2-9-2010

Non-linear dynamic modeling using component mode synthesis

Jamey Bond

Jamey T. Bond
Candidate

Mechanical Engineering
Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Dr. Tariq A. Khraishi, Chairperson

Dr. Zayd C. Leseman

Dr. Rafiqul A. Tarefder

Dr. William H. Greenwood

NON-LINEAR DYNAMIC MODELING USING COMPONENT MODE SYNTHESIS

BY

JAMEY T. BOND

B.S., Mechanical Engineering, North Carolina State University, 2002
M.S., Mechanical Engineering, North Carolina State University, 2003

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Engineering

The University of New Mexico
Albuquerque, New Mexico

December, 2009

NON-LINEAR DYNAMIC MODELING USING COMPONENT MODE SYNTHESIS

BY

JAMEY T. BOND

ABSTRACT OF DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Engineering

The University of New Mexico
Albuquerque, New Mexico

December, 2009

NON-LINEAR DYNAMIC MODELING
USING COMPONENT MODE SYNTHESIS

by

Jamey T. Bond

B.S., Mechanical Engineering, North Carolina State University, 2002

M.S., Mechanical Engineering, North Carolina State University, 2003

Ph.D., Engineering, University of New Mexico, 2009

ABSTRACT

Dynamic simulations provide insight into the operation of complex mechanisms under dynamic loading conditions. These types of analyses are important to understand design margins and assure that a product meets its functional requirements during all operational environments. Rigid body dynamic modeling techniques can be utilized to simulate mechanisms that experience small loads relative to the strength of the piece parts with movement that primarily occurs as rigid body motion.

The main advantage of a rigid body dynamic approach is that the number of unknowns that must be determined at each time step is dramatically less than the number of unknowns for a direct finite element approach. Rigid body dynamic codes can incorporate the capability to model flexible piece parts within a primarily rigid body

model through the use of component mode synthesis (CMS) techniques. CMS is a method of coupling substructures to represent a large finite element problem as a collection of smaller ones. The number of degrees of freedom are reduced but the CMS method also provides design information and capabilities that are not available with a direct finite element approach. One significant limitation of the CMS implementation is that only linear or non-linear elastic responses can be modeled, requiring a different analysis technique for problems with geometric or material non-linearity.

In this dissertation, a framework is developed to couple non-linear material behavior with a fixed interface CMS technique. This new approach allows non-linear material behavior, such as plastic deformation, to be approximated without requiring the transition to a direct finite element model. The plastic strain is determined from the modal response using classical plasticity theory and applied to the modal solution by projecting an effective nodal force vector on the modal coordinates to induce plastic deformation. The method can be tailored to the frequency range of interest to provide excellent correlation with a full-fidelity finite element solution. Numerical examples are provided to investigate the accuracy and convergence characteristics of the new method for specific problems.

Contents

List of Figures

List of Tables

List of Symbols

Latin Symbols

{a}	Incompatible mode displacement vector
A	Area
{b}	Effective internal force vector
[B]	Strain-displacement matrix
[C]	Global damping matrix
[D]	Material stiffness matrix
E	Modulus of elasticity
E_T	Tangential modulus
[E]	Incompatible mode stiffness matrix
{f}	Global force vector
F	Plasticity yield function
G	Shear modulus
[G]	Incompatible mode strain-displacement matrix
H	Plasticity hardening function
[H]	Incompatible mode stiffness matrix
[I]	Identity matrix
J_2	Second invariant of deviatoric stress tensor
[J]	Jacobian matrix
$[k]_e$	Elemental stiffness matrix

$[K]$	Global stiffness matrix
$[L]$	Lagrangian matrix
$[m]_e$	Elemental inertia matrix
M_0	Limit moment
$[M]$	Global inertia matrix
$[N]$	Orthonormal coordinate transformation matrix
$[N^e]$	Element shape function matrix
p_c	Static collapse distributed load
$\{p\}$	Internal resisting force vector
P_i	Incompatible mode shape functions
$[P]$	Transformation matrix
$[P_f]$	Force transformation matrix
$[P_u]$	Displacement transformation matrix
$\{q\}$	Displacement vector in modal coordinates
$[R_f]$	Residual flexibility matrix
$\{S\}$	Deviatoric stress vector
$[T]$	Kinetic energy matrix
u, v, w	Displacement in global coordinate system
$[U]$	Potential energy
$\{v\}$	Velocity vector
V_e	Element volume
x, y, z	Position in Cartesian coordinate system
W	Work energy

Greek Symbols

α	Newmark integration constant
β	Newmark integration constant
γ	Damping ratio
$\{\Gamma\}$	Modal participation factor vector
$\bar{\varepsilon}_p$	Effective plastic strain
$\{\varepsilon\}$	Strain vector
κ	Plasticity material hardening parameter
λ	Natural frequency
$d\lambda$	Plastic multiplier
$[\Lambda]$	Stiffness matrix in orthonormal coordinates
ν	Poisson's ratio
ξ, η, ζ	Position in isoparametric coordinate system
ρ	Density
σ_e	Effective stress
$\{\sigma_h\}$	Hydrostatic stress vector
σ_Y	Material yield stress
$\{\sigma\}$	Stress vector
τ	Material shear stress
$[\varphi_C]$	Component mode synthesis constraint mode matrix
$[\varphi_N]$	Component mode synthesis normal mode matrix
$[\Phi]$	Component mode synthesis coordinate transformation matrix
ω	Circular natural frequency

Chapter 1

Introduction

The goal of this research was to develop a framework for modeling non-linear material behavior using rigid body dynamic solution techniques. Capabilities currently available within rigid body dynamics codes allow flexible bodies to be incorporated with the use of component mode synthesis (CMS) techniques, but the response is limited to elastic behavior. There is no intermediary between a linear elastic response in a rigid body analysis and a full finite element analysis but there is a great difference in the required computational time and the design information provided.

The incorporation of flexible piece parts can improve the accuracy of a primarily rigid body model if specific piece parts experience large deflections as a result of applied dynamic loading. Simulating the flexibility with CMS offers additional design information that is not available with a direct finite element procedure. The natural frequencies and mode shapes of the flexible elements are provided directly during the CMS procedure, allowing the designer to improve designs that could potentially experience resonance or interference with other bodies. Since the modal problem is solved independently from the dynamic solution, it only needs to be solved once and restart points are automatically provided.

The utility of the rigid body dynamic procedure coupled with CMS can be expanded by incorporating the ability to approximate non-linear behavior for flexible elements that

are loaded beyond their elastic limit. This dissertation is devoted to the development of the non-linear theory and integration with a fixed interface CMS reduction technique. The theoretical background and information required to incorporate within a traditional finite element process are provided in the following chapters along with a collection of numerical examples to demonstrate potential applications and accuracy of the newly developed method.

1.1 Motivation

Dynamic modeling can be used to investigate the response of complex mechanisms during operation or when exposed to environments. This type of modeling is important to providing insight into the response, allowing the design to be characterized and improved. For complex systems where the individual piece parts or subassemblies can be approximated as rigid bodies, the dynamic simulation can be performed using rigid body dynamic techniques. The interaction between rigid bodies are approximated using linear or non-linear contact force and joint options.

In reality, no body is truly rigid. This is only an approximation for cases when the body is very stiff or the loading is very small, which leads to very small deformations. Many of today's rigid body dynamic software packages do include the capability to simulate the linear elastic response of selected piece parts through the use of a CMS method. The flexible body is analyzed to determine the natural frequencies and associated mode shapes, with only the lowest frequencies being retained in the solution of the equations of motion to reduce computational expense. The number of modes retained is dependent on the problem of interest and can have a significant impact on the accuracy of the solution.

The benefit of the CMS method is that the size of the problem can be substantially smaller than that of a full fidelity finite element problem. Rather than maintaining the inertia and stiffness terms associated with each degree of freedom, only a reduced set of modal shapes and frequencies are retained. This reduces the size of the equations of motion being solved at each time step, which is an important consideration for dynamic problems. For a static problem, the CMS method would not offer any computational benefit because the static portion of the equations of motion are only solved once. For a dynamic problem, the equations of motion may be solved thousands to billions of times or more depending on the time step used and the simulated time interval.

The limitation of typical CMS method is that they only apply to linear elastic behavior. If material or geometric non-linearity occurs in the structure, the response cannot be accurately predicted. The preferred option for modeling a problem with non-linearity is through the use of a full fidelity finite element procedure. With the newly developed framework for incorporating non-linear material behavior with a CMS solution technique, this gap in capabilities is reduced.

1.2 Use of Rigid Body Dynamics in Mechanism Design

Computational simulations can offer significant insight into the behavior of complex mechanisms under normal and abnormal operating conditions. An example of a complex mechanism is the ratchet-driver shown in Figure 1-1. The ratchet wheel is rigidly attached to a spur gear and mounted on a shaft with two radial ball bearings. The drive arm is actuated by a rotary solenoid that opens the arm against the extension of the drive spring. Once the arm is sufficiently open, the drive pawl drops over the next tooth because of the torque applied from a torsion spring. As the solenoid is de-energized, the

drive arm returns as a result of the force applied by the extended spring and drives the wheel to the next index position. The dynamic performance of this mechanism can readily be analyzed using rigid body modeling techniques.

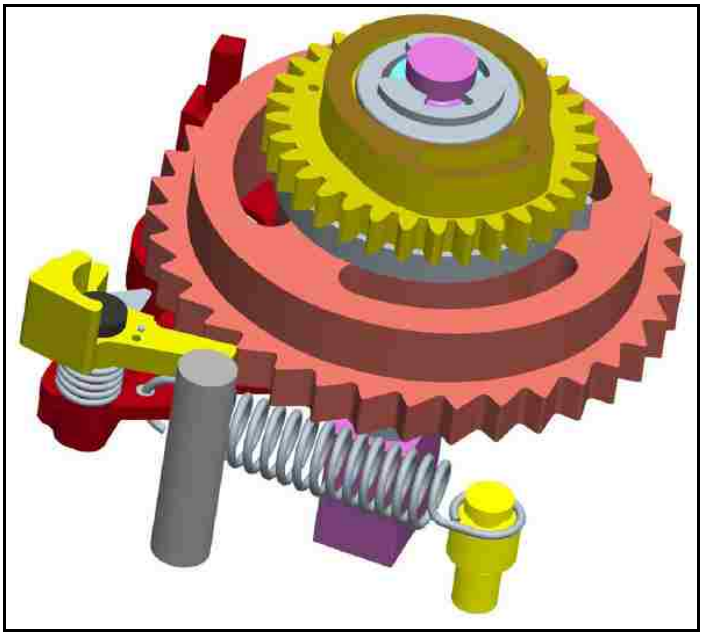

Figure 1-1: Ratchet Driver Mechanism

Experimental testing can be performed to investigate the dynamic performance of complex mechanisms but it is typically impractical to experimentally test all possible loading conditions. Computational simulations can be validated with the limited experimental data and utilized to supplement performance testing, quantify design margins, and identify/characterize failure modes. During the earliest phases of the design process, simplistic analyses can be used to verify the intended function of the mechanism and help identify serious design flaws. As the design matures, the fidelity of the analyses should mature, correspondingly.

The lowest fidelity models used in the early design phase should include many simplifying assumptions since the overall design is immature and subject to frequent

changes. Substantial resources should not be invested in obtaining a high fidelity simulation if high fidelity results are not yet necessary. For mechanisms that experience small loads relative to the strength of the components and motion occurs primarily as rigid body motion during operation, a rigid body solution technique is a good starting point [1]. Consideration of the appropriate integration scheme and time steps are required for the specific geometry and loading of the problem in order to obtain an accurate solution [2]. In the lowest fidelity rigid body dynamic simulation, all components are assumed to remain rigid with interactions represented as idealized joints and contact represented as external forces. This type of solution can be very computationally efficient, which is important for a dynamic response since the simulations will typically be performed over a relatively long time interval. Simulations in early design phases should only require minimal time for setup and solution because the design will likely require several changes.

As the design matures, the simplifying assumptions need to be critically reviewed to determine their impact on the accuracy of the solution. Idealized joints between components may need to be replaced with more representative contact elements. If a majority of the components experience substantial deformation, a different solution technique may be required to obtain higher fidelity results, such as a full fidelity finite element solution. If only a minority of components experience significant deformation, modal techniques have been developed that can be readily incorporated in a primarily rigid body model with minimal impact on computational expense [3]. These modal solutions are an approximation of the full fidelity finite element representation of the substructure that can be solved with substantial computation savings. The savings is

dependent on the particular problem but Figure 1-2 demonstrates the relative number of operations (see Section 6.3) for a modal solution based on the percent of retained modes.

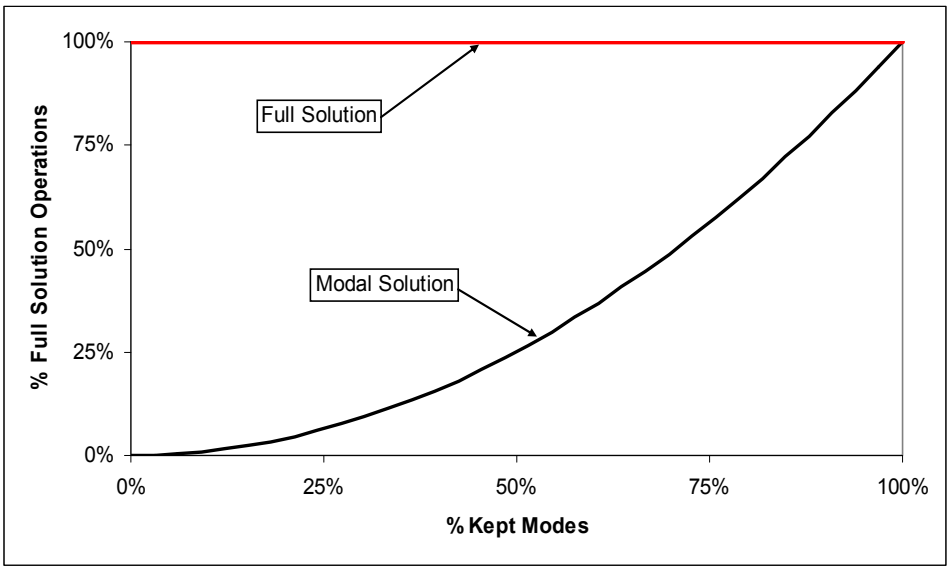

Figure 1-2: Full Fidelity and Modal Computations

The limitation of the modal techniques is that they are typically limited to linear elastic behavior. If the loadings exceed the elastic limits of the material, simulation accuracy will degrade. This dissertation is focused on expanding the useful range of primarily rigid body solution techniques by incorporating non-linear material effects with the modal techniques that are currently available.

1.3 Overview of Rigid Body Dynamic Codes

LMS Virtual.Lab (LMS International, Leuven, Belgium) and MSC.ADAMS (MSC Software, Santa Ana, CA) are examples of commercial rigid body dynamic software packages. These codes incorporate many tools that allow models to be created with varying levels of fidelity. The remainder of this section provides an overview of the process of setting up a rigid body simulation for a typical commercial code.

Model geometry can either be imported from a computer-aided design (CAD) package or can be modeled directly using the integrated CAD software. Once geometry is imported, joints and forces must be set up for each part interface. If parts are mechanically attached through welding or other fastening methods, rigid constraints can be generated to constrain each of the six degrees of freedom between the two parts. There are many other joint options that can approximate various constraint conditions. Idealized joints can have as few as one degree of freedom for a revolute joint or as many as three degrees of freedom for a spherical joint. Compound joints have more degrees of freedom because they are combinations of the simple joints. A representative summary of available joint options is summarized in Table 1-1. The simple joints are computationally inexpensive but the complex joints typically require iterations, which increase the run time of the simulation.

Table 1-1: Rigid Body Dynamics – Joint Options

	Required Inputs		Degrees of Freedom	
Joint Type	**Body 1**	**Body 2**	**Translational**	**Rotational**
Bracket	Axis System	Axis System	0	0
Planar	Plane	Plane	2	1
Spherical	Point	Point	0	3
Cylindrical	Axis	Axis System	1	1
Revolute	Axis, Plane	Axis, Plane	0	1
Screw	Axis	Axis	1	1
Translational	Line, Plane	Line, Plane	1	0
Universal	Line, Point	Line, Point	0	2
Spherical-Spherical	Point	Point	Compound	
Revolute-Sprerical	Line, Plane	Point	Compound	
Revolute-Cylindrical	Line, Point	Line	Compound	
Revolute-Revolute	Line, Plane	Line, Plane	Compound	
Revolute-Translational	Line, Point	Line, Plane	Compound	
CV Joint	Line, Point	Line, Point	Compound	
Point-Curve	Point	Curve	Compound	
Point-Surface	Point	Surface	Compound	
Slide-Curve	Curve	Curve	Compound	
Roll-Curve	Curve	Curve	Compound	

If parts interact through intermittent contact, there are several modeling options. In order to provide a stable numeric solution all contact must be assumed to first occur at a single point. Depending on the geometry of the parts in question, contact can be modeled with the options summarized in Table 1-2. The descriptions provided in these tables are derived from the LMS Virtual.Lab help reference.

Table 1-2: Rigid Body Dynamics – Contact Options

Force Element	Description
Point to Point	The Point-Point Contact element models contact events between two bodies represented by spherical "points". When in contact a force of separation is generated between the two bodies.
Sphere to Extruded Surface	In Sphere-to-Extruded-Surface contact, the first body is designated the Sphere body, and the second is designated the Extruded body. The contact force is based on the depth of penetration and the relative velocity normal to the contact surface.
Sphere to Revolved Surface	Sphere-to-Revolved-Surface contact is the same as Sphere-to-Extruded-Surface contact, except that the surfaces on the first body, rather than being extruded, are revolved about the body 1 axis within a user-defined angular range.
Extruded Surface to Revolved Surface	Extruded-Surface-to-Revolved-Surface contact is similar to Sphere-to-Extruded Surface or Sphere-to-Revolved Surface, except that it allows contact between an extruded surface (as in Extruded-Surface-to-Revolved-Surface) and a revolved surface (as in Sphere-to-Revolved Surface).
Sphere to Rail	The Sphere-to-Rail contact is similar to the Sphere-to-Extruded Surface contact, except that it allows variation along the extrusion direction (Swept instead of Extruded).
CAD Contact	The CAD Contact option allows you to model and simulate contact between bodies with arbitrary geometry. The bodies for which contact is to be calculated are selected, and the complete solid geometry associated with each body participates in contact calculations.
Flexible Contact	The Flexible Contact force element generates action-reaction forces between a sphere on a rigid body and a deforming surface on a flexible body.

Commercial rigid body software codes also have the capability of incorporating flexible elements. If a piece part within a mechanism is expected to deflect during any of the environments being simulated, the rigid piece part can be replaced with a flexible element. This approach offers several advantages over a full finite element model, such as the direct computation of un-restrained natural frequencies, automatic restart points, and computational savings. The primary purpose of rigid body codes is to model rigid

body dynamics. A model is solved through an iterative solution procedure over the desired time interval, with a relatively small set of equations of motion. For a full finite element model, the numerical problem being solved is very large because the full inertia and stiffness matrices are retained. This means that much greater processing capabilities are required to analyze a model over a large solution time.

1.4 Related Research

Methods have been developed to incorporate non-linear behavior in a modal solution for dynamic simulations. For materials that follow a non-linear elastic material stiffness curve, the predicted modal solution can be modified with a manifold calculation [4], [5]. The non-linear normal mode (NNM) method effectively maps the linear response to the non-linear curve. A method to perform non-linear dynamic analyses with modal superposition is presented with example problems of cable and truss structures in reference [6]. A non-linear static and dynamic analysis procedure for framed structures is developed in reference [7]. Coupling of substructuring techniques and mode superposition are explored for the dynamic analysis of structures with perfectly-plastic material stiffness assumptions in reference [8]. This method was incorporated in the ADINA finite element code to reduce the computational time for a certain class of problems.

A collection of reduction techniques are investigated in [9] to demonstrate their effectiveness in non-linear simulations. Techniques of transformation to different basis vectors, prescribed edge displacements, and reduction in mixed finite element modes are evaluated with representative example problems. An explicit time integration method is combined with a non-linear reduction technique in [10]. Transformations of basis vectors

are applied using a Rayleigh-Ritz type technique but the vectors are augmented with vector derivative terms in [10] and [11]. A reduced approach for impulsively loaded structures is developed and evaluated in [12]. Incorporation of localized non-linear effects with a modal reduction framework is investigated in [13]. A survey of various techniques for dynamic substructuring are reviewed and classified in [14].

1.5 Overview of New Non-linear Framework

This dissertation presents a newly developed technique for incorporation of non-linear material behavior with a fixed interface CMS technique. The performance of the technique is investigated with the use of custom Matlab code. All aspects of the finite element formulation, component sub-structuring, iteration of equations of motion, and post-processing operations are performed with the Matlab subroutines, which can be found in the appendices.

The linear elastic response is first predicted at each time step using the reduced modal response. The nodal deformations are evaluated for each element to determine the state of strain and stress. If the effective stress within the element exceeds the predefined yield criteria, the incremental plastic deformation of the element is calculated. This plastic deformation is induced in the linear elastic modal solution by the application of a pseudoforce and an iterative solution technique is used to achieve convergence within the time step [15], [16].

All calculations are performed within the Matlab subroutines, including all necessary pre and post processing. The function that solves the dynamic, three-dimensional problem and calls all subroutines is Master3D.m. In order to generate the global

equations of motion and apply boundary conditions, the subfunctions are called in the order shown in Figure 1-3.

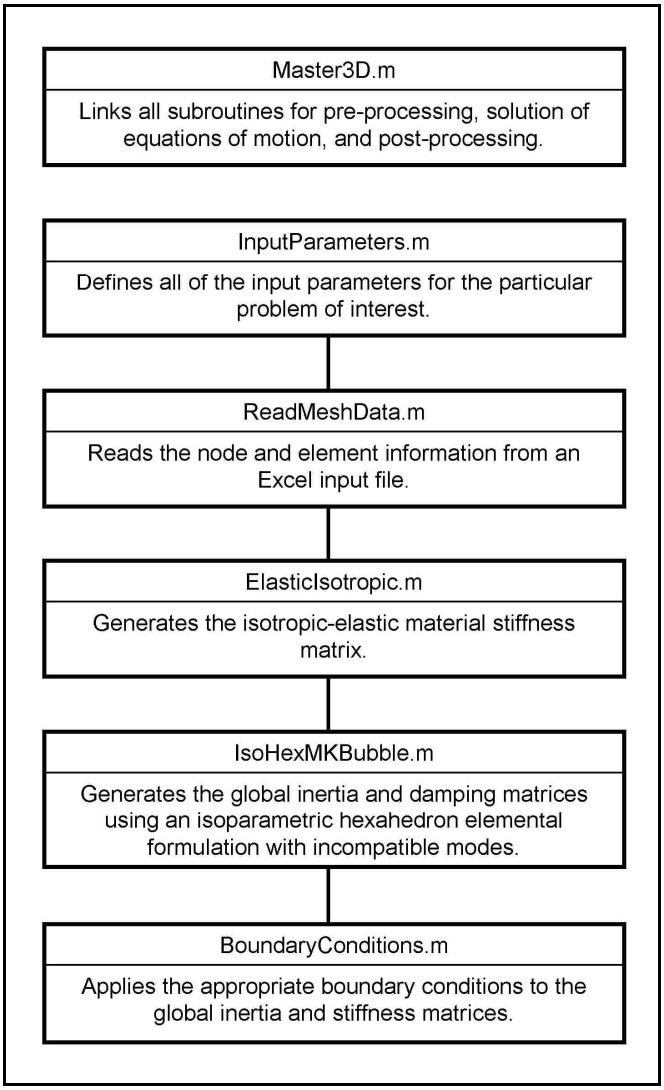

Figure 1-3: Flow Chart for Pre-processing Calculations

The global equations of motion are converted to a CMS representation by applying the reduction techniques of a fixed interface method. This transformation results in inertia and damping matrices that are not orthonormal, so a subsequent

orthonormalization is performed. The inertia and damping matrices are converted to diagonal matrices and input into the iterative, implicit solver as indicated in Figure 1-4.

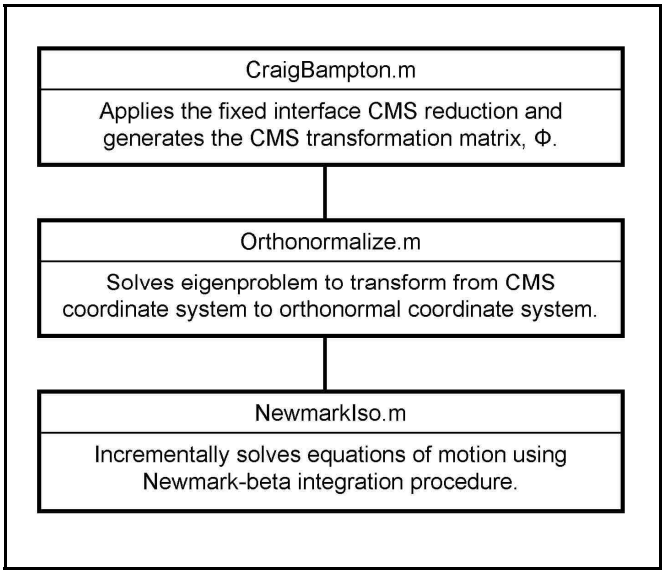

Figure 1-4: Subroutines for CMS Calculations

If a particular time step of the iterative dynamic solution only consists of elastic deformation, the calculations are performed as indicated in Figure 1-5. Because the dynamic equations of motion are solved in a modal coordinate system, the modal displacement vector must be converted to the global coordinate system prior to evaluation of the elemental stress. The plasticity subroutine calculates the effective stress and evaluates the yield function to determine whether the yielding has occurred within the time step. If the deformation is only linear-elastic, the iterative procedure continues until convergence is achieved.

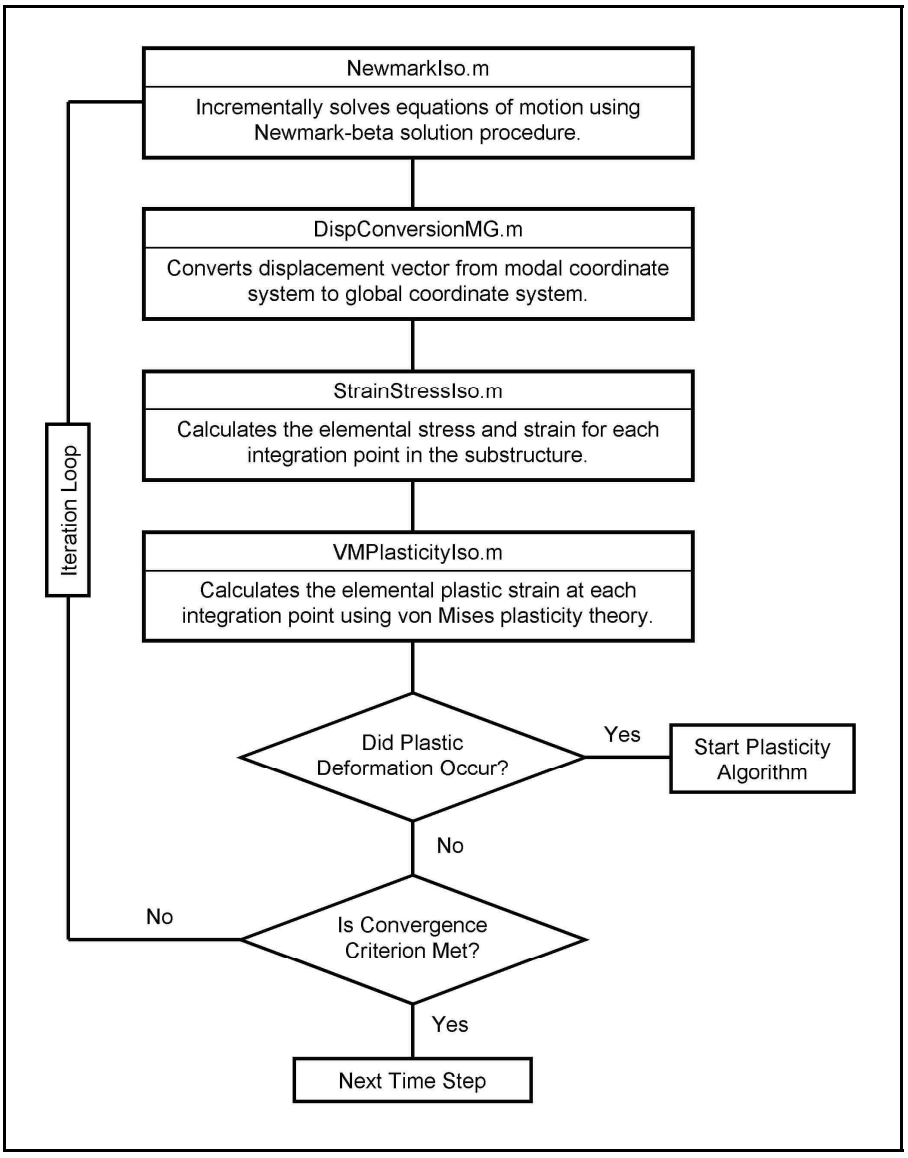

Figure 1-5: Flow Chart for Elastic Time Step

If the time step does result in incremental plastic deformation, the calculations indicated in Figure 1-6 are performed. The tangential stiffness matrix is determined and a plastic pseudoforce vector is calculated to induce the required plastic deformation when introduced into the elastic solution procedure. A combination of initial and tangential stiffness methods are employed, with the stiffness matrix only being generated for the first elastic-plastic iteration.

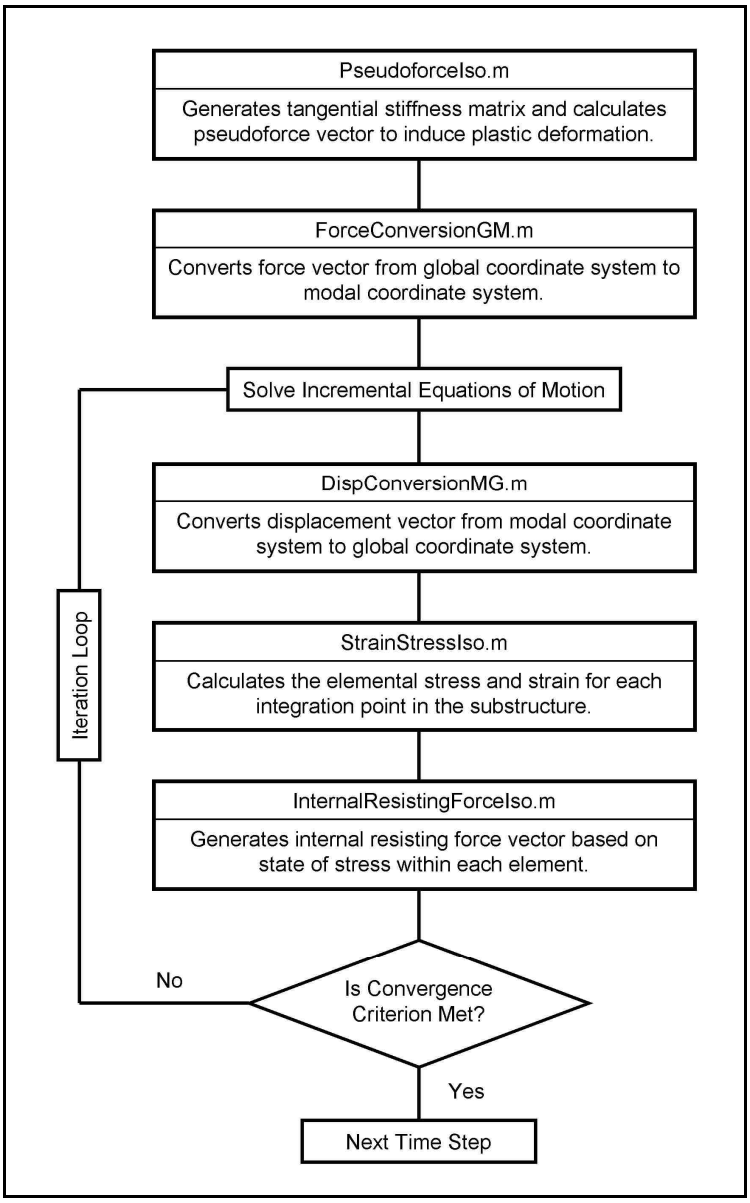

Figure 1-6: Flow Chart for Elastic-Plastic Time Step

Several additional diagnostic subroutines are utilized to calculated additional information and perform routine pre- and post-processing operations. The detailed information about the calculations performed within all subroutines is provided in the remainder of this dissertation, with the complete collection of Matlab code provided in the Appendices.

1.6 Outline of Dissertation

The dissertation is organized into a series of chapters to provide a logical progression of the theory development, implementation, and testing of the non-linear dynamic solution procedure. The initial chapters are devoted to providing the background information necessary to formulate the finite element problem and an overview of the development of modal analysis and sub-structuring techniques. The background information is followed by the development of the plasticity algorithm and the incorporation with the fixed interface CMS technique. Finally, the accuracy and performance of the technique is evaluated through a series of numerical examples and convergence studies.

Chapter 2

Finite Element Analysis

The finite element method is a powerful technique for analyzing the response of complex physical behavior. The finite element procedure basically consists of segmenting a geometric object into a finite number of discrete elements so that the complex problem can be approximated numerically. The elements can be of various sizes and shapes depending on the geometry of the problem being solved. There are also many diverse options in the formulation of the inertia, damping, and stiffness matrices that are tailored to specific loading conditions and problem types. This chapter primarily provides background information of the elements and formulations used in this dissertation for solving structural problems.

One-dimensional elements may be used to solve very basic problems involving the axial extension of a flexible body. Two-dimensional elements may be used to solve problems that can be approximated as plane stress, plane strain, or axi-symmetric. The plane stress approximation can be used for problems where the stress across the thickness is assumed to be zero, such as for thin plates. The plane strain approximation can be used for problems where the strain along the length is assumed to be zero, such as for very long cylinders. The axi-symmetric approximation can be used for bodies that have an axis of symmetry. Further information on one and two-dimensional finite elements can

be found in references [17], [18], and [19]. The remainder of this dissertation will focus on three-dimensional elements since it is the most general case.

2.1 Three-Dimensional Elements

A solid three-dimensional body can be modeled as a combination of a finite number of three-dimensional elements of a prescribed shape. Common examples of three-dimensional finite element shapes are rectangular hexahedron, isoparametric hexahedron, right pentahedron, and tetrahedron. Depending on the geometry being meshed, different element shapes may be required to reasonably approximate the shape. Tetrahedron elements are the easiest to implement in an automated meshing scheme due to their triangular shape but they are typically inaccurate for bending conditions [20]. The problems investigated with this research all utilize hexahedron elements but the method could be easily extended to include other element shapes. Two hexahedron elements are incorporated in the Matlab subroutines to apply to three-dimensional geometries. The rectangular hexahedron, sometimes referred to as the brick element, has six sides and all corners are perpendicular. The isoparametric hexahedron element is also incorporated to mesh more arbitrary shapes. The isoparametric elements are not constrained to have all corners perpendicular and can be shaped to approximate curved surfaces. The rectangular hexahedron is actually a special case of the isoparametric formulation with a slight difference in computational cost.

2.1.1 Rectangular Hexahedron

The rectangular hexahedron consists of eight nodal points positioned at the corners of a six-sided solid element. Various finite element codes and literature use differing

definitions of the nodal arrangement but the form used for this work is identified in Figure 2-1. Differing formulations will have the nodes numbered in other orientations but will provide the same results if carried out consistently.

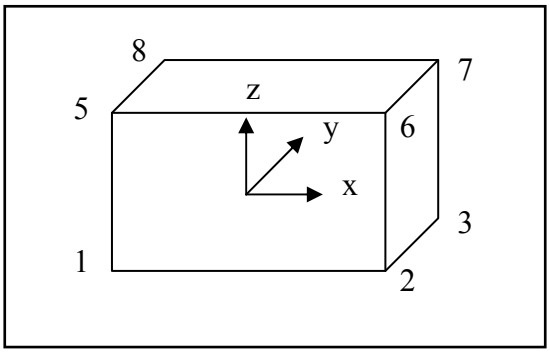

Figure 2-1: Rectangular Hexahedron Element

Each of the eight nodes of the rectangular element has a total of three degrees of freedom because each node can displace in the x, y, and z directions. Other formulations can include rotational degrees of freedom but only the linear displacements are included in this formulation. Therefore, 24 variables are required to completely describe the state of the element. The displacement functions are written in the form:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \left[N^e \right] \{u\}_e \tag{2-1}$$

The terms u, v, and w are the displacements in the x, y, and z direction of the element, which are defined in terms of the shape function of the element and the nodal displacements, u. In terms of the individual displacements of the element, the terms can alternatively be written as:

$$u = \sum_{i=1}^{8} N_i^e u_i$$

$$v = \sum_{i=1}^{8} N_i^e v_i \qquad (2\text{-}2)$$

$$w = \sum_{i=1}^{8} N_i^e w_i$$

The terms u_i, v_i, and w_i are the displacements in the x, y, and z direction of the nodes numbered 1 thru 8. For use in (2-1), the nodal displacements are written as:

$$\{u\}_e = \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ \vdots \\ u_8 \\ v_8 \\ w_8 \end{Bmatrix} \qquad (2\text{-}3)$$

The shape functions, N^e, are written in terms of a mapped set of coordinates (ξ_i, η_i, ζ_i):

$$N_i^e = \frac{1}{8}(1+\xi_i\xi)(1+\eta_i\eta)(1+\zeta_i\zeta) \qquad (2\text{-}4)$$

Expressed in a different form, the shape function can be written as a 3 x 24 matrix for use in (2-1):

$$\left[N^e\right] = \begin{bmatrix} N_1^e & 0 & 0 & \cdots & N_8^e & 0 & 0 \\ 0 & N_1^e & 0 & \cdots & 0 & N_8^e & 0 \\ 0 & 0 & N_1^e & \cdots & 0 & 0 & N_8^e \end{bmatrix} \qquad (2\text{-}5)$$

The mapped coordinate system is based on a defined relationship between the Cartesian coordinate system and the length, width, and height of the element. The dimensions a, b, and c are defined to be half the length, width, and height of the element, respectively:

$$\xi = \frac{x}{a}$$

$$\eta = \frac{y}{b} \tag{2-6}$$

$$\zeta = \frac{z}{c}$$

This relationship maps the locations of the corners of the elements from the Cartesian coordinates to a more generic dimension that varies from -1 to +1.

The inertia matrix of the rectangular hexahedron is determined through integration of:

$$[m]_e = \int_V \rho \left[N^e \right]^T \left[N^e \right] dV \tag{2-7}$$

where V is the volume of the element and ρ is the density of the material. In terms of the physical coordinates of the element:

$$[m]_e = \int\int\int \rho \left[N^e \right]^T \left[N^e \right] dxdydz \tag{2-8}$$

Or, in terms of the mapped coordinates of the element (ξ_i, η_i, ζ_i):

$$[m]_e = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} \rho \left[N^e \right]^T \left[N^e \right] abcd\xi d\eta d\zeta \tag{2-9}$$

The integration over the volume is performed using Gaussian-Legendre integration [21]. Alternatively, the integral can be solved directly for the rectangular hexahedron element.

