2) that have been computed, are printed.

```
      PROGRAM MSUBSP (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,
     1                TAPE3, TAPE4, TAPE7, TAPE8, TAPE9 )
C
C     MAIN PROGRAM TO READ IN CONTROL PARAMETERS AND ALLOCATE STORAGE
C
C *IBM*       IMPLICIT REAL*8 (A-H,O-Z)
C *IBM*       ABS(X)=DABS(X)
C *IBM*       SQRT(X)=DSQRT(X)
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C .       THIS PROGRAM IS USED IN SINGLE PRECISION ARITHMETIC ON     .
C .       CDC EQUIPMENT AND DOUBLE PRECISION ARITHMETIC ON IBM       .
C .       OR UNIVAC MACHINES. ACTIVATE,DEACTIVATE OR ADJUST ABOVE    .
C .       CARDS FOR SINGLE OR DOUBLE PRECISION ARITHMETIC            .
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      COMMON /DIMSSP/ M1,M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12,M13,M14,M15
      COMMON /SOL/ NEQ,NWK,NWM,MA,IMASS,ISTOH
      COMMON /TAPES/ NREAD,NSTIF,IIN,IOUT,NOVER,NSHIFT,NT
      COMMON /ETGIF/ RBMSH,NEREQ,IRBM,IFPR
      COMMON /TOLS/ RTOL,ALPHA,CTOL,RGTOL
      COMMON /SCRAP/ SHIFT,RLQ1,NLQ,NITE,NSTEP,JR,ICONV,NCEV,NP,NQ,IFSS
      COMMON /FAST/ SHIFT1,SHIFT2,IOVER,IRPC,NSTV,IINTER,JROLD,NSCH1,
     1              IACCN,NJUNK,ISVTYP
      COMMON /ITELMT/ NSMAX,NITEM,NITEMM,NOVM
      COMMON /JUNK/ HED(12),MTOT
      COMMON /DPR/ TTWO
C
      COMMON A(20000)
      REAL A
      DIMENSION IA(1)
      EQUIVALENCE (IA(1),A(1))
      DATA FINAL /6HSTOP  /
C
      MTOT=20000
      ITWO=1
C *IBM* ITWO=2
      NREAD=3
      NSTIF=4
      IIN=5
      IOUT=6
      NOVER=7
      NSHIFT=8
      NT=9
C
      REWIND NREAD
   20 READ (IIN,1000) HED
      IF (HED(1).EQ.FINAL) STOP
C
C     READ MAXA, STIFFNESS AND MASS ARRAYS FROM NREAD (NREAD=3)
C
      READ (NREAD) NEQ,NWK,NWM
      ISTOH=NWK
      IMASS=1
      IF (NWM.EQ.NWK) IMASS=2
      NEQ1=NEQ + 1
      M1=1
      M2=M1 + NEQ1
```

```
      M3=M2 + NWK*ITWO
      M4=M3
60    IF (IMASS.EQ.2) M4=M3 + NWM*ITWO
      M5=M4
      IF (IMASS.EQ.1) M5=M4 + NWM*ITWO
C
      NN=M2 - 1
65    READ (NREAD) (IA(I),I=M1,NN)
      NN=M3 - 1
      READ (NREAD) (A(I),I=M2,NN)
      REWIND NSTIF
      WRITE (NSTIF) (A(I),I=M2,NN)
70    NN=M5 - 1
      READ (NREAD) (A(I),I=M3,NN)
C
      WRITE (IOUT,2000) HED,NEQ,NWK,NWM
C
75    C     INPUT FREQUENCY CONTROL DATA
      C
      READ (IIN,1020) NFREQ,NQ,IRBM,RBMSH,IFPR
      READ (IIN,1020) NITEM,IFSS,IACCN,RTOL,ISVTYP,NSTV,
     1                IINTER,SHIFT1,SHIFT2
80    IF (NFREQ.EQ.0) NFREQ=1
      NP=MIN0(2*NFREQ,NFREQ + 8)
      IF (NQ.EQ.0) NQ=NP
      IF (NQ.LT.NP) IACCN=1
      IF (NQ.GT.NEQ) NQ=NEQ
85    IF (NITEM.EQ.0) NITEM=24
      IF (RTOL.EQ.0.) RTOL=1.E-6
      IF (SHIFT2.EQ.0.) SHIFT2=1.EC8
      PI=4.*ATAN(1.)
      SHIFT1=(2.*PI*SHIFT1)**2
90    SHIFT2=(2.*PI*SHIFT2)**2
      NITEMM=NITEM
      NSMAX=24
      ALPHA=1.
      CTOL=0.33
95    IOVER=1
      IPPC=0
      IF (IACCN.EQ.1) IPPC=1
      NCM=NQ
      IF (NQ.LT.NP) NCM=MINO(NFREQ + NQ/2 + 1,NFREQ + 8)
00    IF (IINTER.GT.0) NCM=NQ + 50
      IF (NCM.GT.NEQ) NCM=NEQ
C
C     IF STIFFNESS MATRIX IS NOT POS. DEFINITE, APPLY A NEGATIVE SHIFT
C
05    CALL RBSHFT (A(M1),A(M2),A(M3),A(M4),IRBM,RBMSH,NEQ,ISTOH,IMASS)
C
      WRITE (IOUT,2050) NFREQ,NQ,IRBM,RBMSH,IFPR
      WRITE (IOUT,2100) NITEM,IFSS,IACCN,RTOL,ISVTYP,NSTV
      WRITE (IOUT,2200) IINTER,SHIFT1,SHIFT2
10    IF (RTOL.GT.1.E-6) WRITE (IOUT,3000) RTOL
C
C     SET UP STORAGE LOCATIONS FOR THIS CASE
C
      NC=NQ
```

```
      NNC=NC*(NC + 1)/2
      M6=M5 + NNC*ITWO
      M7=M6 + NNC*ITWO
      M8=M7 + NC*NC*ITWO
      M9=M8 + NC*ITWO
      M10=M9 + NC*ITWO
      M11=M10 + NEQ*ITWO
      M12=M11 + NEQ*ITWO
      M13=M12 + NCM
      M14=M13 + NC*ITWO
      M15=M14 + NC*ITWO
      M16=M15 + NC*ITWO
      M17=M16 + NEQ*NC*ITWO
      M18=M17 + NCM*ITWO
      M19=M18 + NC
      WRITE (IOUT,2300) M19
      IF (M19.LE.MTOT) GO TO 100
      WRITE (IOUT,3100) MTOT,M19
      STOP
C
  100 CALL SSPACE (A(M1),A(M2),A(M3),A(M4),A(M5),A(M6),
     1             A(M7),A(M8),A(M9),A(M10),A(M11),A(M12),A(M13),A(M14),
     2             A(M15),A(M16),A(M17),A(M18),A(M11),A(M16),
     3             A(M18),NEQ,NFREQ,NCM,ISTOH)
C
      GO TO 20
C
C *IBM*       RETURN
C
 1000 FORMAT (12A6)
 1020 FORMAT (3I5,F10.0,3I5,2F10.0)
 2000 FORMAT (1H1,10X,35HS U B S P A C E   I T E R A T I O N,///1X,12A6//
     1 55H NUMBER OF EQUATIONS . . . . . .(NEQ) =,I5/1X,
     2 54HNUMBER OF ELEMENTS IN STIFFNESS MATRIX . . . . (NWK) =,I5/1X,
     3 54HNUMBER OF ELEMENTS IN MASS MATRIX . . . . . .(NWM) =,I5//)
 2050 FORMAT (//32H FREQUENCY SOLUTION CONTROL DATA,//5X,
     1 55HNUMBER OF EIGENPAIRS TO BE CALCULATED. . . . .(NFREQ) =,I5 /5X,
     2 55H   EQ.0, DEFAULT SET TO 1                              //5X,
     A 55HNUMBER OF VECTORS USED FOR SUBSPACE ITERATION . .(NQ) =,I5 /5X,
     B 55H   EQ.0, SET TO MIN. (2*NFREQ, NFREQ + 8)              //5X,
     C 55HFLAG INDICATING PRESENCE OF RIGID BODY MODES. .(IRBM) =,I5 /5X,
     D 55H   EQ.0, NO ZERO FREQUENCIES ARE PRESENT               //5X,
     E 55HRIGID BODY MODE SHIFT APPLIED. . . . . .(RBMSH) =,E10.3,
     F//5X,
     1 55HINTERMEDIATE PRINT-OUT CONTROL PARAMETER. . . .(IFPR) =,I5 /5X,
     2 55H   EQ.0, NO PRINTING                                   /5X,
     3 55H   EQ.1, PRINT DETAILS OF SOLUTION BEHAVIOUR          /5X,
     4 55H   EQ.2, PRINT EXTENSIVE DEBUG INFORMATION              )
 2100 FORMAT (/5X,
     1 55HMAX NUMBER OF SUBSPACE ITERATIONS ALLOWE' . . (NITEM) =,I5 /5X,
     2 55H   EQ.0, SET TO 24                                     //5X,
     3 55HSTURM SEQUENCE CHECK CONTROL PARAMETER . . . . (IFSS) =,I5 /5X,
     4 55H   EQ.0, CHECK NOT PERFORMED                           /5X,
     5 55H   EQ.1, CHECK PERFORMED                              //5X,
     6 55HFLAG FOR APPLYING ACCELERATION PROCEDURES . . (IACCN) =,I5/ 5X,
     7 55H   EQ.0, NO ACCELERATION                               /5X,
     8 55H   EQ.1, SELF-ADAPTIVE SHIFTING AND OVERRELAXATION     /5X,
```

```
     955H         PROCEDURES WILL BE APPLIED, IF NECESSARY          //5X,
     A55HCONVERGENCE TOLERANCE ON EIGENVALUES . . . . . (RTOL) =,E15.5,
     B/5X,30H   EQ.0., SET TO 1.E-6                                 //5X,
75   155HFLAG TO GENERATE STARTING ITERATION VECTORS. (ISVTYP) =,I5 /5X,
     255H   EQ.0, VECTORS GENERATED USING STANDARD PROCEDURE         /5X,
     355H   GT.0, VECTORS GENERATED USING LANCZOS METHOD            //5X,
     C55HNUMBER OF USER PROVIDED STARTING VECTORS . . . (NSTV) =,I5 /5X,
     D55H   GT.0, NSTV STARTING VECTORS ARE READ FROM TAPE8          )
80    2200 FORMAT (/5X,
     155HFLAG FOR INTERMEDIATE EIGENPAIRS CALCULATION (IINTER) =,I5 /5X,
     255H   EQ.0, NEREQ LOWEST EIGENVALUES ARE CALCULATED           /5X,
     355H   EQ.1, CALCULATE ALL EIGENPAIRS BETWEEN SHIFTS 1,2      //5X,
     455HLOWER LIMIT ON EIGENVALUES TO BE CALCULATED  (SHIFT1) =,E15.5//
85   55X55HUPPER LIMIT ON EIGENVALUES TO BE CALCULATED  (SHIFT2) =E15.5)
     2300 FORMAT (/,19H STORAGE REQUIRED =,I9)
     C
     3000 FORMAT (///,12H *** WARNING,/,
     1             27H SPECIFIED VALUE FOR RTOL =,E15.5,/,
90   2             42H RECOMMENDED VALUE FOR RTOL = .LE. 1.E-06  //)
     3100 FORMAT (///28H *** ERROR, STORAGE EXCEEDED,/10X,
     1            20H BLANK COMMON SIZE =,I10,/10X,
     2            26H ACTUAL STORAGE REQUIRED =,I10)
     C
95         END
```

```
      SUBROUTINE SSPACE (MAXA,A,B,XM,AR,BR,VEC,EIGV,D,
     1                   TT,WW,NLOC,RTOLV,EVC1,EVC2,R,FREQ,BUP,BLO,
     2                   BUPC,NSIT,NN,NROOT,NCM,ISTOP)
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C .                                                               .
C .   P R O G R A M                                               .
C .       TO SOLVE FOR THE EIGENVALUES IN A SPECIFIED INTERVAL AND .
C .       THE ASSOCIATED EIGENVECTORS IN THE GENERALISED EIGENPROBLEM.
C .       USING THE ACCELERATED SUBSPACE ITERATION METHOD          .
C .                                                               .
C .   - - INPUT VARIABLES - -                                     .
