INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of "sectioning" the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.

University Microfilms International

300 N. Zeeb Road
Ann Arbor, MI 48106

8402240

Benson, David Joel

THE SIMULATION OF DEFORMABLE MECHANICAL SYSTEMS USING VECTOR PROCESSORS

The University of Michigan PH.D. 1983

University
Microfilms
International 300 N. Zeeb Road, Ann Arbor, MI 48106

Copyright 1983

by

Benson, David Joel

All Rights Reserved

THE SIMULATION OF DEFORMABLE MECHANICAL SYSTEMS USING VECTOR PROCESSORS

by
David Joel Benson

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
1983

Doctoral Committee:

Associate Professor Noboru Kikuchi, Co-Chairman
Professor Milton Chace, Co-Chairman
Professor Donald Calahan
Professor Donald Greenwood
Assistant Professor Joseph Whitesell

RULES REGARDING THE USE OF

MICROFILMED DISSERTATIONS

Microfilmed or bound copies of doctoral dissertations submitted to The University of Michigan and made available through University Microfilms International or The University of Michigan are open for inspection, but they are to be used only with due regard for the rights of the author. Extensive copying of the dissertation or publication of material in excess of standard copyright limits, whether or not the dissertation has been copyrighted, must have been approved by the author as well as by the Dean of the Graduate School. Proper credit must be given to the author if any material from the dissertation is used in subsequent written or published work.

For my parents

Dorothy A. Benson
Jack J. Benson

ACKNOWLEDGMENTS

I would like to extend my thanks to my dissertation committee, especially to Associate Professor Noboru Kikuchi, not only for his technical expertise, but also for his editorial assistance and encouragement. Special thanks are also due Professor Donald Calahan for his contributions on vector processing to this dissertation. The author is also grateful to Professor Milton Chace, Professor Donald Greenwood, and Assistant Professor Joseph Whitesell for their participation on the committee.

My thanks also to Dr. Linda Petzold of Sandia National Laboratories and Dr. John O. Hallquist of Lawrence Livermore National Laboratory for taking time out of their busy schedules to discuss their work and for sending copies of the papers and programs. Any misrepresentation of their work is strictly the responsibility of the author.

The discussions about computational dynamics I have had with John Angell over the years were invaluable, and the financial support I received from Mechanical Dynamics, Inc. was very generous.

Exxon, DuPont, and Rockwell International also generously supported my research through departmental fellowships.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGMENTS iii

LIST OF TABLES vii

LIST OF FIGURES viii

CHAPTER

1. INTRODUCTION 1

    1.1 Scope of the Research 1
    1.2 A Historical Review of the Previous Research 2
    1.3 Organization of the Dissertation . . 4
    1.4 Mathematical Notation 5

2. SCALAR AND VECTOR PROCESSORS 7

    2.1 Introduction 7
    2.2 Scalar Processors 8
    2.3 Vector Processors 9
    2.4 Vectorizable and Nonvectorizable Operations 11
    2.5 Modelling processors 14
    2.6 Summary 18

3. NUMERICAL METHODS 19

    3.1 Introduction 19
    3.2 Numerical Integration Methods 19
        3.2.1 Stability
        3.2.2 Choosing an Integration Method
        3.2.3 The Gear Integration Method
    3.3 Linear Equation Solvers 28
        3.3.1 LU Decomposition
        3.3.2 Randomly Sparse Equation Solvers
        3.3.3 Banded Equation Solvers
        3.3.4 Block-Oriented Solvers
    3.4 Summary 33

4. THE DYNAMICS OF DEFORMABLE MECHANICAL SYSTEMS 35

4.1 Introduction 35
4.2 Desirable Characteristics 35
4.3 The Finite Element Method 36
4.3.1 The General Approach
4.3.2 The Deformation Field From the Lagrangian Viewpoint
4.3.3 The Stiffness Matrix for Linear Elasticity
4.3.4 The Mass Matrix
4.4 Linear Coordinate Reduction 49
4.4.1 Modal Analysis
4.4.2 Component Mode Synthesis
4.5 A Technical Review of the Previous Research 56
4.5.1 Total Displacement at the Nodes
4.5.2 Joint Coordinates for Rigid Body Motion
4.5.3 Rigid Body Displacement Coordinates
4.5.4 Quasi-coordinates
4.5.5 Local Coordinate Reference Frames
4.6 The Dynamics of Deformable Mechanical Systems 67
4.6.1 The Weak Form of Lagrange's Equation
4.6.2 The Choice of Generalized Coordinates
4.6.3 Lagrange's Equation for Quasi-coordinates
4.6.4 The Application of Quasi-coordinates
4.7 Specialization to Linear Elasticity . 88
4.8 Rigid Body Dynamics 91
4.9 Forces and Constraints 93
4.9.1 Notation and Common Expressions
4.9.2 Forces
4.9.3 Joints
4.10 Initial Conditions and Static Equilibrium
4.11 Summary 108

5. COMPUTER IMPLEMENTATION 110

5.1 Program Capabilities and Compromises 110
5.2 The Choice of Rigid Body Rotation Coordinates 112
5.3 The General Approach to Vectorization 114

5.4 The Vectorization of the Part Equations 115
5.5 The Vectorization of the Joints and the Forces 118

6. NUMERICAL EXAMPLES 121

6.1 Rigid Body Dyanmics 121
6.2 A Finite Element with Selective Reduced Integration 131
6.3 The Vibration of a Cantilevered Beam 138
6.3.1 The Natural Frequencies
6.3.2 Critical Excitation of a Bending Mode
6.3.3 Computational Cost as a Function of the Beam Model
6.4 Analysis of a Flexible Slider-Crank Mechanism 148
6.4.1 Problem Description
6.4.2 Comparing the Solutions
6.4.3 Computational Efficiency
6.5 Summary 152

7. CONCLUSIONS 155

7.1 A Summary of the Research 155
7.2 Conclusions 156
7.3 Future Research and Development . . . 157

REFERENCES 159

LIST OF TABLES

Table

3.1 FORTRAN Sparse Matrix Solution Performance 31

3.2 Block-Oriented Solution Performance 34

6.1 Inertial Properties for the Suspension System 123

6.2 Nodal Coordinates for the Suspension System 124

6.3 The Direction Cosines of the Revolute Rotation Axes for the Suspension System . . 126

6.4 The Computational Performance of DAVES and ADAMS for a Suspension System 127

6.5 Comparison of Natural Frequencies 140

6.6 Computational Cost as a Function of the Beam Model 145

LIST OF FIGURES

Figure

3.1 The Execution Rates of Two Unsymmetric Banded Equation Solvers 32

4.1 The Definition of the Generalized Coordinates 74

6.1 Schematic of the Automobile Suspension System 125

6.2 Displacement of the Lower Arm in the Z Direction 127

6.3 Velocity of the Lower Arm in the Z Direction 128

6.4 Acceleration of the Lower Arm in the Z Direction 128

6.5 Theta, Measured in Degrees, of the Lower Arm 129

6.6 Angular Velocity of the Lower Arm About the Y Axis 129

6.7 Angular Acceleration of the Lower Arm About the Y Axis 130

6.8 Eight Node Hexahedral Element 132

6.9 First Bending Mode 140

6.10 Second Bending Mode 141

6.11 Third Bending Mode 141

6.12 The Deflection of the Tip of the Beam . . . 143

6.13 The Velocity of the Tip of the Beam 143

6.14 The Acceleration of the Tip of the Beam . . 144

6.15 The Deflection of Three Beam Models at the Tip 146

6.16 The Velocity of Three Beam Models at the Tip 147

6.17 The Acceleration of Three Beam Models at the Tip 147

6.18 Two Dimensional Slider-Crank Mechanism . . 150

6.19 Three Dimesional Slider-Crank Model for DAVES 150

6.20 Midpoint Connecting Rod Deflection from Song and Haug 153

6.21 Midpoint Connecting Rod Deflection from DAVES 153

CHAPTER 1

INTRODUCTION

1.1 Scope of the Research

This dissertation presents a formulation for simulating systems of deformable bodies undergoing large, nonlinear motion using vector processors. For algorithms tailored to them, vector processors (Class VI computers), such as the Cray-1 and Cyber-205, offer better computational rates than scalar processors (Class IV computers).

The equations of motion are derived by using Lagrange's equation. By introducing an appropriate set of quasi-coordinates and the canonical momenta as variables, arbitrary mechanical systems are represented by a mixture of simple algebraic and first order differential equations. The general formulation is specialized and simplified by representing linearly elastic bodies as truncated series of their vibrational modes.

A generalized program, named DAVES (Dynamic Analysis with Vectorization of Elastic Systems), based on the formulation for linearly elastic and rigid bodies is documented. The program is self-formulating in the sense that the equations are automatically generated from a

description of the bodies, joints, and applied forces in the system. Numerical solutions are obtained by using Gear's stiff integration method with an explicit Jacobian.

The theoretical part of this dissertation was originally presented at the ASME Second International Computer Engineering Conference in 1982 in San Diego, California. The paper is Reference [1].

1.2 A Historical Review of the Previous Research

There is a large body of literature available on kineto-elastodynamics, the study of the motion of flexible mechanisms. The aim of this section is not to give a comprehensive review of the literature, but to discuss briefly the general trends in the area and to mention the most prominent researchers of the past decade. For a general review of the literature prior to 1970, Reference [2] is a good place to start.

Researchers originally assumed that the linear deflections of the flexible bodies could be linearly superposed on the nonlinear rigid body motion. Winfry [3], Erdman et. al. [4], and Bahgat and Willmert [5], among others, used this simple approach.

The decoupling of the equations resulting from the superpostion principle ignores effects that can be important [6]. Recent research includes all of the coupling effects. Likins analyzed tree-structured, flexible systems in 1974 [7]. Boland et al. analyzed closed loop systems in

1975 [8]. De Veubeke, in 1976, proposed the use of quasi-coordinates and a mean axis coordinate system in an important theoretical paper [9].

A two-dimensional linear beam element using center of mass coordinates and elastic deflections for generalized coordinates was introduced by Song and Haug in 1979 [6]. Their generalized, self-formulating program is an extension of the techniques pioneered in the rigid body dynamics program ADAMS (Automatic Dynamic Analysis of Mechanical Systems) [10], written by Orlandea in 1971. A three-dimensional, self-formulating program [11] by Sunada and Dubowsky was released in 1980. Like IMP (Integrated Mechanisms Program) [12], a rigid body dynamics program, it uses four-by-four matrix methods with joint coordinates to describe the rigid body motion. Component mode synthesis [13] is used to represent the flexibility of the bodies. In 1983, Shabana and Wehage extended the coordinate partitioning method for rigid body dynamics, developed by Wehage and Haug [14], for flexible systems [15].

"Flexible" is used in the previous paragraphs to mean linearly elastic. Constraints are usually limited to lower pair contacts such as spherical and revolute joints. The general computational approach extends rigid body dynamics technology to flexible systems by incorporating finite element techniques. Finite element researchers, however, are extending their techniques to include nonlinear rigid body motion. Research has centered on simulating the

high speed impact phenomena associated with a projectile hitting armor. The projectile undergoes both nonlinear rigid body motion and large, nonlinear plastic deformation. The higher pair contact between the deforming projectile and the armor is modelled including the effects of friction. Unlike formulations for linearly elastic systems, the finite strain formulations do not decompose the motion into separate rigid body and body deformation coordinates. Recent, highly vectorized programs such as DYNA3D [16] and NIKE2D [17], developed at Lawerence Livermore under Hallquist, can, in principle, simulate mechanisms but because they assume there is nonlinearity in the rigid body motion, the deformation of the bodies, and the constitutive relations, they are prohibitively expensive for most engineering applications. Such programs currently are, however, the only way to analyze the catastrophic failure of many dynamic systems, e.g., car crashes.

1.3 Organization of the Dissertation

The dissertation is divided into seven chapters.

Chapter Two is a short review of how computers work, with an emphasis on the strengths and weaknesses of vector processors.

Chapter Three presents an overview of the currently popular numerical methods for integrating differential equations and solving linear algebraic equations. The

advantages and disadvantages of the Gear integration algorithm and sparse matrix solution methods, both of which are used in this dissertation, are examined.

In Chapter Four, the formulation for analyzing deformable mechanical systems is presented. A review of the difficulties associated with previous formulations is given. The general formulation is valid for systems with nonlinear deformations of the bodies and nonlinear constitutive relations. By introducing linear coordinate reduction methods, the general formulation is simplified and specialized to linearly elastic systems. The equations for describing the joint constraints and applied forces, including gravity, are derived.

The computer implementation of the formulation is discussed in Chapter Five with the emphasis on vectorization.

In Chapter Six, example problems are solved and compared to solutions available in the literature.

Chapter Seven discusses the results of the research, presents conclusions, and proposes further research.

1.4 Mathematical Notation

Index notation is used throughout this dissertation except for the sections that discuss partitioned matrices. The summation convention is also used unless otherwise

specified. For example, $\sigma_{ij}\delta e_{ij}$ is the same as $\sum_{i,j} \sigma_{ij}\delta e_{ij}$. All summations are over the entire range of the index variables.

Covariant differentiation with respect to the material coordinates is denoted by a comma followed by a subscript, e.g., $u_{i,j}$ is $\frac{\partial u_i}{\partial \xi_j}$.

Following the common convention, differentiation with respect to time is denoted by a dot over the variable, e.g., $\dot{x}_i$ is $\frac{dx_i}{dt}$. Similarly, two dots represent the second derivative: $\ddot{x}_i$ is $\frac{d^2x_i}{dt^2}$.

Both the Kronecker delta and the cyclic permutation operator are used.

$$\delta_{ij} = \begin{bmatrix} 1 & i=j \\ 0 & i\neq j \end{bmatrix} \tag{1.4.1}$$

$$e_{ijk} = \begin{bmatrix} 1 & (i,j,k) = (1,2,3),(2,3,1),(3,1,2) \\ 0 & i=j \text{ or } j=k \text{ or } i=k \\ -1 & (i,j,k) = (1,3,2),(3,2,1),(2,1,3) \end{bmatrix} \tag{1.4.2}$$

where

δ_{ij} is the Kronecker delta.

e_{ijk} is the cyclic permutation operator.

CHAPTER 2

SCALAR AND VECTOR PROCESSORS

2.1 Introduction

Vector processors (Class VI computers), such as the Cray-1 and Cyber-205, offer better computational rates than scalar processors (Class IV computers) for algorithms tailored to them. A basic understanding of the architecture of vector processors is necessary for using them effectively. Since a scalar processor, the traditional computer architecture, is conceptually simpler than a vector processor, a brief description of a scalar processor is included first in this chapter.

The descriptions in this chapter are for highly simplified generic scalar and vector processors. Analogies are made to facilitate the reader's understanding, and they should not be extended beyond the context in which they are used.

One analogy between computers and manufacturing plants provides a very quick and intuitive understanding of the differences between vector and scalar processors: Vector processors are faster than scalar processors for much the same reason that factories are faster than cottage industries. Under some circumstances, however, scalar

processors and cottage industries are faster. After examining the architecture of the processors, several measures of computational performance are presented.

2.2 Scalar Processors

Only those parts of a computer associated with performing floating point calculations, namely the arithmetic-logic unit (ALU), the registers, and the memory, are discussed. The ALU is the part of the computer that actually performs the basic arithmetic and logical operations. The memory stores the numbers for the computer. Each storage location is labelled with an integer number called its "address". Large scientific computers can store several million numbers in their memory. Registers are special high speed storage locations that are used to move numbers between the ALU and memory. The following are analogies between a scalar processor and a cottage industry:

Scalar Processor	Cottage Industry
Scalar Arithmetic Logic Unit	Craftsman
Memory	Warehouse
Arithmetic Operation	Manufacturing Operation
Numbers	Parts and Assemblies
Scalar Register	Storage for One Part
One Clock	One Minute

A clock is not a standard unit of time; it varies in duration from one computer to another.

During the execution of a program, numbers flow from memory to the registers. The ALU operates on numbers from the registers and stores the results in other registers. The numbers are either operated on again or stored back in memory. The important things to note, from the standpoint of computational efficiency, are that two operations take twice as long as one, and that the amount of time to perform a set of operations is independent of the order in which they are performed. Like most generalizations, these have exceptions: additions (or subtractions) and multiplications are performed concurrently in some real scalar computers because the operations are performed in independent sections of the ALU.

2.3 Vector Processors

The analogies between vector processors and factories are:

Vector Processor . . . Factory

Vector ALU . . . . . Assembly Line

Vector Register . . . Pallet of Ordered Parts

Vector Length . . . . Number of Parts on a Pallet

In a manner similar to scalar processors, numbers flow from memory to the vector registers, but vector registers contain ordered blocks of numbers instead of single numbers. Vector ALUs operate on the numbers in the vector registers and store the results back in the vector registers. As with scalar processors, the registers are operated on again or the results are stored back in memory.

All of the elements in a vector are always operated on in an identical manner and in the order that they are stored in the vector. Here is one example of where the analogies can be overextended: assembly lines can operate selectively, e.g., different types of radios are installed in cars on the same assembly line based on the customers' orders.

Vector processors and factories operate faster than scalar processors and cottage industries not because individual operations take less time -- they do not -- but by scheduling a continual flow of identical operations, broken down into suboperations, so that no section of the ALU, or assembly line, is ever idle. Just as with a factory, there is a startup time with a vector ALU. For short vectors, a vector ALU is often slower than a scalar ALU because of the frequent restarting of the vector ALU. The asymptotic rate for a vector ALU is one operation per clock, a rate far higher than for a scalar ALU.

Data must flow to and from memory in a fast and orderly manner if the ALU is not to be starved for operands or clogged with results, both of which are conditions that stop an ALU. Scheduling the movement of numbers is therefore one of the most important considerations in designing vector algorithms. In order to fetch and store numbers quickly from memory, their addresses must be easily calculated, implying that the addresses are consecutive or linearly indexed. Filling or emptying vector registers with randomly stored numbers in memory, called "gathering" and

"scattering" respectively, is so time consuming that sometimes the operations are more efficiently performed in scalar mode.

As mentioned previously, some computers are able to perform several independent operations simultaneously. When the operations are truly independent the term "parallel processing" is applied. For the example of the adder and multiplier operating independently the computational rate is increased by a factor of two. A similar idea, called "chaining", uses the result of one functional unit as an operand for another functional unit. Again, the computational rate increases by a factor of two. Most scientific processors are capable of parallel processing and a few perform chaining. The demand on the memory path is lower for chaining than parallel processing. For two functional units the memory must transfer numbers to and from six vector registers for parallel processing (four operand and two result registers), and four registers for chaining (three operands and one result register).

2.4 Vectorizable and Nonvectorizable Operations

Some concrete examples of vectorizable and nonvectorizable (scalar) operations will clarify the concepts discussed in the previous sections.

The classical vector operation is:

```
      DO 1 I=I1,I2
    1 A(I)=B(I)+C(I)
```

Any of the other three basic arithmetic operations (-,/,*) could substitute for the addition operation. Because the expression has two operands, it is a dyadic operation. The terminology is commonly used through pentadic (operations on five vectors).