The element stiffness matrix is generated by first determining the strain-displacement matrix of the element. The strain-displacement matrix is defined in terms of the Cartesian coordinate system:

$$[B]=\begin{bmatrix} \frac{\partial N_1^e}{\partial x} & 0 & 0 & \cdots & \frac{\partial N_8^e}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_1^e}{\partial y} & 0 & \cdots & 0 & \frac{\partial N_8^e}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_1^e}{\partial z} & \cdots & 0 & 0 & \frac{\partial N_8^e}{\partial z} \\ \frac{\partial N_1^e}{\partial y} & \frac{\partial N_1^e}{\partial x} & 0 & \cdots & \frac{\partial N_8^e}{\partial y} & \frac{\partial N_8^e}{\partial x} & 0 \\ \frac{\partial N_1^e}{\partial z} & 0 & \frac{\partial N_1^e}{\partial x} & \cdots & \frac{\partial N_8^e}{\partial z} & 0 & \frac{\partial N_8^e}{\partial x} \\ 0 & \frac{\partial N_1^e}{\partial z} & \frac{\partial N_1^e}{\partial y} & \cdots & 0 & \frac{\partial N_8^e}{\partial z} & \frac{\partial N_8^e}{\partial y} \end{bmatrix} \tag{2-10}$$

The strain-displacement matrix for a single element is of size 6 x 24 for the rectangular hexahedron formulation. The stiffness matrix can be calculated directly using the strain-displacement matrix in the Cartesian coordinate system, but for subsequent calculations it in be more convenient to define the strain-displacement matrix in terms of a mapped coordinate system for ease of integration. This relationship will require the use of chain rule differentiation and utilizes the following relationships derived from differentiation of (2-6):

$$\begin{aligned} \frac{\partial \xi}{\partial x} &= \frac{1}{a} \\ \frac{\partial \eta}{\partial y} &= \frac{1}{b} \\ \frac{\partial \zeta}{\partial z} &= \frac{1}{c} \end{aligned} \tag{2-11}$$

Application of the chain rule to differentiation of the terms of (2-10) with use of (2-4), results in the following relationships, where i varies from 1 to 8:

$$\frac{\partial N_i^e}{\partial x} = \frac{\partial N_i^e}{\partial \xi}\frac{\partial \xi}{\partial x} = \frac{1}{a}\frac{\partial N_i^e}{\partial \xi} = \frac{\xi_i}{8a}(1+\eta_i\eta)(1+\zeta_i\zeta)$$

$$\frac{\partial N_i^e}{\partial y} = \frac{\partial N_i^e}{\partial \eta}\frac{\partial \eta}{\partial y} = \frac{1}{b}\frac{\partial N_i^e}{\partial \eta} = \frac{\eta_i}{8b}(1+\xi_i\xi)(1+\zeta_i\zeta) \qquad (2\text{-}12)$$

$$\frac{\partial N_i^e}{\partial z} = \frac{\partial N_i^e}{\partial \zeta}\frac{\partial \zeta}{\partial z} = \frac{1}{c}\frac{\partial N_i^e}{\partial \zeta} = \frac{\zeta_i}{8c}(1+\xi_i\xi)(1+\eta_i\eta)$$

The stress within the element is defined as a function of the strain within the element and the material elasticity matrix according to the Hooke's law relationship:

$$\{\sigma\} = [D]\{\varepsilon\} \qquad (2\text{-}13)$$

Note that the variable, D, is defined as the elasticity matrix, which is sometimes represented with the variable C. The variable, C, will be reserved to define the damping matrix for the dynamic equations of motion. For an isotropic elastic constitutive model, the elasticity matrix is defined as:

$$[D] = \frac{E}{(1+\upsilon)(1-2\upsilon)}\begin{bmatrix} f & \upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & f & \upsilon & 0 & 0 & 0 \\ \upsilon & \upsilon & f & 0 & 0 & 0 \\ 0 & 0 & 0 & g & 0 & 0 \\ 0 & 0 & 0 & 0 & g & 0 \\ 0 & 0 & 0 & 0 & 0 & g \end{bmatrix} \qquad (2\text{-}14)$$

$$f = (1+\upsilon)$$

$$g = \frac{1}{2}(1-2\upsilon)$$

where E is the modulus of elasticity and ν is the Poisson's ratio of the material. Analysis indicates that the best position to evaluate the stress of the three-dimensional element is at the center [22].

The element stiffness matrix is defined in terms of the strain-displacement matrix and the material stiffness matrix, integrated over the volume of the element:

$$[k]_e = \int\int\int [B]^T [D][B]\, dxdydz \tag{2-15}$$

Or more generally in terms of the mapped coordinate system:

$$[k]_e = abc\int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} [B]^T [D][B]\, d\xi d\eta d\zeta \tag{2-16}$$

Following the procedure used to find the element inertia matrix, the integration is performed using Gauss-Legendre integration. For the quadratic variation defined in (2-4), the exact solution can be obtained by using a 2 x 2 x 2 integration scheme.

2.1.2 Isoparametric Hexahedron

The isoparametric hexahedron element is a more general variation of the rectangular hexahedron. The primary difference is that the sides of the element do not have to be perpendicular, which results in improved approximation of three-dimensional bodies that have curved surfaces. The rectangular hexahedron formulation is a special case of the isoparametric formulation and does not need to be retained directly. Information on isoparametric hybrid hexahedral elements and assumed stress elements can be found in references [23] and [24]. The general shape of the isoparametric hexahedron is shown in Figure 2-2.

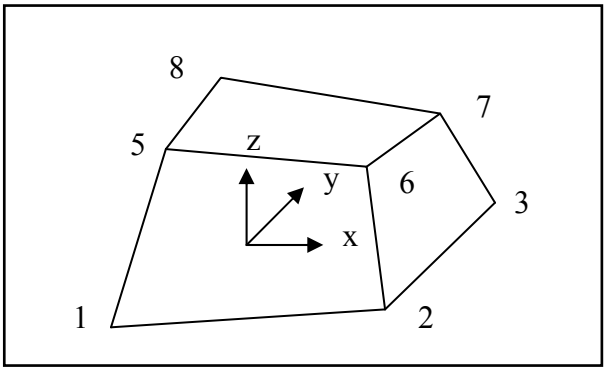

Figure 2-2: Isoparametric Hexahedron (Physical Coordinates)

The nodes of the isoparametric hexahedron are mapped from physical coordinates into isoparametric coordinates in order to improve the ability to integrate over the element. The mapping results in an element that is equivalent to a rectangular hexahedron as shown in Figure 2-3.

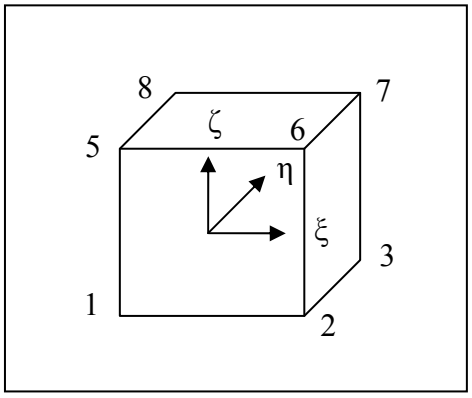

Figure 2-3: Isoparametric Hexahedron (Isoparametric Coordinates)

It is slightly more difficult to map the general Cartesian coordinates into the mapped coordinates for the isoparametric element because the lengths of the sides of the elements are not equal. The introduction of the Jacobian matrix is required to correctly map the volume of the element from the Cartesian coordinates to the mapped coordinates:

$$dxdydz = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{vmatrix} d\xi d\eta d\zeta = |J| d\xi d\eta d\zeta \tag{2-17}$$

The inertia matrix is defined using (2-8) and (2-17):

$$[m]_e = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} \rho \left[N^e\right]^T \left[N^e\right] |J| d\xi d\eta d\zeta \tag{2-18}$$

The element stiffness matrix can be defined using (2-15) and (2-17):

$$[k]_e = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} [B]^T [D][B] |J| d\xi d\eta d\zeta \tag{2-19}$$

After generation of the elemental stiffness matrix, it can be assembled into the global stiffness matrix to correspond with the appropriate degrees of freedom.

The elemental inertia and stiffness matrices are integrated using Gaussian integration for the three-dimensional isoparametric element. The inertia matrix, in (2-18), can be evaluated using a 3 x 3 x 3 integration scheme. This integration requires evaluation at 27 points, but the computational effort can be reduced by employing a fourteen point integration scheme that has be shown to provide similar accuracy [25].

2.1.3 Improvement of Stiffness Matrix Accuracy

The use of the inertia matrices defined for the rectangular and isoparametric elements will produce accurate results for the mass properties of the full system, but the elemental stiffness matrix requires further attention. Analytical analysis of the eight node rectangular elements indicates that the simple element is susceptible to locking, which will produce spurious modes in certain loading situations [26]. The spurious modes occur because the element can deform in specific orientations that will indicate zero

strain energy. Since it is physically impossibly for an element to deform without producing some strain energy, the accuracy of the solution is affected if these loading situations occur.

A possible method for improving the stiffness formulation for the eight node elements is to introduce incompatible modes [27], [28], [29], and [30]. The shape functions of the incompatible modes are defined:

$$P = \begin{bmatrix} 1-\xi^2 \\ 1-\eta^2 \\ 1-\zeta^2 \end{bmatrix} \tag{2-20}$$

These functions are often termed bubble functions because their quadratic shape enables the deformation of the element to approximate a curved shape, resembling a bubble. The incompatible modes are introduced into the original displacement formulation, (2-2):

$$u = \sum_{i=1}^{8} u_i N_i^e + \sum_{i=1}^{3} a_i P_i \tag{2-21}$$

where the variables a_i are the displacements of the bubble degrees of freedom. In terms of matrices and vectors, the relationship is written:

$$\{u\} = \{u\}^T \left[N^e\right] + \{a\}^T [P] \tag{2-22}$$

Since the values of the vector, a, are unknown, they must be determined in order to solve for the elemental displacements. The element strain formulation is then defined:

$$\{\varepsilon\} = [B]\{u\} + [G]\{a\} \tag{2-23}$$

where G is the bubble strain function and is defined:

$$[G_i] = -2\begin{bmatrix} \frac{\partial P_i}{\partial x} & 0 & 0 \\ 0 & \frac{\partial P_i}{\partial y} & 0 \\ 0 & 0 & \frac{\partial P_i}{\partial z} \\ \frac{\partial P_i}{\partial y} & \frac{\partial P_i}{\partial x} & 0 \\ \frac{\partial P_i}{\partial z} & 0 & \frac{\partial P_i}{\partial x} \\ 0 & \frac{\partial P_i}{\partial z} & \frac{\partial P_i}{\partial y} \end{bmatrix} \tag{2-24}$$

for an isoparametric hexahedron element. The subscript i denotes that the matrix is of rank 6 x 9, with i varying from 1 to 3. The partial derivatives in (2-24) are obtained from:

$$\begin{bmatrix} \frac{\partial P_1}{\partial x} & \frac{\partial P_2}{\partial x} & \frac{\partial P_3}{\partial x} \\ \frac{\partial P_1}{\partial y} & \frac{\partial P_2}{\partial y} & \frac{\partial P_3}{\partial y} \\ \frac{\partial P_1}{\partial z} & \frac{\partial P_2}{\partial z} & \frac{\partial P_3}{\partial z} \end{bmatrix} = [J]^{-1} \begin{bmatrix} \frac{\partial P_1}{\partial \xi} & \frac{\partial P_2}{\partial \xi} & \frac{\partial P_3}{\partial \xi} \\ \frac{\partial P_1}{\partial \eta} & \frac{\partial P_2}{\partial \eta} & \frac{\partial P_3}{\partial \eta} \\ \frac{\partial P_1}{\partial \zeta} & \frac{\partial P_2}{\partial \zeta} & \frac{\partial P_3}{\partial \zeta} \end{bmatrix} \tag{2-25}$$

where J is the Jacobian matrix.

The unknown vector, a, is derived by first considering a static situation. The full solution is substructured into two systems, one for the original element stiffness formulation and one for the bubble formulation:

$$\begin{bmatrix} \int [B]^T [D][B] dV & \int [B]^T [D][G] dV \\ \int [G]^T [D][B] dV & \int [G]^T [D][G] dV \end{bmatrix} \begin{Bmatrix} \{u\} \\ \{a\} \end{Bmatrix} = \begin{Bmatrix} \{f\} \\ \{0\} \end{Bmatrix} \tag{2-26}$$

The integrals are defined as matrices for ease of derivation as:

$$\begin{bmatrix} [K] & [E]^T \\ [E] & [H] \end{bmatrix} \begin{Bmatrix} \{u\} \\ \{a\} \end{Bmatrix} = \begin{Bmatrix} \{f\} \\ \{0\} \end{Bmatrix} \tag{2-27}$$

Since the second row of equations is equal to zero, the vector, a, is determined through static condensation:

$$\{a\} = -[H]^{-1}[E]\{u\} \tag{2-28}$$

which can then be substituted into the first row of equation (2-27) to produce:

$$[K]\{u\} - [E]^T [H]^{-1}[E]\{u\} = \{f\} \tag{2-29}$$

Both terms on the left side of the equation are functions of the elemental displacements, u, and it is apparent that the bubble element terms effectively reduce the stiffness of the element. The modified stiffness matrix is defined:

$$\left[\hat{K}\right] = [K] - [E]^T [H]^{-1}[E] \tag{2-30}$$

This modified stiffness replaces the original stiffness matrix in the equations of motion and eliminates the dilatational shear locking of the original hexahedron formulation.

2.2 Selective Substitution

The introduction of the incompatible modes eliminates the shear locking associated with the deviatoric strains but a further modification is required to eliminate the dilatational shear locking. One such technique is to employ selective reduced integration or selective substitution of the shear terms of the strain-displacement matrix [26]. For a three-dimensional element, the xy strain-displacement terms, B^{xy}, are replaced with the Jacobian weighted average over the four Gaussian integration points of a particular face of the element:

$$\bar{B}_{gi}^{xy} = \frac{\sum_{g=1}^{4} J_g B_{gi}^{xy}}{\sum_{g=1}^{4} J_g} \tag{2-31}$$

where J_g is the determinant of the Jacobian at the particular Gaussian integration point. The three-dimensional element will have six faces, requiring the weighted average to be computed six times per element.

2.3 Lagrangian Dynamics

The benefit of Lagrangian dynamics is that the problem is not based on physical coordinate systems. The scalar quantities of energy and work can be substituted for the vector quantities of force, torque, and momentum. The derivation of the Lagrangian equation is based on Newton's Laws and the d'Alembert principle [17]. The general form of Lagrange's equation is defined:

$$[L] = [T] - [U] \tag{2-32}$$

Where T is the kinetic energy and U is the strain energy of the system (or potential energy in the general case). This equation is expressed in a rate form as:

$$\left\{\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{u}}\right)\right\} + \left\{\frac{\partial D}{\partial \dot{u}}\right\} + \left\{\frac{\partial U}{\partial u}\right\} = \{f\} \tag{2-33}$$

Where u is a generalized displacement and f is a generalized force associated with the displacement.

The total strain energy, dissipation function, and kinetic energy of the of the structure are:

$$[T] = \frac{1}{2}\{\dot{u}\}^T [K]\{\dot{u}\} \tag{2-34}$$

$$[D_c] = \frac{1}{2}\{\dot{q}\}^T [C]\{\dot{q}\} \tag{2-35}$$

$$[U] = \frac{1}{2}\{u\}^T [K]\{u\} \tag{2-36}$$

The derivatives of the total strain energy, dissipation function, and kinetic energy are:

$$\left\{\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{u}}\right)\right\} = [M]\{\ddot{u}\} \tag{2-37}$$

$$\left\{\frac{\partial D_c}{\partial \dot{u}}\right\} = [C]\{\dot{u}\} \tag{2-38}$$

$$\left\{\frac{\partial U}{\partial \dot{u}}\right\} = [K]\{u\} \tag{2-39}$$

Lagrange's equation is then written as the familiar equations of motion for a dynamic system:

$$[M]\{\ddot{u}\} + [C]\{\dot{u}\} + [K]\{u\} = \{f\} \tag{2-40}$$

This is the general equations of motion for a multi-body system, where each row of the equations corresponds to a single degree of freedom within the system. For the case when there is only one degree of freedom, (2-40) reduces to the equation of motion for a point:

$$m\ddot{x} + c\dot{x} + kx = f \tag{2-41}$$

Since the finite element method is based on the assumption that a structure is divided into a finite number of individual elements, the multi-body equations of motion are used throughout the remainder of this work.

2.4 Static Analysis

The easiest form of finite element analysis is a static solution. This type of simulation might be used to determine the deflection of a flexible member under a static loading condition. The solution is not dependent on time and all material properties are considered constant. The goal of a static analysis is to determine the solution to equations of the form:

$$[K]\{u\} = \{f\} \tag{2-42}$$

where K is the stiffness matrix, u is a vector of nodal displacements, and f is a vector of nodal forces. The stiffness matrix is symmetric and of the same rank as the number of nodal displacements. The size of the nodal displacement vector is equal to the number of degrees of freedom of the system. For a two-dimensional analysis, each node is able to displace in two directions so the size of the displacement vector is equal to two times the number of nodes. For a three-dimensional analysis, each node has three degrees of freedom and the size of the displacement vector is equal to three times the number of nodes.

For a typical finite element problem, the variables of force are known and the variables of stiffness are determined based on the geometry of the body and the material properties. The variable of interest is the deflection caused by the application of the forces. The solution is then defined by:

$$\{u\} = [K]^{-1}\{f\} \tag{2-43}$$

In very large finite element problems, the stiffness matrix is rarely inverted directly because of the computation cost, but is rather converted through an LU decomposition or

similar approach. Further information about solutions of large systems of equations can be found in references [31], [32], and [33].

2.5 Dynamic Analysis

A typical dynamic analysis is performed when the response of a system must be determined over some time interval. Since the motion of the system is dependent on time, the effects of inertia and damping must be included. A dynamic solution involves determining the solution to the system of equations:

$$[M]\{\ddot{u}\}+[C]\{\dot{u}\}+[K]\{u\}=\{f(t)\} \tag{2-44}$$

The difference between the static and dynamic solutions is the introduction of the mass and damping matrices in (2-44). As defined previously, the variable C is used to indicate the damping matrix of the structure while the variable D is reserved to define the material properties matrix. In the theory of elasticity, the variable C is typically used to indicate the material stiffness matrix but is instead defined as D in this dissertation. The force variable is indicated as a function of time because it is allowable, indeed probable, for the force to vary over the time interval of interest.

Due to the complexity and interdependency of the dynamic equation of motion, the equation cannot be solved directly. Instead, a solution must be obtained using an iterative approach. The two basic types of iterative solution procedures are explicit and implicit. With an explicit solution procedure, the determination of the current iteration is completely based on information obtained during the previous iteration. Of the two methods, the explicit procedure is simpler but it is also highly dependent on the size of

the time step between iterations. If the time step is too large, the integration will be unstable and inaccurate results will result.

Implicit integration procedures can be unconditionally stable for larger time steps but are typically more computationally expensive. With an implicit solution procedure, the determination of the current iteration is based on information from the current iteration as well as information from the previous iteration. This class of solution procedures is sometimes identified as predictor-corrector methods. It is acceptable to use combinations of implicit and explicit integration for many of the mixed integration methods. For the remainder of this dissertation, the Newmark-Beta method was used to determine the iterative solution to the dynamic equations of motion.

2.5.1 Newmark-Beta Integration Method

The Newmark-Beta method is actually a family of solutions based on the assumption that the acceleration varies linearly across the time step. The original formulation of the method can be found in reference [34]. Many variations have been developed to improve the efficiency of the algorithm, such as the method defined in reference [35]. The development of a domain decomposition method can be found in [36]. The algorithm is used in dynamic systems to determine the displacement, velocity, and acceleration of each point at every time step across the time interval of interest. The values based on the information from the previous converged time step will remain constant and are used to determine the following parameters:

$$\{\ddot{u}^*\} = -\frac{1}{\Delta t \beta}\{\dot{u}\}_n - \left(\frac{1}{2\beta} - 1\right)\{\ddot{u}\}_n \tag{2-45}$$

$$\{\dot{u}^*\} = \left(1 - \frac{\gamma}{\beta}\right)\{\dot{u}\}_n + \Delta t\left(1 - \frac{\gamma}{2\beta}\right)\{\ddot{u}\}_n \tag{2-46}$$

$$\{b^*\} = [M]\{\ddot{u}^*\} + [C]\{\dot{u}^*\} \tag{2-47}$$

where the subscript, n, indicates the result at the previous iteration.

The stiffness matrix for the linear-elastic response will not change throughout the dynamic simulation and can be computed and inverted prior to the first time step:

$$\left[K^*\right]^{-1} = \left(\frac{1}{\Delta t^2 \beta}[M] + \frac{\gamma}{\Delta t \beta}[C] + [K]\right)^{-1} \tag{2-48}$$

The effective internal force is required for each iteration with the previous iteration identified as k-1:

$$\{b\}^{(k-1)} = \{f_{n+1}\} - \{p\}^{(k-1)} - \{b^*\} \tag{2-49}$$

where p is the internal resisting force vector. The incremental elastic displacement is calculated as:

$$\{\Delta\Delta u\}^{(k)} = \left[K^*\right]^{-1}\{b\}^{(k-1)} \tag{2-50}$$

The total change in displacement for the iteration and the predicted displacement are defined for the first iteration:

$$\{\Delta u\}^{(k)} = \{\Delta\Delta u\}^{(k)} \tag{2-51}$$

$$\{u\}^{(k)} = \{u_n\} + \{\Delta\Delta u\}^{(k)} \tag{2-52}$$

Once the solution has converged, the final values of displacement, velocity, and acceleration are written:

$$\{u\}_{n+1} = \{u\}^{(k)} \tag{2-53}$$

$$\{\dot{u}\}_{n+1} = \{\dot{u}^*\} + \frac{\gamma}{\Delta t \beta}\{\Delta u\}^{(k)} \tag{2-54}$$

$$\{\ddot{u}\}_{n+1} = \{\ddot{u}^*\} + \frac{1}{\Delta t^2 \beta}\{\Delta u\}^{(k)} \tag{2-55}$$

The values of the variables γ and β determine the assumptions for the variation of acceleration and velocity during the time interval. According to reference [18], the Newmark-Beta can be made unconditionally stable by choosing values for γ and β that conform to the requirements:

$$\gamma \geq 1/2$$
$$\beta \geq \frac{\left(\gamma + 1/2\right)^2}{4} \tag{2-56}$$

For the case when β=¼ and γ=½, the result will be a constant acceleration across the time interval equal to the average of the predicted and corrected acceleration. These parameters of γ and β are used throughout the remainder of this dissertation.

2.5.2 Internal Resisting Force Vector

The internal resisting force vector is defined as the internal force within the structure as a result of the external loadings. This variable is used to define the force induced on the structure after the previous iteration and input into (2-49) for the current iteration. For a linear-elastic problem, the incremental form of the equations of motion are defined:

$$[M]\{\ddot{u}\}_{n+1}^{(k)} + [C]\{\dot{u}\}_{n+1}^{(k)} + [K]\{u\}_{n+1}^{(k)} = \{f\}_{n+1} \tag{2-57}$$

Using (2-52), the stiffness term can be re-written as:

$$[M]\{\ddot{u}\}_{n+1}^{(k)} + [C]\{\dot{u}\}_{n+1}^{(k)} + [K]\{\Delta\Delta u\}^{(k)} = \{f\}_{n+1} \tag{2-58}$$

Since the displacement at the previous iteration is known, that term can be moved to the right side of the equation. If the structure will only experience non-linear deformation, the stiffness matrix will remain unchanged and the equation does not require any further modification. However, if the structure will experience non-linear deformation, the internal force term is alternatively defined in terms of the current state of stress within the element:

$$\{p\}_{el} = \int_V [B]^T [D][B]\{u\}_{el}\, dV = \int_V [B]^T \{\sigma\}_{el}\, dV \qquad (2\text{-}59)$$

where the subscript, el, is used to denote the terms for a single element. This relationship is expressed in the incremental form by only integrating over the incremental change in force. Once the force terms have been computed for each element, they are assembled into the global internal resisting force vector. The process is similar to that used to assemble the global inertia and stiffness matrices of the structure.

Chapter 3

Component Mode Synthesis

Due to the ever increasing complexity of problems being solved and the limitations of computer hardware, methods were developed during the early days of numerical computation to improve the efficiency of numerical simulations. Many techniques were investigated to reduce the computational effort required to solve large static and dynamic problems with minimal computational effort. Modal analysis techniques were developed to decouple the large set of ordinary differential equations and minimize the effort required to solve an iteration of the equations of motions [37], [38]. Substructuring methods were developed to approximate full structures as a collection of discrete substructures, allowing the simulation to be performed in parts. These advances proved especially beneficial in the simulation of dynamic systems, which require the solution of the equations of motion over large time intervals with many individual time steps.

3.1 Modal Analysis Overview

Every flexible structure has some inherent natural frequencies and mode shapes. The modal frequencies of a structure are the eigenvalues, and the mode shapes are the eigenvectors. This information is very important in the design of mechanisms to understand how a piece part or assembly of piece parts will respond under a time dependent forcing function. If the substructures are excited at their natural frequencies,

large deformations and damage can result. Mechanisms must be designed to operate within frequency spectra that will not result in damage or failure due to excitation of the natural frequency of any piece parts.

The natural frequency of an un-damped substructure is based on the distribution of mass and the stiffness of the part. In general, increasing the mass or reducing the stiffness of a structure will result in lower natural frequencies. Decreasing the mass or increasing the stiffness will result in higher natural frequencies. The lowest natural frequencies of a structure occur with minimal deformation energy and usually result in the most simple mode shapes, while the higher frequency mode shapes are typically very complex. Figure 3-1 shows an example of the mode shapes of the drive spring from the ratchet driver mechanism shown in Figure 1-1.

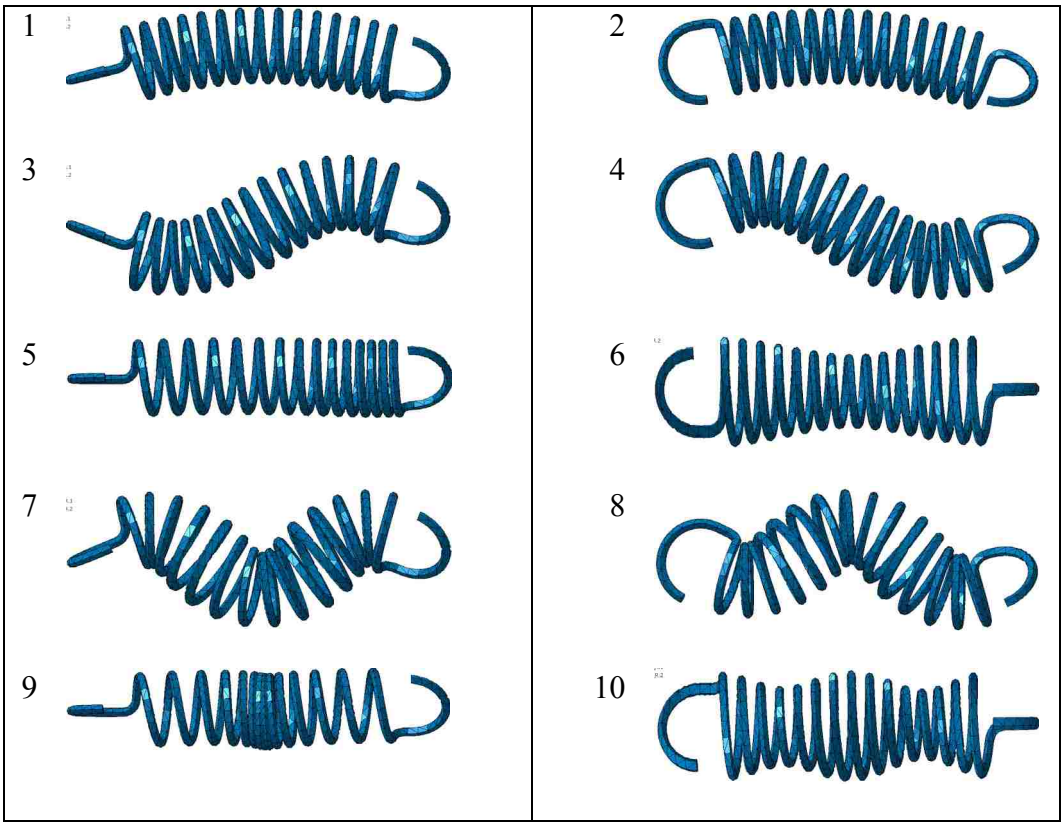

Figure 3-1: Mode Shapes of Extension Spring (10 lowest natural frequencies)

Note that the mode shapes are in the three orthogonal axes of the Cartesian coordinate system. The first and second natural frequencies produce essentially the same mode shape, just along a different axis. The longitudinal mode shapes are not dependent on the radial orientation and occur at higher frequencies due to the increased stiffness in the longitudinal direction. A list of the natural frequencies associated with the mode shapes in Figure 3-1 is shown in Table 3-1.

Table 3-1: Extension Spring Natural Frequencies

Mode	Frequency (Hz)
1	284
2	285
3	1122
4	1123
5	1250
6	1324
7	2379
8	2384
9	2489
10	2634

The process for determining the mode shapes and frequencies for a finite element structure is based on the solution of the un-damped equations of motion. The simplest way to derive the solution procedure is to start with the process for a single degree of freedom system.

3.1.1 Single Degree of Freedom Systems

A single degree of freedom system consists of a single degree of freedom with one lumped mass, damper, and linear stiffness. An example of a single degree of freedom is shown in Figure 3-2.

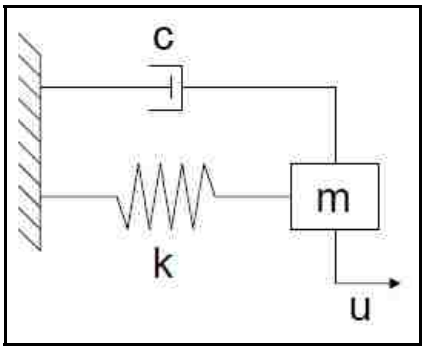

Figure 3-2: Single Degree of Freedom System

The homogenous equation of motion for the single degree of freedom system is:

$$m\ddot{u} + c\dot{u} + ku = 0 \tag{3-1}$$

This homogenous equation can be solved by assuming a solution of the form:

$$u = e^{\lambda t} \tag{3-2}$$

The first and second derivatives of displacement with respect to time are:

$$\dot{u} = \lambda e^{\lambda t} \tag{3-3}$$

$$\ddot{u} = \lambda^2 e^{\lambda t} \tag{3-4}$$

These results are plugged into the original homogeneous equation of motion:

$$m\lambda^2 e^{\lambda t} + c\lambda e^{\lambda t} + ke^{\lambda t} = 0 \tag{3-5}$$

Since there is an exponential in each of the terms, the exponential can be divided out. Also, divide by m:

$$\lambda^2 + \frac{c\lambda}{m} + \frac{k}{m} = 0 \tag{3-6}$$

There are two solutions to (3-6) that can be determined by solving the quadratic equation. The discriminant of the quadratic equation should not be negative to provide two real roots:

$$\lambda_1 = -\frac{c}{m} + \sqrt{\frac{c^2}{m^2} - \frac{4k}{m}}$$
$$\lambda_2 = -\frac{c}{m} - \sqrt{\frac{c^2}{m^2} - \frac{4k}{m}} \tag{3-7}$$

Therefore, the general solution of the homogeneous equation of motion is:

$$u = C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t} \tag{3-8}$$

For a single degree of freedom with a time dependent forcing function, the equation of motion is:

$$m\ddot{u} + c\dot{u} + ku = f(t) \tag{3-9}$$

The homogeneous solution remains unchanged, but the particular solution will depend on the shape of the forcing function. For example, if the forcing function is sinusoidal, then the particular solution will include a sinusoidal term.

3.1.2 Multiple Degree of Freedom Systems

A multiple degree of freedom consists of two or more degrees of freedom with more than one lumped mass, damper, and linear stiffness. An example of a two degree of freedom system is shown in Figure 3-3.

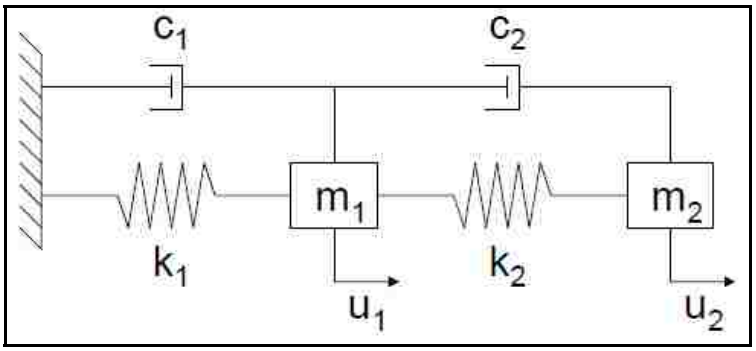

Figure 3-3: Two Degree of Freedom System

The solution of a multiple degree of freedom system follows the same basic approach as the solution for a single degree of freedom system. The homogeneous equation of motion is given by:

$$[M]\{\ddot{u}\}+[C]\{\dot{u}\}+[K]\{u\}=0 \tag{3-10}$$

where M is the inertia matrix of the system, C is the damping matrix of the system, and K is the stiffness matrix of the system. For the example two degree of freedom system in Figure 3-3, the inertia damping, and stiffness matrices are:

$$[M]=\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \tag{3-11}$$

$$[C]=\begin{bmatrix} c_1+c_2 & -c_2 \\ -c_2 & c_2 \end{bmatrix} \tag{3-12}$$

$$[K]=\begin{bmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix} \tag{3-13}$$

The equation of motion for a system that includes forcing functions is given by:

$$[M]\{\ddot{u}\}+[C]\{\dot{u}\}+[K]\{u\}=\{f(t)\} \tag{3-14}$$

For a finite element problem, these matrices will become very large because the rank of each matrix will be equal to the number of degrees of freedom. For a large structure, the matrices will become so large that substantial computing resources will be required to solve the equations of motion. Computational resources proved to be especially problematic for the earliest systems when computing resources were very limited. Much research was conducted to develop methods of decoupling the full problem, allowing a large system could be divided into a collection of discrete subsystems.

3.2 Substructuring Overview

Component substructuring is the technique of dividing a very large finite element problem into a combination of smaller problems. The initial motivation for the creation of such techniques was due to the limited computational resources during the early days of finite element analysis. The key to the substructuring methods is that the number of degrees of freedom are reduced for the individual substructures, which reduces the computational effort required to solve the equations of motion for the overall structure.

3.2.1 Early Component Mode Synthesis

One of the first methods of coupling of substructures was developed and published in 1965 [39] and [40]. Hurty proposed dividing the structure into rigid body modes (R), constraint modes (C), and natural modes (N). The mass matrix is partitioned as:

$$[M] = \begin{bmatrix} M^{RR} & M^{RC} & M^{RN} \\ M^{CR} & M^{CC} & M^{CN} \\ M^{NR} & M^{NC} & M^{NN} \end{bmatrix} \tag{3-15}$$

The stiffness matrix was partitioned as:

$$[K] = \begin{bmatrix} K^{RR} & K^{RC} & K^{RN} \\ K^{CR} & K^{CC} & K^{CN} \\ K^{NR} & K^{NC} & K^{NN} \end{bmatrix} \tag{3-16}$$

Since the first row and first column are rigid body modes, their stiffness matrices are zero. The CN matrix is also zero because the constraints are fixed for normal modes. The simplified stiffness matrix then becomes:

$$[K] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & K^{CC} & 0 \\ 0 & 0 & K^{NN} \end{bmatrix} \tag{3-17}$$

The reduced matrices are then input into the dynamic equations of motion. The elimination of some of the terms reduces the size of the equations of motion and decreases the computational effort required to iteratively solve a dynamic problem. This formulation was incorporated in a finite element program to perform structural dynamic analysis [41].

3.2.2 Free Interface CMS

Free interface methods were developed with the fundamental assumption that the boundary degrees of freedom are free. The natural frequencies of the unrestrained structure are retained directly. The eigenvalues of the structure are determined from the un-damped equations of motion as:

$$\left([K] - \omega_i^2 [M]\right)\{\varphi_i\} = 0 \tag{3-18}$$

where the subscript i indicates that the eigenvalue and eigenvector are determined for each degree of freedom. However, not every eigenvector needs to be retained to obtain an accurate solution because most natural frequencies all well above the force frequencies in real world problems. The individual eigenvectors are assembled into columns in the transformation matrix, φ_N, and neglecting the higher frequency eigenvalues results in a matrix that has fewer columns than rows. Zero frequency modes will be present for a free interface method because the structure will have rigid body modes that must be eliminated. The retained eigenvectors form a transformation matrix that transforms the modal coordinates to the global coordinates:

$$\{u\} = [\varphi_N]\{q\} \tag{3-19}$$

Substituting the transformation matrix into the un-damped equation of motion provides:

$$[M][\varphi_N]\{\ddot{q}\} + [K][\varphi_N]\{q\} = \{f\} \tag{3-20}$$

Premultiply by the transpose of the transformation matrix to obtain:

$$[\varphi_N]^T [M]\{\ddot{q}\} + [\varphi_N]^T [K][\varphi_N]\{q\} = [\varphi_N]^T \{f\} \tag{3-21}$$

The transformed inertia, stiffness, and force matrices are defined for convenience:

$$\begin{aligned} \left[\bar{M}\right] &= [\varphi_N]^T [M][\varphi_N] \\ \left[\bar{K}\right] &= [\varphi_N]^T [K][\varphi_N] \\ \left\{\bar{f}\right\} &= [\varphi_N]^T \{f\} \end{aligned} \tag{3-22}$$

Since the transformation matrix results in orthonormalized coordinates, the effective inertia and stiffness matrix will be diagonal. If the effective matrices are normalized based on the inertia, the equation of motion can be further simplified to:

$$\{\ddot{q}\} + [\Lambda]\{q\} = \left\{\bar{f}\right\} \tag{3-23}$$

where the matrix, Λ, contains the eigenvalues of the un-damped problem along the diagonal.

The difficulty of the free interface method is in the constraints between the individual substructures. The boundary degrees of freedom for any interacting substructures must be equal. Therefore, the boundary degrees of freedom between two substructures is written:

$$\left[\varphi_B^I\right]\left\{q^I\right\} = \left[\varphi_B^{II}\right]\left\{q^{II}\right\} \tag{3-24}$$

where the subscript indicates the boundary displacements and the superscripts indicate substructure I and II. Through linear algebra, the boundary degrees of freedom of one of the structures is calculated in terms of the modal displacements of the other structure and represented with a second transformation matrix:

$$\{q\} = [T]\{r\} \tag{3-25}$$

where r is the reduces set of modal coordinates. The equations of motion of the complete structure are written:

$$[T]^T [M][T]\{\ddot{r}\} + [T]^T [K][T]\{q\} = [T]^T [\varphi_N]^T \{f\} \tag{3-26}$$

The free interface method is particularly useful for problems that will utilize experimental modal information of specific substructures. The coupling of experimental and computational methods for free interface methods have been investigated in reference [42]. Information of the experimental determination of modal results can be found in reference [43]. The convergence of the free interface CMS method is typically weak, meaning that many modes must be retained to obtain a reasonably accurate solution. Other free interface formulations have been developed to improve the results [44], [45], and [46]. A comparison of free interface methods is incorporated in commercial finite element programs is discussed in [47].

3.2.3 Fixed Interface CMS

A method of coupling structures for dynamic analysis was developed and published in 1968 [48]. The coupling of the substructures in this method is fixed. The mass and stiffness matrices are partitioned based on boundary (b) and interior (i) degrees of freedom and the un-damped equations of motion are written:

$$\begin{bmatrix} M^{ii} & M^{ib} \\ M^{bi} & M^{bb} \end{bmatrix} \begin{Bmatrix} \ddot{u}^i \\ \ddot{u}^b \end{Bmatrix} + \begin{bmatrix} K^{ii} & K^{ib} \\ K^{bi} & K^{bb} \end{bmatrix} \begin{Bmatrix} u^i \\ u^b \end{Bmatrix} = \begin{Bmatrix} 0 \\ f^b \end{Bmatrix} \tag{3-27}$$

The upper line of the substructured equation is equal to zero, allowing that portion of the problem to be simplified by applying a reduction technique. Since the equations are homogeneous, the modal solution of the internal portion of the solution can be determined from:

$$\left[M^{ii}\right]\left\{\ddot{u}^i\right\} + \left[K^{ii}\right]\left\{u^i\right\} = 0 \tag{3-28}$$

The eigenproblem is written:

$$\left(\left[K^{ii}\right] + \omega_i^2 \left[M^{ii}\right]\right)\left\{\varphi_i\right\} = 0 \tag{3-29}$$

where the subscript i indicates that the eigenvalues and eigenvectors are determined for each degree of freedom. The eigenvectors are organized into columns and stored in the matrix, φ_N. For real world problems, it is not necessary to retain all of the original natural frequencies of the structure because only the first few modes of the structure are activated. This results in an eigenvalue matrix that has fewer column than rows because the higher frequency modes are neglected. Determination of the number of modes that should be kept is dependent on the problem of interest and the expected excitation frequencies [49].