C .         MAXA(NNM) = VECTOR CONTAINING ADDRESSES OF DIAGONAL   .
C .                     ELEMENTS OF STIFFNESS MATRIX A            .
C .         NCOLBV    = NUMBER OF COLUMNS PER BLOCK VECTOR        .
C .         ICOPL     = LEAST NUMBERED COUPLING BLOCK             .
C .         A(NWK)    = STIFFNESS MATRIX IN COMPACTED FORM        .
C .         B(NWK)    = CONSISTENT MASS MATRIX IN COMPACTED FORM  .
C .         XM(NN)    = LUMPED MASS MATRIX                        .
C .         AR(NNC)   = WORKING MATRIX STORING PROJECTION OF K    .
C .         BR(NNC)   = WORKING MATRIX STORING PROJECTION OF M    .
C .         VEC(NC*NC)= WORKING ARRAY                             .
C .         EIGV(NC)  = WORKING VECTOR                            .
C .         D(NC)     = WORKING VECTOR                            .
C .         TT(NN)    = WORKING VECTOR                            .
C .         WW(NN)    = WORKING VECTOR                            .
C .         NLOC(NCM) = VECTOR STORING DEGREES OF FREEDOM EXCITED IN
C .                     STARTING SUBSPACE                         .
C .         RTOLV(NC) = WORKING VECTOR                            .
C .         EVC1(NC)  = WORKING VECTOR                            .
C .         EVC2(NC)  = WORKING VECTOR                            .
C .         R(NN,NC)  = EIGENVECTORS ON SOLUTION EXIT             .
C .         FREQ(NCM) = FINAL EIGENVALUES                         .
C .         BUP(NC)   = WORKING VECTOR                            .
C .         BLO(NC)   = WORKING VECTOR                            .
C .         BUPC(NC)  = WORKING VECTOR                            .
C .         NSIT(NC)  = WORKING VECTOR                            .
C .         NN        = ORDER OF STIFFNESS AND MASS MATRICES      .
C .         NWK       = NUMBER OF ELEMENTS BELOW SKYLINE OF       .
C .                     STIFFNESS MATRIX                          .
C .         NWM       = NUMBER OF ELEMENTS BELOW SKYLINE OF       .
C .                     MASS MATRIX                               .
C .                       I.E. NWM=NWK FOR CONSISTENT MASS MATRIX .
C .                            NWM=NN  FOR LUMPED MASS MATRIX     .
C .         NROOT     = NUMBER OF REQUIRED EIGENVALUES AND EIGENVECTORS.
C .         NQ        = NUMBER OF ITERATION VECTORS USED          .
C .         NCM       = MIN(NFREQ + NQ/2,NFREQ + 8)     (BUT NCM  .
C .                     CANNOT BE LARGER THAN THE NUMBER OF MASS  .
C .                     DEGREES OF FREEDOM)                       .
C .         IFSS      = FLAG FOR STURM SEQUENCE CHECK             .
C .                       EQ.0  NO CHECK                          .
C .                       EQ.1  CHECK                             .
C .         IFPR      = FLAG FOR PRINTING DURING ITERATION        .
C .                       EQ.0  NO PRINTING                       .
C .                       EQ.1  PRINT                             .
C .                                                               .
C .   - - OUTPUT - -                                              .
C .         FREQ(NROOT) = EIGENVALUES                             .
```

```
C .        R(NN,NROOT) = EIGENVECTORS                                    .
C .                                                                      .
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C *IBM*       IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON /SOL/ NEQ,NWK,NWM,MA,IMASS,ISTORL
      COMMON /TAPES/ NREAD,NSTIF,IIN,IOUT,NOVER,NSHIFT,NT
     1              IACCN,NJUNK,ISVTYP
      COMMON /SCRAP/ SHIFT,RLQ1,NLQ,NITE,NSTEP,JR,ICONV,NCEV,NP,NQ,IFSS
      COMMON /TOLS/ RTOL,ALPHA,CTOL,RCTOL
      COMMON /EIGIF/ RBMSH,NFREQ,IRBM,IFPR
      COMMON /ITELMT/ NSMAX,NITEM,NITEMM,NOVM
C
      DIMENSION A(ISTOH),B(ISTOH),XM(1),AR(1),BR(1),VEC(1),EIGV(1),D(1),
     1          TT(1),WW(1),RTOLV(1),EVC1(1),EVC2(1),R(NN,1),FREQ(1),
     2          BUP(1),BLO(1),BUPC(1),TIM(10)
      INTEGER MAXA(1),NLOC(1),NSIT(1)
C
C     SET TOLERANCE FOR JACOBI ITERATION
C
      TOLJ=0.000000000001
C
C     INITIALIZATION
C
      ICONV=0
      NCEV=0
      JR=0
      JROLD=0
      RCTOL=0.0100000001
      IF (RTOL.LT.RCTOL) RCTOL=RTOL
      SHIFT=SHIFT1
      NC=NQ
      NNC=NC*(NC + 1)/2
      N1=NC + 1
      NC1=NC - 1
      DO 10 I=1,NC
      NSIT(I)=0
      EVC2(I)=0.
   10 D(I)=0.
      DO 15 I=1,10
   15 TIM(I)=0.
      REWIND NT
C
C     ESTABLISH STARTING ITERATION VECTORS
C
      CALL SECOND (TIM1)
      NO=NN-NCM
      J=NN
      IF (IMASS.EQ.2) GO TO 25
      DO 20 I=1,NN
      II=MAXA(I)
      R(I,1)=XM(I)
      WW(I)=XM(I)/A(II)
      IF (XM(I).EQ.0.) J=J - 1
   20 CONTINUE
```

```
      GO TO 35
C
   25 DO 30 I=1,NN
      II=MAXA(I)
      R(I,1)=B(II)
      WW(I)=B(II)/A(II)
   30 CONTINUE
C
   35 IF (NCM.LE.J .AND. NROOT.LE.J) GO TO 50
      WRITE (IOUT,2007) NCM,J
      STOP
C
C     CHECK FOR SINGLE DEGREE-OF-FREEDOM SYSTEM
C
   50 IF (NN.GT.1) GO TO 65
      IF (IMASS.EQ.1) B(1)=XM(1)
      IF (B(1).GT.0.) GO TO 62
      WRITE (IOUT,2008)
      STOP
   62 EIGV(1)=A(1)/B(1)
      JR=1
      NSCH=1
      A(1)=1./SQRT(B(1))
      WRITE(NT) A(1)
      NEI=NSCH
      GO TO 1150
C
   65 IF (NCM.EQ.1) GO TO 95
      IF (NC.EQ.1) GO TO 69
      DO 68 J=2,NC
      DO 68 I=1,NN
   68 R(I,J)=0.
   69 L=NN - NC
      DO 90 J=2,NCM
      RT=0.
      DO 70 I=1,L
      IF (WW(I).LT.RT) GO TO 70
      RT=WW(I)
      IJ=I
   70 CONTINUE
      DO 80 I=L,NN
      IF (WW(I).LE.RT) GO TO 80
      RT=WW(I)
      IJ=I
   80 CONTINUE
      NLOC(J)=IJ
      WW(IJ)=0.
      L=L-NC
      IF (J.LT.NC) R(IJ,J)=1.
   90 CONTINUE
C
      IF (IPRR.EQ.0) GO TO 93
      WRITE (IOUT,2009)
      WRITE (IOUT,2001) (NLOC(J),J=2,NCM)
   93 IF (NC.EQ.1) GO TO 95
C
C     A RANDOM VECTOR IS TAKEN AS THE LAST ITERATION VECTOR
```

```
      C
      C     RANDOM NUMBER XX(N+1)=FRACTIONAL PART OF (PI + XX(N))**5
      C
      PI=3.141592654
      XX=0.5
      DO 92 K=1,NN
      XX=(PI + XX)**5
      IX=INT(XX)
      XX=XX - FLOAT(IX)
   92 R(K,NC)=XX
      C
      C     READ NSTV STARTING VECTORS FROM TAPE18, IF PROVIDED
      C
   95 IF (NSTV.LE.0) GO TO 120
      NV=NSTV
      IF (NV.GT.NC) NV=NC
      REWIND NSHIFT
      DO 110 J=1,NV
      READ (NSHIFT) (TT(I),I=1,NN)
      CALL MULT (R(1,J),B,XM,TT,MAXA,NN,ISTOH,IMASS)
  110 CONTINUE
      C
  120 CALL SECOND (TIM2)
      TIM(1)=TIM2 - TIM1
      C
      C     FACTORIZE MATRIX A INTO (L)*(D)*(L(T))
      C
      CALL DECOMP (A,B,XM,MAXA,NN,ISTOH,SHIFT,NSCH)
      C
      CALL SECOND (TIM3)
      TIM(2)=TIM3 - TIM2
      NSCH1=NSCH
      IF (IFPR.NE.0 .AND. IINTER.GT.0)
     * WRITE (IOUT,2000) NSCH1
      C
      C     GENERATE STARTING VECTORS USING LANCZOS ALGORITHM
      C
      IF (ISVTYP.EQ.0) GO TO 130
      IF (NSTV.GE.NC1 .OR. NC.LE.2) GO TO 130
      MM=NSTV + 1
      DO 125 K=1,NN
  125 R(K,MM)=1.
      CALL STARTV (A,B,XM,TT,WW,R,MAXA,NLOC,EVC1,MM,NC1,NN,ISTOH,1)
      CALL SECOND (TIM4)
      TIM(1)=TIM(1) + TIM4 - TIM3
      TIM3=TIM4
      C
  130 DO 150 J=1,NC
  150 NLOC(J)=0
      C
      C - - - S T A R T   O F   I T E R A T I O N   L O O P
      C
      NSTEP=4
      NITE=0
      NLQ=0
      RLQ1=0.
  200 NITE=NITE + 1
```

```
      IF (IFPR.EQ.0) GO TO 202
      WRITE (IOUT,2010) MITE
C
C     P R O J E C T I O N   O F   A   M A T R I X
C
  202 CALL SECOND (TIM4)
      IF (IPRO.EQ.0) JR=0
      JJ=JR + 1
      DO 255 J=JJ,NC
      DO 228 K=1,NN
  228 TT(K)=R(K,J)
      CALL REORAK (A,R(1,J),MAXA,NN,ISTOH)
C
      IJ=J
      DO 250 I=1,J
      ART=0.
      IF (I - JR) 238,238,242
  238 DO 240 K=1,NN
  240 ART=ART + R(K,J)*R(K,I)
      ART=ART*(EIGV(I) - SHIFT)
      GO TO 248
  242 DO 246 K=1,NN
  246 ART=ART + R(K,I)*TT(K)
  248 AR(IJ)=ART
  250 IJ=IJ + NC - I
  255 CONTINUE
C
C     P R O J E C T I O N   O F   B   M A T R I X
C
      JJ=JR + 1
      DO 310 J=JJ,NC
      CALL MULT (TT,B,XM,R(1,J),MAXA,NN,ISTOH,IMASS)
C
      IJ=J
      DO 290 I=1,J
      BRT=0.
      IF (ICONV.GT.0) GO TO 279
  277 IF (I - J ) 283,279,279
  279 DO 280 K=1,NN
  280 BRT=BRT + R(K,I)*TT(K)
      GO TO 287
  283 DO 285 K=1,NN
  285 BRT=BRT + R(K,J)*R(K,I)
