Two-Step Component Mode Synthesis with Convergence for the Eigensolution of Large-Degree-of-Freedom Systems

by

Anand Ramani

Dissertation submitted to the faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Mechanical Engineering

APPROVED:

Dr. Charles E. Knight, Chairman

Dr. Scott L. Hendricks

Dr. Larry D. Mitchell

Dr. Reginald G. Mitchiner

Dr. Alfred L. Wicks

April 30, 1996
Blacksburg, Virginia

Key Words: Component Mode Synthesis, Non-Linear, Eigenvalue

Two-Step Component Mode Synthesis with Convergence for the Eigensolution of Large-Degree-of-Freedom Systems

by

Anand Ramani

Dr. Charles E. Knight, Chairman

Mechanical Engineering

(ABSTRACT)

Component Mode Synthesis (CMS) is a dynamic substructuring technique for the approximate eigensolution of large-degree-of-freedom (dof) systems divisible into two or more components. System synthesis using component modes results in approximate eigenparameters; in general, using more component modes in synthesis improves the approximation. A two-step CMS approach is developed in this research. The first step involves system synthesis using the minimum number of component modes required to obtain approximate eigenvalues up to a pre-selected cut-off frequency, and the second step introduces additional component modes in a convergence scheme operating on the system eigenparameters calculated in the first step.

The method is developed using constraint modes for system synthesis. The eigenvectors resulting from the initial eigenproblem solution are used to transform the system matrices; this results in a non-linear eigenvalue problem which is solved by a modified shooting method. A perturbation approach is adopted to derive a convergence scheme in which successive iterations are performed for the eigenvalue and eigenvector in each step. A procedure for selecting initial values for the convergence scheme is presented. A condensation procedure is also developed for economical synthesis of systems with many connection coordinates compared to normal mode coordinates. Advantages of the present method include minimal order of system matrices, savings in computation time and a knowledge of the accuracy of the eigenparameters. Numerical examples of spring-mass systems and systems constructed with beam and shell elements are included to demonstrate the applicability of the method and results are compared with full-system eigensolution.

The method is also applicable to the synthesis of generally damped systems. Conventional state-space formulation is used to cast the equations of motion in first-order form for each component. Complex modes are combined with an appropriately defined set of constraint modes to give the mode superset for each component. The method evolves similar to the method for undamped systems. A numerical example is included to demonstrate the method and results from CMS are compared with those obtained by full-system eigensolution. Also, the applicability of the method in solving non-linear eigenvalue problems in structural dynamics is discussed, and examples are included for demonstration.

To My Parents

Acknowledgments

I take this opportunity to thank my adviser Dr. Charles E. Knight, for his support and guidance in my research and in the writing of this dissertation. I wish to convey my sense of gratitude and appreciation for his continuous effort to secure funding for the successful completion of my research. I also wish to thank professors Hendricks, Mitchell, Mitchiner and Wicks for serving on my advisory committee.

My thanks are due to my parents and brother, for their fond love, help and support they extended all through my graduate career.

Finally, I would like to thank my room-mates and friends for their companionship and encouragement that proved instrumental in the development of this work and in the completion of my doctoral degree.

Table of Contents

List of Figures

List of Tables

Notation

$\mathbf{A}$	damped system matrix associated with the derivative of the state vector, after coordinate reduction
$\mathbf{B}$	damped system matrix associated with the state vector, after coordinate reduction
$\mathbf{C}$	matrix of constraint equation coefficients in generalized coordinates
$\mathbf{C}'$	matrix of constraint equation coefficients in physical dof coordinates
$\mathbf{I}$	identity matrix
$\mathbf{K}$	system stiffness matrix after coordinate reduction
$\mathbf{K}'$	system stiffness matrix after introducing additional modes
$\mathbf{M}$	system mass matrix after coordinate reduction
$\mathbf{M}'$	system mass matrix after introducing additional modes
$\mathbf{N}$	functional-lambda matrix
$\mathbf{N}_s$	s^{th} coefficient matrix of functional-lambda matrix
$\overline{\mathbf{N}}$	functional-lambda matrix
$\mathbf{P}$	transformation matrix related to inertia-relief modes
$\mathbf{S}$	transformation matrix for eliminating dependent dofs
$\mathbf{a}$	damped component matrix, associated with the derivative of the state vector
$\mathbf{b}$	damped component matrix, associated with the state vector set of boundary dofs
$\mathbf{c}$	component damping matrix
$\mathbf{e}$	set of excess (redundant) boundary dofs
$\mathbf{f}$	component force vector
f	function
$\mathbf{g}$	component flexibility matrix
$\mathbf{i}$	set of interior dofs
$\mathbf{k}$	component stiffness matrix

$\hat{\mathbf{k}}$ system stiffness matrix in generalized coordinates before coordinate reduction

$\mathbf{m}$ component mass matrix

$\hat{\mathbf{m}}$ system mass matrix in generalized coordinates before coordinate reduction

$\mathbf{p}$ component dof vector in generalized coordinates
system dof vector in generalized coordinates after coordinate reduction

$\tilde{\mathbf{p}}$ system dof vector in generalized coordinates before coordinate reduction

$\mathbf{q}$ final system dof vector in initial system coordinates

$\mathbf{r}$ set of rigid-body dofs

$\mathbf{x}$ component physical dof vector

x spatial coordinate

$\tilde{\mathbf{x}}$ system dof vector in physical coordinates before coordinate reduction

$\mathbf{y}$ state vector for a damped system

y spatial coordinate

Λ diagonal matrix of eigenvalues

Φ initial system eigenvector matrix

α damped component generalized matrix associated with the derivative of the statevector

$\hat{\alpha}$ damped system matrix associated with the derivative of the statevector, before coordinate reduction

β damped component generalized matrix associated with the state vector

$\hat{\beta}$ damped system matrix associated with the statevector, before coordinate reduction

δ Kronecker delta

ε ordering parameter

κ component generalized stiffness matrix

λ eigenvalue

μ component generalized mass matrix

ϕ component mode vector or matrix

ψ component mode superset

ω frequency

χ component generalized damping matrix

ξ damping ratio

Subscripts

0	initial
a	component attachment mode coordinate
b	component boundary coordinate component inertia-relief attachment mode coordinate
c	component constraint mode coordinate system connection coordinate correction
d	component residual inertia-relief attachment mode coordinate dependent coordinate
e	component excess boundary coordinate system additional normal mode coordinate
i	component interior coordinate
ℓ	independent coordinate
m	component inertia-relief mode coordinate
n	component normal mode coordinate system normal mode coordinate natural
r	component rigid-body coordinate

Superscripts

T	transpose
k	iteration number
r	component
s	component
α	component
β	component
$'$	prime symbol

1.0 Introduction

The fundamental problem in structural dynamics is the determination of the displacement response and consequently the internal stresses in a structural system subjected to time-varying force or displacement excitation. Before the advent of computers, such analysis was essentially limited to 'simple' structures like beams and plates, for which some closed-form solutions could be obtained. Numerical methods such as the finite difference method were used for other problems but their applicability was limited. There was a need for a more general numerical technique which could be applied to a wide class of problems. This was felt at a time when there was a widespread need to develop, analyze and optimize the design of complex structures.

The finite element method evolved in the 1960's as a powerful numerical tool for the analysis of complex structural systems, largely spurred by the availability of digital computers to perform numerical computations. The performance and capability of the method was governed by computational speed and memory requirements. The method involves the idealization of a structural system as an assemblage of discrete structural elements to obtain equations of motion of the entire system. These are written in matrix form and the solution of these equations is obtained using operations of matrix algebra. The order of these matrices depends on the number of discretizations. In the analysis of a complex structure, it may be required to discretize the structure into a large number of finite elements, thereby resulting in matrices of large order. Developments in computer science and computer hardware paralleled the demands of even greater performance characteristics. Analysis of structural systems with a few thousand degrees of freedom (dofs) are common, but systems with tens of thousands of dofs can still pose a significant challenge even to the very powerful computers of today.

The linear problem in structural dynamics is usually solved by first solving an eigenvalue problem for natural frequencies and mode shapes up to a pre-determined cut-off frequency. This is determined from an examination of the harmonic content of the excitation. Cook [1] recommends the use of a frequency cut-off of about three times the maximum frequency of excitation. This may require a considerably refined finite element discretization requiring matrices of large order for accurate solution of complex structures. Methods such as the subspace iteration method [2] and Lanczos eigensolution algorithm [3] were developed for the partial eigensolution of multi-dof systems, for which the solution time and memory requirements depend on the size of the system matrices.

Component mode synthesis (CMS) is a dynamic substructuring technique developed for approximate eigensolutions of large dof systems. Also known as substructure coupling, it is the process of partitioning a structure into substructures or components and coupling them using constraint equations. This takes into account the procedures employed in the design of complex structures, wherein different design groups can work independently on different components until the final stage of coupling and also a physically realizable interconnection characterization. The key to the successful application of these methods for large dof systems is the transformation of component physical dof coordinates to generalized coordinates by a modal transformation using substructure modes. With a limited number of substructure modes, this transformation reduces the size of the structure eigenvalue problem while giving surprising accuracy in the eigenparameters.

Component mode synthesis methods have been classified into time-domain and frequency-domain methods. Analytical methods of CMS are based in the time domain and use component generalized stiffness, mass and damping matrices. Frequency-domain CMS methods are based on the use of frequency response functions either directly or after conversion to modal form. The primary aim of frequency-domain CMS is to determine the frequency response functions of a coupled system; determination of system eigenvalues and transient response being secondary.

Another classification of CMS methods is based on the conditions imposed at the interface between two or more substructures when mode shapes are determined for the substructure.

Fixed-interface methods employ normal modes of substructures with interface coordinates fixed, while free-interface methods employ free-free normal modes of all structures. Loaded interface methods are based on compatibility of interface forces and are related to free-interface methods. Hybrid methods of CMS have been developed to permit arbitrary specification of interface conditions and are a combination of the other three types.

Irrespective of the method used, the requirements of a good CMS technique are the following.

(i) The method should be able to predict accurate system response while minimizing memory requirements and solution time.

(ii) The method should be able to use component and coupling data available from conventional analyses.

(iii) The method should permit independent analysis of different components with interaction only at the coupling stage of synthesis.

(iv) The method should allow for interchangability of components so that design changes in the system can be effectively investigated.

(v) The method should allow for arbitrary selection of interface and interior dofs without any restriction.

(vi) The method should allow for the use of any type of component mode in the synthesis. In this respect, the method should be general.

(vii) The component generalized coordinates should include explicit boundary coordinates so that direct-stiffness assembly of system matrices is possible.

(viii) The CMS model should be able to predict pseudo-static response required for the mode-acceleration method accurately.

(ix) The method should allow for use of data obtained from the testing of components.

Early efforts in CMS concentrated on developing various kinds of component modal models from different kinds of component mode supersets, comparison of various modal models and determining the equivalence between them. Initially, the successful application of the method to large dof systems was overwhelming, for the simple reason that such systems could not be solved otherwise in reasonable time and with reasonable accuracy. Subsequently, accuracy was addressed. Since it is not possible to specify beforehand, the number or cut-off frequency of

component modes to obtain a specified accuracy in the system solution, usually a generous number of component modes are used to ensure satisfactory accuracy. This is, however, not optimum. Some studies have been conducted on using the minimum number of component modes for synthesis. Such studies have been done on a limited number of cases and the results are usually case-specific. Approximate and accurate convergence schemes for synthesized modes have been developed, but these trade-off accuracy for solution time or vice-versa, and are usually not practical for large dof systems. The present study addresses these issues and develops a method that is applicable to large-dof systems without any restriction. A key feature of the present method is the ability to use the minimum number of component modes to obtain approximate eigenparameters up to a pre-selected cut-off frequency. In the second step, additional modes are used to converge the approximate eigenvalues previously synthesized. The number of additional modes to be used depends on their availability and cost considerations; in general, using more modes improves the convergence. In the present method, system eigenvalues are converged as accurately as possible and the solution times are less than in conventional methods. The convergence scheme is so developed that individual eigenvalues can be chosen for convergence.

This dissertation is organized into several chapters. Chapter 2 describes a generalized method for CMS and the kinds of component modes and mode supersets used. A survey of relevant literature on CMS methods is also presented. Chapter 3 discusses preliminaries to the development of the present method and establishes the motivation for the present study. This is followed by a detailed derivation of the method. As mentioned before, the method is developed as a two-step approach in which the first step involves eigensolution using the minimum number of component modes. In the second step, a perturbation approach is used to derive a convergence scheme operating on the initial eigenparameters. The convergence scheme is designed so that separate iterations are used to converge the eigenvalue and eigenvector. The final result is a set of iteration expressions for the eigenvalue and eigenvector. A procedure for choosing initial eigenvalues is described. The method was implemented in a FORTRAN code to run on IBM RISC-6000 workstations. Chapter 4 discusses the details of program implementation and the algorithms used. The method was then tested on some example spring-mass systems and structures constructed from beam and shell elements. Details of finite element models

developed, results obtained and solution times are provided in Chapter 5. Chapter 6 discusses the applicability of the method for the synthesis of generally damped systems. A spring-mass-damper system constructed from three components is used for demonstration. Chapter 7 describes a method for solving non-linear eigenvalue problems arising in structural dynamics, based on the present scheme. Final comments and suggestions for future work are presented in Chapter 8.

2.0 Overview of CMS Literature

In this chapter, the general procedure of CMS as found in the literature is first described. This is followed by a survey of literature on CMS methods in general, establishing the motivation for the present research.

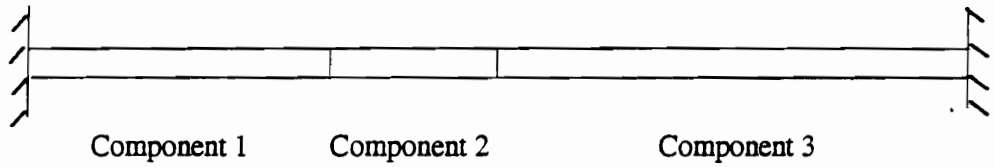
2.1 Method of Component Mode Synthesis

Hurty [4] is credited with first presenting the technique of CMS in 1965. Since then, several methods for the coupling of substructures in dynamic analysis have been presented. In this section, the most general method of component mode synthesis is first developed; this is largely based on a review of time-domain and frequency-domain CMS methods presented by Craig [5].

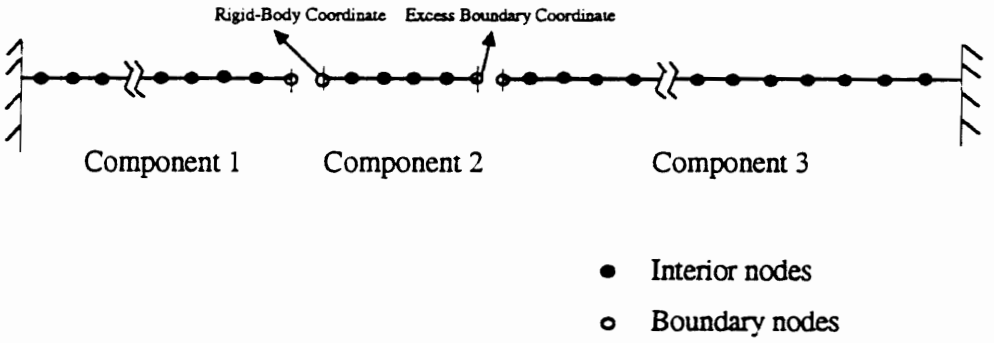
2.1.1 Definition of Components

In CMS, a structure is divided into two or more substructures or components which are interconnected to each other at interfaces. Figure 2.1 shows a schematic layout of components in a beam structure interconnected to each other at the interface or boundary nodes; each interior and boundary node having both translational and rotational dofs.

Let **k**, **m** and **c** denote the stiffness, mass and damping matrices of a component respectively. The linear equation of motion of the component is



System



Finite Element Discretization

Figure 2.1. Components in a Beam Structure with Interior and Boundary Nodes

$$\mathbf{m}\ddot{\mathbf{x}} + \mathbf{c}\dot{\mathbf{x}} + \mathbf{k}\mathbf{x} = \mathbf{f}, \tag{2.1}$$

where **x** is the vector of component displacement dof coordinates and **f**, the vector of applied forces. Let the set of component dof coordinates **x**, be divided into the set **i** of interior coordinates and the set **b** of boundary coordinates. Further, let the set of boundary coordinates be divided into a set **r** that provides statically determinate support of the component (also known as the rigid-body coordinates), and its complementary set **e**, which constitutes the excess (redundant) boundary coordinates. A representation of interior and boundary coordinates is also shown in Figure 2.1. Also shown are the rigid-body and excess boundary coordinates for one of the components.

Component mode synthesis consists of two steps -

(i) definition of component modes and

(ii) coupling of component modes into a system.

2.1.2 Component Modes

Several kinds of component modes are used to characterize the motion of a component. These can be broadly divided into static and dynamic component modes. Static modes are required for the assembly of system matrices by the 'direct-stiffness' approach. These may be constraint modes or attachment modes and are defined specific to each interface dof as described later. Dynamic component modes include the rigid-body modes and the normal modes; these represent the motion of the component. A third kind of dynamic modes called the inertia-relief modes are used to account for the flexibility and inertia effects of the truncated normal modes. A brief description of some of the component modes is presented below.

(i) **Normal Modes:** The real component normal modes, ϕ, are determined by solving the eigenvalue problem

$$(\mathbf{k} - \omega^2 \mathbf{m})\phi = \mathbf{0}, \tag{2.2}$$

where ω is the natural frequency and ϕ is the eigenvector corresponding to the frequency ω. The eigenvectors are arranged column-wise to form the modal matrix ϕ_n, which has the following properties.

$$\phi_n^T \mathbf{m}\, \phi_n = \mathbf{I}_{nn};$$
$$\phi_n^T \mathbf{k}\, \phi_n = \Lambda_{nn}, \tag{2.3}$$

where Λ_{nn} is the diagonal matrix of the square of the natural frequencies. Two types of normal modes can be defined with interface coordinates free or fixed. Accordingly, they are called free and fixed-interface normal modes. For free-interface normal modes, ϕ_n has the form

$$\phi_n = \begin{bmatrix} \phi_{in} \\ \phi_{bn} \end{bmatrix}, \tag{2.4}$$

where the first subscript in the sub-matrices denotes the coordinate and the second subscript denotes the mode. In the above equation, the subscript 'i' denotes interior dofs and the subscript 'b' denotes boundary dofs. For fixed-interface normal modes, ϕ_n has the form

$$\phi_n = \begin{bmatrix} \phi_{in} \\ \mathbf{0}_{bn} \end{bmatrix}. \tag{2.5}$$

It is customary to truncate the number of normal modes to 'k' kept modes. Then the kept component normal mode set is designated as ϕ_k.

(ii) Rigid-Body Modes: The rigid-body modes, ϕ_r, are defined with respect to the **r** set of coordinates as

$$\begin{bmatrix} \mathbf{k}_{ii} & \mathbf{k}_{ie} & \mathbf{k}_{ir} \\ \mathbf{k}_{ei} & \mathbf{k}_{ee} & \mathbf{k}_{er} \\ \mathbf{k}_{ri} & \mathbf{k}_{re} & \mathbf{k}_{rr} \end{bmatrix} \begin{bmatrix} \phi_{ir} \\ \phi_{er} \\ \mathbf{I}_{rr} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{2.6}$$

so that, the rigid-body modes may be obtained by considering the first two rows of equation (2.6) to solve for the sub-matrices ϕ_{ir} and ϕ_{er} as

$$\phi_r = \begin{bmatrix} \phi_{ir} \\ \phi_{er} \\ \mathbf{I}_{rr} \end{bmatrix} = \begin{bmatrix} -\begin{bmatrix} \mathbf{k}_{ii} & \mathbf{k}_{ie} \\ \mathbf{k}_{ei} & \mathbf{k}_{ee} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{k}_{ir} \\ \mathbf{k}_{er} \end{bmatrix} \\ \mathbf{I}_{rr} \end{bmatrix}. \tag{2.7}$$

Here, the subscript 'r' denotes rigid-body dofs (or mode) and 'e' denotes the excess boundary dofs respectively, so that the selection of rigid-body dofs from among the interface dofs is mandatory.

(iii) Redundant Constraint Modes: A constraint mode is defined as the static displacement resulting from an imposed unit displacement on one coordinate of the **e** set, while the component is supported on the **r** set. Such a displacement pattern is obtained for each coordinate in the **e** set fixed in turn. Therefore,

$$\begin{bmatrix} \mathbf{k}_{ii} & \mathbf{k}_{ie} & \mathbf{k}_{ir} \\ \mathbf{k}_{ei} & \mathbf{k}_{ee} & \mathbf{k}_{er} \\ \mathbf{k}_{ri} & \mathbf{k}_{re} & \mathbf{k}_{rr} \end{bmatrix} \begin{bmatrix} \phi_{ie} \\ \mathbf{I}_{ee} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{R}_{ee} \\ \mathbf{R}_{re} \end{bmatrix}, \tag{2.8}$$

where $\mathbf{R}_{ee}$ and $\mathbf{R}_{re}$ are nodal reaction forces. Then, the set of constraint modes, ϕ_c, is given by

$$\phi_c = \begin{bmatrix} \phi_{ie} \\ \mathbf{I}_{ee} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} -\mathbf{k}_{ii}^{-1}\mathbf{k}_{ie} \\ \mathbf{I}_{ee} \\ \mathbf{0} \end{bmatrix}. \tag{2.9}$$

This approach was proposed by Hurty [4]. Craig and Chang [6] however, do not make the distinction between statically determinate and redundant boundary coordinates; instead the rigid-body modes are considered a special case of constraint modes.

(iv) **Attachment Modes:** An attachment mode is defined as the static displacement resulting from applying a unit force on one of the coordinates of the **e** set while the component is supported on the **r** set. Such a displacement pattern is obtained by applying a unit force to each coordinate in the **e** set in turn. Therefore,

$$\begin{bmatrix} \mathbf{k}_{ii} & \mathbf{k}_{ie} & \mathbf{k}_{ir} \\ \mathbf{k}_{ei} & \mathbf{k}_{ee} & \mathbf{k}_{er} \\ \mathbf{k}_{ri} & \mathbf{k}_{re} & \mathbf{k}_{rr} \end{bmatrix} \begin{bmatrix} \phi_{ie} \\ \phi_{ee} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{ee} \\ \mathbf{R}_{re} \end{bmatrix}, \tag{2.10}$$

so that the set of attachment modes, ϕ_a, is given by

$$\phi_a = \begin{bmatrix} \phi_{ie} \\ \phi_{ee} \\ \mathbf{0} \end{bmatrix} = \left[-\begin{bmatrix} \mathbf{k}_{ii} & \mathbf{k}_{ie} \\ \mathbf{k}_{ei} & \mathbf{k}_{ee} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{ee} \end{bmatrix} \right]. \tag{2.11}$$

(v) **Inertia-Relief Modes:** These modes are not strictly necessary for CMS; however they are required to represent complete static response in the mode acceleration method [7]. Two types of inertia-relief modes have been defined. The ϕ_m type of inertia-relief modes were defined by Hintz [7] as the static displacement of a component which is loaded by D'Alembert's forces due to the rigid-body motion and supported in the **r** set. This results in the equation

$$\begin{bmatrix} \mathbf{k}_{ii} & \mathbf{k}_{ie} & \mathbf{k}_{ir} \\ \mathbf{k}_{ei} & \mathbf{k}_{ee} & \mathbf{k}_{er} \\ \mathbf{k}_{ri} & \mathbf{k}_{re} & \mathbf{k}_{rr} \end{bmatrix} \begin{bmatrix} \phi_{im} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_{ii} & \mathbf{m}_{ie} & \mathbf{m}_{ir} \\ \mathbf{m}_{ei} & \mathbf{m}_{ee} & \mathbf{m}_{er} \\ \mathbf{m}_{ri} & \mathbf{m}_{re} & \mathbf{m}_{rr} \end{bmatrix} \begin{bmatrix} \phi_{ir} \\ \phi_{er} \\ \mathbf{I}_{rr} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{R}_{er} \\ \mathbf{R}_{rr} \end{bmatrix}, \tag{2.12}$$

where the rigid-body modes of equation (2.6) and the matrix of reaction forces of equation (2.8) have been used in the definition and there are as many inertia-relief modes as the rigid-body modes.

Therefore, the inertia-relief mode set ϕ_m, is given by

$$\phi_m = \begin{bmatrix} \phi_{im} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{Bmatrix} \mathbf{k}_{ii}^{-1}(\mathbf{m}_{ii}\phi_{ir} + \mathbf{m}_{ie}\phi_{er} + \mathbf{m}_{ir}) \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix}. \tag{2.13}$$

This definition is equivalent to that given by Herting [8].

The second type of inertia-relief modes, also called the inertia-relief attachment modes and denoted by ϕ_b, are defined orthogonal (with respect to the mass matrix) to the rigid-body modes by applying a unit force to each boundary coordinate in the set **b** in turn and equilibrating these forces with the D'Alembert's forces which would result from the application of these unit forces on the boundary. MacNeal [9] and Rubin [10] employed these inertia-relief modes. Following these references, we have

$$\phi_b = \mathbf{P}^T \mathbf{g}\, \mathbf{P}\, \mathbf{f}_b, \tag{2.14}$$

where

$$\mathbf{P} = \mathbf{I} - \mathbf{m}\, \phi_r\, \mu_{rr}^{-1}\, \phi_r{}^T, \tag{2.15}$$

$$\mu_{rr} = \phi_r{}^T \mathbf{m}\, \phi_r, \tag{2.16}$$

$$\mathbf{g} = \begin{bmatrix} \begin{bmatrix} \mathbf{k}_{ii} & \mathbf{k}_{ie} \\ \mathbf{k}_{ei} & \mathbf{k}_{ee} \end{bmatrix}^{-1} & \mathbf{0} \\ \mathbf{0}\ \ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{2.17}$$

and

$$\mathbf{f}_b = \begin{Bmatrix} \mathbf{0} \\ \mathbf{I}_{bb} \end{Bmatrix}. \tag{2.18}$$

(vi) **Residual Inertia-Relief Attachment Modes:** These are similar to inertia-relief attachment modes, but account for the flexibility of the truncated normal modes. These were employed by Craig and Chang [11] and remove the ϕ_r and ϕ_k contributions from the other modes. The residual inertia-relief attachment mode set denoted by ϕ_d, is defined as

$$\phi_d = [\mathbf{P}^T \mathbf{g}\, \mathbf{P} - \phi_k \Lambda_{kk}^{-1} \phi_k^T]\, \mathbf{f}_b, \tag{2.19}$$

where the term in parenthesis is the residual flexibility matrix.

2.1.3 Component Mode Supersets

The set of component modes form a transformation matrix ψ, which transforms the component dof coordinates $\mathbf{x}$ to component generalized coordinates $\mathbf{p}$, i.e.,

$$\mathbf{x} = \psi \mathbf{p}. \tag{2.20}$$

Four different (dynamic) component mode supersets are widely used in CMS. These are listed below, but there are other variations.

(i) **Constraint Mode Superset:** This superset denoted by ψ_c, consists of the constraint modes ϕ_c, the inertia-relief modes ϕ_m, the rigid-body modes ϕ_r, and the kept normal modes ϕ_k. The normal modes are fixed-interface normal modes. Thus

$$\psi_c = [\phi_c \;\; \phi_m \;\; \phi_r \;\; \phi_k]. \tag{2.21}$$

This set was defined by Hintz [7].

(ii) **Attachment Mode Superset:** This superset denoted by ψ_a, consists of the attachment modes ϕ_a, the inertia-relief modes ϕ_m, the rigid-body modes ϕ_r, and the kept normal modes ϕ_k. The normal modes are free-interface normal modes. Thus

$$\psi_a = [\ \phi_a\ \ \phi_m\ \ \phi_r\ \ \phi_k\ ]. \tag{2.22}$$

This set was also defined by Hintz [7].

(iii) Inertia-Relief Attachment Mode Superset: This superset denoted by ψ_b, consists of the inertia-relief attachment modes ϕ_b, the rigid-body modes ϕ_r, and the kept free-interface normal modes ϕ_k. Thus

$$\psi_b = [\ \phi_b\ \ \phi_r\ \ \phi_k\ ]. \tag{2.23}$$

(iv) Residual Inertia-Relief Attachment Mode Superset: This superset denoted by ψ_d, consists of the residual inertia-relief attachment modes ϕ_d, the rigid-body modes ϕ_r, and the kept free-interface normal modes ϕ_k. Thus

$$\psi_d = [\ \phi_d\ \ \phi_r\ \ \phi_k\ ]. \tag{2.24}$$

Any one of these mode supersets can be used for CMS. There are other variations also. For example Hurty [4] employed the constraint mode superset without the inertia-relief modes and Herting [8] used the constraint mode superset with free-interface normal modes instead of fixed-interface normal modes.

2.1.4 Component Modal Models

Let the component mode superset used be denoted by ψ. For each component, the following component generalized matrices can be determined.

$\kappa = \psi^T \mathbf{k} \psi$;

$\mu = \psi^T \mathbf{m} \psi$;

$$\chi = \psi^T \mathbf{c}\, \psi \,. \tag{2.25}$$

The equation of motion of the component in the **p** coordinates is

$$\mu \ddot{\mathbf{p}} + \chi \dot{\mathbf{p}} + \kappa \mathbf{p} = \psi^T \mathbf{f} \,. \tag{2.26}$$

It is customary to omit the damping matrix in the formulation and treat damping as a total system parameter. (Component mode synthesis for generally damped systems is described in Chapter 6.) Hence, the equations of motion for undamped free vibration are given by

$$\mu \ddot{\mathbf{p}} + \kappa \mathbf{p} = \mathbf{0} \,. \tag{2.27}$$

Denoting $\phi_i^T \mathbf{k}\, \phi_j$ by κ_{ij} and $\phi_i^T \mathbf{m}\, \phi_j$ by μ_{ij}, where i and j denote the type of component mode, i.e., constraint, attachment, inertia-relief, inertia-relief attachment, residual inertia-relief attachment, rigid-body, or normal mode, the following component generalized stiffness matrix κ and mass matrix μ are obtained for each mode superset used [5].

For the constraint mode superset,

$$\kappa = \begin{bmatrix} \mathbf{k}_{cc} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{k}_{mm} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \Lambda_{kk} \end{bmatrix} \; ; \quad \mu = \begin{bmatrix} \mu_{cc} & \mu_{cm} & \mu_{cr} & \mu_{ck} \\ \mu_{mc} & \mu_{mm} & \mu_{mr} & \mu_{mk} \\ \mu_{rc} & \mu_{rm} & \mathbf{I}_{rr} & \mathbf{0} \\ \mu_{kc} & \mu_{km} & \mathbf{0} & \mathbf{I}_{kk} \end{bmatrix} . \tag{2.28}$$

For the attachment mode superset,

$$\kappa = \begin{bmatrix} \kappa_{aa} & \mathbf{0} & \mathbf{0} & \kappa_{ak} \\ \mathbf{0} & \kappa_{mm} & \mathbf{0} & \kappa_{mk} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \kappa_{ka} & \kappa_{km} & \mathbf{0} & \Lambda_{kk} \end{bmatrix} \; ; \quad \mu = \begin{bmatrix} \mu_{aa} & \mu_{am} & \mu_{ar} & \mu_{ak} \\ \mu_{ma} & \mu_{mm} & \mu_{mr} & \mu_{mk} \\ \mu_{ra} & \mu_{rm} & \mathbf{I}_{rr} & \mathbf{0} \\ \mu_{ka} & \mu_{km} & \mathbf{0} & \mathbf{I}_{kk} \end{bmatrix} . \tag{2.29}$$

For the inertia-relief attachment mode superset,

$$\kappa = \begin{bmatrix} \kappa_{bb} & 0 & \kappa_{bk} \\ 0 & 0 & 0 \\ \kappa_{kb} & 0 & \Lambda_{kk} \end{bmatrix} ; \quad \mu = \begin{bmatrix} \mu_{bb} & 0 & \mu_{bk} \\ 0 & I_{rr} & 0 \\ \mu_{kb} & 0 & I_{kk} \end{bmatrix} . \tag{2.30}$$

For the residual inertia-relief attachment mode superset

$$\kappa = \begin{bmatrix} \kappa_{dd} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Lambda_{kk} \end{bmatrix} ; \quad \mu = \begin{bmatrix} \mu_{dd} & 0 & 0 \\ 0 & I_{rr} & 0 \\ 0 & 0 & I_{kk} \end{bmatrix} . \tag{2.31}$$

The forms of these component generalized matrices determine the forms of the stiffness and mass matrices obtained after coupling the components to form a system. The forms of system matrices appear in the next chapter.