To complete the transformation matrix, the displacement of the internal nodes must be related to the displacement of the boundary nodes of the structure. In the fixed interface CMS approach, this is accomplished using the Guyan reduction technique [50]. Neglecting the acceleration in (3-27), the equation is written:

$$\begin{bmatrix} K^{ii} & K^{ib} \\ K^{bi} & K^{bb} \end{bmatrix} \begin{Bmatrix} u^i \\ u^b \end{Bmatrix} = \begin{Bmatrix} 0 \\ f^b \end{Bmatrix} \tag{3-30}$$

Upon solving the top portion of the subdivided equation, the following relationship is obtained:

$$\left[K^{ii}\right]\left\{u^{i}\right\}+\left[K^{ib}\right]\left\{u^{b}\right\}=0 \tag{3-31}$$

Equation (3-31) can be solved for the displacement of the internal nodes in terms of the displacement of the boundary nodes to provide:

$$\left\{u^{i}\right\}=-\left[K^{ii}\right]^{-1}\left[K^{ib}\right]\left\{u^{b}\right\}=\left[\varphi_{C}\right]\left\{u^{b}\right\} \tag{3-32}$$

where φ_C is defined to be the constraint modes of the substructure. The complete transformation matrix is then written as:

$$\begin{Bmatrix} u^{i} \\ u^{b} \end{Bmatrix}=\begin{bmatrix} \varphi_{N} & \varphi_{C} \\ 0 & I \end{bmatrix}\begin{Bmatrix} q \\ u^{b} \end{Bmatrix}=[\Phi]\begin{Bmatrix} q \\ u^{b} \end{Bmatrix} \tag{3-33}$$

The transformation matrix is substituted into the original equation of motion by substituting for the global displacements:

$$[M][\Phi]\{\ddot{q}\}+[C][\Phi]\{\dot{q}\}+[K][\Phi]\{q\}=\{f\} \tag{3-34}$$

Premultiply both sides of (3-34) by the transpose of the transformation matrix to obtain:

$$\left[\bar{M}\right]\{\ddot{q}\}+\left[\bar{C}\right]\{\dot{q}\}+\left[\bar{K}\right]\{q\}=\left\{\bar{f}\right\} \tag{3-35}$$

where the following relations have been defined:

$$\left[\bar{M}\right]=[\Phi]^{T}[M][\Phi] \tag{3-36}$$

$$\left[\bar{C}\right]=[\Phi]^{T}[C][\Phi] \tag{3-37}$$

$$\left[\bar{K}\right]=[\Phi]^{T}[K][\Phi] \tag{3-38}$$

$$\left\{\bar{f}\right\}=[\Phi]^{T}\{f\} \tag{3-39}$$

The transformation to the effective inertia and damping matrices reduces the size of the original equations of motion because all of the modes for the internal degrees of freedom were not retained. With the fixed interface CMS method, the convergence is generally good because all of the boundary degrees of freedom are retained without any reduction. One major disadvantage of this method compared to the free interface CMS method is that the size of the reduced problem is larger as a result of the boundary degrees of freedom. If the structure is split into many different substructures, the number of boundary degrees of freedom are increased and must all be retained. Another disadvantage of the original fixed interface CMS method is that the effective inertia and stiffness matrices are not orthonormal but this can be corrected in a subsequent operation.

3.3 Orthonormalization

It is important to note that the equations of motion defined in (3-35) are not a function of orthonormalized coordinates because Φ is a transformation matrix rather than just the eigenvalue matrix of the original equations of motion. The complexity of the dynamic problem can be further reduced by orthonormalizing the un-damped portion of (3-35) to obtain:

$$\left[\left[\bar{K}\right]-\lambda_i\left[\bar{M}\right]\right]\{N_i\}=0 \tag{3-40}$$

The eigenvalue matrix, N, is typically scaled based on the inertia matrix of the structure, allowing the orthonormalized inertia and stiffness matrices to be written:

$$[I]=[N]^T\left[\bar{M}\right][N] \tag{3-41}$$

$$[\Lambda]=[N]^T\left[\bar{K}\right][N] \tag{3-42}$$

The final equations of motion can be written as a set of decoupled equations using index notation:

$$\ddot{\hat{q}}_i + 2\xi_i \lambda_i \dot{\hat{q}}_i + \lambda_i^2 \hat{q}_i = \hat{f}_i \tag{3-43}$$

Equation (3-43) is based on the assumption of Rayleigh type damping with ξ_i defined as the critical damping parameter [17]. Further information on modal as well as non-classical damping can be found in [51], [52], [53], and [54].

3.4 Modal Mass Participation Factor

The modal participation factors can provide an indication of the number of modes that should be kept when using a modal reduction technique [55], [56], and [57]. The lower frequency modes typically have a larger modal mass, which indicates that those frequencies contribute significantly to the dynamic response. The cumulative mass is obtained by summing the current modal participation factor with all lower frequencies to indicate the total mass participation of retaining all frequencies up to the selected frequency. A more useful measure is obtained by dividing the cumulative mass by the total mass of the structure to obtain a percentage. This can be examined to determine the number of modes that should be retained to include the desired portion of the total mass.

The full inertia and stiffness matrices of the substructure must be converted to a modal representation to determine the modal participation. The problem is transformed to a modal representation by solving an the un-damped eigenproblem to obtain the transformation matrix:

$$\{u\} = [N]\{q\} \tag{3-44}$$

where N is a matrix of the eigenvectors of the original problem arranged by column and q is the deformation in the normalized coordinate system. The transformation matrix, N, can be scaled based on the inertia matrix of the substructure to reduce the complication of later calculations. An influence vector, r, is introduced to indicate the displacement of the mass that results from a unit displacement in the global coordinate system. This will typically be equal to one for all degrees of freedom that are not constrained. Using the influence vector, the coefficient vector is defined:

$$\{L\} = [N]^T [M]\{r\} \tag{3-45}$$

The modal participation factors are then obtained as:

$$\Gamma_i = \frac{L_i}{\bar{M}_{ii}} \tag{3-46}$$

If the transformation matrix, N, is scaled based on the inertia matrix, the inertia matrix can be eliminated from (3-45) and the effective inertia matrix in (3-46) can be eliminated as well.

The modal participation factors may be summed for all degrees of freedom or in each direction of the global coordinate system. For problems that will primarily experience deformation in one direction, it may be more beneficial to sum the modal participation factors in that direction to aid in the determination of the number of modes that should be kept. When used in conjunction with a CMS technique, the global inertia and stiffness matrices should be replaced with the appropriate portion of the CMS representation. For the fixed interface CMS approach, the modal reduction is only occurring on the internal degrees of freedom of the substructure. Only the partition of the global inertia and stiffness matrices corresponding to the internal degrees of freedom

need to be evaluated because the remaining degrees of freedom will be retained without any reduction.

Chapter 4

Plasticity Theory

The theory of plasticity and the incorporation within the finite element method has dramatically improved the ability to model complex engineering problems. Plasticity broadens the modeling capabilities to simulate the response of systems that experience non-linear material deformation as a result of loading beyond the elastic capabilities of the material. Such situations occur due to stress concentrations or during abnormal environments that result in the material being loaded beyond its yield strength.

4.1 Yield Criteria

Many yield functions have been developed to predict the response of materials to the application of arbitrary loads [58]. The goal of this dissertation is to investigate the dynamic response of mechanisms composed of primarily metal piece parts, so the yield functions developed for metal materials is most applicable. The two most common yield criteria for metals are the Tresca theory and the von Mises theory. Figure 4-1 graphically demonstrates the difference in the yield surfaces of the Tresca and von Mises theories for a two-dimensional case.

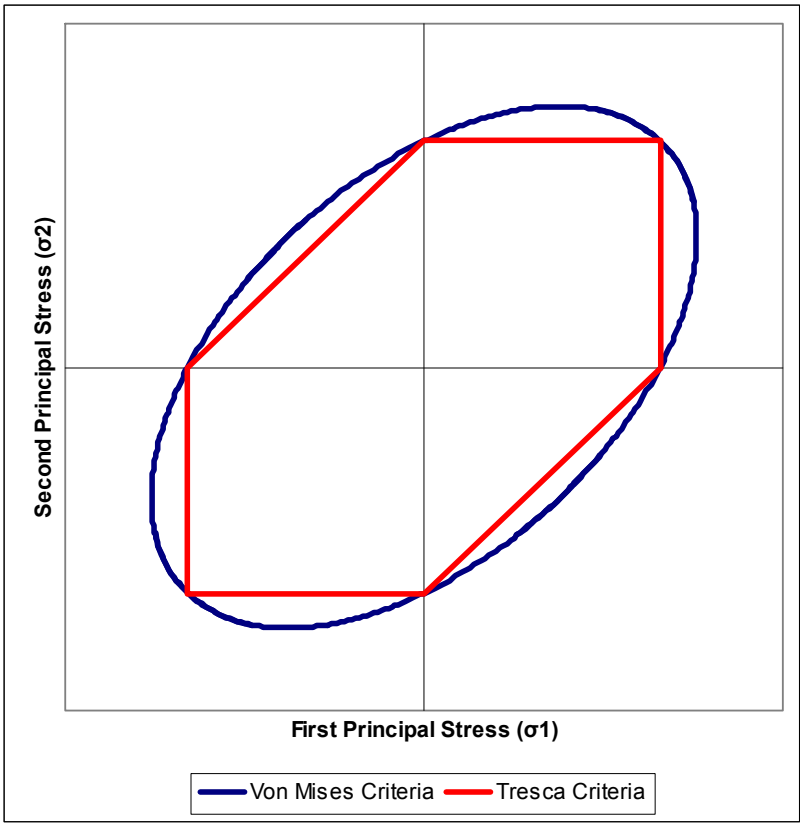

Figure 4-1: Tresca and von Mises Yield Criteria

The Tresca and von Mises criteria each predict yielding at the same points when the stress state is aligned in the principal stress directions or when two principal stresses are equivalent. However, the Tresca criteria will predict yielding before the von Mises criteria in all other stress states, meaning that the Tresca criteria is more conservative than the von Mises criteria.

4.1.1 Tresca Yield Criteria

The Tresca yield criterion was proposed in 1864 as a method of predicting the onset of yielding in metal materials [59]. The Tresca yield criteria predicts that yielding will occur if the maximum shear stress in the element is equal to a critical material parameter, which can be written:

$$\tau_{\max} = \frac{1}{2}\left(\sigma_{\max} - \sigma_{\min}\right) = \kappa \tag{4-1}$$

The critical material parameter is defined:

$$\kappa = \frac{\sigma_Y}{2} \tag{4-2}$$

where σ_Y is the yield strength of the material as determined through a uniaxial tensile test.

4.1.2 von Mises Yield Criteria

The yield criterion commonly attributed to von Mises [61] was actually first published by Huber [60]. This theory is based on the assumption that the onset of yield is based on the value of the second invariant of the deviatoric stress tensor. Yielding will occur if the second invariant of the deviatoric stress tensor is equal to a critical material parameter, which can be written as:

$$J_2' = \kappa^2 \tag{4-3}$$

where the second invariant of the deviator stress tensor is defined as:

$$J_2' = \frac{1}{2}\left(\{S\}^T \{S\}\right) \tag{4-4}$$

if the deviator stress tensor is written in vector notation. The critical material parameter is defined as:

$$\kappa = \frac{\sigma_Y}{\sqrt{3}} \tag{4-5}$$

where σ_Y is the yield strength of the material as determined through a uniaxial tensile test.

4.2 Prandtl-Reuss Plasticity

The fundamentals of the development of finite elements in plasticity are discussed in detail in references [62], [63], [64], [65], [66], and [67]. The first step in the plasticity algorithm is to determine whether the yield stress of the material has been exceeded at any element within the finite element problem. Since only the nodal displacements are known at each time step, the nodal displacements must first be used to calculate the elemental strains. The elastic strain within the element is:

$$\{\varepsilon\}_e = [B]\{u\}_e \tag{4-6}$$

where B is the strain matrix of the element and the vector, u, is the nodal displacements of the element of interest. The elemental elastic stress is calculated according to the Hooke's law relationship:

$$\{\sigma\}_e = [D]\{\varepsilon\}_e \tag{4-7}$$

where D is the elastic material stiffness matrix.

In order to use the von Mises yield criteria, the value of the second invariant of the deviatoric stress tensor must be determined. The deviatoric stress is first calculated by subtracting the hydrostatic stress vector from the vector of stress components:

$$\{S\} = \{\sigma\} - \{\sigma_h\} \tag{4-8}$$

The stress terms are written in a vector form for storage convenience since only six of the nine components are unique. The hydrostatic stress vector is:

$$\{\sigma_h\} = \begin{Bmatrix} \left(\frac{\sigma_x + \sigma_y + \sigma_z}{3}\right) \\ \left(\frac{\sigma_x + \sigma_y + \sigma_z}{3}\right) \\ \left(\frac{\sigma_x + \sigma_y + \sigma_z}{3}\right) \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{4-9}$$

The effective stress is calculated according to the relationship:

$$\sigma_e = \sqrt{\frac{3}{2}}\left(\{S\}^T \{S\}\right)^{\frac{1}{2}} \tag{4-10}$$

This effective stress is compared against the yield strength of the material, which is a material property, to determine whether yielding has occurred [68]. For an isotropic hardening constitutive model, the hardening of the material can be defined according to the following relationship with the assumption of bi-linear hardening:

$$H = H_0 + \frac{H_L - H_0}{\bar{\varepsilon}_L}\bar{\varepsilon}_p \tag{4-11}$$

H_0 is the initial yield stress of the material. H_L and $\bar{\varepsilon}_L$ are the tensile strength and strain limit, respectively. The variables in the second term are the slope of the material hardening multiplied by the effective plastic strain already induced in the material.

The onset of plastic deformation is determined by evaluating the yield function. For a yield function that is defined according to the second invariant of the stress, the relationship will be a function of the stress and any hardening parameters. For the remainder of this dissertation, isotropic hardening will be assumed. For hardening materials, the current yield strength is typically a function of the total plastic work or the

total plastic deformation of the element. For the work hardening hypothesis, the incremental plastic work is defined:

$$dW_p = \{\sigma\}^T \{d\varepsilon_p\} \tag{4-12}$$

For the strain hardening hypothesis, the incremental effective plastic strain is defined:

$$d\bar{\varepsilon}_p = \sqrt{\frac{2}{3}}\left[\{d\varepsilon_p\}^T \{d\varepsilon_p\}\right]^{1/2} \tag{4-13}$$

The value of the incremental effective plastic strain is not simply equal to the incremental plastic strain to ensure that path dependency is maintained. The effective plastic strain is always increasing and never allowed to decrease, which would be physically impossible.

It has been shown that the work hardening hypothesis, rather than the strain hardening hypothesis, is most correct in terms of the laws of thermodynamics [69]. However, when used with isotropic hardening and the von Mises yield criteria, the two hypotheses are equivalent. Therefore, the yield function will be defined:

$$F(\sigma, \bar{\varepsilon}_P) = f(\{\sigma\}) - \kappa(\bar{\varepsilon}_P) = 0 \tag{4-14}$$

The first term is assumed to be only a function of the current stress and the second term is assumed to be only a function of the effective plastic strain. Since the yield function is assumed to be a function of the second invariant of the deviatoric stress, the first term of (4-14) is:

$$f(\{\sigma\}) = J_2' = \frac{1}{2}\left(\{S\}^T \{S\}\right) \tag{4-15}$$

The isotropic hardening function is assumed to be a function of the effective plastic strain:

$$\kappa\left(\bar{\varepsilon}_p\right) = \frac{1}{3} H^2\left(\bar{\varepsilon}_p\right) \tag{4-16}$$

where H is defined to be the hardening function of the material. This function can be determined from a uniaxial tensile test and approximated according to the previously defined linear hardening model in (4-11) or other models [65].

An incremental approach is used because the stress state of the elements can be continuously varying. The equations of motion are solved incrementally and the state of stress in the elements is calculated to determine whether yielding has occurred within the time step. After evaluation of (4-14), a determination is made of whether the element is experiencing plastic deformation. If the yield function is less than zero, then the element only experienced elastic deformation during the increment. If the yield function is greater than zero, then the element is experiencing plastic deformation as a result of the incremental strain. Since it is physically impossible for the yield function to be greater than zero, the state of stress or strain within the element must be adjusted to equate the yield function to zero. For a perfectly plastic constitutive model, the calculated elastic strain must be re-evaluated if the stress exceeds the yield strength of the element.

A Newton-Raphson procedure is used to determine the plastic strain within the element [65]. Using the Prandtl-Reuss method of plasticity [66], the associated flow rule is written:

$$\left(d\varepsilon_{ij}\right)_p = d\lambda \frac{\partial f}{\partial \sigma_{ij}} \tag{4-17}$$

With the use of the von Mises yield criterion, the derivative of the yield function with respect to the stress is:

$$\frac{\partial f}{\partial \sigma_{ij}} = S_{ij} \tag{4-18}$$

Upon substitution of (4-18) into (4-17):

$$\left(d\varepsilon_{ij}\right)_p = d\lambda S_{ij} \tag{4-19}$$

This relationship will provide the full plastic deformation tensor but it must be converted to an effective plastic strain for use in the hardening function. Since plastic deformation is non-conservative, the effective plastic strain is defined:

$$\left(d\bar{\varepsilon}\right)_p = \sqrt{\frac{2}{3}\left[\left(d\varepsilon_{ij}\right)_p^T \left(d\varepsilon_{ij}\right)_p\right]} \tag{4-20}$$

The incremental effective plastic strain is added to the previous effective plastic strain:

$$\left(\bar{\varepsilon}\right)_p = \left(\bar{\varepsilon}\right)_p + \left(d\bar{\varepsilon}\right)_p \tag{4-21}$$

The Newton-Raphson procedure is used to determine the value of dλ that satisfies the relationships. For a linear hardening rule, the iterative procedure will converge to the correct solution in two iterations. For more complex hardening rules, additional iterations will be required.

4.3 Determination of Strain Contributions

During each time step, the calculated nodal deformations are assumed to follow a linear elastic relationship because the stiffness matrix has not been updated. The stress is calculated to determine whether the element is experiencing plastic deformation during the iteration. If the stress within the element exceeds the yield stress, then the incremental plastic strain within the element is determined. This incremental plastic strain effectively increases the total strain within the element. Since the element is

experiencing more deformation than predicted by the linear elastic relationship, the element appears to have a lower stiffness.

After the incremental plastic strain tensor is determined, the value of the total incremental strain is updated:

$$d\varepsilon = d\varepsilon_e + d\varepsilon_p \tag{4-22}$$

For a hardening material, the incremental elastic strain remains unchanged during the Newton-Raphson procedure, but the total incremental strain is increased by the incremental plastic strain. For an elastic-perfectly plastic constitutive model or for softening, the incremental elastic strain would also require modification during the iterative solution of the increment.

Chapter 5

Integration of CMS and Plasticity Theory

Traditional CMS techniques only predict the linear elastic response of a structure to a time varying or constant forcing function. In order to approximate non-linear material response, the linear elastic response must be coupled with plasticity theory to provide the total elastic and inelastic response of the structure. After determination of the linear elastic response, the elemental stress and strain are modified using plasticity theory, and then the two solutions are coupled to determine the total response during the incremental time step.

5.1 Determination of Non-linear Response

Using plasticity theory, the state of stress for each element must be evaluated to determine whether the element is experiencing plastic deformation. If the yield stress of the material has been exceeded within the incremental time step, then the value of plastic strain is calculated and added to the elastic strain to provide the total strain within the element. Since the effect of plastic deformation is to increase the total strain within the element beyond that predicted by the elastic response, the element effectively appears to be less stiff when plastic deformation occurs. Therefore, a tangential elemental stiffness matrix can be determined for the incremental time strep to provide the total (elastic and plastic) strain within the element.

5.1.1 Updated Material Stiffness-Hardening Models

For hardening constitutive models, the yield strength of the material increases with increased plastic strain. Eventually, the material will reach its ultimate tensile strength and softening will begin to occur, but for this dissertation only the hardening portion of the response is investigated. For yielding defined in terms of a uniaxial stress strain hardening curve, the yield function is defined to be a function of the current stress of the element with the yield strength of the material a function of some chosen hardening parameter. The hardening parameter is chosen to be the effective plastic strain; therefore, the yield function is written:

$$F\left(\sigma_{ij},\bar{\varepsilon}^{p}\right)=f\left(\sigma_{ij}\right)-\kappa\left(\bar{\varepsilon}^{p}\right)=0 \tag{5-1}$$

The derivative of the yield function is termed the consistency condition and are written:

$$dF\left(\sigma_{ij},\bar{\varepsilon}^{p}\right)=\frac{\partial F}{\partial\sigma_{ij}}d\sigma_{ij}+\frac{\partial F}{\partial\kappa}d\kappa=0 \tag{5-2}$$

Using the Prandtl-Reuss definition for plastic strain the incremental plastic strain is:

$$d\varepsilon_{kl}{}^{p}=d\lambda\frac{\partial F}{\partial\sigma_{kl}}=d\lambda\frac{\partial f}{\partial\sigma_{kl}} \tag{5-3}$$

The derivative of the yield function with respect to the current stress can be written in terms of f because the hardening parameter is only a function of the effective plastic strain.

Using the relationship for the incremental elastic stress from (4-7), the total incremental strain is written:

$$d\varepsilon_{kl}=D_{ijkl}{}^{-1}d\sigma_{ij}+d\lambda\frac{\partial f}{\partial\sigma_{kl}} \tag{5-4}$$

If (5-4) is premultiplied on both sides by $\frac{\partial f}{\partial \sigma_{ij}} D_{ijkl}$, the following is obtained:

$$\frac{\partial f}{\partial \sigma_{ij}} D_{ijkl} d\varepsilon_{kl} = \frac{\partial f}{\partial \sigma_{mn}} d\sigma_{mn} + d\lambda \frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}} \tag{5-5}$$

The goal is to use (5-5) to define a relationship for the plastic multiplier, dλ, so the first term on the right side must be rewritten in terms of the plastic multiplier. From the consistency condition in (5-2), the following relationship is obtained:

$$\frac{\partial f}{\partial \sigma_{mn}} d\sigma_{mn} = -\frac{\partial F}{\partial \kappa} d\kappa \tag{5-6}$$

The right side of (5-6) can be written in terms of the plastic multiplier by defining a new scalar variable:

$$A = -\frac{1}{d\lambda} \frac{\partial F}{\partial \kappa} d\kappa \tag{5-7}$$

Equation (5-7) can then be rewritten:

$$\frac{\partial f}{\partial \sigma_{mn}} d\sigma_{mn} = A d\lambda \tag{5-8}$$

Using the newly introduced scalar variable, A, (5-5) can be written as:

$$\frac{\partial f}{\partial \sigma_{ij}} D_{ijkl} d\varepsilon_{kl} = A d\lambda + d\lambda \frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}} \tag{5-9}$$

$$\frac{\partial f}{\partial \sigma_{ij}} D_{ijkl} d\varepsilon_{kl} = d\lambda \left(A + \frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}} \right) \tag{5-10}$$

Solving for the plastic multiplier, dλ:

$$d\lambda = \frac{\frac{\partial f}{\partial \sigma_{ij}} D_{ijkl} d\varepsilon_{kl}}{A + \frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}}} \tag{5-11}$$

Substituting (5-11) into the total incremental strain relationship, (5-4), results in:

$$d\varepsilon_{kl} = D_{ijkl}{}^{-1} d\sigma_{ij} + \left(\frac{\partial f}{\partial \sigma_{kl}} \right) d\lambda \tag{5-12}$$

In order to be able to solve (5-12) for the incremental stress, premultiply both sides by the elasticity tensor, D:

$$D_{ijkl} d\varepsilon_{kl} = d\sigma_{ij} + \frac{\left(D_{ijmn} \dfrac{\partial f}{\partial \sigma_{mn}} \right) \left(\dfrac{\partial f}{\partial \sigma_{pq}} D_{pqvw} \right)}{A + \dfrac{\partial f}{\partial \sigma_{rs}} D_{rstu} \dfrac{\partial f}{\partial \sigma_{tu}}} d\varepsilon_{vw} \tag{5-13}$$

Solving for the incremental stress, dσ:

$$d\sigma_{ij} = D_{ijkl} d\varepsilon_{kl} - \frac{\left(D_{ijmn} \dfrac{\partial f}{\partial \sigma_{mn}} \right) \left(\dfrac{\partial f}{\partial \sigma_{pq}} D_{pqvw} \right)}{A + \dfrac{\partial f}{\partial \sigma_{rs}} D_{rsyu} \dfrac{\partial f}{\partial \sigma_{tu}}} d\varepsilon_{vw} \tag{5-14}$$

$$d\sigma_{ij} = \left[D_{ijkl} - \frac{\left(D_{ijmn} \dfrac{\partial f}{\partial \sigma_{mn}} \right) \left(\dfrac{\partial f}{\partial \sigma_{pq}} D_{pqkl} \right)}{A + \dfrac{\partial f}{\partial \sigma_{rs}} D_{rstu} \dfrac{\partial f}{\partial \sigma_{tu}}} \right] d\varepsilon_{kl} \tag{5-15}$$

$$d\sigma_{ij} = D_{ijkl}{}^{ep} d\varepsilon_{kl} \tag{5-16}$$

where the elastic-plastic material stiffness tensor can be defined:

$$D_{ijkl}{}^{ep} = D_{ijkl} - \frac{\left(D_{ijmn} \dfrac{\partial f}{\partial \sigma_{mn}} \right) \left(\dfrac{\partial f}{\partial \sigma_{pq}} D_{pqkl} \right)}{A + \dfrac{\partial f}{\partial \sigma_{rs}} D_{rstu} \dfrac{\partial f}{\partial \sigma_{tu}}} \tag{5-17}$$

The relationship derived for the elastic plastic material stiffness tensor can be simplified by including the assumptions of the Prandtl-Reuss plasticity theory. From (4-18), the derivative of the yield function with respect to the current stress is written:

$$\frac{\partial f}{\partial \sigma_{ij}} = S_{ij} \rightarrow \{S\} \tag{5-18}$$

where the deviatoric stress components are written in vector notation. The scalar value of A in (5-7) can be simplified using (4-16):

$$A = -\frac{1}{d\lambda}\frac{\partial}{\partial \bar{\varepsilon}^p}\left(\frac{1}{3}H\left(\bar{\varepsilon}^p\right)^2\right)d\bar{\varepsilon}^p \tag{5-19}$$

$$A = -\frac{1}{d\lambda}\frac{\partial}{\partial \bar{\varepsilon}^p}\left(\frac{2}{3}H\frac{dH}{d\bar{\varepsilon}^p}\right)d\bar{\varepsilon}^p \tag{5-20}$$

The final simplified version of (5-17) can be written in matrix form as:

$$[D]_{ep} = [D] - \frac{[D]\{S\}\{S\}^T[D]}{A + \{S\}^T[D]\{S\}} \tag{5-21}$$

5.1.2 Updated Material Stiffness-Perfectly Plastic Model

For perfectly plastic constitutive models the procedure used to determine the elastic plastic material matrix must be modified from that used for hardening constitutive models. The yield function of a perfectly plastic constitutive model is defined:

$$F\left(\sigma_{ij}\right) = f\left(\sigma_{ij}\right) - \kappa^0 = 0 \tag{5-22}$$

where the hardening function is replaced with a constant and F is a function of stress only. The consistency condition, or the derivative of the yield function, can then be written:

$$dF = \frac{\partial f}{\partial \sigma_{ij}} d\sigma_{ij} = 0 \tag{5-23}$$

The total strain increment is the same as defined in (5-4):

$$d\varepsilon_{kl} = D_{ijkl}^{-1} d\sigma_{ij} + d\lambda \frac{\partial f}{\partial \sigma_{kl}} \tag{5-24}$$

Premultiply both sides of the equation by $\frac{\partial f}{\partial \sigma_{ij}} D_{ijkl}$

$$\frac{\partial f}{\partial \sigma_{ij}} D_{ijkl} d\varepsilon_{kl} = \frac{\partial f}{\partial \sigma_{pq}} d\sigma_{pq} + d\lambda \frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}} \tag{5-25}$$

However, from the consistency condition (5-23), the first term on the right side of the equation is equal to zero, so (5-25) can be rewritten:

$$\frac{\partial f}{\partial \sigma_{ij}} D_{ijkl} d\varepsilon_{kl} = d\lambda \frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}} \tag{5-26}$$

Solving (5-26) for the plastic multiplier, dλ:

$$d\lambda = \left(\frac{\frac{\partial f}{\partial \sigma_{ij}} D_{ijkl}}{\frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}}} \right) d\varepsilon_{kl} \tag{5-27}$$

Equation (5-27) can then be substituted into the decomposed incremental strain relationship from (5-24) to produce:

$$d\varepsilon_{kl} = D_{ijkl}^{-1} d\sigma_{ij} + \left(\frac{\partial f}{\partial \sigma_{kl}} \right) \left(\frac{\frac{\partial f}{\partial \sigma_{mn}} D_{mnpq}}{\frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}}} \right) d\varepsilon_{pq} \tag{5-28}$$

In order to solve for the incremental stress, premultiply both sides by the elasticity matrix:

$$D_{ijkl}d\varepsilon_{kl} = d\sigma_{ij} + \left(\frac{\left(D_{ijmn}\frac{\partial f}{\partial \sigma_{mn}} \right)\left(\frac{\partial f}{\partial \sigma_{pq}} D_{pqvw} \right)}{\frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}}} \right) d\varepsilon_{vw} \tag{5-29}$$

Solve (5-29) for the incremental stress:

$$d\sigma_{ij} = D_{ijkl}d\varepsilon_{kl} - \left(\frac{\left(D_{ijmn}\frac{\partial f}{\partial \sigma_{mn}} \right)\left(\frac{\partial f}{\partial \sigma_{pq}} D_{pqvw} \right)}{\frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}}} \right) d\varepsilon_{vw} \tag{5-30}$$

$$d\sigma_{ij} = \left[D_{ijkl} - \frac{\left(D_{ijmn}\frac{\partial f}{\partial \sigma_{mn}} \right)\left(\frac{\partial f}{\partial \sigma_{pq}} D_{pqvw} \right)}{\frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}}} \right] d\varepsilon_{vw} \tag{5-31}$$

where the elastic-plastic stiffness tensor can be defined:

$$D_{ijkl}^{\ ep} = D_{ijkl} - \frac{D_{ijmn}\frac{\partial f}{\partial \sigma_{mn}} \frac{\partial f}{\partial \sigma_{pq}} D_{pqkl}}{\frac{\partial f}{\partial \sigma_{rs}} D_{rstu} \frac{\partial f}{\partial \sigma_{tu}}} \tag{5-32}$$

Using the same assumptions as for the hardening constitutive models, the elastic plastic stiffness tensor can be simplified to:

$$[D]_{ep} = [D] - \frac{[D]\{S\}\{S\}^T[D]}{\{S\}^T[D]\{S\}} \tag{5-33}$$

The final form is written is not written in index notation to indicate that the elasticity matrices and deviatoric stress tensor are stored in vector notation.

5.1.3 Plastic Stiffness Matrix

Using the updated elastic-plastic stiffness matrix derived previously, the plastic stiffness matrix can be determined using a procedure similar to the original derivation of the stiffness matrix of the structure. Since not every element within the structure will necessarily experience yielding at the same time, the full stiffness problem does not need to be recomputed. Only the elements that are experiencing plastic deformation contribute to the creation of the plastic stiffness matrix of the structure with the decomposition defined:

$$[K]_{ep} = [K]_e + [K]_p \tag{5-34}$$

This corresponds to the decomposition of the material stiffness matrix for an elastic-plastic problem as:

$$[D]_{ep} = [D]_e + [D]_p \tag{5-35}$$

From the relationship derived in (5-21), the plastic contribution to the elastic-plastic material stiffness matrix is:

$$[D]_{ep} = [D] - \frac{[D]\{S\}\{S\}^T[D]}{A + \{S\}^T[D]\{S\}} \tag{5-36}$$

where D is defined to be the elastic material stiffness matrix. For hardening materials, the scalar variable A is determined from (5-20). For perfectly plastic constitutive models, the scalar variable A is zero.

The plastic stiffness for each element can be derived according to the procedure used to generate the elastic stiffness matrix with a slight modification. The elastic material stiffness matrix is replaced with the plastic decomposition of the elastic-plastic material stiffness matrix to become:

$$[k]_p^{element} = \int\int\int [B]^T [D]_p [B] dxdydz \tag{5-37}$$

Or in terms of isoparametric coordinates:

$$[k]_p^{element} = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} [B]^T [D]_p [B] |J| d\xi d\eta d\zeta \tag{5-38}$$

The elemental stiffness matrices are assembled into a global plastic stiffness matrix for later use in the determination of the global elastic-plastic stiffness matrix.

5.2 Coupling of Linear and Non-linear Responses

In order to combine the linear and non-linear responses, the plastic deformation must be induced through a method that can be superimposed on the elastic response. One potential method of inducing the plastic deformation in an equivalent elastic problem is by imposing a pseudoforce. The pseudoforce is a derived force that produces the required plastic deformation when superimposed on the elastic problem.

5.2.1 Determination of Static Pseudoforce

The primary input for the creation of the pseudoforce for the static condition is the global elastic-plastic stiffness matrix, which is determined by integrating over the elements using the elastic-plastic material stiffness matrix. Everything is ultimately derived from the incremental plastic deformation determined by the plasticity algorithm. In order to have minimal impact, or computation cost, on the linear elastic solution procedure, only the pseudoforce acts on the original linear elastic problem directly. Within the Newmark-Beta incremental integration algorithm, the pseudoforce will be translated into nodal displacements through multiplication with the inverse of the global elastic stiffness matrix. The inertia and damping matrices also affect the translation from

force to displacement but they will be discussed later. The goal is to determine the incremental force, which satisfies the relationship:

$$[K_e]^{-1}\{df_r\} = \{du_p\} \tag{5-39}$$

where K_e is the global elastic stiffness matrix, df_r is the unknown increment of the pseudoforce, and du_p is the increment of nodal deformation produced by the plastic deformation of the elements. Equation (5-39) can be solved for the increment of the pseudoforce as:

$$\{df_r\} = [K_e]\{du_p\} \tag{5-40}$$

The incremental nodal deformation caused by plastic deformation of the elements is unknown but can be determined by starting with a decomposition of the incremental nodal displacement:

$$\{du_{ep}\} = \{du_e\} + \{du_p\} \tag{5-41}$$

Solving for the incremental plastic deformation yields:

$$\{du_p\} = \{du_{ep}\} - \{du_e\} \tag{5-42}$$

The incremental nodal deformation caused by the elastic deformation of the element is already known because it was used to determine the plastic deformation, but the elastic-plastic nodal deformation is unknown. This incremental deformation can be determined by examining the static portion of the equation of motion for the elastic-plastic deformation:

$$[K_{ep}]\{du_{ep}\} = \{df_{ep}\} \tag{5-43}$$

However, the incremental elastic-plastic nodal force is equivalent to the incremental elastic nodal force, because plastic deformation does not change the physical external loads. Equation (5-43) can be rewritten as:

$$\left[K_{ep}\right]\left\{du_{ep}\right\} = \left\{df_e\right\} \tag{5-44}$$

Since every term except the incremental elastic-plastic nodal deformation is known, the relationship can be written:

$$\left\{du_{ep}\right\} = \left[K_{ep}\right]^{-1}\left\{df_e\right\} \tag{5-45}$$

All of the required unknowns have been determined to solve for the incremental pseudoforce. Using (5-42) and (5-45), equation (5-40) becomes:

$$\left\{df_r\right\} = \left[K\right]\left(\left[K_{ep}\right]^{-1}\left\{df_e\right\} - \left\{du_e\right\}\right) \tag{5-46}$$

where the global elastic-plastic stiffness matrix is defined:

$$\left[K_{ep}\right] = \left[K\right] + \left[K_p\right] \tag{5-47}$$

The incremental pseudoforce can then be introduced into the Newmark-Beta solution procedure as a nodal force. The pseudoforce is not a physical external force acting on the nodes but is only used to produce the required nodal deformations predicted by the plasticity algorithm. If the nodal deformations caused by both elastic and plastic deformation are determined in the same Newmark-Beta increment, the pseudoforce can be added to the external nodal forces through superposition. At least one additional iteration of the time step is required to determine the elastic deformation of the increment followed by additional iterations to converge the implicit solution of the equations of motion across the time step.

5.2.2 Determination of Dynamic Pseudoforce

A similar procedure can be used to develop the plastic pseudoforce required for the more general dynamic condition. If plasticity theory predicts that the element has yielded during the time step, additional calculations are required to determine the appropriate nodal displacements within the substructure. The first iteration of the time step is always assumed to be elastic, but if the effective stress within any element has exceeded the yield strength of the material then additional iterations will be required. Special consideration is needed for elements that have just yielded during the current time step because a portion of the deformation during that time step will likely be elastic and a portion will be plastic. The ratio of the incremental stress that contributes to plastic deformation within the time step is:

$$R = \frac{\sigma_e - \sigma_Y}{\Delta\sigma_e} \tag{5-48}$$

where σ_e is the effective stress defined in (4-10), σ_Y is the material yield stress, and $\Delta\sigma_e$ is the incremental effective stress within the current iteration. A portion of the incremental stress causes only elastic deformation, where the components are assumed to be proportional to the original incremental components:

$$\{d\sigma_r\} = (1-R)\{d\sigma\} \tag{5-49}$$

Any remaining portion of stress is contributing to elastic-plastic deformation and must be converged to the yield surface of the material model. For this work, a Newton-Raphson technique was employed to determine the appropriate value of the plastic multiplier, $d\lambda$, to satisfy the yield function in equation (4-14).

After the first iteration of the equations of motion, any elements experiencing plastic deformation require a modification to the vector of the internal resisting forces to produce

the appropriate deformation in further iterations. Based on the elastic prediction and the elastic-plastic material stiffness matrix, the modified components of stress within the element are given by:

$$\{\sigma\} = \{\sigma_p\} + \{d\sigma_r\} + R\left[D_{ep}\right]\{d\varepsilon_e\} \tag{5-50}$$

where σ_p are the converged components of stress at the previous time step, and $d\varepsilon_e$ is incremental elastic strain predicted during the first iteration of the equations of motion at the current time step. These components of stress are used to determine the corrected internal resisting force for the next iteration at the current time step. The internal resisting force of an element, in incremental form, is given by:

$$\{dp\}_e^{(k)} = \int_V [B]^T \{d\sigma\} dV \tag{5-51}$$

This integration is repeated for each element within the substructure and assembled into the global internal resisting force vector. This vector as well as the external force vector is defined in the global coordinate system but the remainder of the residual forces are defined in the reduced modal coordinate system after the first elastic iteration. The portion of the global effective force vector defined in terms of the information from the previous converged time step is calculated using a transformation of the constant acceleration and velocity vectors:

$$\{b_g^*\}^{(k)} = [M][P_u]\{\ddot{q}^*\} + \left[C_{ep}\right][P_u]\{\dot{q}^*\} \tag{5-52}$$

This term is calculated prior to the first elastic-plastic iteration and used for subsequent elastic-plastic iterations of the current time step. The modal displacements from all previous iterations at the current time step also produce inertial and damping forces that are given by:

$$\left\{\tilde{b}_g\right\}^{(k)} = \left(\frac{1}{\Delta t^2 \beta}[M] + \frac{\gamma}{\Delta t \beta}\left[C_{ep}\right]\right)[P_u]\{\Delta q\}^{(k-1)} \tag{5-53}$$

The conversion of the incremental deformation of the elastic iteration can be determined during the process of calculating the elemental stresses to reduce computational expense. The global effective force vector for the current iteration is written:

$$\left\{b_g\right\}^{(k)} = \{f\}_{n+1} - \{p\}^{(k)} - \left\{b_g^*\right\}^{(k)} - \left\{\tilde{b}_g\right\}^{(k)} \tag{5-54}$$

This effective elastic-plastic force can be used in conjunction with an initial stiffness method and iterated until convergence is achieved. Alternatively, the force can be converted to a form that follows the techniques of a tangential stiffness method by:

$$\left\{df_{ep}\right\}^{(k)} = \left[K^*\right]\left[K_{ep}^*\right]^{-1}\left\{b_g\right\}^{(k)} \tag{5-55}$$

where the effective initial and appropriate elastic-plastic stiffness matrices are employed. Combined tangential and initial stiffness methods will reduce the number of operations required for future iterations if the curvature of the material hardening curve is large. The effective elastic-plastic force in (5-55) is defined in terms of the global coordinate system but is projected on the modal coordinates using:

$$\left\{df_q\right\} = [N]^T [\Phi]^T [P]\left\{df_{ep}\right\} = \left[P_f\right]\left\{df_{ep}\right\} \tag{5-56}$$

The effective elastic-plastic force applied in the modal coordinate system can be input directly in the elastic CMS solution procedure defined previously for all remaining iterations.