The following illustrates a loop where chaining is possible. Whether or not the loop is actually chained depends on both the processor and the compiler.

```
      DO 1 I=I1,I2
    1 A(I)=B(I)+C(I)*D(I)
```

When two or more loops are nested only the inner loop is vectorized. The Texas Instruments ASC is an exception for some operations such as matrix multiplications.

```
      DO 2 J=J1,J2
      DO 1 I=I1,I2
    1 A(I,J)=B(I)*C(J)
    2 CONTINUE
```

The following loop is not vectorizable because it contains inner loop logic. Linear indexing through arrays in a loop is not a sufficient condition for vectorization. The inner loop logic violates the rule that all of the elements in a vector must have the same operations performed on them: in the example some of the elements are added, while others are multiplied together. In terms of the factory analogy, airplanes and cars can not be built on the same assembly line.

```
      DO 2 I=I1,I2
      IF (B(I).LT.0.) GO TO 1
      A(I)=B(I)+C(I)
      GO TO 2
    1 A(I)=B(I)*C(I)
    2 CONTINUE
```

Recursive operations, such as the following loop, are not vectorizable because the needed result from a previous operation may not be available. This is the equivalent of trying to jump start a car on the assembly line with another car that has yet to have its battery installed. I3 is assumed positive.

```
      DO 1 I=I1,I2
    1 A(I)=A(I-I3)+B(I)
```

While the following loop is superficially recursive, it does not rely on previous results and is vectorizable. Again, I3 is positive.

```
      DO 1 I=I1,I2
    1 A(I)=A(I+I3)+B(I)
```

The following loops illustrate gather and scatter. Neither is vectorizable.

```
      Gather:
      DO 1 I=I1,I2
    1 A(I)=B(INDEX(I))
      Scatter:
      DO 1 I=I1,I2
    1 A(INDEX(I))=B(I)
```

Gather is performed in finite element programs to get the nodal data necessary to calculate an element stiffness submatrix. Scatter is performed when the global stiffness matrix is assembled from the element submatrices. Despite the unavoidable gathers and scatters, finite element programs benefit from vectorization because many vectorizable operations are performed on the relatively small amount of gathered data when the stiffness submatrices are generated.

Gathering, scattering, and inner loop logic are usually eliminated or reduced by carefully sorting and structuring the data in a preprocessing phase of the program. In finite element programs, for instance, all of the shell element submatrices are evaluated together after sorting all of the elements in a structure by type. For large problems the shell data are sorted by material type to minimize the gather operations for material data.

2.5 Modelling processors

This section borrows heavily from the notes of D. A. Calahan [18].

The total time to perform a vector operation, using the simple model presented above, is given by equation 2.5.1.

$$T_{v_i} = T_{s_i} + T_o L_i \qquad (2.5.1)$$

where

T_{v_i} is the total operation time.

T_{s_i} is the startup time.

T_o is the clock period.

L_i is the vector length.

If L_i is larger than the length of a vector register in a real machine then equation 2.5.1 is obviously not valid, but for simplicity in this analysis an infinite vector length is assumed possible.

For a sequence of m vector operations, the total time is the sum of the individual times.

$$T_v = \sum_{i=1}^{m} T_{s_i} + T_o \sum_{i=1}^{m} L_i \qquad (2.5.2)$$

The average startup time and average vector length are defined as:

$$T_{s_{ave}} = \frac{1}{m} \sum_{i=1}^{m} T_{s_i} \qquad (2.5.2)$$

$$L_{ave} = \frac{1}{m} \sum_{i=1}^{m} L_i \qquad (2.5.3)$$

where

$T_{s_{ave}}$ is the average startup time.

L_{ave} is the average vector length.

The average startup time characterizes the processor and the average vector length characterizes the algorithm.

Computational efficiency is the ratio of the actual operation time and the total time.

$$\eta = \frac{\text{operation time}}{\text{startup time + operation time}}$$

$$= \frac{1}{1 + \frac{T_{s_{ave}}}{T_o}\ \frac{1}{L_{ave}}} \qquad (2.5.4)$$

where

η is the computational efficiency.

The floating point operations per second (FLOPS, or MFLOPS for one million FLOPS) measures the computational rate. For a calculation where the number of operations (FLOP) is proportional to the vector length the computational rate is given by equation 2.5.5.

$$\text{FLOPS} = \text{FLOP} / \text{time}$$

$$= \frac{K \sum_{i=1}^{m} L_i}{\sum_{i=1}^{m} T_{s_i} + T_o \sum_{i=1}^{m} L_i} \qquad (2.5.5)$$

$$= K\eta/T_o$$

where

K is the number of operations per vector length.

From equation 2.5.5, several important conclusions are apparent:

1) All other things being equal, an algorithm with long vectors is faster than one with short vectors.
2) Machines with long startup times perform poorly with short vectors.
3) For long vector lengths, the operations per vector length or "concurrency" and the clock rate determine the computational rate. Concurrency is the number of operations being performed simultaneously through parallel processing and chaining. A bad start in a long race doesn't hurt if you're fast enough.

For optimal scalar and vector codes with infinite vector lengths the greatest possible improvement that the vector code can show over the scalar code is the average number of clocks a single scalar operation takes. This simple result stems from the fact that vector processors asymptotically produce one operation per clock. For the Cray-1 the average number of clocks per scalar operation is somewhere between five and six. A comparison of two carefully assembly-coded equations solvers shows that the vectorized solver runs at a maximum of 115 MFLOPS [19] while the scalar solver runs at 17 MFLOPS [20], giving a ratio of 6.76. Reality exceeds expectation in this case because the analysis was very crude. Most programs are written in a high level language, such as FORTRAN, so that a more typical gain is the factor of four reported by Hallquist for

DYNA3D [21]. Most people are disappointed with their first experiences with vectorization. If only one half of the computation is vectorized, then the overall gain is at most two (even if the vectorized code takes literally no time).

2.6 Summary

Vector processors, like factories, are very efficient for performing repetitive operations on large sets of numbers because of the economies of scale. They are inefficient on small problems because they always set up for large operations, and the large overhead associated with the setup dominates the computation. Conversely, scalar processors, like cottage industries, never tool up for large, repetitive calculations and can not take advantage of the economies of scale. Since their startup time is low, they perform very well on small or unstructured problems.

CHAPTER 3

NUMERICAL METHODS

3.1 Introduction

An understanding of the behavior of different numerical methods is important so that for a particular problem the most productive methods can be chosen. The performance of numerical methods is characterized by their accuracy, stability, and efficiency. Elementary physics and stability considerations are used in this chapter to choose a numerical integration method suitable for simulating deformable mechanical systems. Benchmarks on the Cray-1 of different linear equation solvers are used to reach general conclusions about the desirable structure of the equations of motion.

3.2 Numerical Integration Methods

The dynamics of mechanical systems are described by differential equations that must be integrated with respect to time. A variety of numerical methods exist for solving them [22]. Enough is known about the particular differential equations of flexible mechanisms to narrow the choice of methods down to a general class. The choice of a

particular method is based on its overall convenience. A brief discussion of stability is necessary for understanding the selection criterion.

3.2.1 Stability

Analyzing the stability of an integration method with respect to an arbitrary and unknown set of differential equations is impossible. Mathematicians, therefore, introduce a model equation which is simple enough for analysis but which has enough of the properties of more general systems to be useful. In a neighborhood of a given point an arbitrary system of differential equations is linearized into a set of equations with constant coefficients. Furthermore, the linearized set is rewritten as a set of first order, linear equations. A linear transformation of variables, using the eigenvectors of the equations, decouples the equations into a set of independent first order, linear differential equations.

$$\dot{\underline{y}} - [\lambda]\underline{y} = 0 \qquad (3.2.1.1)$$

where

$\underline{y}$ is the vector of independent variables.

$[\lambda]$ is the diagonal constant coefficient matrix.

The stability of a method is analyzed by using equation 3.2.1.1 as the model equation. Only a single equation is analyzed because the equations are decoupled.

For convenience the initial value of y is one, and the initial time is zero. The model problem is restated in equation 3.2.1.2.

$$\dot{y} - \lambda y = 0 \qquad y(0) = 1 \qquad (3.2.1.2)$$

The exact solution is the exponential function.

$$y = e^{\lambda t} \qquad (3.2.1.3)$$

The Euler forward difference method, equation 3.2.1.4, is used to demonstrate how stability is analyzed using the model equation.

$$y_i = y_{i-1} + hf(y_{i-1}, t_{i-1}) \qquad (3.2.1.4)$$

where

y_i is the dependent variable at the ith time step.

t_i is the value of time at the i-th step.

h is $t_i - t_{i-1}$.

f is the function for $\dot{y}$.

After assuming the stepsize is constant, equation 3.2.1.5 is derived by substituting equation 3.2.1.2 into equation 3.2.1.4.

$$y_i = (1+h\lambda)^i y(0) \qquad (3.2.1.5)$$

If λ has a positive real part then the exact solution is unbounded and stability is not important, but for a negative real part the exact solution approaches zero

with time, and a stable solution must be bounded, although it is not necessarily accurate. All of the eigenvalues discussed in the following sections are assumed to belong in the negative half plane since only eigenvalues with negative real parts affect the stability analysis. This does not imply that the systems under discussion do not have eigenvalues with positive real parts -- they do -- only that the positive eigenvalues are not discussed. The term $(1 + h\lambda)$ in the above equation must have a magnitude less than or equal to one for a bounded solution, imposing equation 3.2.1.6 as a constraint on the stepsize.

$$|1 + h\lambda| \leq 1 \qquad (3.2.1.6)$$

The maximum stepsize calculated in equation 3.2.1.6 is limited by the eigenvalue farthest away from one in a system of equations. Even when the transient part of the solution has decayed to insignificance small timesteps must be taken for a stable solution. Systems of differential equations with a large ratio of the real parts of the maximum and minimum eigenvalues are called "stiff" [23].

A method that is stable for stiff equations is the Euler backward difference method.

$$y_i = y_{i-1} + hf(y_i, t_i) \qquad (3.2.1.7)$$

Equation 3.2.1.7 is implicit in y_i, and generally must be solved iteratively. The model equation is simple enough that an explicit solution is possible. Repeating the steps in the previous analysis, equation 3.2.1.8 is derived.

$$y_i = \left(\frac{1}{1 - h\lambda}\right)^i y(0) \qquad (3.2.1.8)$$

The solution is stable for all stepsizes.

3.2.2 Choosing an Integration Method

The differential equations governing mechanical systems are divided into those that concern the rigid body motion and those that concern the deformations of the bodies. A priori estimates of the eigenvalues associated with the rigid body motion are difficult to give; assume at least one eigenvalue is on the order of 1 radian/second for this section.

The deformations for a body are governed by a set of second order differential equations. Using the arguments of the previous section, they are linearized into a set of independent second order equations with constant coefficients. For linearly elastic bodies these equations are the classical set derived by modal analysis for free vibration. The canonical homogeneous form is equation 3.2.2.1.

$$\ddot{\eta}_i + 2\xi_i\omega_i\dot{\eta}_i + \omega_i^2\eta_i = 0 \qquad (3.2.2.1)$$

where there is no sum on i and

η_i is the i-th modal coordinate.

ξ_i is the i-th damping coefficient.

ω_i is the i-th undamped natural frequency.

The solution to equation 3.2.2.1 is equation 3.2.2.2 for systems that are not overdamped.

$$\eta_i = \exp(-\xi_i\omega_i t)\cdot \left(A_i \sin(\sqrt{1-\xi_i^2}\,\omega_i t + B_i \cos(\sqrt{1-\xi_i^2}\,\omega_i t)\right) \qquad (3.2.2.2)$$

where there is no sum on i and

A_i and B_i are constants determined by the initial conditions.

From equation 3.2.2.2 the time constant is $-\xi_i\omega_i$. The damping coefficient, ξ_i, is a non-negative number that usually assumes a value between zero (no damping) and one (critical damping). An estimate of the system stiffness is made by taking the ratio of the estimates of the time constants in the negative half plane associated with the deformations and rigid body motion.

$$S = \frac{\max\left|Re(\lambda_i)\right|}{\min\left|Re(\lambda_i)\right|} \approx \max\left|\xi_i\omega_i\right| \qquad (3.2.2.3)$$

where there is no sum on i and

S is the mathematical stiffness.

Large natural frequencies represent physically stiff deformation modes, therefore the implication of equation 3.2.2.3 is damped, physically stiff systems are mathematically stiff. Eliminating the damping would result in high computational costs. Any integration method would use small time steps because of the high frequency component

in the solution. Furthermore, physically stiff systems, such as cars, do not "ring" at high frequencies and the undamped solution would not be physically corrrect.

A generalized program must be able to simulate physically stiff systems if it is to have wide application. To satisfy the stability requirements of stiff differential equations, the Gear backward difference integration algorithm is used in the program described in this dissertation for simulating rigid and linearly elastic mechanical systems.

3.2.3 The Gear Integration Method

The interested reader is referred to Reference [24] for a complete description of the Gear method. One advantage of the method, besides its stiff stability, is its ability to solve simultaneous systems of differential and algebraic equations. The general problem is expressed in equation 3.2.3.1.

$$\underline{F}(\underline{y},\dot{\underline{y}},t) = 0 \qquad (3.2.3.1)$$

where

$\underline{F}$ is a vector function.

$\dot{\underline{y}}$ is the derivative of the $\underline{y}$ with time.

Substitution of the Euler backward difference formula for $\dot{\underline{y}}$ shows how this system is solved for a first order method [25].

$$\underline{F}(\underline{y}_i,\frac{\underline{y}_i - \underline{y}_{i-1}}{h},t_i) = 0 \qquad (3.2.3.2)$$

Newton-Raphson iteration is used to solve the above equation. The superscripts in the following equation are the iteration numbers.

$$\underline{y}_i^{m+1} = \underline{y}_i^m - \left(\frac{\partial \underline{F}}{\partial \underline{y}} + \frac{1}{h}\frac{\partial \underline{F}}{\partial \dot{\underline{y}}}\right)^{-1} \underline{F}\left(\underline{y}_i^m, \frac{\underline{y}_i^m - \underline{y}_{i-1}}{h}, t\right) \qquad (3.2.3.3)$$

The Jacobian in equation 3.2.3.3 is usually not updated every iteration to reduce the computational cost.

Higher order methods are derived by susbstituting higher order expressions for $\dot{\underline{y}}$ in equation 3.2.3.1.

Besides stiff stability, the advantages of the Gear method are:

1) The method is self-starting.
2) It is a variable stepsize, variable order method with algorithms for automatically choosing the optimal stepsize and order.
3) Algebraic equations are solved simultaneously with the differential equations, permitting algebraic equations to represent joint constraints.
4) The internal consistency of a program with an explicit Jacobian is readily verified by comparing the explicit Jacobian to one numerically differenced using the residuals.

The disadvantages are:

1) The Jacobian must be explicit for the efficient analysis of large systems [26].

2) Numerical difficulties may be encountered during the LU decomposition of the Jacobian for very small integration steps.

3) Newton-Raphson iteration is unstable when materials deforming plastically unload, and other methods must be used to solve the Gear corrector equation.

4) Implicit integration methods are not necessarily cost effective in comparison to explicit methods, even for stiff problems, if the Jacobian bandwidth is high. For structural dynamics problems the operation count of a function evaluation is roughly proportional to nb, where n is the number of equations and b is the bandwidth, and the operation count for factorization is proportional to nb^2. The ratio of the factorization to the function evaluation count is therefore proportional to b, implying that the implicit method must take time steps on the order of b times larger than the explicit method for it to be practical. While unconditionally stable methods can take any stepsize, accuracy requirements can restrict the stepsize to values such that brute force number crunching with an explicit method is more efficient [21].

3.3 Linear Equation Solvers

Virtually all of the formulations for simulating mechanical systems require the frequent solution of linear algebraic equations. The structure of the equations generated by a formulation determines the choice of the equation solver, and therefore, to some extent, determines the computational cost. Stability and accuracy are generally not issues for direct (i.e. non-iterative) equation solvers provided that good pivots are available. The emphasis in this section is on the computational speed of vectorized equation solvers as a function of vector length and equation structure.

3.3.1 LU Decomposition

Direct equation solvers almost invariably solve the linear equation set by LU decomposition. The details may vary, but the idea is always the same.

$$\begin{aligned} A_{ij}x_j &= b_i \\ &= L_{ij}U_{jk}x_k \end{aligned} \qquad (3.3.1.1)$$

where

A_{ij} is the coefficient matrix.

x_j is the solution vector.

b_i is the right hand side vector.

L_{ij} is a lower triangular matrix.

U_{jk} is an upper triangular matrix.

The solution is produced in two stages.

$$L_{ij}y_j = b_i$$
$$U_{jk}x_k = y_j \qquad (3.3.1.2)$$

where

y_j is a scratch vector.

For m equations roughly $m^3/3$ operations are required for the LU factorization if the coefficient matrix is full and unsymmetric, or $m^3/6$ if it is symmetric. Only m^2 operations, however, are required for the forward and backward substitution. The operation count is reduced substantially if the coefficient matrix has a large number of zeroes and the equation solver is designed to avoid operating on them. Only the nonzero elements are then stored, affording a large reduction in the memory requirements.

All of the equation solvers discussed in the following sections are the work of D. A. Calahan. For detailed descriptions of the algorithms, the reader is referred to the cited references; for a general description of LU decomposition, see any linear algebra text. Unless otherwise specified, all of the equation solvers described below are written in Cray-1 assembly language (CAL), and the computational rates are for in-core solutions.

3.3.2 Randomly Sparse Equation Solvers

Equation sets that have very few nonzero coefficients with no evident structure are called "randomly sparse," or, more simply, "sparse." ADAMS, a rigid body program, and some circuit analysis programs [27], generate hundreds to thousands of equations with an average of three to ten coefficients per equation. For m equations, the number of operations is typically proportional to $m^{1.3}$ to $m^{1.7}$. Sparse equation solvers perform poorly on vector machines because the average vector length is short, as demonstrated in Table 3.1. The performance of a vectorized sparse solver written in FORTRAN, even for reasonably long vectors, is poor. ADAMS has an average vector length of less than two [28] and approximately twenty percent of the computation is solving equations [26]. Only a factor of five improvement in the overall computational rate is possible regardless of how well the rest of the code vectorizes if the equation solver remains unchanged. The equations generated by ADAMS do have a pattern; therefore a hierarchal equation solver [29] is possible.

The vectorized FORTRAN solver is easily outperformed by a solver that runs in scalar mode on the Cray [20]. A preprocessor generates scalar machine code that executes at a rate of sixteen to eighteen MFLOPS. The code length, in 64-bit words, is approximately equal to the

TABLE 3.1

FORTRAN SPARSE MATRIX SOLUTION PERFORMACE

Average Vector Length	Number of Equations	Execution Rate (MFLOPS)
4.2	391	1.59
12.2	507	5.39
27.0	2323	11.0

SOURCE: D. A. Calahan [31]

number of floating point operations performed, and for that reason the FORTRAN code is preferred for large equation sets.

3.3.3 Banded Equation Solvers

Banded equation solvers are for equation sets where all of the nonzero coefficients are in a band along the main diagonal. Again, the performance of a CAL solver is superior to the FORTRAN code [19], as shown in Figure 3.1. Poor memory management and the overhead associated with pivoting account for the performance difference. Both solvers experience a drop in the execution rate at half bandwiths that are one greater than multiples of sixty-four. The Cray ALU is restarted after every sixty-four operations because the vector registers are sixty-four words long. An equation set with a half bandwidth of sixty-five has twice as many ALU startups, but only a few more arithmetic

operations than a set with a half bandwidth of sixty-four. Their execution rates, 117 and 76 MFLOPS respectively, show the importance of keeping the ALU busy at all times.