2.1.5 Coupling of Components

Craig and Chang [12] introduced a generalized procedure for coupling components to give system equations of motion. For explanatory purposes, consider two components 'α' and 'β' in a system. Component generalized stiffness and mass matrices κ^{α} and μ^{α} respectively, can be determined for each component 'α'. The component displacement dof vector is given by

$$\mathbf{x}^{\alpha} = \psi^{\alpha} \mathbf{p}^{\alpha} , \tag{2.32}$$

where the $\mathbf{p}^{\alpha}$ are generalized coordinates for component α. For interface connections with coincident nodes, interface compatibility requires that

$$\mathbf{x}_b{}^{\alpha} = \mathbf{x}_b{}^{\beta} . \tag{2.33}$$

Constraint equations of this type can be written of the form

$$\mathbf{C}'\,\tilde{\mathbf{x}} = \mathbf{0}, \tag{2.34}$$

where $\mathbf{C}'$ is a matrix of coefficients and

$$\tilde{\mathbf{x}} = \begin{Bmatrix} \mathbf{x}^{\alpha} \\ \mathbf{x}^{\beta} \\ \vdots \end{Bmatrix}. \tag{2.35}$$

These equations can also be written in the form

$$\mathbf{C}\,\tilde{\mathbf{p}} = \mathbf{0}, \tag{2.36}$$

where

$$\tilde{\mathbf{p}} = \begin{Bmatrix} \mathbf{p}^{\alpha} \\ \mathbf{p}^{\beta} \\ \vdots \end{Bmatrix}. \tag{2.37}$$

These constraint equations imply the dependence of some coordinates on the remaining coordinates. Thus, the coordinates in $\tilde{\mathbf{p}}$ can be partitioned as dependent coordinates $\tilde{\mathbf{p}}_{\mathrm{d}}$ and independent coordinates $\tilde{\mathbf{p}}_{\ell}$. Then the constraint equations take the form

$$\begin{bmatrix} \mathbf{C}_{\mathrm{dd}} & \mathbf{C}_{\mathrm{d}\ell} \end{bmatrix} \begin{Bmatrix} \tilde{\mathbf{p}}_{\mathrm{d}} \\ \tilde{\mathbf{p}}_{\ell} \end{Bmatrix} = \mathbf{0}, \tag{2.38}$$

so that

$$\tilde{\mathbf{p}} = \mathbf{S}\,\mathbf{p}\,, \tag{2.39}$$

where

$$\mathbf{S} = \begin{bmatrix} -\mathbf{C}_{dd}^{-1}\mathbf{C}_{d\ell} \\ \mathbf{I}_{\ell\ell} \end{bmatrix} \tag{2.40}$$

and

$$\mathbf{p} = \tilde{\mathbf{p}}_{\ell}\,. \tag{2.41}$$

Let

$$\hat{\mathbf{k}} = \begin{bmatrix} \kappa^{\alpha} & \mathbf{0} & \cdots \\ \mathbf{0} & \kappa^{\beta} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}; \quad \hat{\mathbf{m}} = \begin{bmatrix} \mu^{\alpha} & \mathbf{0} & \cdots \\ \mathbf{0} & \mu^{\beta} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}. \tag{2.42}$$

Then, the final system stiffness and mass matrices take the form

$$\begin{aligned} \mathbf{K} &= \mathbf{S}^{\mathrm{T}}\hat{\kappa}\mathbf{S}\,; \\ \mathbf{M} &= \mathbf{S}^{\mathrm{T}}\hat{\mu}\mathbf{S}\,, \end{aligned} \tag{2.43}$$

so that

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{0} \tag{2.44}$$

is the final form of the system equations for free-body motion.

2.2 Conventional Methods

The common approach to solve for system modes up to a cut-off frequency ω_c is to solve for the normal modes of each component up to a multiple of the cut-off frequency $x\omega_c$, where, in general $x > 2$. Conventional methods of CMS may use values of x up to 10 to ensure accuracy in the calculated system modes. When all the calculated component modes are assembled, the system eigenproblem is solved in one step, usually for all the modes with frequency less than or equal to ω_c. Thus, in this one-step solution, the accuracy of the calculated system modes cannot be estimated. To study the convergence of calculated system modes, additional component modes have to be introduced. This also means that the CMS problem has to be reformulated using the additional modes. This results in a larger eigenvalue problem which has to be solved. It will be better if the system solution can be obtained by using the minimum number of component modes, followed by the introduction of additional modes in a convergence scheme operating on the initially calculated system modes. This forms the motivation and the subject of the present research.

2.3 Review of Literature

First, a review of the evolution of CMS as a tool for analyzing large-dof-systems is presented. Studies on its applicability to damped system synthesis and experimental testing are listed. Finally, the issues of accuracy and convergence are addressed. Past studies on accuracy and convergence form the basis for the present research.

Hurty [4] was the first to develop a substructure coupling method for analyzing substructures with redundant interface connections. He used a mode superset consisting of fixed-interface normal modes, rigid-body modes and redundant constraint modes.

Bamford [13] introduced attachment modes and developed a hybrid substructure coupling method.

Craig and Bampton [6] and Bajan and Feng [14] modified Hurty's method. According to them, it was unnecessary to divide the constraint modes into rigid-body modes and redundant constraint modes.

Goldman [15] and Hou [16] developed methods using free-interface normal modes of substructures. Goldman devised an explicit method of coupling of substructures. This method does not involve the elimination of dependent coordinates in the final system equations and hence implies no reduction in the number of generalized coordinates during coupling. Consequently, the Goldman method leads to some extraneous frequencies and modes.

Benfield and Hruda [17] used Guyan reduction [18] to determine interface loading and a different coupling strategy to develop four methods, which they described as free-free, constrained, free-free with interface loading and constrained with interface loading.

MacNeal [9] developed a hybrid method in which some substructure interface coordinates are constrained while others are free. He used statically derived deflection influence coefficients to improve the representation of substructure motion. The advantage of the method is that the boundary conditions at component interfaces may be selected to optimize the accuracy. Also, in the event the modes have been obtained, it permits available data to be used.

Rubin [10] used free-interface normal modes augmented with a low-frequency account for the contribution of residual modes. The residual effects improve the accuracy of forced system response by the mode-acceleration method by adding inertial effects due to the higher modes.

Hintz [7] defined two statically complete interface mode sets - the attachment mode set and the constraint mode set. To produce a dynamic component mode superset, the attachment mode set is combined either with free or fixed-interface normal modes. The constraint mode set is however combined only with fixed-interface normal modes.

Craig and Chang [12] described a generalized procedure for substructure coupling. This was an implicit method in contrast to the method suggested by Goldman. The method results in a reduction in the number of interface coordinates due to coupling by using transformation equations to eliminate the dependent coordinates in the final system equations.

Several papers have been published on the use of analytically derived component eigenfunctions for synthesis of system eigenfunctions, akin to the numerical method of CMS. Arora and Nguyen [19] first published a paper in 1980 describing the technique of analytical CMS. At about the same time, Hurty [20] described an analytical CMS method for beam-type structures using admissible modal functions of substructures. Hale and Meirovitch [21] describe a generalized substructure synthesis method using component eigenfunctions.

CMS methods have also been developed for the synthesis of generally damped systems. Hasselman and Kaplan [22] presented a coupling procedure based on the Craig-Bampton method [6] which employs complex component modes. Chung and Craig [23] developed a first-order form of coupling leading to complex component modes and proposed a procedure for forming a type of attachment mode to represent the component modes left out of the component modal model. Howsman and Craig [24] developed a consistent state-space formulation using variational principles. This employed free-interface component modes along with a set of attachment modes. Beliveau and Soucy [25] extended the Craig-Bampton method for damped systems with symmetric substructure matrices by replacing the real fixed-interface normal modes by the corresponding complex modes. Kubomura [26] formulated and investigated methods with three different types of modes - complex free-free, cantilever and hybrid modes. These methods were capable of retaining the content of any frequency range in the component generalized matrices. Craig and Ni [27] developed a model order-reduction method for damped structures using projection matrices to make the method applicable for systems having rigid-body modes.

On the experimental front, Klosterman [28] made a comprehensive study of the experimental determination of modal representations of a structure, including the use of these models in substructure coupling. Klosterman and McClelland [29] investigated the combination of analytical and experimental data and developed a coupling procedure for two substructures

wherein one is represented by a modal model and the other by a finite element model. Klahs and Townley [30] combined the basic approaches of modal analysis, system dynamic analysis and finite element analysis into an automated procedure for determining component loads and stresses. Lamontia [31] has discussed practical techniques for acquiring residual flexibility matrix data from modal tests. Ewins and Sainsbury [32] have conducted studies on the accuracy and types of data required for experimental CMS in the frequency domain. Poelart [33] has used frequency-domain CMS for determining system eigenvalues, while Craig, Bachmeyer and Howsman [34] used inverse Fourier transformations to compute transient response for frequency-domain CMS models.

Several studies on the convergence of synthesized modes have been conducted. Hale and Meirovitch [35] put forward the concept of intermediate structure and proposed a CMS method which incorporates a form of substructure-based iteration called component-mode iteration. Essentially, the method is a form of subspace iteration for large structures in which iteration is carried out at the sub-system level rather than at the structure level, with care being taken to ensure interface compatibility in each iteration. Bennighof [36] dispensed with the intermediate structure, resulting in a higher order eigenequation. Xu, Luo and Han [37] developed a new concept of intermediate structure in which the step of ensuring interface compatibility is eliminated, while still retaining the merits of the subspace iteration method [2]. Hale [38] extended the method of component mode iteration for damped systems. The objective of these methods was to use parallel computing to iterate at the substructure level simultaneously, doing away with the classical idea of CMS as it was originally intended.

Limited studies have concentrated on converging already calculated system modes with additionally evaluated component modes. Hasselman and Hart [39] developed a minimization procedure for converging synthesized modes. This was based on using the neglected component modes for converging system modes accurately. Engels [40] developed an approximate convergence method using all the neglected component modes. Engels also derived an apriori condition for convergence - that the maximum frequency of component modes used in the initial solution be twice the cut-off frequency of modes to be converged. These methods are usually impracticable for systems with large dofs in which it is not feasible to evaluate all the component

modes. Thus, as mentioned earlier, there is a need for a CMS method with a convergence scheme using additionally evaluated component modes that do not include all the modes neglected in the initial synthesis. Such a method should be time-effective, use only a limited number of additional modes and give the analyst a knowledge of the accuracy of the eigenparameters. The following chapter details the development of the method.

3.0 Development of the Method

This chapter details the development of the present scheme of CMS as a two-step method. But first, preliminaries to the development of the method are described to establish the motivation for the present study.

3.1 Forms of System Matrices

In the following discussion, the explicit inertia-relief modes, ϕ_m, in the set of component modes are disregarded. Consider the initial system, with a total of 'n' component normal modes and 'c' interface (constraint, attachment, inertia-relief attachment or residual inertia-relief attachment) modes. The system matrices can be partitioned as follows.

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{cc} & \mathbf{K}_{cn} \\ \mathbf{K}_{nc} & \mathbf{K}_{nn} \end{bmatrix} ; \mathbf{M} = \begin{bmatrix} \mathbf{M}_{cc} & \mathbf{M}_{cn} \\ \mathbf{M}_{nc} & \mathbf{M}_{nn} \end{bmatrix}, \tag{3.1}$$

where the subscript 'c' refers to connection coordinates and the subscript 'n' refers to normal mode coordinates. The system equations of motion are described by equation (2.44), where $\mathbf{p}$ is partitioned as

$$\mathbf{p} = \begin{Bmatrix} \mathbf{p}_c \\ \mathbf{p}_n \end{Bmatrix}. \tag{3.2}$$

To converge the initial system eigenvalues and eigenvectors, a set of 'e' extra component normal modes are introduced. This set forms a subset of the component modes not solved for initially.

When the extra component modes are introduces, the resulting system matrices (distinguished from those in equation (3.1) by the ' ' ' symbol) can be partitioned as

$$\mathbf{K}' = \begin{bmatrix} \mathbf{K}'_{cc} & \mathbf{K}'_{cn} & \mathbf{K}'_{ce} \\ \mathbf{K}'_{nc} & \mathbf{K}'_{nn} & \mathbf{K}'_{ne} \\ \mathbf{K}'_{ec} & \mathbf{K}'_{en} & \mathbf{K}'_{ee} \end{bmatrix} ; \; \mathbf{M}' = \begin{bmatrix} \mathbf{M}'_{cc} & \mathbf{M}'_{cn} & \mathbf{M}'_{ce} \\ \mathbf{M}'_{nc} & \mathbf{M}'_{nn} & \mathbf{M}'_{ne} \\ \mathbf{M}'_{ec} & \mathbf{M}'_{en} & \mathbf{M}'_{ee} \end{bmatrix}, \tag{3.3}$$

where the subscript 'e' refers to the extra normal mode coordinates and the ' ' ' symbol has been used to distinguish the submatrices from those in equation (3.1). The system dof vector now takes the form

$$\mathbf{p}' = \begin{Bmatrix} \mathbf{p}_c \\ \mathbf{p}_n \\ \mathbf{p}_e \end{Bmatrix}, \tag{3.4}$$

where $\mathbf{p}_e$ are the extra normal mode coordinates. From the structure of the component generalized stiffness and mass matrices given in equation (2.28) through equation (2.31), and using equations (3.1) and (3.3), the following can be inferred.

For systems synthesized using constraint mode supersets,

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{cc} & \mathbf{0} \\ \mathbf{0} & \Lambda_{nn} \end{bmatrix} ; \quad \mathbf{M} = \begin{bmatrix} \mathbf{M}_{cc} & \mathbf{M}_{cn} \\ \mathbf{M}_{nc} & \mathbf{I}_{nn} \end{bmatrix};$$

$$\mathbf{K}' = \begin{bmatrix} \mathbf{K}_{cc} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Lambda_{nn} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Lambda_{ee} \end{bmatrix} ; \; \mathbf{M}' = \begin{bmatrix} \mathbf{M}_{cc} & \mathbf{M}_{cn} & \mathbf{M}_{ce} \\ \mathbf{M}_{nc} & \mathbf{I}_{nn} & \mathbf{0} \\ \mathbf{M}_{ec} & \mathbf{0} & \mathbf{I}_{ee} \end{bmatrix}. \tag{3.5}$$

For systems synthesized using attachment mode supersets or inertia-relief attachment mode supersets,

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{cc} & \mathbf{K}_{cn} \\ \mathbf{K}_{nc} & \mathbf{K}_{nn} \end{bmatrix} ; \quad \mathbf{M} = \begin{bmatrix} \mathbf{M}_{cc} & \mathbf{M}_{cn} \\ \mathbf{M}_{nc} & \mathbf{M}_{nn} \end{bmatrix} ;$$

$$\mathbf{K}' = \begin{bmatrix} \mathbf{K}_{cc} & \mathbf{K}_{cn} & \mathbf{K}_{ce} \\ \mathbf{K}_{nc} & \mathbf{K}_{nn} & \mathbf{K}_{ne} \\ \mathbf{K}_{ec} & \mathbf{K}_{en} & \mathbf{K}_{ee} \end{bmatrix} ; \quad \mathbf{M}' = \begin{bmatrix} \mathbf{M}_{cc} & \mathbf{M}_{cn} & \mathbf{M}_{ce} \\ \mathbf{M}_{nc} & \mathbf{M}_{nn} & \mathbf{M}_{ne} \\ \mathbf{M}_{ec} & \mathbf{M}_{en} & \mathbf{M}_{ee} \end{bmatrix} . \tag{3.6}$$

For systems synthesized using residual inertia-relief attachment mode supersets,

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{cc} & \mathbf{K}_{cn} \\ \mathbf{K}_{nc} & \mathbf{K}_{nn} \end{bmatrix} ; \quad \mathbf{M} = \begin{bmatrix} \mathbf{M}_{cc} & \mathbf{M}_{cn} \\ \mathbf{M}_{nc} & \mathbf{M}_{nn} \end{bmatrix} ;$$

$$\mathbf{K}' = \begin{bmatrix} \mathbf{K}'_{cc} & \mathbf{K}'_{cn} & \mathbf{K}'_{ce} \\ \mathbf{K}'_{nc} & \mathbf{K}'_{nn} & \mathbf{K}'_{ne} \\ \mathbf{K}'_{ec} & \mathbf{K}'_{en} & \mathbf{K}'_{ee} \end{bmatrix} ; \quad \mathbf{M}' = \begin{bmatrix} \mathbf{M}'_{cc} & \mathbf{M}'_{cn} & \mathbf{M}'_{ce} \\ \mathbf{M}'_{nc} & \mathbf{M}'_{nn} & \mathbf{M}'_{ne} \\ \mathbf{M}'_{ec} & \mathbf{M}'_{en} & \mathbf{M}'_{ee} \end{bmatrix} . \tag{3.7}$$

Hence, as seen from equations (3.5) and (3.6), for systems synthesized using constraint mode, attachment mode and inertia-relief attachment mode supersets, the initial system matrices can be simply augmented to give the modified system matrices when additional modes are introduced. For the case of residual inertia-relief attachment mode supersets, this is not possible as both the stiffness and mass matrices change entirely in the modified system. Thus, for convergence of system modes using this approach of augmenting the system matrices, residual inertia-relief attachment mode supersets cannot be used. Of the other three mode supersets, the constraint mode superset is attractive because of the simpler form of the stiffness matrix for the modified system with additional modes. For the case of the constraint-mode superset, the equations of motion for free vibrations of the modified system are given below as

$$\begin{bmatrix} \mathbf{K}_{cc} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Lambda_{nn} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Lambda_{ee} \end{bmatrix} \begin{Bmatrix} \mathbf{p}_c \\ \mathbf{p}_n \\ \mathbf{p}_e \end{Bmatrix} + \begin{bmatrix} \mathbf{M}_{cc} & \mathbf{M}_{cn} & \mathbf{M}_{ce} \\ \mathbf{M}_{nc} & \mathbf{I}_{nn} & \mathbf{0} \\ \mathbf{M}_{ec} & \mathbf{0} & \mathbf{I}_{ee} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{p}}_c \\ \ddot{\mathbf{p}}_n \\ \ddot{\mathbf{p}}_e \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix} . \tag{3.8}$$

The simpler structure of the of the system stiffness matrix makes the constraint-mode superset the best candidate for CMS with convergence. In the discussion henceforth, only system matrices obtained by using constraint-mode supersets will be considered.

3.2 Preliminaries to the Application of the Convergence Scheme

In the two-step CMS method presented here, the initial eigenvalue problem with the matrices given by equation (3.1) is first solved. Let Φ be the matrix of all eigenvectors and Λ be the diagonal matrix of all the eigenvalues of the initial system, whose stiffness and mass matrices are given in equations (3.5). To reform the equations to include the additional modes as by Engels [40], the transformation

$$\mathbf{p} = \Phi\, \mathbf{q}\,, \tag{3.9}$$

is introduced, so that the i^{th} eigenvector p_i can be written as

$$\mathbf{p}_i = \Phi\, \mathbf{q}_i\,, \tag{3.10}$$

where

$$\mathbf{q}_i = \; <0 \;\; 0 \ldots 1 \ldots\ldots 0 \;\; 0>^T \tag{3.11}$$

with 1 in the i^{th} position. When the 'e' extra component modes are introduced, the system degree-of-freedom vector becomes

$$\begin{Bmatrix} \mathbf{p}_c \\ \mathbf{p}_n \\ \mathbf{p}_e \end{Bmatrix} = \begin{bmatrix} \Phi & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{ee} \end{bmatrix} \begin{Bmatrix} \mathbf{q} \\ \mathbf{p}_e \end{Bmatrix}. \tag{3.12}$$

Substituting from equation (3.12) in equation (3.8) and pre-multiplying by $\begin{bmatrix} \Phi^T & 0 \\ 0 & I_{ee} \end{bmatrix}$ results in

$$\begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda_{ee} \end{bmatrix} \begin{Bmatrix} q \\ p_e \end{Bmatrix} + \begin{bmatrix} I & m_{ce} \\ m_{ce}^T & I_{ee} \end{bmatrix} \begin{Bmatrix} \ddot{q} \\ \ddot{p}_e \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}, \tag{3.13}$$

where

$$m_{ce} = \Phi^T \begin{bmatrix} M_{ce} \\ 0 \end{bmatrix}. \tag{3.14}$$

Notice that equation (3.13) is equivalent to equation (3.8). Solution of this eigenproblem for the lowest eigenvalues up to the desired cut-off frequency results in improved eigenvalues compared to those obtained by the initial solution. This would, however, require solution of the larger system matrices. Alternatively, if a method for the convergence of all the initial eigenvalues using the additional component modes can be developed, which does not handle the larger system matrices, the method would be more attractive.

So far, the only efforts directed towards converging initially calculated system modes have been made by Hasselman and Hart [39] and Engels [40]. Hasselman and Hart developed a procedure based on the minimization of the Rayleigh quotient of the structure. Convergence of the individual frequencies and mode shapes can be achieved by employing the conjugate gradient technique in conjunction with an appropriate scaling transformation, as in Hasselman and Hart [39]. However, this method requires that the set of extra component modes include all the remaining modes of all the components.

Engels' method is based on converging the initial system modes one at a time, by using a Newton-Raphson approach using all the remaining modes of all the components. Engels [40] also derives a condition under which the residual vector norm of the improved solution is better than the original solution. This condition is that the lowest frequency in the additional modes

should be at least twice the frequency to be converged. Stated differently, the first eigenvalue problem has to be solved with component normal modes up to a frequency of at least $2\omega_c$. Only one eigenvalue is converged at a time, and the converged system eigenparameters are not exact but close approximations. The calculated system eigenvalue may be greater or less than the actual eigenvalue and the set of additional modes has to include all the remaining normal modes of all the components. This is not practical in a CMS scheme intended for large dof systems.

The disadvantages of the above convergence methods form the motivation for the development of the present scheme. The objective is to use only limited additional modes (those that are available readily or can be calculated, given the limitations of cost) in the convergence scheme. Then, the converged eigenparameters are better approximations to the actual system eigenparameters than the initial values. Thus, an idea of the convergence of initial system eigenparameters can be obtained and more component modes can be introduced if necessary, to converge the unconverged system eigenvalues and eigenvectors.

3.3 The Method

Consider the eigenvalue problem corresponding to equation (3.13),

$$\left(\begin{bmatrix} \Lambda & \mathbf{0} \\ \mathbf{0} & \Lambda_{ee} \end{bmatrix} - \lambda \begin{bmatrix} \mathbf{I} & \mathbf{m}_{ce} \\ \mathbf{m}_{ce}^T & \mathbf{I}_{ee} \end{bmatrix} \right) \begin{Bmatrix} \mathbf{q} \\ \mathbf{p}_e \end{Bmatrix} = \mathbf{0}, \tag{3.15}$$

where λ is the system eigenvalue after the additional modes are introduced. Solving for $\mathbf{p}_e$ from the lower partition, gives

$$\mathbf{p}_e = [\Lambda_{ee} - \lambda \mathbf{I}]^{-1} \lambda \mathbf{m}_{ce}^T \mathbf{q}. \tag{3.16}$$

From the upper partition,

$$\Lambda\, \mathbf{q} - \lambda\, \mathbf{I}\, \mathbf{q} - \lambda\, \mathbf{m}_{ce}\, \mathbf{p}_e = \mathbf{0}\,. \tag{3.17}$$

Substituting for $\mathbf{p}_e$ from equation (3.16) in equation (3.17), results in

$$[\,\Lambda - \lambda\, \mathbf{I} - \lambda^2\, \mathbf{m}_{ce}\, [\,\Lambda_{ee} - \lambda\, \mathbf{I}]^{-1}\, \mathbf{m}_{ce}{}^{T}\,]\, \mathbf{q} = \mathbf{0}\,, \tag{3.18}$$

or

$$\mathbf{N}(\lambda)\, \mathbf{q} = \mathbf{0}\,, \tag{3.19}$$

where

$$\mathbf{N}(\lambda) \equiv \mathbf{N} = [\,\Lambda - \lambda\, \mathbf{I} - \lambda^2\, \mathbf{m}_{ce}\, [\,\Lambda_{ee} - \lambda\, \mathbf{I}]^{-1}\, \mathbf{m}_{ce}{}^{T}\,]\,. \tag{3.20}$$

Notice that the element in the i^{th} row and j^{th} column of the matrix **N** is given by

$$\mathbf{N}(i,j) = \Lambda(i,i)\, \delta_{ij} - \lambda\, \delta_{ij} - \lambda^2 \sum_{r=1}^{e} \frac{\mathbf{m}_{ce}(i,r)\mathbf{m}_{ce}(j,r)}{\Lambda_{ee}(r,r) - \lambda}\,, \tag{3.21}$$

where $\delta_{ij} = \begin{cases} 1 \text{ for } i = j \\ 0 \text{ for } i \neq j \end{cases}$,

and therefore, **N** is symmetric. The third term for **N**(i,j) in equation (3.21) can be expanded as an infinite series to give

$$\lambda^2 \sum_{r=1}^{e} \frac{\mathbf{m}_{ce}(i,r)\mathbf{m}_{ce}(j,r)}{\Lambda_{ee}(r,r) - \lambda}$$

$$= \lambda^2 \sum_{r=1}^{e} \frac{\mathbf{m}_{ce}(i,r)\mathbf{m}_{ce}(j,r)}{\Lambda_{ee}(r,r)} + \lambda^3 \sum_{r=1}^{e} \frac{\mathbf{m}_{ce}(i,r)\mathbf{m}_{ce}(j,r)}{\Lambda_{ee}(r,r)^2} + \lambda^4 \sum_{r=1}^{e} \frac{\mathbf{m}_{ce}(i,r)\mathbf{m}_{ce}(j,r)}{\Lambda_{ee}(r,r)^3} + \ldots\,. \tag{3.22}$$

This suggests that **N** can be expanded in a power series as

$$\mathbf{N}(\lambda) = \mathbf{N}_0 + \lambda\,\mathbf{N}_1 + \lambda^2\,\mathbf{N}_2 + \lambda^3\,\mathbf{N}_3 \ldots\ldots \quad . \tag{3.23}$$

where

$$\mathbf{N}_0(i,j) = \Lambda(i,i)\,\delta_{ij}\,, \tag{3.24}$$

$$\mathbf{N}_1(i,j) = -\,\delta_{ij}\,, \tag{3.25}$$

and

$$\mathbf{N}_s(i,j) = -\sum_{r=1}^{e} \frac{\mathbf{m}_{ce}(i,r)\mathbf{m}_{ce}(j,r)}{\Lambda_{ee}(r,r)^{s-1}}\,,\ s = 2,3 \ldots\ldots \quad . \tag{3.26}$$

Hence the series is convergent if $\lambda < \min\{\mathrm{diag}(\Lambda_{ee})\}$.

Several methods exist in the literature for solving the non-linear eigenvalue problem described by equation (3.19). However, these methods either involve expensive factorizations, forward and backward substitutions or a truncation of the infinite series up to a small finite power of λ, along with the formation of block-companion matrices.

Some of the algorithms in the literature use the vanishing determinant property of the λ-matrix at the exact eigenvalues. Kublanovskaya [41] uses a unitary transformation to obtain a scalar equation in terms of the eigenvalue without evaluating the determinant explicitly, followed by the application of Newton - Raphson iteration for root finding. Leung's algorithm [42] is also based on a similar approach, with the eigenvectors obtained by inverse iteration. An algorithm by Thurston [43] is based on the linearization of the non-linear eigenvalue equation about an approximate root and solving many linear eigenvalue problems till convergence is obtained. Each eigenvalue is converged separately. Ruhe [44] studied algorithms which were extensions

of those used for linear eigenvalue problems, such as inverse iteration and the QR algorithm. He also considered an algorithm that reduces the non-linear eigenvalue problem into a series of linear problems. Rajakumar [45] uses a Lanczos two-sided recursion algorithm with bi-orthogonal transformations for quadratic eigenvalue problems.

Moler and Stewart [46], and Fricker [47] have developed algorithms based on block-companion matrices. These matrices are obtained using the coefficient matrices of the non-linear matrix eigenvalue problem and essentially reduce the non-linear problem to a linear problem of larger order. Moler and Stewart [46] solved the resulting linear problem using a QZ algorithm. Fricker [47] defined master and slave coordinates for a reduction transformation based on a cut-off frequency. The resulting problem is cast in a block-companion matrix form for solution using a linear algorithm. Jain, Singhal and Huseyin [48] developed an algorithm for finding only the eigenvalues through a triangularization scheme for calculating the determinant. Yang [49] also used a similar factorization for evaluating the determinant. Both these methods were able to calculate the eigenvalues only. Osborne and Michaelson [50] developed a method of iteration similar to the method to be developed here, but their method involves coupling between the eigenvalue and eigenvector. This requires factorizations and forward and backward substitutions at each stage of the iteration.

Methods involving the formation of block-companion matrices are not useful in our case as we need to find exact solutions to the infinite-series eigenvalue problem. Methods involving factorization and forward and backward substitutions are expensive to implement and inherit all the numerical difficulties associated with the factorization of matrices, such as ill-conditioning and near-zero pivots.

There is a need for a solution scheme with iteration expressions which do not involve coupling between the eigenvalue and eigenvector, since such a coupling would require matrix factorizations during iteration. Here, the objective is to develop a simple method which takes advantage of the special structure of the matrices $\mathbf{N}_0$ and $\mathbf{N}_1$ in $\mathbf{N}(\lambda)$. Also, if the eigenvalue-eigenvector dependence is decoupled during iteration, the algorithm would not require matrix factorization during each iteration. The following details the development of the method.

Appendix A describes how a general positive semi-definite non-linear eigenvalue problem of the form (3.19), where $\mathbf{N}(\lambda)$ is symmetric and $\mathbf{N}_0$ and $\mathbf{N}_1$ are not necessarily diagonal, can be transformed to a form for which the present method is applicable.

3.3.1 Convergence Scheme

The convergence scheme is developed using a perturbation approach to obtain a basic equation from which iteration expressions are to be derived. Eigenvalue-eigenvector dependence is decoupled in the derivation of the iteration expressions, and this results in two iteration expressions for convergence, one each for the eigenvalue and eigenvector.