  287 BR(IJ)=BRT
  290 IJ=IJ + NC - I
      IF (ICONV.GT.0) GO TO 310
      DO 300 K=1,NN
  300 R(K,J)=TT(K)
  310 CONTINUE
      CALL SECOND (TIM5)
      TIM(3)=TIM(3) + TIM5 - TIM4
C
C     SOLVE FOR EIGENSYSTEM OF SUBSPACE OPERATORS
C
      IF (IFPR.NE.2) GO TO 430
      KKK=1
  400 WRITE (IOUT,2020)
```

```
                      II=1
                      DO 410 I=1,NC
                      ITEMP=II+NC-I
                      WRITE(IOUT,2005) (AR(J),J=II,ITEMP)
  90            410   II=II + N1 - I
                      WRITE (IOUT,2030)
                      II=1
                      DO 420 I=1,NC
                      ITEMP=II+NC-I
  95                  WRITE(IOUT,2005) (BR(J),J=II,ITEMP)
                420   II=II + N1 - I
                      IF (KKK - 1) 430,430,440
                C
                430   CALL JACOBI (AR,BR,VEC,EIGV,TT,NC,NNC,TOLJ,SHIFT,NSMAX,IFPR)
 100                  CALL SECOND (TIM6)
                      TIM(4)=TIM(4) + TIM6-TIM5
                C
                      IF (IFPR.NE.2) GO TO 440
                      WRITE (IOUT,2040)
 105                  KKK=2
                      GO TO 400
                C
                C     ARRANGE EIGENVALUES IN ASCENDING ORDER
                C
 110            440   DO 445 I=1,NC
                445   EIGV(I)=EIGV(I) + SHIFT
                      IF (NC.EQ.1) GO TO 465
                C
                448   IS=0
 115                  DO 460 I=1,NC1
                      IF (EIGV(I + 1).GE.EIGV(I)) GO TO 460
                      IS=IS+1
                      EIGVT=EIGV(I + 1)
                      EIGV(I + 1)=EIGV(I)
 120                  EIGV(I)=EIGVT
                      NCI=NC*I
                      NCI1=NC*(I - 1)
                      DO 450 K=1,NC
                      RT=VEC(NCI + K)
 125                  VEC(NCI + K)=VEC(NCI1 + K)
                      VEC(NCI1 + K)=RT
                450   CONTINUE
                460   CONTINUE
                      IF (IS.GT.0) GO TO 448
 130            465   IF (IFPR.EQ.0) GO TO 470
                      WRITE (IOUT,2035)
                      WRITE (IOUT,2006) (EIGV(I),I=1,NC)
                C
                C     CHECK TO SEE WHETHER ANY NEW ROOTS HAVE SUDDENLY APPEARED, IN THAT
 135            C     CASE DECREASE JR. OTHERWISE CALCULATE HOW MANY MORE HAVE CONVERGED
                C     ALSO MAKE SURE THAT CLUSTERED ROOTS ARE FROZEN TOGETHER
                C
                470   JRN=0
                      IF (NITE.EQ.1 .OR. IACCN.EQ.0 .OR. ICONV.GT.0) GO TO 490
 140                  IF (NC.EQ.1 .OR. EIGV(1).LT.SHIFT1) GO TO 490
                      DO 480 I=1,NC1
                      DUM=ABS(EIGV(I) - D(I))/EIGV(I)
```

```
      IF (DUM.GT.RCTOL) GO TO 490
      IF (1.01*EIGV(I).LT.0.99*EIGV(I + 1)) JRN=I
  490 CONTINUE
  495 IF (JRN.LT.JR) JR=JRN
      IF (JRN.LE.JR) GO TO 500
      JJ=JR + 1
      DO 495 I=JJ,JRN
  495 NLOC(NCEV + I)=NITE - NSIT(I)
C
C     CALCULATE R TIMES APPROXIMATE EIGENVECTORS (ICONV.EQ.0)
C        OR     FINAL EIGENVECTOR APPROXIMATIONS (ICONV.GT.0)
C
  500 JJ=JR + 1
      DO 540 I=1,NN
      DO 510 J=1,NC
  510 TT(J)=R(I,J)
      DO 530 K=JJ,NC
      KK=NC*(K-1)
      RT=0.
      DO 520 L=1,NC
  520 RT=RT + TT(L)*VEC(KK+L)
  530 R(I,K)=RT
  540 CONTINUE
      IF (IPRR.NE.2) GO TO 542
      WRITE (IOUT,2045)
      DO 541 I=1,NC
      K1=I
      K2=NC*(NC - 1) + I
      WRITE (IOUT,2005) (VEC(J),J=K1,K2,NC)
  541 CONTINUE
C
C     UPDATE  JR, THE NUMBER OF CONVERGED ROOTS
C
  542 IF (ICONV.GT.0) GO TO 558
      JR=JRN
C
  558 CALL SECOND (TIM7)
      TIM(5)=TIM(5) + TIM7-TIM6
      IF (ICONV.GT.0) GO TO 1000
C
C     CHECK FOR CONVERGENCE OF EIGENVALUES,EIGENVECTORS
C
      DO 560 I=1,NC
      DIF=ABS(EIGV(I)-D(I))
  560 RTOLV(I)=DIF/EIGV(I)
      IF (IPRR.EQ.0) GO TO 570
      WRITE (IOUT,2050)
      WRITE (IOUT,2250) (RTOLV(I),I=1,NC)
C
C     ACCELERATE CONVERGENCE, IF NECESSARY AND IS POSSIBLE
C
  570 CALL SECOND (TIM8)
      NJUNK=0
      DO 590 J=1,NC
      IF (EIGV(J).GT.SHIFT1) GO TO 600
      NJUNK=J
  590 CONTINUE
```

```
      600 NJ=NJUNK + 1
          NP=NC - NJUNK
C
          CALL RAPID (EIGV(NJ),D(NJ),TT,EVC1(NJ),EVC2(NJ),RTOLV(NJ),
     1              P(1,NJ),R,FREQ,WW,XM,NLOC,NSIT(NJ),NN,NR)
C
          CALL SECOND (TIM9)
          TIM(6)=TIM(6) + TIM9-TIM8
C
      910 DO 920 I=1,NC
          EVC1(I)=EVC2(I)
          EVC2(I)=D(I)
      920 D(I)=EIGV(I)
          IF (JR.EQ.JJ) GO TO 960
          IF (ICONV.GT.0) GO TO 960
C
C     PRESET PROJECTION MATRICES CORRESPONDING TO CONVERGED EIGENVECTORS
C
          II=1
          DO 940 I=1,JR
          IJ=II
          AR(IJ)=EIGV(I) - SHIFT
          BR(IJ)=1.
          IJ=IJ + 1
          IF (I.EQ.JR) GO TO 940
          JJ=I + 1
          DO 930 J=JJ,JR
          AR(IJ)=0.
          BR(IJ)=0.
      930 IJ=IJ + 1
      940 II=II + N1 - I
C
      960 GO TO 200
C
C - - - E N D   O F   I T E R A T I O N   L O O P
C
     1000 CALL SECOND (TIM10)
          JR=NROOT
          J=0
          DO 1010 I=1,NC
          IF (EIGV(I).LT.SHIFT1) GO TO 1010
          J=J + 1
          FREQ(NCEV + J)=EIGV(I)
     1010 CONTINUE
C
          IF (NROOT.EQ.0) GO TO 1160
          REWIND NT
          IF (NCEV.EQ.0) GO TO 1030
          DO 1020 L=1,NCEV
     1020 READ (NT)
     1030 NR=NROOT - NCEV
          IF (NP.EQ.0) GO TO 1080
C
          J=0
          DO 1070 L=1,NC
          IF (EIGV(L).LT.SHIFT1) GO TO 1070
          J=J + 1
```

```
            IF (J.GT.NR) GO TO 1080
            CALL MULT (WW,B,XM,R(1,L),MAXA,NN,ISTOH,IMASS)
            WRITE (NT) (R(I,L),I=1,NN),(WW(I),I=1,NN)
 1070 CONTINUE
C
C     CALCULATE AND PRINT ERROR NORMS
C
 1080 WRITE (NT) (FREQ(I),I=1,NROOT)
      REWIND NT
      REWIND NSTIF
      READ (NSTIF) A
      DO 1130 L=1,NROOT
      RT=FREQ(L)
      READ (NT) (WW(I),I=1,NN),(R(I,1),I=1,NN)
      CALL MULT (TT,A,XM,WW,MAXA,NN,ISTOH,2)
      VNORM=0.
      DO 1100 I=1,NN
 1100 VNORM=VNORM + TT(I)*TT(I)
      WNORM=0.
      DO 1120 I=1,NN
      TT(I)=TT(I) - RT*R(I,1)
 1120 WNORM=WNORM + TT(I)*TT(I)
      VNORM=SQRT(VNORM)
      WNORM=SQRT(WNORM)
      FREQ(L)=WNORM/VNORM
 1130 CONTINUE
      WRITE(IOUT,2115)
      WRITE (IOUT,2006) (FREQ(I),I=1,NROOT)
C
      READ (NT) (FREQ(I),I=1,NROOT)
      WRITE (IOUT,2100)
      PI=4.*ATAN(1.)
      DO 1141 I=1,NROOT
      AA=FREQ(I) + RBMSH
      IF (AA.LE.0.) AA=1.E-10
      AA=SQRT(AA)
      ACIRC=AA/(2.*PI)
      APERD=1./ACIRC
 1141 WRITE (IOUT,2105) I,AA,ACIRC,APERD
      WRITE (IOUT,2110) (NLOC(I),I=1,NROOT)
      IF (IFSS.EQ.0) GO TO 1160
C
C     APPLY STURM SEQUENCE CHECK
C
      NEI=NROOT
      NJUNK=0
      DO 1142 L=1,NC
      IF (EIGV(L).GT.SHIFT1) GO TO 1145
      NJUNK=L
 1142 CONTINUE
 1145 NJ=NJUNK + 1
      NCM=NCEV + NC - NJUNK
      CALL SCHECK (FREQ,RTOLV(NJ),BUP,BLO,BUPC,NLOC,NCM,NEI,RTOL,SHIFT)
C
      WRITE (IOUT,2120) SHIFT
C
C     SHIFT MATRIX A AND FACTORIZE SHIFTED MATRIX
```

```
C
      CALL DECOMP (A,B,XM,MAXA,NN,ISTOH,SHIFT,NSCH)
C
      NSCH=NSCH - NSCH1
 1150 WRITE (IOUT,2130) NSCH,SHIFT1,SHIFT,NEI
      IF (NSCH.GT.NEI) WRITE (IOUT,2140)
 1160 CALL SECOND (TIM11)
      TIM(7)=TIM(7) + TIM11-TIM10
      TIM(10)=TIM11 - TIM1
      IF (IPRR.GT.0)
     *  WRITE (IOUT,2200) NITE,(TIM(I),I=1,7),TIM(10)
C
      RETURN
C
 2001 FORMAT (1H0,10I10)
 2005 FORMAT (1H ,12E11.4)
 2006 FORMAT (1H0,6E22.14)
 2007 FORMAT (///45H *** STOP, REQUESTED NUMBER OF ROOTS, NFREQ =,I5,
     1 63H IS LARGER THAN THE NUMBER OF NON-ZERO MASS DEGREES OF FREEDOM
     2=,I5)
 2008 FORMAT (38H0***ERROR   SOLUTION STOP IN #SSPACE#    , / 12X,
     1        23HNO EIGENVALUES COMPUTED, / 1X)
 2009 FORMAT (   /,62H DEGREES OF FREEDOM EXCITED BY UNIT STARTING ITERA
     1TION VECTORS)
 2010 FORMAT (1H1,32HI T E R A T I O N   N U M B E R ,I4///)
 2020 FORMAT (28H)PROJECTION OF A (MATRIX AR) )
 2035 FORMAT (30H0EIGENVALUES OF AR-LAMBDA*BR  )
 2030 FORMAT (28H0PROJECTION OF B (MATRIX BR) )
 2040 FORMAT (40HJAR AND BR AFTER JACOBI DIAGONALIZATION  )
 2045 FORMAT (29H00 MATRIX                                /)
 2050 FORMAT (43H0RELATIVE TOLERANCE REACHED ON EIGENVALUES  )