This process of forming the effective force vector is similar to the mode acceleration method because the static contributions to the problem are retained without reduction while the inertia and damping contributions are determined from a modal representation. The mode acceleration method has been employed as a technique to increase the

accuracy of a mode superposition method without increasing the number of retained modes [70]. The conversion of the effective force from a global to a reduced modal representation will result in force components that are not represented. A sufficient number of modes must be retained in order to adequately represent the force of the plastic deformation. Additional techniques of recovering the un-projected force are also discussed in Chapter 6.

5.3 Iteration of Plastic Response

The first iteration of each time step is always assumed to be elastic because it is not known whether the elements within the structure are experiencing elastic, plastic, or elastic-plastic deformation. The deformation for the first iteration is determined based on the original stiffness matrix and corrected in later iterations. Upon evaluating the stress after the first iteration, plasticity theory will indicate whether any of the elements have experienced plastic deformation. There are two basic approaches for determining the deformation in the subsequent iterations, which are the initial and tangential stiffness methods. Both methods are iterative techniques and have their own respective advantages and disadvantages.

5.3.1 Tangential Stiffness Method

The tangential stiffness method is based on the assumption that all subsequent iterations of a numerical solution are predicted using the slope at the current iteration. The function is linearized from one iteration to the next, which requires the time step to be sufficiently small. For a monotonically increasing function of a single variable, the solution algorithm will converge according to the steps in Figure 5-1. On the first

iteration, the solution is predicted based on the current slope of the function. Note that the curve is just transitioning from a linear to non-linear during this iteration. If the function had remained linear, the prediction from the first iteration would have been correct and not required further iteration. However, the function is now non-linear and a convergence check indicates that the stress is lower than predicted. On the second iteration, the local slope is determined at the actual stress and strain from the first iteration. The result of the second iteration provided a prediction that is much closer to the correct solution but the stress is still over-predicted. The same process is repeated for the third iteration, which results in a prediction that is nearly equivalent to the true solution.

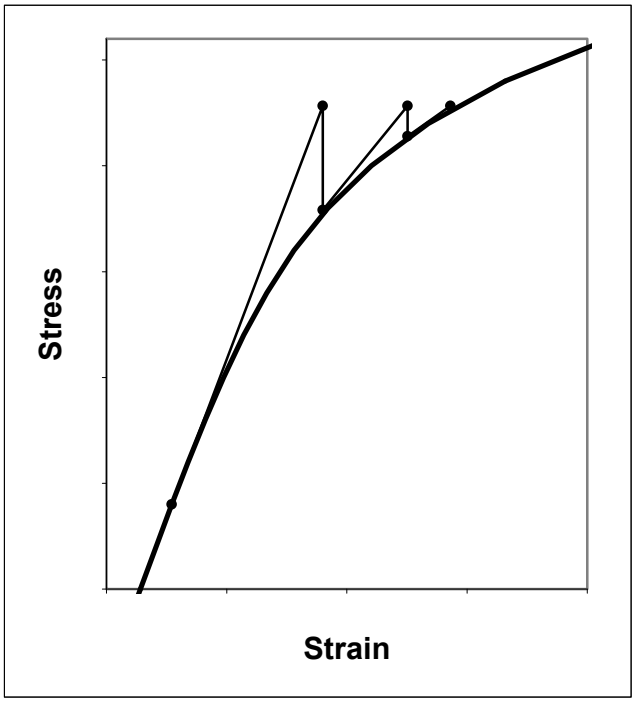

Figure 5-1: Tangential Stiffness Method – Power law hardening

The number of iteration required to achieve convergence is dependent on the curvature of the function and the convergence tolerance. A tight convergence tolerance will produce an accurate result but will typically require more iterations. A loose convergence tolerance will produce a less accurate result but will typically achieve convergence in fewer iterations. For the special case of a bi-linear function, convergence will be achieved in the second iteration because the tangential slope will be equal to the constant slope of the function as shown in Figure 5-2. This is the primary reason that the assumption of a bi-linear material hardening model is computationally cheaper in an elastic-plastic analysis.

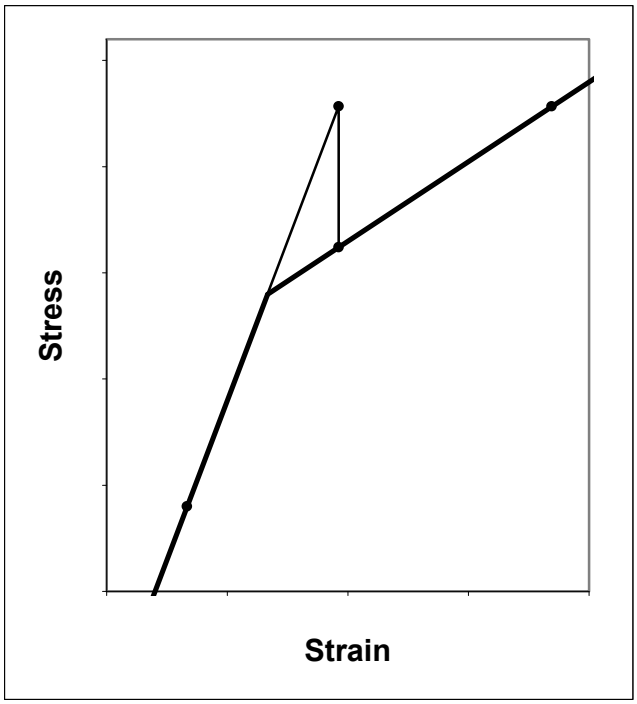

Figure 5-2: Tangential Stiffness Method – Bi-linear hardening

The primary advantage of the tangential stiffness method is that convergence can be achieved with relatively few iterations. A new slope is calculated with each iteration that

improves the accuracy of the prediction for the next iteration. The prediction is only in error by the change in slope that occurs after the current prediction. However, the primary disadvantage of the tangential stiffness method is that the slope must be recalculated at each iteration. For a single variable function, the computational cost is insignificant, but this corresponds to recalculating the elemental stiffness matrix for the finite element method, which is rather computationally expensive. Any element that experiences yielding requires the calculation of a new elastic-plastic stiffness matrix and elemental stiffness matrix for each plastic iteration.

5.3.2 Initial Stiffness Method

The initial stiffness method is based on the assumption that the original stiffness matrix is used for all iterations until convergence is achieved. This method does not require that the slope of the function be determined because the original slope is used for all predictions. The iterative procedure is linearized between time steps but without requiring any further knowledge of the function.

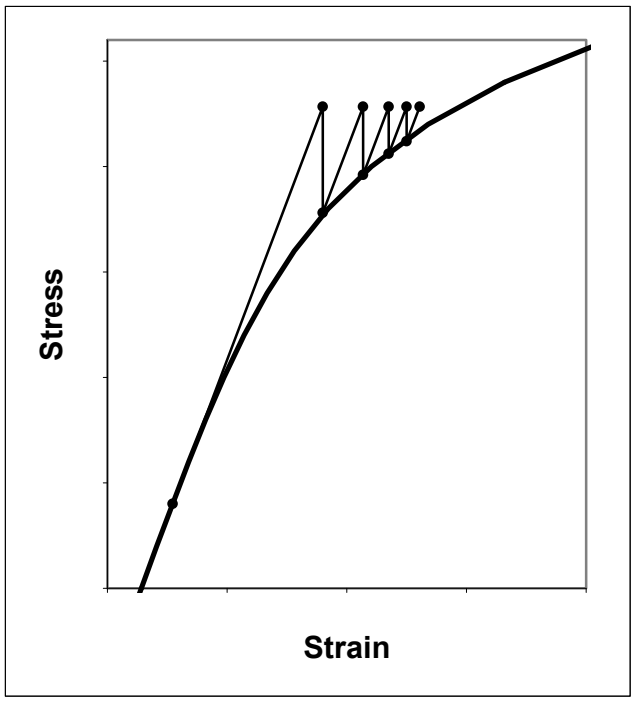

Figure 5-3: Initial Stiffness Method – Power law hardening

For a function of a single variable, the algorithm to achieve convergence is shown in Figure 5-3. After the first iteration, the stress is over-predicted because the slope of the curve has decreased after passing the yield point of the material. The second and following iterations continue to over-predict the function because the actual slope of the curve is always smaller than the prediction slope. With enough iterations, the predictions will converge on the actual solution within the convergence limits defined.

The convergence for a bi-linear hardening model is shown in Figure 5-4. Since the slope of the hardening function immediately changes after the yield point, the initial iterations do not converge as quickly as for the power law hardening function. The average slope of the bi-linear hardening function throughout the time step is smaller than the power law hardening function and the iterations do not converge as quickly.

However, if extended into the softening portion of the curve, the initial stiffness method would converge as quickly as the tangential stiffness method.

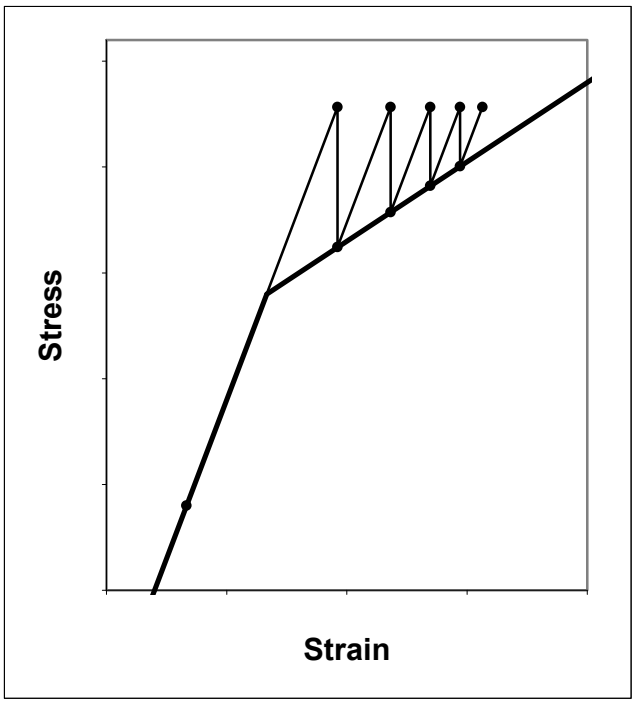

Figure 5-4: Initial Stiffness Method – Bi-linear hardening

The primary advantage of the initial stiffness method is that the elastic-plastic stiffness matrix does not need to be recalculated for each plastic iteration. The original stiffness matrix is used throughout all iterations without modification. The primary disadvantage is that more convergence iterations are typically required with the initial stiffness method because the successive iterations are predicted without updated information about the function.

5.3.3 Combination of Tangential and Initial Stiffness Methods

For this dissertation, a combination of the tangential and initial stiffness methods is employed. When plastic deformation is predicted after the first elastic iteration, the

updated elastic-plastic stiffness matrix is calculated. Only the elements that experienced plastic deformation within the time step require a modified elemental stiffness matrix. This typically minimizes the computation cost of the formulation because plastic deformation is usually a localized effect. For the second iteration of the equations of motion, the elastic-plastic stiffness matrix is used to update the predicted displacement. For all subsequent iteration until convergence is achieved, the same elastic-plastic stiffness matrix is used without modification. This usually achieves convergence quickly because the largest improvement is achieved with the prediction from the second iteration.

5.4 Elastic Response Following Plastic

For structures that are loaded with an impulse function or a function that decreases over time, the elastic-plastic time steps will be followed by purely elastic behavior. Once a structure is loaded, the simulation typically occurs for a longer period of time to investigate how the structure responds and recovers from the load. With the elastic-plastic CMS method outlined, the subsequent elastic time steps can proceed without further computational burden in the iterative solution of the equations of motion. Since the problem is solved iteratively, an incremental pseudoforce is added to induce the required plastic deformation. On subsequent iterations, this incremental force remains and maintains the permanent set associated with plastic deformation.

In a cyclic loading condition, the structure may experience a time period with plastic deformation in tension followed by plastic deformation in compression. This type of loading results in the Bauschinger effect and can be numerically modeled using a kinematic strain hardening function [64]. For this dissertation, only an isotropic

hardening model was incorporated because the primary goal was to model impulse loading that decay with time after reaching a peak load.

5.5 Convergence Check

A test of the convergence of the iteration is applied to ensure that the result within the time step has converged within the defined tolerance. This check is used to determine whether the equations of motion have adequately converged on the correct solution or indicate that more iterations are required prior to completing the time step. There are many options in convergence checks that are based on the change in displacement, velocity, acceleration, or force. The convergence algorithm basically calculates the change in some measure from one iteration to the next. If the checked variable is not changing, or not changing more than the defined tolerance, then the incremental solution is considered converged. Alternatively, the residual variables can be used as the check of convergence. If the residuals are less than the defined tolerance, then the solution is considered converged. For this research, the effective residual force was used as the convergence check variable for time steps that experience elastic-plastic deformation.

Chapter 6

Characteristics of Non-linear Method

Finite element analysis is a technique of approximating a complex structure as a collection of individual elements. The accuracy of the approximation is dependent on various input variables inherent in the method. With the proposed non-linear CMS method, additional approximations of the full fidelity finite element solution are applied. Each assumption in the proposed framework must be clearly understood or the accuracy of the solution could be very poor. Given that the correct assumptions are applied, the modal approach will produce a reasonable accuracy with substantial improvements in computational efficiency.

Adequate assumptions must be applied in the original finite element formulation of the problem as well. In a full fidelity finite element solution, the accuracy can be poor if the mesh of the structure is too course. The typical way of addressing this problem is to model the structure with a slightly finer mesh and comparing with the previous results. If a substantial change occurs in the results, then the mesh is probably not optimized. If relatively little change occurs in the results, then the mesh density is probably acceptable. Similar additional characteristics are observed with the modal techniques.

6.1 Accuracy of Non-linear CMS Method

The accuracy of the CMS prediction is determined by several factors. Starting with the original mesh of the structure, an initial inertia and stiffness matrix are generated. If these matrices do not adequately represent the structure, then the CMS techniques will do nothing to increase the accuracy of the solution. If the modal methods are properly applied, the results will not be degraded from those obtained with the full fidelity solution. Therefore, in all following example problems, the baseline for accuracy comparison is the results obtained from the full fidelity solution with the original inertia and stiffness matrices of the structure.

The accuracy of the plastic prediction is limited by the assumptions applied in the modal reduction. If too few modes are retained, the accuracy of the elastic response will suffer, which will result in an inaccurate prediction of the plastic response. Techniques, such as the modal mass participation factor, should be investigated to aid in the determination of the number of modes to retain. Small convergence studies should also be applied in a similar fashion to those used to determine an appropriate mesh density. For a modal technique, there is an additional advantage in knowing the natural frequencies of the structure. Inherent in the generation of the transformation matrices, the un-restrained natural frequencies of the structure are provided. With information about the forcing frequencies that will be applied to the structure, a reasonable approximation is to retain all modes within the forcing frequency and below as a minimum. Since the structure will most likely be attached to other piece parts with the model, the additional mass will result in an increase in the natural frequencies of the assembly, so the un-restrained modal information can only be used as a rough guide.

6.2 Methods of Improving Accuracy

The accuracy of the non-linear CMS method is determined by variables that are chosen by the person performing the analysis. Just as with any finite element problem, an appropriate mesh must be generated to provide the numerical representation of the structure. If the mesh is too course, the accuracy of the solution may not be adequate. The elemental formulation is important in determining the global stiffness matrix of the structure that does not introduce zero energy displacement modes.

With the non-linear CMS method, the accuracy of the solution is also dependent on the number of retained modes. The numerical efficiency of a modal representation is obtained by neglecting a large portion of the higher frequency modes. The number of modes retained will directly effect the accuracy of the solution for specific loading frequencies, because the modes that are not retained are not represented in the solution. The exception is that some of the flexibility can be recovered with a residual flexibility calculation, which can represent the static contribution of the modes that were originally neglected.

6.2.1 Constraint and Retained Modes

With the fixed interface CMS technique, all boundary degrees of freedom are retained without reduction. This is typically a disadvantage of the method if the structure is divided into too many substructures. The reduction only occurs on the interior degrees of freedom, so no computational benefit is realized by the boundary degrees of freedom. From this point of view, the number of boundary degrees of freedom should be minimized to critically evaluated divisions in the structure.

The constraint modes of the substructure are calculated through a static condensation procedure so the dynamic contribution of the boundary degrees of freedom is not retained. This is typically not an issue because the constraint will either be applied at a joint that is truly fixed in the assembly or at the connection to another free body. At the junction of two flexible substructures, negligible information is lost if the constraints are applied correctly. The number of retained interior modes directly corresponds to the accuracy of the solution. If too few modes are retained, the accuracy of the result will suffer. However, for simple structures, only a very few modes of the full fidelity model require retention. The boundary degrees of freedom will already be retained but sufficient information is required of the internal degrees of freedom. The number to be retained is somewhat problem specific, which is investigated in the example problems that follow.

More retained modes are required for an elastic-plastic solution than would typically be required to obtain an accurate elastic solution. Plastic deformation typically occurs as a more localized effect that is represented by the higher frequency mode shapes. As the plastic pseudoforce is projected onto the modal coordinates, a significant portion of the force can remained un-projected if too few modes are retained. This force will be lost within the increment and cannot be recovered in subsequent iterations. Retaining all modes will produce the most accurate solution but will also be the most computationally expensive, so a balance must be sought. However, prior to the completion of the time step, a residual flexibility technique can be applied to help recover the un-projected force and improve the accuracy of the time step.

6.2.2 Residual Flexibility

Modal reduction techniques are used to reduce the number of calculations in the equations of motion with minimal impact on the accuracy of the solution. Any time there is a reduction applied, information about the response of the degrees of freedom is being neglected. If the problem is set up adequately and an adequate number of modes are retained, then the impact on the accuracy of the solution will be minimal. However, if too few modes are retained, the accuracy will suffer. This is comparable to reducing the mesh density of a finite element representation. As the mesh becomes more course, the number of internal degrees of freedom and flexibility of the structure may not be accurately represented, which will adversely affect the accuracy of the solution.

Modal reduction techniques are applied by eliminating the largest eigenvalues and corresponding eigenvectors of the full eigenproblem for the un-damped equation of motion. In a typical structural problem, the natural frequencies activated as a result of a forcing function are the lowest frequency modes. The highest frequency modes can be many orders of magnitude larger and as a result will have a much smaller contribution to the dynamic response of the structure. The cut-off frequency is dependent of the problem of interest but the each of the modes below the forcing frequency and some above the forcing frequency will need to be retained to accurately represent the response of the structure. Often, it is difficult to the determine how many modes to retain, which can be a source of inaccuracy. Tools such as determining he modal participation factors, Section 3.4, can provide an indication of the importance of the modes but is independent of the forcing function.

Applying forces to approximate the effects of plastic deformation can further decrease the accuracy of the solution. Since the plastic deformation will most likely be a localized effect, it will have more contribution from higher frequency modes than required for the elastic solution. When used in conjunction with a modal reduction technique, the plastic pseudoforce is projected onto the modal coordinates from the global coordinates through the use of a projection matrix. Since some of the modes are neglected in this projection, some of the force will not be projected on the modal coordinates. If no further action is taken, this un-projected force will be lost and cannot be recovered.

Residual flexibility is a means of retaining some of the un-projected force to improve the accuracy of the solution. The basic premise is to incorporate the static effects of the un-projected force in the final solution for the iteration. Further information on experimental validation can be found in [71]. A comparison of residual flexibility and CMS techniques can be found in [72]. Information is being lost with any reduction technique, but since the inertia and damping effects are typically less important that the stiffness, the solution accuracy can be dramatically improved. Starting with the equations of motion of a multi-body system:

$$[M]\{\ddot{u}\}+[C]\{\dot{u}\}+[K]\{u\}=\{f\} \tag{6-1}$$

An incremental form of the equations of motion is required for a plastic solution, requiring the conversion:

$$[M]\{\ddot{u}\}+[C]\{\dot{u}\}+[K]\{\Delta\Delta u\}=\{f\}-\{p\} \tag{6-2}$$

The first step in the conversion to a CMS representation is to partition the matrices into the internal, boundary, and interacting degrees of freedom. A transformation matrix, P, is

used to transform the vector of displacements into the partitioned form. The nodal displacements and their derivatives are replaced with:

$$[M][P]\{\ddot{u}\}+[C][P]\{\dot{u}\}+[K][P]\{\Delta\Delta u\}=\{f\}-\{p\} \tag{6-3}$$

Pre-multiplying by the transpose of the transformation matrix produces:

$$[P]^T[M][P]\{\ddot{u}\}+[P]^T[K][P]\{\Delta\Delta u\}=[P]^T\left(\{f\}-\{p\}\right) \tag{6-4}$$

For convenience, this will be written:

$$[M_P]\{\ddot{u}_P\}+[K_P]\{\Delta\Delta u_P\}=[P]^T\left(\{f\}-\{p\}\right) \tag{6-5}$$

Applying the CMS representation, the partitioned coordinate set is transformed to a CMS representation by applying the appropriate CMS transformation matrix. This transforms the organized global coordinate set to a reduced modal basis that has fewer modes than the original number of degrees of freedom for the substructure. The lower frequency modes are retained and the higher frequency modes are neglected. The application of the transformation is included by replacing the ordered coordinate set with the appropriate transformation:

$$[M_P][\Phi]\{\ddot{u}_{CMS}\}+[K_P][\Phi]\{\Delta\Delta u_{CMS}\}=[P]^T\left(\{f\}-\{p\}\right) \tag{6-6}$$

The equation is pre-multiplied by the CMS transformation matrix to obtain:

$$[\Phi]^T[M_P][\Phi]\{\ddot{u}_{CMS}\}+[\Phi]^T[K_P][\Phi]\{\Delta\Delta u_{CMS}\}=[\Phi]^T[P]^T\left(\{f\}-\{p\}\right) \tag{6-7}$$

The pre- and post-multiply by the transformation matrix effectively transforms the inertia and stiffness matrices and those variables can be replaced with a transformed variable:

$$\left[\bar{M}\right]\{\ddot{u}_{CMS}\}+\left[\bar{K}\right]\{\Delta\Delta u_{CMS}\}=[\Phi]^T[P]^T\left(\{f\}-\{p\}\right) \tag{6-8}$$

As discussed previously, the modified inertia and stiffness matrices are not diagonal because the transformation matrix is not and eigenvector transformation. It is

computationally more efficient in a dynamic simulation to convert the inertia and stiffness matrices to an orthonormal form to reduce the coupled equations of motion to an uncoupled set. This is accomplished by solving the eigenproblem and replacing the CMS coordinate system with the orthonormal set of coordinates through the appropriate transformation:

$$\left[\bar{M}\right][N]\{\ddot{q}\}+\left[\bar{K}\right][N]\{\Delta\Delta q\}=[\Phi]^T[P]^T\left(\{f\}-\{p\}\right) \tag{6-9}$$

To complete the transformation, both sides of the equation are pre-multiplied by the transpose of the transformation matrix to obtain:

$$[N]^T\left[\bar{M}\right][N]\{\ddot{q}\}+[N]^T\left[\bar{K}\right][N]\{\Delta\Delta q\}=[N]^T[\Phi]^T[P]^T\left(\{f\}-\{p\}\right) \tag{6-10}$$

In the generation of the eigenvector transformation matrix, the terms can be scaled to produce an inertia matrix that takes the form of an identity matrix with all terms equal to one on the diagonal and zero in all other positions. This also produces a stiffness matrix equal to the square of the natural frequencies located along the diagonal. The converted matrices are written:

$$[I]\{\ddot{q}\}+[\Lambda]\{\Delta\Delta q\}=[N]^T[\Phi]^T[P]^T\left(\{f\}-\{p\}\right) \tag{6-11}$$

This equation represents the solution if all degrees of freedom are retained in the CMS transformation. Retaining all eigenvectors will result in the full accuracy of the original solution because only transformations have been applied without any reduction to cause loss in accuracy. However, the purpose of the CMS technique is to apply a reduction to improve the computational efficiency and all eigenvectors will not be retained in the CMS transformation matrix. The reduced set of equation will then be represented with a subscript variable term:

$$[I]\{\ddot{q}_R\}+[\Lambda_R]\{\Delta\Delta q_R\}=[N]^T[\Phi_R]^T[P]^T(\{f\}-\{p\}) \tag{6-12}$$

For convenience, the series of force transformation matrices are replaced with a single transformation matrix:

$$[I]\{\ddot{q}_R\}+[\Lambda_R]\{\Delta\Delta q_R\}=\left[P_f\right](\{f\}-\{p\}) \tag{6-13}$$

This equation is solved for the incremental displacement in the modal coordinate system:

$$\{\Delta\Delta q_R\}=-[\Lambda_R]^{-1}[I]\{\ddot{q}_R\}+[\Lambda_R]^{-1}\left[P_f\right](\{f\}-\{p\}) \tag{6-14}$$

In the conversion from the modal representation to global coordinates, the transformation is written:

$$\{\Delta\Delta u\}=[P][\Phi][N]\{\Delta\Delta q_R\}=[P_u]\{\Delta\Delta q_R\} \tag{6-15}$$

A single transformation matrix has been applied for convenience. The complete solution of the incremental modal coordinates is converted to a global representation:

$$\{\Delta\Delta u\}=[P_u][\Lambda_R]^{-1}\left[P_f\right](\{f\}-\{p\})-[P_u][\Lambda_R]^{-1}\left(\{\ddot{q}_R\}+\left[\bar{C}\right]\{\dot{q}_R\}\right) \tag{6-16}$$

Referring back to the original incremental equation of motion, the incremental displacement can be found after rearranging:

$$\{\Delta\Delta u\}=[K]^{-1}(\{f\}-\{p\})-[K]^{-1}([M]\{\ddot{u}\}+[C]\{\dot{u}\}) \tag{6-17}$$

By examining the first term of the full incremental solution and the solution obtained from the modal coordinates, it can be seen that the first term represents the static response to the applied incremental force. The solution can be improved by replacing the static contribution from the full solution with the modal representation because the full solution does not incorporate any reduction:

$$\{\Delta\Delta u\}=[K]^{-1}(\{f\}-\{p\})-[P_u][\Lambda_R]^{-1}\left(\{\ddot{q}_R\}+\left[\bar{C}\right]\{\dot{q}_R\}\right) \tag{6-18}$$

This method is defined as the mode acceleration method. The residual flexibility, or the difference in the modal and full solution can be found by additional algebraic manipulation. The modal representation is rewritten:

$$\left(\{\ddot{q}_R\}+\left[\bar{C}\right]\{\dot{q}_R\}\right)=\left[P_f\right]\left(\{f\}-\{p\}\right)-[\Lambda_R]\{\Delta\Delta q_R\} \tag{6-19}$$

Pre-multiply both sides by the inverse of the modal stiffness matrix and the combined transformation to obtain:

$$[P_u][\Lambda_R]^{-1}\left(\{\ddot{q}_R\}+\left[\bar{C}\right]\{\dot{q}_R\}\right)=[P_u][\Lambda_R]\left[P_f\right]\left(\{f\}-\{p\}\right)-[P_u]\{\Delta\Delta q_R\} \tag{6-20}$$

Applying the pre-multiplication produces a form of the inertia and stiffness and damping terms that match the inertia and damping term in the mode acceleration method result. This produces an equation that is written entirely in terms of the displacement terms and not the velocity or acceleration terms:

$$\{\Delta\Delta u\}=[K]^{-1}\left(\{f\}-\{p\}\right)-[P_u][\Lambda_R]^{-1}\left[P_f\right]\left(\{f\}-\{p\}\right)+[P_u]\{\Delta\Delta q_R\} \tag{6-21}$$

The third term is the incremental displacements transformed from the modal coordinates. The first and second term are the static contributions to the incremental displacements in global coordinates. The first terms represents the solution obtained by retaining the full stiffness matrix, while the second term represents the static contribution through the reduced modal stiffness matrix. The difference in these two terms is the residual static incremental displacement that is added to the calculated displacement from the reduced equations of motion.

As the solutions are obtained at each increment during the integration of the equations of motion, the residual flexibility term must be calculated. It is an additional operation that is performed after the reduced solution has converged. The residual term can be simplified to improve computational efficiency by combining the terms prior to

the integration of the equations of motion. Ultimately the residual flexibility can be calculated using one matrix vector multiplication by introducing a new variable, R_f:

$$\left[R_f\right] = [K]^{-1} - [P_u][\Lambda_R]^{-1}\left[P_f\right] \tag{6-22}$$

This matrix is calculated once prior to the integration of the equations of motion and is used throughout the integration to account for the static contribution from the un-projected force.

6.2.3 Residual Flexibility in Plastic Solution

If plasticity is incorporated with the CMS representation, the calculation of the residual flexibility become more important. This is due to the application of forces to individual nodes that were not significant contributor to the accuracy of the elastic solution. Once plastic deformation begins to occur, the deflection of the substructure begins to deviate from the motion predicted as a linear superposition of the lowest few frequencies. The deformation becomes highly localized, which requires the use of the higher frequency mode shapes that are typically not significantly activated during a linear-elastic solution. The inertia and damping terms will not retain the full terms and subsequently their accuracy is not improved. This is typically not a significant problem, because the inertia and damping are secondary effects to the static.

There are options available in deciding when residual flexibility can or should be applied. If applied after the first elastic iteration is completed, the stresses will be more accurate. This leads to an improved prediction of the plastic response. However, since the solution is only being obtained incrementally, the elastic predictions should be quite accurate if enough of the low frequency modes are retained. This is more difficult to achieve for the plastic solution because a prohibitively high number of modes will

probably need to be retained to obtain the same accuracy that will be achieved with the inclusion of the residual flexibility. The residual flexibility has a relatively large effect on the plastic solution but a relatively small effect on the elastic solution. For this dissertation, the residual flexibility is only calculated after the plastic iterations. This helps preserve the computational benefits of the elastic iterations, which should comprise the vast majority of the complete set of calculations.

6.3 Computational Comparison

The computational benefits of the non-linear CMS formulation are attributed to the reduction applied as part of the CMS technique. The first reduction is achieved by neglecting the higher frequency modes and mode shapes. Since this information is not retained, the size of the equations solved at each time step are reduced from the full finite element representation. With a direct modal representation, it is very difficult to determine the specific modes that can be safely eliminated without dramatically affecting the accuracy of the solution. The fixed interface CMS method nearly eliminates this problem because the boundary nodes are retained without reduction and the number of kept modes must be determined to adequately represent the internal degrees of freedom.

The second computational benefit is produced by orthonormalizing the CMS representation. After transformation from the global coordinates to the CMS representation, the inertia and stiffness matrices are not diagonal because the transformation matrix is composed of the eigenvectors of only a partition of the original matrices. By orthonormalizing, the coupled equations of motion are reduced to an uncoupled set of equations that further reduces the computation expense by reducing the number of operations.

To quantitatively compare the computational expense of a full fidelity and a CMS representation, the number of multiplications were determined for each method. The multiplication operations are typically the most computationally expensive operations to perform, compared to addition operations [73]. Table 6-1 provides a comparison of the number of multiplications required in a single iteration of the equations of motion. The variable, n, represents the number of degrees of freedom in the unreduced structure. The variable, m, represents the total number of modes in the modal representation (retained as well as interface degrees of freedom). The computational benefits of storing the global stiffness and inertia matrices in a banded form in not included in this representation but would decrease the computational cost of the full solution. The ratio of m to n is dependent on the specific problem of interest, but generally, a ratio of 1 to 20 is more than sufficient.

Table 6-1: Computational Comparison for Dynamic Elastic Iteration

Variable	Full Solution	CMS Solution
a*	2 * n	2 * m
v*	2 * n	2 * m
b*	2 * n^2	2 * m^2
b	-	-
ΔΔu	n^2	m^2
Δu	-	-
u	-	-
a	n	m
v	n	m
p	n^2	m^2

The largest differences in the solution of the equations of motion for a purely elastic iteration are the number of operations required to determine the effective acceleration and velocity. Because the CMS representation is calculated as an uncoupled series of equations only one multiplication is required for each row of the inertia and stiffness

matrices. If the inertia and stiffness matrices are fully populated, the number of operations is equal to the square of the number of degrees of freedom. The remaining variables are also reduced by the ratio from n to m with the greatest impact provided by the squared terms.

The computational expense of the plastic portion of the dynamic response and the calculation of the stress within each element will be opposite of the elastic response due to the applied reduction. Since the solution is performed in an orthonormal coordinate set, the modal deformations must be transformed to a global representation to determine the state of stress within the elements. As indicated previously, this calculation can be performed as a single matrix-vector multiplication but still requires more operations than the equivalent full solution. Table 6-2 provides a summary of the number of operations required to determine the state of stress, elastic strain, and plastic strain within each element. The stress is calculated at each of the eight integration points for a 2 x 2 x 2 integration scheme. This table represents the number of operation required by assuming that all elements are experiencing plastic deformation, which is never really the case but provides a worst case scenario. The new variable introduced in this table is e, which is the number of elements in the finite element representation. Most of the number of operations are based on the number of elements because stress is determined on an elemental basis.

Table 6-2: Computational Comparison for Stress Calculation

Variable	Full Solution	CMS Solution
Stress	96 * e	96 * e + n * m
S	48 * e	48 * e
e_p	48 * e	48 * e
e_bar	48 * e	48 * e

The only difference in the number of operations in Table 6-2 is in the calculation of the elemental strain. Because the global displacements are not directly provided with each incremental solution, the global displacements must be calculated by using a transformation from modal to global coordinates. This transformation matrix will have one row corresponding to each global degree of freedom and one column corresponding to each modal degree of freedom.

The calculation of the plastic response for a single iteration is also more computationally expensive for a CMS representation than for a direct solution. This increase in the number of operation is attributed to the calculation of the pseudoforce and the projection of this global force on the orthonormal coordinates. Table 6-3 provides a summary of the number of operations required for each technique.

Table 6-3: Computational Comparison for Dynamic Elastic-Plastic Iteration

Variable	Full Solution	CMS Solution
Kep	290 * e	290 * e
b	-	n * m
ddu	n^3	n^3
p	n^2	$n^2 + n * m$

The primary differences in the number of operations is in the generation of the effective force and the generation of the pseudoforce. The projection of the elastic-plastic force onto the modal coordinates is achieved by a single matrix-vector multiplication. The matrix has one row corresponding to each modal degree of freedom and one column corresponding to each global degree of freedom. However, computational savings can be achieved by only projecting the rows of the global vector that corresponds to the degrees of freedom that were affected by plastic deformation within the iteration.

The calculation of the stress and determination of a single plastic iteration is more computationally expensive for the CMS representation than for a direct representation. However, the total number of operation is reduced for the elastic iterations that employ the modal representation. The balance will be dependent on the specific problem of interest, but if the structure primarily experiences elastic deformation, the non-linear CMS method will offer significant computational savings. This is typically the case for structures that experience an impulse loading followed by a long time interval to evaluate the response of the structure to the impulse. This type of problem results in a few elastic iterations, followed by relatively few plastic iterations, followed by many elastic iterations as the structure responds.

Chapter 7

Numerical Examples

A collection of numerical examples is provided to demonstrate the performance of non-linear response coupled with fixed interface CMS reduction. Two of the examples are based on problems that have analytical solutions to serve as a baseline for judging accuracy. The first example is a cantilever beam subjected to an axial load. Since this problem can be solved analytically, the response is investigated for the quasi-static loading condition as well as an impulse-loading situation. The second analytical problem is a simply supported beam loaded with a step pressure load. An analytical solution exists for this problem for the assumption of a perfectly-plastic yield function. A variation of the simply supported beam is presented with a pressure load only applied to the center portion of the beam rather than the full length. The final example is a rigid body mechanism that represents an intended use of the non-linear CMS method. Only one component within the mechanism, the shaft, is modeled as a non-linear flexible element while the other components remain rigid.

The accuracy of the non-linear CMS method is compared against the full fidelity solution using the full global inertia, stiffness, and damping matrices. Each of the two result types are calculated using the Matlab code provided in the appendices. Since the accuracy of the CMS method is dependent on the number of retained modes and other factors, convergence studies are presented to indicate the values required to obtain the

desired accuracy. Selected full fidelity dynamic solutions from the Matlab code is also compared with the dynamic solution from a commercial finite element code, ABAQUS (Version 6.8, Dassault Systemes Simulia Corp., Providence, RI). The model solved with the commercial code utilizes the same mesh of the geometric part and a similar elemental formulation to allow accuracy comparisons.

7.1 Quasi-Static Axial Loading

The axial stress and deflection of a cantilever beam can be determined analytically using the engineering stress and strain assumptions. The axial load is chosen to ensure that the deflections are small for both the elastic and plastic responses. For simplicity, the cantilever beam is chosen with a square cross-section with a 10 to 1 ratio of length to width. The dimensions of the cross-section of the beam are 0.1 in. by 0.1 in., and the length of the beam is 1.0 in. A representation of the beam is shown in Figure 7-1.

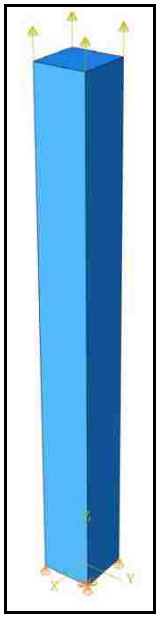

Figure 7-1: Cantilever Beam Geometry

The boundary conditions are applied such that the length and width of the beam are allowed to decrease as a result of the Poisson effect from the axial load. One corner of the beam is constrained in all three directions of the Cartesian coordinate system and all other nodes on the base of the beam are only constrained in the z-direction, which is the longitudinal axis of the beam. These constraints are comparable to an axi-symmetric constraint with the full constraints being applied along the longitudinal axis.

The properties of the beam are chosen as the common values for mild steel for simplicity. The elastic properties of the beam are listed in Table 7-1. The material is considered to be isotropic, requiring only two material properties to fully define the elastic material properties. The density is assumed to be 7.485 x 10^{-4} lbf-s^2/in^4 based on common values for mild steel. However, the assumed density is divided by 10^3 to ensure that inertia of the material does not contribute significantly to the quasi-static solution.

Table 7-1: Elastic Material Properties of Cantilever Beam

E	Young's Modulus	29,000	ksi
v	Poisson's Ratio	0.29	

The beam is subjected to loading in excess of the yield strength of the material, which induces plastic deformation. For ease of analytical calculation, it is assumed that the material follows a bi-linear hardening model for all stress in excess of the initial yield strength of the material. The plastic material properties are listed in Table 7-2.

Table 7-2: Plastic Material Properties of Cantilever Beam

H_0	Initial Yield Stress	36	ksi
E_T	Hardening Modulus	3,600	ksi

A graphic view of the elastic-plastic stress-strain curve is shown as Figure 7-2.

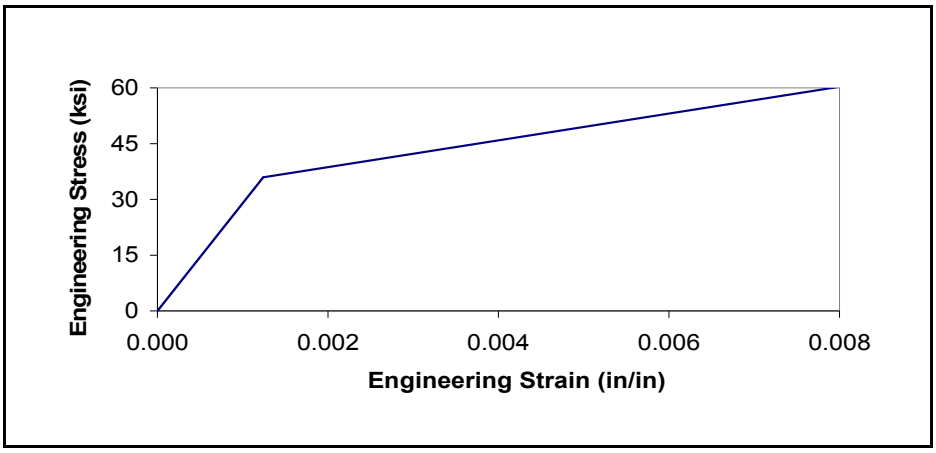

Figure 7-2: Elastic-Plastic Stress-Strain Curve

The goal of this dissertation is to introduce a dynamic solution procedure for non-linear deformation, but a quasi-static solution is beneficial to provide insight about the accuracy of the solution. For the dynamic solution, the axial load on the structure is applied as a time dependent function starting with the unloaded condition, loaded to the peak force, then unloaded back to zero. To prevent discontinuities during the loading and unloading, it is assumed that the curves followed a haversine shape. The full loading and unloading curve is shown as Figure 7-3.