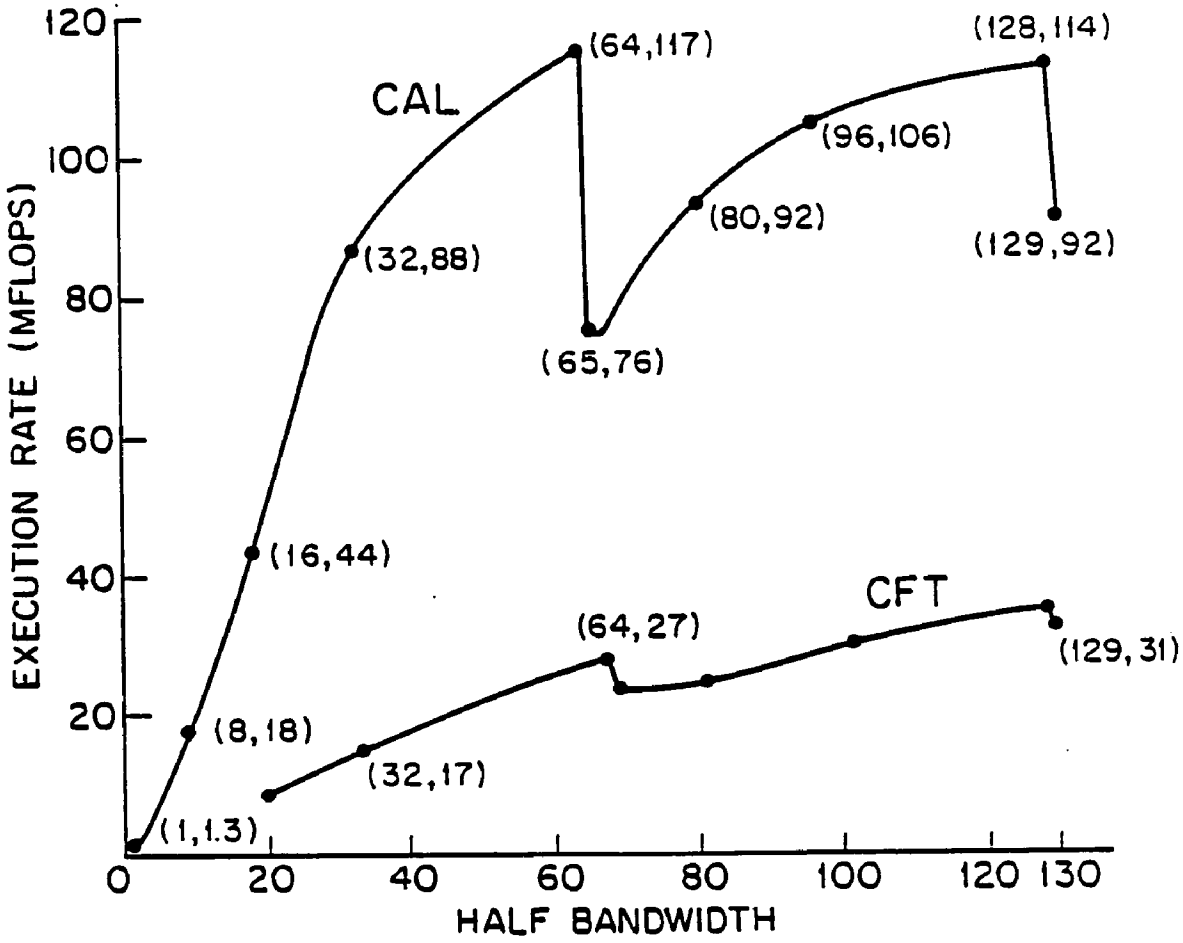


Fig. 3.1. The execution rates of two unsymmetric banded equation solvers for 1024 equations. CAL stands for Cray Assembly Language, and CFT, for Cray FORTRAN. Note that the CFT code, from LINPACK, includes pivoting. Reprinted from D. A. Calahan [19].

3.3.4 Block-Oriented Solvers

Equations with a globally sparse structure and a dense local structure are efficiently solved with block oriented equation solvers [30]. The nonzero entries are partitioned into rectangular blocks that allow compact storage. Blocks on the main diagonal are decomposed using a dense matrix decomposition kernel, and the off diagonal blocks are factored by using matrix multiplication and subtraction kernels. The execution rates, given in Table 3.2, compare favorably to the rates for banded solvers. The rates for the decompostion and substitution kernels are also applicable to completely dense equation sets.

3.4 Summary

From elementary considerations physically stiff systems were shown to produce stiff differential equations, and special integration methods, such as the Gear backward difference method, are necessary for solving them.

The effects of equation structure and vector length on the execution rates of linear equation solvers for the Cray-1 were explored by comparing several benchmarks from the literature. Not surprisingly, long vector lengths and a highly patterned structure (at least locally) are necessary for high computational rates.

TABLE 3.2

BLOCK-ORIENTED SOLUTION PERFORMANCE

Square Block Size	LU Decomposition Kernel (MFLOPS)	Substitution Kernel (MFLOPS)	Multiplication Kernel (MFLOPS)
1	1.1	.5	2.7
2	2.5	3.0	7.6
3	6.9	5.7	17.
4	8.5	12.	26.
6	18.	21.	43.
8	23.	30.	60.
12	45.	46.	84.
16	60.	58.	98.
32	94.	89.	124.
64	123.	117.	141.

SOURCE: D. A. Calahan [30]

CHAPTER 4

THE DYNAMICS OF DEFORMABLE MECHANICAL SYSTEMS

4.1 Introduction

A short discussion of the finite element method and a technical review of the previous research available in the literature on the dynamics of deformable mechanical systems in the beginning of this chapter gives the necessary background information for understanding the later sections. The weak form of Lagrange's equation for quasi-coordinates is derived from Cauchy's first law of motion, and it is then used to develop the equations of motion for a deformable body. Linear coordinate reduction methods are introduced and used for specializing the general formulation to linear elasticity, and the equations of motion for rigid bodies are derived. Equations for concentrated forces, follower forces, the body forces from gravity, and several common joint types are also presented.

4.2 Desirable Characteristics

In computational dynamics the desirable characteristics for a formulation are often contradictory, and therefore all formulations are, to some extent, a compromise. Listing the most desirable characteristics,

however, serves to focus the direction of the research and prevent it from becoming an elegant but computationally useless excercise in abstract reasoning.

1) The formulation should be simple enough that a self-formulating program can be based on it.
2) The necessary input data should be little more than a minimal description of the geometrical and inertial properties of the system.
3) Generality is important. A library of commonly used joints and forces should be available, and a mixture of plastic, elastic, and rigid bodies should be handled in a logical and convenient manner.
4) Vector processing is important for computational speed, and therefore the formulation must be structured in a way to take advantage of vector processors.
5) The numerical methods should be both stable and efficient, and, of course, accurate.

4.3 The Finite Element Method

4.3.1 The General Approach

The finite element method is, in essence, just a method for approximating a function on a domain that is in a finite dimensional continuum. It is used primarily for numerically solving partial differential equations, but Zienkiewicz [32] uses it to derive several numerical

integration methods for ordinary differential equations, including the Crank-Nicholson and Newmark-β methods. While Turner et. al. [33] are generally given credit for inventing the finite element method, it is most readily understood by examining its mathematical antecedent, the Rayleigh-Ritz method [34].

Consider the following linear operator problem:

$$L\underline{x} = \underline{f} \qquad (4.3.1.1)$$

where

L is a linear operator.

$\underline{x}$ is an unknown vector.

$\underline{f}$ is a known vector function.

To solve equation 4.3.1.1 by the Rayleigh-Ritz method, an approximation of the solution is expressed as a series of functions multiplied by scalar coeffients. The series is not necessarily infinite, but for special cases of this method, such as the Fourier series, the series is infinite and converges to the exact solution.

$$\tilde{\underline{x}} = a_i \underline{u}_i \qquad (4.3.1.2)$$

where

$\tilde{\underline{x}}$ is the approximate solution.

a_i are the unknown scalar coefficients.

$\underline{u}_i$ are the basis functions.

The solution of equation 4.3.1.3 determines the scalar coefficients.

$$\langle \underline{u}_i, L\underline{u}_j \rangle a_j = \langle \underline{u}_i, \underline{f} \rangle \qquad (4.3.1.3)$$

where

$\langle \cdot, \cdot \rangle$ is the inner product.

Choosing basis functions that satisfy the boundary conditions on a complicated domain is difficult. One approach introduces a function that satisfies the boundary conditions, but which does not necessarily satisfy equation 4.3.1.1, and then basis functions that are zero on the boundary are chosen.

$$\begin{aligned} \langle \underline{u}_i, L\underline{u}_j \rangle a_j &= \langle \underline{u}_i, \underline{f}-\underline{g} \rangle \\ \tilde{\underline{x}} &= \underline{g} + a_i \underline{u}_i \\ \underline{u}_i &= 0|_\Gamma \\ \underline{g} &= \underline{G}(\Gamma) \end{aligned} \qquad (4.3.1.4)$$

where

$\underline{g}$ satisfies the boundary conditions.

Γ is the boundary of the domain.

While choosing the basis functions is easier, the task is still difficult for oddly-shaped domains, and $\underline{g}$ is often difficult to construct. The finite element method is a convenient way to construct the basis functions. First, partition the domain into regular subdomains, or finite elements, and associate with each element a finite number of nodes interior to the finite element or on its boundary. Several elements can share a common node, but interior nodes

belong to one, and only one, element. Then, for each node, choose a basis function that is nonzero only within the elements associated with the node, assumes the value of one at the node, and is zero at all other nodes. The scalar coefficients are the values of the approximate solution at the nodes. Rewriting equation 4.3.1.2 using finite element notation makes the relationship clear.

$$\tilde{x} = \tilde{x}_{\alpha} \cdot N_{\alpha} \qquad (4.3.1.5)$$

where

$\tilde{x}_{\alpha}$ is the approximate solution at node α.

N_{α} is the basis function associated with node α.

Choosing simple geometrical shapes for the finite elements allows the use of standard polynomial interpolation functions, such as the Lagrangian interpolation functions, for the basis functions, and therefore the basis functions are sometimes called "interpolation" functions. The basis functions are also called "shape" functions because their choice determines the possible deformed shapes that an element can assume.

The basis functions are nearly orthogonal because, by construction, the domain of compact support for a basis function intersects the domains of compact support for only a few other basis functions. Most of the coefficients in equation 4.3.1.4 are therefore zero, and the matrix is diagonally dominant. Depending on the ordering of the

equations, either sparse matrix or banded equation solvers are appropriate for efficiently solving the system of equations.

4.3.2 The Deformation Field From the Lagrangian Viewpoint

When a structure deforms in a finite, nonlinear manner the domain of integration, the shape of the mesh and the deformation field must be defined much more carefully than for linear elasticity. The two basic approaches are the Eulerian and the Lagrangian [35].

The mesh does not deform in the Eulerian approach, and as a consequence, mass transport terms are added to the problem. While the velocity field of the points has meaning, the deformation field does not since the particles are not individually defined. Fluid dynamics problems are often solved using an Eulerian reference frame. For solid mechanics the Eulerian reference frame has two disadvantages: the boundaries of an object are not accurately known, and the lack of a deformation field makes the implementation of strain dependent material models difficult. Individual particles with a Lagrangian reference frame are usually used in conjunction with the Eulerian reference frame to overcome its disadvantages. When a structure deforms to the point of tearing, or when voids are created, the Eulerian frame is useful.

Each particle in a body is labelled with its coordinates in the reference configuration, $\underline{\xi}$, in the Lagrangian reference frame. The deformation field is

defined as the difference between the current position of a particle $\underline{\xi}$, namely, $\underline{x}(\underline{\xi},t)$, and its initial position. The velocity field is simply the derivative of $\underline{x}$ with respect to time. For the finite element method, a direct application of equation 4.3.1.5 gives the approximate deformation and velocity fields.

$$\begin{aligned} u_i(\underline{\xi},t) &= (x_{\alpha i}(t) - \xi_{\alpha i})N_\alpha(\underline{\xi}) \\ &= u_{\alpha i}(t)N_\alpha(\underline{\xi}) \\ \dot{u}_i(\underline{\xi},t) &= \dot{x}_{\alpha i}(t)N_\alpha(\underline{\xi}) \end{aligned} \qquad (4.3.2.1)$$

where

$u_i(\underline{\xi},t)$ is the i-th displacement component of particle $\underline{\xi}$ at time t.

$u_{\alpha i}(t)$ is the i-th displacement component of node α at time t.

$x_{\alpha i}(t)$ is the i-th component of the position of node α at time t.

$N_\alpha(\underline{\xi})$ is the basis function of node α.

More sophisticated approximations than equation 4.3.2.1 are necessary in some applications, e.g., shell theory. The components of the deformation gradient sometimes are, for example, used with Hermitian interpolation to represent the deformation field for Kirchhoff plate elements [32].

$$u_i(\underline{\xi},t) = u_{\alpha i}(t)N_\alpha + u'_{\alpha i}(t)M_\alpha(\underline{\xi}) \qquad (4.3.2.2)$$

where

$u'_{\alpha i}(t)$ is a selected component of the

of the deformation gradient

$M_\alpha(\underline{\xi})$ is a basis function that is zero at the nodes.

Other names for the Lagrangian coordinate system are "material coordinates" because they are associated with particular particles, and "convected coordinates" because they move with the particles (think of convective heat transfer).

The integration domain changes as the structure deforms. To eliminate the problems associated with the changes in the shape of the domain, an additional coordinate transformation from classical calculus is used. The integrals are usually evaluated numerically by using Gaussian quadrature [23].

$$\int_{\Omega(t)} f(\underline{x},\dot{\underline{x}},t)d\Omega = \int_{\Omega_o} f(\underline{x}(\underline{\xi},t),\dot{\underline{x}}(\underline{\xi},t),t)\ J\ d\Omega_o \qquad (4.3.2.3)$$

where

f is an arbitrary integrable function.

$\Omega(t)$ is the current domain.

Ω_o is the reference domain.

J is $\det(\frac{\partial x_i}{\partial \xi_j})$

4.3.3 The Stiffness Matrix for Linear Elasticity

In this section the stiffness matrix for a linearly elastic finite element with the deformations described by equation 4.3.2.1 is derived to demonstrate how the material in sections 4.3.1 and 4.3.2 is used. The concepts of a stiffness matrix and a mass matrix (section 4.3.4) are introduced now as a preface to the discussion of the linear coordinate reduction methods commonly used in structural dynamics. No boundary conditions are introduced for this example, and therefore the stiffness matrix derived here is singular.

The constitutive relation used here is valid for an anisotropic material, but only for small strains [35].

$$\sigma_{ij} = E_{ijkn}\varepsilon_{kn} \tag{4.3.3.1}$$

$$\begin{aligned}\varepsilon_{ij} &= \tfrac{1}{2}(u_{i,j} + u_{j,i}) \\ &= \bar{B}_{ijk}u_k\end{aligned} \tag{4.3.3.2}$$

where

σ_{ij} is the symmetric small strain stress tensor.

ε_{ij} is the symmetric small strain tensor.

E_{ijkn} is a constant fourth order tensor.

$\bar{B}_{ijk}$ is the linear operator defined by equation 4.3.3.2.

The principle of virtual displacements is also called the principle of virtual work [36].

$$\int_{\Omega}\sigma_{ij}\delta\varepsilon_{ij}d\Omega = \int_{\Omega}f_i\delta u_i d\Omega + \int_{\Gamma_t}t_i\delta u_i d\Gamma$$
$$\delta\varepsilon_{ij} = \varepsilon_{ij}(\delta\underline{u}) \qquad (4.3.3.3)$$

where

f_i is the i-th component of the body force.

t_i is the i-th component of the surface traction.

Γ_t is the surface subject to $\underline{t}$.

Introducing the finite element approximation of the deformations into equation 4.3.3.2 results in an expression for the strain throughout the element as a function of the deformations at the nodes.

$$\varepsilon_{ij} = \bar{B}_{ijk}N_{\alpha}u_{\alpha k} \equiv B_{ij\alpha k}u_{\alpha k} \qquad (4.3.3.4)$$

Substituting equations 4.3.3.1, 4.3.3.2, and 4.3.3.4 into equation 4.3.3.3, and canceling of the arbitrary variations on both sides results in a set of linear equations for the deformations at the nodes.

$$K_{\beta n\alpha m}u_{\alpha m} = f_{\beta n}$$
$$K_{\beta n\alpha m} = \int_{\Omega}E_{ijkr}B_{ij\beta n}B_{kr\alpha m}d\Omega \qquad (4.3.3.5)$$
$$f_{\beta n} = \int_{\Omega}f_n N_{\beta}d\Omega + \int_{\Gamma_t}t_n N_{\beta}d\Gamma$$

where

$K_{\beta n\alpha m}$ is the stiffness matrix.

$f_{\beta n}$ is the generalized force vector.

For nonlinear problems the strain measure is a nonlinear operator (unless the deformation gradient is used), and equation 4.3.3.3 is nonlinear. Taking the first variation of the left hand side of equation 4.3.3.3 gives the tangential stiffness matrix necessary for the incremental and Newton-Raphson methods.

4.3.4 The Mass Matrix

Inertial forces are treated as applied body forces by using the D'Alembert principle [37].

$$\begin{aligned} \int_\Omega f_i \delta u_i d\Omega &= \int_\Omega (-\rho \ddot{u}_i) \delta u_i \, d\Omega \\ &= - (\delta_{ij} \int_\Omega \rho N_\alpha N_\beta d\Omega) \ddot{u}_{\beta j} \delta u_{\alpha i} \\ &= - M_{\alpha i \beta j} \ddot{u}_{\beta j} \delta u_{\alpha i} \end{aligned} \qquad (4.3.4.1)$$

where

$$M_{\alpha i \beta j} \equiv \delta_{ij} \int_\Omega \rho N_\alpha N_\beta d\Omega$$

and is referred to as the consistent mass matrix.

ρ is the density.

Alternatively, the mass matrix is derived from the expression for kinetic energy.

$$
\begin{aligned}
T &= \frac{1}{2}\int_{\Omega}\rho\dot{u}_i\cdot\dot{u}_i d\Omega \\
&= \frac{1}{2}\int_{\Omega}\rho(\dot{u}_{\alpha i}N_\alpha)\delta_{ij}(\dot{u}_{\beta j}N_\beta)\ d\Omega \\
&= \frac{1}{2}\dot{u}_{\alpha i}(\delta_{ij}\int_{\Omega}\rho N_\alpha N_\beta d\Omega)\ \dot{u}_{\beta j} \\
&= \frac{1}{2}\dot{u}_{\alpha i}M_{\alpha i\beta j}\dot{u}_{\beta j}
\end{aligned}
\qquad (4.3.4.2)
$$

where

T is the kinetic energy.

The consistent mass matrix is not diagonal, and for several reasons a diagonal or lumped mass matrix may be preferable:

1) A diagonal mass matrix takes less storage.
2) The nodal coupling between the inertial terms in a consistent matrix smears any shock waves in the structure.
3) A diagonal mass matrix is trivially inverted.
4) The transient response and the eigenvalues are sometimes more accurate when a lumped mass matrix is used.

While the consistent mass matrix is uniquely defined, there are many methods for calculating a lumped mass matrix [32]. Several of the more popular methods follow:

1) Basis functions different from those used in evaluating the stiffness matrix are used. Namely, the basis functions are piecewise constants that are one in the neighborhood of the node and zero everywhere else. Remembering that

the sum of all of the basis functions is one, obviously at any point only one piecewise constant basis function is nonzero, and therefore the mass matrix is diagonal.

2) The diagonal terms of the consistent mass matrix are scaled so that the total mass of the body is preserved and the off-diagonal terms are set to zero.

3) One interesting approach uses a numerical integration method where the sampling points are at the nodes, and therefore all of the basis functions except the one associated with the sample point are zero, again resulting in a diagonal mass matrix.

4) Another approach is to simply sum across the columns of a row in the consistent mass matrix for the row entry in the diagonal mass matrix [16].