 2060 FORMAT (//,37H NUMBER OF EIGENVALUES BELOW SHIFT1 =,I5)
 2100 FORMAT (// 24H   F R E Q U E N C I E S   //  2X,
     121H    FREQUENCY NUMBER   ,10X,20H FREQUENCY (RAD/SEC) ,4X,
     224H  FREQUENCY (CYCLES/SEC),8X,16HPERIOD (SECONDS)  / )
 2105 FORMAT (9X,I5,24X,E11.4,15X,E11.4,15X,E11.4)
 2110 FORMAT (///59H NUMBER OF SUBSPACE ITERATIONS PERFORMED FOR EACH EI
     1GENPAIR/,(1H0,20I5))
 2115 FORMAT ( 1H1,36HPRINT ERROR NORMS ON THE EIGENVALUES  )
 2120 FORMAT  (///,23H CHECK APPLIED AT SHIFT   E22.14)
 2130 FORMAT (/40H BASED ON STURM SEQUENCE CHECK THERE ARE,I4,
     1         29H EIGENVALUES BETWEEN SHIFT1 =,E15.5,12H AND SHIFT =,
     2         E15.5,//46H NUMBER OF EIGENVALUES CALCULATED BY THE PROGR
     3         21HAM IN THIS INTERVAL =,I4 //)
 2140 FORMAT (95H TO CALCULATE THE MISSING EIGENVALUES REPEAT THE SOLUTI
     1ON USING LARGER NUMBER OF TRIAL VECTORS.,/19H ALSO DECREASE RTOL )
 2200 FORMAT (1H1,///23H EIGENSOLUTION TIME LOG ,///1X,
     A51HNUMBER OF SUBSPACE ITERATIONS PERFORMED            =,I5,/1X,
     151HTIME FOR CALCULATION OF STARTING SUBSPACE          =,F9.3,/1X,
     251HTIME FOR LDLT FACTORIZATION OF STIFFNESS MATRIX    =,F9.3,/1X,
     351HTIME FOR CALCULATION OF PROJECTIONS OF A AND B     =,F9.3,/1X,
     451HTIME FOR SOLVING EIGENSYSTEM OF SUBSPACE OPERATORS=,F9.3,/1X
     551HTIME FOR SORTING EIGENVALUES, NORMALISING VECTORS =,F9.3,/1X,
     651HTIME FOR CALCULATING AND APPLYING SHIFTS           =,F9.3,/1X
     751HTIME FOR ERROR NORMS AND STURM SEQUENCE CHECK      =,F9.3,//1X,
     851HTIME FOR EIGENSOLUTION                             =,F9.3,///)
 2250 FORMAT (6E15.5/)
```

```
C
      END
```

```
      SUBROUTINE STARTV (A,B,XM,TT,WW,R,MAXA,NLOC,D,M1,M2,NN,ISTOH,KKK)
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C .                                                                   .
C .   PROGRAM TO GENERATE STARTING VECTORS FOR SUBSPACE ITERATION     .
C .   USING LANCZOS ALGORITHM                                         .
C .                                                                   .
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C *IBM*       IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON /SOL/ NEQ,NWK,NWM,MA,IMASS,ISTORL
      COMMON /TAPES/ NREAD,NSTIF,IIN,IOUT,NOVER,NSHIFT,NT
      COMMON /FAST/ SHIFT1,SHIFT2,IOVER,IRPC,NSTV,IINTER,JROLD,NSCH1,
     1              IACCN,NJUNK,ISVTYP
      COMMON /SCRAP/ SHIFT,RLQ1,NLQ,NITE,NSTEP,JR,ICONV,NCEV,NP,NQ,IFSS
      COMMON /EIGIF/ RBMSH,NFREQ,IRBM,IFPR
C
      DIMENSION A(ISTOH),B(ISTOH),XM(1),TT(1),WW(1),R(NN,1),D(1)
      INTEGER MAXA(1),NLOC(1)
C
      REWIND NSHIFT
      NJ=1
      RETOL=0.0001
      RQTOL=0.00000001
      GSTOL=0.001
      IF (IFPR.NE.0) WRITE (IOUT,2000)
C
C     ORTHOGONALISE STARTING VECTOR TO CONVERGED VECTORS ON TAPE NT
C
   10 IF (NCEV.EQ.0) GO TO 60
      REWIND NT
      DO 40 I=1,NCEV
      READ (NT) (TT(K),K=1,NN),(WW(K),K=1,NN)
      AL=0.
      DO 20 K=1,NN
   20 AL=AL + R(K,M1)*WW(K)
      DO 30 K=1,NN
   30 R(K,M1)=R(K,M1) - AL*TT(K)
   40 CONTINUE
C
   60 CALL MULT (TT,B,XM,R(1,M1),MAXA,NN,ISTOH,IMASS)
C
C     ORTHOGONALISE STARTING VECTOR TO THE VECTORS IN-CORE CURRENTLY
C
      IF (M1.EQ.1) GO TO 110
      IF (KKK.EQ.1) REWIND NSHIFT
      MM=M1 - 1
      DO 90 I=1,MM
      AL=0.
      DO 80 K=1,NN
   80 AL=AL + R(K,M1)*R(K,I)
      DO 85 K=1,NN
   85 TT(K)=TT(K) - AL*R(K,I)
      IF (KKK.GT.1) GO TO 90
      READ (NSHIFT) (WW(K),K=1,NN)
      DO 88 K=1,NN
```

```
      88 R(K,M1)=R(K,M1) - AL*WW(K)
      90 CONTINUE
   50    IF (KKK.EQ.1) GO TO 110
C
         DO 92 K=1,NN
      92 R(K,M1)=TT(K)
         CALL REDBAK (A,TT,MAXA,NN,ISTOH)
   65    ALFA=0.
         DO 100 K=1,NN
     100 ALFA=ALFA + TT(K)*R(K,M1)
C
         CALL MULT (R(1,M1),B,XM,TT,MAXA,NN,ISTOH,IMASS)
   70    GO TO 130
C
     110 DO 120 K=1,NN
         DUM=TT(K)
         TT(K)=R(K,M1)
   75    R(K,M1)=DUM
         ALFA=0.
C
     130 BETA=0.
         DO 135 K=1,NN
   80    BETA=BETA + TT(K)*R(K,M1)
         D(NJ)=ALFA/BETA
         BETA=SQRT(BETA)
         GAMA=BETA
         DO 140 K=1,NN
   85    TT(K)=TT(K)/BETA
     140 R(K,M1)=R(K,M1)/BETA
         IF (IFPR.NE.0) WRITE (IOUT,2020) M1,ALFA,BETA,GAMA,D(NJ)
C
         WRITE (NSHIFT) (TT(K),K=1,NN)
   90    IF (M1.EQ.M2) GO TO 500
         BETA=0.
         DO 210 K=1,NN
     210 WW(K)=0.
         MM=M1 + 1
   95 C
         DO 400 J=MM,M2
         J1=J - 1
C
C        INVERSE ITERATION
  100 C
         DO 220 K=1,NN
     220 R(K,J)=R(K,J1)
         CALL REDBAK (A,R(1,J),MAXA,NN,ISTOH)
C
  105 C    COMPUTE AND CHECK THE RAYLEIGH QUOTIENT
C
         ALFA=0.
         DO 230 K=1,NN
     230 ALFA=ALFA + R(K,J)*R(K,J1)
  110    RQT=ALFA
         DO 240 K=1,NN
     240 R(K,J)=R(K,J) - ALFA*TT(K) - BETA*WW(K)
         DO 250 K=1,NN
     250 WW(K)=TT(K)
```

```
      RQ0=ALFA*ALFA + BETA*BETA
C
C     NORMALISE THE NEW VECTOR
C
      CALL MULT (TT,B,XM,R(1,J),MAXA,NN,ISTOH,IMASS)
      BETA=0.
      DO 280 K=1,NN
  280 BETA=BETA + R(K,J)*TT(K)
      RQ0=RQ0 + BETA
      RQ=RQT/RQ0
      BETA=SQRT(BETA)
      GAMA=SQRT(RQ0)
      DO 285 K=1,NN
  285 R(K,J)=R(K,J)/BETA
      IF (IFPR.NE.0) WRITE (IOUT,2020) J,ALFA,BETA,GAMA,RQ
C
C     CHECK FOR THE LINEAR INDEPENDENCE OF THE LANCZOS VECTORS
C
      DO 290 I=1,NJ
      IF (ABS(D(NJ) - RQ).LE.ABS(RQ)*RQTOL) GO TO 295
  290 CONTINUE
      IF (BETA/GAMA.GT.BETOL) GO TO 300
  295 DO 298 K=1,NN
  298 R(K,J)=0.
      IJ=NLOC(NCEV + J)
      R(IJ,J)=1.
      M1=J
      NJ=NJ + 1
      GO TO 10
C
C     MASS ORTHOGONALISE THE NEW VECTOR TO PREVIOUSLY OBTAINED VECTORS
C
  300 REWIND NSHIFT
      IFLAG=0
      DO 330 I=1,NJ
      READ (NSHIFT) (TT(K),K=1,NN)
      IF (IFLAG.GT.0) GO TO 330
      AL=0.
      DO 310 K=1,NN
  310 AL=AL + R(K,J)*R(K,I)
      IF (ABS(AL).GT.GSTOL) IFLAG=1
      DO 320 K=1,NN
  320 R(K,J)=R(K,J) - AL*TT(K)
  330 CONTINUE
      IF (IFLAG.GT.0) GO TO 295
C
      DO 340 K=1,NN
  340 TT(K)=R(K,J)
      CALL MULT (R(1,J),B,XM,TT,MAXA,NN,ISTOH,IMASS)
C
      AL=0.
      DO 380 K=1,NN
  380 AL=AL + TT(K)*R(K,J)
      AL=SQRT(AL)
      DO 390 K=1,NN
      TT(K)=TT(K)/AL
  390 R(K,J)=R(K,J)/AL
```

```
C
      NJ=NJ + 1
      D(NJ)=RO
      WRITE (NSHIFT) (TT(K),K=1,NN)
  400 CONTINUE
C
C     ORTHOGONALISE THE RANDOM STARTING VECTOR TO THE OTHERS
C
  500 IF (KKK.GT.1 .OR. IINTER.EQ.0) GO TO 600
      REWIND NSHIFT
      DO 530 J=1,M2
      AL=0.
      DO 510 K=1,NN
  510 AL=AL + P(K,NQ)*R(K,J)
      READ (NSHIFT) (TT(K),K=1,NN )
      DO 520 K=1,NN
  520 P(K,NQ)=R(K,NQ) - AL*TT(K)
  530 CONTINUE
C
  600 RETURN
C
 2000 FORMAT (///,49H STARTING VECTORS ARE GENERATED BY LANCZOS METHOD /
     1     /,7H VECTOR,59H     ALFA            BETA            GAMA
     1  RAYL. QUO.     /)
 2020 FORMAT (I7,4E15.5)
 2050 FORMAT (21H GRAM-SCHMIDT FACTORS,(/8E15.5))
C
      END
```

```
      SUBROUTINE RAPID (EIGV,D,TT,EVC1,EVC2,RTOLV,R,RR,FREQ,WW,XM,
     1                  NLOC,NSIT,NN,NQ)
C
C     PROGRAM TO STUDY THE CONVERGENCE OF SUBSPACE ITERATIONS AND
C     INVESTIGATE THE POSSIBILITIES OF ACCELERATING IT
C
C *IBM*       IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /SOL/ NEQ,NWK,NWM,MA,IMASS,ISTOH
      COMMON /EIGIF/ RBMSH,NFREQ,IRBM,IFPR
      COMMON /SCRAP/ SHIFT,RLQ1,NLD,NITE,NSTEP,JR,ICONV,NCEV,NP,NQA,IFSS
      COMMON /ITELMT/ NSMAX,NITEM,NITEMM,NOVM
      COMMON /FAST/ SHIFT1,SHIFT2,IOVER,IRPC,NSTV,IINTER,JROLD,NSCH1,
     1              IACCN,NJUNK,ISVTYP
      COMMON /DIMSSP/ M1,M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12,M13,M14,M15
      COMMON /TAPES/ NREAD,NSTIF,IIN,IOUT,NOVER,NSHIFT,NT
      COMMON /TOLS/ RTOL,ALPHA,CTOL,RCTOL
C
      REAL A
      COMMON A(1)
      INTEGER IA(1)
      EQUIVALENCE (A(1),IA(1))
C
      DIMENSION EIGV(1),D(1),TT(1),EVC1(1),EVC2(1),RTOLV(1),R(NN,1)
     1         ,RR(NN,1),FREQ(1),WW(1),XM(1),NLOC(1),NSIT(1)
      DATA NFAC/0/, NGS/0/
C
      RMUS=SHIFT
      SHIFTO=SHIFT
      RITOL=RTOL
      IF (IINTER.GT.0) RITOL=RCTOL
      NP=NFREQ - NCEV
      IF (NQ.LT.NP) RITOL=RCTOL
      IVSHF=0
C
      NEIG=0
      DO 580 I=1,NQ
      IF (RTOLV(I).GT.RITOL) GO TO 590