Assume that approximate eigenparameters λ_0 and $\mathbf{q}_0$ are known. In general, λ_0 and $\mathbf{q}_0$ will not satisfy equation (3.19), so that corrections λ_c to λ_0 and $\mathbf{q}_c$ to $\mathbf{q}_0$ such that λ and $\mathbf{q}$ will satisfy the equation are to be determined. Let

$$\lambda = \lambda_0 + \varepsilon \lambda_c ;$$
$$\mathbf{q} = \mathbf{q}_0 + \varepsilon \mathbf{q}_c , \qquad (3.27)$$

where ε is an ordering parameter. Using ε for ordering indicates the relative magnitude of each term. Ordering equation (3.23) for N gives

$$\mathbf{N}(\lambda) = \mathbf{N}_0 + \lambda\mathbf{N}_1 + \varepsilon \lambda^2\mathbf{N}_2 + \varepsilon^2 \lambda^3\mathbf{N}_3 \ldots . \qquad (3.28)$$

Substituting for λ and $\mathbf{q}$ from equation (3.27) in equation (3.19), and using the expansion for $\mathbf{N}(\lambda)$ given by equation (3.28) results in

$$[\mathbf{N}_0 + (\lambda_0 + \varepsilon \lambda_c)\mathbf{N}_1 + \varepsilon(\lambda_0 + \varepsilon \lambda_c)^2\mathbf{N}_2 + \varepsilon^2(\lambda_0 + \varepsilon\lambda_c)^3\mathbf{N}_3 \ldots.][\mathbf{q}_0 + \varepsilon \mathbf{q}_c] = \mathbf{0} , \qquad (3.29)$$

i.e.,

$$\mathbf{N}(\lambda_0)\,[\mathbf{q}_0 + \varepsilon\,\mathbf{q}_c] + \varepsilon\,\lambda_c\,\mathbf{N}'(\lambda_0)[\,\mathbf{q}_0 + \varepsilon\,\mathbf{q}_c] + (\frac{1}{2!})\,\varepsilon^2\lambda_c^2\mathbf{N}''(\lambda_0)\,[\mathbf{q}_0 + \varepsilon\,\mathbf{q}_c] + \ldots\ldots = \mathbf{0}\,. \qquad (3.30)$$

If only the terms with the first power of ε in equation (3.30) are retained and the higher order terms are neglected, equation (3.30) becomes

$$\mathbf{N}(\lambda_0)\,\mathbf{q}_0 + \varepsilon\,\mathbf{N}(\lambda_0)\mathbf{q}_c + \varepsilon\,\lambda_c\,\mathbf{N}'(\lambda_0)\,\mathbf{q}_0 = 0\,. \qquad (3.31)$$

Here $\mathbf{N}'(\lambda)$ is the matrix consisting of the derivatives of the elements of $\mathbf{N}$ with respect to λ and is given by

$$\mathbf{N}'(\lambda) = \mathbf{N}_1 + 2\,\varepsilon\,\lambda\,\mathbf{N}_2 + 3\,\varepsilon^2\lambda^2\mathbf{N}_3 \ldots\ldots\,. \qquad (3.32)$$

Notice that equation (3.31) represents a system of equations with one more unknown than the number of equations. This is the case with the solution of any eigenvalue problem because the eigenvector can be multiplied by an arbitrary scale factor. The problem is resolved by choosing to normalize the eigenvector $\mathbf{q}$ in a specific manner described later. In order to decouple the dependence between the eigenvalue and eigenvector during convergence, two different iteration expressions are derived. One is for the eigenvalue, assuming that the eigenvector is known, and the other is for the eigenvector, assuming that the eigenvalue is known.

3.3.2 Solution for the Eigenvalue Assuming that the Eigenvector is Known

If, in equation (3.31), the eigenvector $\mathbf{q}$ is assumed to be known, i.e., $\mathbf{q}_0 = \mathbf{q}$ and $\mathbf{q}_c = \mathbf{0}$, then

$$\mathbf{N}(\lambda_0)\,\mathbf{q} + \varepsilon\,\lambda_c\,\mathbf{N}'(\lambda_0)\,\mathbf{q} = \mathbf{0}\,. \qquad (3.33)$$

Pre-multiplying by $\mathbf{q}^T$ and solving for λ_c gives

$$\lambda_c = -\frac{\mathbf{q}^T\mathbf{N}(\lambda_0)\mathbf{q}}{\varepsilon\,\mathbf{q}^T\mathbf{N}'(\lambda_0)\mathbf{q}} = -\frac{\mathbf{q}^T\mathbf{N}(\lambda_0)\mathbf{q}}{\mathbf{q}^T\mathbf{N}'(\lambda_0)\mathbf{q}}\,, \tag{3.34}$$

where the ordering parameter ε has been removed. Therefore, from equation (3.27),

$$\lambda = \lambda_0 - \frac{\mathbf{q}^T\mathbf{N}(\lambda_0)\mathbf{q}}{\mathbf{q}^T\mathbf{N}'(\lambda_0)\mathbf{q}}\,, \tag{3.35}$$

is a better approximation to the actual eigenvalue corresponding to equation (3.19). Hence, if $\mathbf{q}$ is known, λ can be iterated for using the above equation to find progressively better eigenvalues. In an iterative scheme

$$\lambda^{(k+1)} = \lambda^{(k)} - \frac{\mathbf{q}^T\mathbf{N}(\lambda^{(k)})\mathbf{q}}{\mathbf{q}^T\mathbf{N}'(\lambda^{(k)})\mathbf{q}}\,, \tag{3.36}$$

where the superscript for λ denotes the iteration number. Notice however, that in equation (3.36), $\mathbf{N}'(\lambda^{(k)})$ has to be computed in every iteration. The number of computations can be reduced by substituting for $\mathbf{N}'(\lambda)$ from equation (3.32) in equation (3.33), so that

$$\mathbf{N}(\lambda_0)\,\mathbf{q} + \varepsilon\,\lambda_c(\mathbf{N}_1 + 2\varepsilon\lambda_0\,\mathbf{N}_2 + 3\varepsilon^2\lambda_0^2\mathbf{N}_3 \ldots..)\,\mathbf{q} = \mathbf{0}\,, \tag{3.37}$$

and assuming in the parenthetical expression in the second term that the contribution of terms with powers of ε is small compared to $\mathbf{N}_1$, the equation

$$\mathbf{N}(\lambda_0)\,\mathbf{q} + \varepsilon\,\lambda_c\mathbf{N}_1\mathbf{q} \cong \mathbf{0} \tag{3.38}$$

is obtained. Pre-multiplying equation (3.38) by $\mathbf{q}^T$ and solving for λ_c,

$$\lambda_c = -\{\mathbf{q}^T\mathbf{N}(\lambda_0)\,\mathbf{q}\} / \{\mathbf{q}^T\mathbf{N}_1\,\mathbf{q}\}\,, \tag{3.39}$$

where again ε has been removed. Now, if $\mathbf{q}$ is always normalized such that $\mathbf{q}^T\mathbf{q} = 1$, then

$$\lambda_c = \mathbf{q}^T \mathbf{N}(\lambda_0)\, \mathbf{q}\,, \tag{3.40}$$

since $\mathbf{N}_1 = -\mathbf{I}$, the identity matrix, and $\mathbf{q}^T\mathbf{N}_1\mathbf{q} = -\mathbf{q}^T\mathbf{q} = -1$. Hence, the iteration scheme for λ can be modified as

$$\lambda^{(k+1)} = \lambda^{(k)} + \mathbf{q}^T\mathbf{N}(\lambda^{(k)})\, \mathbf{q}\ , \tag{3.41}$$

and this is more attractive than equation (3.36) since it avoids the computation of $\mathbf{N}'(\lambda^{(k)})$.

3.3.3 Solution for the Eigenvector Assuming that the Eigenvalue is Known

If it is assumed that the eigenvalue λ is known, i.e., $\lambda = \lambda_0$ and $\lambda_c = 0$, then equation (3.31) reduces to

$$\mathbf{N}(\lambda)\, \mathbf{q}_0 + \varepsilon\, \mathbf{N}(\lambda)\, \mathbf{q}_c = \mathbf{0}\,. \tag{3.42}$$

Solving for $\mathbf{q}_c$,

$$\mathbf{q}_c = -\{\varepsilon\, \mathbf{N}(\lambda)\}^{-1}\, \mathbf{N}(\lambda)\, \mathbf{q}_0\,. \tag{3.43}$$

Notice that $\mathbf{N}(\lambda)$ is singular (because λ is the exact eigenvalue) and hence $\mathbf{q}_c$ cannot be determined. Even if λ were not known, this equation would give $\mathbf{q}_c = -\mathbf{q}_0$ and this is undesirable. Recognize that from the expansion for $\mathbf{N}(\lambda)$, $\varepsilon\mathbf{N}(\lambda)$ can be written as

$$\varepsilon\, \mathbf{N}(\lambda) = (\varepsilon\, \mathbf{N}_0 + \varepsilon\, \lambda\, \mathbf{N}_1 + \varepsilon^2 \lambda^2\, \mathbf{N}_2 + \varepsilon^3\lambda^3\, \mathbf{N}_3 \ldots .)\,, \tag{3.44}$$

and retaining only terms with the first power of ε for the term $\{\varepsilon\, \mathbf{N}(\lambda)\}$ in equation (3.43) gives

$$\mathbf{q}_c \cong -(\varepsilon \mathbf{N}_0 + \varepsilon \lambda \mathbf{N}_1)^{-1} \mathbf{N}(\lambda)\mathbf{q}_0 \,, \tag{3.45}$$

or with the ordering parameter ε removed,

$$\mathbf{q}_c = -(\mathbf{N}_0 + \lambda \mathbf{N}_1)^{-1} \mathbf{N}(\lambda)\, \mathbf{q}_0 \,. \tag{3.46}$$

Therefore, from equation (3.27)

$$\mathbf{q} = \mathbf{q}_0 - (\mathbf{N}_0 + \lambda \mathbf{N}_1)^{-1} \mathbf{N}(\lambda)\, \mathbf{q}_0 \tag{3.47}$$

is a better approximation to the eigenvector. When used iteratively

$$\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} - (\mathbf{N}_0 + \lambda \mathbf{N}_1)^{-1} \mathbf{N}(\lambda)\mathbf{q}^{(k)} \,, \tag{3.48}$$

where the superscript on $\mathbf{q}$ denotes the iteration number. Since the matrix $(\mathbf{N}_0 + \lambda \mathbf{N}_1)$ is diagonal, its inverse can be easily computed.

3.3.4 Solution for the Eigenvalue and the Eigenvector Starting from Approximate Initial Values

If both the eigenvalue and eigenvector are not known, the above iteration schemes can still be used to simultaneously converge both λ and $\mathbf{q}$. Then, an iteration becomes

$$\begin{aligned} &\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} - (\mathbf{N}_0 + \lambda^{(k)} \mathbf{N}_1)^{-1} \mathbf{N}(\lambda^{(k)})\mathbf{q}^{(k)} \,; \\ &\lambda^{(k+1)} = \lambda^{(k)} + \mathbf{q}^{(k)T} \mathbf{N}(\lambda^{(k)})\mathbf{q}^{(k)} \,. \end{aligned} \tag{3.49}$$

The fact that the above equations will lead to convergence of both the eigenvalue and eigenvector can be reasoned as follows.

In the solution scheme, values for λ and $\mathbf{q}$ that would satisfy equation (3.19) are sought. If the entire vector $\mathbf{q}$ is considered as a single entity, the solution of equation (3.19) can be interpreted as the search for zeroes of a function of two variables. For explanatory purposes, consider a function f of two variables x and y, whose zeroes are to be found, i.e., x and y are sought such that

$$f(x,y) = 0 \,. \tag{3.50}$$

If an approximate solution x_0 and y_0 is known, then

$$f(x,y) = f(x_0,y_0) + (x - x_0)\, f_x(x_0,y_0) + (y - y_0)\, f_y(x_0,y_0) = 0, \tag{3.51}$$

where $f_x = \partial f / \partial x$; $f_y = \partial f / \partial y$ and the higher order terms have been neglected. A better value for x is obtained by temporarily disregarding the variation of f with y, i.e., by searching for a solution for equation (3.50) along the plane $y = y_0$, which solves the equation

$$f(x_0,y_0) = (x - x_0)\, f_x(x_0,y_0) \,. \tag{3.52}$$

for x. Similarly, by temporarily disregarding the variation of f with x, and looking for a solution along the plane $x = x_0$, a better value for y is obtained.

Equation (3.19) is comparable to equation (3.50) and the two individual solution schemes for the eigenvalue and eigenvector assuming that the other is known, are in reality, a search for a better solution along the hyper-planes $\mathbf{q} = \mathbf{q}_0$ and $\lambda = \lambda_0$ respectively. In other words, separate iterations are performed in the $\mathbf{q}$-subspace and λ-subspace, respectively. Strictly speaking, the proposed schemes are slightly modified versions of this search since the expressions to be used in an iteration have been further simplified for better computational efficiency.

3.3.5 Initial Values

Since the first eigenproblem has already been solved for its eigenvalues and eigenvectors, it appears that the convergence scheme for the i^{th} eigenvalue and eigenvector can be started with the initial values

$$\lambda_i^{(0)} = \Lambda(i,i) \ ,$$
$$\mathbf{q}_i^{(0)} = <0\ 0\ \ldots 0\ 1\ 0 \ldots 0\ 0>^T \qquad (3.53)$$

where $\mathbf{q}_i^{(0)}$ has 1 in the i^{th} position. However, $(\mathbf{N}_0 + \lambda_i^{(0)} \mathbf{N}_1)$ is singular, so convergence must be started with the eigenvalue equation to determine $\lambda_i^{(1)}$. Therefore, to start convergence, the eigenvalue has to be iterated for first. Thereafter, equations (3.49) can be used in turn to converge both the eigenvalue and the eigenvector. Details of implementation of this scheme are presented in the following chapter.

4.0 Implementation

The method of CMS developed in the previous chapter was implemented in a FORTRAN code to run on IBM RISC-6000 workstations. Details of this implementation are presented in this chapter.

4.1. General Outline for Program Development

The following is a list of the essential sequence of steps involved in the CMS of a structural system using the method developed.

1. Divide the structure into components and identify interface dofs and their nature (whether dependent or independent).
2. Analyze components to determine rigid-body modes, constraint modes and normal modes (mass normalized) and write to component data files. In doing so, identify normal modes to be used for initial synthesis and for convergence.
3. Determine and construct partitions of component generalized stiffness and mass matrices, i.e., $\mathbf{K}_{cc}$, Λ_{nn}, Λ_{ee}, $\mathbf{M}_{cc}$, $\mathbf{M}_{cn}$ and $\mathbf{M}_{ce}$ according to equation (3.8) on page 26 and write to component data files.
4.. Perform CMS using component data.
   (i) Make component connections between dependent and independent dofs.
   (ii) Read component modes of each component.
   (ii) For each component, read and partition component generalized stiffness and mass matrices.
   (iii) Using component connectivity information, form matrices $\mathbf{C}_{dd}$ and $\mathbf{C}_{d\ell}$ of

equation (2.38) on page 17.

(iv) Form system matrices using partitions of component generalized matrices and the matrices $\mathbf{C}_{dd}$ and $\mathbf{C}_{d\ell}$.

(v) Solve initial eigenvalue problem.

5. Converge synthesized modes.

(i) Use eigenvectors from initial eigenvalue problem to form matrix $\mathbf{m}_{ce}$ according to equation (3.14) on page 28.

(ii) For each mode,

(a) obtain initial values for convergence,

(b) apply convergence scheme and

(c) back-calculate eigenvector in component physical dof coordinates and write to output.

4.2 Component Identification

The first step involves the user's judgment in dividing the structure to be analyzed into two or more components and developing finite element models of them. Components are natural in many systems. Finite element model development involves conventional mesh-generation procedures and hence will not be discussed here. However, an observation has to be made regarding nodes on interfaces. Interface nodes on a component should be coincident with interface nodes of the connected component, so that constraint equations such as equation (2.33) on page 16 are applicable. Once the models are constructed, the model data (nodal coordinates, connectivity, element physical and material properties, rigid-body and excess boundary dofs, dependent and independent dofs) are to be written to a data file to be used in steps 2 and 3.

4.3 Generation of Component Modes and Partitions of Component Generalized Matrices

Steps 2 and 3 involve the use of a finite element processor which calculates element stiffness and mass matrices, assembles element matrices into global structure matrices and imposing restraints. Details of implementation of this step follow usual procedures in a general finite element analysis and will not be discussed here as this is not the focus of the study. However, some salient features of the processor have to be pointed out. The processor uses a subspace-iteration solver based on the subspace vector iteration method developed by Bathe [2] to calculate the fixed-interface normal modes. Ramani [51] provides a listing of subroutine SUBSPACE and those called by SUBSPACE for this purpose. Rigid-body modes and redundant constraint modes are also determined by the processor according to equations (2.7) and (2.8) respectively on page 10. Subroutines used for this purpose are available in [51]. Once component modes are determined, the partitions of component generalized stiffness and mass matrices are constructed using matrix multiplication operations. These are written to component data files.

4.4 CMS - Initial Solution

Component Mode Synthesis is performed in step 3 using component data available from previous steps. This is accomplished using the program CMSCON [51]. The program is designed to use component mode data for each component to synthesize and converge system modes. The program performs all the necessary operations listed in steps 3 and 4. Execution is automatic and the user's intervention is required initially only to specify how the components are connected.

4.4.1 Input Processing

Execution of program CMSCON begins by reading component input data. The following constants are required for program execution -

(i) the number of components, NN,

(ii) the total number of dependent coordinates, `ID`,

(iii) the total number of independent coordinates, `IL`,

(iv) the total number of normal modes, `IN`, to be used in initial synthesis and

(v) the total number of normal modes, `IE`, to be used for convergence.

Next, for each component, the following data are required -

(i) the number of dependent coordinates,

(ii) the number of independent coordinates,

(iii) the number of component normal modes to be used in initial synthesis, and

(iv) the number of additional normal modes to be used for convergence.

These data are supplied in a data file to be read at the beginning of program execution, using which arrays `IID`, `IIL`, `IIN`, and `IIE` of dimension are constructed. Consider array `IID` for instance. Each row corresponds to a component. If we consider the i^{th} row, the element in the first column stores the number of dependent coordinates of that component and the element in the second column stores the cumulative number of dependent coordinates up to the i^{th} component. Similarly, the other arrays store data as indicated below.

`IIL` - number of independent coordinates

`IIN` - number of normal mode coordinates used in the first synthesis

`IIE` - number of additional normal mode coordinates used for convergence

Similar to the above arrays, the following arrays of dimension `NN x 2` are constructed.

`IIPSUM` - number of data elements in constraint modes (equal to the number of physical dofs of the component times the number of constraint modes)

`IINSUM` - number of data elements in normal modes

`IIESUM` - number of data elements in additional modes

Also, for each component, the number of component physical dofs are read into array `IINDOF`.

Also required, for each component are the names of component data files storing

(i) the constraint modes (redundant constraint and rigid-body modes),

(ii) the normal modes, and

(iii) the partitions of component generalized matrices, i.e., $\mathbf{K}_{cc}$, Λ_{nn}, Λ_{ee}, $\mathbf{M}_{cc}$, $\mathbf{M}_{cn}$ and

$\mathbf{M}_{ce}$.

The system eigenvectors determined by CMS can be back-calculated in terms of the physical dofs of components. The eigenvector can be partitioned into sub-vectors pertaining to each component. Associated with each component is an output data file, which, on program exit will store the partitions belonging to that component of the system eigenvector. All the above information is supplied in a file read at the beginning of program execution.

The next step is to read the component modes. Constraint modes of all components are read into a single-dimensional array, `PHIC`, and all component normal modes (including those to be used for convergence) are read into a single dimensional array, `PHIN`.

For each component 'α', we have files 'depα' and 'indepα' storing the dependent and independent dofs respectively. In the present implementation, all components have six dofs per node and component connections require all dofs at a node of one component to be related to corresponding dofs at the corresponding node of the connected component. In the files 'depα' and 'indepα', the dependent and independent coordinates are stored in two columns. The first column stores the component node number and the second stores the equation number pertaining to one of the dofs at that node. Hence, for each nodal dof that is dependent or independent, there are six entries in the file 'depα' or 'indepα' respectively with the six equation numbers corresponding to the dofs at that node.

The equation numbers of all the connection coordinates of all components are read into a single dimensional array, `IRST`, in component order.

Following the input of component modes, the program requires user input in specifying component connections. This information is used to construct array `ICOORD` of dimension `ID x 4`. Each row of this array corresponds to a unique dependent coordinate. Let node 'i' of component 'α' be connected to node 'j' of component 'β', and let the dofs at node 'i' be dependent. Then, the entries in the corresponding rows of array `ICOORD` are as follows

α i1 β j1

α	i2	β	j2
	.		
	.		
	.		
α	i6	β	j6

where 'i1' through 'i6' and 'j1' through 'j6' are equation numbers that correspond to nodes 'i' and 'j' of components 'α' and 'β' respectively. Figure 4.1 lists a section of the FORTRAN code in CMSCON that constructs array ICOORD. The array is constructed as follows.

1. Loop through each component IALPHA.
2. For each component, loop through each dependent node INODE.
3. Prompt user to enter component number IBETA and node number JNODEDUM to which the dependent node is connected.
4. Search for this node in file 'indepβ'.
5. Once found, for the next six rows of array ICOORD, assign
   (i) component number IALPHA to the first column,
   (ii) equation numbers i1 through i6 to the second column,
   (iii) component number IBETA to the third column, and
   (iv) equation numbers j1 through j6 to the fourth column.

Once array ICOORD has been constructed, the matrix partitions C_{dd} and $C_{d\ell}$ of the transformation matrix used for coordinate reduction can be constructed. These partitions are stored in the two-dimensional arrays CDD and CDL. Recall the transformation matrix S from equation (2.40) on page 18. For our purpose, this matrix can be rewritten as

$$S = \begin{bmatrix} -C_{dd}^{-1}C_{d\ell} & 0 \\ I_{\ell\ell} & 0 \\ 0 & I_{nn} \end{bmatrix}, \tag{4.1}$$

where now, all the 'ℓ' independent coordinates have been split into independent interface coordinates (the same subscript 'ℓ' has been reused) and independent normal mode coordinates

```
C
C  Loop through components
C
        DO 800 IALPHA = 1,NN
C
C  For each component, open file containing dependent coordinates
C
        OPEN(UNIT=1,FILE=IIDEP(IALPHA))
        I6 = IID(IALPHA,1)/6
          DO 805 J = 1,I6
C
C  Prompt user to input connected component and node number
C
          WRITE(6,4)IALPHA,IDUM
 4        FORMAT(2x,'Component ',I3,', Node ',I4,', Connected to :')
          WRITE(6,44)
 44       FORMAT(2X,'Enter "COMPONENT #, NODE # ":')
          READ(*,*)IBETA,JNODEDUM
C
C  Search for this node in the component independent
C    coordinate file IINDEP(IBETA)
C
          OPEN(UNIT=2,FILE=IINDEP(IBETA))
            DO 815 K = 1,IIL(IBETA,1)
            READ(2,*)JNODE,IDUM
C
C  When a match is found, assign component numbers and
C    equation numbers to elements of array ICOORD
C
              IF(JNODE.EQ.JNODEDUM)THEN
                ICOORD(IID(IALPHA,2)-IID(IALPHA,1)+(J-1)*6+1,3)=IBETA
                ICOORD(IID(IALPHA,2)-IID(IALPHA,1)+(J-1)*6+1,4)=IDUM
                  DO 820 K = 2,6
                  ICOORD(IID(IALPHA,2)-IID(IALPHA,1)+(J-1)*6+K1,3)=IBETA
820               READ(2,*)IDUM,ICOORD(IID(IALPHA,2)-
     .            IID(IALPHA,1)+(J-1)*6+K,4)
                CLOSE(2)
                  DO 810 K = 1,6
                  ICOORD(IID(IALPHA,2)-IID(IALPHA,1)+(J-1)*6+K,1)=IALPHA
810               READ(1,*)IDUM,ICOORD(IID(IALPHA,2)-IID(IALPHA,1)+
     .            (J-1)*6+K,2)
                GOTO 805
              ENDIF
815         CONTINUE
805       CONTINUE
        CLOSE(1)
800     CONTINUE
```

Figure 4.1. Code Section for Making Component Connections

(identified by subscript 'n') and it has been observed that the partition $\mathbf{C}_{dn} = \mathbf{0}$. It is the new partition $\mathbf{C}_{d\ell}$ that is to be constructed. The section of code that is used for constructing these partitions is shown in Figure 4.2. After initializing the matrices, we loop through each component IALPHA. For each component, an index variable IMARK is used to loop through each interface coordinate IROW1. If the coordinate is independent, we return back to the start of the loop; however, if the coordinate is dependent, a new row for each matrix partition $\mathbf{C}_{dd}$ and $\mathbf{C}_{d\ell}$ has to be constructed. Once dependency of a coordinate is determined, the component number IBETA and dof number for the connected independent coordinate are determined from array ICOORD and the column numbers of the arrays CDD and CDL are indexed to one. There are two different constraint modes corresponding to each of these dof coordinates IROW1 and IROW2 respectively and these can be located in array PHIC. Next, we consider the constraint mode corresponding to dof IROW1 and execute another loop through each interface dof IROW of component IALPHA. If dependent, the corresponding element of the constraint mode vector is located in array PHIC and assigned to the indexed row and column of array CDD and the CDD column index marker IDM is incremented. If independent, the corresponding element of the constraint mode vector is assigned to the indexed row and column of array CDL and the CDL column index marker ILM is incremented. Next, we execute a similar loop through each interface coordinate of component IBETA with the constraint mode vector corresponding to dof IROW2. This pass uses the column index markers IDDM and ILLM respectively for the dependent and independent coordinates.

Having constructed the partitions of the transformation matrices $\mathbf{C}_{dd}$ and $\mathbf{C}_{d\ell}$, partitions of the component generalized stiffness and mass matrices are to be read. For this, equation (2.28) on page 15 is rewritten for the partitions of the component generalized matrices for the case of constraint mode supersets. In doing so, the inertia-relief modes have been disregarded, the 'k' kept normal mode coordinates have been split into 'n' initial mode coordinates and 'e' additional mode coordinates and the 'c' constraint mode coordinates have been split into 'd' dependent coordinates and 'ℓ' independent coordinates. With all subscripts pertaining to a particular component, the component generalized matrices take the form

```
C
C  Construct CDD and CDL matrices.  Initialize
C
         DO 18 I = 1,ID
           DO 19 J = 1,ID
 19        CDD(I,J) = 0.D0
           DO 18 J = 1,IL
 18      CDL(I,J) = 0.D0
C
C  Loop through components
C
         DO 20 IALPHA = 1,NN
C
C  Initialize index IMARK for counting equation numbers of
C    interface coordinates
C
         IMARK = 1
C
C  Execute the following as many times as the number
C    of dependent coordinates in that component
C
           DO 20 J = 1,IID(IALPHA,1)
C
C  Extract (IMARK)th equation number of interface coordinate
C
 33        IROW1 = IRST(IID(IALPHA,2)+IIL(IALPHA,2)-IID(IALPHA,1)-
     .     IIL(IALPHA,1)+IMARK)
C
C  Search for this dof number in the second column of array ICOORD
C
             DO 31 J1 = 1,IID(IALPHA,1)
C
C  If found, i.e., if dependent, extract component number IBETA
C    and dof number IROW2 for the connected component from the
C    third and fourth columns of ICOORD.  Otherwise.......(1)
C
               IF(ICOORD(IID(IALPHA,2)-IID(IALPHA,1)+J1,2).EQ.IROW1)THEN
               IBETA = ICOORD(IID(IALPHA,2)-IID(IALPHA,1)+J1,3)
               IROW2 = ICOORD(IID(IALPHA,2)-IID(IALPHA,1)+J1,4)
C
C  Set dependent and independent coordinate column
C    pointers for first component
C
               IDM = IID(IALPHA,2)-IID(IALPHA,1)+1
               ILM = IIL(IALPHA,2)-IIL(IALPHA,1)+1
C
C  ...... Continued......
C
```

Figure 4.2. Code Section for Constructing Matrix Partitions C_{dd} and $C_{d\ell}$

```
C
C  Set dependent and independent coordinate column pointers
C    for second component IBETA and increment marker IMARK to
C    point at the next interface coordinate
C
                IDDM = IID(IBETA,2)-IID(IBETA,1)+1
                ILLM = IIL(IBETA,2)-IIL(IBETA,1)+1
                IMARK = IMARK + 1
                GOTO 32
                ENDIF
 31           CONTINUE
C
C  (1)..... If independent, simply increment marker IMARK and
C     consider next interface coordinate
C
             IMARK = IMARK + 1
             GOTO 33
C
C  Loop through each component interface coordinate
C
 32           DO 25 K = 1,IID(IALPHA,1)+IIL(IALPHA,1)
C
C  Extract dof number
C
                IROW = IRST(IID(IALPHA,2)+IIL(IALPHA,2)-IID(IALPHA,1)-
     .          IIL(IALPHA,1)+K)
C
C  Scan second column of array ICOORD to determine
C    if the coordinate is dependent
C
                  DO 35 I1 = 1,IID(IALPHA,1)
C
C ....... Continued .......
C
```

Figure 4.2. Code Section for Constructing Matrix Partitions C_{dd} and $C_{d\ell}$ (continued)

```
C
C  If dependent, assign appropriate element of array PHIC to
C    the marked column position of array CDD, increment column
C    pointer and exit loop.  Otherwise....(2)
C
                IF(ICOORD(IID(IALPHA,2)-IID(IALPHA,1)+I1,2).EQ.
     .          IROW) THEN
                CDD(IID(IALPHA,2)-IID(IALPHA,1)+J,IDM) =
     .          -PHIC(IIPSUM(IALPHA,2)- IIPSUM(IALPHA,1)+(IROW1-
     .          1)*(IID(IALPHA,1)+IIL(IALPHA,1))+K)
                IDM = IDM + 1
                GOTO 25
                ENDIF
 35           CONTINUE
C
C  (2)....Otherwise,the coordinate is independent.  Hence, assign
C    appropriate element of array PHIC to the marked column
C    position of array CDL and increment pointer.
C
             CDL(IID(IALPHA,2)-IID(IALPHA,1)+J,ILM) =
     .       PHIC(IIPSUM(IALPHA,2)- IIPSUM(IALPHA,1)+(IROW1-1)*
     .       (IID(IALPHA,1)+IIL(IALPHA,1))+K)
             ILM = ILM+1
 25          CONTINUE
C
C  Repeat DO 25...CONTINUE Loop for the connected
C    component dof coordinate
C
             DO 26 K = 1,IID(IBETA,1)+IIL(IBETA,1)
             IROW=IRST(IID(IBETA,2)+IIL(IBETA,2)-IID(IBETA,1)-
     .       IIL(IBETA,1)+K)
               DO 36 I1 = 1,IID(IBETA,1)
                 IF(ICOORD(IID(IBETA,2)-IID(IBETA,1)+I1,2).EQ.IROW)THEN
                 CDD(IID(IALPHA,2)-IID(IALPHA,1)+J,IDDM) =
     .           PHIC(IIPSUM(IBETA,2)-IIPSUM(IBETA,1)+(IROW2-1)*
     .           (IID(IBETA,1)+IIL(IBETA,1))+K)
                 IDDM = IDDM + 1
                 GOTO 26
                 ENDIF
 36            CONTINUE
             CDL(IID(IALPHA,2)-IID(IALPHA,1)+J,ILLM) =
     .       -PHIC(IIPSUM(IBETA,2)- IIPSUM(IBETA,1)+(IROW2-1)*
     .       (IID(IBETA,1)+IIL(IBETA,1))+K)
             ILLM = ILLM + 1
 26          CONTINUE
 20       CONTINUE
```

Figure 4.2. Code Section for Constructing Matrix Partitions C_{dd} and $C_{d\ell}$ (continued)

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{dd} & \mathbf{K}_{d\ell} & \mathbf{0} & \mathbf{0} \\ \mathbf{K}_{\ell d} & \mathbf{K}_{\ell\ell} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Lambda_{nn} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \Lambda_{ee} \end{bmatrix} ; \quad \mathbf{M} = \begin{bmatrix} \mathbf{M}_{dd} & \mathbf{M}_{d\ell} & \mathbf{M}_{dn} & \mathbf{M}_{de} \\ \mathbf{M}_{\ell d} & \mathbf{M}_{\ell\ell} & \mathbf{M}_{\ell n} & \mathbf{M}_{\ell e} \\ \mathbf{M}_{nd} & \mathbf{M}_{n\ell} & \mathbf{I}_{nn} & \mathbf{0} \\ \mathbf{M}_{ed} & \mathbf{M}_{e\ell} & \mathbf{0} & \mathbf{I}_{ee} \end{bmatrix}. \tag{4.2}$$

These matrix partitions are used to form partitions of the system matrices $\mathbf{K}_{dd}$, $\mathbf{K}_{d\ell}$, $\mathbf{K}_{\ell\ell}$, Λ_{nn}, Λ_{ee}, $\mathbf{M}_{dd}$, $\mathbf{M}_{d\ell}$, $\mathbf{M}_{\ell\ell}$, $\mathbf{M}_{dn}$, $\mathbf{M}_{de}$, $\mathbf{M}_{\ell n}$ and $\mathbf{M}_{\ell e}$, so that transformation equations can be used for coordinate reduction. The section of code that accomplishes this is shown in Figure 4.3. Looping through each component, diagonal elements of matrices Λ_{nn} and Λ_{ee} in arrays AKNN and AKEE respectively are read in from the appropriate component data file. Next, the component data files storing the matrix partitions $\mathbf{K}_{cc}$, $\mathbf{M}_{cc}$ and $\mathbf{M}_{cn}$ are opened. Executing an outer loop through each connection coordinate of the component in turn, if the coordinate is dependent, a row of each of the matrices $\mathbf{M}_{dn}$ and $\mathbf{M}_{de}$ is read from the appropriate files. Executing an inner loop through each interface coordinate of the component, if the coordinate is dependent, the input from the appropriate data files is read into matrices $\mathbf{K}_{dd}$ and $\mathbf{M}_{dd}$; otherwise the input is read into matrices $\mathbf{K}_{d\ell}$ and $\mathbf{M}_{d\ell}$ and both the inner and outer loops are exited. However, if the active coordinate of the outer loop is independent, input is read into matrices $\mathbf{M}_{\ell n}$, $\mathbf{M}_{\ell e}$, $\mathbf{M}_{\ell\ell}$ and $\mathbf{K}_{\ell\ell}$, taking care to skip those elements that do not belong to the partition. This is repeated for each interface coordinate and for each component. Upon completion, all the system matrix partitions required for transformation to construct system matrices are available.