For small deformations the mass matrix, intuitively, is constant, just as the stiffness matrix is constant for linear elasticity. With large, nonlinear deformations, the invariance of the mass matrix is not as intuitive since the inertia tensor of the body and the domain of integration for equation 4.3.4.1 are clearly functions of the deformations. The proof that the mass matrix is constant regardless of the magnitude and nonlinearity of the deformations is based on the conservation of mass law.

$$\int_{\Omega_o} \rho_o d\Omega_o = \int_{\Omega(t)} \rho d\Omega(t)$$

$$= \int_{\Omega_o} \rho J d\Omega_o \qquad (4.3.4.3)$$

$$\rho_o d\Omega_o = \rho J d\Omega_o = dm$$

where

Ω is the deformed configuration.

Ω_o is the reference configuration.

ρ is the density function.

ρ_o is the density in the reference configuration.

dm is a differential mass.

By using equation 4.3.4.3, the domain of integration for the mass matrix is changed from a spatial domain to a material domain, which is clearly independent of the deformations of the body.

$$M_{\alpha i \beta j} = \delta_{ij} \int_{\Omega} N_\alpha N_\beta \rho d\Omega$$

$$= \delta_{ij} \int_{\Omega_o} N_\alpha N_\beta \rho J d\Omega_o \qquad (4.3.4.4)$$

$$= \delta_{ij} \int_{B} N_\alpha N_\beta dm$$

where

B is the material domain.

4.4 Linear Coordinate Reduction

The transient response of a linearly elastic structure requires the solution of a system of simultaneous differential equations.

$$M_{\alpha i \beta j}\ddot{u}_{\beta j} + C_{\alpha i \beta j}\dot{u}_{\beta j} + K_{\alpha i \beta j}u_{\beta j} = f_{\alpha i} \qquad (4.4.1)$$

where

$C_{\alpha i \beta j}$ is the damping matrix.

For most problems the solution is adequately approximated by a linear combination of a relatively small number of deformation modes.

$$u_{\beta j}(t) = H_{\beta j k}\eta_k(t) \qquad (4.4.2)$$

where

$H_{\beta j k}$ is the k-th deformation mode shape.

η_k is the k-th mode amplitude.

Noting that $H_{\beta j k}$ is constant, a small set of differential equations is derived by substituting equation 4.4.2 into equation 4.4.1, and applying the principle of virtual work.

$$M^R_{ij}\ddot{\eta}_j + C^R_{ij}\dot{\eta}_j + K^R_{ij}\eta_j = f^R_i \qquad (4.4.3)$$

$$M^R_{ij} = H_{\alpha mi} M_{\alpha m\beta k} H_{\beta kj}$$
$$C^R_{ij} = H_{\alpha mi} C_{\alpha m\beta k} H_{\beta kj}$$
$$K^R_{ij} = H_{\alpha mi} K_{\alpha m\beta k} H_{\beta kj}$$
$$f^R_i = H_{\alpha mi} f_{\alpha m}$$

where

M^R_{ij} is the reduced mass matrix.

C^R_{ij} is the reduced damping matrix.

K^R_{ij} is the reduced stiffness matrix.

f^R_i is the reduced force vector.

While reducing the number of equations is attractive, the solution to the reduced set of equations is meaningless unless the approximation made in equation 4.4.2 is accurate. Although experimental methods exist for determining the mode shapes, and the mass, stiffness, and damping matrices, analytical methods are emphasized here.

4.4.1 Modal Analysis

One choice of mode shapes is a truncated series of eigenvectors calculated from an undamped system that is otherwise equivalent to equation 4.4.1 [37].

$$(K_{\alpha i\beta j} - \omega_k^2 M_{\alpha i\beta j})H_{\beta jk} = 0 \qquad (4.4.1.1)$$

where there is no summing on k and

ω_k^2 is the k-th eigenvalue.

$H_{\beta jk}$ is the k-th eigenvector.

The eigenvectors are usually normalized.

$$\begin{aligned} H_{\alpha ki}M_{\alpha k\beta n}H_{\beta nj} &= \delta_{ij} \\ H_{\alpha ki}K_{\alpha k\beta n}H_{\beta nj} &= \omega_i^2\delta_{ij} \quad \text{no sum on i} \end{aligned} \qquad (4.4.1.2)$$

The reduced mass and stiffness matrices are diagonal, and the damping matrix is diagonalized by defining it as a linear combination of the mass and stiffness matrices.

$$\begin{aligned} C_{\alpha i\beta j} &= aM_{\alpha i\beta j} + bK_{\alpha i\beta j} \\ C^R_{ij} &= H_{\alpha ki}C_{\alpha k\beta n}H_{\beta nj} \\ &= (a + b\omega_j^2)\delta_{ij} \quad \text{no sum on j} \end{aligned} \qquad (4.4.1.3)$$

where

a and b are positive constants.

Choosing how many and which eigenvectors to use is a problem without a well defined solution. Certainly any modes with natural frequencies near the dominant frequency components of the applied forces are kept. Beyond that, however, keeping the lowest modes is the traditional approach because they are the "softest" deformation modes, and they will therefore account for the largest contributions to the stresses in the system. Systems with redundant constraints require additional modes to prevent

the body from locking up. Care must also be exercised so that chosen modes are not nearly orthogonal to the applied forces. Consider a simple beam that extends in the z direction, is one inch thick in the x direction, and three inches thick in the y direction. The lowest modes are primarily in the x-z plane and are nearly orthogonal to any forces applied in the y direction. The designer of the beam undoubtedly made the beam thick in the y direction in anticipation of large loads in that direction, and substantial errors in the stress calculations are likely unless modes with significant components in the y direction are included.

The boundary conditions imposed on equation 4.4.1.1 affect the mode shapes, and therefore they affect the number of modes that must be retained for an accurate solution. Free-free modes are often used to represent deformable bodies because of their nice mathematical properties (section 4.5.5), but they are not necessarily the best choice of mode shapes, e.g., a large number of free-free modes are necessary for accurately representing the lowest mode of a cantilevered beam.

4.4.2 Component Mode Synthesis

Another method for creating the modes for linear coordinate reduction is component mode synthesis [13],[38]. It is an extension of modal analysis that allows the assembly of complicated structures from smaller substructures. When the components of a structure are designed by different groups, or when frequent design changes are made, component mode synthesis is useful.

Each substructure is represented by a combination of static deformation modes and vibrational modes. The static modes are used to enforce the geometric compatibility at the boundaries where the substructures are joined. The deformations at the boundary nodes are functions only of the static modes while the deformations at the interior nodes are functions of both the static and vibrational modes.

Each coordinate at each boundary node is perturbed in turn while holding the remaining boundary coordinates to zero to generate a static mode.

$$\left[\begin{array}{c|c} K^{BB} & K^{BI} \\ \hline K^{IB} & K^{II} \end{array}\right] \left[\begin{array}{c} u^B \\ \hline u^I \end{array}\right] = \left[\begin{array}{c} f^B \\ \hline 0 \end{array}\right]$$

$$\left[K^{IB}\right]\left[u^B\right] + \left[K^{II}\right]\left[u^I\right] = \left[0\right]$$

$$\left[u^I\right] = -\left[K^{II}\right]^{-1}\left[K^{IB}\right]\left[u^B\right] \qquad (4.4.2.1)$$

$$\left[H^S\right] = \left[\begin{array}{c} I \\ \hline -\left[K^{II}\right]^{-1}\left[K^{IB}\right] \end{array}\right]$$

where

$\left[K\right]$ is the partitioned stiffness matrix.

$\left[u^B\right]$ is the boundary deformation vector.

$\left[u^I\right]$ is the interior deformation vector.

$\left[H^S\right]$ is the static mode transformation matrix.

The vibrational modes are generated by using a truncated series of eigenvectors from the interior node eigenvalue problem. All of the boundary nodes are constrained to zero.

$$\left[\left[K^{II}\right] - \left[\omega^2\right]\left[M^{II}\right]\right]\left[H^{I}\right] = \left[0\right] \qquad (4.4.2.2)$$

where

$\left[M^{II}\right]$ is the interior node mass matrix.

$\left[\omega^2\right]$ is the diagonal eigenvalue matrix.

$\left[H^{I}\right]$ is the interior mode transformation matrix.

The complete transformation matrix is assembled from equations 4.4.2.1, and 4.4.2.2.

$$\left[\frac{u^B}{u^I}\right] = \left[H^S \middle| \frac{0}{H^I}\right]\left[\frac{u^B}{\zeta^I}\right] \qquad (4.4.2.3)$$

$$= H_{\beta jk}\eta_k$$

where

$\left[\zeta^I\right]$ is the amplitude of the interior modes.

Reduced mass and stiffness matrices are calculated from their definitions in equation 4.4.1.3. Unlike for pure modal analysis, the reduced matrices are not diagonal because of the mixture of nodal and modal coordinates. Global matrices are assembled from the substructure matrices in the same manner that the substructure matrices were

assembled from their elements since the boundary node coordinates are retained in the coordinate reduction. Although modal analysis is applicable to the reduced system, the reduced system is usually small enough for the direct solution of the transient response. Boundary conditions are applied to eliminate the rigid body modes since the rigid body modes do not necessarily appear explicitly in the transformation.

4.5 A Technical Review of the Previous Research

The diverse and abundant research on deformable mechanical systems is characterized in this section by the choices of coordinates and the method the researchers used to eliminate the redundancy between the rigid body coordinates and deformation field of the body. Once the choices are made, then both the theoretical and computational structure of a formulation are largely determined.

4.5.1 Total Displacement at the Nodes

Defining the displacement at the nodes as the sum of the rigid body displacements and the body deformations is the simplest approach. There is no redundancy in the coordinate set, indeed, since there are no rigid body coordinates, no distinction is made between different bodies in the program. The vectorized programs DYNA3D and NIKE2D,

both mentioned in Chapter 1, use this approach very successfully for analyzing dynamic finite strain problems with material nonlinearities.

The total displacement coordinate system is used for dynamics problems where only the initial transient is of any importance. Compared to other formulations, the solution cost for linearly elastic systems is high because linear coordinate reduction methods can not be used without having separate rigid body coordinates to handle the finite rigid body rotations.

4.5.2 Joint Coordinates for Rigid Body Motion

The early work in computational rigid body dynamics used joint coordinates to generate small, dense equation sets that are readily solved by small computers [50]. Likins [7] and Boland [8] extended the rigid body dynamics formulations to include linearly elastic parts represented by their vibrational modes. Sunada and Dubowsky [11] used a similar approach, but they included component mode synthesis in their work.

To understand why joint coordinates are a near minimal coordinate set, consider a chain of bodies connected with joints and one end attached to ground. Attach a local coordinate frame at each joint contact on every body. The relative rotations and translations between the two local coordinate frames, one on each body defining the joint, are the joint coordinates. Deleting the coordinates that are constant regardless of the configuration of the bodies

creates a minimal coordinate set for the chain. The remaining coordinates uniquely specify the configuration of the system.

Real systems are usually more complex, but the principle still applies. For any system that does not have closed loops, no modification of the approach in the preceding paragraph is necessary because there is only one part path to ground from any point in the system, i.e., the joint coordinates are not redundant. Closed loops have more than one unique path to ground. Each loop is broken at one joint to restore the uniqueness of the paths to ground and constraint equations are introduced to close the loops. The coordinate set is no longer minimal, e.g., a planar four bar linkage has only one degree of freedom, but joint coordinates use three coordinates to describe the system.

The acceleration vector of any point is a function of all of the coordinates on its part path to ground. Each joint coordinate, associated with joint J, generates a j×j submatrix that is scattered into the equations of motion, where j is the number of joint coordinates on the path from joint J, including joint J. For a chain of M parts connected with single degree of freedom joints, the number of operations to generate the rigid body mass matrix is proportional to M^3 [26].

Adding flexibility to the parts is computationally equivalent to adding joints because the additional deformation coordinates increase the part path length to

ground. A system with a large number of deformation modes is very expensive to simulate because of the cubic growth rate of the cost for evaluating the equations of motion. To alleviate the problem, Sunada and Dubowsky treat the nodes associated with joint constraints as boundary nodes and apply component mode synthesis. Only the static modes associated with the joints on a particular path to ground contribute to the part path length since the deformation of the joint nodes is independent of the other static modes and the interior vibrational modes. Component mode synthesis also eliminates the problem of locking the vibratonal modes by inadvertently overconstraining the system.

Vectorization of joint coordinate formulations is difficult. Associated with the calculation of each jxj submatrix are many gather operations across all of the parts and their nodes on the part path to ground. Each coefficient in the submatrix requires only a few arithmetic operations to evaluate it, and therefore the ratio of the gather time to the calculation time is high and the overall computational rate is low.

Joint coordinates are usually used with a nonstiff integration method. Generalized accelerations are calculated from the linear algebraic equations of motion, and are then integrated into velocities, which are, in turn, integrated into displacements. An explicit formulation for the Jacobian in a stiff integration method is very

complicated and the number of operations for numerically differencing it is proportional to M^4 for systems of rigid bodies [26].

4.5.3 Rigid Body Displacement Coordinates

To avoid the large computational growth rates associated with joint coordinates, each body is assigned its own set of independent, rigid body displacement coordinates. Orlandea used center of mass coordinates for the rigid body dynamics program ADAMS [10]. Attached to each body is a local reference frame that locates the nodes used for defining the joints and applied forces. The equations of motion are formulated by using Lagrange's equation. In addition to the generalized coordinates and the Lagrange multipliers, the angular momenta of each body and the applied force values are variables to simplify the explicit evaluation of the Jacobian. The equations are generated in much the same way as the stiffness matrix for structural analysis. Different components, such as parts or joints, generate submatrices that are scattered into a global equation set. Sparse matrix equation solvers are used to invert the Jacobian for the Newton iteration used in the Gear integration method.

Song and Haug extended Orlandea's work by developing a two dimensional formulation with an Euler-Bernoulli beam element. Structures larger than a beam are created by

connecting individual beams with zero degree of freedom joints. As with ADAMS, the Gear integration method is used with an explicit, sparse Jacobian.

Shabana and Wehage followed with a generalized, three dimensional analysis code [15] that includes component mode synthesis and which does not require that the mesh origin coincide with the center of mass. A coordinate partitioning algorithm [14] eliminates the redundant coordinates and constraints numerically, and the minimal coordinate set is integrated with a nonstiff method. The redundant coordinates are solved for by using Newton interation on the constraint equations.

Vectorization is straightforward. Each body is represented by its own set of rigid body and deformation coordinates so that the momentum equations are calculated without any gather operations. The equations defining the constraints and applied forces between bodies unavoidably require gather operations, but the gathers do not dominate the computational cost. The main limiting factor on the execution rate is the sparse matrix solver, but a block-oriented or hierarchal solver could replace it.

4.5.4 Quasi-coordinates

A major difficulty with formulations using generalized coordinates is the complexity of the expressions for the kinetic energy and the generalized accelerations. While the rigid body coordinates are in the ground frame, the deformations are in the mesh frame, and, in order to add

the rigid body velocities to the deformation rates, the rigid body velocities are rotated into the mesh frame by a transformation that is a complicated function of the rigid body angular coordinates. Successive differentiation of the kinetic energy expression complicates the equations of motion with nonlinear terms involving the rigid body angular coordinates and their time derivatives. Not only do the nonlinear terms from the angular coordinates add complexity and cost to the evaluation of the equations of motion, but they can also slow down the Newton iteration convergence rate and force the frequent re-evaluation of the Jacobian.

The introduction of velocity measures that do not integrate into generalized displacements in the kinetic energy expression simplifies the equations of motion. The time integrals of these velocity measures are called "quasi-coordinates". In 1976, de Veubeke proposed a set of quasi-coordinates for the rigid body coordinates and a linearization of the Tisserand axes (discussed in the next section) in an important theoretical paper [9] on linearly elastic systems. Although quasi-coordinates appear frequently in theoretical papers [39], there is little evidence of their use in generalized programs. The author proposed the same set of quasi-coordinates for analyzing sytems with large deformations and nonlinear materials in a paper written in 1981, and presented in 1982 [1]. The major difference in the two papers is definition of the local

coordinate system. A detailed development of quasi-coordinates and their application is presented later in this chapter.

4.5.5 Local Coordinate Reference Frames

A set of generalized coordinates must be both onto and one-to-one with respect to the configuration space. The deformation field has both rigid body translations and rotations in its range which must be constrained out to eliminate the redundancy with the rigid body coordinates. Several different types of local reference frames are available to eliminate the extra degrees of freedom and possibly simplify the rigid body equations of motion [40].

The locally attached reference frame is the simplest. A body is constrained to its local reference frame by applying displacement boundary conditions. Both in theory and in application, this approach is the same as the imposition of displacement boundary conditions in structural analysis to eliminate the singularity of the stiffness matrix where the degrees of freedom are eliminated by either explicitly deleting them or by a penalty method. The rigid body modes are dropped when the deformations of the body are represented by their free-free vibrational modes, thereby fixing the location of the center of mass and the orientation of the body in the local frame of the body. No simplification of the rigid body equations is possible with

the locally attached frame because the origin of the local frame is not necessarily at the center of mass, or aligned with the proper axes.

All of the methods proposed for eliminating the coordinate redundancy and simplifying the rigid body equations constrain the instantaneous center of mass to the origin of the local coordinate frame. Three linear equations are necessary.

$$\int_{\Omega}(r_i + u_i)\rho d\Omega = 0 \qquad (4.5.5.1)$$

where

r_i is the i-th coordinate of a material point in the reference configuration.

u_i is the i-th deformation coordinate of a material point in the deformed configuration.

The principal axes frame diagonalizes the inertia tensor with three constraint equations.

$$e_{ijk}\int_{\Omega}(r_j + u_j)(r_k + u_k)\rho d\Omega = 0 \qquad (4.5.5.2)$$

where

$i = 1,2,3$ and there is no sum on j and k.

While the constraints do diagonalize the inertia tensor, it is still a function of the deformations. Compared to the other approaches in the literature, this one does not enjoy much favor; the additional complexity of adding the constraint equations is not outweighed by the simplifications in the rigid body equations.

The Tisserand, or mean axes, frame simplifies the angular momenta expressions by imposing three nonholonomic constraint equations that set the angular momenta relative to the floating frame to zero.

$$e_{ijk}\int_{\Omega}(r_j + u_j)\dot{u}_k \rho d\Omega = 0 \qquad (4.5.5.3)$$

Nonholonomic constraints are, in general, more difficult to apply than holonomic constraints, and the Tisserand frame therefore has little computational utility except for the special case of linear elasticity where the free-free vibrational modes are used to represent the deformations of the body. De Veubeke, in his previously cited paper [9], linearized equation 4.5.5.3 by deleting u_j from the equation. By noting that equation 4.5.5.1 and the linearized version of equation 4.5.5.3 are equivalent to requiring the inner product of the rigid body modes and the deformation modes over the material domain to be zero, and that the free-free modes with non-zero eigenvalues satisfy the requisite orthogonality conditions, he eliminates the constraint equations from his equations of motion. The location of the center of mass remains fixed in the mesh coordinate system, and the rigid body coordinates are defined from ground to the center of mass.

Buckens introduced a holonomic constraint set to replace the Tisserand constraints [41].

$$e_{ijk}\int_{\Omega} r_j u_k \rho d\Omega = 0 \qquad (4.5.5.3)$$

Differentiating the constraint set with time gives the linearized nonholonomic approximation of the Tisserand constraints. Buckens also showed that deleting the rigid body modes guarantees that the constraints are satisfied.

A component modes synthesis representation of the deformations is more difficult to use with floating axes than a free-free modal representation because its rigid body modes do not necessarily appear explicitly in the transformation matrix, and the modes are generally not orthogonal with respect to the mass matrix.