      NEIG=I
      IF (NLOC(NCEV + I).GT.0) GO TO 580
      NLOC(I + NCEV)=NITE - NSIT(I)
  580 CONTINUE
C
  590 IF (NEIG.EQ.0) GO TO 600
      IF (EIGV(NEIG).GE.SHIFT2) GO TO 595
      IF (IINTER.GT.0) GO TO 600
      IF (NCEV + NEIG.LT.NFREQ) GO TO 600
  595 NFREQ=NCEV + NEIG
      ICONV=1
      JR=0
      IF (IFPR.NE.0) WRITE (IOUT,2060) RTOL,NITE
      GO TO 910
C
  600 IF (NITE.LT.NITEMM) GO TO 700
      WRITE (IOUT,2070)
      ICONV=2
      IFSS=0
      NFREQ=NCEV + NEIG
```

```
      JP=0
      GO TO 910
C
C     CALCULATE RATE OF CONVERGENCE
C
  700 NC=NEREQ - NCEV
      IF (NC.GT.NQ .OR. IINTER.GT.0) NC=NQ
      IF (IACON.EQ.0) GO TO 1300
      IF (RITOL.EQ.RCTOL) GO TO 720
      NEIG=0
      DO 710 I=1,NQ
      IF (RTOLV(I).GT.RCTOL) GO TO 720
      NEIG=I
  710 CONTINUE
  720 IF (NEIG.LT.NQ) GO TO 730
      IVSHF=1
      GO TO 910
C
  730 IF (NITE.LE.NSTEP - 2) GO TO 910
      DO 740 I=1,NC
      TEMP=D(I) - EVC2(I)
      EVC2(I)=0.
      IF (ABS(TEMP).GT.D(I)*1.E-10) EVC2(I)=(EIGV(I) - D(I))/TEMP
  740 CONTINUE
      IF (IFPR.GT.1)
     1 WRITE (IOUT,2300) (EVC2(I),I=1,NC)
      IF (NITE.EQ.NSTEP - 1) GO TO 910
      IF (IFPR.GT.1)
     1 WRITE (IOUT,2350) (EVC1(I),I=1,NC)
C
C     ESTABLISH LAMBDA (Q + 1) FROM FAIRLY CONVERGED ROOTS
C
      DO 750 I=1,NC
      TT(I)=EIGV(NQ)*2.**30
      IF (RTOLV(I).GT.1.E-3 .OR. RTOLV(I).LT.RCTOL) GO TO 750
      IF (EVC2(I).GT.0.99 .OR. EVC2(I).LT.1.E-5) GO TO 750
      TEMP=EVC2(I) - EVC1(I)
      TEMP=ABS(TEMP/EVC2(I))
      IF (TEMP.GT.CTOL) GO TO 750
      TT(I)=(EIGV(I) - RMUS)/SQRT(EVC2(I)) + RMUS
      IF (TT(I).LT.RMUS) GO TO 750
      IF (RTOLV(I).GT.RCTOL) NLOC(NCEV + I)=-1
  750 CONTINUE
      IF (IFPR.GT.1)
     1 WRITE (IOUT,2400) (TT(I),I=1,NC)
C
      NLQ=0
      TLQ=0.
      DO 760 I=1,NC
      IF (TT(I).GE.EIGV(NQ)*2.**30) GO TO 760
      IF (TT(I).LT.RMUS) GO TO 760
      NLQ=NLQ + 1
      TLQ=TLQ + TT(I)
  760 CONTINUE
C
      IF (NLQ.NE.0)
     1 RLQ1=(NLQ*RLQ1 + TLQ)/(NLQ + NLQ)
```

```
5                NLQ=MLQ + NLQ
                 IF (IEPR.NE.0)
                1 WRITE (IOUT,2500) RLQ1,NLQ
                 IF (RLQ1.GT.0.) GO TO 762
                 IF (NITE.EQ.NITEMM - 1 .AND. NEIG.GT.0) IVSHF=1
0                GO TO 910
           C
           C     ESTABLISH SHIFT  FROM THE SET OF VECTORS CONVERGED TO RTOL SO FAR
           C
             762 J=1
5                DO 763 I=1,NC
                 IF (EIGV(I).GT.RMUS) GO TO 765
                 J=J + 1
             763 CONTINUE
           C
0            765 IF (J.GT.NC) J=NC
                 NR=J
                 DO 770 I=J,NC
                 IF (RTOLV(I).GT.RCTOL) GO TO 773
                 NR=NR + 1
5            770 CONTINUE
           C
             773 IF (NR.GT.NC) NR=NC
                 K=NR
                 BAND=EIGV(1) + (RLQ1 - EIGV(1))/3
0          C
             775 NR=NR - 1
                 IF (NR.LE.J) GO TO 778
                 SHIFT=0.5*(EIGV(NR) + EIGV(NR - 1))
                 IF (SHIFT.LT.1.01*EIGV(NR - 1)) GO TO 775
5                IF (SHIFT.GT.0.99*EIGV(NR)) GO TO 775
                 IF (SHIFT.GT.BAND) GO TO 775
                 J=NR
                 GO TO 790
           C
0          C     SPECIAL CASE - (1) SHIFT TO THE LEFT OF THE FIRST EIGENVALUE
           C
             778 IF (J.GT.1) GO TO 788
                 IF (NCEV.GT.0) GO TO 780
                 SHIFT=0.5*EIGV(1)
5                IF (SHIFT.LE.RMUS) GO TO 910
                 J=1
                 GO TO 790
           C
           C     SPECIAL CASE - (2) SHIFT BETWEEN EIGENVALUES IN BACK-UP STORE
0          C
             780 NR=NCEV + 2
                 FREQ(NCEV + 1)=EIGV(1)
                 IF (RTOLV(1).GT.RCTOL) NR=NR - 1
             785 NR=NR - 1
5                IF (NR.LT.1) GO TO 910
                 SHIFT=0.5*FREQ(NR)
                 IF (NR.GT.1) SHIFT=SHIFT + 0.5*FREQ(NR - 1)
                 IF (SHIFT.LE.RMUS) GO TO 910
                 IF (SHIFT.LT.1.01*FREQ(NR - 1)) GO TO 785
1                IF (SHIFT.GT.0.99*FREQ(NR)) GO TO 785
                 J=NR - NCEV
```

```
      GO TO 790
C
C     IF THE MAXIMUM POSSIBLE SHIFT WITHIN THE CURRENT BASIS HAS BEEN
C     REACHED, CONSIDER EXPANSION OF THE BASIS FOR ACCELERATION
C
  788 TEMP=0.5*(EIGV(K) + EIGV(K - 1))
      IF (TEMP.GT.BAND) IVSHF=1
      GO TO 910
C
C     CHECK HOW MANY MORE ITERATIONS WOULD BE REQUIRED WITH THIS SHIFT
C
  790 SI=0.
      II=NC
      DO 800 I=K,NC
      IF (EIGV(I).GE.RLQ1) GO TO 800
      IF (RTOLV(I).LE.RITOL .OR. RTOLV(I).GT.0.01) GO TO 800
      TOLI=RITOL/RTOLV(I)
      DI=((EIGV(I) - RMUS)/(RLQ1 - RMUS))**2
      ST=ALOG10(TOLI)/ALOG10(DI)
      DP=((EIGV(I) - SHIFT)/(RLQ1 - SHIFT))**2
      STP=ALOG10(TOLI)/ALOG10(DP)
      SII=ST - STP
      IF (SII.LE.SI) GO TO 800
      SI=SII
      II=I
  800 CONTINUE
      I=II
      TOLI=RITOL/RTOLV(I)
      DI=((EIGV(I) - RMUS)/(RLQ1 - RMUS))**2
      ST=ALOG10(TOLI)/ALOG10(DI)
      DP=((EIGV(I) - SHIFT)/(RLQ1 - SHIFT))**2
      STP=ALOG10(TOLI)/ALOG10(DP)
C
C     CHECK TO SEE WHETHER IT IS WORTH SHIFTING
C
      IF (SI.GT.3.) GO TO 820
      NOFAC=0
      NOSI=0
      IF (K.LE.1) GO TO 850
      TEMP=0.5*(EIGV(K) + EIGV(K - 1))
      IF (TEMP.LT.BAND) GO TO 850
      IVSHF=1
      GO TO 910
C
  820 MA=NWK/NN + 1
      NOFAC=(MA*MA + 3*MA)/2
      IF (IMASS.EQ.2) NOFAC=NOFAC + MA
      NP=NQ - JP
      NOSI=(2*MA*NP + 2*NP*NP + 3*NP + 2*NP*JR + 2*NQ*NCEV)*SI
      IF (IMASS.EQ.2) NOSI=NOSI + 2*MA*NP*SI
      NOSI=NOSI + 18*NP*NP*NP*SI/NN
      IF (NOFAC.GE.ALPHA*NOSI) GO TO 850
C
C      SHIFT IS GOOD. PRINT ALL INFORMATION. SHIFT K MATRIX.
C
      IF (IPR.NE.0) WRITE (IOUT,2600)
      NSTEP=4 + NITE
```

```
                RMUS=SHIFT
0               NFAC=NFAC + 1
          C
                CALL DECOMP (A(M2),A(M3),A(M4),A(M1),NN,ISTOH,SHIFT,NSCH)
          C
          C     PERFORM STURM SEQUENCE CHECK AT THE CURRENT SHIFT.
5         C
                IF (IFPR.NE.0) WRITE (IOUT,2650) NSCH
                IF (NSCH.EQ.NSCH1 + NCEV + J - 1) GO TO 855
                WRITE (IOUT,2680)
                IFSS=0
0               GO TO 595
          C
            850 IF (IFPR.NE.0) WRITE (IOUT,2690)
            855 IF (IFPR.EQ.0) GO TO 910
                J=J + NCEV + NSCH1
5               I=I + NCEV + NSCH1
                WRITE (IOUT,2700) J,I,SHIFT
                WRITE (IOUT,2800) TOLI,DI,DP,ST,STP,NOFAC,NOSI
          C
            910 SHIFT=RMUS
0               IF (IACCN.EQ.0) GO TO 1300
          C
          C     REORTHOGONALISE W.R.T. CONVERGED VECTORS IN THE CRITICAL BAND
          C
                IF (NCEV.EQ.0) GO TO 985
5               REWIND NT
                BAND=ABS(EIGV(NQ) - SHIFT)
                JJ=JR + 1
                NORTHO=0
                DO 980 J=1,NCEV
0               IF (NJUNK.GT.0) GO TO 930
                RT=ABS(FREQ(J) - SHIFT)
                IF (RT.LE.BAND) GO TO 930
                READ (NT)
                GO TO 980
5         C
            930 READ (NT) (TT(I),I=1,NN),(WW(I),I=1,NN)
                NORTHO=NORTHO + 1
                DO 970 K=JJ,NQA
                AL=0.
0               DO 940 I=1,NN
            940 AL=AL + TT(I)*RR(I,K)
                DO 950 I=1,NN
            950 RR(I,K)=RR(I,K) - AL*WW(I)
            970 CONTINUE
5           980 CONTINUE
                IF (IFPR.NE.0) WRITE (IOUT,2850) NORTHO
                NGS=NGS + NORTHO
          C
          C     OVERRELAX ITERATION VECTOR IF LINEAR CONVERGENCE HAS BEEN OBTAINED
0         C
            985 IF (IOVER.NE.1) GO TO 1000
                IF (NITE.GT.NSTEP-4 .AND. NITE.LE.NSTEP-1) GO TO 1000
                REWIND NOVER
          C
5               JJ=JROLD + 1
```

```
      DO 995 J=JJ,NC
      IF (NLOC(NCEV + J).EQ.-1) GO TO 988
      READ (NOVER)
      GO TO 995
C
  988 NLOC(NCEV + J)=0
      READ (NOVER) (TT(K),K=1,NN)
      RC=(EIGV(I) - SHIFTO)/(RLQ1 - SHIFTO)
      RA=1./(1. - RC)
      DO 990 K=1,NN
  990 R(K,J)=TT(K) + (R(K,J) - TT(K))*RA
      K=J + NJUNK
      IF (IFPR.NE.0) WRITE (IOUT,2820) K
  995 CONTINUE
C
C     IF ALLOWED, THROW SOME VECTORS FROM THE CURRENT BASIS OUT
C