4.4.2 Assembly of Initial System Matrices and Solution

Once the matrix partitions are read, the partitions $\mathbf{K}_{cc}$, Λ_{nn}, $\mathbf{M}_{cc}$ and $\mathbf{M}_{cn}$ of equation (3.5) on page 25 for the initial eigenvalue problem are to be constructed. These matrices are evaluated using the expressions

$$\mathbf{K}_{cc} = \mathbf{K}_{\ell\ell} - (\mathbf{C}_{dd}^{-1}\mathbf{C}_{d\ell})^{T}\mathbf{K}_{\ell d} - \mathbf{K}_{d\ell}^{T}(\mathbf{C}_{dd}^{-1}\mathbf{C}_{d\ell}) + (\mathbf{C}_{dd}^{-1}\mathbf{C}_{d\ell})^{T}\mathbf{K}_{dd}(\mathbf{C}_{dd}^{-1}\mathbf{C}_{d\ell}) ; \tag{4.3}$$

```
C
C  Index row pointers to first row
C
        MD = 1
        ML = 1
        MDL = 1
        MDN = 1
        MLN = 1
C
C  Loop through components
C
          DO 50 I = 1,NN
C
C  Open component data file containing diagonal elements
C    of matrices Lambda-nn and Lambda-ee
C
          OPEN(UNIT=2,FILE=IIKNN(I))
C
C  Read AKNN and AKEE matrices
C
            DO 51 J = IIN(I,2)-IIN(I,1)+1,IIN(I,2)
 51         READ(2,*)AKNN(J)
            DO 52 J = IIE(I,2)-IIE(I,1)+1,IIE(I,2)
 52         READ(2,*)AKEE(J)
          CLOSE(2)
C
C  Open component data files containing elements
C    of matrices Kcc, Mcc and Mcn
C
          OPEN(UNIT=1,FILE=IIMCC(I))
          OPEN(UNIT=2,FILE=IIKCC(I))
          OPEN(UNIT=3,FILE=IIMCN(I))
C
C  Loop through each component interface coordinate
C
          DO 55 J = 1,IID(I,1)+IIL(I,1)
C
C  Index column pointers to first column
C
          ND = 1
          NL = 1
          NDL = 1
          IROW = IRST(IID(I,2)+IIL(I,2)-IID(I,1)-IIL(I,1)+J)
C
C  ....... Continued ........
C
```

Figure 4.3. Code Section for Reading and Assembling Matrix Partitions

```
C
C  If it is a dependent coordinate, read a row from file
C    containing Mcn matrix into matrix partitions Mdn and
C    Mde (stored in two-dimensional arrays AMDN and AMDE
C     respectively).  Otherwise ..... (1)
C
              DO 60 I1 = 1,IID(I,1)
                IF(ICOORD(IID(I,2)-IID(I,1)+I1,2).EQ.IROW)THEN
                  DO 63 K = 1,IIN(I,1)
C
C  Read part of the row as AMDN
C
 63               READ(3,*)AMDN(MDN,IIN(I,2)-IIN(I,1)+K)
                  DO 64 K = 1,IIE(I,1)
C
C  Read the remaining part as AMDE and increment
C    corresponding row pointer
C
 64               READ(3,*)AMDE(MDN,IIE(I,2)-IIE(I,1)+K)
                MDN = MDN+1
C
C  Loop again through each component interface coordinate and
C    extract equation number
C
                  DO 65 K = 1,IID(I,1)+IIL(I,1)
                  IROW2 = IRST(IID(I,2)+IIL(I,2)-IID(I,1)-IIL(I,1)+K)
C
C  If it is a dependent coordinate, then read input from files
C    containing Kcc and Mcc matrices into matrix partitions
C    Kdd and Mdd respectively ( stored in two-dimensional
C    arrays AKDD and AMDD respectively) and increment the
C    corresponding column pointer.  Otherwise.....(2)
C
                    DO 70 I2 = 1,IID(I,1)
                      IF(ICOORD(IID(I,2)-IID(I,1)+I2,2).EQ.IROW2)THEN
                      READ(1,*)AMDD(MD,IID(I,2)-IID(I,1)+ND)
                      READ(2,*)AKDD(MD,IID(I,2)-IID(I,1)+ND)
                      ND = ND+1
                      GOTO 65
                      ENDIF
 70                 CONTINUE
C
C  ....... Continued ........
C
```

Figure 4.3. Code Section for Reading and Assembling Matrix Partitions (continued)

```
C
C  (2)... read input from files containing Kcc and Mcc matrices
C    into matrix partitions Kdl and Mdl respectively (stored in
C    two-dimentional arrays AKDL and AMDL respectively) and
C    increment corresponding column pointer.
C
                READ(1,*)AMDL(MDL,IIL(I,2)-IIL(I,1)+NDL)
                READ(2,*)AKDL(MDL,IIL(I,2)-IIL(I,1)+NDL)
                NDL = NDL+1
 65           CONTINUE
C
C  Having completed reading one row, Increment corresponding
C    row pointers
C
             MD = MD + 1
             MDL = MDL + 1
             GOTO 55
             ENDIF
 60        CONTINUE
           DO 73 K = 1,IIN(I,1)
C
C  (1).....Read a row from file containing Mcn matrix into matrix
C    partitions Mln and Mle (stored in two-dimensional arrays AMLN
C    and AMLE respectively).  Read part of the row as AMLN
C
 73        READ(3,*)AMLN(MLN,IIN(I,2)-IIN(I,1)+K)
           DO 74 K = 1,IIE(I,1)
C
C  Read remaining part of the row as AMLE and
C    increment corresponding row pointer
C
 74        READ(3,*)AMLE(MLN,IIE(I,2)-IIE(I,1)+K)
      MLN = MLN + 1
C
C  ....... Continued ........
C
```

Figure 4.3. Code Section for Reading and Assembling Matrix Partitions (continued)

```
C
C  Loop through each component interface coordinate.  If dependent,
C    it forms a partition of Kdl or Mdl.  Hence skip (read it as
C    dummy input). Otherwise....(3)
C
            DO 71 K = 1,IID(I,1)+IIL(I,1)
            IROW2 = IRST(IID(I,2)+IIL(I,2)-IID(I,1)-IIL(I,1)+K)
              DO 75 I2 = 1,IID(I,1)
                IF(ICOORD(IID(I,2)-IID(I,1)+I2,2).EQ.IROW2)THEN
                READ(1,*)DUM
                READ(2,*)DUM
                GOTO 71
                ENDIF
 75             CONTINUE
C
C  (3)... Read input from files storing Mcc and Kcc matrices
C    into matrix partitions Mll and Kll respectively (stored
C    in two-dimensional arrays AMLL and AKLL respectively) and
C    increment corresponding column pointer
C
              READ(1,*)AMLL(ML,IIL(I,2)-IIL(I,1)+NL)
              READ(2,*)AKLL(ML,IIL(I,2)-IIL(I,1)+NL)
              NL = NL + 1
 71           CONTINUE
C
C  Increment independent coordinate column pointer
C
            ML = ML+1
 55         CONTINUE
C
C  Close input files before proceeding to the next component
C
          CLOSE(1)
          CLOSE(2)
          CLOSE(3)
 50       CONTINUE
```

Figure 4.3. Code Section for Reading and Assembling Matrix Partitions (continued)

$$\mathbf{M}_{cc} = \mathbf{M}_{\ell\ell} - (\mathbf{C}_{dd}^{-1}\mathbf{C}_{d\ell})^{T}\mathbf{M}_{\ell d} - \mathbf{M}_{d\ell}^{T}(\mathbf{C}_{dd}^{-1}\mathbf{C}_{d\ell}) + (\mathbf{C}_{dd}^{-1}\mathbf{C}_{d\ell})^{T}\mathbf{M}_{dd}(\mathbf{C}_{dd}^{-1}\mathbf{C}_{d\ell}); \tag{4.4}$$

$$\mathbf{M}_{cn} = \mathbf{M}_{\ell n} - (\mathbf{C}_{dd}^{-1}\mathbf{C}_{d\ell})^{T}\mathbf{M}_{dn}, \tag{4.5}$$

where the subscript 'c' is equivalent to 'ℓ'. The section of code that accomplishes this is shown in Figure 4.4. Subroutines used for this purpose are available in [51]. First, the expression $-(\mathbf{C}_{dd}^{-1}\mathbf{C}_{d\ell})$ is evaluated and the result is stored in the two-dimensional array `CDL`. The matrix partitions $\mathbf{K}_{cc}$, $\mathbf{M}_{cc}$ and $\mathbf{M}_{cn}$ are then constructed as seen in the figure. The system matrices are constructed in the partitioned form as

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{cc} & \mathbf{0} \\ \mathbf{0} & \Lambda_{nn} \end{bmatrix}; \mathbf{M} = \begin{bmatrix} \mathbf{M}_{cc} & \mathbf{M}_{cn} \\ \mathbf{M}_{nc} & \mathbf{I}_{nn} \end{bmatrix}. \tag{4.6}$$

in the case of systems with no rigid-body modes. However, the presence of rigid-body modes results in a singular system stiffness matrix which cannot be used effectively for eigensolution. In such circumstances, the shifting technique suggested by Bathe [2] can be used to construct a shifted stiffness matrix $\mathbf{K} + \rho\,\mathbf{M}$ where ρ is the eigenvalue shift. For our purpose, we calculate the shift `ASF` as the lowest eigenvalue of all the components. This shifted stiffness matrix along with the mass-matrix is used for eigensolution by the Jacobi method. Subroutine JACOBI, which is used to find the complete eigensolution of the generalized eigenvalue problem of the form

$$\mathbf{K}\,\mathbf{x} = \lambda\,\mathbf{M}\,\mathbf{x}, \tag{4.7}$$

where $\mathbf{K}$ and $\mathbf{M}$ are symmetric and positive definite, is available in [51]. The use of a spectrum shift ρ results in eigenvalues greater by ρ, so that the original system eigenvalues are determined by subtracting ρ from all the eigenvalues; the eigenvectors are stored in the two-dimensional array `PHI`.

```
C
C  Find NEGATIVE(INVERSE(CDD)*CDL) assign back to CDL.
C   (Note: negative sign already taken care of during the
C   formation of CDD and CDL). ITEMP is a temporary array
C
      CALL SIMEQ(CDD,CDL,ID,IL,ITEMP,6)
C
C  Construct matrix partitions AMLL, AMLN, AKLL,
C
C  Form matrix AMLN
C
      CALL TRANS(CDL,CLD,ID,IL)
      CALL MATMUL(CLD,AMDN,TEMP,IL,ID,IN)
      CALL MATADD(AMLN,TEMP,AMLN,IL,IN)
C
C  Form matrix AMLL
C
      CALL MATMUL(CLD,AMDD,TEMPD,IL,ID,ID)
      CALL MATMUL(CLD,AMDL,TEMPL,IL,ID,IL)
      CALL MATADD(AMLL,TEMPL,AMLL,IL,IL,IL)
      CALL TRANS(TEMPL,TEMPL1,IL,IL)
      CALL MATADD(AMLL,TEMPL1,AMLL,IL,IL)
      CALL SYMMATMUL(TEMPD,CDL,TEMPL,IL,ID,IL)
      CALL MATADD(AMLL,TEMPL,AMLL,IL,IL)
C
C  Form matrix AKLL
C
      CALL MATMUL(CLD,AKDD,TEMPD,IL,ID,ID)
      CALL MATMUL(CLD,AKDL,TEMPL,IL,ID,IL)
      CALL MATADD(AKLL,TEMPL,AKLL,IL,IL,IL)
      CALL TRANS(TEMPL,TEMPL1,IL,IL)
      CALL MATADD(AKLL,TEMPL1,AKLL,IL,IL)
      CALL SYMMATMUL(TEMPD,CDL,TEMPL,IL,ID,IL)
      CALL MATADD(AKLL,TEMPL,AKLL,IL,IL)
C
C  ..... Continued ........
C
```

Figure 4.4. Code Section for Assembling System Matrix Partitions and Solving the Initial Problem

```
C
C  Find lowest component eigenvalue and determine shift
C    ASF for Jacobi solution
C
        IF(IR.EQ.0)THEN
        ASF = 0.D0
        ELSE
        ASF = AKNN(1)
          DO 316 I = 2,NN
          IF(AKNN(IIN(I,2)-IIN(I,1)+1).LT.ASF)ASF =
     .    AKNN(IIN(I,2)-IIN(I,1)+1)
 316      CONTINUE
        ENDIF
C
C  Construct system stiffness and mass matrices KK and MM
C
 505    DO 520 I = 1,IL
          DO 510 J = I,IL
          MM(J,I) = AMLL(I,J)
          MM(I,J) = MM(J,I)
          KK(J,I) = AKLL(I,J)+ASF*AMLL(I,J)
 510      KK(I,J) = KK(J,I)
          DO 520 J = 1,IN
          KK(IL+J,I) = ASF*AMLN(I,J)
          KK(I,IL+J) = KK(IL+J,I)
          MM(IL+J,I) = AMLN(I,J)
          MM(I,IL+J) = MM(IL+J,I)
 520    CONTINUE
        DO 530 I = 1,IN
          DO 530 J = I+1,IN
          KK(IL+I,IL+J) = 0.D0
          KK(IL+J,IL+I) = 0.D0
          MM(IL+I,IL+J) = 0.D0
 530    MM(IL+J,IL+I) = 0.D0
        DO 540 I = 1,IN
        KK(IL+I,IL+I) = AKNN(I)+ASF
 540    MM(IL+I,IL+I) = 1.D0
C
C  Solve Eigen problem by the Jacobi method, for accuracy up
C    to IS significant digits in the frequency.  (The
C    corresponding value for the eigenvalues is 2*10**(-IS))
C
 200  CALL JACOBI(KK,MM,PHI,AN0,IL+IN,2.D0*10.D0**(-IS),25,DD,SING,6)
```

Figure 4.4. Code Section for Assembling System Matrix Partitions and Solving the Initial Problem (continued)

4.5. Convergence of Initial Modes

After determining the initial eigenvalues, the convergence scheme is invoked. Figure 4.5 shows the section of code that is used for this purpose. For this, the matrix $\mathbf{m}_{cc}$ of equation (3.4) on page 25 is determined. First, the system $\mathbf{M}_{cc}$ matrix is determined in a similar way as the $\mathbf{M}_{cn}$ matrix using the expression

$$\mathbf{M}_{cc} = \mathbf{M}_{\ell e} - (\mathbf{C}_{dd}^{-1}\,\mathbf{C}_{d\ell})^{T}\,\mathbf{M}_{de}\,, \tag{4.8}$$

and stored in the two-dimensional array AMLE. (Again, the subscripts 'c' and 'ℓ' are equivalent.) Then, the elements of the eigenvectors of the initial problem corresponding to the connection coordinates are used to construct the $\mathbf{m}_{cc}$ matrix according to equation (3.4) on page 25. Each eigenvalue-eigenvector pair for which the frequency is less than the user specified cut-off is converged in turn. Prior to convergence, a convergence tolerance is set. This is specified as the number of significant digits of accuracy required (assigned to the parameter IS) and iterations are performed until this is achieved or until the maximum number of iterations, IMAX, is reached. First the matrix $\mathbf{N}(\lambda)$ is evaluated and stored in array AN. This is done by calling subroutine SUBAN, whose operation is described later. Next, an iteration is performed for the eigenvalue AL using the matrix AN and the vector Q, according to equation (3.49) on page 37. This is done through subroutine SUBDL which evaluates a correction to the eigenvalue and updates it. Detailed operation of the subroutine is given later in the chapter. The change in the eigenvalue is checked to see if convergence has been obtained. If found converged, one iteration for the eigenvector is performed as the last step in the iteration. If not converged, an iteration is performed for the eigenvector according to equation (3.49) on page 37. This is done using the subroutine SUBDQ, which, given an eigenvector Q, calculates a correction and uses it to update Q. In doing so, the norm of the correction vector is calculated and stored in the variable DELQ. This norm can also be used to determine if convergence has been attained. More details on this subroutine are described later. This completes one iteration. This cycle is repeated until all iterations are performed or convergence to the specified tolerance has been attained. Finally, the converged eigenvector is back-calculated in the component physical dof coordinates.

```
C
C  Calculate MCE matrix
C  First, construct matrix AMLE
C
        CALL MATMUL(CLD,AMDE,TEMPE,IL,ID,IE)
        CALL MATADD(AMLE,TEMPE,AMLE,IL,IE)
C
C  Construct MCE matrix as MCE = (PHI)*(AMLE).  First extract
C    portion of PHI corresponding to connection coordinates
C
       DO 310 I = 1,IL
         DO 310 J = 1,IL+IN
 310   MC(J,I) = PHI(I,J)
       CALL MATMUL(MC,AMLE,MCE,IL+IN,IL,IE)
C
C  Converge eigenvalues less than the cut-off value FCUT
C
        DO 710 I = 1,IL+IN
        AL = AN0(I)
        IF(AL.GT.FCUT) GOTO 710
C
C  Initialize eigenvector in initial mode coordinates
C
          DO 730 J = 1,IL+IN
 730      Q(J,1) = 0.D0
       Q(I,1) = 1.D0
C
C  Converge eigenvalue and eigenvector. Initialize iteration counter I1
C
        I1 = -1
C
C  Evaluate AN at AL
C
        CALL SUBAN(AN,MCE,IL+IN,IE,AN0,AL,AKEE)
C
C  Increment iteration counter. Iterate a maximum of IMAX times or till
C    the convergence tolerance has been reached for the eigenvalue. For
C    convergence to IS significant digits in frequency, the tolerance
C    for the eigenvalues is 2 * 10(**-IS). If the maximum iteration
C    count IMAX has been reached, stop iterating
C
 743    I1 = I1+1
        IF(I1.GE.IMAX)GOTO 745
C
C  Iterate for the eigenvalue AL
C
 742    CALL SUBDL(AN,IL+IN,AL,Q,Q1,DL)C

C
C ...........  Continued ..............
C
```

Figure 4.5. Code Section for Converging Eigenparameters

```
C
C  If there seems to be a convergence problem, i.e., there is a
C    negative change in the eigenvalue after the first iteration,
C    stop iterating.  Otherwise .... (1)
C
          IF(I1.GT.0.AND.DL.LT.0.D0)THEN
          AL = AL-DL
          GOTO 745
          ENDIF
C
C  (1)....  Check if the eigenvalue has converged. If so, iterate
C    once for the eigenvector and stop iterating
C
        IF(DABS(DL/AL).LE.2.D0*10.D0**(-IS))THEN
        CALL SUBDQ(AN,AN0,IL+IN,AL,Q,Q1,DQ,DELQ)
        GOTO 745
        ENDIF
C
C  Evaluate AN at AL
C
        CALL SUBAN(AN,MC2,IL+IN,IE,AN0,AL,AKEE)
C
C  For the given AL, evaluate the eigenvector Q
C
 740    CALL SUBDQ(AN,AN0,IL+IN,AL,Q,Q1,DQ,DELQ)
C
C  If there is a convergence problem, i.e., if the i'th element in
C    vector Q changes more than in the previous iteration, stop
C    iterating
C
        IF(I1.EQ.0)THEN
        DQSTD = DABS(DQ(I,1))
        ELSE
          IF(DABS(DQ(I,1)).GE.DQSTD)THEN
            DO 756 II1 = 1,IL+IN
 756        Q(II1,1) = Q(II1,1)-DQ(II1,1)
          AL = AL-DL
          GOTO 745
          ELSE
          DQSTD = DABS(DQ(I,1)
          ENDIF
        ENDIF
C
C  Start next iteration
C
        GOTO 743
 745  CONTINUE
C
C  Back-calculate eigenvector and write to output
C   ..... (Include code here) ......
C
 710    CONTINUE
```

Figure 4.5. Code Section for Converging Eigenparameters (continued)

During trial runs with the present algorithm, it was found that sometimes multiple initial eigenvalues converged to the same final eigenvalue. To avoid this, the iteration is aborted whenever there is a tendency for this to happen and the current eigenvalue when the iteration is stopped is taken as the converged eigenvalue. This tendency to converge to a different eigenvalue is indicated when the largest element of the vector **q** that starts off as unity, changes significantly after the first iteration. For proper convergence, the maximum change in the largest element of the vector **q** occurs during the first iteration and changes during subsequent iterations are much smaller than this. Also, during the first iteration, the eigenvalue drops to a value less than its converged value and increases during subsequent iterations. Whenever deviations from this behavior are observed, convergence is aborted.

Figure 4.6 shows a section of code for the back-calculation of the system eigenvectors upon completion of convergence. The converged eigenvector (array Q) has elements corresponding to the connection and initial normal mode coordinates only. To obtain the eigenvector components in the additional mode coordinates, equation (3.16) on page 29 is used. Thus, array PE is determined. The resulting eigenvector is of the form $< \mathbf{q}\ \mathbf{p}_e >^T$ and is not mass normalized. Therefore, first the norm of the eigenvector with respect to the mass matrix is calculated as

$$\left\| \begin{Bmatrix} \mathbf{q} \\ \mathbf{p}_e \end{Bmatrix} \right\| = \left(\begin{Bmatrix} \mathbf{q} \\ \mathbf{p}_e \end{Bmatrix}^T \begin{bmatrix} \mathbf{I} & \mathbf{m}_{ce} \\ \mathbf{m}_{ce}^T & \mathbf{I} \end{bmatrix} \begin{Bmatrix} \mathbf{q} \\ \mathbf{p}_e \end{Bmatrix} \right)^{1/2} . \tag{4.9}$$

This norm is stored in the variable ANORM and is used to scale the eigenvector. Next, equation (3.12) on page 27 is used to transform the eigenvector to component generalized coordinates, i.e.,

$$\mathbf{q}_l = \begin{bmatrix} \Phi & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{Bmatrix} \mathbf{q} \\ \mathbf{p}_e \end{Bmatrix} = \begin{Bmatrix} \Phi\, \mathbf{q} \\ \mathbf{p}_e \end{Bmatrix}. \tag{4.10}$$

Once $\mathbf{q}_l$ is determined, components belonging to the connection coordinates are separated and assigned to the array PL. To obtain the eigenvector components in dependent coordinates, the relation

```
C
C  Calculate converged eigenvectors in component physical dof
C    coordinates and write to component output files.  Calculate
C    eigenvector components in additional mode coordinates
C
 745      DO 262 J = 1,IE
          PE(J,1) = 0.D0
            DO 263 JJ = 1,IL+IN
 263        PE(J,1) = PE(J,1)+MCE(JJ,J)*Q(JJ,1)
 262      PE(J,1) = PE(J,1)*AL/(AKEE(J)-AL)
C
C  Calculate eigenvector components in initial mode coordinates
C
        CALL MATMUL(MCE,PE,Q1,IL+IN,IE,1)
C
C  Calculate eigenvector norm ANORM with respect to the mass matrix
C
        ANORM = 0.D0
          DO 847 J = 1,IL+IN
 847      ANORM = ANORM + Q(J,1)*Q(J,1) + 2.D0*Q(J,1)*Q1(J,1)
          DO 848 J = 1,IE
 848      ANORM = ANORM + PE(J,1)*PE(J,1)
        ANORM = 1.D0/DSQRT(ANORM)
C
C  Scale eigenvector using ANORM
C
        CALL SCALMUL(ANORM,Q,Q,IL+IN,1)
        CALL SCALMUL(ANORM,PE,PE,IE,1)
C
C  Calculate components of eigenvector in component
C    generalized coordinates.  Calculate components
C    in independent coordinates
C
        CALL MATMUL(PHI,Q,Q1,IL+IN,IL+IN,1)
          DO 257 J = 1,IL
 257      PL(J,1) = Q1(J,1)
C
C  Calculate components in dependent coordinates
C
        CALL MATMUL(CDL,PL,PD,ID,IL,1)
C
C  Calculate components in normal mode coordinates
C
          DO 258 J = 1,IN
 258      PN(J,1) = Q1(IL+J,1)
C
C  ..... Continued .......
C
```

Figure 4.6. Code Section for the Back-Calculation of Eigenvectors

```
C
C  Initialize dependent and independent coordinate pointers
C
      MD = 1
      ML = 1
C
C  Loop through components
C
        DO 253 II = 1,NN
C
C  Open component eigenvector data file for output and write eigenvalue
C
        OPEN(UNIT=1,FILE=IIFINV(II),STATUS='OLD')
        WRITE(1,9)DSQRT(AL)/(2.D0*PI),AL
 9      FORMAT(10X,'Converged Frequency = ',G13.8,' Hz',/,G21.14)
C
C  Initialize eigenvector in component physical dof coordinates
C
          DO 871 JJ = 1,NMAX
 871      EIGVEC(JJ) = 0.D0
C
C  Loop through each interface coordinate of the component
C
          DO 254 JJ = 1,IID(II,1)+IIL(II,1)
          IROW = IRST(IID(II,2)+IIL(II,2)-IID(II,1)-IIL(II,1)+JJ)
C
C  Search in second column of array ICOORD to see if the
C    coordinate is dependent.  If so, multiply component of PD
C    with the appropriate constraint mode of the component
C    and increment pointer
C

            DO 255 JJ1 = IID(II,2)-IID(II,1)+1,IID(II,2)
              IF(IROW.EQ.ICOORD(JJ1,2))THEN
                DO 872 JJ2 = 1,IINDOF(II)
                  EIGVEC(JJ2)=EIGVEC(JJ2)+PD(MD,1)*PHIC(IIPSUM(II,2)-
     .            IIPSUM(II,1)+(JJ2-1)*(IID(II,1)+IIL(II,1))+JJ)
 872            CONTINUE
              MD = MD + 1
              GOTO 254
              ENDIF
 255        CONTINUE
C
C  ........Continued..........
C
```

Figure 4.6. Code Section for the Back-Calculation of Eigenvectors (continued)

```
C
C  Otherwise, multiply component of PL with the appropriate
C    constraint mode of the component and increment pointer
C
                DO 873 JJ2 = 1,IINDOF(II)
                  EIGVEC(JJ2)=EIGVEC(JJ2)+PL(ML,1)*PHIC(IIPSUM(II,2)-
     .            IIPSUM(II,1)+(JJ2-1)*(IID(II,1)+IIL(II,1))+JJ)
 873            CONTINUE
              ML = ML + 1
 254          CONTINUE
C
C  Multiply normal mode components of the eigenvector with the
C    appropriate component normal modes
C
              DO 256 JJ = 1,IIN(II,1)
                DO 874 JJ2 = 1,IINDOF(II)
                EIGVEC(JJ2) = EIGVEC(JJ2)+PN(IIN(II,2)-IIN(II,1)+JJ,1)*
     .          PHIN(IINSUM(II,2)+IIESUM(II,2)-IINSUM(II,1)-IIESUM(II,1)+
     .          (JJ2-1)*(IIN(II,1)+IIE(II,1))+JJ)
 874            CONTINUE
  256         CONTINUE
C
C  Multiply additional mode components of the eigenvector with the
C    appropriate component additional normal modes
C
              DO 264 JJ = 1,IIE(II,1)
                DO 875 JJ2 = 1,IINDOF(II)
                EIGVEC(JJ2) = EIGVEC(JJ2)+PE(IIE(II,2)-IIE(II,1)+JJ,1)*
     .          PHIN(IINSUM(II,2)+IIESUM(II,2)-IINSUM(II,1)-IIESUM(II,1)+
     .          (JJ2-1)*(IIN(II,1)+IIE(II,1))+IIN(II,1)+JJ)
 875            CONTINUE
 264          CONTINUE
C
C  Write system eigenvector in component physical dof
C    coordinates to component eigenvector output file
C
           WRITE(1,*)(EIGVEC(JJ),JJ=1,IINDOF(II))
           CLOSE(1)
 253       CONTINUE
```

Figure 4.6. Code Section for the Back-Calculation of Eigenvectors (continued)

$$p_d = -(C_{dd}^{-1} C_{d\ell})\, p_\ell \tag{4.11}$$

is used to determine array PD. Components of the eigenvector in normal mode coordinates are also extracted and stored in the array PN. Once this is complete, appropriate elements of the eigenvector are written to component eigenvector output files. The system eigenvector ϕ in component physical dof coordinates is given by the expression

$$\phi = \phi_d p_d + \phi_\ell p_\ell + \phi_n p_n + \phi_e p_e \,. \tag{4.12}$$

To calculate this, a loop is executed through the components. For each component, the component eigenvector EIGVEC is initialized to zero. This array is of dimension NMAX which is the maximum number of dofs in any component. This done, a loop is executed through each component interface coordinate in turn. If this is a dependent coordinate, the corresponding element of the array PD is multiplied by the appropriate constraint mode of that component; otherwise, the component of array PL is multiplied. Again, looping through each normal mode coordinate and additional normal mode coordinate, the appropriate eigenvector components are multiplied with the corresponding initial or additional mode of that component respectively. Finally, the eigenvector in component physical dof coordinates is written to the component eigenvector output file.