In fact, the use of the Tisserand or Bucken's frame with mode shapes having any displacement boundary conditions, e.g., a cantilevered beam, must be done with care. First, the modes are made orthogonal to the rigid body modes with respect to the mass matrix.

$$H_{\alpha ic} = \bar{H}_{\alpha ic} - \sum_{a=1}^{6} \left(\frac{\bar{H}_{\gamma kc} M_{\gamma k\beta j} R_{\beta ja}}{R_{\gamma ka} M_{\gamma k\beta j} R_{\beta ja}}\right) R_{\alpha ia} \qquad (4.5.5.4)$$

where

$\bar{H}_{\alpha ic}$ is the c-th mode with displacement constraints.

$R_{\beta ja}$ is the a-th rigid body mode.

Second, the reduced mass and reduced stiffness matrices are calculated from the original free-free mass and stiffness matrices by using equation 4.4.3. While the resulting modes are not eigenvectors of either the free-free or the constrained system, they do satisfy the floating axes constraint equations. The reduced mass and stiffness matrices are generally not diagonal and the displacement

constraints are violated in the local coordinate system of the body, but the arbitrary mode shapes are accomodated by the floating axes coordinate systems to within a rigid body displacement without any constraint equations.

4.6 The Dynamics of Deformable Mechanical Systems

4.6.1 The Weak Form of Lagrange's Equation

Lagrange's equation for a deformable body is derived in this section from Cauchy's first equation of motion and the appropriate boundary conditions.

Mathematical rigor is not enforced. The standard assumptions are made: the integrals are Lebesgue, the derivatives are generalized, and all of them exist. All time dependent functions are assumed to be continuous in their highest time derivative that appears in the equations except possibly on a set of measure zero with respect to time. Functions with a spatial dependence are assumed to belong to the proper Sobolev space so that the strain energy is finite and the boundary conditions are satisfied. For those interested, Ciarlet [42] and Oden [43] present a detailed mathematical analysis of the finite element method in their work. For continuum mechanics, Truesdell and Noll [44], and Malvern [35] are recommended for nonlinear materials, and Nečas and Hlaváček [36] for linear elasticity. Much of the early application of the theory of the finite element method and nonlinear continuum mechanics to nonlinear structural dynamics was done by Bathe and

Wilson [51] and Belytschko, et al. [52]. For a short and easily read summary of current methods, including material models, see Hallquist [16].

A Lagrangian viewpoint is used (section 4.3.2) where the particles in a body are identified by their convected coordinates, $\underline{\xi}$, corresponding to their positions in a reference configuration. The displacement vector, measured from ground to the material point $\underline{\xi}$, is denoted $\underline{x}$, and it is decomposed into three orthonormal components in an inertial reference frame. Furthermore, the displacements are functions of the generalized coordinate set $\underline{q}$.

Initial displacements and velocities consistent with the constraints are specified. Only holonomic equality constraints are considered, although the general formulation is readily extended to more general constraints.

$$x_i(\underline{\xi},0) = D_i(\underline{\xi}) \qquad (4.6.1.1)$$

$$\dot{x}_i(\underline{\xi},0) = V_i(\underline{\xi}) \qquad (4.6.1.2)$$

$$\phi_i(\underline{x},0) = 0 \qquad (4.6.1.3)$$

where

D_i and V_i are the i-th component of the initial displacements and velocities, respectively. velocities.

ϕ_i is the i-th constraint function.

Surface tractions, body forces, and concentrated forces (e.g., springs) are applied to the bodies. The motion of the system is governed by Cauchy's first equation of motion [35] subject to the constraint equations and the boundary conditions.

$$\tau_{ij,j} + \rho_o f_i = \rho_o \ddot{x}_i \qquad (4.6.1.4)$$

$$\tau_{ij} n_j = t_i \text{ on } \Gamma_o^t \qquad (4.6.1.5)$$

where

τ_{ij} is the Lagrange stress tensor (the transpose of the first Kirchhoff-Piola stress).

f_i is the i-th component of the body force.

n_j is the j-th component of the outward normal to the boundary of the undeformed body.

t_i is the i-th component of the traction vector per unit of undeformed area.

Γ_o^t is the boundary surface subject to traction.

The weak form, equation 4.6.1.6, is written using equations 4.6.1.4 and 4.6.1.5, where δx_i is an arbitrary function, chosen from the same Sobolev space as x_i, such that $x_i + \delta x_i$ satisfies the constraint conditions.

$$\int_{\Omega_o}(\rho_o\ddot{x}_i - \tau_{ij,j} - \rho_o f_i)\delta x_i d\Omega_o$$

$$+ \int_{\Gamma_o^t}(\tau_{ij}n_j - t_i)\delta x_i d\Gamma_o - \sum_k f^*_{ki}\delta x_i = 0 \qquad (4.6.1.6)$$

where

f^*_{ki} is the i-th component of the k-th applied concentrated force.

By using the divergence theorem and the product differentiation rule, the two terms involving the Lagrange stress are combined.

$$\int_{\Omega_o}(\tau_{ij}\delta x_i)_{,j}d\Omega_o = \int_{\Gamma_o}\tau_{ij}n_j\delta x_i d\Gamma_o \qquad (4.6.1.7)$$

$$\tau_{ij}\delta x_{i,j} = (\tau_{ij}\delta x_i)_{,j} - \tau_{ij,j}\delta x_i \qquad (4.6.1.8)$$

Substitution of equations 4.6.1.7 and 4.6.1.8 into equation 4.6.1.6 gives the principle of virtual work.

$$\delta\pi = \int_{\Omega_o}\rho_o\ddot{x}_i\delta x_i d\Omega_o + \int_{\Omega_o}\tau_{ij}\delta x_{i,j}d\Omega_o$$

$$- \int_{\Omega_o}\rho_o f_i\delta x_i d\Omega_o - \int_{\Gamma_o^t}t_i\delta x_i d\Gamma_o - \sum_k f^*_{ki}\delta x_i = 0 \qquad (4.6.1.9)$$

Note that the continuity requirements on the stress tensor and the first variation changed between equation 4.6.1.6 and equation 4.6.1.9 because of the divergence theorem.

The following identities are useful:

$$x_i = x_i(\underline{q}(t)) \tag{4.6.1.10}$$

$$\dot{x}_i = \frac{\partial x_i}{\partial q_k}\dot{q}_k \tag{4.6.1.11}$$

$$\ddot{x}_i = \frac{\partial^2 x_i}{\partial q_j \partial q_k}\dot{q}_j\dot{q}_k + \frac{\partial x_i}{\partial q_k}\ddot{q}_k \tag{4.6.1.12}$$

$$\delta x_i = \frac{\partial x_i}{\partial q_k}\delta q_k \tag{4.6.1.13}$$

$$\delta x_{i,j} = \frac{\partial x_i}{\partial q_k}\delta q_{k,j} \tag{4.6.1.14}$$

Substitution of the above equations into equation 4.6.1.9 results in an equation that is computationally more useful since it is written in terms of the generalized coordinates.

$$\begin{aligned} &\int_{\Omega_o} \rho_o \left(\frac{\partial^2 x_i}{\partial q_j \partial q_k}\dot{q}_j\dot{q}_k + \frac{\partial x_i}{\partial q_j}\ddot{q}_j\right)\frac{\partial x_i}{\partial q_m}\delta q_m \, d\Omega_o \\ &+ \int_{\Omega_o} \tau_{ij}\frac{\partial x_i}{\partial q_k}\delta q_{k,j} d\Omega_o - \int_{\Omega_o} \rho_o f_i \frac{\partial x_i}{\partial q_k}\delta q_k d\Omega_o \\ &- \int_{\Gamma_o^t} t_i \frac{\partial x_i}{\partial q_k}\delta q_k d\Gamma_o - \sum_m f^*_{mi}\frac{\partial x_i}{\partial q_j}\delta q_k = 0 \end{aligned} \tag{4.6.1.15}$$

The first term is simplified by writing it in terms of the kinetic energy per unit mass.

$$T^* = \frac{1}{2}\dot{x}_i \cdot \dot{x}_i$$
$$= \frac{1}{2}\frac{\partial x_i}{\partial q_j}\cdot\frac{\partial x_i}{\partial q_k}\dot{q}_j\dot{q}_k \qquad (4.6.1.16)$$

Differentiation gives the following identity:

$$\ddot{x}_i \delta x_i = [\frac{d}{dt}(\frac{\partial T^*}{\partial \dot{q}_k}) - \frac{\partial T^*}{\partial q_k}]\delta q_k \qquad (4.6.1.17)$$

Substitution of equation 4.6.1.17 into equation 4.6.1.15, and the introduction of Lagrange multipliers to remove the constraint conditions on $\underline{q}$, gives the weak form of Lagrange's equation.

$$\int_{\Omega_o}\rho_o[\frac{d}{dt}(\frac{\partial T^*}{\partial \dot{q}_k}) - \frac{\partial T^*}{\partial q_k}]\delta q_k d\Omega_o + \int_{\Omega_o}\tau_{ij}\frac{\partial x_i}{\partial q_k}\delta q_{k,j} d\Omega_o$$
$$- \int_{\Omega_o}\rho_o f_i\frac{\partial x_i}{\partial q_k}\delta q_k d\Omega_o - \int_{\Gamma_o^t} t_i\frac{\partial x_i}{\partial q_k}\delta q_k d\Gamma_o \qquad (4.6.1.18)$$
$$- \sum_m f^*_{mi}\frac{\partial x_i}{\partial q_k}\delta q_k - \lambda_j\delta\phi_j - \delta\lambda_j\phi_j = 0$$

where

λ_j is the Lagrange multiplier associated with the j-th constraint equation.

For K generalized coordinates and J constraints, equation 4.6.1.18 is actually the sum of K + J equations, each independently equal to zero, because all of the δq_k and $\delta\lambda_j$ are arbitrary.

4.6.2 The Choice of Generalized Coordinates

The total displacement of a material point from the origin of the inertial reference frame is decomposed into its rigid body displacements and the deformations of the body.

$$x_i = X_i + A_{ij}(\underline{\theta})(r_j + u_j) \qquad (4.6.2.1)$$

where

x_i is the total displacement in the ground frame.

X_i is the rigid body displacement of the body in the inertial reference system.

A_{ij} is the orthonormal rotation transformation from the local frame of the body to the inertial frame.

r_j is the location of the point in its undeformed configuration in the local frame of the body.

u_i is the deformation field in the local frame of the body.

$\underline{\theta}$ is the rigid body rotation measure.

The advantages of this decomposition were discussed in section 4.5.3.

The choice of coordinates for representing the rigid body rotation is not specified in equation 4.6.2.1. From the theoretical standpoint there is no great advantage of one set over another because none of them allow the

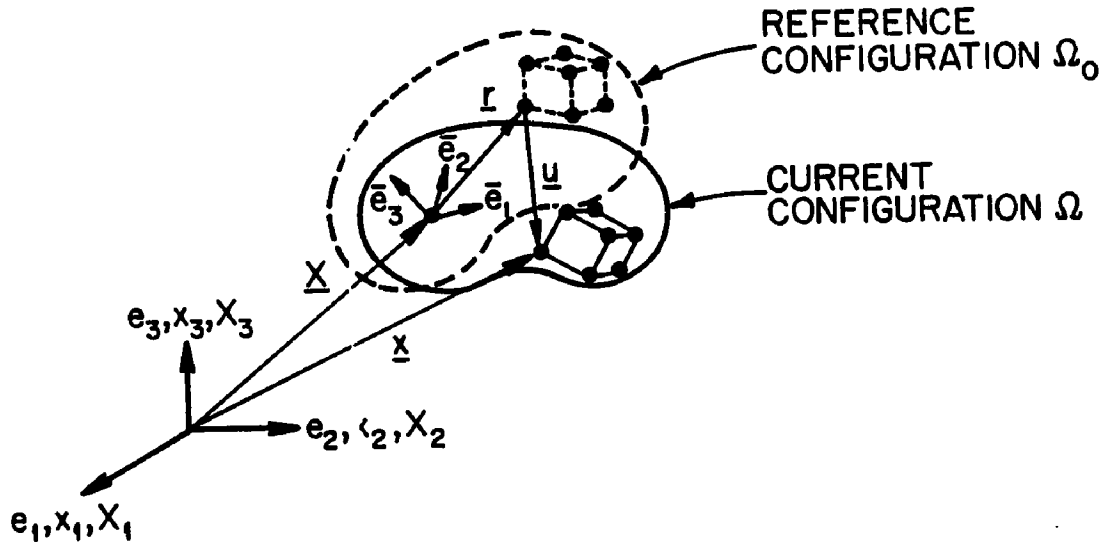


Fig. 4.1. The definition of the generalized coordinates.

treatment of large rotations as vectors. There are tradeoffs, however, between them from the computational standpoint, and they are discussed in the next chapter.

The tangential stiffness matrix is independent of the rigid body motion if both the deformations and the forces acting within the body are measured in the local coordinate system. Although this result is intuitively obvious, its mathematical demonstration is worthwhile. The virtual work term associated with the stresses is rewritten in the local coordinate frame by substituting the following identities:

$$\tau^G_{ij} = A_{im}\tau^L_{mj} \qquad (4.6.2.2)$$

$$\delta x_{i,j} = A_{ip}\delta u_{p,j} \qquad (4.6.2.3)$$

The superscript G and L refer to the ground and the local frame of the body respectively.

$$\begin{aligned}
\delta\pi_\tau &= \int_{\Omega_o} \tau^G_{ij}\delta x_{i,j} d\Omega_o \\
&= \int_{\Omega_o} A_{im}A_{ip}\tau^L_{mj}\delta u_{p,j} d\Omega_o \\
&= \int_{\Omega_o} \tau^L_{ij}\delta u_{i,j} d\Omega_o \\
&= \int_{\Omega_o} \tau^L_{ij}\delta F_{ij} d\Omega_o
\end{aligned} \qquad (4.6.2.4)$$

where

F_{ij} is the deformation gradient, $u_{i,j}$.

Substitution of the approximate deformation field in equation 4.3.2.1 and the subsequent differentiation with respect to the deformations at the nodes gives the tangential stiffness matrix. The internal virtual work is independent of the rigid body motion from equation 4.6.2.4, and therefore the stiffness matrix is also independent.

The Lagrange stress is evaluated in the local coordinate frame of the body for the remainder of this dissertation and the superscripts are therefore dropped from the stress tensor.

A locally attached frame is used to define the local coordinate frame of the body. The Buckens frame is valid for only small deformations, and the Tisserand frame is nonholonomic. Although the principal axes frame is both holonomic and valid for large deformations, as discussed earlier, it does not offer enough in return for the additional constraint equations.

4.6.3 Lagrange's Equation for Quasi-coordinates

Quasi-coordinates are useful for simplifying the expressions that occur in dynamics, but the classical form of Lagrange's equation is valid only for generalized coordinates. A modified version of Lagrange's equation is derived in this section for quasi-coordinates, and a particular set of quasi-coordinates are used in the next section to simplify the expression for the kinetic energy of deformable bodies.

Following Meirovitch [45], the quasi-velocities are defined as linear combinations of the generalized velocities.

$$\chi_i = \Psi_{ji}(\underline{q})\dot{q}_j$$
$$\dot{q}_i = \phi_{ij}(\underline{q})\chi_j \qquad (4.6.3.1)$$

where

χ_i is the i-th quasi-velocity.

q_i is the i-th generalized coordinate.

Ψ_{ji} and ϕ_{ij} are linear transformations that are functions of $\underline{q}$ only.

The kinetic energy (either per unit mass or for the entire body) is expressed as a function of the quasi-velocities and the generalized coordinates.

$$\bar{T}(\underline{\chi},\underline{q}) = T(\dot{\underline{q}},\underline{q}) \qquad (4.6.3.2)$$

Application of the chain rule gives the relationship between the generalized momenta and the canonical momenta of the quasi-coordinates.

$$\begin{aligned} P_{q_i} &= \frac{\partial T}{\partial \dot{q}_i} \\ &= \Psi_{ij}\frac{\partial \bar{T}}{\partial \chi_j} \\ &= \Psi_{ij}\bar{P}_{\chi_j} \end{aligned} \qquad (4.6.3.3)$$

where

P_{q_i} is the generalized momentum of q_i.

$\bar{P}_{\chi_j}$ is the canonical momentum of χ_j.

Repetitive application of the chain rule yields Lagrange's equation for quasi-coordinates.

$$\int_{\Omega_o}\rho_o[\Psi_{ij}\frac{d}{dt}(\frac{\partial\bar{T}^*}{\partial\chi_j}) - \frac{\partial\bar{T}^*}{\partial q_i} + (\frac{\partial\Psi_{ij}}{\partial q_k} - \frac{\partial\Psi_{kj}}{\partial q_i})\phi_{kn}\chi_n\frac{\partial\bar{T}^*}{\partial\chi_j}]\delta q_i d\Omega_o + \int_{\Omega_o}\tau_{ij}\delta F_{ij}d\Omega_o - \int_{\Omega_o}\rho_o f_i\frac{\partial x_i}{\partial q_k}\delta q_k d\Omega_o - \int_{\Gamma_o^t}t_i\frac{\partial x_i}{\partial q_k}\delta q_k d\Gamma_o - \sum_m f^*_{mi}\frac{\partial x_i}{\partial q_k}\delta q_k - \lambda_j\delta\phi_j - \delta\lambda_j\phi_j = 0 \qquad (4.6.3.4)$$

Noting the appearence of the canonical momenta in many of the terms in equation 4.6.3.4, it is rewritten using both the generalized momenta and the canonical momenta. The introduction of the momenta as variables simplifies the evaluation of the Jacobian used in the Gear integration method [10].

$$\int_{\Omega_o}\rho_o[\frac{d}{dt}(P^*_{q_i}) - \frac{\partial\bar{T}^*}{\partial q_i} - \frac{\partial\Psi_{kj}}{\partial q_i}\phi_{kn}\chi_n\bar{P}^*_{\chi_j}]\delta q_i d\Omega_o + \int_{\Omega_o}\tau_{ij}\delta F_{ij}d\Omega_o - \int_{\Omega_o}\rho_o f_i\frac{\partial x_i}{\partial q_k}\delta q_k d\Omega_o - \int_{\Gamma_o^t}t_i\frac{\partial x_i}{\partial q_k}\delta q_k d\Gamma_o - \sum_m f^*_{mi}\frac{\partial x_i}{\partial q_k}\delta q_k - \lambda_j\delta\phi_j - \delta\lambda_j\phi_j = 0 \qquad (4.6.3.5)$$

where

$P^*_{q_i}$ and $\bar{P}^*_{\chi_j}$ are per unit mass.

4.6.4 The Application of Quasi-coordinates

To simplify the expression for the kinetic energy, the angular velocity about the axes of the body and the rigid body translational velocities, rotated into the local frame of the body, are chosen as quasi-velocities.

$$\begin{aligned} \dot{X}_i &= A_{ij}Q_j \\ \dot{\theta}_i &= B_{ij}\omega_j \\ \dot{u}_i &= I_{ij}\dot{u}_j \end{aligned} \qquad (4.6.4.1)$$

where

A_{ij} and B_{ij} are functions of $\underline{\theta}$.

I_{ij} is the identity matrix.

Q_j is the j-th component in the local frame of the body of the rigid body translational velocity.

ω_i is the angular velocity of the body about its i-th local axis.