 1000 IF (IVSHF.EQ.0) GO TO 1200
      K=NEIG
      IF (JR.GT.0) K=JR
      IF (IINTER.GT.0) GO TO 1015
C
      NP=NFREQ - NCEV
      NQ=MIN0(2*NP,NP + 8)
      IF (NQ.LT.NC) GO TO 1010
      GO TO 1200
C
 1010 NC=NQ/2
      IF (NQ.GT.16) NC=NQ - 8
      IF (NP.LT.NC + K) K=NP - NC
      IF (NCEV+K+NQ.GT.NN) K=NN - (NCEV + NQ)
 1015 IF (K.EQ.0) GO TO 1200
      NLQ=0
      RLQ1=0.
      NSTEP=NITE + 4
      NITEMM=NITE + NITEM
      IF (NCEV.EQ.0) REWIND NT
      DO 1050 J=1,K
      FREQ(NCEV + J)=EIGV(J)
      DO 1030 I=1,NN
 1030 TT(I)=(EIGV(J) - SHIFT)*R(I,J)
C
      CALL REDBAK (A(M2),TT,A(M1),NN,ISTOH)
C
      CALL MULT (R(1,J),A(M3),A(M4),TT,A(M1),NN,ISTOH,IMASS)
      AL=0.
      DO 1042 I=1,NN
 1042 AL=AL + TT(I)*R(I,J)
      AL=SQRT(AL)
      DO 1045 I=1,NN
      TT(I)=TT(I)/AL
 1045 R(I,J)=R(I,J)/AL
      WRITE (NT) (TT(I),I=1,NN),(R(I,J),I=1,NN)
 1050 CONTINUE
C
C     SHIFT EIGENPAIRS IN THE CURRENT BASIS
C
```

```
      IF (IFPR.NE.0) WRITE (IOUT,2900) NCEV,NP,K
      NCEV=NCEV + K
      IF (IINTER.EQ.0) GO TO 1060
      IF (NCEV.LE.50) GO TO 1052
      WRITE (IOUT,3000)
      IVSHF=0
      NCEV=NCEV - K
      GO TO 595
C
C     SPECIAL CASE - INTERMEDIATE EIGENVALUES SOLUTION ONLY
C
 1052 IF (NJUNK.EQ.0) GO TO 1060
      NR=NCEV + 1
 1055 NR=NR - 1
      IF (NR.LT.2) GO TO 1060
      RMUS=0.5*(FREQ(NR) + FREQ(NR - 1))
      IF (RMUS.LE.SHIFT) GO TO 1060
      IF (RMUS.LT.1.01*FREQ(NR - 1)) GO TO 1055
      IF (RMUS.GT.0.99*FREQ(NR)) GO TO 1055
      NFAC=NFAC + 1
      SHIFT=RMUS
      IF (IFPR.NE.0) WRITE (IOUT,2920) SHIFT
C
      CALL DECOMP (A(M2),A(M3),A(M4),A(M1),NN,ISTOH,SHIFT,NSCH)
C
      IF (IFPR.NE.0) WRITE (IOUT,2650) NSCH
      IF (NSCH.EQ.NSCH1 + NR - 1) GO TO 1060
      WRITE (IOUT,2680)
      IVSHF=0
      NCEV=NCEV - K
      IFSS=0
      GO TO 595
C
 1060 NP=NQ - K
      IF (NP.EQ.0) GO TO 1090
      JR=JR - K
      IF (JR.LT.0) JR=0
      DO 1080 N=1,NP
      J=N + K
      EIGV(N)=ETGV(J)
      NSIT(N)=NSIT(J)
      DO 1070 I=1,NN
 1070 R(I,N)=R(I,J)
 1080 CONTINUE
C
C     ESTABLISH NEW STARTING VECTORS AND ORTHOGONALISE THEM
C
 1090 NP=NP + 1
      DO 1092 J=NP,NQ
      EIGV(J)=0.
      NSIT(J)=NITE
 1092 CONTINUE
C
      DO 1097 J=NP,NQ
      IF (NCEV + J.GT.NSTV) GO TO 1098
      READ (NSHIFT) (TT(I),I=1,NN)
      NLOC(NCEV + J)=0
```

```
                CALL MULT (R(1,J),A(M3),A(M4),TT,A(M1),NN,ISTOH,IMASS)
 1097 CONTINUE
      GO TO 1115
C
 1098 K=J
      DO 1105 J=K,NQ
      DO 1100 I=1,NN
 1100 R(I,J)=0.
      II=NLOC(NCEV + NJUNK + J)
      R(II,J)=1.
      IF (ISVTYP.GT.0) GO TO 1107
 1105 CONTINUE
      GO TO 1109
C
 1107 MM=NJUNK + J
      CALL STARTV (A(M2),A(M3),A(M4),TT,WW,RR,A(M1),NLOC,0,MM,NQA,NN,
     1             ISTOH,2)
C
 1109 DO 1110 J=K,NQ
 1110 NLOC(NCEV + J)=0
C
 1115 REWIND NT
      DO 1150 J=1,NCEV
      READ (NT) (TT(I),I=1,NN),(WW(I),I=1,NN)
      DO 1140 K=NP,NQ
      AL=0.
      DO 1120 I=1,NN
 1120 AL=AL + TT(I)*R(I,K)
      DO 1130 I=1,NN
 1130 R(I,K)=R(I,K) - AL*WW(I)
 1140 CONTINUE
 1150 CONTINUE
C
C     WRITE TRIAL VECTORS ONTO TAPE NOVER FOR FUTURE OVERRELAXATION
C
 1200 IF (NITE.LT.NSTEP-1) GO TO 1300
      REWIND NOVER
      JROLD=JR
      JJ=JR + 1
      NC=NFREQ - NCEV
      IF (NC.GT.NQ .OR. IINTER.GT.0) NC=NQ
      DO 1205 J=JJ,NC
 1205 WRITE (NOVER) (R(K,J),K=1,NN)
C
 1300 RETURN
C
C
 2060 FORMAT (///,30H CONVERGENCE REACHED FOR RTOL  E10.4,2X,
     1        14H  AT ITERATION,I4//)
 2070 FORMAT (1H1,51H*** NO CONVERGENCE IN MAXIMUM NUMBER OF ITERATIONS
     1             9HPERMITTED/35H WE ACCEPT CURRENT ITERATION VALUES/
     2            42H THE STURM SEQUENCE CHECK IS NOT PERFORMED  )
 2300 FORMAT (///44H RATE OF CONVERGENCE ESTIMATES,RI(I + 1) ARE,//(6E15.
     1             5/))
 2350 FORMAT (///44H RATE OF CONVERGENCE ESTIMATES,RI(I    ) ARE,//(6E15.
     1             5/))
 2400 FORMAT (///31H ESTIMATES OF LAMBDA(Q + 1) ARE,/,(6E15.5/))
```

```
 2500 FORMAT (///36H AVERAGE ESTIMATE OF LAMDA(Q+1) IS =,E20.10,     /
     1          9H BASED ON,I5,10H ESTIMATES   )
 2600 FORMAT (//51H NOFAC IS LESS THAN ALPHA*NOSI AND SHIFT IS APPLIED/)
 2650 FORMAT (//46H STURM SEQUENCE CHECK PERFORMED AT THIS SHIFT. /
     1          60H ESTIMATED NUMBER OF EIGENVALUES TO THE LEFT OF THIS
     2SHIFT =,I5/)
 2680 FORMAT (14H CHECK FAILED. /
     1          81H ESTIMATED NUMBER OF EIGENVALUES IS MORE THAN THE NUMBE
     2R OF COMPUTED EIGENVALUES. /55H REPEAT SOLUTION WITH A LARGER NUMB
     3ER OF TRIAL VECTORS.   /35H ALSO USE A SMALLER VALUE FOR RTOL //)
 2690 FORMAT (//52H SHIFT CALCULATED DOES NOT SATISFY SHIFTING CRITERIA)
 2700 FORMAT (//,31H SHIFT ESTIMATED TO THE LEFT OF,I3,15HTH EIGENVALUE
     1,12H BASED ON ,I3,18HTH EIGENVALUE IS =,E15.5/)
 2800 FORMAT (///11X,4HTOLI,14X,1HD,13X,2HDP,14X,1HT,13X,2HTP,4X,5HNOFAC
     1        ,5X,4HNOSI//,5E15.5,2I9)
 2820 FORMAT (42H OVER-RELAXATION IS PERFORMED FOR VECTOR =,I4)
 2850 FORMAT (//33H GRAM-SCHMIDT ORTHOGONALISATION OF CURRENT TRIAL VECT
     1 36HORS IS PERFORMED W.R.T. THE PREVIOUS,I4,13H EIGENVECTORS  )
 2900 FORMAT (//35H EIGENVECTORS ARE TAKEN OUT          , /
     1          45H NUMBER OF EIGENVECTORS ALREADY REMOVED     =,I5,/,
     2          44H NUMBER OF EIGENPAIRS YET TO BE CALCULATED =,I5/,
     3          49H NUMBER OF EIGENVECTORS CURRENTLY BEING REMOVED =,I5)
 2920 FORMAT (//,81H IN ORDER TO MOVE THE POLE OF ATTRACTION TO THE RIGH
     1T, A SHIFT  IS ALSO PERFORMED /,16H SHIFT APPLIED =,E15.5)
C
 3000 FORMAT (1H1,///,14H *** ERROR ***,/,
     1        55H STORAGE OVERFLOW OCCURED.  IN A GIVEN INTERVAL ONLY A
     2        47H MAXIMUM OF FIFTY EIGENVALUES CAN BE CALCULATED  )
C
      END
```

```
      SUBROUTINE DECOMP (A,B,XM,MAXA,NN,ISTOH,RA,NSCH)
C . . . . . .
C .                                                                    .
C .   P R O G R A M                                                    .
C .       TO PERFORM (L)*(D)*(L)(T) FACTORIZATION OF SHIFTED           .
C .       STIFFNESS MATRIX,  A - RA*B                                  .
C .                                                                    .
C . . . . . .
C *IBM* IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON /SOL/ NEQ,NWK,NWM,MA,IMASS,ISTORL
      COMMON /TAPES/ NREAD,NSTIF,IIN,IOUT,NOVER,NSHIFT,NT
      DIMENSION A(ISTOH),B(ISTOH),XM(1),MAXA(1)
C
      TOL=10.**10
      RTOL=0.00001
      NTF=3
      IS=1
      NSCH=0
C
   50 REWIND NSTIF
      READ (NSTIF) A
      IF (RA.EQ.0.) GO TO 100
      IF (IMASS.EQ.1) GO TO 60
      DO 55 I=1,ISTOH
   55 A(I)=A(I) - RA*B(I)
      GO TO 100
   60 DO 80 I=1,NN
      II=MAXA(I)
   80 A(II)=A(II) - RA*XM(I)
C
  100 DO 260 N=1,NN
      KN=MAXA(N)
      KL=KN + 1
      KU=MAXA(N+1) - 1
      KH=KU - KL
      IF (KH) 304,240,210
  210 K=N - KH
      IC=0
      KLT=KU
      DO 260 J=1,KH
      IC=IC + 1
      KLT=KLT - 1
      KI=MAXA(K)
      ND=MAXA(K+1) - KI - 1
      IF (ND) 260,260,270
  270 KK=MIN0(IC,ND)
      C=0.
      DO 280 L=1,KK
  280 C=C + A(KI+L)*A(KLT+L)
      A(KLT)=A(KLT) - C
  260 K=K + 1
C
  240 K=N
      E=0.
      DO 300 KK=KL,KU
      K=K - 1
```

```
      KI=MAXA(K)
      C=A(KK)/A(KI)
      IF (ABS(C).LT.TOL) GO TO 290
      WRITE (IOUT,2010) N,C
      GO TO 315
  290 E=E + C*A(KK)
  300 A(KK)=C
      A(KN)=A(KN) - E
C
  304 IF (A(KN)) 200,310,200
  310 IF (RA.EQ.0.) GO TO 320
  315 IS=IS + 1
      IF (IS.LE.NTF) GO TO 330
  320 WRITE (IOUT,2000) NTF,RA
      STOP
  330 RA=RA*(1. - RTOL)
      RTOL=RTOL*10.
      GO TO 50
C
  200 CONTINUE
C
C     COUNT THE NUMBER OF NEGATIVE ELEMENTS IN D FACTOR
C
      DO 350 I=1,NN
      II=MAXA(I)
      IF (A(II).LT.0.) NSGN=NSGN + 1
  350 CONTINUE
C
      RETURN
 2000 FORMAT (37H0***ERROR  SOLUTION STOP IN *DECOMP*, / 12X,
     1        1H(,I3,37H) TRIANGULAR FACTORIZATIONS ATTEMPTED, / 12X,
     2        16HCURRENT SHIFT = ,E20.14 /1X,
     3        87HEITHER STIFFNESS MATRIX IS NOT POSITIVE DEFINITE OR SHI
     4FT IS TOO CLOSE TO AN EIGENVALUE   )
 2010 FORMAT (//52H *WARNING, DURING FACTORIZATION OF STIFFNESS MATRIX
     135HMULTIPLIER GROWTH FOR COLUMN NUMBER,I4,//12H MULTIPLIER=,E20.8)
C
      END
```