This completes the execution of the program. Now, the operations of subroutines SUBAN, SUBDL and SUBDQ are discussed.

Figure 4.7 shows a listing of subroutine SUBAN to evaluate the $N(\lambda)$ matrix and store it in the two-dimensional array AN. The value of λ at which this matrix is computed is stored in the variable AL. Diagonal symmetry of the matrix is exploited by only evaluating elements above the main diagonal. From equation (3.21) on page 30, the diagonal element in the i^{th} row is evaluated using the expression

```
C  ------------------------------------------------------------
C
C  Subroutine SUBAN to evaluate the functional lambda matrix AN
C
C  ------------------------------------------------------------
C
      SUBROUTINE SUBAN(AN,MCE,N,IE,AN0,AL,AKEE)
C
C  Uses - MCE,N,IE,AN0,AL,AKEE
C  Returns - AN
C
C  AN is the functional lambda matrix of order N x N, where N is the
C    number of initial eigenvalues and eigenvectors already solved for
C  MCE is the addendum matrix to the mass matrix and is of order N x IE,
C    where IE is the number of additional modes introduced
C  AN0 is the vector of the initially computed eigenvalues (size N)
C  AL is the value of lambda at which the AN matrix is to be evaluated
C  AKEE is the vector (of size IE) of additional eigenvalues
C    used for convergence, and corresponds to the columns of MCE
C
      IMPLICIT DOUBLE PRECISION(A-H,L,M,O-Z)
      INTEGER N,IE
      DIMENSION AN(N,N),MCE(N,IE),AN0(N),AKEE(IE)
C
      DO 10 I = 1,N
C
C  Assign elements below diagonal by symmetry
C
        DO 20 J = 1,I-1
 20     AN(I,J) = AN(J,I)
        DO 10 J = I,N
        AN(I,J) = 0.D0
C
C  Add (AN0(I)-AL) for a diagonal element.  Otherwise ... (1)
C
        IF(I.EQ.J)AN(I,J) = AN0(I)-AL
C
C  (1)... for non-diagonal elements compute AN matrix as follows
C
          DO 10 K = 1,IE
          AN(I,J) = AN(I,J) - AL*AL*MCE(I,K)*MCE(J,K)/(AKEE(K)-AL)
 10   CONTINUE
      RETURN
      END
C
C  -----------------  End subroutine SUBAN --------------------
C
```

Figure 4.7. Listing of Subroutine SUBAN

$$N(i,i) = \Lambda(i,i) - \lambda - \lambda^2 \sum_{r=1}^{e} \frac{m_{ce}(i,r)^2}{\Lambda_{ee}(r,r) - \lambda}, \tag{4.13}$$

while a non-diagonal element is evaluated by the expression

$$N(i,j) = -\lambda^2 \sum_{r=1}^{e} \frac{m_{ce}(i,r) m_{ce}(j,r)}{\Lambda_{ee}(r,r) - \lambda}. \tag{4.14}$$

The input to this subroutine is the matrix $\mathbf{m}_{ce}$ stored in the two-dimensional array `MCE`, the array `AN0` storing the initial eigenvalues, the value `AL` of λ at which the matrix `AN` is to be evaluated, and the array `AKEE` storing the diagonal elements of the Λ_{ee} matrix. The subroutine returns the matrix $N(\lambda)$ stored in array `AN` upon exit.

Figure 4.8 shows a listing of subroutine SUBDL which evaluates a correction and updates the eigenvalue. The input to the subroutine is the $N(\lambda)$ matrix stored in the array `AN`, the current eigenvalue `AL`, and the current eigenvector stored in array `Q`. Subroutine MATMUL [51] is called to evaluate $\mathbf{q}_1$ as

$$\mathbf{q}_1 = N(\lambda)\, \mathbf{q} \tag{4.15}$$

and the result is stored in the array `Q1`. The correction λ_c to the eigenvalue is evaluated as

$$\lambda_c = \mathbf{q}^T \mathbf{q}_1. \tag{4.16}$$

This is stored in the variable `DL`. The current eigenvalue is updated with the correction prior to exit.

Figure 4.9 shows a listing of subroutine SUBDQ. This is used to update the eigenvector during an iteration. The subroutine uses arrays `AN` and `AN0` which store the matrices $N(\lambda)$ and N_0

```
C  ---------------------------------------------------------------
C
C  Subroutine SUBDL to evaluate the eigenvalue during an iteration
C
C  ---------------------------------------------------------------
C
      SUBROUTINE SUBDL(AN,N,AL,Q,Q1,DL)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION AN(N,N),Q(N,1),Q1(N,1)
C
C  Uses - AN,N,AL,Q
C  Returns - AL,DL
C  Modifies - AL
C
C  AN is the functional lambda matrix of size N x N
C  Q is the Eigenvector
C  Q1 is a working vector
C  DL is the change in the eigenvalue AL
C
      CALL MATMUL(AN,Q,Q1,N,N,1)
      DL = 0.D0
        DO 10 I = 1,N
        DL = DL+Q(I,1)*Q1(I,1)
 10     CONTINUE
      AL = AL+DL
      RETURN
      END
C
C  ------------------ End subroutine SUBDL ----------------------
C
```

Figure 4.8. Listing of Subroutine SUBDL

```
C -------------------------------------------------------------
C
C  Subroutine SUBDQ to evaluate the vector Q during an iteration
C
C -------------------------------------------------------------
C
      SUBROUTINE SUBDQ(AN,AN0,N,AL,Q,Q1,DQ,DELQ)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION AN(N,N),AN0(N),Q(N,1),DQ(N,1),Q1(N,1)
C
C  Calls - MATMUL
C  Uses - AN,AN0,N,AL,Q
C  Returns - Q,DQ,DELQ
C  Modifies - Q
C
C  AN is the functional lambda matrix of size N x N
C  AN0 is the array of initially computed eigenvalues size N
C  Q is the eigenvector
C  DELQ is the norm of the change in Q
C  AL is the value of lambda at which all evaluations are performed
C  DQ and Q1 are working vectors
C
C  Find (AN)(Q) = (DQ) and Q1 = Q - INVERSE(AN0+AL*AN1)*AN*Q
C
      CALL MATMUL(AN,Q,DQ,N,N,1)
      SUM = 0.D0
        DO 10 I = 1,N
        DQ(I,1) = DQ(I,1)/(AN0(I)-AL)
        Q1(I,1) = Q(I,1)+DQ(I,1)
C
C  Calculate norm and scale Q1 using norm
C
 10     SUM = SUM+Q1(I,1)*Q1(I,1)
      SUM = DSQRT(SUM)
      DELQ = 0.D0
        DO 20 I = 1,N
        Q1(I,1) = Q1(I,1)/SUM
C
C  Calculate DQ = (Q-Q1) and norm DELQ of DQ
C
        DQ(I,1) = Q1(I,1) - Q(I,1)
        Q(I,1) = Q1(I,1)
 20     DELQ = DELQ+DQ(I,1)*DQ(I,1)
      DELQ = DSQRT(DELQ)
      RETURN
      END
C
C  --------------- End subroutine SUBDQ ---------------------
C
```

Figure 4.9. Listing of Subroutine SUBDQ

respectively, and the current eigenvector **q** stored in array Q which is updated using the equation (3.49) on page 37. Again, the diagonal structure of the matrix to be inverted is used to simplify the calculations. It is to be noted that the norm of the updated eigenvector is not necessarily unity. Hence, it has to be scaled. First, the norm of the updated eigenvector is calculated and stored in the variable SUM, which is now used to scale Q1. The change in the eigenvector is then calculated as the difference between Q1 and Q and stored in the array DQ. A measure of this change is obtained by evaluating its norm and storing it in the variable DELQ. This can be used as a measure of convergence. The updated eigenvector stored in array Q1 is reassigned to Q before exit.

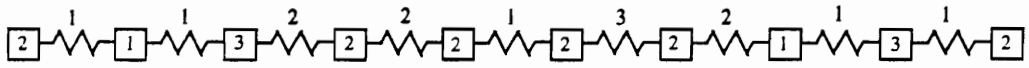
5.0 Numerical Examples

In this chapter, some numerical examples will be presented to demonstrate the applicability of the CMS method developed in the previous chapters. The first example involves the synthesis of a spring-mass system from two components. The second example involves the synthesis of a free-free beam from three components. Both beam and shell-element models are studied. The third example involves the synthesis of a rectangular plate with both free-free and cantilevered boundary conditions. In the final example, a refrigeration compressor assembly composed of a housing, compressor mechanism, mounting springs and a discharge tube is synthesized using the modes of its constituent components. Details of modeling, analysis and results obtained are given below.

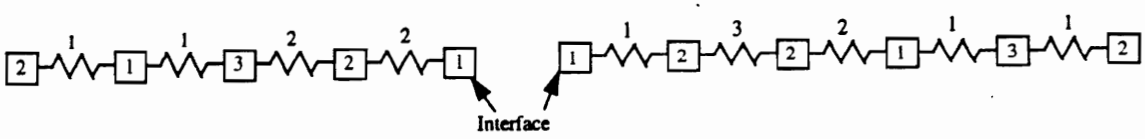
5.1 Example 1 - Synthesis of a Spring-Mass System

The first case was the synthesis of a two-component spring-mass system shown in Figure 5.1. Mass and stiffness values are indicated in the figure with the mass values shown inside the boxes. The components had five and six dofs respectively, so that the complete system had ten dofs.

First, for comparison, the system was solved for all its ten eigenvalues and eigenvectors which included a rigid-body mode with a zero eigenvalue. Next, the two components were solved for all their eigenvalues and eigenvectors, by fixing the interface coordinates. The eigenvalues for the two components are given in Table 5.1. Next, the rigid-body modes for the two components were determined. For both the components, the rigid-body modes are the same as the constraint modes since we have only one connection dof and one rigid-body mode for each component.



System



Component 1 **Component 2**

Figure 5.1. Full System and Components used for Example 1

Table 5.1. Eigenvalues of Components

Mode	Component 1 Eigenvalues	Component 2 Eigenvalues
1	1.078×10^{-1}	5.404×10^{-2}
2	5.290×10^{-1}	3.962×10^{-1}
3	2.174	9.628×10^{-1}
4	2.690	2.614
5		4.639

For the initial solution, the first normal mode of Component 1 and the first two normal modes of Component 2 were used in synthesis, the cut-off eigenvalue being about 0.4. These are the eigenvalues shown above the dotted line in Table 5.1. Thus, four system modes could be calculated of which three are within the cut-off frequency. These are compared with the exact system modes obtained from a complete eigensolution of the entire system and labeled as reference in Table 5.2. It is seen that the error is as high as 7.14% for the third eigenvalue. In the second step, the next two normal modes of Component 1 and the next normal mode of Component 2 were introduced as additional modes for convergence. In Table 5.1, these eigenvalues appear above the solid line. The converged eigenvalues are also reported in Table 5.2. Note that the error in the third eigenvalue is only about 0.08% after convergence. Before introducing the convergence scheme, these additional modes were used along with the initial modes in a one-step CMS solution, and the synthesized eigenvalues were identical.

Table 5.2. Results for Example Case 1

Mode	Reference Eigenvalues	Initial Eigenvalues (Error from Reference Solution - %)	Converged Eigenvalues (Error from Reference Solution - %)
1	0	0	0
	(0)	(0)	(0)
2	7.31855×10^{-2}	7.32913×10^{-2}	7.31874×10^{-2}
		(0.15)	(0.01)
3	2.52264×10^{-1}	2.70215×10^{-1}	2.52447×10^{-1}
		(7.14)	(0.08)

5.2 Example 2 - Beam Synthesis from Three Components

A free-free beam of length 1 m, width 1.5 cm and thickness 1 mm. The beam model was chosen for study so that natural frequencies could be obtained analytically and can be used for comparison with CMS results and eigensolution using the full model. The beam was arbitrarily divided into three components of lengths 40 cm, 5 cm and 55 cm, which when connected end to end form the complete system as shown in the figure. The second beam component played the role of a connector. The objective of the study was to study the synthesis capability of the method when the stiffness of the connector component was made low compared to the other two components. Four different cases were studied. Finite element models for the first two cases were constructed with beam elements. Component 1 was discretized into 80 elements, Component 2 into 40 and Component 3 into 110 elements. For the first case, all components had an elastic modulus of 2×10^{11} Pa, Poisson's ratio of 0.3 and a mass density of 7800 kg/m^3 corresponding to steel. For the second case, the stiffness of the connector component (Component 2) was decreased by changing its modulus of elasticity to 2×10^9 Pa. The mass density value used for this component was 2600 kg/m^3, while the property values of other components were unaltered.

The beam elements were formulated in three dimensions by superposing bending in two planes with axial deformation and torsion. This gave each node six dofs and the complete element a 12 x 12 stiffness matrix. Details of formulation are not provided as this is not the focus of the study; however some features of the element used will be mentioned [1]. The beam portion of the stiffness matrix was formed by the reduced integration of a shear-deformable beam element with linear interpolation functions for both displacements and rotations. The element is free from zero-energy deformation modes and shear-locking [1]. The element mass matrix is derived consistently and has terms representing both translatory and rotary inertia. Subroutines for element formulation are available in [51]. The element x-axis is defined as pointing in the direction of the line joining its nodes, the element y-axis is defined as the cross product of the element x-axis with the global z-axis unless they are parallel, in which case, the element y-axis is the global x-axis. The third axis is then automatically defined as being perpendicular to and forming a right-handed triad with the other two.

The third and fourth cases were similar to the first two respectively, but used shell elements instead of beam elements for the finite element models. A regular mesh pattern with six elements across the width of the beam was used; Component 1 was discretized into 360 elements, Component 2 into 120 and Component 3 into 480 elements.

Flat shell elements with four nodes were used for model construction. Although in a linear analysis, there is no coupling between membrane and bending behavior for a structure such as the beam used in these examples, shell elements were required so that in-plane and out-of plane bending modes could be determined. These elements have six degrees of freedom per node, viz., three translations and rotations about the three axes. The elements were formulated by superposing a two-dimensional membrane element with a plate-bending element. The membrane portion of the stiffness matrix was formulated based on MacNeal and Harder's four-noded membrane element with drilling dofs [52]. Complete details are not provided here; only a short description of the element formulation is made. Subroutines that were employed in the calculation of element matrices are available in [51]. The membrane stiffness matrix is initially formulated for an eight-noded parabolic membrane element. Following MacNeal and Harder, transformation equations are written to relate the displacements of the mid-side nodes to rotations of the corner nodes. These equations take the form of a transformation matrix, which operates on the eight-noded membrane element stiffness matrix and reduces it to that appropriate for a four-noded element with corner rotations. Such a formulation however results in a spurious mode with all the corner rotations having identical values. A rank-one stabilization matrix evaluated on the basis of shear strain-energy stored in the element is added to the element stiffness matrix to eliminate the spurious deformation mode as in MacNeal and Harder [52]. However, no smoothing of gauss-point strains was made as discussed by MacNeal and Harder. Thus formulated, the element passes the patch test [1].

A plate bending element developed by Bathe and Dvorkin [53] was used to construct the plate portion of the stiffness matrix. This element is formulated based on the Mindlin-Reissner theory. Use of special interpolation functions results in an element which is free from shear-locking that is a common feature of Mindlin-type plate elements. The key to the formulation is the use of

separate interpolation functions for bending strains and shear strains. The expression for the element strain energy is written as the sum of two terms - the shear strain energy term and the bending strain energy term, each of which leads to two different element stiffness matrices corresponding to shear and bending deformation respectively. The element stiffness matrix is then obtained as the sum of the two stiffness matrices. Thus formulated, the element is free from spurious modes and the problem of shear locking. Also, the element passes the patch test. The element mass matrix was generated consistently by including terms representing both translatory and rotary inertia.

In all cases, a fully assembled finite element model was constructed to provide reference frequencies and mode shapes for verification. This full model was constructed by combining the models of the individual components and hence inherited the same mesh pattern as the components. It was decided to determine the first twenty flexible modes of the system by full eigensolution and by CMS. Subspace iteration was used to obtain the frequencies of the full model with a convergence tolerance of six significant digits. Results from the full model were used for reference and for comparison with those from two-step component mode synthesis.

Next, each component was analyzed separately to determine all the necessary input required for CMS solution, viz., constraint modes, rigid-body modes and fixed-interface normal modes. Two-step component mode synthesis was performed with the cut-off frequency selected to give the first twenty frequencies in all four cases. The number of component modes used for the initial solution up to the cut-off frequency and the total number of additional modes used for convergence are reported in Table 5.3. A comparison of the system cut-off frequency and the highest frequency of component-normal modes used for initial solution is consistent with our philosophy that the minimum number of component normal modes be used for system synthesis. For all the three cases, the highest frequency of component normal modes used was only slightly greater than the maximum system frequency of interest. Two different $x\omega_c$ frequencies for the additional modes - (i) about two and (ii) about four times the cut-off frequency - were studied so

Table 5.3. Number of Initial and Additional Modes used for Example 2

Case	Cut-off Frequency ω_c (Hz)	Total Number of Component Normal Modes Used for Initial Solution	Highest Frequency of Component Modes Used for Initial Solution (Hz)	(i) Total Number of Additional Modes Used for Convergence	(i) Highest Frequency of Additional Modes Used for Convergence (Hz) $\{2\omega_c\}$	(ii) Total Number of Additional Modes Used for Convergence	(ii) Highest Frequency of Additional Modes Used for Convergence (Hz) $\{4\omega_c\}$
1	560	23	609.1	10	1278	30	2420
2	475	23	557.7	12	1197	31	2282
3	560	23	617.6	9	1229	30	2499
4	475	23	570.2	12	1229	29	2255

that convergence could be examined. These frequency limits are designated (i) and (ii) respectively and are also reported in Table 5.3. A convergence tolerance of six significant digits was used for the iterative scheme.

The initial and converged frequencies are reported in Tables 5.4 through 5.7 along with the reference frequencies. For verification, conventional CMS using the Jacobi method and the subspace iteration method were performed for all the four cases using component modes up to about two and four times the cut-off frequencies (the same number that was used in the two-step method), and the frequencies thus obtained were almost identical (maximum difference - 0.0008%).

As mentioned in Chapter 4, a small problem occurred in the iterative process which caused one or two eigenvalues to converge to an adjacent value. The eigenvalue would approach the correct value closely but then wander off to the adjacent value before the six-digit tolerance was reached. The eigenvalue strayed when the value of the corresponding element of the vector **q** which begins at unity suddenly fell below an adjacent value. Convergence was aborted when this was found to occur and the eigenparameters at that stage of iteration were taken as the final values. These frequencies are pointed out in Tables 5.4 through 5.7.

From the tables, it can be observed that there is good general agreement between the reference frequencies and the synthesized frequencies, after convergence. For the initial solution, the

Table 5.4. Comparison of Initial and Converged CMS Frequencies with those by Full-System Eigensolution for Example 2 - Case 1

Mode	Reference Frequencies (Hz)	Initial Frequencies (Hz)	Difference from Reference (%)	(i) Converged Frequencies (Hz)	Difference from Reference (%)	(ii) Converged Frequencies (Hz)	Difference from Reference (%)
1	5.20497	5.20497	0.00	5. 20497	0.00	5.20497	0.00
2	14.3482	14.3483	0.00	14.3482	0.00	14.3482	0.00
3	28.1302	28.1306	0.00	28.1304	0.00	28.1302	0.00
4	46.5052	46.5070	0.00	46.5061	0.00	46.5053	0.00
5	69.4829	69.4868	0.01	69.4836	0.00	69.4830	0.00
6	78.0192	78.0561	0.05	78.0272	0.01	78.0202	0.00
7	97.0584	97.0756	0.02	97.0678	0.01	97.0589	0.00
8	129.257	129.291	0.03	129.264	0.01	129.258	0.00
9	166.054	166.129	0.05	166.090	0.02	166.055	0.00
10	204.462	204.533	0.03	204.468	0.00	204.464	0.00
11	207.493	207.653	0.08	207.547	0.03	207.498	0.00
12	214.806	214.973	0.08	214.963	0.07	214.827	0.01
13	253.558	253.922	0.14	253.652	0.04	253.567	0.00
14	304.246	304.581	0.11	304.450	0.07	304.254	0.00
15	359.642	361.341	0.47	359.916	0.08	359.687	0.01
16	408.935	416.086	1.75	409.874	0.23	409.152	0.05
17	419.612	420.139	0.13	420.110	0.12	419.616	0.00
18	420.385	437.729	4.13	421.726	0.32	420.482	0.02
19	484.370	490.201	1.20	485.126	0.16	484.492	0.03
20	553.714	556.974	0.59	554.866	0.21	553.766	0.01

Table 5.5. Comparison of Initial and Converged CMS Frequencies with those by Full-System Eigensolution for Example 2 - Case 2

Mode	Reference Frequencies (Hz)	Initial Frequencies (Hz)	Difference from Reference (%)	(i) Converged Frequencies (Hz)	Difference from Reference (%)	(ii) Converged Frequencies (Hz)	Difference from Reference (%)
1	1.51202	1.51203	0.00	1.51203	0.00	1.51203	0.00
2	12.4811	12.4812	0.00	12.4811	0.00	12.4811	0.00
3	22.6685	22.6695	0.00	22.6688	0.00	22.6686	0.00
4	23.8279	23.8283	0.00	23.8280	0.00	23.8279	0.00
5	39.2530	39.2558	0.01	39.2535	0.00	39.2531	0.00
6	55.5810	55.5902	0.02	55.5823	0.00	55.5813	0.00
7	58.4602	58.4720	0.02	58.4625	0.00	58.4605	0.00
8	87.9478	87.9625	0.02	87.9511	0.00	87.9484	0.00
9	99.6708	99.7298	0.06	99.6789	0.01	99.6729	0.00
10	151.506	151.554	0.03	151.517	0.01	151.509	0.00
11	173.798	173.962	0.09	173.825	0.02	173.804	0.00
12	185.334	185.505	0.09	185.442	0.06	185.352	0.01
13	224.687	224.838	0.07	224.712	0.01	224.695	0.00
14	277.029	277.246	0.08	277.062	0.01	277.039	0.00
15	311.719	312.264	0.17	311.780	0.02	311.739	0.01
16	352.180	358.336	1.75	353.167	0.28	352.307	0.04
17	370.908	375.150	1.14	371.870	0.26	371.024	0.03
18	375.088	377.340	0.60	375.271	0.05	375.121[a]	0.01
19	424.880	426.184	0.31	425.000	0.03	424.937	0.01
20	463.394	469.617	1.34	464.213	0.18	463.541	0.03

[a] Iteration aborted

Table 5.6. Comparison of Initial and Converged CMS Frequencies with those by Full-System Eigensolution for Example 2 - Case 3

Mode	Reference Frequencies (Hz)	Initial Frequencies (Hz)	Difference from Reference (%)	(i) Converged Frequencies (Hz)	Difference from Reference (%)	(ii) Converged Frequencies (Hz)	Difference from Reference (%)	(iii) Two-Step (Modified) (Hz)	Difference from Reference (%)
1	5.20377	5.20377	0.00	5.20377	0.00	5.20377	0.00	5.20377	0.00
2	14.3477	14.3477	0.00	14.3477	0.00	14.3477	0.00	14.3477	0.00
3	28.1372	28.1375	0.00	28.1373	0.00	28.1372	0.00	28.1372	0.00
4	46.5345	46.5363	0.00	46.5355	0.00	46.5346	0.00	46.5346	0.00
5	69.5597	69.5635	0.01	69.5604	0.00	69.5598	0.00	69.5598	0.00
6	78.0993	78.1364	0.05	78.1075	0.01	78.1004	0.00	78.1004	0.00
7	97.2227	97.2393	0.02	97.2318	0.01	97.2232	0.00	97.2232	0.00
8	129.561	129.595	0.03	129.567	0.00	129.562	0.00	129.562	0.00
9	166.574	166.647	0.04	166.609	0.02	166.576	0.00	166.576	0.00
10	207.528	207.602	0.04	207.541	0.01	207.530	0.00	207.530	0.00
11	208.320	208.477	0.08	208.372	0.03	208.324	0.00	208.324	0.00
12	215.002	215.171	0.08	215.160	0.07	215.023	0.01	215.023	0.01
13	254.812	255.169	0.14	254.903	0.04	254.820	0.00	254.820	0.00
14	306.075	306.403	0.11	306.271	0.06	306.083	0.00	306.083	0.00
15	362.208	363.892	0.47	362.474	0.07	362.250	0.01	362.253	0.01
16	415.162	422.019	1.65	416.433	0.31	415.330	0.04	415.368	0.05
17	420.689	423.660	0.71	422.041[a]	0.32	420.788	0.02	421.646	0.23
18	423.139	438.073	3.53	423.623	0.11	423.142	0.00	423.142	0.00
19	489.074	494.900	1.19	489.813	0.15	489.189	0.02	489.257	0.04
20	559.892	563.127	0.94	561.029	0.20	559.940	0.01	559.964	0.01

[a] Iteration aborted

Table 5.7. Comparison of Initial and Converged CMS Frequencies with those by Full-System Eigensolution for Example 2 - Case 4

Mode	Reference Frequencies (Hz)	Initial Frequencies (Hz)	Difference from Reference (%)	(i) Converged Frequencies (Hz)	Difference from Reference (%)	(ii) Converged Frequencies (Hz)	Difference from Reference (%)	(iii) Two-Step (Modified) (Hz)	Difference from Reference (%)
1	1.52642	1.52642	0.00	1.52642	0.00	1.52642	0.00	1.52642	0.00
2	12.5076	12.5076	0.00	12.5076	0.00	12.5076	0.00	12.5076	0.00
3	22.7396	22.7407	0.00	22.7399	0.00	22.7397	0.00	22.7397	0.00
4	23.9039	23.9042	0.00	23.9040	0.00	23.9039	0.00	23.9039	0.00
5	39.3884	39.3911	0.01	39.3889	0.00	39.3886	0.00	39.3886	0.00
6	55.9891	55.9982	0.02	55.9904	0.00	55.9894	0.00	55.9894	0.00
7	63.5989	63.6130	0.02	63.6016	0.00	63.5993	0.00	63.5993	0.00
8	88.1008	88.1150	0.02	88.1041	0.00	88.1016	0.00	88.1016	0.00
9	100.265	100.323	0.06	100.273	0.01	100.267	0.00	100.267	0.00
10	152.045	152.091	0.03	152.055	0.01	152.048	0.00	152.048	0.00
11	174.586	174.746	0.09	174.612	0.01	174.594	0.00	174.594	0.00
12	185.010	185.185	0.09	185.120	0.06	185.029	0.01	185.029	0.01
13	225.879	226.026	0.07	225.903	0.01	225.887	0.00	225.887	0.00
14	278.941	279.153	0.08	278.974	0.01	278.952	0.00	278.952	0.00
15	314.067	314.586	0.17	314.125	0.02	314.089	0.01	314.090	0.01
16	351.198	357.362	1.76	352.202	0.29	351.333	0.04	351.423	0.06
17	377.366	381.186	1.01	378.309	0.25	377.484	0.03	377.537	0.05
18	380.677	382.892	0.58	380.841[a]	0.04	380.709[a]	0.01	380.747	0.02
19	428.739	429.989	0.29	428.853	0.03	428.796	0.01	428.800	0.01
20	470.861	477.534	1.42	471.668	0.17	471.034	0.04	471.316	0.10

[a] Iteration aborted

maximum difference was 4.1% and for the approximate $x\omega_c$ frequency ratios of two to one between the additional modes and the cut-off frequency, the maximum difference was 0.32%. When this ratio was approximately four to one, the maximum difference was 0.05%.

Some initial and converged mode shapes (with about four to one ratio of additional mode frequency limit to the cut-off frequency) are shown in Figures 5.2 through 5.9. Figure 5.2 shows the initial and converged mode shapes for mode number 18 for the first case, which is a bending mode about the axis about which the cross-sectional moment of inertia is greater. Notice the difference between the initial and converged mode shapes. Figure 5.3 shows a bending mode about the axis about which the cross-sectional moment of inertia is smaller. Figure 5.4 and 5.5 show bending modes about different axes for the second case. Notice the effect of the flexible connector in the response of the system. Figure 5.6 shows a torsional mode for the third case, in which there is a pronounced difference between the initial and converged mode shapes. Figure 5.7 shows the bending mode for case 3, which is a counterpart of the bending mode shown in Figure 5.4 for case 1. Figure 5.8 shows a bending mode for the fourth case in which the effect of the flexible connecting component can be observed. This effect is also seen in the torsional mode shape shown in Figure 5.9.

A comparison of solution times for two-step CMS and conventional CMS using the Jacobi method and the subspace iteration method are reported in Table 5.8 for the four to one $x\omega_c$ frequency ratio between the additional and initial modes. For the first two cases, the two-step method performs in about 50 % to 60 % of the time required for conventional CMS using a Jacobi solution and in 20% to 25% of the time required using a subspace iteration solution. For the third and fourth cases, solution times are 74 % to 98 % of those for conventional CMS using a Jacobi solution and a subspace iteration solution. This is because of the large number of connection coordinates in the shell-element models requiring considerably more solution time for the complete eigensolution of the initial problem compared to beam-element models.