The velocity of a point, evaluated in an inertial frame and rotated into the local frame of the body, is free of any terms involving the rigid body rotation measures.

$$\dot{x}_i = Q_i + \dot{u}_i + e_{ijk}\omega_j(r_k + u_k) \qquad (4.6.4.2)$$

The derivatives of equation 4.6.4.2 are much simpler than those of equation 4.6.2.4. The simplicity of the expression for the kinetic energy per unit mass is a direct consequence of equation 4.6.4.2.

$$\bar{T}^* = \frac{1}{2}\sum_{i=1}^{3}(Q_i + \dot{u}_i + e_{ijk}\omega_j\,(r_k + u_k))^2 \qquad (4.6.4.3)$$

The equations of motion for the deformation coordinates are derived from equation 4.6.3.4. Note that the resulting equations of motion are identical to the formal application of equation 4.6.1.18 because Ψ_{ij} and ϕ_{ij} are the identity operators for the deformation coordinates.

$$\begin{aligned}
&\int_{\Omega_o}\rho_o(\dot{Q}_i + e_{imn}\omega_m Q_n + \ddot{u}_i \\
&+ 2e_{ijk}\omega_j\dot{u}_k + e_{ijk}\dot{\omega}_j(r_k + u_k) \\
&+ e_{imn}\omega_m e_{njk}\omega_j(r_k + u_k))\delta u_i d\Omega_o \\
&+ \int_{\Omega_o}\tau_{ij}\delta F_{ij}d\Omega_o - \int_{\Omega_o}\rho_o f_j A_{ji}\delta u_i d\Omega_o \qquad (4.6.4.4)\\
&- \int_{\Gamma_o^t}t_j A_{ji}\delta u_i d\Gamma_o - \sum_k f^*_{kj}A_{ji}\delta u_i \\
&- \lambda_j\frac{\partial\phi_j}{\partial u_i}\delta u_i - \delta\lambda_j\phi_j = 0
\end{aligned}$$

The term $\dot{Q}_i + e_{imn}\omega_m Q_n$ is readily shown to equal $A_{ji}\ddot{X}_j$, the rigid body acceleration rotated into the local frame of the body, by using equation 4.6.3.1. The remaining terms in the first integral are recognized as the classical form of the acceleration vector. The applied force terms include the rotation matrix which is consistent with the convention that the applied force components are in the ground frame and the deformation components are in the local frame of the body.

Applying the finite element discretization of equation 4.3.2.1, the displacement field of equation 4.6.2.4 is replaced by its discrete approximation. Note that although the basis functions are introduced at this point there is no restriction on the element type. In Chapter 5 the particular elements used for the demonstration problems are discussed.

$$\begin{aligned} x_i(\underline{\xi}) &= X_i + A_{ij}(\underline{\theta})N_\alpha(\underline{\xi})(r_{\alpha j} + u_{\alpha j}) \\ \delta x_i(\underline{\xi}) &= \delta X_i + \frac{\partial A_{ij}(\underline{\theta})}{\partial \theta_k} N_\alpha(\underline{\xi})(r_{\alpha j} + u_{\alpha j})\delta\theta_k \\ &\quad + A_{ij}(\underline{\theta})N_\alpha \delta u_{\alpha j} \\ \delta u_{i,j}(\underline{\xi}) &= N_{\alpha,j}(\underline{\xi})\delta u_{\alpha i} \end{aligned} \qquad (4.6.4.5)$$

The equation of motion for a particular deformation component i at a node α is obtained by substituting equation 4.6.4.5 into equation 4.6.4.4. Elimination of the integral involving the inertial terms is accomplished by using the definition of the mass matrix, given in equation 4.3.4.4.

$$\begin{aligned} &[\sum_\beta M_{\alpha i \beta t}(\dot{Q}_t + e_{tmn}\omega_m Q_n) + M_{\alpha i \beta t}(\ddot{u}_{\beta t} + 2e_{tjk}\omega_j \dot{u}_{\beta k} \\ &+ e_{tjk}\dot{\omega}_j(r_{\beta k} + u_{\beta k}) + e_{tmn}\omega_m e_{njk}\omega_j(r_{\beta k} + u_{\beta k})) \\ &+ \int_{\Omega_o} \tau_{ij}N_{\alpha,j}d\Omega_o - \int_{\Omega_o}\rho_o f_j A_{ji}N_\alpha d\Omega_o - \int_{\Gamma_o^t} t_j A_{ji}N_\alpha d\Gamma_o \\ &- \sum_k f^*_{kj}A_{ji}N_\alpha - \lambda_j\frac{\partial \phi_j}{\partial u_{\alpha i}}]\delta u_{\alpha i} = 0 \end{aligned} \qquad (4.6.4.6)$$

Noting that $\delta u_{\alpha i}$ is arbitrary, the terms within the square brackets ([,]) must sum to zero.

Unlike the deformation field, the rigid body coordinates are constant over the domain of the body. The derivation of the equations of motion is simplified by introducing the momenta of the bodies as variables and moving all of the terms that are independent of the domain of integration outside of the integrals. Applying equations 4.6.3.3 and 4.6.3.5, the equations of motion are written in a form simple enough for an explicit Jacobian.

$$\bar{P}_{Q_i} - \int_{\Omega_o} \rho_o (Q_i + \dot{u}_i + e_{ikn}\omega_k (r_n + u_n)) d\Omega_o = 0 \qquad (4.6.4.7)$$

$$\bar{P}_{\omega_j} - \int_{\Omega_o} \rho_o e_{ijk}(r_k + u_k)(Q_i + \dot{u}_i + e_{imn}\omega_m(r_n + u_n)) d\Omega_o = 0 \qquad (4.6.4.8)$$

$$P_{X_i} - A_{ij}\bar{P}_{Q_j} = 0 \qquad (4.6.4.9)$$

$$\frac{d}{dt}(P_{X_i}) - \int_{\Omega_o} \rho_o f_i d\Omega_o - \int_{\Gamma_o^t} t_i d\Gamma_o - \sum_k f^*_{ki} - \lambda_j \frac{\partial \phi_j}{\partial X_i} = 0 \qquad (4.6.4.10)$$

$$\frac{d}{dt}(\bar{P}_{\omega_i}) + e_{ijk}\omega_j \bar{P}_{\omega_k} + e_{ijk}Q_j\bar{P}_{Q_k} - B_{ji}\left(\frac{\partial A_{nk}}{\partial \theta_j}\int_{\Omega_o} \rho_o f_n (r_k + u_k) d\Omega_o + \int_{\Gamma_o^t} t_n (r_k + u_k) d\Gamma_o + \sum_m f^*_{mn}(r_k + u_k)\right) - B_{ji}\lambda_k \frac{\partial \phi_k}{\partial \theta_j} = 0 \qquad (4.6.4.11)$$

Equations 4.6.4.9, 4.6.4.10, and 4.6.4.11 are the classical equations of motion written in a noninertial coordinate system [45], which in this case is the local frame of the body. Substituting the finite element deformation field approximation and equation 4.3.4.4, as was done in equation 4.6.4.6, gives the discrete forms of equation 4.6.4.7, 4.64.8, 4.6.4.10, and 4.6.4.11.

$$\bar{P}_{Q_i} - \sum_{\alpha,\beta} M_{\alpha i \beta j} Q_j - \sum_{\alpha} M_{\alpha i \beta j}(\dot{u}_{\beta j} + e_{jkn}\omega_k(r_{\beta n} + u_{\beta n})) = 0 \tag{4.6.4.12}$$

$$\bar{P}_{\omega_j} - e_{ijk}(r_{\alpha k} + u_{\alpha k})\left[\sum_{\beta} M_{\alpha i \beta n} Q_n + M_{\alpha i \beta n}(\dot{u}_{\beta n} + e_{nmp}\,\omega_m(r_{\beta p} + u_{\beta p}))\right] = 0 \tag{4.6.4.13}$$

$$\frac{d}{dt}(P_{X_i}) - \sum_{\alpha\Omega_o}\int \rho_o f_i N_\alpha d\Omega_o - \sum_{\alpha\Gamma_o^t}\int t_i N_\alpha d\Gamma_o - \sum_k f^*_{ki} - \lambda_j \frac{\partial \phi_j}{\partial X_i} = 0 \tag{4.6.4.14}$$

$$\frac{d}{dt}(\bar{P}_{\omega_i}) + e_{ijk}\omega_j \bar{P}_{\omega_k} + e_{ijk} Q_j \bar{P}_{Q_k} - B_{ji}\left\{\frac{\partial A_{nk}}{\partial \theta_j}(r_{\alpha k} + u_{\alpha k})\left[\int_{\Omega_o} \rho_o f_n N_\alpha d\Omega_o + \int_{\Gamma_o^t} t_n N_\alpha d\Gamma_o\right] + \sum_m f^*_{mn}\frac{\partial A_{nk}}{\partial \theta_j}(r_{\alpha(m)k} + u_{\alpha(m)k}) + \lambda_k \frac{\partial \phi_k}{\partial \theta_j}\right\} = 0 \tag{4.6.4.15}$$

where

$\alpha(m)$ is the node on which the m-th force acts.

Equations 4.6.4.1, 4.6.4.6, 4.6.4.9, 4.6.4.12, 4.6.4.13, 4.6.4.14, and 4.6.4.15 define the complete set of equations for a deformable body. While they may appear complicated, remember that they are written out in complete detail. Evaluating both the residuals and the Jacobian is not difficult: the inertial terms are, at most, third degree polynomials, and the remaining terms are routinely evaluated in structural analysis programs.

The equations describing mechanical systems are created by applying the complete set of equations for a deformable body to each body in the system. Bodies are coupled by the constraint equations and the applied forces. Nothing in the development of the equations of motion limits the constraint and force functions to acting on, or being a function of, a single body.

The equations developed above take advantage of the ability of the Gear integration method to solve systems of simultaneous differential and algebraic equations by introducing the momenta variables in order to simplify the evaluation of the Jacobian. As argued in Chapter 3, physically stiff systems are described by stiff differential equations, and a stiff integration method, such as the Gear method, is necessary, but there are important classes of problems where a nonstiff integration method is beneficial. For some problems only the transient part of the solution is of any interest, e.g., projectile impact simulation, and the constant reformulation of the tangential stiffness

matrix is prohibitively expensive. Other systems, such as solar power arrays on satellites, have low-frequency structural modes, deform in a nonlinear manner, and are nonstiff in both the mathematical and physical senses. By eliminating the momenta variables and differentiating the the constraint equations twice with respect to time, the equations of motion are a set of linear algebraic equations for the generalized accelerations, the quasi-accelerations, and the Lagrange multipliers. The accelerations are integrated into velocities, the quasi-velocities are transformed into generalized velocities by using equation 4.6.3.1, and the generalized velocities are integrated into generalized displacements.

For each $u_{\alpha i}$:

$$
\begin{aligned}
&[\sum_{\beta} M_{\alpha i \beta t}]\dot{Q}_t + [M_{\alpha i \beta t}]\ddot{u}_{\beta t} \\
&+ [M_{\alpha i \beta t} e_{tjk}(r_{\beta k} + u_{\beta k})]\dot{\omega}_j - [\frac{\partial \phi_j}{\partial u_{\alpha i}}]\lambda_j \\
&= -[\sum_{\beta} M_{\alpha i \beta t} e_{tmn}\omega_m Q_n + M_{\alpha i \beta t}(2e_{tjk}\omega_j \dot{u}_{\beta k} \\
&\quad + e_{tmn}\omega_m e_{njk}\omega_j (r_{\beta k} + u_{\beta k})] \qquad (4.6.4.16) \\
&- \{\int_{\Omega_o} \tau_{ij} N_{\alpha,j} d\Omega_o - A_{ji}(\int_{\Omega_o} \rho_o f_j N_\alpha d\Omega_o \\
&\quad + \int_{\Gamma_o^t} t_j N_\alpha d\Gamma_o) - A_{ji} \sum_k f^*_{kj} N_\alpha\}
\end{aligned}
$$

For each Q_i:

$$\begin{aligned}
&[\sum_{\alpha,\beta} M_{\alpha i \beta t}]\ddot{Q}_{\beta t} + [\sum_{\alpha} M_{\alpha i \beta t} e_{tjk}(r_{\beta k} + u_{\beta k})]\dot{\omega}_j \\
&\quad + [\sum_{\alpha} M_{\alpha i \beta t}]\ddot{u}_{\beta t} - [\frac{\partial \phi_j}{\partial x_i}]\lambda_j \\
&= -[\sum_{\alpha} M_{\alpha i \beta t} e_{tkn}\omega_k \dot{u}_{\beta n} + e_{ikn}\omega_k \bar{P} Q_n] \\
&+ A_{ji}\{\sum_{\alpha}(\int_{\Omega_o} \rho_o f_j N_\alpha d\Omega_o + \int_{\Gamma_o^t} t_j N_\alpha d\Gamma_o) + \sum_k f^*_{kj}\}
\end{aligned} \tag{4.6.4.17}$$

For each ω_i:

$$\begin{aligned}
&[\sum_{\beta} e_{jik}(r_{\alpha k} + u_{\alpha k}) M_{\alpha j \beta t}]\ddot{Q}_t \\
&+ [e_{jip}(r_{\alpha p} + u_{\alpha p}) M_{\alpha j \beta t} e_{tnk}(r_{\beta k} + u_{\beta k})]\dot{\omega}_n \\
&+ [e_{jik}(r_{\alpha k} + u_{\alpha k}) M_{\alpha j \beta t}]\ddot{u}_{\beta t} - [B_{ji}\frac{\partial \phi_k}{\partial \theta_j}]\lambda_k \\
&= -[e_{jik}(r_{\alpha k} + u_{\alpha k}) M_{\alpha j \beta t} e_{tmp}\omega_m \dot{u}_{\beta p} \\
&\quad + e_{jik}\dot{u}_{\alpha k}(\sum_{\beta} M_{\alpha j \beta n} Q_n + M_{\alpha j \beta n}(\dot{u}_{\beta n} \\
&+ e_{nmp}\omega_m(r_{\beta p} + u_{\beta p}))) + e_{ijk}\omega_j \bar{P}\omega_k + e_{ijk} Q_j \bar{P} Q_k] \\
&\quad + B_{ji}\{\frac{\partial A_{nk}}{\partial \theta_j}(r_{\alpha k} + u_{\alpha k})(\int_{\Omega_o} \rho_o f_n N_\alpha d\Omega_o \\
&+ \int_{\Gamma_o^t} t_n N_\alpha d\Gamma_o) + \sum_m f^*_{mn}\frac{\partial A_{nk}}{\partial \theta_j}(r_{\alpha(m)k} + u_{\alpha(m)k})\}
\end{aligned} \tag{4.6.4.18}$$

For each ϕ_s:

$$\begin{aligned}
&[\frac{\partial \phi_s}{\partial q_i}\phi_{ij}]\dot{\chi}_j = -(\frac{\partial \phi_s}{\partial q_i}\frac{\partial \phi_{in}}{\partial q_k}\phi_{km} \\
&+ \frac{\partial^2 \phi_s}{\partial q_j \partial q_k}\phi_{jm}\phi_{kn})\chi_m \chi_n - 2\frac{\partial^2 \phi_s}{\partial q_k \partial t}\phi_{km}\chi_m - \frac{\partial^2 \phi_s}{\partial t^2}
\end{aligned} \tag{4.6.4.19}$$

The terms enclosed in the square brackets ([,]) on the left hand side of the the above equations are the coefficients of the linear equations. Although the symmetry of the coefficient matrix is not obvious, it is easily demonstrated by using the symmetry properties of the mass matrix. Each body, if the rigid body equations and variables occur last, is represented by a submatrix with a diagonal band and a border six wide on the right edge and the bottom edge. If the penalty method is used instead of Lagrange multipliers to enforce the constraints, then the coefficient submatrix for each body is completely decoupled from all of the other bodies. The terms on the right hand side enclosed in the square brackets are the remaining inertial terms. Inside the curly brackets ({,}) are the structural and applied force terms.

Using equations 4.6.4.16 through 4.6.4.19 and the penalty method, an explicit dynamics code, such as DYNA3D, could without serious difficulty be modified to use the coordinate decomposition of equation 4.6.2.1. Although such an extension appears formidable, DYNA3D uses a lumped mass matrix which reduces many of the summations to single terms. NIKE2D, which uses the unconditionally stable, one-step Newmark-β integration method, could also be modified to use equations 4.6.4.16 through 4.6.4.19.

4.7 Specialization to Linear Elasticity

Linear coordinate reduction methods, discussed in section 4.4, are used in this section to develop a reduced set of equations for linearly elastic bodies undergoing only small, linear deformations. Requirements on the mode shapes are less restrictive than the requirements for the Tisserand or Bucken's frames, but there are no simplifications to the equations of motion.

Substitution of equation 4.4.2 into equation 4.6.4.5 gives the displacement of a point in terms of the rigid body displacements and the modal amplitudes.

$$
\begin{aligned}
x_i &= X_i + A_{ij} N_\alpha (r_{\alpha j} + H_{\alpha jk}\eta_k) \\
\delta x_i &= \delta X_i + \frac{\partial A_{ij}}{\partial \theta_k} N_\alpha (r_{\alpha j} + H_{\alpha jk}\eta_k)\delta\theta_k \\
&\quad + A_{ij} N_\alpha H_{\alpha jk}\delta\eta_k \\
\delta u_{i,j} &= N_{\alpha,j} H_{\alpha ik}\delta\eta_k
\end{aligned}
\qquad (4.7.1)
$$

The equations of motion are derived by following the procedures of section 4.6. A reduced stiffness matrix, defined in equation 4.4.1.3, replaces the stress term because, by definition, the stiffness matrix is constant. Also included is a reduced damping matrix, also defined in equation 4.4.13.

The equations of motion for the modes closely follow equation 4.6.4.6 in form.

$$H_{\alpha ic}\sum_{\beta}M_{\alpha i\beta t}(\dot{Q}_t + e_{tmn}\omega_m Q_n)$$
$$+ H_{\alpha ic}M_{\alpha i\beta t}(H_{\beta tb}\ddot{\eta}_b$$
$$+ 2e_{tjk}\omega_j H_{\beta kb}\dot{\eta}_b + e_{tjk}\dot{\omega}_k(r_{\beta k} + H_{\beta kb}\eta_b)$$
$$+ e_{tmn}\omega_m e_{njk}\omega_j(r_{\beta k} + H_{\beta kb}\eta_b)) + C^R_{cb}\dot{\eta}_b + K^R_{cb}\eta_b \qquad (4.7.2)$$
$$- A_{ji}H_{\alpha ic}\left(\int_{\Omega_o} f_j N_\alpha d\Omega_o + \int_{\Gamma^t_o} t_j N_\alpha d\Gamma_o\right)$$
$$- \sum_k f^*_{kj}A_{ji}N_\alpha H_{\alpha ic} - \lambda_j\frac{\partial\phi_j}{\partial\eta_c} = 0$$

Equations 4.6.4.12, 4.6.4.13, and 4.6.4.15 are replaced in a similar manner. Equations 4.6.4.9 and 4.6.4.14 remain unchanged.

$$\bar{P}_{Q_i} - \sum_{\alpha,\beta}M_{\alpha i\beta j}Q_j$$
$$- \sum_{\alpha}M_{\alpha i\beta j}(H_{\beta jb}\dot{\eta}_b + e_{jkn}\omega_k(r_{\beta n} + H_{\beta nb}\eta_b)) = 0 \qquad (4.7.3)$$

$$\bar{P}_{\omega_j} - e_{ijk}(r_{\alpha k} + H_{\alpha kc}\eta_c)\left[\sum_{\beta}M_{\alpha i\beta n}Q_n\right.$$
$$\left. + M_{\alpha i\beta n}(H_{\beta nb}\dot{\eta}_b + e_{nmp}\omega_m(r_{\beta p} + H_{\beta pb}\eta_b))\right] = 0 \qquad (4.7.4)$$

$$\frac{d}{dt}(\bar{P}_{\omega_i}) + e_{ijk}\omega_j\bar{P}_{\omega_k} + e_{ijk}Q_j\bar{P}_{Q_k}$$
$$- B_{ji}\left\{\frac{\partial A_{nk}}{\partial\theta_j}(r_{\alpha k} + H_{\alpha kc}\eta_c)\left[\int_{\Omega_o}\rho_o f_n N_\alpha d\Omega_o\right.\right.$$
$$\left.\left. + \int_{\Gamma^t_o} t_n N_\alpha d\Gamma_o\right] + \sum_m f^*_{mn}\frac{\partial A_{nk}}{\partial\theta_j}(r_{\alpha k} + H_{\alpha kc}\eta_c)\right. \qquad (4.7.5)$$
$$\left. + \lambda_k\frac{\partial\phi_k}{\partial\theta_j}\right\} = 0$$

Equations comparable to equations 4.6.4.16 through 4.6.4.18 are readily derived for modal coordinates. A nonstiff integration method, however, is not recommended for the reasons discussed in Chapter 3. The following equations are included for completeness.