SUBROUTINE REDBAK

```
      SUBROUTINE REDBAK (A,V,MAXA,NN,ISTOH)
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C .                                                                 .
C .   P R O G R A M                                               .
C .        TO REDUCE AND BACK-SUBSTITUTE ITERATION VECTORS          .
C .                                                                 .
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C *IBM* IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(ISTOH),V(1),MAXA(1)
C
      DO 400 N=1,NN
      KL=MAXA(N) + 1
      KU=MAXA(N+1) - 1
      IF (KU-KL) 400,410,410
  410 K=N
      C=0.
      DO 420 KK=KL,KU
      K=K - 1
  420 C=C + A(KK)*V(K)
      V(N)=V(N) - C
  400 CONTINUE
C
      DO 480 N=1,NN
      K=MAXA(N)
  480 V(N)=V(N)/A(K)
C
      IF (NN.EQ.1) RETURN
      N=NN
      DO 500 L=2,NN
      KL=MAXA(N) + 1
      KU=MAXA(N+1) - 1
      IF (KU-KL) 500,510,510
  510 K=N
      DO 520 KK=KL,KU
      K=K - 1
  520 V(K)=V(K) - A(KK)*V(N)
  500 N=N - 1
C
      RETURN
      END
```

```
      SUBROUTINE MULT (TT,B,XM,RR,MAXA,NN,ISTOH,KKK)
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C .                                                                  .
C .   P R O G R A M                                                  .
C .        TO EVALUATE PRODUCT OF B TIMES RR AND STORE RESULT IN TT  .
C .        WHERE B IS STORED IN COMPACTED FORM, RR AND TT ARE VECTORS .
C .                                                                  .
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C *IBM* IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION TT(1),B(ISTOH),XM(1),RR(1),MAXA(1)
C
      IF (KKK.EQ.2) GO TO 20
      DO 10 I=1,NN
   10 TT(I)=XM(I)*RR(I)
      RETURN
C
   20 DO 40 I=1,NN
   40 TT(I)=0.
      DO 100 I=1,NN
      KL=MAXA(I)
      KU=MAXA(I+1) - 1
      II=I + 1
      CC=RR(I)
      DO 100 KK=KL,KU
      II=II - 1
  100 TT(II)=TT(II) + B(KK)*CC
C
      IF (NN.EQ.1) RETURN
      DO 200 I=2,NN
      KL=MAXA(I) + 1
      KU=MAXA(I+1) - 1
      IF (KU-KL) 200,210,210
  210 II=I
      AA=0.
      DO 220 KK=KL,KU
      II=II - 1
  220 AA=AA + B(KK)*RR(II)
      TT(I)=TT(I) + AA
  200 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE RBSHFT (MAXA,S,B,XM,IRBM,RBMSH,NN,ISTOH,IMASS)
C
C *IBM*       IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON /TAPES/ NREAD,NSTIF,IIN,IOUT,NOVER,NSHIFT,NT
      DIMENSION S(ISTOH),B(ISTOH),XM(1)
      INTEGER MAXA(1)
C
      DO 10 I=1,NN
      II=MAXA(I)
      AA=S(II)
      IF (AA.GT.0.) GO TO 10
      WRITE (6,1000) I,AA
      STOP
   10 CONTINUE
C
      IF (IRBM.EQ.0) GO TO 200
      IF (RBMSH.NE.0.) GO TO 19
C
      SHIFT=0.
      IF (IMASS.EQ.2) GO TO 15
      DO 12 I=1,NN
      II=MAXA(I)
      DUM=XM(I)/S(II)
      IF (DUM.GT.SHIFT) SHIFT=DUM
   12 CONTINUE
      GO TO 19
C
   15 DO 17 I=1,NN
      II=MAXA(I)
      DUM=B(II)/S(II)
      IF (DUM.GT.SHIFT) SHIFT=DUM
   17 CONTINUE
C
C     APPLY SHIFT IF RIGID BODY MODES ARE PRESENT
C
   19 IF (RBMSH) 30,25,22
   22 WRITE (6,3000)
      STOP
   25 RBMSH=-0.001/SHIFT
C
   30 REWIND NSTIF
      IF (IMASS.EQ.2) GO TO 104
      DO 106 I=1,NN
      II=MAXA(I)
  106 S(II)=S(II) - RBMSH*XM(I)
      GO TO 102
C
  104 DO 108 I=1,ISTOH
  108 S(I)=S(I) - RBMSH*B(I)
C
  102 WRITE (NSTIF) S
C
  200 RETURN
C
 1000 FORMAT (43H ***ERROR   NEG OR ZERO DIAGONAL ELEMENT A(,I4,4H) =  ,
     1        E11.4,21HBEFORE DECOMPOSITION   )
```

```
      SUBROUTINE RBSHFT

     3000 FORMAT (//46H *** ERROR IN FREQUENCY CONTROL CARD INPUT ***,/,
     1       54H THE VALUE FOR RIGID BODY SHIFT INPUT MUST BE NEGATIVE )
0           C
            END
```

```
      SUBROUTINE SCHECK (EIGV,RTOLV,BUP,BLO,BUPC,NEIV,NC,NEI,RTOL,SHIFT)
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C .                                                                    .
C .   P R O G R A M                                                    .
C .       TO EVALUATE SHIFT FOR STURM SEQUENCE CHECK                   .
C .                                                                    .
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C *IBM*       IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /TAPES/ NREAD,NSTIF,LIN,IOUT,NOVER,NSHIFT,NT
      COMMON /SCRAP/ RMUSS,RLQ1,NLQ,NITE,NSTEP,JR,ICONV,NCEV,NP,NQ,IFSS
C
      DIMENSION EIGV(1),RTOLV(1),BUP(1),BLO(1),BUPC(1),NEIV(1)
C
      FTOL=0.001
C
      DO 100 I=1,NC
      BUP(I)=EIGV(I)*(1. + FTOL)
  100 BLO(I)=EIGV(I)*(1. - FTOL)
C
      NROOT=NCEV
      II=NC - NCEV - 1
      DO 120 I=1,II
      IF (RTOLV(I).GT.RTOL) GO TO 130
      IF (BUP(I).LE.BLO(I + 1)) NROOT=NCEV + I
  120 CONTINUE
      IF (RTOLV(NC - NCEV).LE.RTOL) NROOT=NC
  130 IF (NROOT.GE.1) GO TO 200
      WRITE (IOUT,1010)
      STOP
C
C      FIND UPPER BOUNDS ON EIGENVALUE CLUSTERS
C
  200 DO 240 I=1,NROOT
  240 NEIV(I)=1
      IF (NROOT.NE.1) GO TO 260
      BUPC(1)=BUP(1)
      LM=1
      L=1
      I=2
      IF (NC.EQ.1) GO TO 300
      GO TO 295
  260 L=1
      I=2
  270 IF (BUP(I-1).LE.BLO(I)) GO TO 280
      NEIV(L)=NEIV(L)+1
      I=I+1
      IF (I.LE.NROOT) GO TO 270
  280 BUPC(L)=BUP(I-1)
      IF (I.GT.NROOT) GO TO 290
      L=L+1
      I=I + 1
      IF (I.LE.NROOT) GO TO 270
      BUPC(L)=BUP(I-1)
  290 LM=L
      IF (NROOT.EQ.NC) GO TO 300
  295 IF (BUP(I-1).LE.BLO(I)) GO TO 300
```

```
      IF (I.GT.NCEV .AND. RTOLV(I - NCEV).GT.RTOL) GO TO 300
      BUPC(L)=BUP(I)
      NEIV(L)=NEIV(L)+1
      NROOT=NROOT+1
      IF (NROOT.EQ.NC) GO TO 300
      I=I+1
      GO TO 295
C
C      FIND SHIFT
C
  300 WRITE (IOUT,1020)
      WRITE (IOUT,1005) (BUPC(I),I=1,LM)
      WRITE (IOUT,1030)
      WRITE (IOUT,1006) (NEIV(I),I=1,LM)
      LL=LM-1
      IF (LM.EQ.1) GO TO 310
  330 DO 320 I=1,LL
  320 NEIV(L)=NEIV(L)+NEIV(I)
      L=L-1
      LL=LL-1
      IF (L.NE.1) GO TO 330
  310 WRITE (IOUT,1040)
      WRITE (IOUT,1006) (NEIV(I),I=1,LM)
      L=0
      DO 340 I=1,LM
      L=L+1
      IF (NEIV(I).GE.NROOT) GO TO 350
  340 CONTINUE
  350 SHIFT=BUPC(L)
      NEI=NEIV(L)
C
      RETURN
C
 1005 FORMAT (1H0,6E22.14)
 1006 FORMAT (1H0,6I22)
 1010 FORMAT (37H0***ERROR   SOLUTION STOP IN *SCHECK*, / 12X,
     1        21HNO EIGENVALUES FOUND., / 1X)
 1020 FORMAT (///,37H UPPER BOUNDS ON EIGENVALUE CLUSTERS  )
 1030 FORMAT (34H0NO OF EIGENVALUES IN EACH CLUSTER  )
 1040 FORMAT (42H0NO OF EIGENVALUES LESS THAN UPPER BOUNDS  )
      END
```

```
      SUBROUTINE JACOBI (A,B,X,EIGV,D,N,NWA,RTOL,SHIFT,NSMAX,IPPR)
C ...................................................................
C .                                                                 .
C .   P R O G R A M                                                 .
C .       TO SOLVE THE GENERALIZED EIGENPROBLEM USING THE           .
C .       GENERALIZED JACOBI ITERATION                              .
C .                                                                 .
C . - - INPUT VARIABLES - -                                         .
C .       A(NWA)    = STIFFNESS MATRIX (ASSUMED POSITIVE DEFINITE)  .
C .             (UPPER TRIANGULAR PART STORED ROWWISE FROM DIAGONAL) .
C .       B(NWA)    = MASS MATRIX (ASSUMED POSITIVE DEFINITE)       .
C .             (UPPER TRIANGULAR PART STORED ROWWISE FROM DIAGONAL) .
C .       X(N,N)    = MATRIX STORING EIGENVECTORS ON SOLUTION EXIT  .
C .       EIGV(N)   = VECTOR STORING EIGENVALUES ON SOLUTION EXIT   .
C .       D(N)      = WORKING VECTOR                                .
C .       N         = ORDER OF MATRICES A AND B                     .
C .       RTOL      = CONVERGENCE TOLERANCE (USUALLY SET TO 10.**-12).
C .       NSMAX     = MAXIMUM NUMBER OF SWEEPS ALLOWED              .
C .                             (USUALLY SET TO 15)                 .
C .       IPPR      = FLAG FOR PRINTING DURING ITERATION            .
C .           EQ.0    NO PRINTING                                   .
C .           EQ.1    INTERMEDIATE RESULTS ARE PRINTED              .
C .       IOUT      = OUTPUT DEVICE NUMBER                          .
C .                                                                 .
C . - - OUTPUT - -                                                  .
C .       A(NWA)    = DIAGONALIZED STIFFNESS MATRIX                 .
C .       B(NWA)    = DIGONALIZED MASS MATRIX                       .
C .       X(N,M)    = EIGENVECTORS STORED COLUMNWISE                .
C .       EIGV(N)   = EIGENVALUES                                   .
C .                                                                 .
C ...................................................................
C *IBM*       IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON /TAPES/ NREAD,NSTIF,IIN,IOUT,NOVER,NSHIFT,NT
      DIMENSION A(NWA),B(NWA),X(N,N),EIGV(N),D(N)
C
C     INITIALIZE EIGENVALUE AND EIGENVECTOR MATRICES
C
      N1=N + 1
      II=1
      DO 10 I=1,N
      IF (B(II).LT.0.) GO TO 3
      IF (A(II).GT.0.) GO TO 4
      IF (SHIFT.GT.0.) GO TO 4
    3 WRITE(IOUT,2020) II,A(II),B(II)
      STOP
    4 D(I)=A(II)/B(II)
      EIGV(I)=D(I)
   10 II=II + N1 - I
      DO 30 I=1,N
      DO 20 J=1,N
   20 X(I,J)=0.
   30 X(I,I)=1.
      IF (N.EQ.1) GO TO 255
C
C     INITIALIZE SWEEP COUNTER AND BEGIN ITERATION
C
```

```
      NSWEEP=0
      NP=N-1
   40 NSWEEP=NSWEEP+1
      IF(IFPR.EQ.2) WRITE(IOUT,2000)NSWEEP
C
C     CHECK IF PRESENT OFF-DIAGONAL ELEMENT IS LARGE ENOUGH TO REQUIRE Z
C
      EPS=(.01**NSWEEP)**2
      DO 210 J=1,NP
      JP1=J+1
      JM1=J-1
      LJK=JM1*N - JM1*J/2
      JJ=LJK + J
      DO 210 K=JP1,N
      KP1=K+1
      KM1=K-1
      JK=LJK + K
      KK=KM1*N - KM1*K/2 + K
      EPTOLA=(A(JK)*A(JK))
      EPTLA1=ABS(A(JJ)*A(KK)*EPS)
      EPTOLB=(B(JK)*B(JK))
      EPTLB1=(B(JJ)*B(KK)*EPS)
      IF((EPTOLA.LT.EPTLA1) .AND. (EPTOLB.LT.EPTLB1)) GO TO 210
C
C     IF ZEROING IS REQUIRED, CALCULATE THE ROTATION MATRIX ELEMENTS CA
C
      AKK=A(KK)*B(JK)-B(KK)*A(JK)
      AJJ=A(JJ)*B(JK)-B(JJ)*A(JK)
      AB=A(JJ)*B(KK)-A(KK)*B(JJ)
      CHECK=(AB*AB+4.*AKK*AJJ)/4.
      IF(CHECK)50,60,60
   50 WRITE (IOUT,2020) JJ,A(JJ),B(JJ),KK,A(KK),B(KK),JK,A(JK),B(JK)
      STOP
   60 SQCH=SQRT(CHECK)
      D1=AB/2.+SQCH
      D2=AB/2.-SQCH
      DEN=D1
      IF(ABS(D2).GT.ABS(D1))DEN=D2
      IF(DEN)80,70,80
   70 CA=0.
      CG=-A(JK)/A(KK)
      GO TO 90
   80 CA=AKK/DEN
      CG=-AJJ/DEN
C
C     PERFORM THE GENERALIZED ROTATION TO ZERO THE PRESENT OFF-DIAGONAL
C
   90 IF(N-2)100,190,100
  100 IF(JM1-1)130,110,110
  110 DO 120 I=1,JM1
      IM1=I - 1
      IJ=IM1*N - IM1*I/2 + J
      IK=IM1*N - IM1*I/2 + K
      AJ=A(IJ)
      BJ=B(IJ)
      AK=A(IK)
      BK=B(IK)
```

```
5                A(IJ)=AJ+CG*AK
                 B(IJ)=BJ+CG*BK
                 A(IK)=AK+CA*AJ
             120 B(IK)=BK+CA*BJ
             130 IF(KP1-N)140,140,160
0            140 LJI=JM1*N - JM1*J/2
                 LKI=KM1*N - KM1*K/2
                 DO 150 I=KP1,N
                 JI=LJI + I
                 KI=LKI + I
5                AJ=A(JI)
                 BJ=B(JI)
                 AK=A(KI)
                 BK=B(KI)
                 A(JI)=AJ+CG*AK
0                B(JI)=BJ+CG*BK
                 A(KI)=AK+CA*AJ
             150 B(KI)=BK+CA*BJ
             160 IF(JP1-KM1)170,170,190
             170 LJI=JM1*N - JM1*J/2
5                DO 180 I=JP1,KM1
                 JI=LJI + I
                 IM1=I - 1
                 IK=IM1*N - IM1*I/2 + K
                 AJ=A(JI)
0                BJ=B(JI)
                 AK=A(IK)
                 BK=B(IK)
                 A(JI)=AJ+CG*AK
                 B(JI)=BJ+CG*BK
5                A(IK)=AK+CA*AJ
             180 B(IK)=BK+CA*BJ
             190 AK=A(KK)
                 BK=B(KK)
                 A(KK)=AK+2.*CA*A(JK)+CA*CA*A(JJ)
0                B(KK)=BK+2.*CA*B(JK)+CA*CA*B(JJ)
                 A(JJ)=A(JJ)+2.*CG*A(JK)+CG*CG*AK
                 B(JJ)=B(JJ)+2.*CG*B(JK)+CG*CG*BK
                 A(JK)=0.
                 B(JK)=0.
5            C
             C     UPDATE THE EIGENVECTOR MATRIX AFTER EACH ROTATION
             C
                 DO 200 I=1,N
                 XJ=X(I,J)
0                XK=X(I,K)
                 X(I,J)=XJ+CG*XK
             200 X(I,K)=XK+CA*XJ
             210 CONTINUE
             C
5            C     UPDATE THE EIGENVALUES AFTER EACH SWEEP
             C
                 II=1
                 DO 220 I=1,N
                 IF (B(II).LT.0.) GO TO 212
0                IF (A(II).GT.0.) GO TO 215
                 IF (SHIFT.GT.0.) GO TO 215
```

```
      212 WRITE(IOUT,2020) II,A(II),B(II)
          STOP
      215 EIGV(I)=A(II)/B(II)
      220 II=II + N1 - I
          IF(IFPR.LT.2)GO TO 230
          WRITE(IOUT,2030)
          WRITE(IOUT,2010) (EIGV(I),I=1,N)
C
C     CHECK FOR CONVERGENCE
C
      230 DO 240 I=1,N
          TOL=RTOL*ABS(D(I))
          DIF=ABS(EIGV(I)-D(I))
          IF(DIF.GT.TOL)GO TO 280
      240 CONTINUE
C
C     CHECK ALL OFF-DIAGONAL ELEMENTS TO SEE IF ANOTHER SWEEP IS REQUIRE
C
          EPS=RTOL**2
          DO 250 J=1,NR
          JM1=J-1
          JP1=J+1
          LJK=JM1*N - JM1*J/2
          JJ=LJK + J
          DO 250 K=JP1,N
          KM1=K-1
          JK=LJK + K
          KK=KM1*N - KM1*K/2 + K
          EPSA=(A(JK)*A(JK))
          EPSB=(B(JK)*B(JK))
          EPSA1=ABS(A(JJ)*A(KK)*EPS)
          EPSB1=(B(JJ)*B(KK)*EPS)
          IF((EPSA.LT.EPSA1) .AND. (EPSB.LT.EPSB1)) GO TO 250
          GO TO 280
      250 CONTINUE
C
C     FILL OUT BOTTOM TRIANGLE OF RESULTANT MATRICES AND SCALE EIGENVECT
C
      255 II=1
          DO 275 I=1,N
          BB=SQRT(B(II))
          DO 270 K=1,N
      270 X(K,I)=X(K,I)/BB
      275 II=II + N1 - I
          IF (IFPR.GT.0) WRITE (IOUT,2040) NSWEEP
          RETURN
C
C     UPDATE  D  MATRIX AND START NEW SWEEP, IF ALLOWED
C
      280 DO 290 I=1,N
      290 D(I)=EIGV(I)
          IF(NSWEEP.LT.NSMAX)GO TO 40
          WRITE (IOUT,2050)
          STOP
C
     2010 FORMAT(1H0,6E20.12)
     2020 FORMAT(27H0SWEEP NUMBER IN *JACOBI* = ,I4)
```

```
     2020 FORMAT (25H0*** ERROR  SOLUTION STOP  /
0       1        31H MATRICES NOT POSITIVE DEFINITE /
        2 (4H  I=,I4,6HA(II)=,E20.12,6HB(II)=,E20.12))
     2030 FORMAT(36H0CURRENT EIGENVALUES IN *JACOBI* ARE,/)
     2040 FORMAT (//,33H NUMBER OF SWEEPS IN JACOBI ARE =,I4//)
     2050 FORMAT (1H1,12H ** STOP ** /,
5       1            39H NO CONVERGENCE IN *JACOBI ITERATIONS*  /,
        2            26H EIGEN SOLUTION ABANDONED )
         END
```