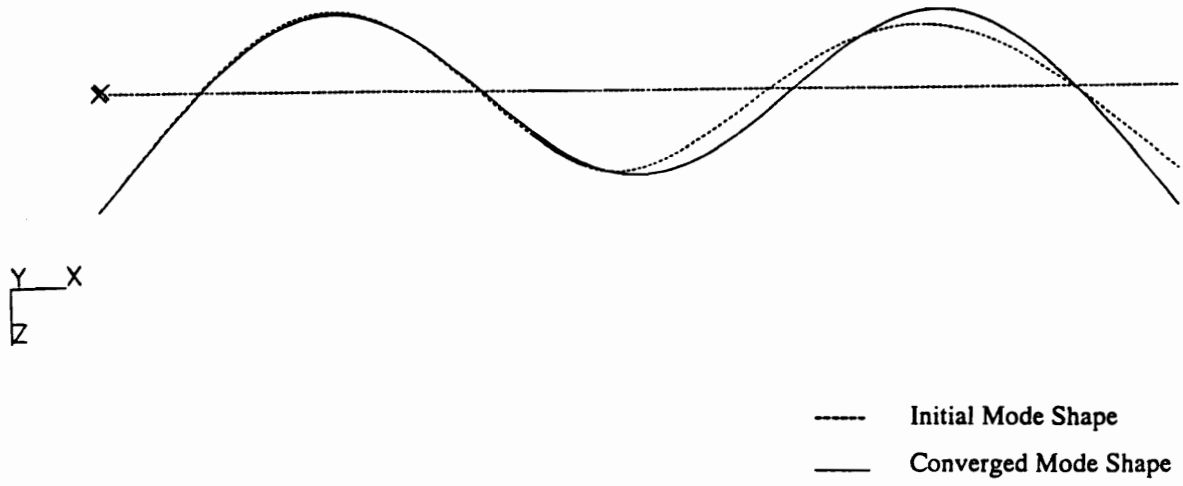


Figure 5.2. Initial and Final Mode Shapes for Example 2, Case 1, Mode 18

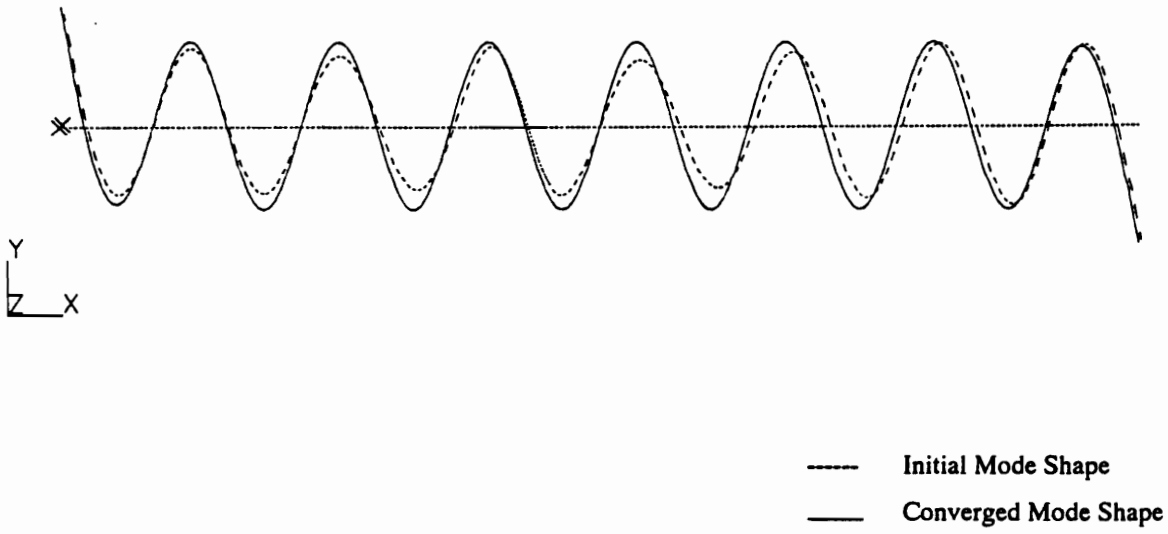


Figure 5.3. Initial and Final Mode Shapes for Example 2, Case 1, Mode 19

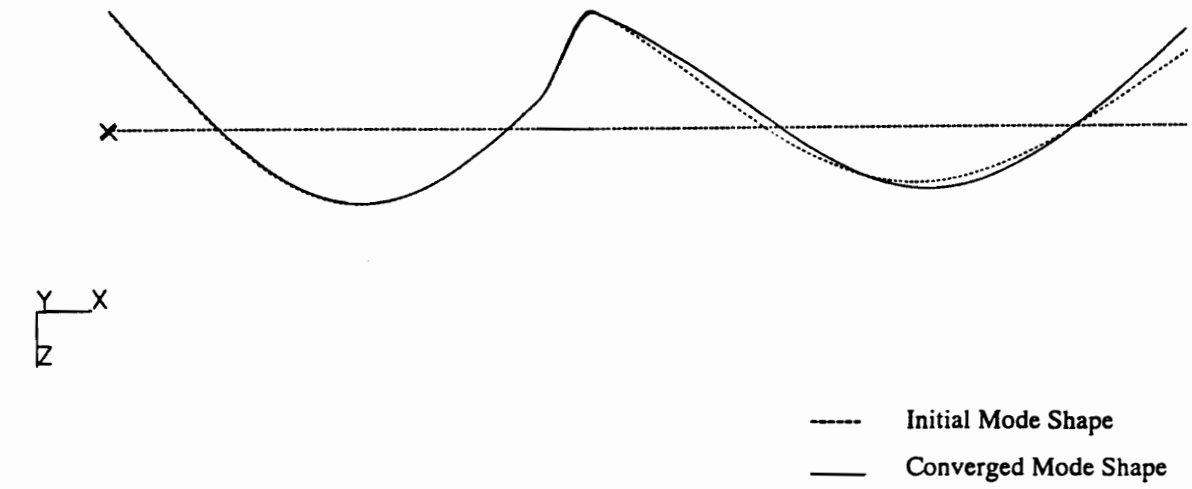


Figure 5.4. Initial and Final Mode Shapes for Example 2, Case 2, Mode 16

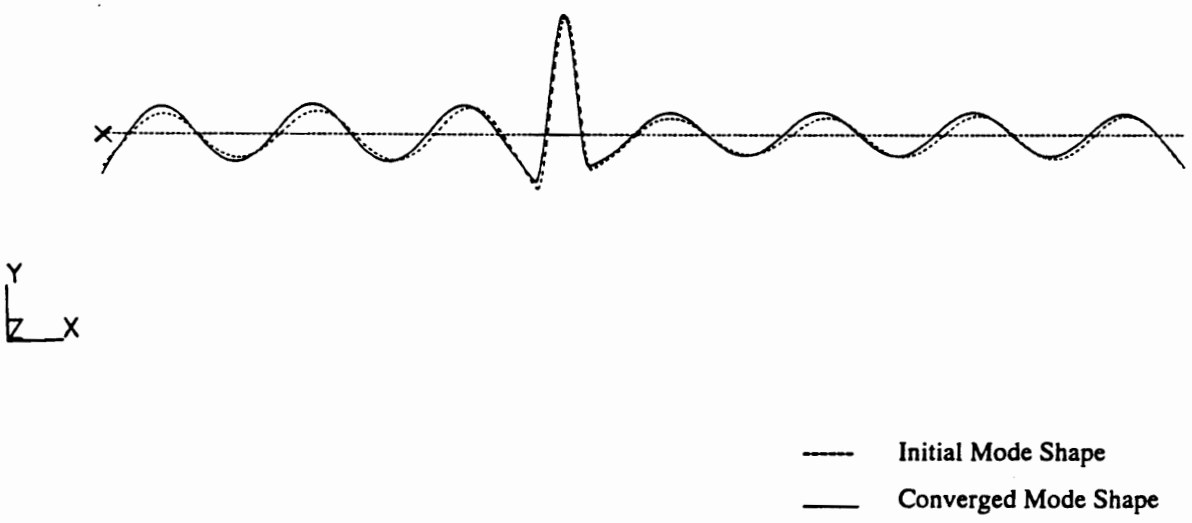


Figure 5.5. Initial and Final Mode Shapes for Example 2, Case 2, Mode 20

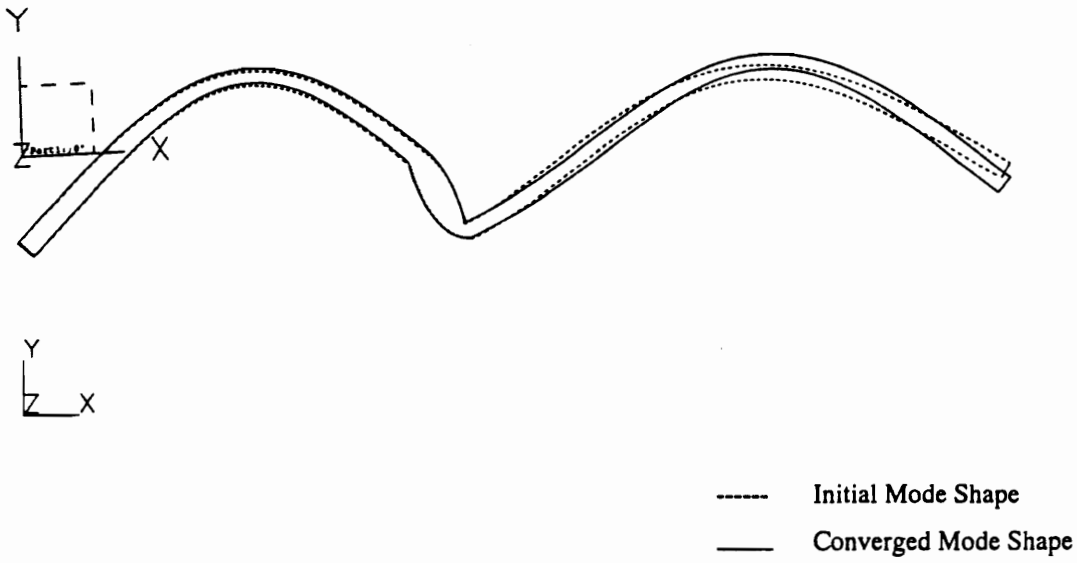


Figure 5.8. Initial and Final Mode Shapes for Example 2, Case 4, Mode 16

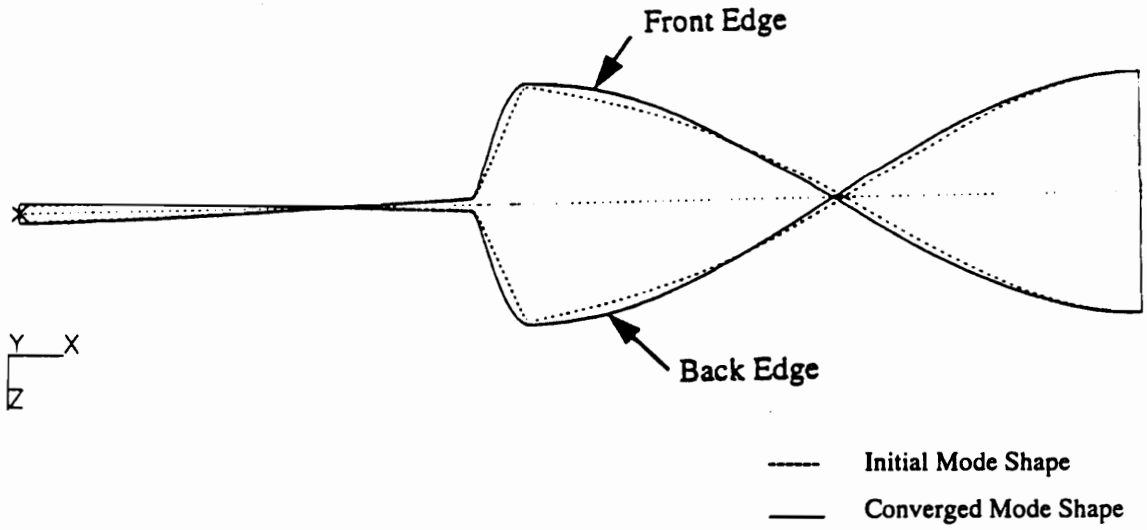


Figure 5.9. Initial and Final Mode Shapes for Example 2, Case 4, Mode 17

Table 5.8. Solution Times for Example 2

Case	Solution Time in One-Hundredth of a Second			
	Two-Step Method	Jacobi Method	Subspace Iteration Method	Two-Step Method (Modified)
1	295	500	1145	-
2	295	580	1500	-
3	7050	7142	8330	1678
4	6905	9302	8588	1786

Recall that the transformation equation (3.9) on page 27 uses all the modes of the initial eigenvalue problem as basis vectors and, therefore, necessitates complete eigensolution of the initial eigenproblem. However, in systems like these, wherein there is a large number of interface dofs, complete eigensolution of the initial problem is a time-consuming process. Instead, it is possible to solve the initial problem for only a few normal modes and use the resulting eigenvectors for transformation according to equation (3.9). Such a transformation results in a non-linear eigenvalue equation identical to equation (3.19) on page 30 but of order equal to the number of modes solved. Solution of this non-linear problem will result in converged eigenvalues whose accuracy depends on the number of modes used in the transformation.

Condensation reduction was employed for the third and fourth cases with a solution of the initial eigenvalue problem for thirty modes (of which six were rigid-body modes). Thus, the number of modes solved for in the initial eigenvalue problem exceeds the number of system modes required (including the rigid-body modes) only by four. This solution was obtained using the subspace iteration method. Results from using the two-step CMS method with this modification are reported in Tables 5.6 and 5.7 for the third and fourth cases respectively under the column 'Two-Step Method (Modified)', for the frequency ratio of about four to one between the additional mode frequency limit to the cut-off frequency. It is observed that there is excellent agreement with using the two-step method with complete eigensolution of the initial eigenvalue problem, with a maximum difference of only 0.23% from the reference solution. It is also observed that the differences between the two-step method and the reference solution are practically identical to the differences between the two-step method with partial eigensolution of the initial eigenvalue problem and the reference solution. Table 5.8 lists solution times when the

two-step method is used with partial solution of the initial eigenvalue problem for the third and fourth cases. These are reported under the column labeled 'Two-Step Method (Modified)'. It is observed that more than 75 % reduction in solution time is obtained over using the two-step method with complete eigensolution of the initial problem.

5.3 Example 3 - Synthesis of a Rectangular Plate

A 1 m by 25 cm plate was considered for the third example. Two thickness values were chosen for the study, viz. 1 mm and 4 mm. The plate was divided into two components across the longer dimension, the first measuring 40 cm by 25 cm and the second measuring 60 cm by 25 cm. Component 1 had a 12 x 20 shell-element mesh and Component 2 had a 12 x 30 element mesh (with twelve elements used across the width). An elastic modulus of 2×10^{11} Pa, Poisson's ratio of 0.3 and a mass density of 7800 kg/m^3 were used for all the components. For the 1 mm thick plate, two different cases were studied - (1) free-free boundary conditions and (2) cantilevered plate with a clamped boundary condition on one of the edges representing the width of the plate. The analysis details are similar to that for the second example with frequencies desired up to 50 Hz. For the free-free case, a total of ten component modes were used for the initial solution, with the highest frequency being 70.32 Hz. Eight total component modes were used for the initial solution for the cantilevered case, with the same value of the highest frequency. Again, two different $x\omega_c$ frequencies for the additional modes were studied. For the free-free case, the $x\omega_c$ frequencies considered were 98.58 Hz and 202.5 Hz, while for the cantilevered case, the $x\omega_c$ frequencies considered were 94.86 Hz and 202.5 Hz. These represent about twice and four times the cut-off frequency within which eigenpairs are to be determined.

The initial and converged frequencies are reported in Tables 5.9 and 5.10 along with the reference frequencies for the free-free and cantilevered cases respectively. Again, for verification, conventional CMS using the Jacobi method and the subspace iteration method were performed and the frequencies thus obtained were almost identical (maximum difference - 0.06%). From Tables 5.9 and 5.10, it can be observed that the maximum difference between the

Table 5.9. Comparison of Initial and Converged CMS Frequencies with those by Full-System Eigensolution for Example 3 - Free-Free Case

Mode	Reference Frequencies (Hz)	Initial Frequencies (Hz)	Difference from Reference (%)	(i) Converged Frequencies (Hz)	Difference from Reference (%)	(ii) Converged Frequencies (Hz)	Difference from Reference (%)	(iii) Two-Step (Modified) (Hz)	Difference from Reference (%)
1	5.22284	5.22347	0.01	5.22297	0.00	5.22288	0.00	5.22288	0.00
2	12.7181	12.7188	0.01	12.7188	0.01	12.7183	0.00	12.7183	0.00
3	14.5101	14.5221	0.08	14.5119	0.01	14.5106	0.00	14.5106	0.00
4	26.3860	26.4274	0.16	26.4274	0.16	26.3921	0.02	26.3921	0.02
5	28.6746	28.7760	0.35	28.6918	0.06	28.6788	0.01	28.6788	0.01
6	41.8809	42.0440	0.39	42.0440	0.39	41.9081	0.06	41.9081	0.06
7	47.7490	49.0657	2.76	47.9336[a]	0.39	47.8173[a]	0.14	47.8174[a]	0.14

[a] Iteration aborted

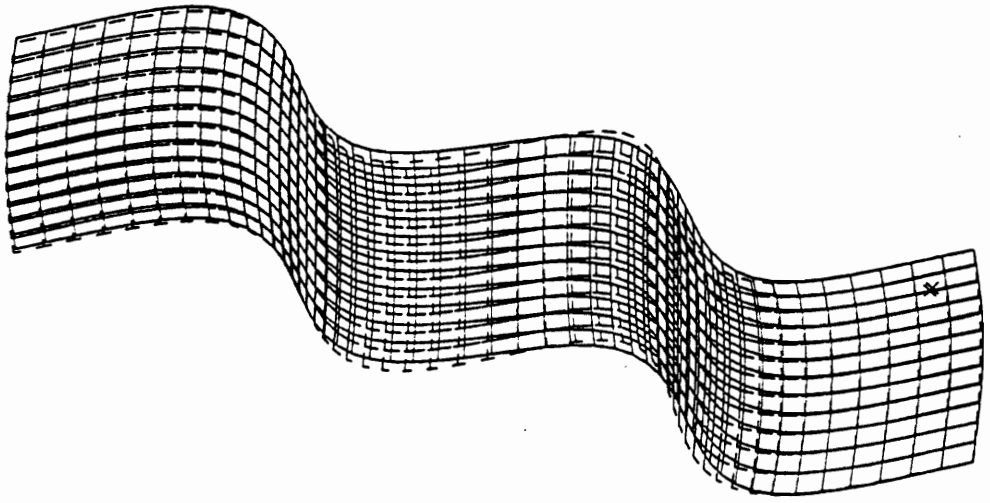
Table 5.10. Comparison of Initial and Converged CMS Frequencies with those by Full-System Eigensolution for Example 3 - Cantilevered Case

Mode	Reference Frequencies (Hz)	Initial Frequencies (Hz)	Difference from Reference (%)	(i) Converged Frequencies (Hz)	Difference from Reference (%)	(ii) Converged Frequencies (Hz)	Difference from Reference (%)	(iii) Two-Step (Modified) (Hz)	Difference from Reference (%)
1	0 .83056	0.83056	0.00	0.83056	0.00	0.83056	0.00	0.83056	0.00
2	5.20042	5.20142	0.02	5.20062	0.00	5.20048	0.00	5.20048	0.00
3	6.70874	6.70908	0.01	6.70908	0.01	6.70879	0.00	6.70879	0.00
4	14.6275	14.6420	0.10	14.6297	0.02	14.6281	0.00	14.6281	0.00
5	20.7245	20.7600	0.17	20.7600	0.17	20.7299	0.03	20.7299	0.03
6	28.8586	28.9587	0.35	28.8752	0.06	28.8626	0.01	28.8626	0.01
7	36.4542	36.4931	0.11	36.4931[a]	0.11	36.4592	0.01	36.4592	0.01
8	48.0417	49.3550	2.73	48.2293[a]	0.39	48.1105[a]	0.14	48.1106[a]	0.14

[a] Iteration aborted

initial solution and the reference solution was 2.76% and after convergence with the approximate $x\omega_c$ frequency ratio of two to one between the additional modes and the cut-off frequency, the maximum difference was 0.39%. When this ratio was approximately four to one, the maximum difference was 0.14%.

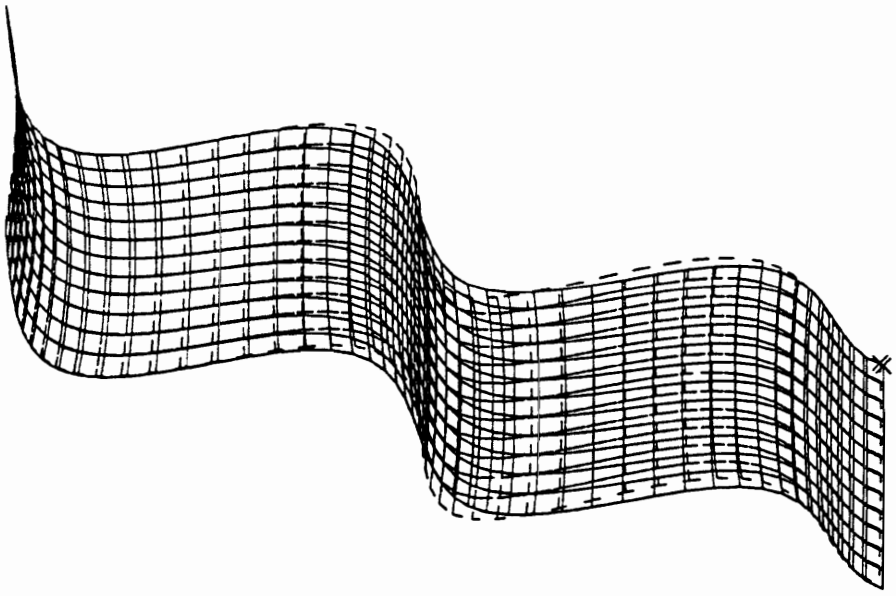
Some initial and converged mode shapes (with about four to one ratio of additional mode frequency limit to the cut-off frequency) are shown in Figures 5.10 and 5.11 for the free-free and cantilevered cases respectively.



------ Initial Mode Shape

____ Converged Mode Shape

Figure 5.10. Initial and Final Mode Shapes for Example 3, Free-Free Case, Mode 7



------ Initial Mode Shape

____ Converged Mode Shape

Figure 5.11. Initial and Final Mode Shapes for Example 3, Cantilevered Case, Mode 8

Table 5.11. Solution Times for Example 3

Case	Solution Time in One-Hundredth of a Second			
	Two-Step Method	Jacobi Method	Subspace Iteration Method	Two-Step Method (Modified)
Free-Free	1198	1739	1025	657
Cantilevered	1155	1635	634	476

Solution times are reported in Table 5.11. For the free-free case, the two-step method performs in about 69% of the time required for conventional CMS using a Jacobi solution and in 117% of the time required using a subspace iteration solution. For the cantilevered case, these values were 71% and 182% respectively. Again, in these cases, there is a large number of interface dofs, causing considerable time for the solution of the initial eigenvalue problem. Therefore, the two-step method was used with a partial eigensolution of the initial eigenvalue problem for seventeen modes for the free-free case and twelve modes for the cantilevered case. Thus, the number of modes solved for in the initial eigenvalue problem exceeds the number of system modes required by four. This solution was obtained using the subspace iteration method. Results from using the two-step CMS method with this modification are reported in Tables 5.9 and 5.10, for the ratio of about four to one between the additional modes and the cut-off frequency. Again, there is excellent agreement with using the two-step method with complete eigensolution of the initial eigenvalue problem, with a maximum difference of only 0.14% from the reference solution. It is also observed that the differences between the two-step method and the reference solution are practically identical to the differences between the two-step method with partial eigensolution of the initial eigenvalue problem and the reference solution. Table 5.11 also lists solution times when the two-step method is used with partial solution of the initial eigenvalue problem. About a 50% reduction in solution time is obtained over that with full solution of the initial problem.

For the 4 mm thickness case, only free-free boundary conditions were considered. The analysis details are similar to that for the previous cases, with three frequencies desired up to 70 Hz. A total of four component modes were used for the initial solution, with the highest frequency being 58.07 Hz. Again, two different $x\omega_c$ frequencies for the additional modes were studied -

150.1 and 393.5 Hz These represent about twice and five times the cut-off frequency within which eigenpairs are to be determined.

The initial and converged frequencies are reported in Table 5.12. Again, for verification, conventional CMS using the Jacobi method and the subspace iteration method were performed and the frequencies thus obtained were almost identical (maximum difference - 0.36 %). From Table 5.12, it can be observed that the maximum difference between the initial solution and the reference solution was 10.44 % and after convergence with the approximate $x\omega_c$ frequency ratio of two to one between the additional modes and the cut-off frequency, the maximum difference was 0.34 %. When this ratio was approximately five to one, the maximum difference was 0.37 %.

A sample initial and converged mode shape (with about five to one ratio of additional mode frequency limit to the cut-off frequency) is shown in Figure 5.12. Notice that there is a pronounced difference between the mode shapes, with the converged mode shape showing considerable smoothness.

The two-step method performs in about 88 % of the time required for conventional CMS using a Jacobi solution and in 275 % of the time required using a subspace iteration solution. Again, in these cases, there is a large number of interface dofs, causing considerable time for the solution of the initial eigenvalue problem. Therefore, the two-step method was used with a partial eigensolution of the initial eigenvalue problem for thirteen modes. Thus, the number of modes solved for in the initial eigenvalue problem exceeds the number of system modes required by four. This solution was obtained using the subspace iteration method. Results from using the two-step CMS method with this modification are reported in Table 5.12, for the ratio of about five to one between the additional modes and the cut-off frequency. Again, there is excellent agreement with results obtained by the two-step method with complete eigensolution of the initial eigenvalue problem, with no difference from the two-step method used without this modification. About a 60% reduction in solution time is obtained over that with full solution of the initial problem.

Table 5.12. Comparison of Initial and Converged CMS Frequencies with those by Full-System Eigensolution for the 4 mm Thick Plate - Free-Free Boundary

Mode	Reference Frequencies (Hz)	Initial Frequencies (Hz)	Difference from Reference (%)	(i) Converged Frequencies (Hz)	Difference from Reference (%)	(ii) Converged Frequencies (Hz)	Difference from Reference (%)	(iii) Two-Step (Modified) (Hz)	Difference from Reference (%)
1	20.8697	20.8952	0.12	20.8838	0.07	20.8703	0.00	20.8703	0.00
2	50.7714	56.0768	10.44	50.9489	0.34	50.9596[1]	0.37	50.9596	0.37
3	57.9747	59.6061	2.81	58.0747	0.17	58.0134[1]	0.07	58.0134	0.07

[1] Iteration aborted

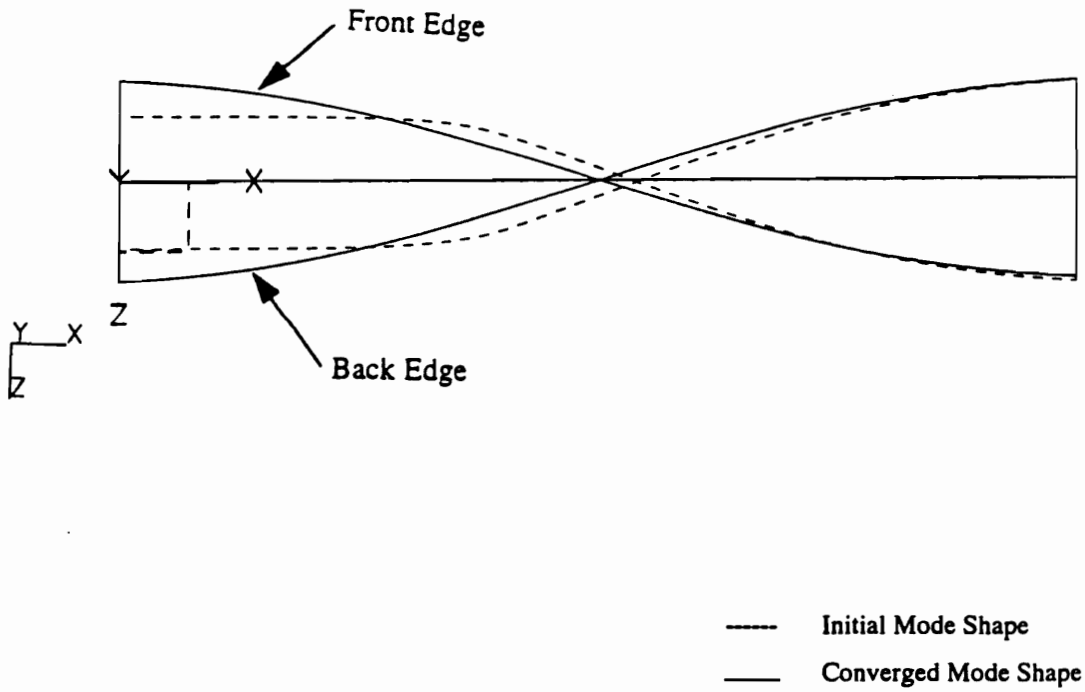
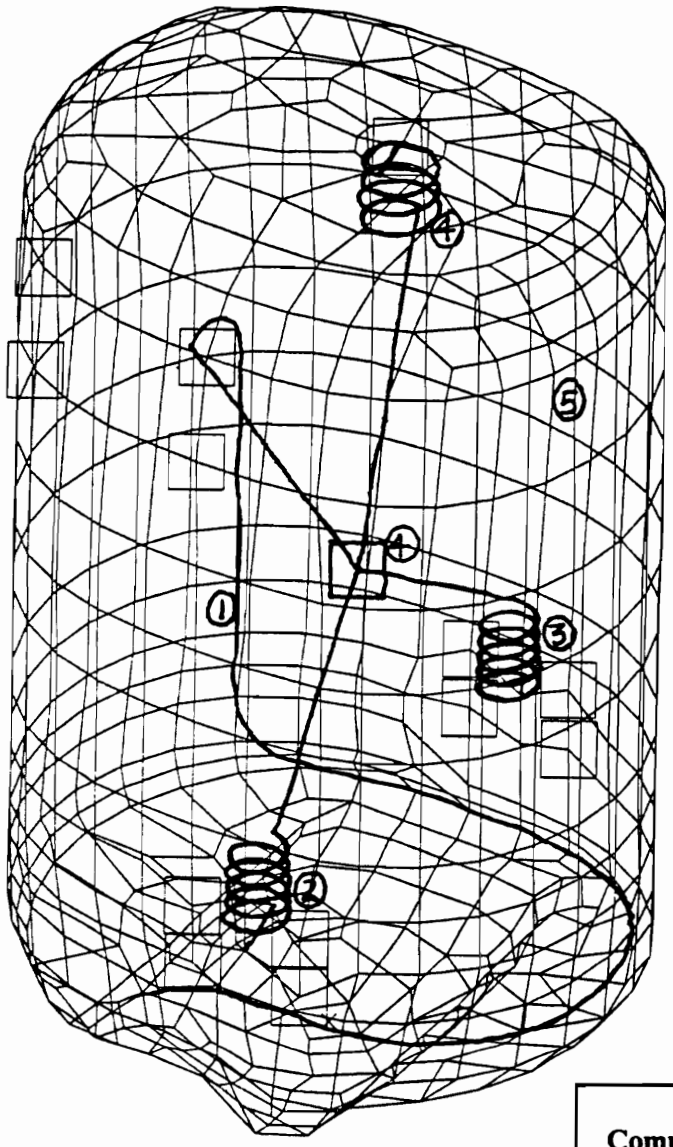


Figure 5.12. Initial and Final Mode Shapes for Example 3, 4 mm Thick Plate, Mode 2

5.4. Example 4 - Synthesis of a Compressor Assembly

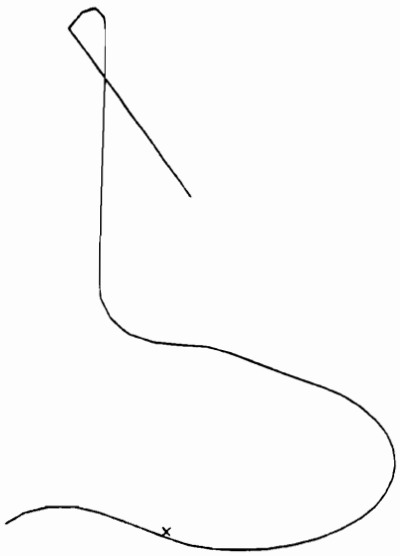
The fourth example involved the synthesis of a refrigeration compressor assembly whose finite element model is shown in Figure 5.13. The compressor was the subject of study for a noise-reduction program [54]. The compressor assembly consists of the compressor mechanism (crankcase, cylinders, pistons, crankshaft, stator, rotor, etc.) enclosed in a housing and mounted on suspension springs. The compressor mechanism was assumed to be a rigid body in the analysis. Figure 5.14 shows the compressor mechanism idealized as a point mass and connected to the housing through suspension springs. A top-spring, two side-springs and a shockloop discharge tube form the suspension system and are shown in Figure 5.13. The housing is made in two halves which are welded together to form a belt-line. Investigations of radiated sound revealed that the modes of the assembly up to about 1500 Hz played a significant role in sound radiation. This meant that accurate eigensolution of the compressor assembly up to 1500 Hz is necessary for acoustic predictions. Earlier studies on a similar compressor by Kelly [55] identified the existence of about sixty assembly natural frequencies for a similar compressor emphasizing the importance of a considerably refined finite element model of the entire assembly. This problem is a classical candidate for solution by CMS. Figure 5.13 shows the essential components of the system, as considered for the present analysis. Five components were identified - the housing, the shockloop, the two side-springs and the top-spring combined with the compressor mass and inertia. Normally, the top-spring and the compressor mass would be identified as separate components or only the top-spring would be modeled as a component with the compressor mass and inertia properties to be specified as interface mass loading during the coupling stage. But features of the code did not allow for either of these approaches; hence the present strategy was used.