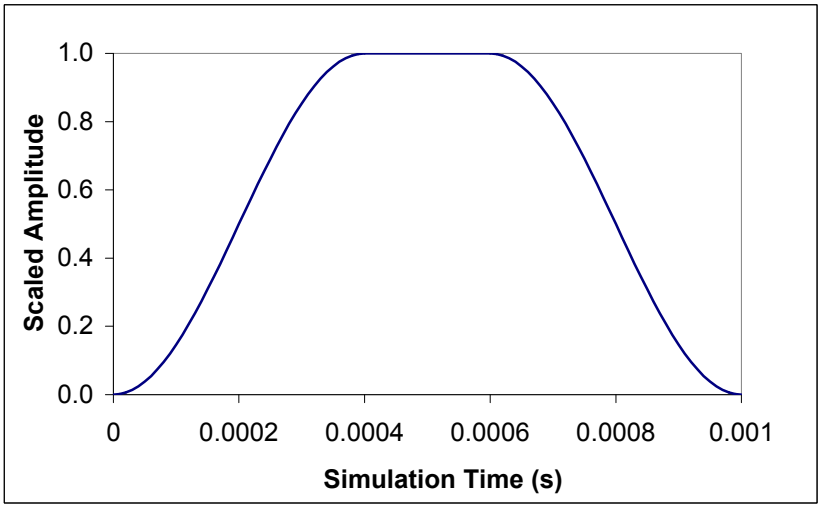

Figure 7-3: Load-Unload Scaled Amplitude

7.1.1 Analytical Solution

In order to determine the elastic-plastic deformation of the beam, the stress must be calculated first to determine the response of the material:

$$\sigma = \frac{P}{A} = \frac{400lbf}{(0.1in)^2} = 40ksi \tag{7-1}$$

Since the material is assumed to follow a hardening constitutive model, the elastic strain of the beam can be calculated as a function of the total stress within the structure. The elastic change in length of the beam is calculated:

$$\varepsilon_e L = \frac{\sigma L}{E} = \frac{40ksi(1in)}{29,000ksi} = 0.00137931in \tag{7-2}$$

Since the stress within the structure has exceeded the yield strength of the material, the plastic strain must also be determined. The material is assumed to follow a linear hardening law, so the yield strain is proportional to the stress in excess of the yield stress. The plastic change in length of the beam is written:

$$\varepsilon_p L = \frac{(\sigma - \sigma_Y)L}{E_T} \tag{7-3}$$

where E_T is the tangential hardening modulus. For the linear hardening assumption, the modulus is defined:

$$E_T = \frac{H_L - H_0}{\varepsilon_L} \tag{7-4}$$

where H_L is the limit stress and ε_L is the effective strain limit. Substitution of (7-4) into (7-3) provides:

$$\varepsilon_p L = \frac{(\sigma - H_0)\bar{\varepsilon}_L L}{H_L - H_0} \tag{7-5}$$

Substitution of the appropriate variables provides:

$$\varepsilon_p L = \frac{(40ksi - 36ksi)(0.015)(1in)}{90ksi - 36ksi} = 0.00111111in \tag{7-6}$$

Therefore, the total deformation of the structure is the sum of the elastic and plastic length changes provided in (7-5) and (7-6):

$$\varepsilon L = \varepsilon_e L + \varepsilon_p L = 0.00249042in \tag{7-7}$$

7.1.2 ABAQUS Solution

The axially loaded structure was modeled using ABAQUS to provide a baseline dynamic solution for comparison. The analytical solution is intended to provide verification of the steady-state deformations with the ABAQUS solution providing a comparison response during the remainder of the loading and unloading curve. The same material properties and basic geometric variables defined previously were used. The boundary conditions of the structure were also applied consistently. One node at the base of the structure was constrained in each of the three directions of the Cartesian coordinate system. The remaining nodes at the base of the structure were only constrained in the z-direction, which is aligned with the length of the beam. Since the problem is set up to only contain uniaxial stress, the size of the mesh is not critical. For simplicity, the beam was meshed as ten 0.1in. by 0.1in. by 0.1in. elements.

The elastic problem was evaluated first to determine the linear elastic response of the beam when subjected to the quasi-static dynamic loading curve. Only the density and two elastic parameters (modulus of elasticity and Poisson's ratio) were input to fully define the material. A plot of the deformation of the structure in the z-direction as a result of the axial load is shown in Figure 7-4. This plot indicates the peak deflection of

the beam as shown at a simulation time of 0.6ms. The complete history of the deformation is examined by plotting the deflection of one of the nodes on the load application surface throughout the entire simulation time. A plot of the deflection history is shown as Figure 7-5.

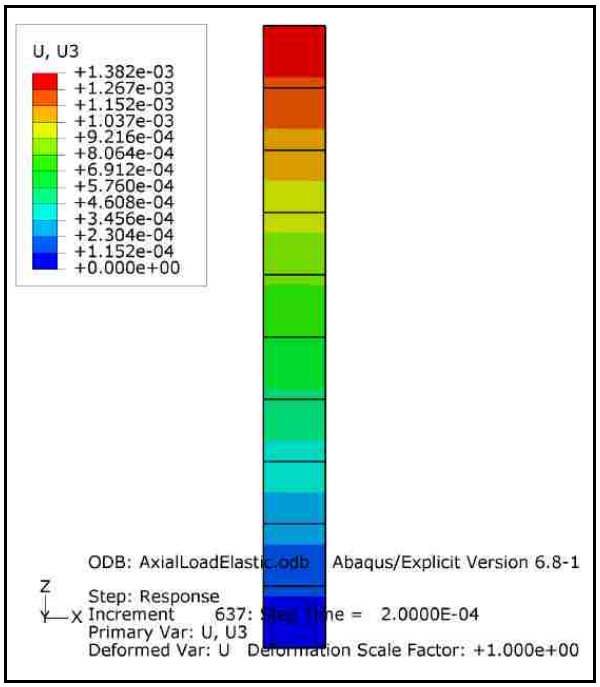

Figure 7-4: ABAQUS Elastic Deflection (Peak)

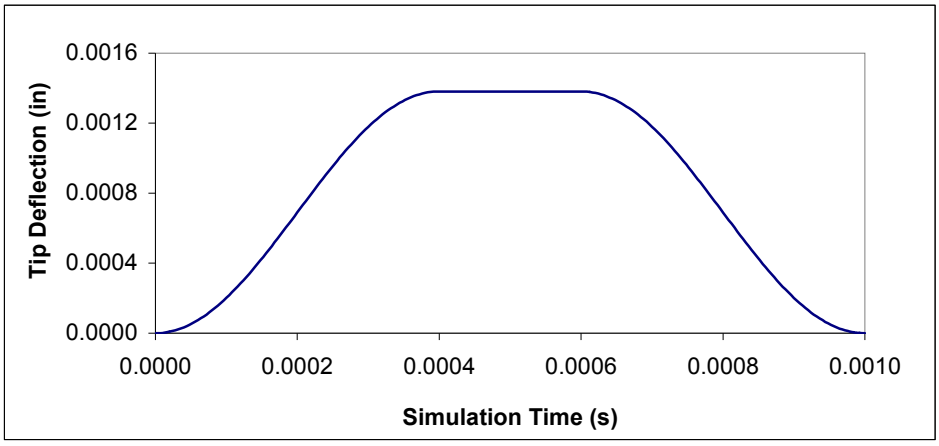

Figure 7-5: ABAQUS Elastic Tip Deflection

In order to investigate the elastic-plastic response of the structure, the plastic material properties were input with the assumption of linear isotropic hardening. A plot of the tip deflection of the beam in the z-direction is shown as Figure 7-6 for the elastic-plastic model. The response is overlaid with the elastic response for comparison. During the simulation times between 0.3 ms and 0.4 ms, plastic deformation is causing a dramatic increase in the rate of deflection. Once the peak load is achieved, the deflection stabilizes at a maximum deflection of 0.002518 in. Upon unloading, the elastic-plastic response follows the general shape of the elastic response but is offset by the plastic deformation that remains. After complete unloading, the plastic deformation of 0.001134 in. is the permanent deformation associated with the plastic strain.

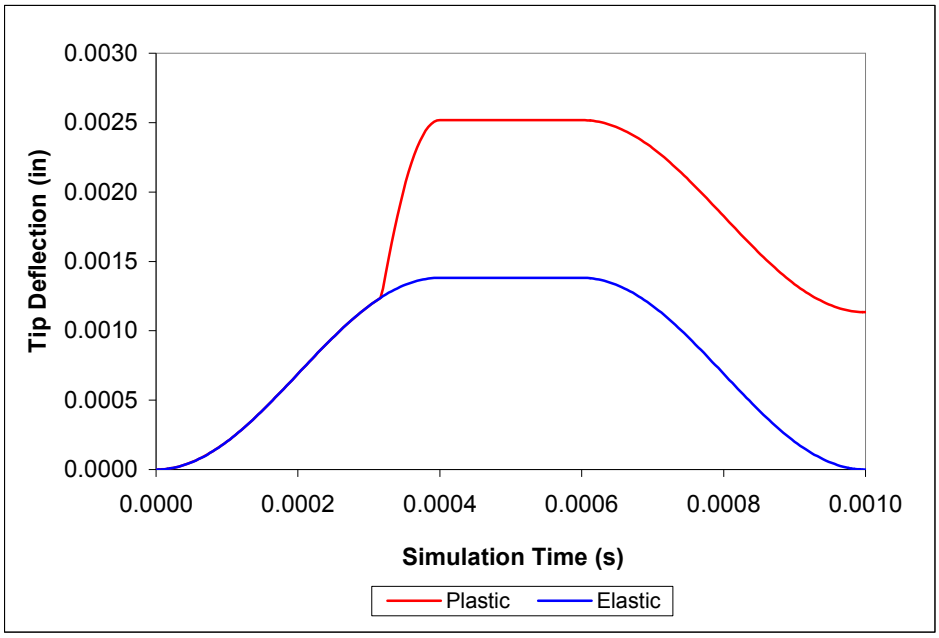

Figure 7-6: ABAQUS Elastic-Plastic Tip Deflection

7.1.3 Component Mode Synthesis Solution

Using the Matlab code available in the appendices, which implements the method defined in the previous chapters, the elastic-plastic response of the cantilever beam is analyzed. The Matlab code includes the capability to solve the solution through a variety of different methods to aid in the validation of results. The solution can either be full fidelity, full fidelity with modal superposition, or reduced fidelity with CMS reduction.

The first option is a full fidelity traditional finite element solution. The natural frequencies and mode shapes are not used in the calculations of the dynamic response. Instead, the full mass and stiffness matrices are used in the determination of the dynamic response. This solution is intended to provide direct correlation with the commercial finite element programs because there is no reduction in the fidelity of the problem. A plot of the deflection history of the cantilever beam is shown as Figure 7-7. The two traces of the elastic-plastic response are essentially identical and the curves appear overlaid. Investigation of the data reveals that the total plastic deformation predicted by the full fidelity Matlab code was 0.001113 in., compared to 0.001134 in. from ABAQUS. The result from the full fidelity Matlab code provides better correlation with the analytical solution, but the error of either solution is very small.

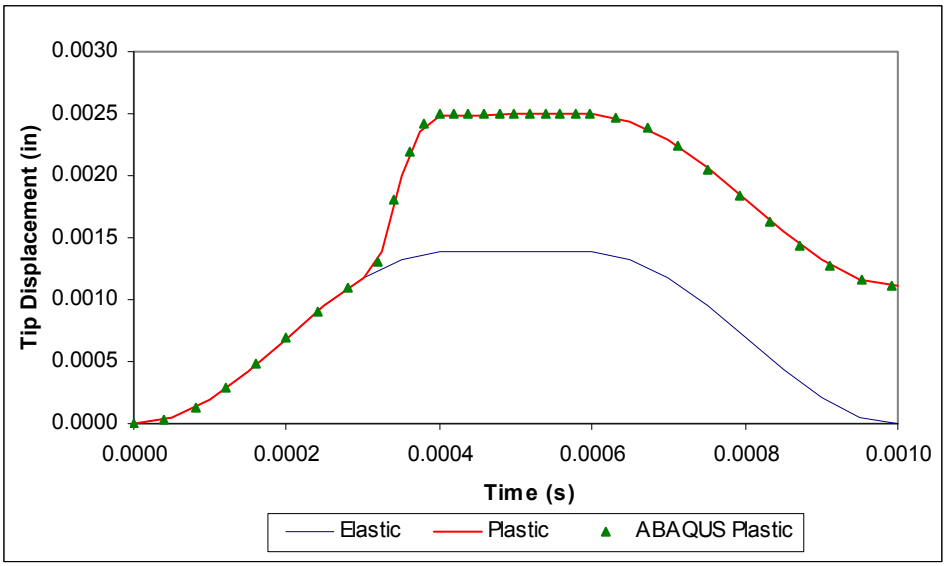

Figure 7-7: Full Fidelity Elastic-Plastic Tip Deflection

The second solution option with the Matlab codes is a full fidelity modal solution. This solution is essentially the same fidelity as previously discussed but the mass and stiffness matrices are replaced with their equivalent modal reductions. This allows the mass and stiffness matrices to be reduced to diagonal matrices, which dramatically reduces the number of calculations during the iterative solution of the equations of motion. The tip deflection is evaluated using this solution technique and the results were essentially identical to the results from the full fidelity Matlab solution, so a plot is not provided. The total plastic deformation from the full fidelity modal solution is 0.0011127 in.

With the CMS approach, the problem is further reduced through substructuring and modal reduction. The initial CMS reduction does not result in diagonal inertia and stiffness matrices, but the matrices are orthonormalized prior to the iteration of the equations of motion, which does result in diagonal matrices. The tip deflection as a function of simulation time from the full fidelity CMS solution is shown as Figure 7-8.

The ABAQUS result is also provided for comparison but the two results are essentially overlaid. The total plastic deformation from the full fidelity CMS solution was 0.001124 in., which is an error of about 1% compared to the analytical solution. This error will be increased by further reduction in the fidelity of the CMS solution as discussed in Section 6.2.

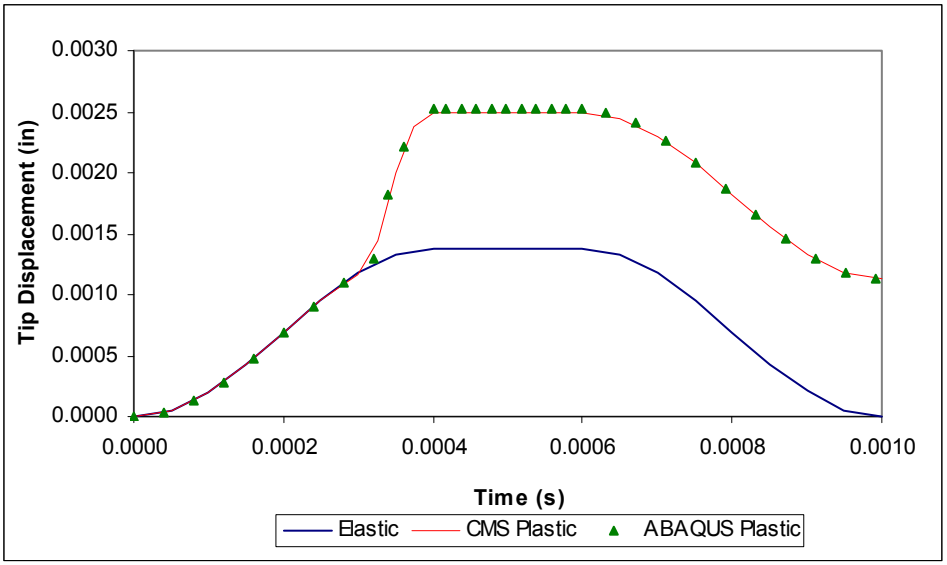

Figure 7-8: CMS Elastic-Plastic Tip Deflection

7.2 Impulse Loading of Simply Supported Beam – Full Load

A simply supported beam is subjected to a distributed load, applied instantaneously as investigated previously in [74], [75], [76]. The geometry of the beam, shown in Figure 7-9, is rectangular with a height of 2in and width of 1in. The length of the beam in 30in. The elastic modulus is 30,000ksi and the Poisson's ratio is 0.3. The material is assumed to follow a bi-linear strain hardening model with a hardening modulus equal to 0.25 of the elastic modulus after exceeding an initial yield stress of 50ksi.

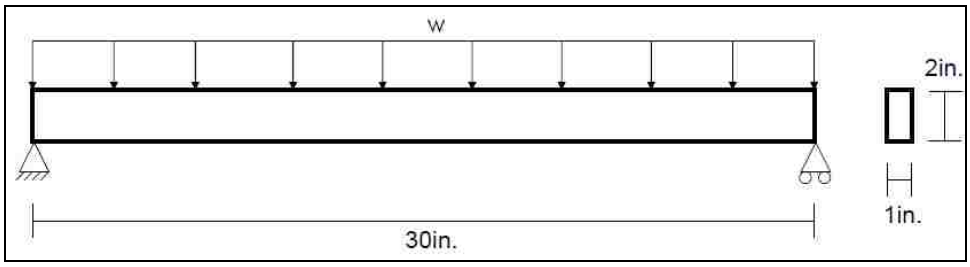

Figure 7-9: Simply Supported Beam Geometry

The mid-span deflection of the beam can be determined analytically by assuming an elastic-perfectly plastic material model [77]. The limit moment, M_0, is defined as the moment at the center of the beam that produces a stress equal to the initial yield stress throughout the cross-section:

$$M_0 = \frac{\sigma_0 b h^2}{4} = 50,000 in - lbf \tag{7-8}$$

The static collapse load is defined in terms of the limit moment and the length of the beam:

$$p_c = \frac{2M_0}{\left(L/2\right)^2} = 444.444 \, lbf/in \tag{7-9}$$

The static deflection at the center of the beam as a result of the static collapse load is:

$$\Delta = \frac{5 p_c L^4}{384 EI} = 0.234375 in \tag{7-10}$$

The initial mesh for the example problem is quite course for a bending stress problem. The beam was divided into 40 elements with 110 nodes. An 8-node linear brick element was incorporated, see Chapter 2, and is consistent with that used in Salinas structural dynamics code [78]. For improved performance with the bending loads, the bubble element formulation was used.

Using the method outlined, the beam is subjected to a stepped distributed load and the dynamic response is simulated over a 5ms interval after the initial load application. A distributed load equal to 0.625 of the critical load was applied, which corresponds to previous research [74] and [75]. The elastic and plastic mid-span deflection as a result of the applied loads is shown for the full finite element solution in Figure 7-10. The full solution does not include any reduction and serves as the baseline for comparison with the later CMS solution. For reference, the elastic and plastic dynamic solutions are provided over the same time interval using ABAQUS. The incompatible mode brick element, C3D8I, was used with the same mesh defined above. The axes of the plot are scaled based on the peak static deflection and the period of the elastic response.

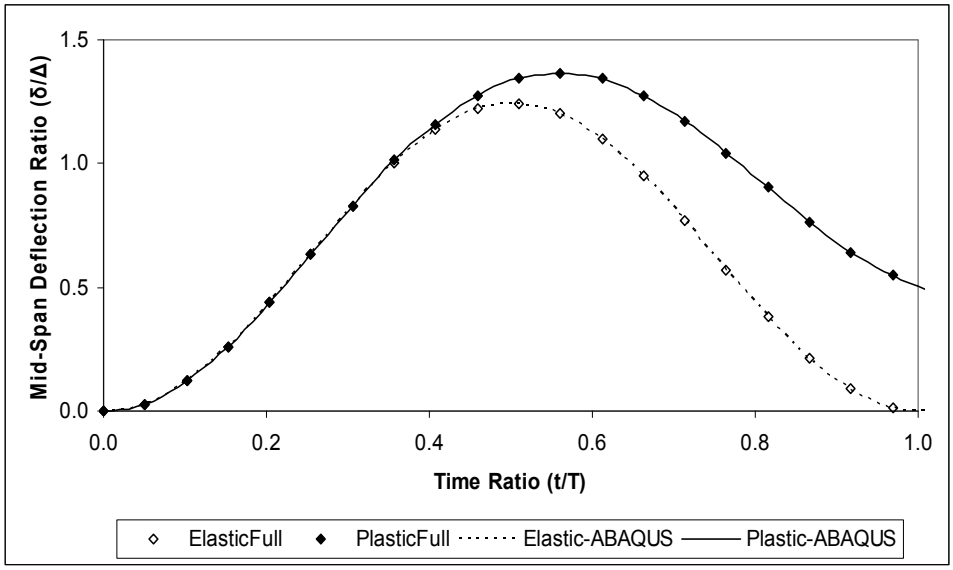

Figure 7-10: Elastic and Plastic Full Fidelity Responses

Since the accuracy of the stress calculation is dependent on the accuracy of the elastic response, a convergence study was conducted on the elastic response using the fixed interface CMS method. As the number of kept modes is increased, the accuracy of

the solution for the mid-span deflection is improved as shown in Figure 7-11. This relationship corresponds to the improved accuracy typically obtained through mesh refinement by increasing the number of elements. The relationship is not a smooth curve due to the orthogonality of the mode shapes. For the simply supported example, only mode shapes that correspond to the direction of deflection due to the applied load will be activated. Mode shapes in the other two directions do not significantly improve the accuracy of the response.

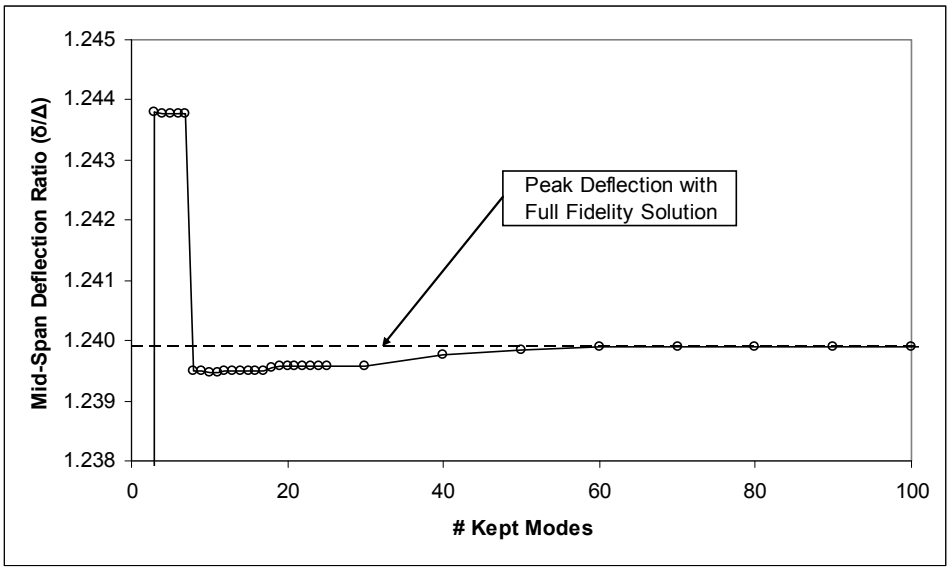

Figure 7-11: Convergence of Elastic Solution – 0.625*Pc

As shown in Figure 7-12, the elastic CMS response with 100 kept modes is overlaid with the full solution but the plastic response is under-predicted by the CMS method with 100 kept modes. This discrepancy is due to the approximations in the generation and application of the pseudoforce for the static case, which induces the plastic deformation. The iterative equations of motion are solved in orthonormal coordinates but must be converted to global coordinates for determination of the elemental stress and prediction

of the plastic deformation. The pseudoforce is generated in global coordinates and must be converted to orthonormal coordinates for use in the solution to the equations of motion. The accuracy of the solution can be improved by incorporating dynamic as well as static stiffness effects as investigated in the next example problem.

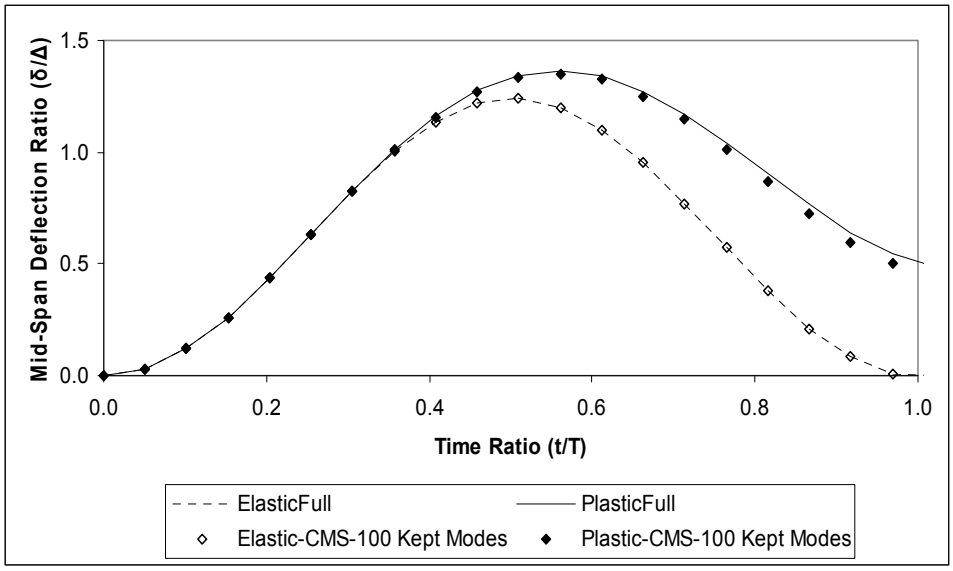

Figure 7-12: Non-linear CMS Response

The elastic portion of the dynamic solution using CMS techniques is less computationally expensive than the full solution due to the reduction in the size of the dynamic problem solved at each time step. For this particular example, the elastic CMS solution was solved in approximately 75% of the time required to solve the full elastic solution. The plastic portion of the solution is more computationally expensive for the CMS approach compared to the full solution but only occurs during a small interval during the simulation. The primary benefit of the proposed approach is the ability to integrate with a rigid body dynamic technique, which offers substantial computational savings over a full finite element approach.

7.3 Impulse Loading of Simply Supported Beam – Partial Load

A variation of the simply supported beam with a stepped pressure load is analyzed with the pressure loading only applied to the center of the beam rather than the complete length. The distributed load is applied to the center portion of the beam to reduce the number of boundary degrees of freedom in the CMS representation. The geometric dimensions and loading of the beam is shown in Figure 7-13.

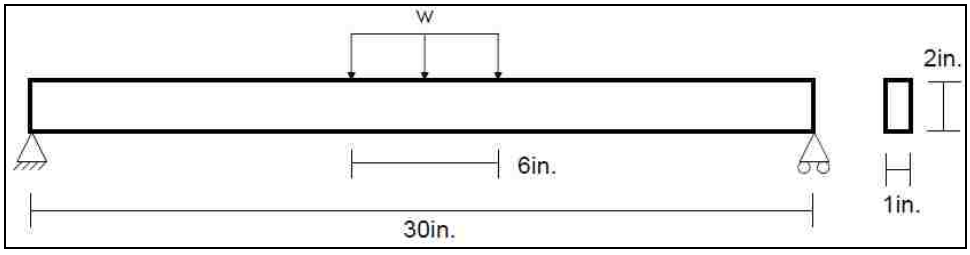

Figure 7-13: Simply Supported Beam Geometry

The elastic modulus is 30,000ksi, the Poisson's ratio is 0.3, and the distributed load is 850psi. The material is assumed to be isotropic and follow three types of strain hardening models, with an initial yield stress of 50ksi for each. The first model is an elastic-perfectly plastic model that results in completely plastic deformation after reaching the yield point. The second is a bi-linear model with a hardening modulus of 0.25 after reaching the initial yield stress. The third is a power law hardening curve, which is the most complex but most representative of actual uniaxial stress test results for metals. The three hardening models are shown in Figure 7-14.

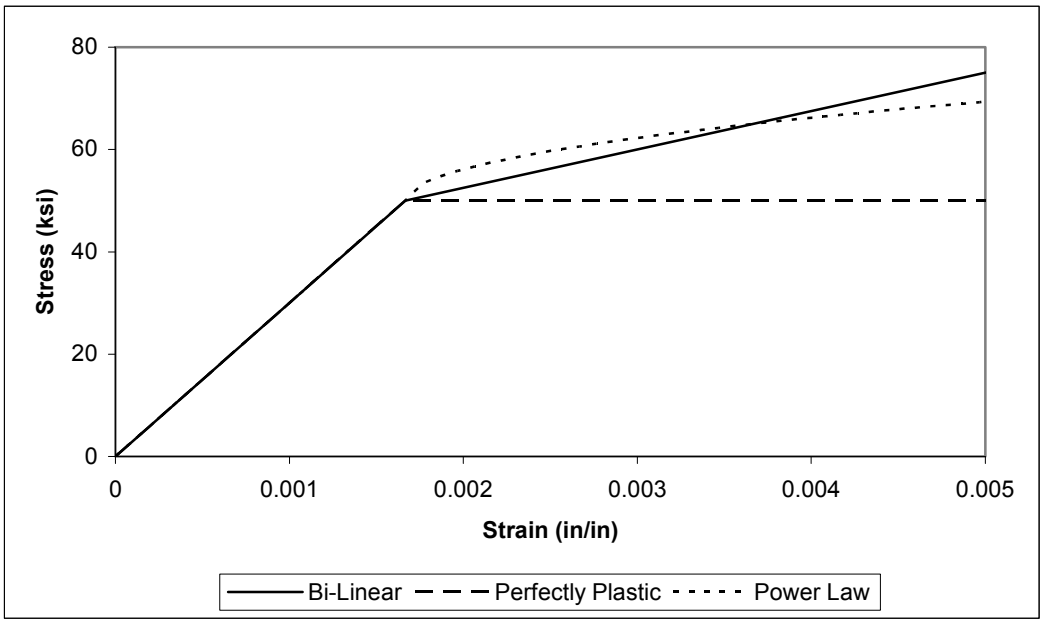

Figure 7-14: Plastic Strain Hardening Models

The beam is initially meshed using 40 elements with 110 nodes. There are 10 elements along the length, 1 across the width, and 4 along the height. An 8-node linear brick element is incorporated but for improved performance with the bending loads, the incompatible modes formulation was used, as defined in Chapter 2. Figure 7-15 shows the mid-span deflection of the beam as a result of the applied loading with the three material hardening models. The perfectly-plastic model has no plastic hardening and results in the largest peak deflection. The power law model exhibits more hardening than the bi-linear model for the initial elastic-plastic deformation and results in the smallest peak deflection.

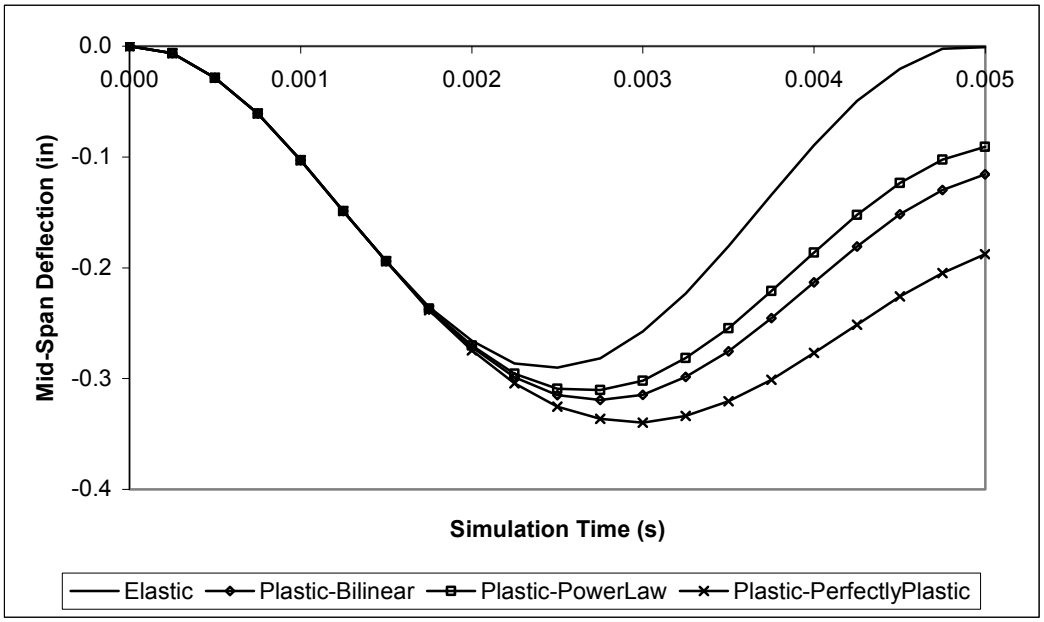

Figure 7-15: Mid-span Deflection - Hardening Models

The cumulative modal participation factor, from Section 3.4, is investigated to provide a visual indication of the represented mass of the beam as the number of retained modes is increased. Figure 7-16 shows the modal participation factors for each of the three Cartesian coordinates as well as the average of all three directions. The plot indicates that the modal mass converges to within 95% of the total mass with retention of approximately 25% of the interior modes.

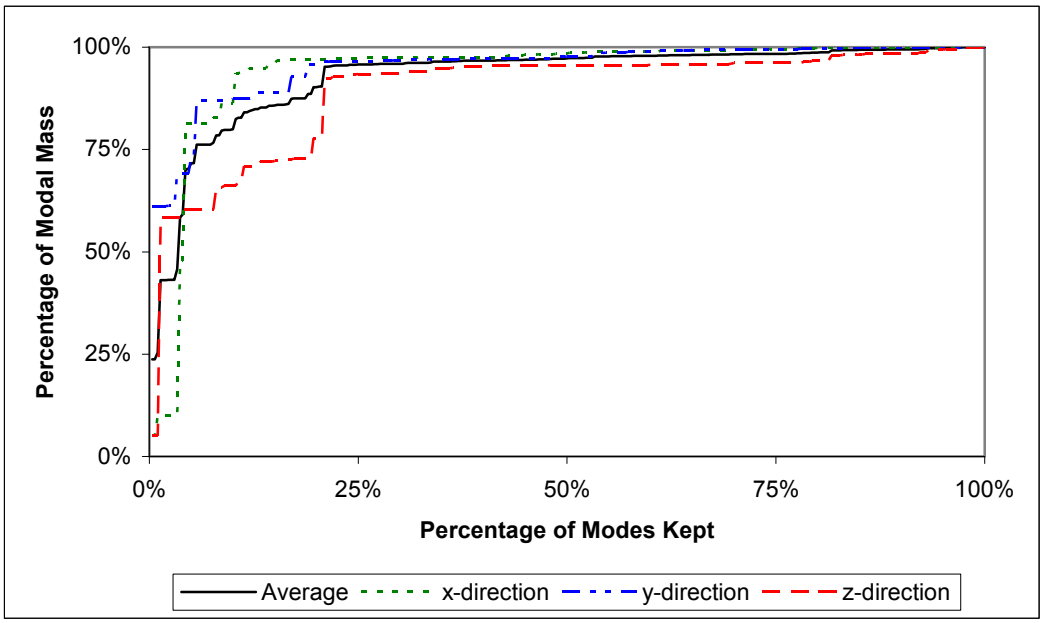

Figure 7-16: Modal Participation Factors

A zoomed plot of the same data for the final 5% of the modal mass is shown in Figure 7-17.

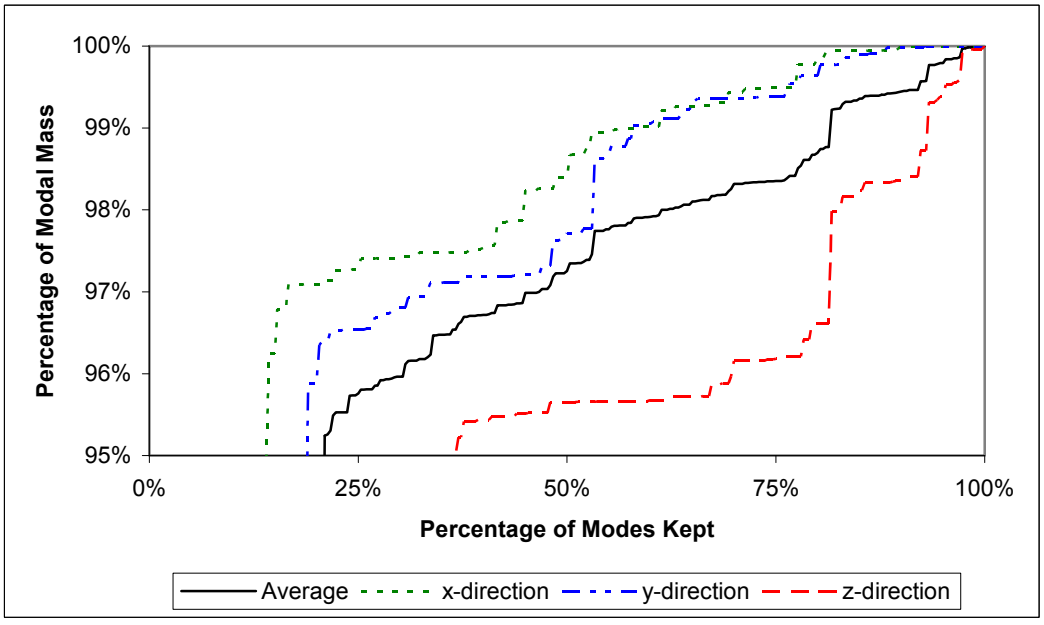

Figure 7-17: Modal Participation Factors – Final 5% of mass

Since the accuracy of the stress calculation is dependent on the accuracy of the elastic response, a convergence study was conducted on the elastic response using the fixed interface CMS method. As the number of kept modes is increased, the accuracy of the solution for the mid-span deflection is improved as shown in Figure 7-18. The figure indicates that only a small portion of the modes, less than 4%, are required to accurately represent the elastic deformation of the beam with an error of less than 0.1%. The accuracy is improving with the increase in retained modes but the improvement is insignificant given the plot scale shown. The relationship is not a smooth curve because of the orthogonality of the mode shapes. For the simply supported example, only mode shapes that correspond to the direction of deflection from the applied load will be activated. Mode shapes in the other two directions do not significantly improve the accuracy of the response.