For each η_c:

$$
\begin{aligned}
&[H_{\alpha ic}\sum_{\beta} M_{\alpha i\beta t}]\dot{Q}_t + [H_{\alpha ic}M_{\alpha i\beta t}e_{tjk}(r_{\beta k} + H_{\beta kb}\eta_b)]\dot{\omega}_j \\
&\qquad + [H_{\alpha ic}M_{\alpha i\beta t}H_{\beta tb}]\ddot{\eta}_b - [\frac{\partial \phi_j}{\partial \eta_c}]\lambda_j \\
&= -[H_{\alpha ic}\sum_{\beta} M_{\alpha i\beta t}e_{tmn}Q_n + H_{\alpha ic}M_{\alpha i\beta t}(2e_{tjk}\omega_j H_{\beta kb}\dot{\eta}_b \\
&\qquad + e_{tmn}\omega_m e_{njk}\omega_j(r_{\beta k} + H_{\beta kb}\eta_b)] \\
&\qquad - \{C^R_{cb}\dot{\eta}_b + K^R_{cb}\eta_b - A_{ji}H_{\alpha ic}(\int_{\Omega_o}\rho_o f_j N_\alpha d\Omega_o \\
&\qquad + \int_{\Gamma^t_o} t_j N_\alpha d\Gamma_o) - \sum_k f^*_{kj}A_{ji}N_\alpha H_{\alpha ic}\}
\end{aligned}
\tag{4.7.6}
$$

For each Q_i:

$$
\begin{aligned}
&[\sum_{\alpha,\beta} M_{\alpha i\beta t}]\dot{Q}_t + [\sum_{\alpha} M_{\alpha i\beta t}e_{tjk}(r_{\beta k} + H_{\beta kb}\eta_b)]\dot{\omega}_j \\
&\qquad + [\sum_{\alpha} M_{\alpha i\beta t}H_{\beta tb}]\ddot{\eta}_b - [\frac{\partial \phi_j}{\partial X_i}]\lambda_j \\
&= -[\sum_{\alpha} M_{\alpha i\beta t}e_{tkn}\omega_k H_{\beta nb}\dot{\eta}_b + e_{ijk}\omega_k \bar{P}Q_n] \\
&\qquad + A_{ji}\{\sum_{\alpha}(\int_{\Omega_o}\rho_o f_j N_\alpha d\Omega_o + \int_{\Gamma^t_o} t_j N_\alpha d\Gamma_o) + \sum_k f^*_{kj}\}
\end{aligned}
\tag{4.7.7}
$$

For each ω_i:

$$
\begin{aligned}
&[\sum_{\beta} e_{jik}(r_{\alpha k} + H_{\alpha kc}\eta_c)M_{\alpha j\beta t}]\dot{Q}_t \\
&+ [e_{jik}(r_{\alpha k} + H_{\alpha kc}\eta_c)M_{\alpha j\beta t}e_{tmp}(r_{\beta p} + H_{\beta pb}\eta_b)]\dot{\omega}_n \\
&+ [e_{jik}(r_{\alpha k} + H_{\alpha kc}\eta_c)M_{\alpha j\beta t}H_{\beta tb}]\ddot{\eta}_b - [B_{ji}\frac{\partial \phi_k}{\partial \theta_j}]\lambda_k \\
&= - [e_{jik}(r_{\alpha k} + H_{\alpha kc}\eta_c)\, M_{\alpha j\beta t}e_{tmp}\omega_m H_{\beta pb}\dot{\eta}_b \\
&\quad + e_{jik}H_{\alpha kc}\dot{\eta}_c(\sum_{\beta} M_{\alpha j\beta n}Q_n \\
&\quad + M_{\alpha j\beta n}(H_{\beta nb}\dot{\eta}_b + e_{nmp}\omega_m(r_{\beta p} + H_{\beta pb}\eta_b))) \\
&\quad + e_{ijk}\omega_j\bar{P}_{\omega_{ok}} + e_{ijk}Q_k\bar{P}_{Q_k}] \\
&\quad + \{B_{ji}(\frac{\partial A_{nk}}{\partial \theta_j}(r_{\alpha k} + H_{\alpha kc}\eta_c)(\int_{\Omega_o}\rho_o f_n N_\alpha d\Omega_o \\
&\quad + \int_{\Gamma_o^t} t_n N_\alpha d\Gamma_o) + \sum_{m} f^*_{mn}\frac{\partial A_{nk}}{\partial \theta_j}(r_{\alpha(m)k} + H_{\alpha(m)kc}\eta_c))\}
\end{aligned}
\tag{4.7.8}
$$

4.8 Rigid Body Dynamics

Following Orlandea [10], a center of mass, principal axes coordinate system is used for each body. The displacement of a point relative to ground is a simplification of equation 4.6.2.1.

$$x_i = X_i + A_{ij}r_j \tag{4.8.1}$$

Only the generalized angular velocity is replaced with quasi-velocities because the translational momenta expressions are very simple in the ground frame.

$$P_{X_i} = M\dot{X}_i \tag{4.8.2}$$

$$\bar{P}_{\omega_i} = J_i\omega_i \quad \text{no sum on } i \tag{4.8.3}$$

where

M is the mass of the body.

J_i is the diagonal inertia tensor.

The substitution of equations 4.8.2 and 4.8.3 into equations 4.6.4.10 and 4.6.4.11 gives the equations of motion for a rigid body.

$$M\ddot{X}_i - \int_{\Omega_o}\rho_o f_i d\Omega_o - \int_{\Gamma_o^t} t_i d\Gamma_o - \sum_k f^*_{ki} - \lambda_j\frac{\partial\phi_j}{\partial X_i} = 0 \tag{4.8.4}$$

$$\begin{aligned} &J_i\dot{\omega}_i + \sum_k e_{ijk}\omega_j J_k\omega_k \\ &- B_{ji}\Big(\int_{\Omega_o}\rho_o f_n\frac{\partial A_{nk}}{\partial\theta_j}r_k d\Omega_o + \int_{\Gamma_o^t} t_n\frac{\partial A_{nk}}{\partial\theta_j}r_k d\Gamma_o \\ &+ \sum_k f^*_{kn}\frac{\partial A_{nm}}{\partial\theta_j}r_{\alpha(k)m} + \lambda_k\frac{\partial\phi_k}{\partial\theta_j}\Big) - \sum_k T_{ki} = 0 \end{aligned} \tag{4.8.5}$$

where there is no summing on i

T_{ki} is the i-th component of an applied torque

in the local frame of the body.

The equations are suitable for either stiff or nonstiff integration methods.

4.9 Forces and Constraints

The equations for some of the most commonly used forces and constraints are developed in this section. Higher pair constraint equations will not be developed; the penalty method is better suited for modelling higher pair contact. Readers interested in the area are referred to the theoretical manual for DYNA3D [16], where Hallquist explains the application of the penalty method for surface contact, or, for a more a more detailed and theoretical development, read Kikuchi and Oden [46]. Surface tractions are not discussed since they are discussed in great detail in most of the standard references on the finite element method [32], [37]. Instead, concentrated forces, follower forces, gravity, and lower pair joints are discussed because of their usefulness in describing mechanical systems.

The expressions developed in this section are written as functions of the rigid body coordinates and the deformations at the nodes. To use them with a linear coordinate reduction method, equation 4.4.2 need only be substituted in the appropriate manner. Generalization of the expressions to arbitrary points in the body, instead of limiting them to the nodes, is also possible by using equation 4.3.2.1.

4.9.1 Notation and Common Expressions

To differentiate between the different bodies of a system, their variables are superscripted with letters. For example, $u^K_{\alpha j}$ refers to $u_{\alpha j}$ in the Kth body. Capital letters are used to avoid confusion with the Greek letters denoting nodes and the lower case letters denoting the components of the vectors and tensors. The summing convention does not apply to superscripts.

Revolute joints are associated not only with a location, but also with an axis of rotation, and concentrated forces have both a point of application and a direction. A local orthonormal coordinate system is associated with each node used in defining a joint or force to define the necessary directions. The basis of the system is $\varepsilon^K_{\alpha ij}$, where i refers to the i-th component in the local frame of the body of the j-th basis vector. By definition, $\varepsilon^K_{\alpha ij}$ defines a rotation matrix from the node α^K orthonormal system to the local system of the body.

For rigid bodies $\varepsilon^K_{\alpha ij}$ is constant, but for deformable bodies it is a function of the current configuration. One way to define an orthonormal system uses three neighboring non-collinear nodes, which are labelled α, β, and θ.

$$\varepsilon^K_{\alpha i 3} = \frac{u^K_{\beta i} - u^K_{\alpha i}}{\sqrt{(u^K_{\beta j} - u^K_{\alpha j})^2}}$$

$$\tilde{\varepsilon}^K_{\alpha i 1} = u^K_{\theta i} - u^K_{\alpha i} \qquad (4.9.1.1)$$

$$\varepsilon^K_{\alpha i 2} = \frac{e_{ijk}\varepsilon^K_{\alpha j 3}\tilde{\varepsilon}^K_{\alpha k 1}}{\sqrt{(e_{ijk}\varepsilon^K_{\alpha j 3}\tilde{\varepsilon}^K_{\alpha k 1})^2}}$$

$$\varepsilon^K_{\alpha i 1} = e_{ijk}\varepsilon^K_{\alpha j 2}\varepsilon^K_{\alpha k 3}$$

where there is no summing on i or a.

The mesh definition may preclude a choice of nodes that results in a set of axes that are aligned with the desired directions of the forces or constraints, but the directions can be specified in terms of their projections on the basis $\varepsilon^K_{\alpha i j}$.

$$\zeta^K_{\alpha ij} = \varepsilon^K_{\alpha im} \bar{\varepsilon}^K_{\alpha nm} \nu^K_{\alpha nj} \qquad (4.9.1.2)$$

where

there is no summing on α.

$\zeta^K_{\alpha ij}$ is an orthonormal set of vectors aligned with the force or joint directions in the current local frame of the body.

$\bar{\varepsilon}^K_{\alpha mn}$ is the transformation defined in equation equation 4.9.1.1 for the original configuration.

$\nu^K_{\alpha mn}$ is an orthonormal set of vectors aligned with the force or joint directions in the local frame of the undeformed body.

By using $\zeta^K_{\alpha ij}$ the axis of rotation or the direction of a force remains constant with respect to the material in a neighborhood of node α^k regardless of the global deformations of the body. For simplicity and clarity, the unit vectors $\zeta^K_{\alpha ij}$, for j equals one, two, and three, are called $\hat{x}^K$, $\hat{y}^K$, and $\hat{z}^K$ respectively.

4.9.2 Forces

The most common force element in a mechanical system is one that acts between two bodies along the line connecting the two application points. Springs and hydraulic cylinders are two common examples. By convention

a positive force acts to separate the bodies, and a negative force brings them together. The components of a force are determined by taking the dot product of the force vector and the ground axes.

$$\Delta x_i = x^I_{\alpha i} - x^J_{\beta i} \qquad (4.9.2.1)$$

$$f^*_k - \text{function}(\underline{q},\dot{\underline{q}},t) = 0 \qquad (4.9.2.2)$$

$$L - \sqrt{\Delta x_i \cdot \Delta x_i} = 0 \qquad (4.9.2.3)$$

$$f^*_{ki} - \frac{\Delta x_i}{L} f^*_k = 0 \qquad (4.9.2.4)$$

$$f^{*I}_{ki} = f^*_{ki} \quad f^{*J}_{ki} = - f^*_{ki} \qquad (4.9.2.5)$$

where

Δx_i are the displacement components between application points.

f^*_k is the magnitude of the force.

L is the magnitude of the displacement.

f^*_{ki} are the force components of the k-th force.

A follower force acts on a single body. Its direction is usually determined by the orientation of a small neighborhood of the material. Applying equation 4.9.1.2, the force components of a follower force are readily calculated.

$$f^*_{ki} = f^*_k A_{ij} \zeta^I_{\alpha j 3} \tag{4.9.2.6}$$

where

f^*_k is directed along $\hat{z}^I$.

A direct application of the weak form for a body force gives the expressions for the gravitational force.

$$\int_{\Omega_o} \rho_o g_i \delta x_i d\Omega_o = \int_{\Omega_o} \rho_o g_i (\delta X_i + \frac{\partial A_{ij}}{\partial \theta_k} N_\alpha (r_{\alpha j} + u_{\alpha j}) \delta\theta_k + A_{ij} N_\alpha \delta u_j)\, d\Omega_o = 0 \tag{4.9.2.7}$$

where

g_i are the gravitational components.

By using the definition of the mass matrix and noting that the sum of the basis function is unity, explicit force expressions are derived.

$$\begin{aligned} f^{*g}_i &= \sum_{\alpha,\beta} M_{\alpha i \beta j} g_j \\ T^g_i &= B_{ji} \sum_\beta M_{\alpha n \beta t} g_t \frac{\partial A_{nk}}{\partial \theta_j} (r_{\alpha k} + u_{\alpha k}) \\ f^{*g}_{\alpha i} &= A_{ji} \sum_\beta M_{\alpha j \beta k} g_k \end{aligned} \tag{4.9.2.7}$$

where

f^{*g}_i is the gravitational force associated with X_i.

T^g_i is the gravitational force associated with ω_i.

$f^{*g}_{\alpha i}$ is the gravitational force associated with $u_{\alpha i}$.

4.9.3 Joints

Holonomic constraints are functions of time and the generalized coordinates of the system only, and they are usually formulated as algebraic equations, although for some cases, such as a constant velocity joint, they are more readily formulated as first order differential equations. The Gear integration method solves systems of differential and algebraic equations so there is considerable flexibility in how the constraint equations are implemented in the program, but nonstiff methods generally require the constraints in second order form, as in equation 4.6.4.19.

Numerical methods are not exact, and therefore the constraints are subject to two types of errors. First, since the solution of the equations is approximated as a polynomial, the constraints are only approximately satisfied between time integration points. Second, a constraint is only satisfied for the order of the equation in which it is posed. Consider, for example, a spherical joint defined by three algebraic equations. While the Gear integration method ensures that the displacements between the two nodes defining the joint are zero at each timestep, there is a small relative velocity and a somewhat larger relative acceleration between them. Conversely, implementing the joint as second order differential equations with a nonstiff integration method results in no relative acceleration, but the system slowly drifts apart.

Different methods are available to enforce the constraints. The penalty method is popular for inequality constraints, but it is not used in mechanisms programs for joint constraints because it introduces unwanted high-frequency components into the solution. Imposing the constraint separately in its zeroth, first, and second order forms to eliminate the second type of error discussed above is not attractive. First of all, the constraints are redundant and may lock up the system, and secondly, only the displacement errors have any practical effect on the calculation so that if the integration method can handle algebraic constraints, then imposing the higher order forms is not necessary. There are three popular approaches for eliminating the displacement errors with nonstiff integration methods:

1) Perform an initial conditions iteration (section 4.10) whenever the displacement errors are unacceptably large.
2) Coordinate partitioning [14] numerically chooses independent and dependent coordinates by applying Gauss-Jordan elimination with full pivoting to the constraint equations. Sparse matrix operations are performed to generate a reduced set of equations of motion for the independent coordinates. At every iteration in the corrector

loop, the reduced equation set is solved and the dependent coordinates are iteratively determined from the constraint equations.

3) Use equation 3.2.2.1 with η_i replaced by ϕ_i, and choose ξ_i for critical damping and ω_i large enough to keep the displacement error acceptable small [58]. Unsurprisingly, high frequencies are introduced into the solution for large values of ω_i.

The joints defined in this section act between two nodes, and therefore the stress distributions are too high locally. For many problems St. Venant's principle [36] is an adequate rationalization for using such simple constraints. When a better approximation is necessary, a weighted average of the variables over several nodes is used to smear the reaction forces.

$$\begin{aligned} u_i &= W_\alpha u_{\alpha i} \\ r_i &= W_\alpha r_{\alpha i} \\ \zeta_{ij} &= W_\alpha \zeta_{\alpha ij} \end{aligned} \qquad (4.9.3.1)$$

where

W_α is a weighting function, but not necessarily the basis function N_α.

The reaction force of joint j on a generalized coordinate, q_k, is $\lambda_j \frac{\partial \phi_j}{\partial q_k}$ (no summing on j). Replacing a single node with a combination of nodes, as in equation 4.9.3.1, therefore distributes the reaction force over several nodes.

A spherical joint is the simplest joint to visualize. Two nodes are superimposed and their respective bodies are free to rotate. Three equations are necessary to superimpose the nodes in three dimensional space.

$$\phi_i^s = x_{\alpha i}^I - x_{\beta i}^J = 0 \qquad (4.9.3.2)$$

where

ϕ_i^s is the i-th spherical joint constraint.

Revolutes are similar to spherical joints, but the bodies are free to rotate about only a single specified axis. Two constraints to eliminate the two rotational degrees of freedom are necessary in addition to the three spherical constraints. Constraining $\hat{z}^I$ and $\hat{x}^J$, and $\hat{z}^J$ and $\hat{x}^I$ to be orthogonal is one approach.

$$\phi_i^R = A_{rj}^I \zeta_{\alpha jk}^I A_{rm}^J \zeta_{\beta mn}^J = 0 \quad (k,n) = (3,1),(1,3) \qquad (4.9.3.3)$$

where

ϕ_i^R is the i-th revolute joint constraint.

Setting the projections of $\hat{z}^I$ and $\hat{z}^J$ equal on the X_1 and X_2 ground axes is computationally simpler, but the the simpler constraints are singular when the axis of rotation is lined up with either the X_1 or X_2 axes.

$$\phi_i^R = A_{ij}^I \zeta_{\alpha jk}^I - A_{ij}^J \zeta_{\beta jk}^J = 0 \quad k=3, \; i=1,2 \qquad (4.9.3.4)$$

A cylindrical joint allows both rotation and translation along a single axis, therefore two rotational and two translational constraints are needed. Either equation 4.9.3.3 or 4.9.3.4 is suitable for constraining the two rotational degrees of freedom. The translational degrees of freedom are removed by constraining the displacement between the nodes to zero in the $\hat{x}^I$ and $\hat{y}^I$ directions.

$$\phi_i^c = A_{kj}^I \zeta_{\alpha ji}^I \phi_k^s = 0 \qquad (4.9.3.5)$$

where

ϕ_i^c is the i-th cylindrical joint constraint.

Translational joints co-align the axes of each node, and allow translation only along the $\hat{z}$ axis. Five constraint equations are needed, namely, the four constraints that define a cylindrical joint plus an additional one to remove the last rotational degree of freedom. One approach is to constrain $\hat{x}^I$ and $\hat{y}^J$ to be orthogonal by using equation 4.9.3.3 and setting (k,n) to (1,2). Another approach with more symmetry and a lower

computational cost uses equation 4.9.3.4 for the two cylindrical joint angular constraints, and the last angular constraint is equation 4.9.3.4 with (i,k) equal to (1,1). The two displacement constraints make use of the co-alignment of the local axes.

$$\phi_i^T = A_{kj}^I \zeta_{\alpha ji}^I x_{\alpha k}^I - A_{kj}^J \zeta_{\beta ji}^J x_{\beta k}^J = 0 \qquad (4.9.3.6)$$

where there is no summing on α and β, and

ϕ_i^T is the i-th translational constraint, and i=1,2.