Finite element models of the components were then constructed. These models are shown in Figure 5.14. The first component to be modeled was the shockloop discharge tube connecting the compressor mechanism to the housing. It is a hollow tube of circular section and, therefore, was idealized using beam elements. Ideally, a rigid link would be necessary to connect one end of the tube to the lumped-mass element at the center of mass of the compressor. However, the



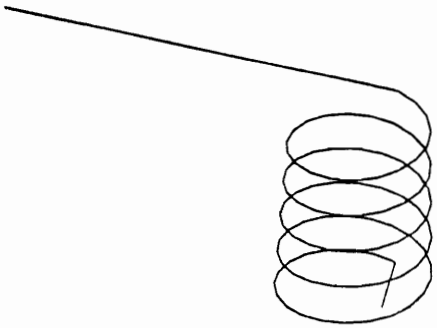
Component #	Description
1	Shockloop
2	First Side-spring
3	Second Side-spring
4	Top-spring and Compressor Mass
5	Housing

Figure 5.13. Finite Element Model of the System Used for Example 4



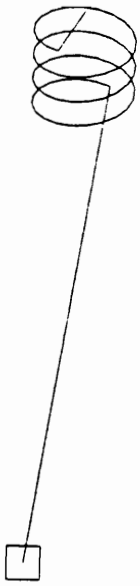
53 nodes
52 beam elements

Component 1 - Shockloop



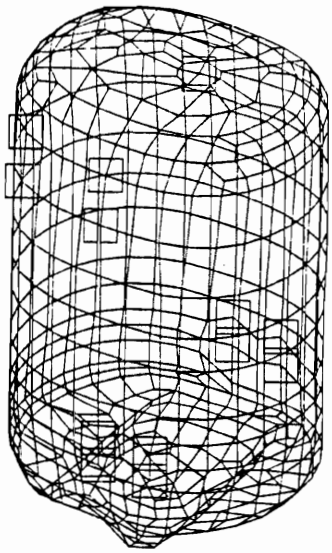
103 nodes
102 beam elements

Components 2 and 3 - Side-springs



73 nodes
72 beam elements and
1 lumped mass element

Component 4 - Top-spring and Compressor Mass



616 nodes
627 shell and lumped
mass elements

Component 5 - Housing

Figure 5.14. Components of the System Used for Example 4

present implementation of the code did not allow for a rigid link connecting non-coincident nodes. Hence, this connection was also modeled by using an artificially stiff beam element so that it closely approximates a rigid connection. The two side-springs were also modeled using beam elements so that internal spring resonances could be accounted for. This is in contrast to modeling for static analysis, where the entire spring would be modeled using a single spring element with the appropriate stiffness properties. Again, highly stiff beam elements were used to approximate rigid links. The top-spring with the compressor mass was similarly modeled. The lumped-mass element with the compressor mass and inertia properties has no stiffness associated with it. The values of mass, moments and products of inertia were determined from a detailed solid model by Ramani [54].

The housing was modeled using the shell elements described earlier. It is noted that automatic meshing on a curved surface such as the housing does not naturally generate elements with co-planar nodes. Efforts were made to minimize the deviation of the element nodes from a plane, but there was a need to use the best approximation of the flat shell element stiffness and mass matrices. For this, the nodes of the element were first projected to the plane of best fit passing through the centroid of the element and the element local coordinate system was defined using the projected node coordinates. The element stiffness and mass matrices were generated after redefining the element nodal coordinates in the local coordinate system thus defined.

The analysis details are similar to that for the previous two examples. All frequencies less than 1500 Hz were to be synthesized. Sixty six component modes were used for the initial solution with the highest frequency of component normal mode used being 2040 Hz. As before, two different $x\omega_c$ frequencies for the additional modes - (i) 3240 Hz, i.e., about twice the cut-off frequency and (ii) 5972 Hz, i.e., about four times the cut-off frequency - were studied. The initial and converged frequencies are reported in Table 5.13 along with the reference frequencies. It can be observed that the maximum difference between the initial solution and the reference solution was 2.31% and after convergence the maximum difference was about 1.49%, but most of the frequencies converged to much smaller errors.

Table 5.13. Comparison of Initial and Converged CMS Frequencies with those by Full-System Eigensolution for Example 4

Mode	Reference Frequencies (Hz)	Initial Frequencies (Hz)	Difference from Reference (%)	(i) Converged Frequencies (Hz)	Difference from Reference (%)	(ii) Converged Frequencies (Hz)	Difference from Reference (%)
1	7.32275	7.32263	0.00	7.32263	0.00	7.32263	0.00
2	8.40055	8.40058	0.00	8.40058	0.00	8.40058	0.00
3	10.9353	10.9346	0.00	10.9346	0.00	10.9346	0.00
4	12.6591	12.6589	0.00	12.6589	0.00	12.6589	0.00
5	16.8866	16.8864	0.00	16.8864	0.00	16.8864	0.00
6	31.5983	31.5988	0.00	31.5985	0.00	31.5984	0.00
7	85.9628	85.9716	0.01	85.9649	0.00	85.9631	0.00
8	91.2084	91.2172	0.01	91.2101	0.00	91.2087	0.00
9	94.2918	94.3025	0.01	94.2946	0.00	94.2920	0.00
10	128.687	128.687	0.00	128.687	0.00	128.687	0.00
11	210.714	210.715	0.00	210.715	0.00	210.715	0.00
12	225.185	225.186	0.00	225.186	0.00	225.186	0.00
13	287.976	287.976	0.00	287.976	0.00	287.976	0.00
14	306.585	306.585	0.00	306.585	0.00	306.585	0.00
15	386.176	386.178	0.00	386.176	0.00	386.176	0.00
16	387.478	387.479	0.00	387.478	0.00	387.478	0.00
17	405.423	405.425	0.00	405.424	0.00	405.423	0.00
18	406.340	406.347	0.00	406.341	0.00	406.340	0.00
19	453.288	453.290	0.00	453.288	0.00	453.288	0.00
20	506.957	506.959	0.00	506.958	0.00	506.957	0.00
21	516.692	516.705	0.00	516.693	0.00	516.692	0.00
22	581.224	581.235	0.00	581.228	0.00	581.226	0.00
23	642.593	642.596	0.00	642.594	0.00	642.594	0.00
24	667.515	667.616	0.02	667.530	0.00	667.516	0.00
25	668.094	668.167	0.01	668.109	0.00	668.096	0.00
26	699.610	699.615	0.00	699.612	0.00	699.611	0.00
27	713.669	713.800	0.02	713.700	0.00	713.672	0.00
28	714.312	714.420	0.02	714.338	0.00	714.314	0.00
29	768.539	769.301	0.10	768.742	0.03	768.559	0.00
30	770.131	770.193	0.01	770.142	0.00	770.132	0.00
31	817.683	820.565	0.35	818.182	0.06	817.729	0.01
32	879.229	881.750	0.29	879.529	0.03	879.257	0.00
33	901.463	901.551	0.01	901.498	0.00	901.469	0.00
34	907.731	907.788	0.01	907.744	0.00	907.731	0.00
35	936.740	936.764	0.00	936.748	0.00	936.742	0.00
36	951.529	951.903	0.04	951.574	0.00	951.534	0.00
37	954.002	954.063	0.01	954.013	0.00	954.003	0.00
38	983.384	983.533	0.02	983.407	0.00	983.390	0.00
39	1002.75	1003.04	0.03	1002.78	0.00	1002.75	0.00
40	1049.41	1049.61	0.02	1049.49	0.01	1049.42	0.00
41	1111.19	1123.14	1.08	1112.94	0.15	1111.73	0.05
42	1152.99	1156.07	0.27	1153.69	0.06	1153.34	0.03
43	1210.03	1221.32	0.93	1210.86	0.06	1210.13	0.01
44	1227.77	1256.09	2.31	1233.03	0.43	1227.67	-0.01
45	1245.75	1259.51	1.10	1251.22[a]	0.44	1250.05[a]	0.34
46	1255.06	1266.79	0.93	1249.15[a]	-0.47	1247.25[a]	-0.62
47	1278.35	1288.57	0.80	1262.49[a]	-1.24	1259.28[a]	-1.49
48	1304.62	1326.04	1.64	1304.79	0.01	1301.46[a]	-0.24
49	1313.82	1331.38	1.34	1317.23	0.26	1314.05	-0.02
50	1378.54	1380.09	0.11	1379.17	0.05	1378.62	0.01
51	1385.97	1396.64	0.77	1382.27[a]	-0.27	1380.18[a]	-0.42
52	1395.21	1407.64	0.89	1393.15[a]	-0.15	1393.64[a]	-0.11
53	1428.12	1430.51	0.17	1427.89[a]	-0.02	1427.09[a]	-0.07
54	1440.50	1441.05	0.04	1440.51[a]	-0.00	1440.37[a]	-0.01
55	1445.06	1446.86	0.12	1445.12	0.00	1444.68[a]	-0.03
56	1471.74	1476.29	0.31	1472.46	0.05	1471.82	0.01

[a] Iteration aborted

Some initial and converged mode shapes (with about four to one ratio of additional to initial mode frequency limits) are shown in Figures 5.15 and 5.16. In Figure 5.16, the suspension components are active, while in Figure 5.17, the housing motion is predominant.

In terms of solution time, the two-step method performs in about 64% of the time required for conventional CMS using a Jacobi solution and in 32% of the time required using a subspace iteration solution.

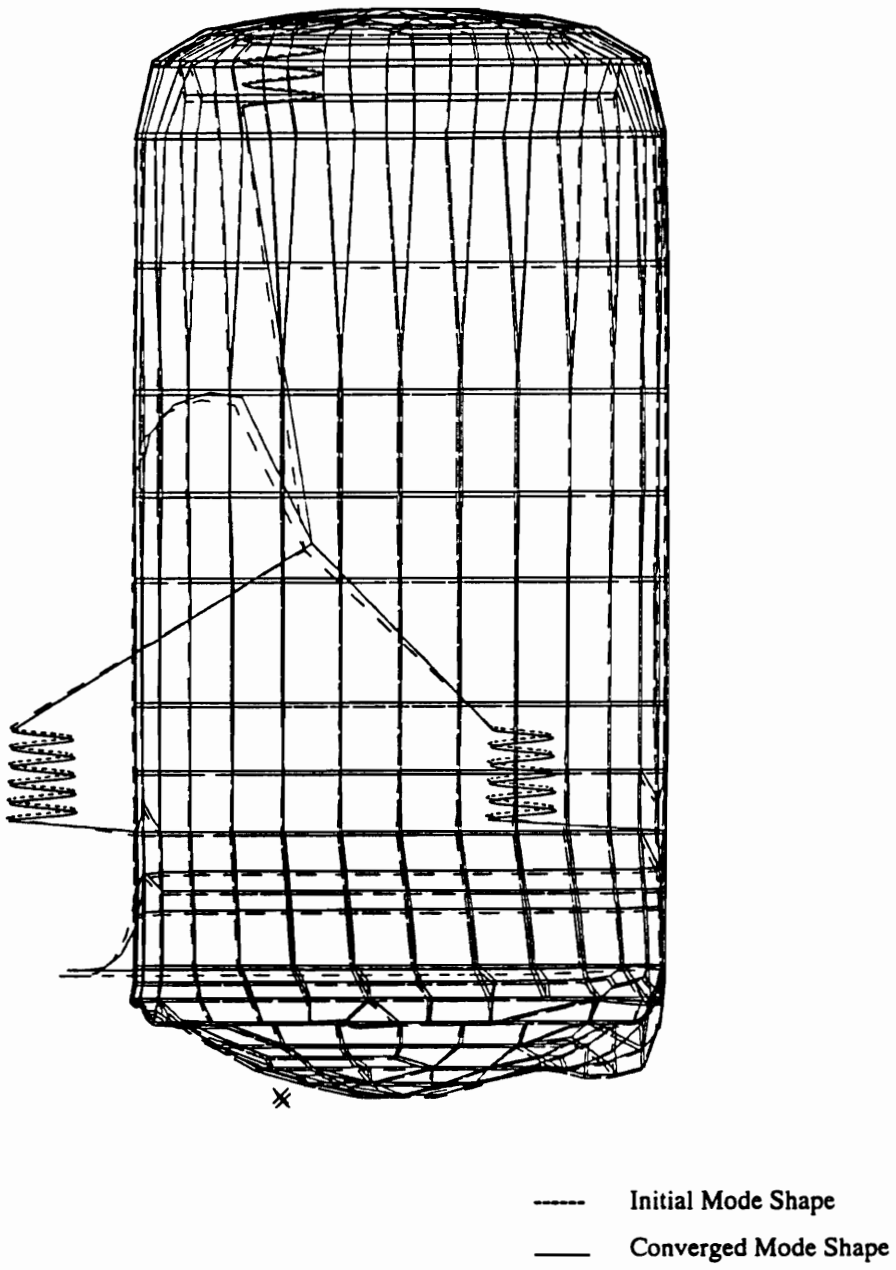


Figure 5.15. Initial and Final Mode Shapes for Example 4, Mode 48

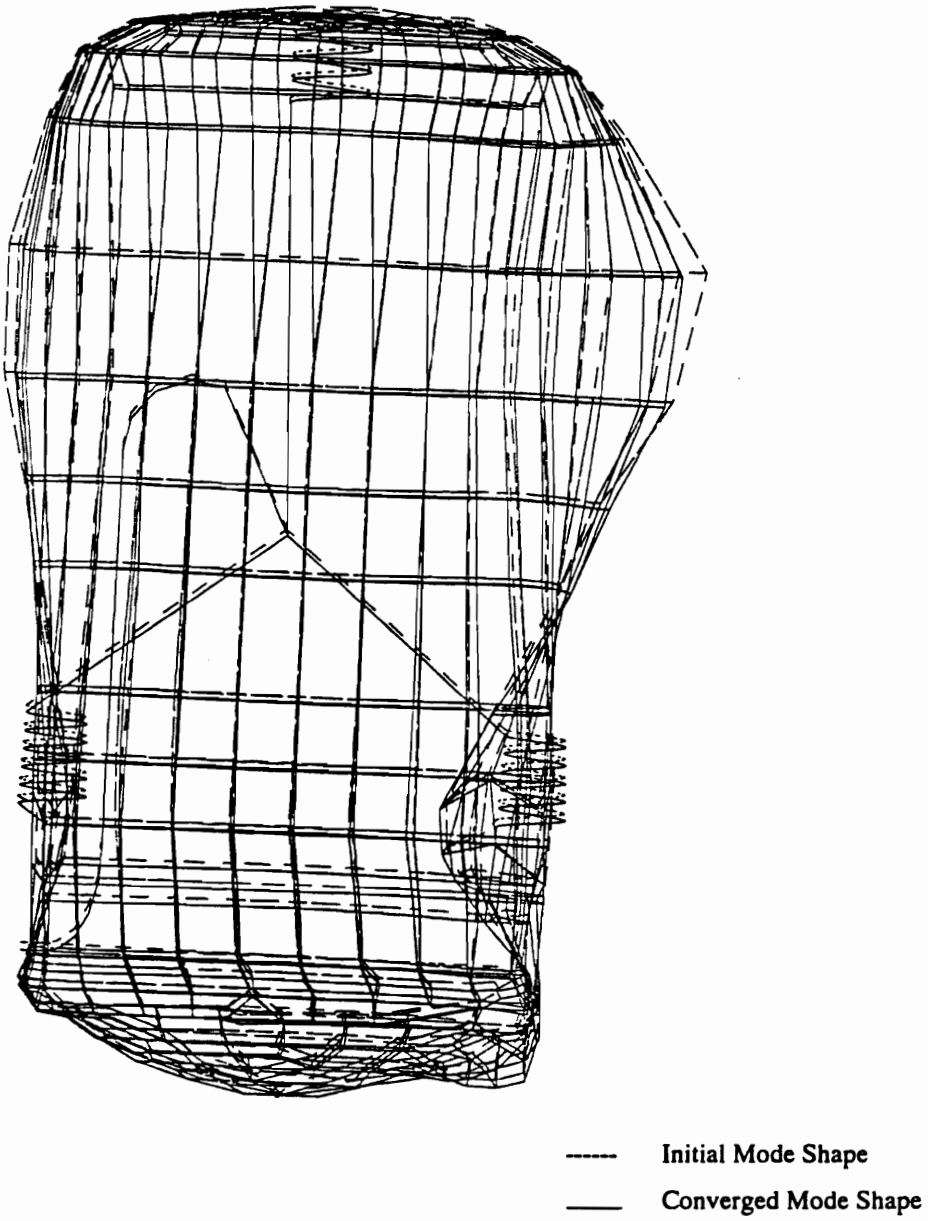


Figure 5.16. Initial and Final Mode Shapes for Example 4, Mode 51

6.0 Synthesis of Damped Systems

This chapter examines the applicability of the method for the synthesis of generally damped systems. Component mode synthesis techniques have been developed for damped systems for which the assumption of proportional damping is inappropriate. A survey of literature on CMS for damped systems has already been presented in Chapter 2. Similar to the method developed for undamped systems, the objective here is to develop a CMS method for generally damped systems such that the minimum number of component complex modes are used in system synthesis to obtain system complex eigenvalues up to a pre-selected cut-off frequency and available additional component modes are used in a convergence scheme operating on the calculated system eigenparameters. A numerical example demonstrates the working of the method and the results are compared with full-system eigensolution.

6.1 Component Modal Model

The equations of motion for a damped system subjected to time-varying excitation forces are given by equation (2.1) on page 8. These equations are true for any component in the system. When the component undergoes free vibrations, state-space formulation can be used to cast the component equations of motion in first order form as

$$\mathbf{a}\dot{\mathbf{y}} - \mathbf{b}\mathbf{y} = \mathbf{0}\,, \tag{6.1}$$

where

$$\mathbf{a} = \begin{bmatrix} \mathbf{0} & -\mathbf{m} \\ -\mathbf{m} & -\mathbf{c} \end{bmatrix}; \quad \mathbf{b} = \begin{bmatrix} -\mathbf{m} & \mathbf{0} \\ \mathbf{0} & \mathbf{k} \end{bmatrix}. \tag{6.2}$$

and $\mathbf{y}$ is the component dof vector which contains both displacements and velocities, i.e.,

$$\mathbf{y} = \begin{Bmatrix} \dot{\mathbf{x}} \\ \mathbf{x} \end{Bmatrix} \tag{6.3}$$

As described in Chapter 2, the set of component dof coordinates are partitioned into a set **i** of interior coordinates and a set **b** of boundary coordinates, with the partitions including both displacements and velocities. For the development of the method, there is no necessity to divide the boundary coordinates into rigid-body coordinates and redundant boundary coordinates as is usual for undamped systems.

Analogous to the definition of the constraint mode superset for undamped systems, the component constraint mode superset for a generally damped component is defined to enable the applicability of the synthesis and the convergence scheme. The following component modes are used in the constraint mode superset.

(i) Fixed-Interface Complex Modes: The fixed-interface complex modes, ϕ_n , are determined by solving the eigenvalue problem

$$\left(\begin{bmatrix} -\mathbf{m}_{ii} & \mathbf{0} \\ \mathbf{0} & \mathbf{k}_{ii} \end{bmatrix} - \lambda \begin{bmatrix} \mathbf{0} & -\mathbf{m}_{ii} \\ -\mathbf{m}_{ii} & -\mathbf{c}_{ii} \end{bmatrix} \right) \phi = \mathbf{0}, \tag{6.4}$$

where the subscript 'i' denotes the interior dofs, λ is the complex eigenvalue and ϕ is the complex eigenvector corresponding to the eigenvalue λ. The eigenvectors can be updated to include the boundary coordinates and arranged column-wise to form the complex modal matrix ϕ_n with 'n' normal modes as

$$\phi_n = \begin{Bmatrix} \phi_{2i,n} \\ \mathbf{0}_{2b,n} \end{Bmatrix}. \tag{6.5}$$

where the subscript 'b' denotes the boundary dofs. (There are 2i and 2b interior and boundary coordinates respectively because the dof vector includes both displacements and velocities.) The matrix ϕ_n has the following properties.

$$\begin{aligned} &\phi_n{}^T \mathbf{a}\, \phi_n = \mathbf{I}_{nn}\,; \\ &\phi_n{}^T \mathbf{b}\, \phi_n = \Lambda_{nn}\,, \end{aligned} \tag{6.6}$$

where Λ_{nn} is the diagonal matrix of the complex eigenvalues.

When the number of complex component complex modes determined is truncated to 'k' kept modes, the component truncated complex normal mode set is denoted by ϕ_k.

(ii) Constraint Modes: The constraint modes are defined in a special way so that the system matrices are obtained in a form from which the convergence scheme can be derived. The constraint modes are defined analogous to those defined using the stiffness matrix for the undamped system synthesis and are obtained by solving

$$\begin{bmatrix} -\mathbf{m}_{ii} & \mathbf{0} & -\mathbf{m}_{ib} & \mathbf{0} \\ \mathbf{0} & \mathbf{k}_{ii} & \mathbf{0} & \mathbf{k}_{ib} \\ -\mathbf{m}_{bi} & \mathbf{0} & -\mathbf{m}_{bb} & \mathbf{0} \\ \mathbf{0} & \mathbf{k}_{bi} & \mathbf{0} & \mathbf{k}_{bb} \end{bmatrix} \begin{Bmatrix} \phi_{2i,2b} \\ \mathbf{I}_{2b,2b} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{R}_{2i,2b} \end{Bmatrix} \tag{6.7}$$

where $\mathbf{R}_{2i,2b}$ are the terms denoting the reactions. Then, the set of constraint modes ϕ_c, is given by

$$\phi_c = \begin{Bmatrix} \phi_{2i,2b} \\ \mathbf{I}_{2b,2b} \end{Bmatrix}. \tag{6.8}$$

The constraint mode superset ψ is now defined to include the constraint modes and the fixed-interface complex normal modes, i.e.,

$$\psi = [\phi_c \ \phi_k]. \tag{6.9}$$

As before, the set of component modes is used to transform the component dof coordinates **y** to component generalized coordinates p, i.e.,

$$\mathbf{y} = \psi \mathbf{p}. \tag{6.10}$$

The matrix ψ is used to form the component generalized matrices α and β as

$$\begin{aligned} \alpha &= \psi^T \mathbf{a} \, \psi \ ; \\ \beta &= \psi^T \mathbf{b} \, \psi \ . \end{aligned} \tag{6.11}$$

The equations of motion of the component in the **p** coordinates is

$$\alpha \, \dot{\mathbf{p}} - \beta \, \mathbf{p} = \mathbf{0} \, . \tag{6.12}$$

Denoting $\phi_i^T \mathbf{a} \, \phi_j$ by α_{ij} and $\phi_i^T \mathbf{b} \, \phi_j$ by β_{ij}, the component generalized matrices take the form

$$\alpha = \begin{bmatrix} \alpha_{2b,2b} & \alpha_{2b,k} \\ \alpha_{k,2b} & \Lambda_{kk} \end{bmatrix};$$

$$\beta = \begin{bmatrix} \beta_{2b,2b} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{kk} \end{bmatrix}. \tag{6.13}$$

6.2 Coupling of Components

The coupling of damped components essentially follows the same procedure as for undamped systems; however, some differences exist. For explanatory purposes, consider two components 'r' and 's' in a system. Component generalized matrices α^r and β^r for component 'r' can be obtained.

$$\mathbf{y}^r = \psi^r \mathbf{p}^r , \tag{6.14}$$

where the $\mathbf{p}^r$ are modal coordinates. Interface compatibility requires that

$$\mathbf{y}_{2b}{}^r = \mathbf{y}_{2b}{}^s . \tag{6.15}$$

Constraint equations of this type can be written in the form

$$\mathbf{C}' \tilde{\mathbf{y}} = \mathbf{0} , \tag{6.16}$$

where $\mathbf{C}'$ is a matrix of coefficients and

$$\tilde{\mathbf{y}} = \begin{Bmatrix} \mathbf{y}^r \\ \mathbf{y}^s \\ \vdots \end{Bmatrix} . \tag{6.17}$$

These equations can also be written in the form

$$\mathbf{C} \tilde{\mathbf{p}} = \mathbf{0} , \tag{6.18}$$

where

$$\tilde{\mathbf{p}} = \begin{Bmatrix} \mathbf{p}^r \\ \mathbf{p}^s \\ \vdots \end{Bmatrix}. \tag{6.19}$$

As before, the coordinates in $\tilde{\mathbf{p}}$ can be partitioned into dependent coordinates $\tilde{\mathbf{p}}_d$ and independent coordinates $\tilde{\mathbf{p}}_\ell$. Then the constraint equations take the form

$$\begin{bmatrix} \mathbf{C}_{dd} & \mathbf{C}_{d\ell} \end{bmatrix} \begin{Bmatrix} \tilde{\mathbf{p}}_d \\ \tilde{\mathbf{p}}_\ell \end{Bmatrix} = \mathbf{0}, \tag{6.20}$$

so that

$$\tilde{\mathbf{p}} = \mathbf{S}\,\mathbf{p}, \tag{6.21}$$

where

$$\mathbf{S} = \begin{bmatrix} -\mathbf{C}_{dd}^{-1}\mathbf{C}_{d\ell} \\ \mathbf{I}_{\ell\ell} \end{bmatrix} \tag{6.22}$$

and

$$\mathbf{p} = \tilde{\mathbf{p}}_\ell. \tag{6.23}$$

Let

$$\hat{\alpha} = \begin{bmatrix} \alpha^r & \mathbf{0} & \cdots \\ \mathbf{0} & \alpha^s & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix};$$

$$\hat{\beta} = \begin{bmatrix} \beta^r & 0 & \cdots \\ 0 & \beta^r & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}. \tag{6.24}$$

Then, the final system matrices are obtained as

$$\begin{aligned} \mathbf{A} &= \mathbf{S}^T \hat{\alpha}\, \mathbf{S}; \\ \mathbf{B} &= \mathbf{S}^T \hat{\beta}\, \mathbf{S}, \end{aligned} \tag{6.25}$$

so that

$$\mathbf{A}\dot{\mathbf{p}} - \mathbf{B}\mathbf{p} = \mathbf{0} \tag{6.26}$$

is the final form of the free-body system equations. Let the system dof vector be partitioned as

$$\mathbf{p} = \begin{Bmatrix} \mathbf{p}_c \\ \mathbf{p}_n \end{Bmatrix}. \tag{6.27}$$

where the subscript 'c' refers to connection coordinates and 'n' refers to normal mode coordinates. The system matrices can accordingly be partitioned as

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} \mathbf{A}_{cc} & \mathbf{A}_{cn} \\ \mathbf{A}_{nc} & \mathbf{I}_{nn} \end{bmatrix}; \\ \mathbf{B} &= \begin{bmatrix} \mathbf{B}_{cc} & \mathbf{0} \\ \mathbf{0} & \Lambda_{nn} \end{bmatrix}, \end{aligned} \tag{6.28}$$

where Λ_{nn} is a diagonal matrix of component eigenvalues and $\mathbf{I}_{nn}$ is the identity matrix of order 'n'.

The eigenvalue problem corresponding to equation (6.26) can be solved to determine the initial system eigenparameters. To converge the initial system eigenvalues and eigenvectors, a set of 'e' extra component normal modes are introduced. The resulting system matrices can be partitioned as

$$\mathbf{A}' = \begin{bmatrix} \mathbf{A}_{cc} & \mathbf{A}_{cn} & \mathbf{A}_{ce} \\ \mathbf{A}_{nc} & \mathbf{I}_{nn} & \mathbf{0} \\ \mathbf{A}_{ec} & \mathbf{0} & \mathbf{I}_{ee} \end{bmatrix};$$

$$\mathbf{B}' = \begin{bmatrix} \mathbf{B}_{cc} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Lambda_{nn} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Lambda_{ee} \end{bmatrix}, \tag{6.29}$$

where Λ_{ee} is a diagonal matrix of the extra component eigenvalues and $\mathbf{I}_{ee}$ is the identity matrix of order 'e'. The system dof vector is now given by

$$\mathbf{p}' = \begin{Bmatrix} \mathbf{p}_c \\ \mathbf{p}_n \\ \mathbf{p}_e \end{Bmatrix}, \tag{6.30}$$

where $\mathbf{p}_e$ is the system dof vector corresponding to the extra component modes.

6.3 Method of Solution

Let Φ be the matrix of all eigenvectors and Λ be the diagonal matrix of all eigenvalues of the initial system. As before, the transformation

$$\begin{Bmatrix} \mathbf{p}_c \\ \mathbf{p}_n \\ \mathbf{p}_e \end{Bmatrix} = \begin{bmatrix} \Phi & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{ee} \end{bmatrix} \begin{Bmatrix} \mathbf{q} \\ \mathbf{p}_e \end{Bmatrix}, \tag{6.31}$$

is used to obtain the system eigenvalue problem with the additional modes as

$$\left(\begin{bmatrix} \Lambda & \mathbf{0} \\ \mathbf{0} & \Lambda_{ee} \end{bmatrix} - \lambda \begin{bmatrix} \mathbf{I} & \mathbf{a}_{ce} \\ \mathbf{a}_{ec}^{T} & \mathbf{I}_{ee} \end{bmatrix} \right) \begin{Bmatrix} \mathbf{q} \\ \mathbf{p}_e \end{Bmatrix} = \mathbf{0}\,, \tag{6.32}$$

where

$$\mathbf{a}_{ce} = \Phi^{T} \begin{bmatrix} \mathbf{A}_{ce} \\ \mathbf{0}_{ne} \end{bmatrix}\,. \tag{6.33}$$

Solving for $\mathbf{p}_e$ from the lower partition and substituting in the upper partition results in

$$\mathbf{N}(\lambda)\,\mathbf{q} = \mathbf{0}\,, \tag{6.34}$$

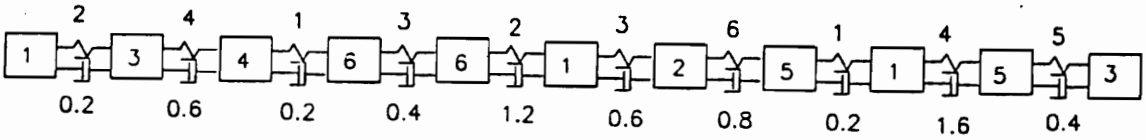
where

$$\mathrm{N}(\lambda) \equiv \mathbf{N} = \left[\Lambda - \lambda\,\mathbf{I} - \lambda^{2}\,\mathbf{a}_{ce}\left[\,\Lambda_{ee} - \lambda\,\mathbf{I}\right]^{-1}\mathbf{a}_{ce}^{T}\right]. \tag{6.35}$$

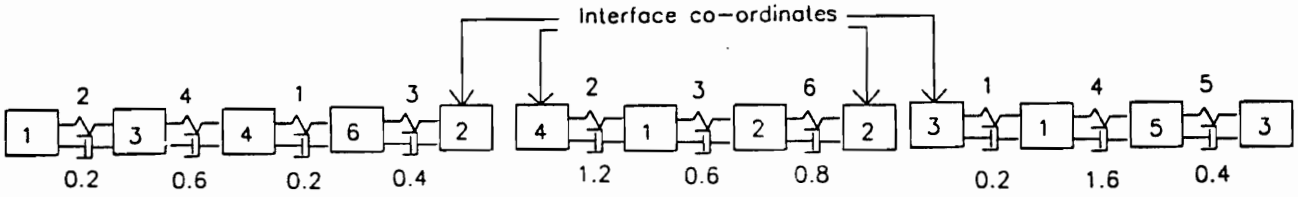
Equation (6.33) is similar to equation (3.19) on page 30, so that the same convergence scheme can be applied to this system of equations.