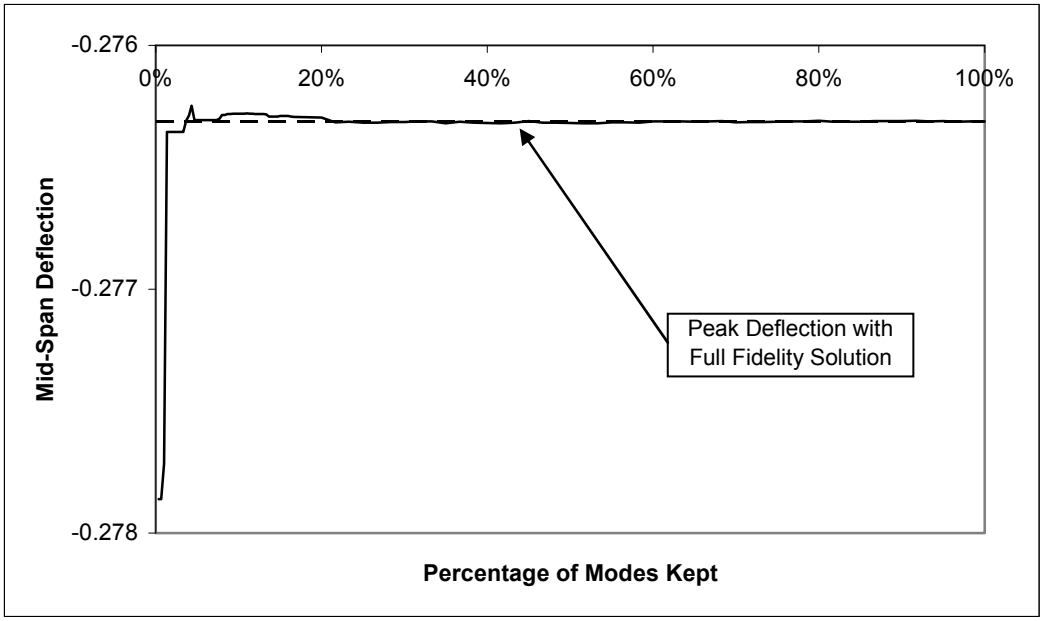

Figure 7-18: Accuracy of CMS Elastic Response

The convergence of the plastic CMS solution for the bi-linear hardening model is shown in Figure 7-19. The accuracy of the CMS solution roughly corresponds with the shape of the mass participation plot for the remaining 5% of modal mass but the overall solution is quite good, with less than 2% error, even with the retention of only a few modes. Since the plastic deformation is localized to elements at the mid-span of the beam, the plastic deflection is influenced by the high frequency mode shapes. The figure also shows the relative improvement obtained by incorporating the residual flexibility, which is similar in shape to the elastic convergence plot, Figure 7-18, because convergence of the plastic solution is directly dependent on the accuracy of the elastic solution. Use of the residual flexibility is computationally expensive but is offset by the ability to retain fewer modes.

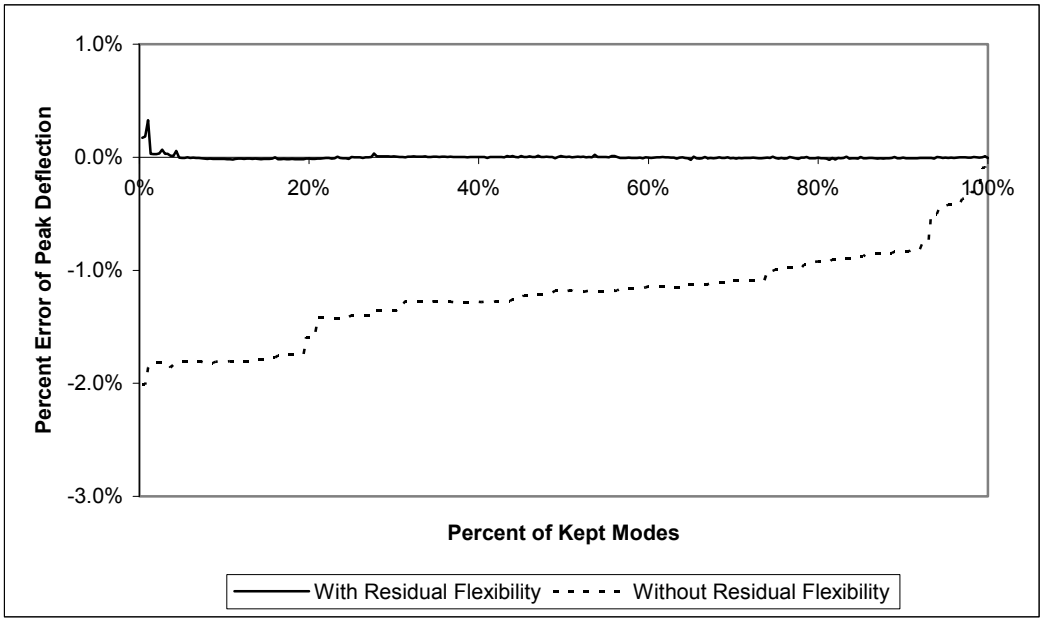

Figure 7-19: Accuracy of CMS Plastic Response

The accuracy of the full fidelity solution is dependent on the finite element mesh and conventional convergence studies are required to determine the appropriate mesh density.

The CMS techniques provide an approximation of the full fidelity response and can only be as accurate as the full fidelity results with the particular mesh. As the mesh density is increased, the number of retained interior modes does not need to increase if the lowest natural frequencies do not change significantly. Figure 7-20 shows that the computational savings for the non-linear CMS method increase as the mesh density increases, with the assumption of a fixed number of retained interior CMS modes. However, the number of degrees of freedom should be minimized to reduce the total computational time.

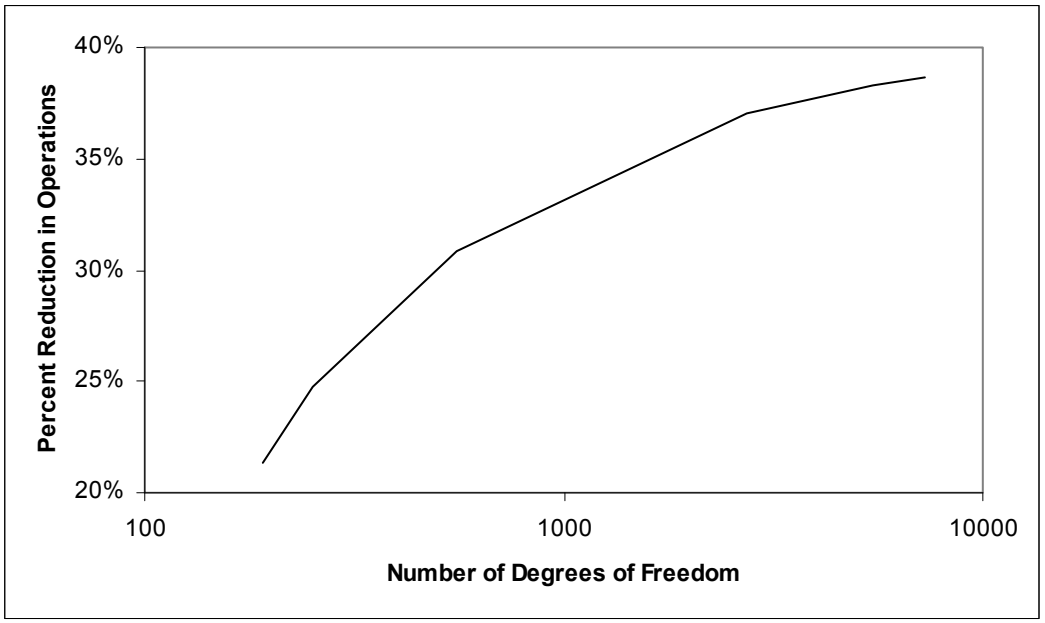

Figure 7-20: Computational Saving of CMS Method with Increased Mesh Density

7.4 Rigid Body Mechanism

An example problem of a primarily rigid body mechanism is presented to demonstrate the application to a component with a general geometric shape. A ratchet-driver mechanism provides intermittent rotary motion with the geometry shown in Figure

7-21. The drive arm is actuated by torque provided by a rotary solenoid acting on the arm that causes rotation from the rest position to a fully open position. During this transition, the drive pawl drops over the current tooth as a result of the torque provided by a torsion spring between the drive pawl and drive arm. Once the solenoid is de-energized, the drive arm returns under the torque provided by the extended drive arm spring to advance the ratchet wheel to the next index position. The relatively large ratchet wheel is assembled onto a shaft that is cantilevered at the base. Only the shaft is modeled as a flexible element with all other bodies remaining rigid.

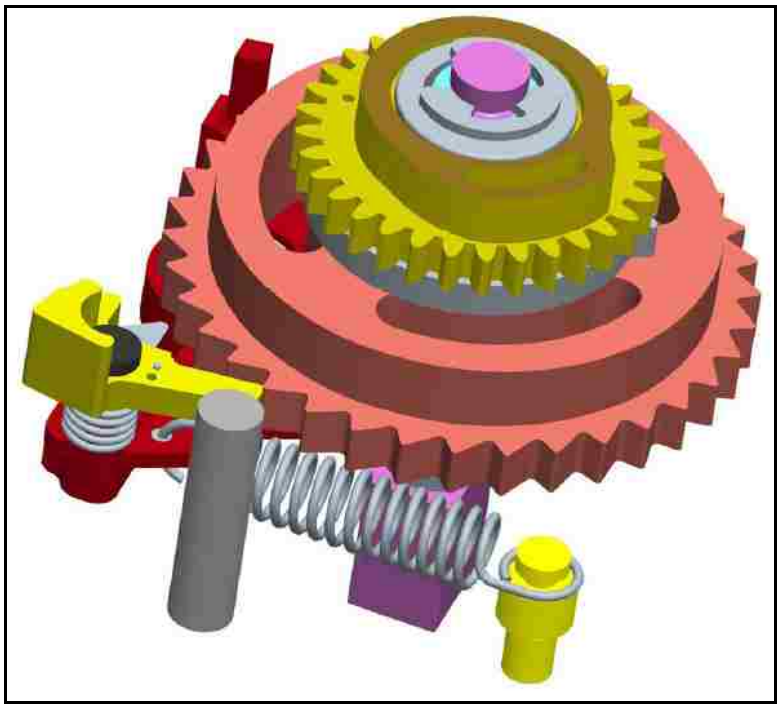

Figure 7-21: Rigid Body Mechanism Geometry

The entire assembly is subjected to a large impulse acceleration of 3500g with a haversine pulse shape and a duration of 0.5 ms. The shaft material is assumed to be a precipitation hardened steel with an initial yield strength of 90 ksi, Poisson's ratio of 0.3, and Young's modulus of 30,000 ksi. For simplicity, only a bi-linear material model is

investigated with a hardening modulus of 25% of the elastic modulus. Other hardening models could be readily incorporated using the same solution techniques. Due to the circular geometry of the shaft, isoparametric elements are used to approximate the circles as a collection of 8 linear segments, shown in Figure 7-22. In order for the mesh to be consistent throughout the volume, the circular pattern continues thru to the base of the shaft with the rectangle built up from the circle. A total of 156 isoparametric hexahedron elements with 236 nodes comprise the finite element representation. The global inertia and stiffness matrices are determined using the formulation in Chapter 2, with 8 nodes per element.

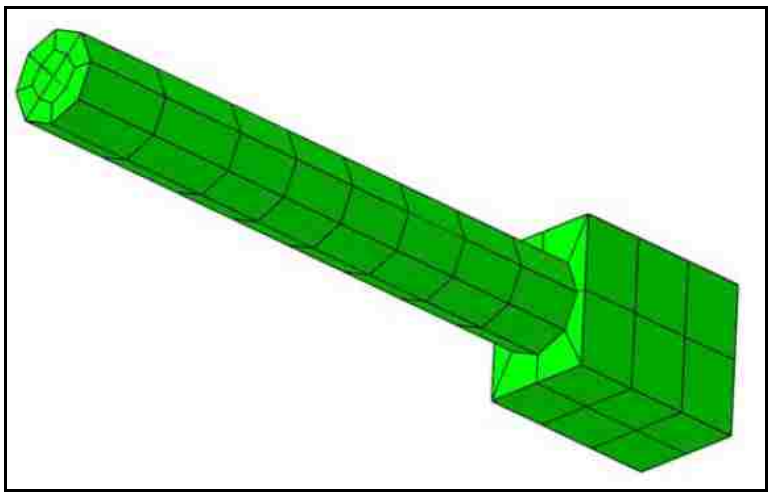

Figure 7-22: Mesh of Mechanism Shaft

To approximate the influence of the mass of the pattern wheel assembly on the shaft, an effective force is applied to the circumferential nodes aligned with the two radial bearings. The assembly is subjected to an impulse acceleration and the response is simulated over a 2 ms time interval. The elements at the base of the circular portion of the shaft experience plastic deformation and contribute to increased deformation at the tip. The tip deflection is shown in Figure 7-23 for the full solution as well as the CMS

solution with retention of 5% of the modes. The elastic solutions are essentially overlaid but the CMS plastic solution deviates slightly from the full fidelity plastic solution with only 5% retained modes. The peak plastic deformation is under-predicted by the CMS solution because a portion of the plastic force is not projected on the modal coordinates. The dynamic problem is simulated with 32% less computational effort than the full fidelity solution but the peak displacement at 1.7ms is under-predicted by 4%. The accuracy of the modal solution can be improved by increasing the number of retained modes or incorporating residual flexibility, described in Section 6.2.2.

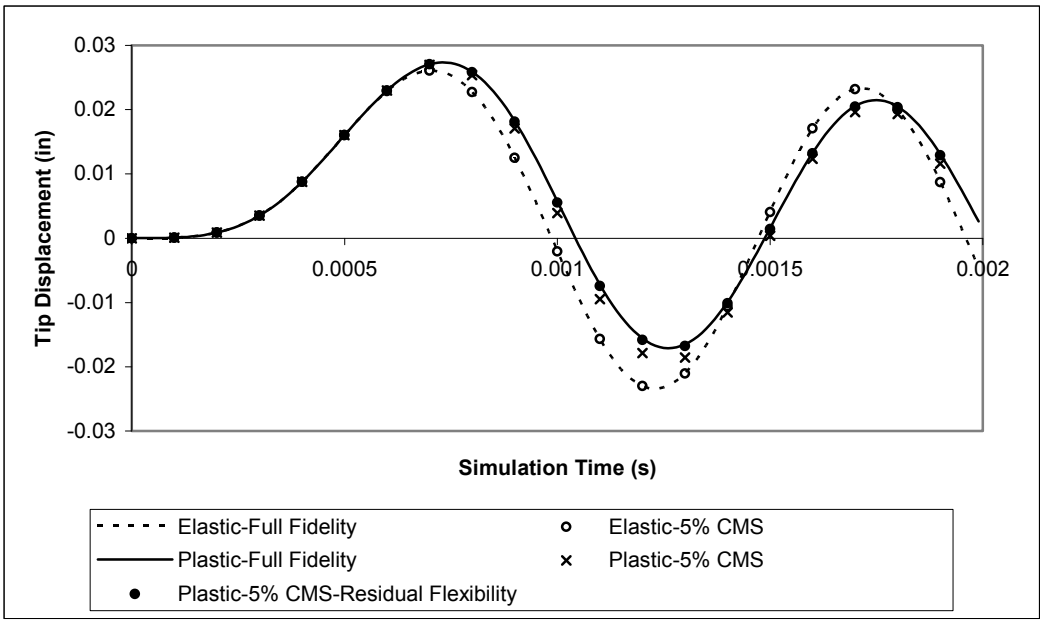

Figure 7-23: Mechanism Shaft Tip Deflection

As shown in Figure 7-23, the accuracy of the CMS plastic solution with 5% retained modes is essentially overlaid with the full solution response when residual flexibility is incorporated, but with 19% reduction in computational effort. The response with 100% of the retained modes is equivalent to the full fidelity solution and does not result in a loss in accuracy.

Chapter 8

Summary and Conclusions

Modeling and simulation are becoming an increasingly important aspect of the design process for a wide range of products. Budget and schedule goals are driving products to be developed and fielded with less time for design and development and with higher expectations for quality and reliability. These design pressures are especially high for complex mechanisms used in aerospace and automotive applications. Since it is not feasible, or impossible, to experimentally test every possible normal and abnormal operational requirement of a complex mechanism, modeling and simulation can help fill the gap.

Mechanisms that are composed of many components that receive relatively low loading relative to the strength of the part and are primarily expected to move as a rigid part or assembly can be modeled using rigid body dynamic techniques. Such techniques can greatly decrease the computational time required to solve a dynamic problem when compared to a full fidelity simulation because the size of the equations of motion solved at each increment are dramatically reduced. This computational efficiency can be preserved and the accuracy of the simulation can be improved by selectively modeling highly stressed components with a modal representation. Such an enhancement can be very effective if the vast majority of the parts can still be approximated as rigid bodies. The fixed interface CMS method has emerged as a very popular technique for

incorporating flexibility within a primarily rigid body simulation. Since all of the boundary degrees of freedom are retained without any reduction, the fixed interface CMS method can easily be incorporated within a simulation without substantial setup required. The CMS representation is determined independent of the rigid body simulation and only solved once, with only the modal information being required for the dynamic simulation.

The limitation of the application of CMS methods to primarily rigid body simulations is that the response is limited to a linear-elastic assumption. An enhanced framework for solving of non-linear dynamic problems utilizing fixed interface CMS has been proposed and investigated in this dissertation. This enhancement can extend the utility of currently available techniques to include the ability to adequately simulate the non-linear responses associated with plastic deformation of components. The stress within each element is determined from the CMS representation and evaluated against a user defined yield criteria, such as von Mises. If the effective stress within any element has exceeded the yield strength of the material, the plastic deformation is determined using classical plasticity theory.

An equivalent force is calculated to provide the predicted amount of plastic deformation and introduced into the reduced CMS representation of the equations of motion. The pseudoforce allows the plastic deformation to be induced purely by a force without requiring modification to the original CMS representation. This allows the remainder of the dynamic solution to continue just as it would for any elastic response. Since the equations of solved incrementally, the pseudoforce is never completely unloaded and remains to represent the plastic set that has been induced by the applied

loading. The strains remain irreversible by maintaining a measure of the effective plastic strain within each element.

The proposed framework can be integrated within a primarily rigid body dynamic code as an external subroutine that returns a force based on an input modal displacement. If the step remains elastic, the additional algorithms are not required and the only impact on the number of computations is the stress calculation. If this calculation is limited to specific highly-stressed regions of the component, the computation expense can be further reduced. The procedure required to induce the plastic deformation can be more computationally expensive than a direct method due to the required transformations between modal and global coordinate systems. The benefits and disadvantages of the proposed method are somewhat dependent on the specific problem of interest but significant benefits can be realized for primarily rigid body dynamic applications.

8.1 Future Work

The computational framework developed in this dissertation was investigated using custom Matlab code found within the Appendices. The intent was to demonstrate the utility and charactistics of the method on a small scale using a high-level code. Many of the calculations performed within the Matlab code should ideally be generated by a commercial rigid body dynamic code, with these non-linear calculations performed within a supplemental set of subroutines. For the programming in this dissertation, little emphasis was placed on improving the efficiency and minimizing memory storage requirements since the primary goal was to develop an educational understanding. Computational expense can be reduced by improving the efficiency of the code and transitioning to a general purpose programming language.

The numerical examples provided in Chapter 7 were not validated against experimental results. The accuracy of the non-linear CMS solutions were based on the full-fidelity finite element results, with selected comparisons to results from a commercial finite element package or analytical solutions. More complete measures of accuracy can be obtained through a rigorous validation with experimental data. All numerical and experimental analyses require assumptions that must be critically evaluated during any verification or validation activities.

The primary motivation for the development of the non-linear CMS framework was to computationally simulate the dynamic performance of complex mechanisms under a wide variety of loading conditions. The choice of element formulations, yield functions, and constitutive models reflects this influence and is not indicitave of the limitations of the method. Capabilities could easily be extended to include a wide variety of more complex elemental formulations, damping models, yield functions, etc. Only a limited set of options were investigated with the application of traditional, rate-independent, plasticity formulations but the method can be extended to more complex models.

The goal of this dissertation was to present the theoretical formulation of a newly developed framework for dynamic simulation and present numerical examples to demonstrate potential applications to indicate the accuracy with a given set of assumptions. Using the same theoretical formulation, the method can easily be expanded to include a much wider range of capabilities. When applied to other applications, the assumptions must be clearly understood with a thorough investigation of the convergence properties of the results obtained.