## APPENDIX C - GLOSSARY OF TERMS

Banded Matrix: A symmetric matrix $\underline{K}$ is a banded matrix if the condition -- $k_{ij}=0$ for $j > i + m_{\underline{K}}$, is satisfied; where $k_{ij}$ is the element of the matrix $\underline{K}$ in the $i^{th}$ row, $j^{th}$ column; $(2\, m_{\underline{K}} + 1)$ is the bandwidth of the matrix $\underline{K}$ and $m_{\underline{K}}$ is the half-bandwidth of $\underline{K}$.

Basis: A basis for a vector space is a linearly independent set of vectors which spans the vector space; any vector in this vector space can be expressed as a linear combination of the basis vectors.

Characteristic Equation: Given a square matrix $\underline{K}$ of order n, the characteristic equation of the matrix $\underline{K}$ is $\det|\underline{K}-\lambda\underline{I}|=0$, where $\underline{I}$ represents the identity matrix. The expansion of the determinant yields a $n^{th}$ degree polynomial in $\lambda$, called the characteristic polynomial.

Deflated Polynomial: The deflated polynomial $P_j(\lambda)$ is defined as $P_j(\lambda) = P(\lambda)/\prod_{i=1}^{j}(\lambda-\lambda_i)$, where $P(\lambda)$ is the characteristic polynomial and $\lambda_i$, i=1, ..., j, are the calculated j smallest roots of the polynomial $P(\lambda)$.

Determinant: The determinant of a matrix $\underline{K}$ of order n is denoted as det $\underline{K}$ and is defined as $\det \underline{K} = \prod_{i=1}^{n} d_{ii}$, where $d_{ii}$ are the diagonal elements of the matrix D in the $\underline{L}\,\underline{D}\,\underline{L}^T$ factorization of the matrix $\underline{K}$ and $\pi$ is the product symbol.

Gaussian Elimination: Gaussian elimination is a commonly used algorithm for the solution of large systems of simultaneous equations of the form $\underline{K}\,\underline{U} = \underline{R}$, where $\underline{U}$ and $\underline{R}$ are vectors. In this algorithm, when $\underline{K}$ is symmetric, the matrix $\underline{K}$ is decomposed into the factors $\underline{L}\,\underline{D}\,\underline{L}^T$ where $\underline{L}$ is a lower triangular matrix and $\underline{D}$ is a diagonal matrix.

Eigenproblem: The problem $\underline{K}\,\underline{\phi} = \lambda\,\underline{M}\,\underline{\phi}$ is referred to as a generalized eigenproblem, if the solution of this problem is considered in order to obtain the eigenvalues $\lambda$ and the eigenvectors $\underline{\phi}$; whereas if only the eigenvalues are to be calculated $\underline{K}\,\underline{\phi} = \lambda\,\underline{M}\,\underline{\phi}$ is called a generalized eigenvalue problem. When the matrix $\underline{M}$ is an identity matrix, the problem $\underline{K}\,\underline{\phi} = \lambda\underline{\phi}$ is called a standard eigenproblem.

Eigenpair: An eigenvalue $\lambda_i$ of the generalized (or standard) eigenproblem and the associated eigenvector $\underline{\phi}_i$ that satisfies the eigenproblem, are together referred to as the $i^{th}$ eigenpair. Also, all the eigenvalues and eigenvectors of an eigenproblem constitute the eigensystem of the eigenproblem.

Eigenspace: Corresponding to each multiple eigenvalue, i.e., when several eigenvalues have the same magnitude, an eigenspace with dimension equal to the multiplicity of the eigenvalue exists. When all the eigenvalues of an eigenproblem are discrete, all the corresponding eigenspaces are one dimensional, namely the

corresponding eigenvectors. It should be noted that the eigenvectors corresponding to a multiple eigenvalue are not unique, since any vector in the eigenspace corresponding to the multiple eigenvalue satisfies the eigenproblem.

Eigenvalue Separation Property: In addition to the eigenproblem, $\underline{K}\,\underline{\phi} = \lambda\,\underline{\phi}$, (where the order of the matrices $\underline{K}$ is $n$) consider the problems $\underline{K}^{(m)}\,\underline{\phi}^{(m)} = \lambda^{(m)}\,\underline{\phi}^{(m)}$ where $\underline{K}^{(m)}$ is obtained by omitting the last m rows and columns of $\underline{K}$. Using also the notation $\underline{K}^{(0)} = \underline{K}$, $\lambda^{(0)} = \lambda$, $\underline{\phi}^{(0)} = \underline{\phi}$, the eigenvalue separation property states that the eigenvalues of the problem $\underline{K}^{(m+1)}\,\underline{\phi}^{(m+1)} = \lambda^{(m+1)}\,\underline{\phi}^{(m+1)}$ separate the eigenvalues of the problem $\underline{K}^{(m)}\,\underline{\phi}^{(m)} = \lambda^{(m)}\,\underline{\phi}^{(m)}$; i.e., $\lambda_1^{(m)} \leq \lambda_1^{(m+1)} \leq \lambda_2^{(m)} \leq \lambda_2^{(m+1)} \ldots \leq \lambda_{n-m-1}^{(m)} \leq \lambda_{n-m-1}^{(m+1)} \leq \lambda_{n-m}^{(m)}$ for $m=0,\ldots,n-2$. In other words, the characteristic polynomials of the problems $\underline{K}^{(m)}\,\underline{\phi}^{(m)} = \lambda^{(m)}\,\underline{\phi}^{(m)}$, $m=0,1,\ldots,n-1$, i.e., $P_{n-m}(\lambda^m) = \det|\underline{K}^{(m)} - \lambda^{(m)}\underline{I}|$, form a Sturm sequence.

Gram-Schmidt Orthogonalization: The Gram-Schmidt orthogonalization is employed to derive from a vector $\underline{x}$ a new vector $\underline{x}_1$ such that the vector $\underline{x}_1$ is orthogonal to a specified vector $\underline{y}$ of unit norm. The orthogonalization is effected in the following manner: $\underline{x}_1 = \underline{x} - \alpha\underline{y}$ where $\alpha = \underline{x}^T\underline{y}$. Note that $\underline{x}_1{}^T\underline{y} = (\underline{x}^T\underline{y} - \alpha\underline{y}^T\underline{y}) = 0$ since $\underline{y}^T\underline{y}$, namely the Euclidean norm of the vector $\underline{y}$, is unity.

Linear Dependence: The vectors $\{\underline{x}_1, \underline{x}_2, \ldots \underline{x}_n\}$ are linearly dependent if a linear combination of the vectors $\sum_{k=1}^{n} \alpha_k \underline{x}_k = 0$, where all the coefficients $\alpha_k$ are not identically equal to zero. If the linear combination $\sum_{k=1}^{n} \alpha_k \underline{x}_k = 0$ only for $\alpha_1 = \alpha_2 = \ldots = \alpha_n = 0$, the vectors are linearly independent.

Orthonormal vectors: The vectors $\{\underline{x}_1, \ldots, \underline{x}_n\}$ form a set of $\underline{M}$-orthonormal vectors, if $\underline{x}^T_i \underline{M} \underline{x}_j = \delta_{ij}$, where $\delta_{ij} = 1$, if $i = j$ and $\delta_{ij} = 0$, if $i \neq j$.

Rayleigh quotient: Considering the generalized eigenproblem $\underline{K}\,\underline{\phi} = \lambda \underline{M}\,\underline{\phi}$, the Rayleigh quotient, $\rho(\underline{x})$ of any vector $\underline{x}$ is defined as $\rho(\underline{x}) = \underline{x}^T \underline{K} \underline{x} / \underline{x}^T \underline{M} \underline{x}$. It should be noted that (i) when the vector $\underline{x}$ is an eigenvector $\underline{\phi}_i$, $\rho(x) = \lambda_i$ (ii) for any vector $\underline{x}$, $\lambda_1 \leq \rho(\underline{x}) \leq \lambda_n$, where $\lambda_1$ and $\lambda_n$ are the extremal eigenvalues of the eigenproblem $\underline{K}\,\underline{\phi} = \lambda \underline{M}\,\underline{\phi}$.

Ritz Analysis: The Ritz analysis is an approximate method that is often employed to solve the eigenproblem $\underline{K}\,\underline{\phi} = \lambda \underline{M}\,\underline{\phi}$. The Ritz analysis consists of: (i) establish a linearly independent set of q vectors, $\underline{X} = \{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_q\}$ (ii) compute the projections of the matrices $\underline{K}$ and $\underline{M}$ onto $\underline{X}$, namely $\tilde{\underline{K}} = \underline{X}^T \underline{K} \underline{X}$ and $\tilde{\underline{M}} = \underline{X}^T \underline{M} \underline{X}$, (iii) solve the qxq eigenproblem $\tilde{\underline{K}} \underline{Z} = \tilde{\underline{M}} \underline{Z} \underline{\rho}$, (iv) the approximation to the q smallest eigenvalues of the eigenproblem

$\underline{K}\,\underline{\phi} = \lambda\,\underline{M}\,\underline{\phi}$ are the elements of the diagonal matrix $\underline{\rho}$ and the approximations to the corresponding q eigenvectors are $\underline{\tilde{\phi}} = \underline{X}\,\underline{Z}$.

The elements of the matrix Z (qxq matrix) are called the Ritz coefficients that relate the vectors in $\underline{X}$ to the eigenvector approximation $\underline{\tilde{\phi}}$.

Subspace: Consider a n-dimensional vector space that is spanned by n-linearly independent vectors $\{\underline{x}_1, \underline{x}_2, \ldots \underline{x}_n\}$. Then, a vector space spanned by any q of the above n basis vectors is called a q-dimensional subspace of the n-dimensional vector space.

Strum Sequence Check: The number of eigenvalues (of the eigenproblem $\underline{K}\,\underline{\phi} = \lambda\,\underline{M}\,\underline{\phi}$) that is smaller than a shift $\mu$ is equal to the number of negative diagonal elements measured in the factor $\underline{D}$, where $\underline{K} - \mu\underline{M} = \underline{L}\,\underline{D}\,\underline{L}^T$.