6.4 An Example

The method was used to synthesize and converge the eleven-dof system shown in Figure 6.1 with stiffness, mass and damping properties as indicated. The system was divided into three components as shown in the figure. First, state-space formulation was used to obtain the eigenvalues of the full system for comparison. The eigenvalues are of the form



System



Component 1 **Component 2** **Component 3**

Figure 6.1. System and Components used for the Example Problem

$$\lambda = -\omega_n \xi \pm i\omega_n \sqrt{1-\xi^2}\,, \tag{6.36}$$

where ω_n is the natural frequency and ξ is the modal damping ratio. The values of ω_n and ξ for the first four non-zero eigenvalues are reported in columns 1 and 2 of Table 6.1. Constraint modes and fixed-interface complex modes of each component were determined. The component natural frequencies and damping ratios are listed in Table 6.2.

Table 6.1. Results of CMS Solution for the Example Problem

Mode	Exact Solution		Initial Solution		Converged Solution	
	ω_n	ξ	ω_n (% Error)	ξ (% Error)	ω_n (% Error)	ξ (% Error)
1.	0.2385	0.03347	0.2394 (0.38)	0.03329 (−0.52)	0.2387 (0.11)	0.03341 (−0.14)
2.	0.4247	0.04338	0.4413 (3.89)	0.04568 (5.30)	0.4230 (0.41)	0.04349 (0.26)
3.	0.6175	0.09831	0.6699 (8.48)	0.09936 (1.06)	0.6237 (0.99)	0.09811 (−0.2)
4.	1.078	0.08459	1.083 (0.53)	0.08946 (5.75)	1.081 (0.32)	0.08908 (5.31)

Table 6.2. Eigenvalues of Components

Mode	Component					
	1		2		3	
	ω_n	ξ	ω_n	ξ	ω_n	ξ
1.	0.2911	0.0257	1.669	0.2311	0.3012	0.0348
2.	0.8234	0.0622	2.542	0.3400	1.5904	0.0720
3.	1.2904	0.0857			2.4107	0.4321
4.	1.8667	0.1147				

The initial system was synthesized using the first three pairs of complex conjugate normal modes of the first component and the first pair of complex conjugate modes of the third component. The cut-off frequency ω_n used in the synthesis was 1.290. The initial values of ω_n and ξ are reported in Table 6.1, with the percentage difference from the exact solution shown in parenthesis. For convergence, one additional mode from each component was used and about three iterations performed. The values of ω_n and ξ after convergence are also reported in Table 6.1 and the percentage difference from the exact values are shown in parenthesis. It is observed, that even with one additional mode from each component the system eigenparameters are well-converged, with the exception of the fourth eigenvalue. The method was programmed using the software MATLAB [56]. No time calculations were made.

7.0 Application to Non-Linear Eigenvalue Problems

The use of frequency-dependent matrices in structural dynamics leads to eigenvalue problems that are non-linear in the eigenvalue λ. This is in contrast to the classical linear eigenvalue problem which results when frequency-independent matrices are used. In general, the order of frequency-dependent matrices is smaller than that of frequency-independent matrices for comparable accuracy in the eigenparameters. In what follows, eigenvalue problems that can be described by the equation

$$\overline{\mathbf{N}}(\lambda)\,\overline{\mathbf{q}} = \mathbf{0}, \tag{7.1}$$

are considered, where $\overline{\mathbf{N}}(\lambda)$ is a symmetric matrix and is a function of λ. Further, let $\overline{\mathbf{N}}(\lambda)$ be such that it can be expanded in series form as

$$\overline{\mathbf{N}}(\lambda) = \overline{\mathbf{N}}_0 + \lambda\overline{\mathbf{N}}_1 + \lambda^2\overline{\mathbf{N}}_2 + ... + \lambda^r\overline{\mathbf{N}}_r\,. \tag{7.2}$$

where r may be either finite or infinite and the matrices $\overline{\mathbf{N}}_s$, s = 0,1,2,...r, are constant coefficient matrices. It is also assumed that the system is positive semi-definite and so is the basic linear system, i.e.,

$$[\,\overline{\mathbf{N}}_0 + \lambda\overline{\mathbf{N}}_1\,]\,\overline{\mathbf{q}} = \mathbf{0}. \tag{7.3}$$

7.1 The Method

To solve equation (7.1) for λ and $\bar{\mathbf{q}}$, the linear problem given by equation (7.3) is first solved for all the eigenvalues and eigenvectors and the eigenvectors are collected in a matrix Φ. Further, the eigenvectors are normalized such that

$$\Phi^T \bar{\mathbf{N}}_1 \Phi = -\mathbf{I}, \tag{7.4}$$

where $\mathbf{I}$ is the identity matrix. Introducing the transformation

$$\bar{\mathbf{q}} = \Phi\, \mathbf{q}, \tag{7.5}$$

substituting in equation (7.1) and pre-multiplying by Φ^T results in

$$\mathbf{N}(\lambda)\mathbf{q} = 0, \tag{7.6}$$

where

$$\mathbf{N}(\lambda) = \Phi^T \bar{\mathbf{N}}(\lambda) \Phi\,. \tag{7.7}$$

Substituting for $\bar{\mathbf{N}}(\lambda)$ from equation (7.2) in equation (7.7) and using the resulting expression for $\mathbf{N}(\lambda)$ in equation (7.6), the equation

$$[\, \mathbf{N}_0 + \lambda\, \mathbf{N}_1 + \lambda^2\, \mathbf{N}_2 + \ldots + \lambda^r \mathbf{N}_r \,]\, \mathbf{q} = \mathbf{0}\,, \tag{7.8}$$

is obtained, where

$$\mathbf{N}_s = \Phi^T \bar{\mathbf{N}}_s \Phi \;,\; s = 0,1,2,\ldots..r. \tag{7.9}$$

Note that

$$\mathbf{N}_0 = \Lambda, \tag{7.10}$$

where Λ is the diagonal matrix of initial eigenvalues of equation (7.3). Notice that equation (7.6) is equivalent to equation (7.1) and there are no approximations involved, since all the eigenvectors of the basic linear eigenvalue problem were used to transform the coefficient matrices. Equation (7.6) is identical to equation (3.19) on page 30, so that the convergence scheme developed in Chapter 3 can be used to obtain the eigenvalues of the non-linear eigenvalue problem (7.1) with the initial values being those resulting from the solution of the linear eigenvalue problem.

7.2 Condensation Reduction

The method described above may be cumbersome to implement if the order of the matrices is large and only the lowest few eigenvalues are desired, in which case, solving the linear eigenvalue problem for all its eigenvalues and eigenvectors may not be feasible. In such cases, the basic eigenvalue problem may be solved only for its lowest few eigenvalues. The number of such eigenvalues to be solved for depends on the final number of eigenvalues of the non-linear eigenvalue problem desired and cost considerations; in general, using more initial eigenvalues improves the accuracy of the final eigenparameters of the non-linear eigenproblem. Such a partial eigensolution of the linear problem can be obtained by methods such as subspace iteration and the Lanczos method. Then, the matrix Φ forms an incomplete set of eigenvectors of the basic linear eigenvalue problem. The transformation given by equation (7.5) will now result in equation (7.6), but with matrices of order equal to the number of eigenvalues of the linear eigenvalue problem. The convergence scheme can now be applied to the resulting system of equations.

7.3 Examples

The above method was used to solve the quadratic eigenvalue problem

$$(\overline{\mathbf{N}}_0+\lambda\overline{\mathbf{N}}_1+\lambda^2\overline{\mathbf{N}}_2)\overline{\mathbf{q}}=\mathbf{0}\,, \tag{7.12}$$

where $\overline{\mathbf{N}}_0 = \begin{bmatrix} 3 & -1 & 0 & 0 \\ -1 & 3 & -2 & 0 \\ 0 & -2 & 5 & -3 \\ 0 & 0 & -3 & 4 \end{bmatrix}$, $\overline{\mathbf{N}}_1 = \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & -4 \end{bmatrix}$,

and $\overline{\mathbf{N}}_2 = \begin{bmatrix} -0.1 & 0.04 & 0 & 0 \\ 0.04 & -0.12 & 0.08 & 0 \\ 0 & 0.08 & -0.1 & 0.02 \\ 0 & 0 & 0.02 & -0.06 \end{bmatrix}$.

Such an eigenvalue equation could result from the use of dynamic stiffness matrices, with the matrix $\overline{\mathbf{N}}_0$ representing the static stiffness matrix, the matrix $\overline{\mathbf{N}}_1$ representing the negative of the mass matrix and the matrix $\overline{\mathbf{N}}_2$ representing the dynamic stiffness matrix associated with the second power of the eigenvalue. The results are reported in Table 7.1.

The eigenvalues of the basic linear problem are reported in the first column; the second column shows the exact eigenvalues of the non-linear problem obtained by converging the initial values. These were obtained using the complete set of linear problem eigenvectors for transformation. The fourth eigenvalue experienced convergence problems and therefore could not be converged. Exact eigenvalues for this problem were obtained by forming block-companion matrices [46,47] and solving the resulting linear eigenvalue problem. The eigenvalues thus obtained are also reported in the second column. The third column shows the eigenvalues of the non-linear problem with only the eigenvectors corresponding to the lowest three eigenvalues of the linear

problem used for the transformation. Note the accuracy in eigenvalues obtained by the condensation algorithm.

Table 7.1. Results for Example 1

Mode	Linear Eigenvalues	Eigenvalues on Convergence (Exact Eigenvalues)	Eigenvalues by Condensation (% Difference from Exact)
1.	0.19068	0.19042 (0.19042)	0.19042 (0)
2.	1.1760	1.1333 (1.1133)	1.1333 (0.01)
3.	1.9112	1.7958 (1.7958)	1.7983 (0.14)
4.	3.8887	- (2.8002)	-

For the second example, consider the problem

$$(\overline{N}_0+\lambda\overline{N}_1+\lambda^2\overline{N}_2+....)\overline{q}=0 \tag{7.13}$$

where $\overline{N}_0 = \begin{bmatrix} 1.2405 & 0 & 0 & 0 & 0 \\ 0 & 0.4508 & 0 & 0 & 0 \\ 0 & 0 & 0.2530 & 0 & 0 \\ 0 & 0 & 0 & 0.0732 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, $\overline{N}_1 = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}$,

and $\overline{N}_s(i,j) = \sum_{r=1}^{3} \frac{-m(i,r)m(j,r)}{\Lambda(r,r)^{s-1}}$ $\quad s = 2,3,4,....$,

$$\text{where } \mathbf{m} = \begin{bmatrix} -0.1032 & 0.2426 & -0.0976 \\ 0.0180 & -0.0424 & 0.0171 \\ -0.0438 & 0.1028 & -0.0414 \\ 0.0158 & -0.0372 & 0.0150 \\ -0.0419 & 0.0985 & -0.0396 \end{bmatrix},$$

$$\text{and } \Lambda = \begin{bmatrix} 0.9628 & 0 & 0 \\ 0 & 2.1737 & 0 \\ 0 & 0 & 2.6145 \end{bmatrix}.$$

This is an infinite-series eigenvalue problem. This problem is obtained when two-step CMS is performed on the system shown in Example 1 in Chapter 5. The first two modes of each component are used in the initial synthesis along with the rigid-body modes. For convergence, one additional mode of the first component and two additional modes of the second component were used. The matrices given above are obtained during the CMS process. The matrix $\overline{N}_0$ is already in the diagonal form and $\overline{N}_1$ is the negative of the identity matrix. Hence, the initial eigenvalues (of the linear problem) are the diagonal elements of $\overline{N}_0$ and are listed in the first column of Table 7.2. Exact eigenvalues for a general infinite-series eigenvalue problem are difficult to determine, but because of the special nature of this infinite-series problem, exact solutions could be obtained. The eigenvalues resulting from the convergence scheme and the exact eigenvalues are reported in the second column. For the fifth eigenvalue, the series becomes divergent as the lowest diagonal element of the matrix Λ is greater than one of the initial eigenvalues. Hence, the convergence scheme cannot be used for this eigenvalue.

Thus, the method is successful in finding both the eigenvalues and eigenvectors. The method can be used as an alternative to the companion-matrix method so that the order of the matrices is small. The method also works successfully for infinite-series eigenvalue problems which cannot be solved using the companion-matrix method.

Table 7.2. Results for Example 2

Mode	Initial Eigenvalue	Final Eigenvalue (Exact Eigenvalue)
1.	0	0 (0)
2.	0.073192	0.073187 (0.073187)
3.	0.25297	0.25240 (0.25240)
4.	1.2405	- (0.93375)

8.0 Conclusions and Recommendations

A CMS method with a convergence scheme for the eigensolution of large dof systems has been developed and successfully implemented. The results of the examples presented in the previous chapters show that the method is a viable approach for the analysis of complex structures divisible into two or more components. In conclusion, the method offers the following advantages over other conventional methods.

(i) The method gives the analyst an idea of the accuracy of the synthesized eigenparameters by converging the initial eigenvalues using additional modes of components. There are two disadvantages if additional modes are not introduced in this kind of a convergence scheme but used along with other normal modes in a one-step eigensolution. One is that, we have to solve a larger eigenvalue problem and the other is that the accuracy of the calculated eigenvalues and eigenvectors is unknown.

(ii) The method is developed using constraint mode supersets of components. The resulting structure of system matrices is such that there is no need for reformulation and resolution of the larger eigenproblem when additional modes are introduced. Instead, a non-linear eigenvalue problem of the same order as the initial system is obtained and solved by the present iterative approach.

(iii) The method lends itself to easy computer implementation and was developed with an intention to minimize solution time. The iteration expressions for the eigenvalue and eigenvector were derived so that the equations for convergence do not involve eigenvalue-eigenvector coupling. As a result, the method does not involve expensive matrix decompositions and

solutions of a linear system of equations, which are currently used to solve the non-linear eigenvalue problem described by equation (3.19).

(iv) This method also suggests that for system eigensolution up to a frequency ω_c, we use component normal modes up to a frequency slightly greater than ω_c be used in the initial system eigensolution. Hence, the first eigenproblem is small, and with the introduction of additional modes, convergence can be studied. Convergence is not possible if the initial synthesis yields a system eigenvalue which is greater than the lowest of the additional modes introduced, as in that case, the series expansion of the matrix function of the eigenvalue is divergent. For this reason, the first eigenvalue problem is formulated including component modes up to the next frequency greater than the cut-off frequency.

(v) If, after using the proposed convergence scheme, it is found that some eigenvalues still do not satisfy the analyst's accuracy requirements, the convergence scheme can be re-used with still additional component modes.

(vi) Usually, the lowest frequencies are well converged. This scheme allows the analyst to select only the required eigenvalues for convergence.

(vii) The method is applicable to the solution of generally damped systems as illustrated in Chapter 6.

(viii) The method is applicable in the solution of positive semi-definite non-linear eigenvalue problems arising in structural dynamics. The method is an alternative to the companion-matrix method which is widely used to solve non-linear eigenvalue problems. A significant advantage of the method over the companion-matrix method is the preservation of bandwidth in large finite element models. Because of rearrangement of the coefficient matrices in the companion-matrix method, advantages of banded storage is lost and there is also an increase in the order of the system to be solved. However, the present method does not require rearrangement of the coefficient matrices, thus preserving bandwidth. Also, the order of the problem is not increased.

A critical factor in the development of the method was that, for accurate convergence, the initial eigenproblem had to be fully solved for all its eigenvalues and eigenvectors, so that a modal transformation using the eigenvectors as basis functions did not involve any approximations. In synthesis using components with a large number of interface coordinates compared to normal-mode coordinates, as in the case of the second example in Chapter 5, this procedure is not time-effective. Therefore, the technique of condensation reduction was successfully utilized to obtain convergence to practically the same eigenvalues in much less solution time. This broadens the applicability of the method to the synthesis of systems where a large number of interface dofs of components is not a concern.

At this point, it is worthwhile to make some suggestions for future work. Chief among those is the selection of modes for convergence. In the current implementation, all additionally evaluated component modes were used for convergence. In situations where storage of these modes is a concern, mode selection can be employed to select only those modes that offer the greatest potential for convergence. Then, it is necessary to define a convergence potential of an additional mode with respect to all initial modes. Morosow and Abbot [57] discuss a method for the selection of modes of one component with respect to another component for obtaining best results in synthesis. Such a technique can be modified to select modes from the available set of additional modes for best convergence results. This can be accomplished either by defining a cut-off convergence potential or by specifying the total number of additional modes to be used. In the former case, only modes with convergence potential greater than the cut-off will be used, while in the latter, the modes have to be ranked on the basis of their potential for convergence, so that the best modes can be used. It is possible however, that more time may be spent trying to decide which modes to use for convergence than if all the available modes are used without any such consideration.

It was observed in the studies that on some occasions, the convergence scheme resulted in more than one initial eigenvalue converging to the same final eigenvalue. It appears that there are certain regions in the eigenvalue-eigenvector space, within which an eigenvalue-eigenvector pair would converge within the region to the desired final values. This is true if the region forms an attracting set with the final eigenpair as the attractor. However, when an eigenpair falls in a

region which repels an eigenpair to a different final set of eigenparameters, convergence problems result. This is not a serious flaw, since the iteration equations did not involve any coupling between the eigenvalue and the eigenvector. The problem was overcome by aborting the iterations as described in Chapter 5. Considering all other merits of the method, this is not a heavy price to pay.

With the present implementation, the code can handle only those interface connections in which all dofs at a node of one component are connected to all dofs at nodes of other components. This can be modified so that interface connections can be less restrictive, i.e., allowing only for partial coupling of dofs at interface nodes. In several cases of component connections, only translational dofs need to be coupled, allowing for free rotations at the interface. Features which allow for such connections to be made can be incorporated. Another desirable feature is the possibility for allowing arbitrary mass loading at the interface and connection through a connector with a specified stiffness. The code can be modified so that arbitrary mass and stiffness values can be specified at the interface. A third feature that can be incorporated in the code is the coupling of components through a rigid link. At present, component coupling is possible only between coincident nodes on the interfaces. In cases where components do not share a common interface but are required to be connected through a rigid connector, a transformation matrix has to be determined to relate the dofs of the two components so that the proper connection is ensured. This feature can also be implemented as a modification to the existing code.

Since a graphic pre-processor was not designed, the code, at present, requires the analyst to specify component dofs to be connected and to choose dependent and independent dofs. This feature can be automated with a graphic pre-processor in which the user can make connections between component dofs, allowing the program to automatically choose the dependent and independent coordinates.

Also, the primary aim of the research was to develop the method for undamped system synthesis. Damping synthesis was briefly discussed in Chapter 6 and was illustrated using a simple example

analyzed using the software MATLAB. The present code could be expanded to include features which allow for the synthesis of damped systems also.

References

[1] Cook, R.D., Malkus, D.S., and Plesha, M.E., 1989, Concepts and Applications of Finite Element Analysis, John Wiley and Sons, New York.

[2] Bathe, K.J., 1982, Finite Element Procedures in Engineering Analysis, Prentice-Hall, Englewood Cliffs, New Jersey.

[3] Lanczos, C., 1956, Applied Analysis, Prentice-Hall, Englewood Cliffs, New Jersey.

[4] Hurty, W.C., 1965, "Dynamic Analysis of Structural Systems using Component Modes", AIAA Journal, Vol. 3 No. 4, pp. 678-685.

[5] Craig, R.R., 1985, "A Review of Time-Domain and Frequency-Domain Component Mode Synthesis Method", Combined Experimental/Analytical Modeling of Dynamic Structural Systems, Presented at the Joint ASCE/ASME Mechanics Conference, Albuquerque, New Mexico, pp. 1-30.

[6] Craig, R.R., and Bampton, M.C.C., 1968, "Coupling of Substructures for Dynamic Analysis", AIAA Journal, Vol. 6 No. 7, pp. 1313-1319.

[7] Hintz, R.M., 1975, "Analytical Methods in Component Modal Synthesis", AIAA Journal, Vol. 13 No. 8, pp. 1007-1016.

[8] Herting, D.N., 1985, "A General Purpose, Multi-stage, Component Modal Synthesis Method", Finite Elements in Analysis and Design, Vol. 1 No. 2, pp. 153-164.

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

[10] Rubin, S., 1975, "Improved Component-Mode Representation for Structural Dynamic Analysis", AIAA Journal, Vol. 13 No. 8, pp. 995-1006.

[11] Craig, R.R., and Chang, C-J., 1977, "On the use of Attachment Modes in Substructure Coupling for Dynamic Analysis", Proceedings of the AIAA/ASME 18th Structures, Structural Dynamics, and Materials Conference, Vol. B, pp. 89-99.

[12] Craig, R.R., and Chang, C-J., 1976, "A Review of Substructure Coupling Methods for Dynamic Analysis", 13th Annual Meeting, Society for Engineering Science, Advances in

Engineering Science, Vol. 2, NASA CP-2001, pp. 393-408.

[13] Bamford, R.M., 1967, "A Modal Combination Program for Dynamic Analysis of Structures", Technical Memorandum 33-290, Jet Propulsion Laboratory, Pasadena, California.

[14] Bajan, R.L., and Feng, C.C., 1968, "Free Vibration Analysis by the Modal Substitution Method", American Astronautical Society, Paper No. 68-8-1, AAS Symposium on Space Projections from the Rocky Mountain Region, Denver, Colorado, July 1968.

[15] Goldman, R.L., 1969, "Vibration Analysis by Dynamic Partitioning", AIAA Journal, Vol. 7 No. 6, pp. 1152-1154.

[16] Hou, S., 1969, "Review of Modal Synthesis Techniques and a New Approach", The Shock and Vibration Bulletin, No. 40 Pt. 4, Naval Research Laboratory, Washington, DC, pp. 25-39.

[17] Benfield, W.A., and Hruda, R.F., 1971, "Vibration Analysis of Structures by Component Mode Substitution", AIAA Journal, Vol. 9 No. 7, pp. 1255-1261.

[18] Guyan, R.J., 1965, "Reduction of Stiffness and Mass Matrices", AIAA Journal, Vol. 32 No. 2, pp. 380.

[19] Arora, J.S., and Nguyen, D.T., 1980, "Eigensolution for Large Structural Systems with Substructures", International Journal for Numerical Methods in Engineering, Vol. 15 No. 3, pp. 333-341.

[20] Hurty, W.C., 1960, "Vibrations of Structural Systems by Component Mode Synthesis", Journal of the Engineering Mechanics Division, Proceedings of the American Society of Civil Engineers, pp. 51-69.

[21] Hale, A.L., and Meirovitch, L., 1980, "A General Substructure Synthesis Method for the Dynamic Simulation of Complex Structures", Journal of Sound and Vibration, Vol. 69, pp. 309-326.

[22] Hasselman, T.K., and Kaplan, A., 1974, "Dynamic Analysis of Large Systems by Complex Mode Synthesis", Journal of Dynamic Systems, Measurement, and Control, Vol. 96 Series G, pp. 327-333.

[23] Chung, Y.T., and Craig, R.R., 1983, "State Vector Formulation of Substructure Coupling for Damped Systems", Proceedings of the AIAA/ASME/ASCE/AHS 24th Structures, Structural Dynamics, and Materials Conference, New York, pp. 520-528.

[24] Howsman, T.G., and Craig, R.R., 1984, "A Substructure Coupling Procedure Applicable to General Linear Time-Invariant Dynamic Systems", Proceedings of the AIAA/ASME/ASCE/AHS 25th Structures, Structural Dynamics, and Materials Conference, pp. 164-171.

[25] Beliveau, J.G., and Soucy, Y., 1985, "Damping Synthesis Using Complex Substructure Modes and a Hermitian System Representation", Proceedings of the AIAA/ASME/ASCE/AHS 26th Structures, Structural Dynamics, and Materials Conference, New York, pp. 581-586.

[26] Kubomura, K., 1987, "Component Mode Synthesis for Damped Structures", AIAA Journal, Vol. 25 No. 5, pp. 740-745.

[27] Craig, R.R., and Ni, Z., 1989, "Component Mode Synthesis for Model Order Reduction of Non-classically Damped Systems", Journal of Guidance, Control, and Dynamics, Vol. 12 No. 4, pp. 577-584.

[28] Klosterman, A.L., 1971, "On the Experimental Determination and Use of Modal Representations of Dynamic Characteristics, Ph. D. Dissertation, University of Cincinnati, Cincinnati, Ohio, 1971.

[29] Klosterman, A.L., and McClelland, W.A., 1973, "Combining Experimental and Analytical Techniques for Dynamic System Analysis", Presented at the 1973 Tokyo Seminar on Finite Element Analysis .

[30] Klahs, J.W., and Townley, G.E., 1985, "Determining Component Loads and Stresses with Improved System Modeling Techniques", Proceedings of the 3rd International Modal Analysis Conference, Orlando, Florida, Union College, Schenectady, New York, pp. 941-948.

[31] Lamontia, M.A., 1982, "On the Determination and Use of Residual Flexibilities, Inertia Restraints, and Rigid-Body Modes", Proceedings of the 1st International Modal Analysis Conference, Orlando, Florida, Union College, Schenectady, New York, pp. 153-159.

[32] Ewins, D.J., and Sainsbury, M.G., 1972, "Mobility Measurements for the Vibration Analysis of Connected Structures", The Shock and Vibration Bulletin, Bulletin 42 Pt. 1, pp. 105-122.

[33] Poelart, D.H.L., 1977, "Dynamic Analysis of a Non-rigid Spacecraft - An Eigenvalue Approach", ESA Journal, Vol. 1 No. 3, pp. 269-281.

[34] Craig, R.R., Bachmeyer, R.C., and Howsman, T.G., 1984, "Some Approaches to Substructure Coupling with Damping, "Proceedings of the 4th International Conference on Applied Numerical Modeling, National Cheng-Kung University, Tainan, Taiwan, pp. 172-177.

[35] Hale, A.L., and Meirovitch, L., 1982, "A Procedure for Improving Discrete Substructure Representation in Dynamic Synthesis", AIAA Journal, Vol. 20 No. 8, pp. 1128-1136.

[36] Bennighof, J.K., 1987, "Component Mode Iteration for Frequency Calculation", AIAA Journal, Vol. 25 No. 7, pp. 996-1002.

[37] Xu, K., Luo, Z., and Han, Z., 1991, "On a New Iterative Approach of Component Mode Synthesis", International Journal for Numerical Methods in Engineering, Vol. 31 No. 6, pp. 1195-1202.

[38] Hale, A.L., 1984, "Substructure Synthesis and its Iterative Improvement for Large Non-conservative Vibratory Systems", AIAA Journal, Vol. 22 No. 2, pp. 265-272.

[39] Hasselman, T.K., and Hart, G.C., 1974, "A Minimization Method for treating Convergence in Modal Synthesis", AIAA Journal, Vol. 12 No. 3, pp. 316-322.

[40] Engels, R.C., 1992, "Convergence Improvement for Component Mode Synthesis", AIAA Journal, Vol. 30 No. 2, pp. 490-495.

[41] Kublanovskaya, V.N., 1970, "On an Approach to the Solution of the Generalized Latent Value Problem for λ - Matrices", SIAM Journal on Numerical Analysis, Vol. 7 No. 4, pp. 532-537.

[42] Leung, A.Y.T., 1992, "An Algorithm for Matrix Polynomial Eigenvalue Problems", Journal of Sound and Vibration, Vol. 158 No. 2, pp. 363-368.

[43] Thurston, G., 1978, "Roots of Lambda Matrices", ASME Journal of Applied Mechanics, Vol. 45 No. 4, pp. 674-689.

[44] Ruhe, A., 1973, "Algorithms for the Nonlinear Eigenvalue Problem", SIAM Journal on Numerical Analysis, Vol. 10 No. 4, pp. 674-689.

[45] Rajakumar, C., 1993, "Lanczos Algorithm for the Quadratic Eigenvalue Problem in Engineering Applications", Computer Methods in Applied Mechanics and Engineering, Vol. 105 No. 1, pp. 1-22.

[46] Moler, C.B., and Stewart, G.W., 1973, "An Algorithm for Generalized Matrix Eigenvalue Problems", SIAM Journal on Numerical Analysis, Vol. 10 No. 2, pp. 241-256.

[47] Fricker, A.J., 1983, "A New Approach to the Dynamic Analysis of Structures using Fixed-Frequency Dynamic Stiffness Matrices, International Journal for Numerical Methods in Engineering, Vol. 19 No. 8, pp. 1111-1129.

[48] Jain, N.K., Singhal, K., and Huseyin, K., 1983, "On Roots of Functional Lambda Matrices", Computer Methods in Applied Mechanics and Engineering, Vol. 40 No. 3, pp. 277-292.

[49] Yang, W.H., 1983, "A Method for Eigenvalues of Sparse λ Matrices", International Journal for Numerical Methods in Engineering, Vol. 19 No. 6, pp. 943-948.

[50] Osborne, M.R., and Michaelson, S., 1964, "The Numerical Solution of Eigenvalue

Problems in which the Eigenvalue Parameter Appears Non-linearly, with an Application to Differential Equations", Computer Journal, Vol. 7, pp. 66-71.

[51] Ramani, A., 1996, "A Computer Program for Two-Step Component Mode Synthesis with Convergence", Report, Department of Mechanical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

[52] MacNeal, R.H., and Harder, R.L., 1988, "A Refined Four-Noded Membrane Element with Rotational Degrees of Freedom", Computers and Structures, Vol. 28 No. 1, pp 75-84.

[53] Bathe, K.J., and Dvorkin, E.N., 1985, "Short Communication: A Four-Node Plate Bending Element Based on Mindlin/Reissner Plate Theory and a Mixed Interpolation" International Journal for Numerical Methods in Engineering, Vol.21, pp. 367-383.

[54] Ramani, A., 1993, "Finite Element Modeling of a Refrigeration Compressor for Noise Prediction Applications", Master's Thesis, Department of Mechanical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

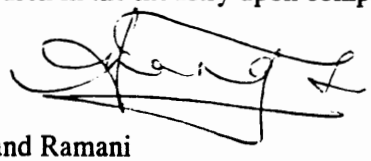
[55] Kelly, A.D., 1992, "Dynamic Finite Element Modeling of a Hermetic Reciprocating Compressor", Master's Thesis, Department of Mechanical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

[56] Anonymous, 1992, MATLAB User's Guide, The Mathworks, Inc., Natwick, Massachussets.

[57] Morosow, G., and Abbot, 1971, "Mode Selection", Synthesis of Vibrating Systems, The Winter Annual Meeting of the ASME, Washington, DC, pp. 72-79.

Vita

Anand Ramani was born in Udamalpet, India on February, the 7th, 1971, to Mrs. Latha Ramani and Mr. R. Ramani. He received his primary and secondary education in the state of Tamilnadu, India, and passed his higher secondary examinations in May 1988 as the top-ranker in the district of Coimbatore, Tamilnadu. Subsequently, he received his Bachelor's degree in Mechanical Engineering as the university gold medalist from the College of Engineering, Guindy, Anna University, Madras, India, in the year 1992. The author enrolled as a graduate student in Virginia Polytechnic Institute and State University in Fall 1992. He graduated with a Master's degree in Mechanical Engineering in December 1993. The author hopes to pursue a career in research in the industry upon completion of his Ph.D.



Anand Ramani