Appendices

Appendix A

Matlab Code for Finite Element Setup

This appendix contains the Matlab code associated with the generation of the global mass, damping, and stiffness matrices of the structure. The main function, Master3D.m, calls all subroutines and performs some of the ancillary calculations prior to the integration of the equations of motion. The input data is read from other subroutines as well as Excel files that contain the nodal and elemental data for a substructure.

```
function Master3D
clear;clc;

% Input Parameters
mesh=1;
[meshfile,numn,nume,E,Nu,rho,BndN0,BndNF,BndN,FP,F,Pbc,Fbc,gtype,...
    numbndM]=InputParameters(mesh);

% Other Input Parameters
stype=1;   % Solver type - 1=MatlabEig, 2=Lanczos
rtype=2;   % Reduction type - 1=none, 2=CraigBampton
DirMod=0;  % Only used if rtype=1, 0- Direct Solution, 1- Modal Solution
if rtype==1
    Nmodes=3*numn;
elseif rtype==2
    %Nmodes=252;%108;%big;
    Nmodes=big;
end
zeta=0;  % Modal Damping Parameter

% Read Mesh Data From Excel Files
[nodes,elements] = ReadMeshData(meshfile,numn,nume);

% Generate D Matrix - Isotropic Elasticity
[D]=ElasticityIsotropic(E,Nu);  %Override Later

% Generate Mass and Stiffness Matrix
[M,K,Pstrain,Pstress,detJstore,Bstore,G_hatstore,Kelstore] = ...
    IsoHexMKBubble(nodes,elements,D,rho,gtype);
F0=F;

% Sanity Check-Calculate Mass of Structure
```

```
cmass=0;
if cmass==1
    mass=0;
    for i=1:3*numn
        for j=1:3*numn
            mass=mass+abs(M(i,j));
        end
    end
    mass=mass/3;
end

% Generate Lumped Mass Matrix (Diagonal)
lmass=1;   % Inertia Matrix - 0=Consistent, 1=Lumped
if lmass==1
    mass=0;
    for i=1:3*numn
        for j=1:3*numn
            mass=mass+abs(M(i,j));
        end
    end
    mass=mass/3;
    Ml(1:3*numn,1:3*numn)=0;
    sumdiag=0;
    for i=1:3*numn
        sumdiag=sumdiag+M(i,i);
    end
    mscale=3*mass/sumdiag;
    for i=1:3*numn
        Ml(i,i)=M(i,i)*mscale;
    end
    M=Ml;
end

% Sanity Check-Calculate Mass of Structure
cmass=0;
if cmass==1
    mass=0;
    for i=1:3*numn
        for j=1:3*numn
            mass=mass+abs(M(i,j));
        end
    end
    mass=mass/3;
end

% Sanity Check-Calculate Normal Modes of Structure
cmodes=0;
if cmodes == 1
    [modes,omega] = MatlabEig(M,K,3*numn);
    omega(:,1:3)
    MPlot=[12];
    xlswrite('M-K_Matrices',modes,'modes')
    ModePlots(modes,nodes,elements,MPlot);
    return
end

% Apply Appropriate Boundary Conditions to Substructure
[IntN,K,PTstrain,PTstress,Pbc,Fbc,Tdisp]=BoundaryConditions(BndN,BndN0,..
      .
    elements,mesh,numn,nume,Pbc,Fbc,K,Pstrain,Pstress);

% Sanity Check-Calculate Normal Modes of Structure
cmodes=0;
```

```
if cmodes == 1
    [modes,omega] = MatlabEig(M,K,size(M,2));
    omega
    return
end

% Use Damping
usedamp=0;
if usedamp == 1
    [Nc,lambdac] = MatlabEig(M0,K0,Nmodes);
    lambdac=abs(lambdac);
    for i=1:Nmodes;
        C0(i,i)=2*zeta*(2*pi*lambdac(i));
    end
    invNc=sparse(inv(Nc));
    C0=invNc'*C0*invNc;
else
    C0(1:3*numn,1:3*numn)=0;
end

% Apply Reduction Method
M0=M;
K0=K;
if rtype == 1
    PHI=eye(Nmodes);
    Pf=1;  % Placeholder for Newmark
    Pu=1;  % Placeholder for Newmark
    RF=1;  % Placeholder for Newmark
    if DirMod==0   % Full Fidelity Solution
        %C=C0;
        C=zeros(3*numn);   % To turn off damping, debugging
        %M=zeros(3*numn);   % To turn off inertia, debugging
        N=eye(3*numn);
    elseif DirMod==1   % Full Modal Solution
        % Normalize Modal Matrix
        [N,lambda] = MatlabEig(M,K,Nmodes);
        Nscale=N'*M*N;
        for i=1:Nmodes
            for j=1:Nmodes
                N(i,j)=N(i,j)/sqrt(Nscale(j,j));
            end
        end
        for i=1:Nmodes;
            C(i,i)=2*zeta*(2*pi*lambda(i));
        end
        M=eye(Nmodes);
        clear K
        for i=1:Nmodes;
            K(i,i)=(lambda(i)*2*pi)^2;
        end
        if mesh >= 8
            FN(1:Nmodes,1)=N'*F(1:3*numn,1);
        end
        PTstrain=PTstrain*N;
        PTstress=PTstress*N;
    end
elseif rtype == 2   % CMS Solution
    numbnd=size(BndN,2);
    Tmodes=Nmodes+3*numbnd;

    [Mbar,Kbar,Mn,Kn,PHI] = ...
        CraigBampton(M,K,Nmodes,Tmodes,stype,BndN,IntN);
```

```
    % Determine Effective Modal Mass
    checkmeff=0;
    if checkmeff==1
        numM0=size(M,1);
        [Gamma,Gamma3,meff,meff3]=...
            EffectiveModalMass(numM0,numbnd,Mn,Kn,mass);
        return
    end

    % Reorder the Force Vector to Match Inertia and Damping Matrices
    [Fo(1:3*numn,1),P]=Order2(F(1:3*numn,1),IntN,BndNF,BndN0);
    Fq=PHI'*Fo;

    % Orthonormalize the Inertia and Damping Matrices
    [N,lambda,FqO,numrgd]=...
        Orthonormalize(Mbar,Kbar,Fq,Nmodes,Tmodes,numbndM);

    % Generate Conversion Matrices
    Pf=N'*PHI'*P;
    Pu=P'*PHI*N;
    size(N')
    Y=eye(Tmodes-numrgd);

    % Populate C Matrix for Modal Damping
    moddamp=0;   % 0 = No Modal Damping, 1 = Use Modal Damping
    C(1:Nmodes-numrgd,1:Nmodes-numrgd)=0;
    if moddamp==1
        for i=1:Nmodes-numrgd;
            C(i,i)=2*zeta*(2*pi*lambda(i));
        end
    end

    M=eye(Tmodes-numrgd);  % After Orthonormalization, Inertia is
       Itentity
    clear K   % Clear Prior to Storing Eigenvalues
    K(1:Tmodes-numrgd,1:Tmodes-numrgd)=0;
    for i=1:Tmodes-numrgd;
        K(i,i)=(lambda(i)*2*pi)^2;
    end
    K=sparse(K);

    % Residual Flexibility Calculation
    useRF=0;
    if useRF == 1
        RF=sparse(K0\eye(3*numn)-Pu*(K\Pf));
    else
        RF=1;
    end
end

%% Generate Force Vector for Axial Load
if mesh==3
    mesh=1;
end
if mesh==2
    T0=0;
    TF=0.001;
    Tstep=1E-6;
    numS=round((TF-T0)/Tstep);
    % Create Force Vector
    t=0;
    d=.0004;
    Fu(1:3*numn,1)=F(1:3*numn,1);
```

```
    for i=2:numS
        t=t+Tstep;
        if t < d
            Fu(1:3*numn,i)=Fu(1:3*numn,1)*(1/2*(1-cos(pi()*t/d)));
        elseif t < (TF-d)
            Fu(1:3*numn,i)=Fu(1:3*numn,1);
        else
            Fu(1:3*numn,i)=Fu(1:3*numn,1)*(1/2*(1-cos(pi()*(TF-t)/d)));
        end
    end
    Fu(1:3*numn,1)=0;  % Set force at first time step equal to 0

    if rtype == 1
        if DirMod == 0
            F=Fu;
        elseif DirMod == 1
            for i=2:numS
                FN(1:Nmodes,i)=N'*Fu(1:Nmodes,i);
            end
            F=FN;
        end
    end
    if rtype == 2
        t=0;
        for i=2:numS
            t=t+Tstep;

            if t < d
                FqO(1:Nmodes,i)=FqO(1:Nmodes,1)*(1/2*(1-cos(pi()*t/d)));
            elseif t < (TF-d)
                FqO(1:Nmodes,i)=FqO(1:Nmodes,1);
            else
                FqO(1:Nmodes,i)=FqO(1:Nmodes,1)*(1/2*(1-cos(pi()*(TF-
   t)/d)));
            end
        end
        F=FqO;
    end
elseif mesh==1%9  % Use for Liu Beam
    T0=0;
    TF=0.005;
    Tstep=1E-6;
    numS=round((TF-T0)/Tstep);
    Fu=F0;
    F(1:3*numn,1:numS)=0;
    for i=1:numS
        F(1:3*numn,i)=Fu(1:3*numn,1);
    end
    Fu=F;
    if rtype == 1
        if DirMod == 1
            FNN(1:Nmodes,1:numS)=0;
            for i=2:numS
                FNN(1:Nmodes,i)=FN(1:3*numn,1);
            end
            F=FNN;
        end
    elseif rtype == 2
        FqO(1:size(N,2),2:numS)=0;
        for i=2:numS
            FqO(:,i)=FqO(:,1);
        end
```

```
        F=sparse(FqO);
        Fu=sparse(Fu);
    end
elseif mesh==11
    T0=0;
    TF=0.002;
    Tstep=1E-6;
    numS=round((TF-T0)/Tstep);
    % Create Force Vector
    t=0;
    d=.001;
    Fu(1:3*numn,1:numS)=0;
    Fu(1:3*numn,1)=F(1:3*numn,1);
    for i=2:numS
        t=t+Tstep;
        if t < d
            Fu(1:3*numn,i)=Fu(1:3*numn,1)*(1/2*(1-cos(2*pi()*t/d)));
        else
            Fu(1:3*numn,i)=0;
        end
    end
    Fu(1:3*numn,1)=0;  % Set force at first time step equal to 0
    Fu=sparse(Fu);

    if rtype == 1
        if DirMod == 0
            F=sparse(Fu);
        elseif DirMod == 1
            FN(1:Nmodes,1:numS)=0;
            for i=2:numS
                FN(1:Nmodes,i)=N'*Fu(1:Nmodes,i);
            end
            F=FN;
        end
    end
    if rtype == 2
        FqO(1:size(N,2),2:numS)=0;
        for i=1:numS
            FqO(:,i)=Pf*Fu(:,i);
        end
        F=sparse(FqO);
    end
end

% Calculation of Static Solution
static=0;
if static==1
    u=K\Fu(1:3*numn,600);
    DefPlots(nodes,u)
    return
end

[X] = NewmarkIso(M,M0,C,C0,K,K0,Fu,F,T0,TF,Tstep,nodes,elements,D,N,...
    rtype,DirMod,Pbc,Fbc,mesh,Pf,Pu,RF,PTstrain,PTstress,detJstore,...
    Bstore,G_hatstore,Kelstore,Tdisp);

u=X;

if mesh == 1
    eplot=8;
    udef=u(3*(eplot-1)+3,1:numS);
elseif mesh == 2
    eplot=1;
```

```
    udef=u(3*(eplot-1)+3,1:numS);
elseif mesh == 9
    eplot=105;
    udef=u(3*(eplot-1)+3,1:numS);
elseif mesh == 11
    eplot=14;
    udef=u(3*(eplot-1)+2,1:numS);
end

plotincr=20;
countm=0;
for i=1:numS
    if i>=numS/plotincr*countm
        countm=countm+1;
        udefp(1,countm)=udef(1,i);
    end
end

% Determine Maximum Deflection
maxd=udef(1,1);
maxs=1;
for i=1:numS
    maxn=udef(1,i);
    if maxn < maxd
        maxd=maxn;
        maxs=i;
    end
end
maxd

end   % End Function
```

```
function [IntN,K,PTstrain,PTstress,Pbc,Fbc,Tdisp]=...
    BoundaryConditions(BndN,BndN0,elements,mesh,numn,nume,Pbc,Fbc,K,...
    Pstrain,Pstress)

% Create Vector of Interface Nodes
numbnd=size(BndN,2);
BndNS=sort(BndN);
IntN(1:(numn-numbnd))=0;
c=0;
for i=1:BndNS(1)-1
    c=c+1;
    IntN(c)=i;
end
for i=1:numbnd-1
    for j=BndNS(i)+1:BndNS(i+1)-1
        c=c+1;
        IntN(c)=j;
    end
end
for i=BndNS(numbnd)+1:numn
    c=c+1;
    IntN(c)=i;
end

runFbc=0;
if runFbc==1;
    % Generate Force BC Matrix
    Fbc(1:3*numn,1:3*numn)=0;
    for i=1:3*numn
        Fbc(i,i)=1;
    end
    if mesh==1
        for i=[4,7,9,11]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[1 0 0; 0 1 0; 0 0 0];
        end
        for i=[1,8]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[0 0 0; 0 1 0; 0 0 1];
        end
        for i=[8]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[1 0 0; 0 0 0; 0 0 1];
        end
        Fbc=sparse(Fbc);
        %save('LiuBeamCrude1Fbc.mat','Fbc')
    elseif mesh==2
        for i=[21,22,43,44]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[1 0 0; 0 1 0; 0 0 0];
        end
        for i=[21,22]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[0 0 0; 0 1 0; 0 0 1];
        end
        for i=[21]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[1 0 0; 0 0 0; 0 0 1];
        end
        Fbc=sparse(Fbc);
        %save('RectBeam2Fbc.mat','Fbc')
    elseif mesh==3
        for i=[4,7,9,11]
```

```
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[1 0 0; 0 1 0; 0 0 0];
        end
        for i=[9,11]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[0 0 0; 0 1 0; 0 0 1];
        end
        for i=[9,11]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[1 0 0; 0 0 0; 0 0 1];
        end
        Fbc=sparse(Fbc);
        save('LiuBeamCrude3Fbc.mat','Fbc')
    elseif mesh==9
        for i=[45,55,100,110]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[1 0 0; 0 1 0; 0 0 0];
        end
        for i=[50,105]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[0 0 0; 0 1 0; 0 0 1];
        end
        for i=[105]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[1 0 0; 0 0 0; 0 0 1];
        end
        Fbc=sparse(Fbc);
        %save('LiuBeamCrude2Fbc.mat','Fbc')
    elseif mesh==11
        for i=[19,20,22,24,90,91,92,93,154]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[1 0 0; 0 1 0; 0 0 0];
        end
        for i=[90,154]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[0 0 0; 0 1 0; 0 0 1];
        end
        for i=[154]
            m=3*(i-1);
            Fbc(m+1:m+3,m+1:m+3)=[1 0 0; 0 0 0; 0 0 1];
        end
        Fbc=sparse(Fbc);
        %save('PWMesh1Fbc.mat','Fbc')
    end
end

% Generate Boundary Condition Matrix
runPbc=0;
if runPbc == 1
    % Apply Boundary Conditions
    Pbc(1:3*numn,1:3*numn)=1;
    PbcScale=10^6;
    if mesh==1
        % Constrain boundary nodes in the z direction
        for i=[4,7,9,11]
            m=3*(i-1);
            for j=[4,7,9,11]
                n=3*(j-1);
                for k=3:3    % Only constrain z direction
                    for l=3:3    % Only constrain z direction
                        Pbc(m+k,n+l)=PbcScale;
                    end
                end
```

```
        end
    end
    % Constrain boundary nodes in the x direction
    for i=[1,8]
        m=3*(i-1);
        for j=[1,8]
            n=3*(j-1);
            for k=1:1
                for l=1:1
                    Pbc(m+k,n+l)=PbcScale;
                end
            end
        end
    end
    % Constrain boundary nodes in the y direction
    for i=[8]
        m=3*(i-1);
        for j=[8]
            n=3*(j-1);
            for k=2:2
                for l=2:2
                    Pbc(m+k,n+l)=PbcScale;
                end
            end
        end
    end
    %save('LiuBeamCrude1Pbc.mat','Pbc')
elseif mesh==2
    % Constrain boundary nodes in the z direction
    for i=[21,22,43,44]
        m=3*(i-1);
        for j=[21,22,43,44]
            n=3*(j-1);
            for k=3:3    % Only constrain z direction
                for l=3:3    % Only constrain z direction
                    Pbc(m+k,n+l)=PbcScale;
                end
            end
        end
    end
    % Constrain boundary nodes in the x direction
    for i=[21,22]
        m=3*(i-1);
        for j=[21,22]
            n=3*(j-1);
            for k=1:1
                for l=1:1
                    Pbc(m+k,n+l)=PbcScale;
                end
            end
        end
    end
    % Constrain boundary nodes in the y direction
    for i=[21]
        m=3*(i-1);
        for j=[21]
            n=3*(j-1);
            for k=2:2
                for l=2:2
                    Pbc(m+k,n+l)=PbcScale;
                end
            end
        end
    end
```

```
    end
    %save('RectBeam2Pbc.mat','Pbc')
elseif mesh==3
    % Constrain boundary nodes in the z direction
    for i=[4,7,9,11]
        m=3*(i-1);
        for j=[4,7,9,11]
            n=3*(j-1);
            for k=3:3    % Only constrain z direction
                for l=3:3    % Only constrain z direction
                    Pbc(m+k,n+l)=PbcScale;
                end
            end
        end
    end
    % Constrain boundary nodes in the x direction
    for i=[9,11]
        m=3*(i-1);
        for j=[9,11]
            n=3*(j-1);
            for k=1:1
                for l=1:1
                    Pbc(m+k,n+l)=PbcScale;
                end
            end
        end
    end
    % Constrain boundary nodes in the y direction
    for i=[9,11]
        m=3*(i-1);
        for j=[9,11]
            n=3*(j-1);
            for k=2:2
                for l=2:2
                    Pbc(m+k,n+l)=PbcScale;
                end
            end
        end
    end
    save('LiuBeamCrude3Pbc.mat','Pbc')
elseif mesh==9
    % Constrain boundary nodes in the z direction
    for i=[45,55,100,110]
        m=3*(i-1);
        for j=[45,55,100,110]
            n=3*(j-1);
            for k=3:3    % Only constrain z direction
                for l=3:3    % Only constrain z direction
                    Pbc(m+k,n+l)=PbcScale;
                end
            end
        end
    end
    % Constrain boundary nodes in the x direction
    for i=[50,105]
        m=3*(i-1);
        for j=[50,105]
            n=3*(j-1);
            for k=1:1
                for l=1:1
                    Pbc(m+k,n+l)=PbcScale;
                end
            end
        end
```

```
            end
        end
        % Constrain boundary nodes in the y direction
        for i=[105]
            m=3*(i-1);
            for j=[105]
                n=3*(j-1);
                for k=2:2
                    for l=2:2
                        Pbc(m+k,n+l)=PbcScale;
                    end
                end
            end
        end
        %save('LiuBeamCrude2Pbc.mat','Pbc')
    elseif mesh==11
        % Constrain boundary nodes in the z direction
        for i=[19,20,22,24,90,91,92,93,154]
            m=3*(i-1);
            for j=[19,20,22,24,90,91,92,93,154]
                n=3*(j-1);
                for k=3:3    % Only constrain z direction
                    for l=3:3    % Only constrain z direction
                        Pbc(m+k,n+l)=PbcScale;
                    end
                end
            end
        end
        % Constrain boundary nodes in the x direction
        for i=[90,154]
            m=3*(i-1);
            for j=[90,154]
                n=3*(j-1);
                for k=1:1
                    for l=1:1
                        Pbc(m+k,n+l)=PbcScale;
                    end
                end
            end
        end
        % Constrain boundary nodes in the y direction
        for i=[154]
            m=3*(i-1);
            for j=[154]
                n=3*(j-1);
                for k=2:2
                    for l=2:2
                        Pbc(m+k,n+l)=PbcScale;
                    end
                end
            end
        end
        %save('PWMesh1Pbc.mat','Pbc')
    end   % End Mesh Conditional
end  % End runPbc Loop

for i=1:3*numn
    for j=1:3*numn
        K(i,j)=K(i,j)*Pbc(i,j);
    end
end

% Generate Displacement Transformation Matrix
```

```
Tdisp(1:24*nume,1:3*numn)=0;
for e=1:nume
    for j=1:8
        % Transformation from Global to Organized by Element
        node=elements(e,j);
        m=24*(e-1)+3*(j-1);
        n=3*(node-1);
        Tdisp(m+1:m+3,n+1:n+3)=[ 1 0 0 ; 0 1 0 ; 0 0 1 ];
    end
end
PTstrain=sparse(Pstrain*Tdisp);
PTstress=sparse(Pstress*Tdisp);

end   % End Subfunction
```

```
function [meshfile,numn,nume,E,Nu,rho,BndN0,BndNF,BndN,FP,F,Pbc,Fbc,...
    gtype,numbndM]= InputParameters(mesh)


if mesh == 1
    meshfile = 'LiuBeamCrude1.xls';
    gtype=4;
    numn = 110;
    nume = 40;
    E = 30E6;
    Nu = 0.3;
    rho(1:nume) = 0.000733;
    BndN0=[1,4,7,8,9,11];
    BndNF=[2,3,5,6,10,12,16:19,30:33,48:51,59:62];
    BndN=[BndNF BndN0];
    Pbc=0;
    Fbc=0;
    load LiuBeamCrude1Pbc.mat
    load LiuBeamCrude1Fbc.mat
    numbndM=7;
    Pc=444.444444;
    FP=-.625*Pc*3/4;
    F(1:3*numn,1)=0;
    for i=1:size(BndNF,2)
        j=BndNF(1,i);
        F(3*(j-1)+3,1)=2*FP;
    end
    for i=[3,6,10,12]  % End Points
        F(3*(i-1)+3,1)=FP;
    end
elseif mesh == 2
    meshfile = 'RectBeamMesh2.xls';
    gtype=1;
    numn = 44;
    nume = 10;
    E = 29E6;
    Nu = 0.29;
    rho(1:nume) = 0.0000007485;
    BndN0=[21,22,43,44];
    BndNF=[1,2,23,24];
    BndN=[BndNF BndN0];
    numbndM=7;
    load RectBeam2Pbc.mat
    load RectBeam2Fbc.mat
    FP=400/4;
    F(1:3*numn,1)=0;
    for i=1:size(BndNF,2)
        j=BndNF(1,i);
        F(3*(j-1)+3,1)=FP;
    end
    MPlot=[10,11,12];
elseif mesh == 3
    meshfile = 'LiuBeamCrude1.xls';
    gtype=4;
    numn = 110;
    nume = 40;
    E = 30E6;
    Nu = 0.3;
    rho(1:nume) = 0.000733;
    BndN0=[4,7,9,11];
    BndNF=[2,5,16,33,51,59];
    BndN=[BndNF BndN0];
    load LiuBeamCrude3Pbc.mat
```

```
        load LiuBeamCrude3Fbc.mat
        numbndM=7;
        FP=-850*3/4;  % For Center
        F(1:3*numn,1)=0;
        for i=1:size(BndNF,2)
            j=BndNF(1,i);
            F(3*(j-1)+3,1)=2*FP;
        end
        for i=[16,33,51,59]  % End Points
            F(3*(i-1)+3,1)=FP;
        end

    elseif mesh == 9
        meshfile = 'LiuBeamCrude2.xls';
        gtype=2;
        numn = 110;
        nume = 40;
        E = 30E6;
        Nu = 0.3;
        rho(1:nume) = 0.000733;
        BndN0=[45,50,55,100,105,110];
        BndNF=[1:11,56:66]; % Original
        BndN=[BndNF BndN0];
        load LiuBeamCrude2Pbc.mat
        load LiuBeamCrude2Fbc.mat
        numbndM=7;
        Pc=444.444444;
        FP=-.625*Pc*3/4; % Original
        F(1:3*numn,1)=0;
        for i=1:size(BndNF,2)
            j=BndNF(1,i);
            F(3*(j-1)+3,1)=2*FP;
        end
        for i=[1,11,56,66]  % Original and Every Other
            F(3*(i-1)+3,1)=FP;
        end
    elseif mesh == 10
        meshfile = 'LiuBeamCrude3.xls';
        gtype=2;
        numn = 132;
        nume = 50;
        E = 30E6;
        Nu = 0.3;
        rho = 0.000733;
        BndN0=[25,26,35,36];
        BndNF=[29:34,109:124];
        BndN=[BndNF BndN0];
        Pbc=0;
        load LiuBeamCrude3Pbc.mat
        Pc=444.444444;
        FP=-.625*Pc*3/4;
        F(1:3*numn,1)=0;
        for i=1:size(BndNF,2)
            j=BndNF(1,i);
            F(3*(j-1)+3,1)=2*FP;
        end
        for i=[1,11,56,66]
            F(3*(i-1)+3,1)=FP;
        end
    elseif mesh == 11
        meshfile = 'PWMesh1.xls';
        gtype=3;
        numn = 236;
```

```
    nume = 156;
    E = 30E6;
    Nu = 0.3;
    rho(1:72) = 0.000733;
    rho(73:96) = 0.026;
    rho(97:156) = 0.000733;
    BndN0=[19,20,22,24,90,91,92,93,154];
    BndNF=[14];
    BndN=[BndNF BndN0];
    numbndM=7;
    load PWMesh1Pbc.mat
    load PWMesh1Fbc.mat
    FP=2000*.25*pi*.1^2/16;
    F(1:3*numn,1)=0;
    for i=[14]
        F(3*(i-1)+2,1)=16*FP;
    end
end

Fsum=0;
for i=1:3*numn
    Fsum=Fsum+F(i,1);
end

end   % End Subfunction
```

```
function [M,K,Pstrain,Pstress,detJstore,Bstore,G_hatstore,Kelstore] = ...
    IsoHexMKBubble(nodes,elements,D,rho,gtype)

% Options
imode=2;   % 0 for None, 1 for Centroid, 2 for Average Correction,
           % 3 for Simo, 4 for Nastran
sri=0;   % Selectively Reduced Integration:  0 for Off, 1 for On

numn=size(nodes,1);
nume=size(elements,1);

% Initialize Matrices
M(1:3*numn,1:3*numn)=0;
K(1:3*numn,1:3*numn)=0;
e(1:8)=0;
xyz(1:8,1:3)=0;
dNdC(1:3,1:8)=0;
dNdC0(1:3,1:8)=0;
dPdC(1:3,1:8)=0;
N(1:3,1:24)=0;
B(1:6,1:24)=0;
B0(1:6,1:24)=0;
G(1:6,1:9)=0;
G_hat(1:6,1:9)=0;
detJstore(1:8*nume,1)=0;
Bstore(1:8*nume,1:6,1:24)=0;
G_hatstore(1:8*nume,1:6,1:9)=0;
Kelstore(1:nume,1:24,1:24)=0;
Pstrain(1:48*nume,1:24*nume)=0;
Jsum(1:6)=0;
Bsum(1:6,1:24)=0;
Bsri(1:8,1:6,1:24)=0;
detJsri(1:8)=0;
Gsri(1:8,1:6,1:9)=0;
D=sparse(D);

if gtype == 1
    psi= [+1 +1 +1 +1 -1 -1 -1 -1];
    eta= [-1 +1 +1 -1 -1 +1 +1 -1];
    zeta=[+1 +1 -1 -1 +1 +1 -1 -1];
    Csri=[ 1 2 5 6  ;   % xy positive
           2 3 6 7  ;   % xz positive
           1 2 3 4  ;   % yz positive
           3 4 7 8  ;   % xy negative
           1 4 5 8  ;   % xz negative
           5 6 7 8 ];   % yz negative
elseif gtype == 2
    psi= [-1 +1 +1 -1 -1 +1 +1 -1];
    eta= [-1 -1 -1 -1 +1 +1 +1 +1];
    zeta=[+1 +1 -1 -1 +1 +1 -1 -1];
    Csri=[ 1 2 5 6  ;   % xy positive
           5 6 7 8  ;   % xz positive
           2 3 6 7  ;   % yz positive
           3 4 7 8  ;   % xy negative
           1 2 3 4  ;   % xz negative
           1 4 5 8 ];   % yz negative
elseif gtype == 3
    psi= [-1 +1 +1 -1 -1 +1 +1 -1];
    eta= [-1 -1 +1 +1 -1 -1 +1 +1];
    zeta=[-1 -1 -1 -1 +1 +1 +1 +1];
    Csri=[ 5 6 7 8  ;   % xy positive
           3 4 7 8  ;   % xz positive
           2 3 6 7  ;   % yz positive
```

```
            1 2 3 4  ;   % xy negative
            1 2 5 6  ;   % xz negative
            1 4 5 8 ];   % yz negative
elseif gtype == 4
    psi= [-1 -1 +1 +1 -1 -1 +1 +1];
    eta= [-1 -1 -1 -1 +1 +1 +1 +1];
    zeta=[-1 +1 +1 -1 -1 +1 +1 -1];
    Csri=[ 2 3 6 7  ;   % xy positive
           5 6 7 8  ;   % xz positive
           3 4 7 8  ;   % yz positive
           1 4 5 8  ;   % xy negative
           1 2 3 4  ;   % xz negative
           1 2 5 6 ];   % yz negative
end

% Gauss Points for a 2x2x2 Array
gauss=8;
if gtype == 1
    psiG= [+1 +1 +1 +1 -1 -1 -1 -1]/3^.5;
    etaG= [-1 +1 +1 -1 -1 +1 +1 -1]/3^.5;
    zetaG=[+1 +1 -1 -1 +1 +1 -1 -1]/3^.5;
    wG(1:8)=1;
elseif gtype == 2
    psiG= [-1 +1 +1 -1 -1 +1 +1 -1]/3^.5;
    etaG= [-1 -1 -1 -1 +1 +1 +1 +1]/3^.5;
    zetaG=[+1 +1 -1 -1 +1 +1 -1 -1]/3^.5;
    wG(1:8)=1;
elseif gtype == 3
    psiG= [-1 +1 +1 -1 -1 +1 +1 -1]/3^.5;
    etaG= [-1 -1 +1 +1 -1 -1 +1 +1]/3^.5;
    zetaG=[-1 -1 -1 -1 +1 +1 +1 +1]/3^.5;
    wG(1:8)=1;
elseif gtype == 4
    psiG= [-1 -1 +1 +1 -1 -1 +1 +1]/3^.5;
    etaG= [-1 -1 -1 -1 +1 +1 +1 +1]/3^.5;
    zetaG=[-1 +1 +1 -1 -1 +1 +1 -1]/3^.5;
    wG(1:8)=1;
end

% Gauss Points for a 3x3x3 Array
gauss3=27;
bG=sqrt(0.6);
psiG3 = [ -bG -bG -bG -bG -bG -bG -bG -bG -bG  0   0   ...
     0   0   0   0   0   0   0  +bG +bG +bG +bG +bG +bG +bG +bG +bG ];
etaG3 = [ -bG -bG -bG  0   0   0  +bG +bG +bG -bG -bG  ...
    -bG  0   0   0  +bG +bG +bG -bG -bG -bG  0   0   0  +bG +bG +bG ];
zetaG3= [ -bG  0  +bG -bG  0  +bG -bG  0  +bG -bG  0   ...
    +bG -bG  0  +bG -bG  0  +bG -bG  0  +bG -bG  0  +bG -bG  0  +bG ];
wpsiG3 = [ 5 5 5 5 5 5 5 5 5 8 8 8 8 8 8 8 8 5 5 5 5 5 5 5 5 5 ]/9;
wetaG3 = [ 5 5 5 8 8 8 5 5 5 5 5 5 8 8 8 5 5 5 5 5 5 8 8 8 5 5 5 ]/9;
wzetaG3= [ 5 8 5 5 8 5 5 8 5 5 8 5 5 8 5 5 8 5 5 8 5 5 8 5 5 8 5 ]/9;
wG3(1:27)=0;
for i=1:27
    wG3(i)=wpsiG3(i)*wetaG3(i)*wzetaG3(i);
end

% Calculate Mass and Stiffness Matrices
for i=1:nume
    for j=1:8
        e(1,j)=elements(i,j);
        for k=1:3
            xyz(j,k)=nodes(e(1,j),k);
        end
```

```
end

Mel(1:24,1:24)=0;
Kel0(1:24,1:24)=0;
if imode==4
    H_G(1:6,1:6)=0;
    E_G(1:6,1:24)=0;
else
    H_G(1:9,1:9)=0;
    E_G(1:9,1:24)=0;
end
Gvol(1:6,1:9)=0;
Vol=0;

% Generate Mass Matrix
for j=1:gauss3
    for k=1:8
        % Generate dNdC Matrix
        dNdC(1:3,k) = 1/8* ...
            [  psi(k)*(1+eta(k)*etaG3(j))*(1+zeta(k)*zetaG3(j))  ;...
               eta(k)*(1+psi(k)*psiG3(j))*(1+zeta(k)*zetaG3(j))  ;...
               zeta(k)*(1+psi(k)*psiG3(j))*(1+eta(k)*etaG3(j))  ];
    end
    J=dNdC*xyz;
    detJ=det(J);

    for k=1:8
        % Generate N Matrix
        Nterm=(1+psi(k)*psiG3(j))*(1+eta(k)*etaG3(j))* ...
              (1+zeta(k)*zetaG3(j));
        N(1:3,3*(k-1)+1:3*(k-1)+3)=1/8*...
            [ Nterm    0      0    ;...
                0    Nterm    0    ;...
                0      0    Nterm ];
    end

    % Generate M Matrix
    Mel=Mel+rho(i)*wG3(j)*detJ*(N)'*N;
end  % End Gauss Loop

% Generate Stiffness Correction Terms
if imode==0   % For None
    % Placeholder
elseif imode==1   % For Centroid
    % Placeholder
elseif imode==2   % For Average Correction
    for j=1:gauss
        dPdC(1:3,1:3) = [ -2*psi(j)     0          0      ;...
                             0     -2*eta(j)      0      ;...
                             0          0     -2*zeta(j) ];

        for k=1:8
            % Generate dNdC Matrix
            dNdC(1:3,k) = 1/8* ...
              [  psi(k)*(1+eta(k)*etaG(j))*(1+zeta(k)*zetaG(j))  ;...
                 eta(k)*(1+psi(k)*psiG(j))*(1+zeta(k)*zetaG(j))  ;...
                 zeta(k)*(1+psi(k)*psiG(j))*(1+eta(k)*etaG(j))  ];
        end
        J=dNdC*xyz;
        dPdxyz=J\dPdC;

        for k=1:3
            G(1:6,3*(k-1)+1:3*(k-1)+3) = ...
```

```
                    [ dPdxyz(1,k)       0             0       ;...
                            0       dPdxyz(2,k)       0       ;...
                            0           0         dPdxyz(3,k) ;...
                      dPdxyz(2,k)   dPdxyz(1,k)       0       ;...
                      dPdxyz(3,k)       0         dPdxyz(1,k) ;...
                            0       dPdxyz(3,k)   dPdxyz(2,k) ];
            end

            detJ2=det(J);
            Gvol=Gvol+wG(j)*detJ2*G;
            Vol=Vol+wG(j)*detJ2;
        end
        Gvol=Gvol/Vol;
    elseif imode==3;   % For Simo
        % Placeholder
    end   % End imode conditional

    % Generate Stiffness Matrix
    for j=1:gauss
        index=8*(i-1)+j;
        dPdC(1:3,1:3) = [ -2*psi(j)     0          0      ;...
                             0     -2*eta(j)      0      ;...
                             0          0     -2*zeta(j) ];

        for k=1:8
            % Generate dNdC Matrix
            dNdC(1:3,k) = 1/8* ...
                [  psi(k)*(1+eta(k)*etaG(j))*(1+zeta(k)*zetaG(j))  ;...
                   eta(k)*(1+psi(k)*psiG(j))*(1+zeta(k)*zetaG(j))  ;...
                   zeta(k)*(1+psi(k)*psiG(j))*(1+eta(k)*etaG(j))  ];
            dNdC0(1:3,k) = 1/8* ...
                [  psi(k)   ;...
                   eta(k)   ;...
                   zeta(k) ];
        end
        J=dNdC*xyz;
        detJ=det(J);
        detJstore(index,1)=detJ;
        dNdxyz=J\dNdC;
        if imode==0   % For None
            J0=dNdC0*xyz;
            dPdxyz=J0\dPdC;
        elseif imode==1   % For Centroid
            J0=dNdC0*xyz;
            dPdxyz=J0\dPdC;
        elseif imode==2   % For Average Correction
            dPdxyz=J\dPdC;
        elseif imode==3   % For Simo
            J0=dNdC0*xyz;
            detJ0=det(J0);
            dPdxyz=J0\dPdC;
        elseif imode==4   % For Nastran
            J0=dNdC0*xyz;
            detJ0=det(J0);
            dPdxyz=-1/2*J0\dPdC;
        end

        % Calculate the Strain Displacement Matrix at the Centroid
        if j==1
            dNdxyz0=J\dNdC0;
            for k=1:8
                % Generate B0 Matrix
                B0(1:6,3*(k-1)+1:3*(k-1)+3) = ...
```

```
            [ dNdxyz0(1,k)         0              0        ; ...
                    0        dNdxyz0(2,k)         0        ; ...
                    0              0        dNdxyz0(3,k)   ; ...
              dNdxyz0(2,k)   dNdxyz0(1,k)         0        ; ...
              dNdxyz0(3,k)         0        dNdxyz0(1,k)   ; ...
                    0        dNdxyz0(3,k)   dNdxyz0(2,k) ];
    end
end

for k=1:8
    % Generate B Matrix
    B(1:6,3*(k-1)+1:3*(k-1)+3) = ...
        [ dNdxyz(1,k)        0             0        ; ...
                0        dNdxyz(2,k)       0        ; ...
                0              0       dNdxyz(3,k)   ; ...
          dNdxyz(2,k)   dNdxyz(1,k)       0        ; ...
          dNdxyz(3,k)        0        dNdxyz(1,k)   ; ...
                0        dNdxyz(3,k)   dNdxyz(2,k) ];
end
Bstore(index,1:6,1:24)=B;

for k=1:3
    G(1:6,3*(k-1)+1:3*(k-1)+3) = ...
        [ dPdxyz(1,k)        0             0        ;...
                0        dPdxyz(2,k)       0        ;...
                0              0       dPdxyz(3,k)   ;...
          dPdxyz(2,k)   dPdxyz(1,k)       0        ;...
          dPdxyz(3,k)        0        dPdxyz(1,k)   ;...
                0        dPdxyz(3,k)   dPdxyz(2,k) ];
end

% Additional Incompatible Modes (Simo)
if imode==0
    % Placeholder
elseif imode==1
    G_hat=G;
    G_hatstore(index,1:6,1:9)=G_hat;
elseif imode==2
    G_hat=G-Gvol;
    G_hatstore(index,1:6,1:9)=G_hat;
elseif imode==3
    Et(1:6,1:6) = ...
        [psi(j)   0       0      0      0      0      ;...
           0    eta(j)    0      0      0      0      ;...
           0      0    zeta(j)   0      0      0      ;...
           0      0       0    psi(j) eta(j)   0      ;...
           0      0       0    psi(j)   0    zeta(j)  ;...
           0      0       0      0    eta(j) zeta(j) ];
    Et(1:6,7:9) = ...
       [  psi(j)*eta(j)   psi(j)*zeta(j)         0           ;...
        -psi(j)*eta(j)         0          eta(j)*zeta(j)    ;...
                0        -psi(j)*zeta(j)  -eta(j)*zeta(j)    ;...
        psi(j)^2-eta(j)^2       0                0           ;...
                0       psi(j)^2-zeta(j)^2       0           ;...
                0               0        eta(j)^2-zeta(j)^2 ];

    F0(1:6,1:3) = ...
        [    J0(1,1)^2    J0(2,1)*J0(1,2) J0(3,1)*J0(1,3)  ;...
          J0(1,2)*J0(2,1)    J0(2,2)^2    J0(3,2)*J0(2,3)  ;...
          J0(1,3)*J0(3,1) J0(2,3)*J0(3,2)    J0(3,3)^2     ;...
          J0(1,1)*J0(2,1) J0(1,2)*J0(2,2)        0         ;...
          J0(1,1)*J0(3,1)        0        J0(1,3)*J0(3,3)  ;...
                 0        J0(2,2)*J0(3,2) J0(2,3)*J0(3,3) ];
```

```
            F0(1:6,4:4) = ...
                [       2*J0(1,1)*J0(1,2)          ;...
                        2*J0(2,1)*J0(2,2)          ;...
                                  0                ;...
                 J0(1,1)*J0(2,2)+J0(1,2)*J0(2,1)   ;...
                                  0                ;...
                                  0               ];
            F0(1:6,5:5) = ...
                [       2*J0(1,1)*J0(1,3)          ;...
                                  0                ;...
                        2*J0(3,1)*J0(3,3)          ;...
                                  0                ;...
                 J0(1,1)*J0(3,3)+J0(1,3)*J0(3,1)   ;...
                                  0               ];
            F0(1:6,6:6) = ...
                [                 0                ;...
                        2*J0(2,2)*J0(2,3)          ;...
                        2*J0(3,2)*J0(3,3)          ;...
                                  0                ;...
                                  0                ;...
                 J0(2,2)*J0(3,3)+J0(2,3)*J0(3,2) ];

            G_hat=detJ0/detJ*inv(F0)'*Et;
            G_hatstore(index,1:6,1:9)=G_hat;
        elseif imode==4   % For Nastran
            clear G_hat
            G_hat(1:6,1:6)=0;
            G_hat(1:6,1:3)= 1/detJ0* ...
                [ psi(j)  0      0     ;...
                    0   eta(j)   0     ;...
                    0     0   zeta(j)  ;...
                    0     0      0     ;...
                    0     0      0     ;...
                    0     0      0    ];
            G_hat(1:6,4:6)= 1/detJ0* ...
                [ psi(j)*eta(j)       0        psi(j)*zeta(j)  ;...
                  psi(j)*eta(j) eta(j)*zeta(j)       0         ;...
                       0       eta(j)*zeta(j) psi(j)*zeta(j)   ;...
                       0             0              0          ;...
                       0             0              0          ;...
                       0             0              0         ];
            G_hatstore(index,1:6,1:6)=G_hat;
        end

        H_G=H_G+wG(j)*detJ*G_hat'*D*G_hat;
        E_G=E_G+wG(j)*detJ*G_hat'*D*B;

        % Generate K Matrix
        Kel0=Kel0+wG(j)*detJ*(B)'*D*B;

        % Store SRI Variables
        if sri==1
            Bsri(j,1:6,1:24)=B;
            detJsri(j)=detJ;
            if imode==4
                Gsri(j,1:6,1:6)=G_hat;
            else
                Gsri(j,1:6,1:9)=G_hat;
            end
        end

    end  % End Gauss Loop
```

```
if sri==0   % No Selective Reduced Integration
    % Placeholder
elseif sri==1   % Selective Reduced Integration
    Jsum(1:6)=0;
    Bsum(1:6,1:24)=0;
    for j=1:3
        for k=1:4
            m=Csri(j,k);
            n=Csri(j+3,k);
            Jsum(j)=Jsum(j)+detJsri(m);
            Jsum(j+3)=Jsum(j+3)+detJsri(n);
            Btemp(1,1:24)=Bsri(m,j+3,1:24);
            Btemp(2,1:24)=Bsri(n,j+3,1:24);
            Bsum(j,1:24)=Bsum(j,1:24)+detJsri(m)*Btemp(1,1:24);
            Bsum(j+3,1:24)=Bsum(j+3,1:24)+detJsri(n)*Btemp(2,1:24);
        end
        for k=1:4
            m=Csri(j,k);
            n=Csri(j+3,k);
            Bsri(m,j+3,1:24)=Bsum(j,1:24)/Jsum(j);
            Bsri(n,j+3,1:24)=Bsum(j+3,1:24)/Jsum(j+3);
        end
    end
    % Reform K Element Matrix
    if imode==4
        E_G(1:6,1:24)=0;
    else
        E_G(1:9,1:24)=0;
    end
    Kel0(1:24,1:24)=0;
    for j=1:8
        index=8*(i-1)+j;
        B(1:6,1:24)=Bsri(j,1:6,1:24);
        Bstore(index,1:6,1:24)=B;
        if imode==4
            G_hat(1:6,1:6)=Gsri(j,1:6,1:6);
        else
            G_hat(1:6,1:9)=Gsri(j,1:6,1:9);
        end
        detJ=detJsri(j);
        E_G=E_G+wG(j)*detJ*G_hat'*D*B;
        Kel0=Kel0+wG(j)*detJ*(B)'*D*B;
    end
end   % End sri Loop

if imode==0
    a(1:9,1:24)=0;
    Kel=Kel0;
else
    a=-H_G\E_G;
    Kel=Kel0+E_G'*a;
end
Kelstore(i,1:24,1:24)=Kel(1:24,1:24);

n=24*(i-1);
for j=1:gauss
    index=8*(i-1)+j;
    B(1:6,1:24)=Bstore(index,1:6,1:24);
    if imode==4
        G_hat(1:6,1:6)=G_hatstore(index,1:6,1:6);
    else
        G_hat(1:6,1:9)=G_hatstore(index,1:6,1:9);
    end
```

```
            m=6*(index-1);
            B_bar=sparse(B+G_hat*a);
            Pstrain(m+1:m+6,n+1:n+24)=B_bar;
            Pstress(m+1:m+6,n+1:n+24)=D*B_bar;
        end

        % Fill in M and K Matrices
        for j=1:8
            for k=1:8
                for m=1:3
                    for n=1:3
                        M(3*(e(j)-1)+m,3*(e(k)-1)+n)=M(3*(e(j)-1)+m, ...
                            3*(e(k)-1)+n)+Mel(3*(j-1)+m,3*(k-1)+n);
                        K(3*(e(j)-1)+m,3*(e(k)-1)+n)=K(3*(e(j)-1)+m, ...
                            3*(e(k)-1)+n)+Kel(3*(j-1)+m,3*(k-1)+n);
                    end
                end
            end
        end
    end

    % Generate Stress Transformation Matrix
    Pstrain=sparse(Pstrain);
    Pstress=sparse(Pstress);

    end   % End Subfunction
```

Appendix B

Matlab Code for Non-linear CMS

This appendix contains the Matlab codes used to solve the dynamic equations of motion, include the plastic response. Individual subroutines are used to determine the transformation matrices for conversion to CMS and orthonormal coordinate systems. Most of the remaining subroutines are called from within the NewmarkIso.m subroutine, including the determination of the state of stress in each element and the calculation of the plastic deformation, if necessary.

```
function [Mbar,Kbar,Mn,Kn,PHI] = ...
    CraigBampton(M,K,Nmodes,Tmodes,stype,BndN,IntN)

numbnd=size(BndN,2);
numn=size(M,1)/3;

% Generate M and K Submatrices
for i=1:numbnd
    LB=3*(i-1)+1;
    LT=3*(BndN(i)-1)+1;
    for j=1:numbnd
        LBB=3*(j-1)+1;
        LTT=3*(BndN(j)-1)+1;
        Mbb(LB:LB+2,LBB:LBB+2)=M(LT:LT+2,LTT:LTT+2);
        Kbb(LB:LB+2,LBB:LBB+2)=K(LT:LT+2,LTT:LTT+2);
    end
    for j=1:numn-numbnd
        LBB=3*(j-1)+1;
        LTT=3*(IntN(j)-1)+1;
        Mbi(LB:LB+2,LBB:LBB+2)=M(LT:LT+2,LTT:LTT+2);
        Kbi(LB:LB+2,LBB:LBB+2)=K(LT:LT+2,LTT:LTT+2);
    end
end
for i=1:numn-numbnd
    LB=3*(i-1)+1;
    LT=3*(IntN(i)-1)+1;
    for j=1:numbnd
        LBB=3*(j-1)+1;
        LTT=3*(BndN(j)-1)+1;
```

```
            Mib(LB:LB+2,LBB:LBB+2)=M(LT:LT+2,LTT:LTT+2);
            Kib(LB:LB+2,LBB:LBB+2)=K(LT:LT+2,LTT:LTT+2);
        end
        for j=1:numn-numbnd;
            LBB=3*(j-1)+1;
            LTT=3*(IntN(j)-1)+1;
            Mii(LB:LB+2,LBB:LBB+2)=M(LT:LT+2,LTT:LTT+2);
            Kii(LB:LB+2,LBB:LBB+2)=K(LT:LT+2,LTT:LTT+2);
        end
    end

    % Assemble New Subdivided M and K Matrices
    Mn = [ Mii  Mib ;
           Mbi  Mbb ];
    Kn = [ Kii  Kib ;
           Kbi  Kbb ];

    % Solve for Internal Normal Modes and Frequencies
    if stype==1
        [phinbar,omegan]=MatlabEig(Mii,Kii,Nmodes);
        %save('LiuBeamCrude','phinbar','omegan');
    elseif stype==2
        tol=.0001;
        choice=3;
        [phinbar,omegan]=Lanczos(Mii,Kii,choice,Nmodes,tol);
    end

    % Solve for Constraint Modes
    phicbar=-Kii\Kib;

    %Generate Tranformation Matrix PHI
    PHI = [        phinbar                phicbar       ;
            zeros(3*numbnd,Nmodes)     eye(3*numbnd)    ];
    Kbar=PHI'*Kn*PHI;
    Mbar=PHI'*Mn*PHI;

    end  % End Subfunctions
```

```
function [dpe]=InternalResistingForceIso(nodes,elements,dstressVr,...
    updatep,detJstore,Bstore)

numn=size(nodes,1);
nume=size(elements,1);

% Initialize Matrices
e(1:8)=0;
dpe(1:3*numn,1)=0;

gauss=8;
if gauss == 8
    wG(1:8)=1;
end

index=0;
for i=1:nume
    for j=1:8
        e(j)=elements(i,j);
    end

    dpel(1:24,1)=0;
    for j=1:8
        index=index+1;
        if updatep(index)==1
            B(1:6,1:24)=Bstore(index,1:6,1:24);
            dpel=dpel+wG(j)*detJstore(index)*B'*dstressVr(1:6,index);
        end
    end

    % Fill in dp Vector
    for j=1:8
        for k=1:3
            m=3*(e(j)-1)+k;
            n=3*(j-1)+k;
            dpe(m,1)=dpe(m,1)+dpel(n,1);
        end
    end

end  % End nume Loop

end   % End Subfunction
```

```
function [u2] = NewmarkIso(M,M0,C,C0,K,K0,Fu,F,T0,TF,Tstep,nodes,...
    elements,D,N,rtype,DirMod,Pbc,Fbc,mesh,Pf,Pu,RF,PTstrain,PTstress,...
    detJstore,Bstore,G_hatstore,Kelstore,Tdisp)

Nmodes=size(M,2);
numn=size(M0,2)/3;
nume=size(elements,1);
numeG=8*nume;
numM0=3*numn;
Nstep=(TF-T0)/Tstep;
imax=3;
u2(1:numM0,1:Nstep)=0;
ddu2(1:numM0,1:2)=0;
u(1:Nmodes,1:2)=0;
ud(1:Nmodes,1:2)=0;
udd(1:Nmodes,1:2)=0;
ui(1:Nmodes,1:imax)=0;
udi(1:Nmodes,1:imax)=0;
uddi(1:Nmodes,1:imax)=0;
pe(1:Nmodes,1)=0;
dpe(1:Nmodes,1)=0;
peg(1:numM0,1)=0;
be(1:Nmodes,1:imax)=0;
ddui(1:Nmodes,1:imax)=0;
dui(1:Nmodes,1:imax)=0;
ddu2(1:numM0,1:imax)=0;
Deps(1:numeG,1:6,1:6)=0;
Beps(1:numeG,1:6,1:24)=0;
for i=1:numeG
    Deps(i,1:6,1:6)=D;
end
if size(RF,2)==1
    useRF=0;
else
    useRF=1;
end

alpha=0;
Beta=1/4*(1-alpha)^2;
gamma=1/2-alpha;
term=0;

M=sparse(M);
M0=sparse(M0);
C(1:Nmodes,1:Nmodes)=0;
C=sparse(C);
C0(1:numM0,1:numM0)=0;
K=sparse(K);
K0=sparse(K0);
F=sparse(F);

M0Pu=sparse(M0*Pu);
Kstar=sparse(M/(Tstep^2*Beta)+gamma*C/(Tstep*Beta)+K);
invKstar=sparse(Kstar\eye(Nmodes));
Kstar0=sparse(M0/(Tstep^2*Beta)+gamma*C0/(Tstep*Beta)+K0);
Kstar0Part=sparse(M0/(Tstep^2*Beta)+gamma*C0/(Tstep*Beta));
NT=N';

strainV(1:6,1:8*nume)=0;
stressV(1:6,1:8*nume)=0;
S(1:6,1:8*nume)=0;
e_p(1:6,1:8*nume)=0;
e_bar(1:8*nume)=0;
```

```
sigma_bar(1:8*nume)=0;
ptest(1:8*nume)=0;
pteste(1:nume)=0;
ptesti(1:8*nume)=0;
tcount=1;
time=Tstep;
pcount=0;
psum=0;
pj=0;
pjj=0;

for j=2:Nstep
    %disp(j)
    if j>=1350
        %term=20000;
    end
    tcount=tcount+1;
    time=time+Tstep;
    if tcount == 100
        disp(time)
        tcount=0;
        if mesh==1
            disp(u2(24,j-1))
        elseif mesh==9
            disp(u2(315,j-1))
        elseif mesh==11
            disp(u2(41,j-1))
        end
    end

    u(1:Nmodes,1)=u(1:Nmodes,2);
    ud(1:Nmodes,1)=ud(1:Nmodes,2);
    udd(1:Nmodes,1)=udd(1:Nmodes,2);

    ui(1:Nmodes,1)=u(1:Nmodes,1);
    astare(1:Nmodes,1)=-1/(Tstep*Beta)*ud(1:Nmodes,1)-...
        (1/(2*Beta)-1)*udd(1:Nmodes,1);
    vstare(1:Nmodes,1)=(1-gamma/Beta)*ud(1:Nmodes,1)+...
        (1-gamma/(2*Beta))*Tstep*udd(1:Nmodes,1);
    bstare(1:Nmodes,1)=M*astare(1:Nmodes,1)+C*vstare(1:Nmodes,1);
    dui(1:Nmodes,1)=0;
    du2(1:numM0,1)=0;
    for i=2:2  % Iterate Elastic Equations of Motion
        be(1:Nmodes,i)=F(1:Nmodes,j)-pe(1:Nmodes,1)-bstare(1:Nmodes,1)-
       ...
            (M/(Tstep^2*Beta)+C*gamma/(Tstep*Beta))*dui(1:Nmodes,i-1);
        ddui(1:Nmodes,i)=invKstar*be(1:Nmodes,i);
        dui(1:Nmodes,i)=dui(1:Nmodes,i-1)+ddui(1:Nmodes,i);
        ui(1:Nmodes,i)=ui(1:Nmodes,i-1)+ddui(1:Nmodes,i);
        [ddu2(1:numM0,1)]=DispConversionMG(ddui(1:Nmodes,i),Nmodes,...
            numM0,rtype,DirMod,N,Pu);
        if useRF==1
            ddu2(1:numM0,1)=ddu2(1:numM0,1)+RF*(Fu(1:numM0,j)-...
                peg(1:numM0,1));
        end
        du2(1:numM0,1)=du2(1:numM0,1)+ddu2(1:numM0,1);

        stressV_p=stressV;
        strainV_p=strainV;
        psum=0;
        pteste(1:nume)=0;
        [dstressV,dstrainV]=StrainStressIso(ddu2(1:numM0,1),nume,D,...
            Deps,psum,ptesti,pteste,PTstrain,PTstress,Beps,Tdisp);
```

```
    e_p_p=e_p;
    e_bar_p=e_bar;
    sigma_bar_p=sigma_bar;

    [e_p,Deps,ptest,e_bar,sigma_bar,peg,sumupdatep,dstressV,...
        stressV,R]=VMPlasticityIso(dstressV,stressV_p,e_p_p,D,...
        nodes,elements,e_bar_p,sigma_bar_p,dstrainV,ptesti,mesh,...
        detJstore,Bstore,2,Deps,Fbc);
    [pe]=ForceConversionGM(peg,Nmodes,numM0,rtype,DirMod,NT,Pf);
    strainV=strainV_p+dstrainV;
end

psum=0;
for index=1:8*nume
    psum=psum+ptest(index);
end

%psum=0;   % To Turn Off Plastic Deformation, Debugging
if psum == 0
    u2(1:numM0,j)=u2(1:numM0,j-1)+du2(1:numM0,1);

    u(1:Nmodes,2)=ui(1:Nmodes,i);
    udd(1:Nmodes,2)=astare(1:Nmodes,1)+...
        1/(Tstep^2*Beta)*dui(1:Nmodes,i);
    ud(1:Nmodes,2)=vstare(1:Nmodes,1)+...
        gamma/(Tstep*Beta)*dui(1:Nmodes,i);
else
    pcount=pcount+1;
    if pcount==1
        %term=20000;  % Used for debugging
    end
    du2(1:numM0,1)=ddu2(1:numM0,1);
    stressV_i=stressV;
    dstressV_i(1:6,1:numeG)=0;
    dstrainV_i(1:6,1:numeG)=0;
    ptest_p=ptest;

    for i=3:imax
        [Kep,Beps,pteste]=PseudoforceIso(numn,nume,elements,D,...
            Deps,Beps,ptest_p,Pbc,detJstore,Bstore,G_hatstore,...
            Kelstore);

        bstarg(1:numM0,1)=M0Pu*astare(1:Nmodes,1);
        ddui(1:Nmodes,i)=invKstar*Pf*Kstar0*((Kstar0Part+Kep)\...
            (Fu(1:numM0,j)-peg(1:numM0,1)-...
            Kstar0Part*Pu*dui(1:Nmodes,i-1)-bstarg(1:numM0,1)));
        dui(1:Nmodes,i)=dui(1:Nmodes,i-1)+ddui(1:Nmodes,i);
        ui(1:Nmodes,i)=ui(1:Nmodes,i-1)+ddui(1:Nmodes,i);

   [ddu2(1:numM0,i)]=DispConversionMG(ddui(1:Nmodes,i),Nmodes,...
            numM0,rtype,DirMod,N,Pu);
        du2(1:numM0,1)=du2(1:numM0,1)+ddu2(1:numM0,i);


   [dstressV,dstrainV]=StrainStressIso(ddu2(1:numM0,i),nume,D,...
            Deps,psum,ptest_p,pteste,PTstrain,PTstress,Beps,Tdisp);
        dstressV_i=dstressV_i+dstressV;
        dstrainV_i=dstrainV_i+dstrainV;

        [e_p,Depsee,ptestee,e_bar,sigma_bar,peg,sumupdatepee,...
            dstressV,stressV,R]=VMPlasticityIso(dstressV_i,...
            stressV_i,e_p_p,D,nodes,elements,e_bar_p,sigma_bar_p,...
```

```
                    dstrainV_i,ptest_p,mesh,detJstore,Bstore,i,Deps,Fbc);
                [pe]=ForceConversionGM(peg,Nmodes,numM0,rtype,DirMod,NT,Pf);
                strainV=strainV_p+dstrainV;

            end   % i iteration loop

            u2(1:numM0,j)=u2(1:numM0,j-1)+du2(1:numM0,1);

            u(1:Nmodes,2)=ui(1:Nmodes,i);
            udd(1:Nmodes,2)=astare(1:Nmodes,1)+...
                1/(Tstep^2*Beta)*dui(1:Nmodes,i);
            ud(1:Nmodes,2)=vstare(1:Nmodes,1)+...
                gamma/(Tstep*Beta)*dui(1:Nmodes,i);

        end   % psum loop

        % Fill in Displacement Matrix if Terminated
        if term >= 10000
            for k=j+1:Nstep
                u2(:,k)=u2(:,j);
            end
            break
        end
    end   % Nstep Loop

    end   % End Subfunction
```

```
function [N,lambda,FqO,numrgd]=Orthonormalize(Mbar,Kbar,Fq,Nmodes,...
    Tmodes,numbndM)

% Orthonormalize Craig Bampton Modes
[Ni,lambdai] = MatlabEig(Mbar,Kbar,Tmodes);

% Locate Rigid Body Modes
for i=1:Tmodes
    if lambdai(i,1) > 1
        numrgd=i-1;
        break
    end
end

% Move Boundary Condition Modes
N=Ni;
lambda=lambdai;

clear Ni lambdai

% Normalize N Modal Matrix
Nscale=N'*Mbar*N;
for i=1:(Tmodes-numrgd)
    NN=sqrt(Nscale(i,i));
    for j=1:Tmodes
        N(j,i)=N(j,i)/NN;
    end
end

FqO=N'*Fq;

end   % End Subfunction
```

```
function [Kep,Beps,pteste]=PseudoforceIso(numn,nume,elements,D,Deps,...
    Beps,ptest,Pbc,detJstore,Bstore,G_hatstore,Kelstore)

imode=2;    % 0 for None, 1 for Centroid, 2 for Average Correction,
            % 3 for Simo, 4 for Nastran

% Initialize Matrices
Kep(1:3*numn,1:3*numn)=0;
e(1:8)=0;
Dep(1:6,1:6)=0;
pteste(1:nume)=0;

gauss=8;
if gauss == 8
    wG(1:8)=1;
end

% Calculate Mass and Stiffness Matrices
for i=1:nume
    Kelsum=0;
    for j=1:8
        e(j)=elements(i,j);
        index=8*(i-1)+j;
        if ptest(index) == 1
            Kelsum=Kelsum+1;
            pteste(i)=1;
        end
    end

    if Kelsum == 0
        Kel(1:24,1:24)=Kelstore(i,1:24,1:24);
    else
        % Generate Stiffness Matrix
        KelP(1:24,1:24)=0;
        if imode==4
            H_G(1:6,1:6)=0;
            E_G(1:6,1:24)=0;
        else
            H_G(1:9,1:9)=0;
            E_G(1:9,1:24)=0;
        end

        for j=1:gauss
            index=8*(i-1)+j;
            if ptest(index) == 1
                % Reform Dep Matrix for Element
                Dep(1:6,1:6)=Deps(index,1:6,1:6);
            else
                Dep=D;
            end

            detJ(1,1)=detJstore(index,1);

            if imode==4
                G_hat(1:6,1:6)=G_hatstore(index,1:6,1:6);
            else
                G_hat(1:6,1:9)=G_hatstore(index,1:6,1:9);
            end
            B(1:6,1:24)=Bstore(index,1:6,1:24);

            H_G=H_G+wG(j)*detJ*G_hat'*Dep*G_hat;
            E_G=E_G+wG(j)*detJ*G_hat'*Dep*B;
```

```
                % Generate K Matrix
                KelP=KelP+wG(j)*detJ*(B)'*Dep*B;
            end

            if imode==0
                Kel=KelP;
            else
                a=-H_G\E_G;
                Kel=KelP+E_G'*a;
            end

            for j=1:gauss
                index=8*(i-1)+j;
                B(1:6,1:24)=Bstore(index,1:6,1:24);
                if imode==4
                    G_hat(1:6,1:6)=G_hatstore(index,1:6,1:6);
                else
                    G_hat(1:6,1:9)=G_hatstore(index,1:6,1:9);
                end
                if imode==0
                    Beps(index,1:6,1:24)=B;
                else
                    Beps(index,1:6,1:24)=B+G_hat*a;
                end
            end
        end   % End Kelsum Conditional

        % Fill in K Matrix
        for j=1:8
            for k=1:8
                for m=1:3
                    for n=1:3
                        Kep(3*(e(j)-1)+m,3*(e(k)-1)+n)=Kep(3*(e(j)-1)+m,...
                            3*(e(k)-1)+n)+Kel(3*(j-1)+m,3*(k-1)+n);
                    end
                end
            end
        end
    end

    for i=1:3*numn
        for j=1:3*numn
            Kep(i,j)=Kep(i,j)*Pbc(i,j);
        end
    end
    Kep=sparse(Kep);

    end   % End Subfunction
```

```
function [dstressV,dstrainV]=StrainStressIso(du,nume,D,Deps,psum,...
    ptest,pteste,PTstrain,PTstress,Beps,Tdisp)

dstrainV(1:6,1:8*nume)=0;
dstressV(1:6,1:8*nume)=0;

% Generate Global Strain Vector
dstrainVG=PTstrain*du;
dstressVG=PTstress*du;

if psum~=0
    dus=Tdisp*du;
end

% Generate Strain and Stress Vectors
for e=1:nume
    if pteste(e)==0
        for j=1:8
            index=8*(e-1)+j;
            m=6*(index-1);
            dstrainV(1:6,index)=dstrainVG(m+1:m+6,1);
            dstressV(1:6,index)=dstressVG(m+1:m+6,1);
        end
    else
        for j=1:8
            index=8*(e-1)+j;
            m=24*(e-1);

            B(1:6,1:24)=Beps(index,1:6,1:24);
            dstrainV(1:6,index)=B*dus(m+1:m+24,1);
            if ptest(index)==0
                dstressV(1:6,index)=D*dstrainV(1:6,index);
            else
                Dep(1:6,1:6)=Deps(index,1:6,1:6);
                dstressV(1:6,index)=Dep*dstrainV(1:6,index);
            end
        end
    end
end

end   % End Subfunction
```

```
function [e_p,Deps,ptest,e_bar,sigma_bar,pe,sumupdatep,dstressV,...
    stressV,R]=VMPlasticityIso(dstressV,stressV_p,e_p_p,D,nodes,...
    elements,e_bar_p,sigma_bar_p,dstrainV,ptest,mesh,detJstore,Bstore,...
    iteration,Deps,Fbc)

nume=size(elements,1);
numeG=8*nume;
if mesh==1
    H_0=50000;
    H_L=125000;
    e_L_bar=.01166667;
    m=335410;
    n=0.5;
    del_lam_1=.000001;
elseif mesh==2
    H_0=360000;
    H_L=900000;
    e_L_bar=.015;
    m=100000;
    n=0.5;
    del_lam_1=.000001;
elseif mesh==9
    H_0=50000;
    H_L=125000;
    e_L_bar=.010;
    m=335410;
    n=0.5;
    del_lam_1=.000001;
elseif mesh==11
    H_0=900000;
    H_L=1650000;
    e_L_bar=.013;
    del_lam_1=.000001;
end
E_T=(H_L-H_0)/e_L_bar;
F_error=.0001;
c_max=50;
sigma_bar(1:numeG)=0;
del_sigma_bar(1:numeG)=0;
R(1:numeG)=1;
HR(1:numeG)=0;
H(1:numeG)=0;
F(1:numeG)=0;
F_p(1:numeG)=0;
F_c(1:numeG)=0;
dstressVr(1:6,1:numeG)=0;
stressV(1:6,1:numeG)=0;

Constit_type=1;

if Constit_type==1
    %del_lam_1=.00000000001;
elseif Constit_type==2
    del_lam_1=.00000001;
elseif Constit_type==3
    del_lam_1=.00000001;
end

A(1:numeG)=0;
P(1:numeG)=0;
S(1:6,1:numeG)=0;
del_lam_p(1:numeG)=0;
del_lam_c(1:numeG)=0;
```

```
e_p(1:6,1:numeG)=0;
del_e_p(1:6,1:numeG)=0;
e_bar(1:numeG)=e_bar_p(1:numeG);
del_e_bar(1:numeG)=0;
del_e_e(1:6,1:numeG)=0;
skip(1:numeG)=0;
updatep(1:numeG)=0;

for e=1:nume
for gauss=1:8
    index=8*(e-1)+gauss;

    stressV(1:6,index)=stressV_p(1:6,index)+dstressV(1:6,index);

    if Constit_type == 1
        HR(index)=H_0+E_T*e_bar_p(index);
    elseif Constit_type == 2
        HR(index)=H_0;
    elseif Constit_type == 3
        HR(index)=H_0+m*(e_bar(index))^n;
    end
    P(index)=-1/3*(stressV(1,index)+stressV(2,index)+stressV(3,index));
    for i=1:3
        j=i+3;
        S(i,index)=stressV(i,index)+P(index);
        S(j,index)=sqrt(2)*stressV(j,index);
    end
    sigma_bar(index)=sqrt(3/2)*sqrt(S(1:6,index)'*S(1:6,index));
    del_sigma_bar(index)=sigma_bar(index)-sigma_bar_p(index);

    if ptest(index)==1   % Did Yield in Previous Step
        if sigma_bar(index)>sigma_bar_p(index)   % Still Plastic
            R(index)=1;
            skip(index)=0;
            updatep(index)=0;
        else   % Unloading Elastic
            e_p(1:6,index)=e_p_p(1:6,index);
            e_bar(index)=e_bar_p(index);
            ptest(index)=0;
            R(index)=0;
            skip(index)=1;
            updatep(index)=0;
        end
    elseif ptest(index)==0  % Did Not Yield in Previous Step
        if sigma_bar(index)<HR(index)   % Still Elastic
            e_p(1:6,index)=e_p_p(1:6,index);
            e_bar(index)=e_bar_p(index);
            R(index)=0;
            skip(index)=1;
            updatep(index)=0;
        else   % First Plastic Deformation
            ptest(index)=1;
            skip(index)=0;
            R(index)=(sigma_bar(index)-HR(index))/del_sigma_bar(index);
            dstressVr(1:6,index)=(1-R(index))*dstressV(1:6,index);
            updatep(index)=1;
        end
    end

    if skip(index)==1
        c=1;
    elseif skip(index)==0
    for c=1:c_max
```

```
% Calculate F
if Constit_type == 1   % Linear Hardening
    H(index)=H_0+E_T*e_bar(index);
    F(index)=sigma_bar(index)-H(index);
elseif Constit_type == 2   % Perfectly Plastic
    F(index)=sigma_bar(index)-H_0;
elseif Constit_type == 3   % Power Law
    H(index)=H_0+m*(e_bar(index))^n;
    F(index)=sigma_bar(index)-H(index);
end

if ((F(index) <= F_error) && (c==1))
    e_p(1:6,index)=e_p_p(1:6,index);
    e_bar(index)=e_bar_p(index);
    break
end
if (abs(F(index)) <= F_error)
    break
end

algor=2;
if algor==1   % Newton-Raphson
    if c==1
        F_c(index)=F(index);
        del_lam_c(index)=del_lam_1;
    else
        F_p(index)=F_c(index);
        F_c(index)=F(index);
        del_lam_p(index)=del_lam_c(index);
        del_lam_c(index)=del_lam_p(index)+...
            (F_c(index)/(F_p(index)-
F_c(index)))*del_lam_p(index);
    end
elseif algor==2   % Bi-section Method
    if c==1
        F_a=F(index);
        lam_a=0;
        del_lam_c(index)=del_lam_1;
    elseif c==2
        lam_b=del_lam_c(index);
        F_b=F(index);
        del_lam_c(index)=lam_a+(lam_b-lam_a)/2;
        if F_b>0
            index
            F_b
        end
    else
        F_p=F(index);
        if F_a*F_p>0
            lam_a=del_lam_c(index);
            F_a=F_p;
            del_lam_c(index)=lam_a+(lam_b-lam_a)/2;
        else
            lam_b=del_lam_c(index);
            F_b=F_p;
            del_lam_c(index)=lam_a+(lam_b-lam_a)/2;
        end
    end
end

del_e_p(1:6,index)=del_lam_c(index)*S(1:6,index);

if Constit_type == 1
```

```
            del_e_bar(index)=sqrt(2/3)*sqrt(del_e_p(1:6,index)'*...
                del_e_p(1:6,index));
            e_bar(index)=e_bar_p(index)+del_e_bar(index);
        elseif Constit_type == 2
            % Update Stress Vector
            del_e_p(4:6,index)=del_e_p(4:6,index)*sqrt(2);
            del_e_e(1:6,index)=dstrainV(1:6,index)-del_e_p(1:6,index);
            stressV(1:6,index)=stressV_p(1:6,index)+D*del_e_e(1:6,index);
            %stressV(1:6,index)
            P(index)=-1/3*(stressV(1,index)+stressV(2,index)+...
                stressV(3,index));
            for i=1:3
                j=i+3;
                S(i,index)=stressV(i,index)+P(index);
                S(j,index)=sqrt(2)*stressV(j,index);
                %S(j,index)=stressV(j,index);
            end
            sigma_bar(index)=sqrt(3/2)*sqrt(S(1:6,index)'*S(1:6,index));
        elseif Constit_type == 3
            del_e_bar(index)=sqrt(2/3)*sqrt(del_e_p(1:6,index)'*...
                del_e_p(1:6,index));
            e_bar(index)=e_bar_p(index)+del_e_bar(index);
        end

    end   % End c Loop
    end   % Skip Loop

    if c==1
        % Placeholder
    else
        if c==c_max
            disp(['Did Not Converge:  ',num2str(index),'  ',...
                num2str(F(index))])
        end
        if Constit_type == 1
            del_e_p(4:6,index)=del_e_p(4:6,index)*sqrt(2);
            e_p(1:6,index)=e_p_p(1:6,index)+del_e_p(1:6,index);
            a(1:3,1)=S(1:3,index);
            a(4:6,1)=S(4:6,index)*sqrt(2);
            dstressVA(1:3,1)=R(index)*dstressV(1:3,index);
            dstressVA(4:6,1)=R(index)*dstressV(4:6,index)*sqrt(2);
            A(index)=a(1:6,1)'*dstressVA(1:6,1)/del_lam_c(index);
        elseif Constit_type == 2
            a(1:3,1)=S(1:3,index);
            a(4:6,1)=S(4:6,index)*sqrt(2);
            e_p(1:6,e)=e_p_p(1:6,e)+del_e_p(1:6,1);
            del_e_p(4:6,index)=del_e_p(4:6,index)/sqrt(2);
            del_e_bar(index)=sqrt(2/3)*sqrt(del_e_p(1:6,index)'*...
                del_e_p(1:6,index));
            e_bar(index)=e_bar_p(index)+del_e_bar(index);
            del_e_p(4:6,index)=del_e_p(4:6,index)*sqrt(2);
            A(index)=0;
            dstressVr(1:6,index)=0;
        elseif Constit_type == 3
            a(1:3,1)=S(1:3,index);
            a(4:6,1)=S(4:6,index)*sqrt(2);
            del_e_p(4:6,index)=del_e_p(4:6,index)*sqrt(2);
            e_p(1:6,index)=e_p_p(1:6,index)+del_e_p(1:6,index);
            E_T=(H(index)-HR(index))/del_e_bar(index);
            A(index)=1/del_lam_c(index)*(2/3*H(index)*E_T)*...
                del_e_bar(index);
        end
```

```
            if iteration == 2
                Dep=D-(D*a)*((a'*D)/(A(index)+(a'*D*a)));
                Deps(index,1:6,1:6)=Dep;
            else
                Dep(1:6,1:6)=Deps(index,1:6,1:6);
            end
            if iteration == 2
                dstressV(1:6,index)=(1-R(index))*dstressV(1:6,index)+...
                    R(index)*Dep*dstrainV(1:6,index);
                stressV(1:6,index)=stressV_p(1:6,index)+dstressV(1:6,index);
                sigma_bar(index)=sigma_bar_p(index);
                e_p(1:6,index)=e_p_p(1:6,index);
                e_bar(index)=e_bar_p(index);
            else
                % Placeholder
            end
        end
    end   % End Gauss Loop
    end   % End Element Loop

    sumupdatep=0;
    for index=1:numeG
        sumupdatep=sumupdatep+updatep(index);
    end

    updatep(1:numeG)=1;
    [pe]=InternalResistingForceIso(nodes,elements,stressV,updatep,...
        detJstore,Bstore);
    pe=Fbc*pe;

    end   % End Subfunction
```

```
function [dpr]=ForceConversionGM(dp,Nmodes,numM0,rtype,DirMod,NT,Pf)

if rtype == 1
    if DirMod ==0
        dpr(1:Nmodes,1)=dp(1:numM0,1);
    elseif DirMod == 1
        dpr(1:Nmodes,1)=NT*dp(1:numM0,1);
    end
elseif rtype == 2
    dpr(1:Nmodes,1)=Pf*dp(1:numM0,1);
end

end   % End Subfunction
```

```
function [ddu]=DispConversionMG(ddum,Nmodes,numM0,rtype,DirMod,N,Pu)

if rtype == 1
    if DirMod ==0
        ddu(1:numM0,1)=ddum(1:Nmodes,1);
    elseif DirMod == 1
        ddu(1:numM0,1)=N*ddum(1:Nmodes,1);
    end
elseif rtype == 2
    ddu(1:numM0,1)=Pu*ddum(1:Nmodes,1);
end

end   % End Subfunction
```

Appendix C

Supplemental Matlab Code

This appendix contains the supplemental Matlab code for various calculations. The supplemental codes are used for calculation of the effective modal mass parameters, creating plots, solving an eigen-problem, and ordering matrices. Some of the subroutines in this section are used indirectly in the solution of the equations of motion and are called by other subroutines.

```
function [Gamma,Gamma3,meff,meff3]=...
    EffectiveModalMass(numM0,numbnd,Mn,Kn,mass)

numii=numM0-3*numbnd;
Mii=Mn(1:numii,1:numii);
Kii=Kn(1:numii,1:numii);
[Ni,lambdai] = MatlabEig(Mii,Kii,numii);

% Locate Rigid Body Modes
numrgd=0;
for i=1:numii
    if lambdai(i,1) > 1
        numrgd=i-1;
        break
    end
end
nterms=numii-numrgd;

% Move Boundary Condition Modes
N=Ni;
lambda=lambdai;

% Scale Transformation Matrix
Nscale=N'*Mii*N;
for i=1:nterms
    NN=sqrt(Nscale(i,i));
    for j=1:nterms
        N(j,i)=N(j,i)/NN;
    end
end
Mbar=eye(nterms);
```

```
% Define Influence Vector
r3(1:nterms,1:3)=0;
for i=1:nterms/3
    n=3*(i-1);
    r3(n+1:n+3,1:3)=eye(3);
end
r(1:nterms,1)=1;

% Determine Coefficient Vector
L=N'*Mii*r;
L3=N'*Mii*r3;

% Determine Modal Participation Factor Matrix
Gamma(1:nterms,1)=0;
Gamma3(1:nterms,1:3)=0;
for i=1:nterms
    Gamma(i,1)=L(i,1)/Mbar(i,i);
    for j=1:3
       Gamma3(i,j)=L3(i,j)/Mbar(i,i);
    end
end

% Determine Effective Modal Mass
meff(1:nterms,1:4)=0;
meff3(1:nterms,1:6)=0;
for i=1:nterms
    meff(i,1)=i;
    meff(i,2)=(L(i,1))^2/Mbar(i,i);
    for j=1:3
        meff3(i,j)=(L3(i,j))^2/Mbar(i,i);
    end
end

% Sum Modal Mass
summeff=0;
summeff3(1,1:3)=0;
for i=1:nterms
    summeff=summeff+meff(i,2);
    meff(i,3)=summeff;
    for j=1:3
        summeff3(1,j)=summeff3(1,j)+meff3(i,j);
        meff3(i,j+3)=summeff3(1,j);
    end
end
for i=1:nterms
    meff(i,4)=(3*mass-summeff+meff(i,3))/(3*mass);
    for j=1:3
        meff3(i,j+6)=(mass-summeff3(1,j)+meff3(i,j+3))/mass;
    end
end
summeff
summeff3
meff(:,4)
meff3(:,7:9)
end   % End Subfunction
```

```
function [D]=ElasticityIsotropic(E,Nu)

f=1-Nu;
g=(1-2*Nu)/2;

[D]=E/((1+Nu)*(1-2*Nu))*...
    [  f   Nu   Nu    0    0    0  ;...
      Nu    f   Nu    0    0    0  ;...
      Nu   Nu    f    0    0    0  ;...
       0    0    0    g    0    0  ;...
       0    0    0    0    g    0  ;...
       0    0    0    0    0    g  ];

end   % End Subroutine
```

```
function [modes,omega] = MatlabEig(M,K,Nmodes)

[mod,val]=eig(K,M);

val=abs(val);
numval=size(M,1);

% Pre-allocate Variables
omegar(1:numval,1)=0;
omegam(1:numval,1:2)=0;
omega(1:Nmodes,1:3)=0;
modess(1:numval,1:numval)=0;

for i=1:numval
    omegar(i,1)=sqrt(val(i,i))/(2*pi);
end

for i=1:numval
    omegam(i,1:2)=[i omegar(i,1)];
end
omegams=sortrows(omegam,2);
for i=1:numval
    omega(i,1:3)=[omegams(i,2) omegams(i,1) i];
end

for i=1:numval
    modess(1:numval,i)=mod(1:numval,omega(i,2));
end

% Truncate Solution to Nmodes
useupper=0;
if useupper==0
    modes=modess(1:numval,1:Nmodes);
    omega=omega(1:Nmodes,1:3);
else
    if Nmodes>20
        upperM=Nmodes-20;
        lowerM=Nmodes-upperM;
    else
        upperM=0;
        lowerM=Nmodes;
    end
    if upperM>30
        upperM=30;
        lowerM=Nmodes-upperM;
    end
    modes(1:numval,1:lowerM)=modess(1:numval,1:lowerM);
    modes(1:numval,lowerM+1:lowerM+upperM)=...
        modess(1:numval,numval-upperM+1:numval);
    omega(1:lowerM,1:3)=omega(1:lowerM,1:3);
    omega(lowerM+1:lowerM+upperM,1:3)=omega(numval-upperM+1:numval,1:3);
end

end   % End Subfunction
```

```
function ModePlots(modes,nodes,elements,MPlot)

numn=size(modes,1)/3;
nume=size(elements,1);
nump=size(MPlot,2);

for i=1:nump
    rrv=MPlot(i);
    figure(MPlot(i))
    scale=0.01;
    for i=1:numn
        G(i,1)=modes(3*(i-1)+1,rrv);
        H(i,1)=scale*G(i,1)+nodes(i,1);
        G(i,2)=modes(3*(i-1)+2,rrv);
        H(i,2)=scale*G(i,2)+nodes(i,2);
        G(i,3)=modes(3*(i-1)+3,rrv);
        H(i,3)=scale*G(i,2)+nodes(i,3);
    end
    plot3(H(1:numn,1),H(1:numn,2),H(1:numn,3),'b+',nodes(1:numn,1),...
    nodes(1:numn,2),nodes(1:numn,3),'r+');
end

end   % End Subfunction
```

```
function [fr,P]=Order2(f,IntN,BndNF,BndN0)

[numN,numS]=size(f);
numint=size(IntN,2);
numbndF=size(BndNF,2);
numbnd0=size(BndN0,2);

% Initialize ur Matrix
fr(1:3*(numint+numbndF+numbnd0),1:numS)=0;
P(1:3*(numint+numbndF+numbnd0),1:3*(numint+numbndF+numbnd0))=0;

% Populate with Displacements of Interior Nodes
for i=1:numint
    nr=3*(IntN(i)-1)+1;
    nu=3*(i-1)+1;
    fr(nu:nu+2,1:numS)=f(nr:nr+2,1:numS);
    P(nu:nu+2,nr:nr+2)=[1 0 0; 0 1 0; 0 0 1];
end

% Populate with Displacements of Force Boundary Nodes
for i=1:numbndF
    nr=3*(BndNF(i)-1)+1;
    j=numint+i;
    nu=3*(j-1)+1;
    fr(nu:nu+2,1:numS)=f(nr:nr+2,1:numS);
    P(nu:nu+2,nr:nr+2)=[1 0 0; 0 1 0; 0 0 1];
end

% Populate with Displacements of Boundary Nodes
for i=1:numbnd0
    nr=3*(BndN0(i)-1)+1;
    j=numint+numbndF+i;
    nu=3*(j-1)+1;
    fr(nu:nu+2,1:numS)=f(nr:nr+2,1:numS);
    P(nu:nu+2,nr:nr+2)=[1 0 0; 0 1 0; 0 0 1];
end

end   % End Subfunction
```

```
function [nodes,elements] = ReadMeshData(meshfile,numn,nume)

rangen=['B4:D',num2str(numn+3)];
rangee=['B4:I',num2str(nume+3)];

[nodes]=xlsread(meshfile,'Nodes',rangen);
[elements]=xlsread(meshfile,'Elements',rangee);

end   % End Subfunction
```

```
function [M,K,IntN]=ReduceMK(M,K,BndN0)

numn=size(M,1)/3;
numbnd=size(BndN0,2);

% Create Vector of Interface Nodes
IntN(1:(numn-numbnd))=0;
c=0;
for i=1:BndN0(1)-1
    c=c+1;
    IntN(c)=i;
end
for i=1:numbnd-1
    for j=BndN0(i)+1:BndN0(i+1)-1
        c=c+1;
        IntN(c)=j;
    end
end
for i=BndN0(numbnd)+1:numn
    c=c+1;
    IntN(c)=i;
end

for i=1:numn-numbnd
    LB=3*(i-1)+1;
    LT=3*(IntN(i)-1)+1;
    for j=1:numn-numbnd;
        LBB=3*(j-1)+1;
        LTT=3*(IntN(j)-1)+1;
        Mii(LB:LB+2,LBB:LBB+2)=M(LT:LT+2,LTT:LTT+2);
        Kii(LB:LB+2,LBB:LBB+2)=K(LT:LT+2,LTT:LTT+2);
    end
end

Mii=M(1:3*40,1:3*40);
Kii=K(1:3*40,1:3*40);

M=Mii;
K=Kii;

end   % End Subfunction
```

```
function [u]=Reorder(u,IntN,BndNF,BndN0)

[numN,numS]=size(u);
numint=size(IntN,2);
numbndF=size(BndNF,2);
numbnd0=size(BndN0,2);

% Initialize ur Matrix
ur(1:3*(numint+numbndF+numbnd0),1:numS)=0;

% Populate with Displacements of Interior Nodes
for i=1:numint
    nr=3*(IntN(i)-1)+1;
    nu=3*(i-1)+1;
    ur(nr:nr+2,1:numS)=u(nu:nu+2,1:numS);
end

% Populate with Displacements of Force Boundary Nodes
for i=1:numbndF
    nr=3*(BndNF(i)-1)+1;
    j=numint+i;
    nu=3*(j-1)+1;
    ur(nr:nr+2,1:numS)=u(nu:nu+2,1:numS);
end

% Populate with Displacements of Boundary Nodes
for i=1:numbnd0
    nr=3*(BndN0(i)-1)+1;
    j=numint+numbndF+i;
    nu=3*(j-1)+1;
    ur(nr:nr+2,1:numS)=u(nu:nu+2,1:numS);
end

u=ur;

end   % End Subfunction
```

References

[1] Hahn, H., *Rigid Body Dynamics of Mechanisms*, Berlin: Springer-Verelag, 2002.

[2] Anitescu, M., Potra, F., and Stewart, D., 'Time-stepping for Three-dimensional Rigid Body Dynamics', *Computer Methods in Applied Mechanics and Engineering*, Vol. 177, No. 3, pp. 183-197, 1999.

[3] Herting, D., MSC/NASTRAN Advanced Dynamic Analysis User's Guide, Version 70, Los Angeles, California: MacNeal/Schwendler Corporation, 1997.

[4] Vakakis, A., 'Non-linear Normal Modes (NNMs) and their Applications in Vibration Theory: An Overview', *Mechanical Systems and Signal Processing*, Vol. 11, No. 1, pp. 3-22, 1997.

[5] Wang, F. and Bajaj, A., 'Nonlinear Normal Modes in Multi-mode Models of an Inertially Coupled Elastic Structure', *Nonlinear Dynamics*, Vol. 47, No. 1, pp. 25-47, 2007.

[6] Morris, N., 'The Use of Modal Superposition in Nonlinear Dynamics', *Computers & Structures*, Vol. 7, No. 1, pp. 65-72, 1977.

[7] Remseth, S., 'Nonlinear Static and Dynamic Analysis of Framed Structures', *Computers & Structures*, Vol. 10, No. 6, pp. 879-897, 1979.

[8] Bathe, K. and Gracewski, S., 'On Nonlinear Dynamic Analysis Using Substructuring and Mode Superposition', *Computers & Structures*, Vol. 13, No. 6, pp. 699-707, 1981.

[9] Noor, A., 'Recent Advances in Reduction Methods for Nonlinear Problems', *Computers & Structures*, Vol. 13, No. 1, pp. 31-44, 1981.

[10] Idelsohn, S. and Cardona, A., 'Reduction Methods and Explicit Time Integration Technique in Structural Dynamics', *Advances in Engineering Software*, Vol. 6, No. 1, pp. 36-44, 1984.

[11] Idelsohn, S. and Cardona, A., 'A Load-dependent Basis for Reduced Nonlinear Structural Dynamics', *Computers & Structures*, Vol. 20, No. 1, pp. 203-210, 1985.

[12] Chang, C. and Engblom, J., 'Nonlinear Dynamical Response of Impulsively Loaded Structures: A Reduced Basis Approach', *American Institute of Aeronautics and Aeronautics Journal*, Vol. 29, No. 4, pp. 613-618, 1991

[13] Qu, Z., 'Model Reduction for Dynamical Systems with Local Nonlinearities', *American Institute of Aeronautics and Aeronautics Journal*, Vol. 40, No. 2, pp. 327-333, 2002

[14] de Klerk, D., Rixen, D., and Voormeeren, S., 'General Framework for Dynamic Substructuring: History, Review, and Classification of Techniques', *American Institute of Aeronautics and Aeronautics Journal*, Vol. 46, No. 5, pp. 1169-1181, 2008.

[15] Bond, J. and Khraishi, T., 'Non-linear Dynamic Modelling using Component Mode Synthesis', *International Journal of Theoretical and Applied Multiscale Mechanics*, Vol. 1, No. 2, pp. 150-163, 2009.

[16] Bond, J. and Khraishi, T., 'Transient Non-linear Simulation with Component Mode Synthesis', *International Journal of Mechanics and Materials in Design*, In Press, 2009.

[17] Petyt, M., *Introduction to Finite Element Vibration Analysis*, Cambridge: Cambridge University Press, 1990.

[18] Bathe, K. and Wilson, E., *Numerical Methods in Finite Element Analysis*, Englewood Cliffs, New Jersey: Prentice-Hall, 1976.

[19] Hughes, T., *The Finite Element Method*, Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[20] Carr, J., 'The Effect of Shear Flexibility and Rotatory Inertia on the Natural Frequencies of Uniform Beams', *Aeronautical Quarterly*, Vol. 21, pp. 79-91, 1970.

[21] Kopal, Z., *Numerical Analysis*, London: Chapman and Hall, 1961.

[22] Barlow, J., 'Optimal Stress Locations in Finite Element Models', *International Journal for Numerical Methods in Engineering*, Vol. 10, No. 2, pp. 243-251, 1976.

[23] Cheung, Y. and Wanji, C., 'Isoparametric Hybrid Hexahedral Elements for Three Dimensional Stress Analysis', *International Journal for Numerical Methods in Engineering*, Vol. 26, No. 3, pp. 677-693, 1988.

[24] Pian, T. and Sumihara, K., 'Rational Approach for Assumed Stress Finite Elements', *International Journal for Numerical Methods in Engineering*, Vol. 20, No. 9, pp. 1685-1695, 1984.

[25] Irons, B., 'Quadrature Rules for Brick Based Finite Elements', *International Journal for Numerical Methods in Engineering*, Vol. 3, No. 2, pp. 293-294, 1971.

[26] MacNeal, R., *Finite Elements: Their Design and Performance*, New York, New York: Marcel Dekker, 1994.

[27] Ibrahimbegovic, A. and Wilson, E., 'A Modified Method of Incompatible Modes', *Communications in Applied Numerical Methods*, Vol. 7, No. 3, pp. 187-194, 1991.

[28] Taylor, R., Beresford, P., and Wilson, E., 'A Non-Conforming Element for Stress Analysis', *International Journal for Numerical Methods in Engineering*, Vol. 10, No. 6, pp. 1211-1219, 1976.

[29] Simo, J. and Rifai, M., 'A Class of Mixed Assumed Strain Methods and the Method of Incompatible Modes', *International Journal for Numerical Methods in Engineering*, Vol. 29, No. 8, pp. 1595-1638, 1990.

[30] Simo, J. and Armero, F., 'Geometrically Non-linear Enhanced Strain Mixed Methods and the Method of Incompatible Modes', *International Journal for Numerical Methods in Engineering*, Vol. 33, No. 7, pp. 1413-1449, 1992.

[31] Constantinides, A. and Mostoufi, N., *Numerical Methods for Chemical Engineers with MATLAB Applications*, Upper Saddle River, New Jersey: Prentice-Hall, 1999.

[32] Gerald, C. and Wheatley, P., *Applied Numerical Analysis*, Third Edition, Reading, Massachusetts: Addison-Wesley, 1984.

[33] Haggerty, G., *Elementary Numerical Analysis with Programming*, Boston, Massachusetts: Allyn and Bacon, 1972.

[34] Newmark, N., 'A Method of Computation for Structural Dynamics', *Journal of Engineering Mechanics*, ASCE, Vol. 85, pp. 67-94, 1959.

[35] Jacob, B. and Ebecken, N., 'An Optimized Implementation of the Newmark/Newton-Raphson Algorithm for the Time Integration of Non-linear Problems', *Communications in Numerical Methods in Engineering*, Vol. 10, No. 12, pp. 983-992, 1994.

[36] Rodrigues, M., Correa, F., and Jacob, B., 'Implicit Domain Decomposition Methods for Coupled Analysis of Offshore Platforms', *Communications in Numerical Methods in Engineering*, Vol. 23, No. 6, pp. 599-621, 2007.

[37] Craig, R. Jr., 'Coupling of Substructures for Dynamic Analyses: An Overview', AIAA Dynamic Specialists Conference, Atlanta, AIAA Paper No. 2000-1573, 2000.

[38] Hurty, W., Collins, J., and Hart, G., 'Dynamic Analysis of Large Structures by Modal Synthesis Techniques', *Computers & Structures*, Vol. 1, No. 4, pp. 535-563, 1971.

[39] Hurty, W., 'Dynamic Analysis of Structural Systems Using Component Modes', *American Institute of Aeronautics and Aeronautics Journal*, Vol. 3, No. 4, pp. 678-685, 1965.

[40] Hurty, W., 'Dynamic Analysis of Structural Systems by Component Mode Synthesis', *NASA Jet Propulsion Laboratory*, Technical Report No. 32-530, 1964.

[41] Bamford, R., 'A Modal Combination Program for Dynamic Analysis of Structures', *NASA Jet Propulsion Laboratory*, Technical Report No. 33-290, 1966.

[42] Cromer, J., Lalanne, M., Bonnecase, D., and Gaudriot, L., 'A Building Block Approach to the Dynamic Behavior of Complex Structures using Experimental and Analytical Modal Modeling Techniques', *Shock and Vibration Bulletin*, Vol. 48, pp. 77-91, 1978.

[43] Ewins, D., *Modal Testing: Theory and Practice*, Letchworth: Research Studies Press, 1984.

[44] MacNeal, R., 'A Hybrid Method of Component Mode Synthesis', *Computers & Structures*, Vol. 1, No. 4, pp. 581-601, 1971.

[45] Rubin, S., 'Improved Component-mode Representation for Structural Dynamic Analysis', *American Institute of Aeronautics and Aeronautics Journal*, Vol. 13, No. 8, pp. 995-1006, 1975.

[46] Rixen, D., 'A Dual Craig-Bampton Method for Dynamic Substructuring', *Journal of Computational and applied Mathematics*, Vol. 168, No. 1, pp. 383-391, 2004.

[47] Martinez, D. and Gregory, D., 'A Comparison of Free Component Mode Synthesis Techniques using MSC/NASTRAN', *Sandia National Laboratories*, SAND83-0025, 1984.

[48] Craig, R. Jr. and Bampton, M., 'Coupling of Substructures for Dynamic Analysis', *American Institute of Aeronautics and Aeronautics Journal*, Vol. 6, No. 7, pp. 1313-1319, 1968.

[49] Benfield, W. and Hruda, R., 'Vibration Analysis of Structures by Component Mode Substitution', *American Institute of Aeronautics and Aeronautics Journal*, Vol. 9, No. 7, pp. 1255-1261, 1971.

[50] Guyan, R., 'Reduction of Stiffness and Mass Matrices', *American Institute of Aeronautics and Aeronautics Journal*, Vol. 3, No. 2, p. 380, 1965.

[51] Wilson, E., 'Evaluation of Orthogonal Damping Matrices', *International Journal for Numerical Methods in Engineering*, Vol. 4, No. 1, pp. 5-10, 1972.

[52] Chu, C. and Milman, M., 'Eigenvalue Error Analysis of Viscously Damped Structures Using a Ritz Reduction Method', *American Institute of Aeronautics and Aeronautics Journal*, Vol. 30, No. 12, pp. 2935-2944, 1992.

[53] Craig, R. Jr. and Ni, Z., 'Component Mode Synthesis for Modal Order Reduction of Nonclassically Damped Systems', *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 12, No. 4, pp. 577-584, 1989.

[54] Liu, M. and Zheng, G., 'Improved Component-Mode Synthesis for Nonclassically Damped Systems', *American Institute of Aeronautics and Aeronautics Journal*, Vol. 46, No. 5, pp. 1160-1168, 2008.

[55] Thompson, W., *Theory of Vibration with Applications*, Englewood Cliffs, New Jersey: Prentice-Hall, 1972.

[56] Clough, R. and Penzien, J., *Dynamics of Structures*, New York, New York: McGraw-Hill, 1975.

[57] Craig, R. Jr., *Structural Dynamics: An Introduction to Computer Methods*, New York, New York: John Wiley & Sons, 1981.

[58] Simo, J. and Hughes, T., *Computational Inelasticity*, New York, New York: Springer, 1998.

[59] Tresca, H., 'Sur l'écoulement des Corps Solides Soumis á de Fortes Pressions', *Comptes Rendus de l'Académie des Sciences*, Vol. 59, p. 754, 1864.

[60] Huber, M., *Mechanik Czasopismo Techniczne*, Lemberg, Austria, Vol. 22, p. 181, 1904.

[61] von Mises, R., 'Mechanik der Festen Körper im Plastisch Deformablen Zustand', *Nachr. Ges. Wiss. Göttingen*, Vol. 1, pp. 582-592, 1913.

[62] Zienkiewicz, O., *The Finite Element Method*, New York, New York: McGraw-Hill, 1977.

[63] Gallagher, R., *Finite Element Analysis: Fundamentals*, Englewood Cliffs, New Jersey: Prentice-Hall, 1975.

[64] Owen, D. and Hinton, E., *Finite Elements in Plasticity: Theory and Practice*, Swansea: Pineridge Press Limited, 1980.

[65] Khan, A. and Huang, S., *Continuum Theory of Plasticity*, New York, New York: John Wiley & Sons, 1995.

[66] Hill, R., *The Mathematical Theory of Plasticity*, Oxford: Oxford University Press, 1950.

[67] Wu, R. and Witmer, E., 'Finite-Element Analysis of Large Elastic-Plastic Deformations of Simple Structures', *American Institute of Aeronautics and Aeronautics Journal*, Vol. 9, No. 9, pp. 1719-1724, 1971.

[68] White, F. and Drucker, D., 'Effective Stress and Effective Strain in Relation to Stress Theories of Plasticity', *Journal of Applied Physics*, Vol. 21, No. 10, pp. 1013-1021, 1950.

[69] Bland, D., 'The Associated Flow Rule of Plasticity', *Journal of the Mechanics and Physics of Solids*, Vol. 6, No. 1, pp. 71-78, 1957.

[70] Cornwell, R., Craig, R. Jr., and Johnson, C., 'On the Application of the Mode-acceleration Method to Structural Engineering Problems', *Earthquake Engineering and Structural Dynamics*, Vol. 11, No. 5, pp. 679-688, 1983.

[71] Tinker, M., 'Hybrid Residual Flexibility/Mass-additive Method for Structural Dynamic Testing', *NASA Marshall Space Flight Center*, TM-2003-212343, 2003.

[72] Kammer, D. and Baker, M., 'A Comparison of the Craig-Bampton and Residual Flexibility Methods for Component Substructure Representation', *Structural Dynamics Research Corporation*, 85-0817, 1985.

[73] Fausett, L., *Numerical Methods: Algorithms and Applications*, Upper Saddle River, New Jersey: Prentice Hall, 2003.

[74] Liu, S. and Lin, T., 'Elastic-plastic Dynamic Analysis of Axisymmetric Solid', *Earthquake Engineering and Structural Dynamics*, Vol. 7, No. 2, pp. 147-159, 1979.

[75] Nagarajan, S. and Popov, E., 'Elastic-plastic Dynamic Analysis of Axisymmetric Solid', *Computers and Structures*, Vol. 4, No. 6, pp. 1117-1134, 1974.

[76] Baron, M., Bleich, H., and Weidlinger, P., 'Dynamic Elastic-Plastic Analysis of Structures', *Journal of the Engineering Mechanics Division*, Proceedings of the American Society of Civil Engineers, EM 1, pp. 23-42, 1961.

[77] Jones, N., *Structural Impact*, Cambridge: Cambridge University Press, 1997.

[78] Bhardwaj, M. and Walsh, T., 'Salinas: Theory Manual', *Sandia National Laboratories*, SAND2004-4438W, 2004.