Constraining the displacement along the $\hat{z}$ axis of cylindrical and translational joints as a function of time is frequently useful. An approach that is valid for both cylindrical and translational joints uses equation 4.9.3.5.

$$\phi^{GT} = \phi_3^C - G(t) = 0 \qquad (4.9.3.7)$$

where

ϕ^{GT} is the translational generator constraint.

The simpler alternative approach is valid for only translational joints.

$$\phi^{GT} = \phi_3^T - G(t) = 0 \qquad (4.9.3.8)$$

Constraining the relative rotation about the $\hat{z}$ axis for cylindrical and revolute joints is more complicated.

$$\phi^{GR} = \tan^{-1}\left(\frac{A^I_{ij}\zeta^I_{\alpha j1}A^J_{ik}\zeta^J_{\beta k2}}{A^I_{ij}\zeta^I_{\alpha j1}A^J_{ik}\zeta^J_{\beta k1}}\right) - G(t) = 0 \quad (4.9.3.9)$$

where

ϕ^{GR} is the rotational generator constraint.

The constraint equations developed in this section are by no means unique; in several cases alternative forms are presented. Two cautionary notes are in order. First, when the dot product is used in a constraint equation, it must be used for enforcing orthogonality. Any attempt to use it to enforce co-alignment will result in a singular equation set at the solution since ±1 is an extremal value of the dot product function and therefore all of its first derivatives are zero. Second, the use of translational or cylindrical joints with deformable bodies may produce deceptive results. When two bodies slide relative to each other the point of contact changes, but the cylindrical and translational joints apply the reaction forces and moments at the same nodes regardless of the actual contact points.

4.10 Initial Conditions and Static Equilibrium

The displacement and velocity initial conditions must satisfy the constraints of the system in order for the numerical integration to procede smoothly. To ensure that the constraints are satisfied, an iterative refinement method is used that was first proposed by John Angell in 1979 and implemented by the author in ADAMS during 1980.

The initial displacement problem is to find the configuration closest to the one specified by the input data that satisfies the constraints.

$$\min J(\underline{q}) = \frac{1}{2} w_i (q_i - q_i^*)^2 \qquad (4.10.1)$$

subject to

$$\phi_k(\underline{q}, 0) = 0$$

where

w_i is a weighting factor.

q_i is the initial displacement.

q_i^* is the initial guess.

The higher the weighting factor, the less a variable is incremented during each iteration. In practice, a weight of 10^4 will hold a displacemnent to its initial value, and it is only applied when the input data specifies it. A weight of 10^{-2} is assigned to displacements that are not supplied in the input data, and 1 is used for displacements that are supplied but are not specified as being exact in the input data. The deformations of the bodies are assumed to be zero, and they are assigned a weight of 10^6.

By using Lagrange multipliers and setting the first variation of the Lagrangian to zero, a set of nonlinear equations are derived that are usually solved in less than ten iterations with Newton-Raphson iteration.

$$W_i(q_i - q_i^*) + \lambda_k \frac{\partial \phi_k}{\partial q_i} = 0$$
$$\phi_k = 0 \qquad (4.10.2)$$

Applying the same procedure to the initial velocities gives a similar set of equations. Only a single iteration gives a solution that satisfies the constraints because the derivatives with respect to time of the holonomic constraints are linear functions of the velocities. The solution of the equations gives the generalized velocities from which the quasi-velocities are calculated.

$$W_i(\dot{q}_i - \dot{q}_i^*) + \lambda_k \frac{\partial \phi_k}{\partial q_i} = 0$$
$$\frac{\partial \phi_k}{\partial q_i}\dot{q}_i + \frac{\partial \phi_k}{\partial t} = 0 \qquad (4.10.3)$$

The static equilibrium configuration is calculated by setting the accelerations and velocities of the system to zero, and solving the nonlinear force balance equations subject to the constraints on the system. Noting that the equations for static equilibrium are identical to the equations of motion with the velocity dependent and inertial terms discarded, Newton-Raphson iteration is applied to the equations of motion with the accelerations and velocities set to zero, and only the displacements, the Lagrange multipliers, and the applied forces are updated after every iteration.

4.11 Summary

A simple, generalized formulation for deformable mechanical systems was derived by using quasi-coordinates to eliminate the complexity associated with the rigid body rotation matrix from the inertial terms. A quick review of how the formulation addresses the desirable characteristics listed in section 4.2 shows the utility of the formulation.

1) The formulation is simple enough to program. Each body is described by its own set of equations. Systems are assembled by applying generic constraint and force expressions developed in section 4.9.
2) Other than a description of the finite element mesh and the material properties, no other information is necessary beyond what self-formulating rigid body dynamics programs, such as ADAMS, require.
3) Generality was achieved. The formulation handles plastic, elastic, and rigid bodies in a nearly uniform manner, but it is able to take advantage of the simplifications that elastic and rigid bodies allow. Additionally, the use of a locally attached frame allows the use of a variety of methods to generate the mode shapes without any of the difficulties associated with the floating

axes systems. A number of applied forces and joints were developed in section 4.9, and the possibilities are by no means exhausted.

4) The formulation is vectorizable. Equations are generated in the same manner as in finite element programs. Vectorization is discussed in detail in the next chapter.

5) An explicit Jacobian is readily evaluated so that using the stiffly stable integration method is not difficult.

CHAPTER 5

COMPUTER IMPLEMENTATION

This chapter documents the computer implementation, with the emphasis on vectorization, of the formulations presented in the previous chapter. Developed only to demonstrate the general formulation and that it vectorizes in a straightforward manner, the program neither represents the ultimate capabilites of the formulation, nor is it optimally-coded for vectorization.

5.1 Program Capabilities and Compromises

The program is self-formulating in the sense that the equations are automatically formulated and solved from input data describing the mechanical system. Both the displacement and the velocity initial conditions are refined before calculating the transient response of the system. Only the displacement solution is performed before a static equilibrium solution. In addition to rigid bodies and linearly elastic bodies with modal representations, a library of three types of joints, a concentrated force and gravity are available in the program. The output includes the relative displacements, velocities and accelerations of user-specified points in the system, and an interface to the

ADAMS Postprocessor [47] for plotting the output and displaying the configuration of the system with hidden line graphics.

Despite the smaller equation set, the specialized formulation for linearly elastic bodies differs from the general formulation only by the addition of the linear coordinate reduction transformation, therefore essentially all of the theory developed in the previous chapter is implemented, but at a lower computational cost than for the general formulation. Naturally, the smaller set of equations has a shorter average vector length and, consequently, a lower overall computational rate.

The use of a sparse matrix solver, even though its code is vectorized, is the biggest compromise from the standpoint of vectoriztion (see section 3.3.2). Replacing the FORTRAN equation solver with an assembly-coded solver would improve the execution rate at the expense of the memory requirements. A block-oriented solver is even faster on a vector processor, but the issues of ordering the equations to minimize the fill and to guarantee good pivots is beyond the scope of this dissertation, and they are handled automatically by the sparse equation solver. Gather and scatter operations are programmed in FORTRAN to make the code system independent, and they should be replaced by assembly-coded routines for better performance.

The program is completely variably dimensioned. Although the computational overhead is increased slightly, the cost is recovered by better memory useage and the ability to accomodate a wide range of problem sizes.

5.2 The Choice of Rigid Body Rotation Coordinates

The angular orientation of a body with respect to the ground reference frame is uniquely specified by three angles through which the body is successively rotated. Different angular coordinate sets are specified by choosing different axes of rotation. Some of the more familiar choices are Euler angles, Bryant angles, and yaw, pitch and roll. Although the axes of rotation are different, the different sets of angles have two important characteristics in common: first, the angles are singular for some value of the second angle, and second, the partial derivatives for the first and third angles of the transformation matrix A_{ij} are available, to within a sign, by interchanging rows and columns (or else they are zero).

To eliminate the singularity in the coordinate set more than three coordinates are used, and constraint equations are imposed to eliminate the redundancy of the coordinates. Of the nine direction cosines and their six constraints, and the four Euler parameters with one constraint equation, only the latter is computationally useful [48]. Although the partial derivatives of the

transformation matrix are not available by row and column interchanges, they are simple to calculate because the transformation is a quadratic function of the coordinates.

Despite the undeniable advantage of having no singular position with Euler parameters, Bryant angles are used in the program for expediency. The transformation matrix and its derivatives are needed throughout the program, and the reliability of the code is enhanced by calculating them once in a loop over the bodies and scattering them to the joint and force arrays. The same approach for Euler parameters would require a prohibitive amount of scattering, and calculating the partials in the joint and force loops requires more code and therefore increases the number of errors in the program. For program debugging, an added advantage of Bryant angles is the ability to directly compare them to the output from ADAMS without any additional coordinate transformations for simple problems.

The transformation matrices defined in equation 4.6.4.1, are given below for Bryant angles. Rotations θ_1, θ_2, and θ_3 are successively performed about the local x, y, and z axes of the body respectively, and the singular position occurs when θ_2 assumes the values $\pi/2 + \pi n$, where n is an integer.

$$A_{ij} = \begin{bmatrix} c_2c_3 & -c_2s_3 & s_2 \\ c_1s_3 + s_1s_2c_3 & c_1c_3 - s_1s_2s_3 & -s_1c_2 \\ s_1s_3 - c_1s_2c_3 & s_1c_3 + c_1s_2s_3 & c_1c_2 \end{bmatrix} \quad (5.3.1)$$

$$B_{ij} = \begin{bmatrix} c_3/c_2 & -s_3/c_2 & 0 \\ s_3 & c_3 & 0 \\ -c_3s_2/c_2 & s_3s_2/c_2 & 1 \end{bmatrix} \quad (5.3.2)$$

where

s_i and c_i are $\sin(\theta_i)$ and $\cos(\theta_i)$ respectively.

5.3 The General Approach to Vectorization

The equations of motion are generated and solved in the same manner as in a finite element program. Overall speedups as high as seventy-four to one over the original scalar codes for finite element analysis are reported in the literature [49], their successful vectorization being one of the major reasons why the equations of motion were derived from the weak form with independent rigid body coordinates.

Each element in the system, such as a spherical joint or a concentrated force, is represented by a submatrix that is algebraically identical but numerically different from the other submatrices of its type. Their calculation is vectorized by ordering the data in the program by the type of element and then calculating all of the submatrices of a particular element type in a DO-loop free from instruction branches. The vector length is, of course,

equal to the number of element submatrices calculated in the loop. Sometimes different element types have terms in common that are calculated together, thereby giving an even longer vector length.

The remainder of this chapter describes how particular elements in the program are vectorized. See the program for more detail.

5.4 The Vectorization of the Part Equations

Four loops are used to vectorize the part equations.

The first loop is over the rigid bodies for the terms that occur only with the rigid bodies, namely the rigid body inertial terms in equations 4.8.4 and 4.8.5, and equation 4.6.3.1 for the cartesian displacements X_1, X_2 and X_3, where Ψ_{ij} is the identity operator. The force of gravity for the rigid bodies is also calculated.

All of the force and joint reaction terms in the equations of motion for the rigid body coordinates are calculated in a single loop across both the rigid and flexible bodies. Additionally, the relation involving the angular velocities in equation 4.6.4.1 is evaluated for each body.

The third loop evaluates all of the terms common to the elastic bodies, such as the momenta terms, that are not functions of the modes, and the fourth loop is actually a pair of nested loops where the remaining terms that are functions of the modes are calculated. For the best computational performance on a vector processor, the inner

loop must have a greater range on the index variable than the outer loop since only the inner loop is vectorized. Typical mechanical systems have on the order of tens of bodies, with each body having tens of modes or hundreds to thousands of nodes. In keeping with the goal of demonstrating the general formulation with the computationally less demanding specialized formulation, the inner loop is over the modes and the outer loop is over the bodies to maintain an architectural similarilty to the general formulation.

Extra arithmetic operations that are associated with the linear coordinate reduction occur in the specialized formulation that do not occur in the general formulation, and they are eliminated from the inner loop of the simulation by calculating a set of constant matrices for each linearly elastic body in the beginning of the program.

$$C^1_{cj} = - H_{\alpha ic} \sum_{\beta} M_{\alpha i \beta j} \tag{5.4.1}$$

$$C^2_{cb} = - H_{\alpha ic} M_{\alpha i \beta j} H_{\beta jb} \tag{5.4.2}$$

$$C^3_{ck} = - H_{\alpha ic} M_{\alpha i \beta j} e_{jkn} r_{\beta n} \tag{5.4.3}$$

$$C^4_{cbk} = - H_{\alpha ic} M_{\alpha i \beta j} e_{jkn} H_{\beta nb} \tag{5.4.4}$$

$$C^5_{ckj} = e_{ikn} H_{\alpha nc} \sum_{\beta} M_{\alpha i \beta j} \tag{5.4.5}$$

$$C^6_{ckb} = -\,2H_{\alpha ic}M_{\alpha i\beta j}e_{jkn}H_{\beta nb} \tag{5.4.6}$$

$$C^7_{cmk} = e_{imn}H_{\alpha nc}M_{\alpha i\beta j}e_{jkp}r_{\beta p} \tag{5.4.7}$$

$$C^8_{cbmk} = e_{imn}H_{\alpha nc}M_{\alpha i\beta j}e_{jkp}H_{\beta pb} \tag{5.4.8}$$

$$C^9_{mj} = -\,e_{imn}r_{\alpha n}\sum_{\beta}M_{\alpha i\beta j} \tag{5.4.9}$$

$$C^{10}_{mk} = e_{imp}r_{\alpha p}M_{\alpha i\beta j}e_{jkn}r_{\beta n} \tag{5.4.10}$$

$$C^{11} = \sum_{\alpha,\beta} M_{\alpha i\beta i} \tag{5.4.11}$$

Note that C^{10}_{mk} and C^{11} are the finite element approximations of the inertia tensor and the mass of the body respectively. Equations 4.7.2, 4.7.3, and 4.7.4 are rewritten as equations 5.4.12, 5.4.13, and 5.4.14.

$$\begin{aligned} &-C^1_{cj}\ddot{Q}_j - C^2_{cb}\ddot{\eta}_b - [C^3_{cj} + C^4_{cbj}\eta_b]\dot{\omega}_j \\ &- [\,C^5_{cjk}Q_k + C^6_{cbj}\dot{\eta}_b + (C^7_{cjk} + C^8_{cbjk}\eta_b)\omega_k]\omega_j \\ &+ C^R_{cb}\dot{\eta}_b + K^R_{cb}\eta_b - A_{ji}H_{\alpha ic}\Big(\int_{\Omega_o}\rho_o f_j N_\alpha d\Omega_o \\ &+ \int_{\Gamma^t_o} t_j N_\alpha d\Gamma_o\Big) - \sum_k f^*_{kj}A_{ji}N_\alpha H_{\alpha ic} - \lambda_j\frac{\partial\phi_j}{\partial\eta_c} = 0 \end{aligned} \tag{5.4.12}$$

$$\bar{P}_{Q_i} - C^{11}\ddot{Q}_i + C^1_{bi}\ddot{\eta}_b + (C^9_{ki} - C^5_{bki}\eta_b)\omega_k = 0 \tag{5.4.13}$$

$$\bar{P}_{\omega_i} + (C^9_{ij} - C^5_{cij}\eta_c)Q_j + (C^3_{bi} + C^4_{bci}\eta_c)\dot{\eta}_b - [C^{10}_{ik} + (C^7_{cik} + C^7_{cki} + C^8_{cbik}\eta_b)\eta_c]\omega_k = 0 \qquad (5.4.14)$$

5.5 The Vectorization of the Joints and the Forces

By choosing a family of simple constraint equations and applied forces that are constructed out of similar terms, a vector length longer than what any single element type could provide occurs for the dominant part of the joint and force calculations.

The following functions, each involving only a single body, are used in the evaluation of the constraint equations and applied forces developed in the previous chapter.

$$\psi_1(i,\alpha,K) = X^K_i + A^K_{ij}(r^K_{\alpha j} + u^K_{\alpha j}) \qquad (5.5.1)$$

$$\psi_2(i,j,\alpha,K) = A^K_{in}\zeta^K_{\alpha nj} \qquad (5.5.2)$$

$$\psi_3(i,\alpha,K) = (X^K_n A^K_{nj} + (r^K_{\alpha j} + u^K_{\alpha j}))\zeta^K_{\alpha ji} \qquad (5.5.3)$$

where there is no sum on a and K.

Using the above equations, the constraints for the spherical, revolute, and translational joints developed in section 4.9 are rewritten.

Spherical joint

$$\phi^S_i = \psi_1(i,\alpha,I) - \psi_1(i,\beta,J) = 0 \quad i=1,2,3 \qquad (5.5.4)$$

Revolute joint

$$\begin{aligned} \phi_i^S &= 0 \quad i=1,2,3 \\ \phi_i^R &= \psi_2(i,3,\alpha,I) - \psi_2(i,3,\beta,J) = 0 \quad i=1,2 \end{aligned} \tag{5.5.5}$$

Translational joint

$$\begin{aligned} \phi_i^R &= 0 \quad i=1,2 \\ \phi_3^R &= \psi_2(1,1,\alpha,I) - \psi_2(1,1,\beta,J) = 0 \\ \phi_i^T &= \psi_3(i,\alpha,I) - \psi_3(i,\beta,J) = 0 \quad i=1,2 \end{aligned} \tag{5.5.6}$$

Equation 4.9.2.1, used in defining the direction of a concentrated force, is also rewritten.

$$\Delta x_i = \psi_1(i,\alpha,I) - \psi_1(i,\beta,J) \quad i=1,2,3 \tag{5.5.7}$$

A term of the form $A_{ij}^K \Lambda_j$ occurs in equations 5.5.1 through 5.5.3, and therefore a corresponding term of the form $B_{ji}^K \frac{\partial A_{kn}^K}{\partial \theta_j} \kappa_k \Lambda_n$ also occurs in the equations of motion for ω_i. Calculating the two common terms and some of their derivatives in a partitioned array for all of the constraints and forces gives a large vector length for even small problems.

$$L = 2N_{SJ} + 4N_{RJ} + 6N_{TJ} + 2N_{CF} \qquad (5.5.8)$$

where

L is the vector length.

N_{SJ} is the number of spherical joints.

N_{RJ} is the number of revolute joints.

N_{TJ} is the number of translational joints.

N_{CF} is the number of concentrated forces.

The remaining terms associated with ϕ_i^S, ϕ_i^R, ϕ_i^T, and the concentrated forces are calculated in separate loops free from instruction branches.

CHAPTER 6

NUMERICAL EXAMPLES

The solutions to three problems are presented in this chapter to demonstrate the accuracy and efficiency of the formulation developed in Chapter 4. All calculations were performed using double precision (64-bit) arithmetic on the Amdahl 5860 at the University of Michigan.

Regrettably, for reasons beyond the control of the author and the committee, DAVES was never run on a vector processor, and therefore no benchmarks showing the benefits of vectorization are available.

6.1 Rigid Body Dyanmics

To verify the accuracy of the rigid body dynamics calculations performed by DAVES, its output was compared to analytical solutions for simple problems, e.g., a simple pendulum, and to ADAMS for more complex problems. ADAMS, originally released in 1972, has undergone nearly constant development over the past decade, and its accuracy has been verified by comparing its output to both analytical solutions and experimental results. The particular version of ADAMS used in this dissertation was released by Mechanical Dynamics, Inc. on May 26, 1983.

The final verification problem, a suspension system, is diagrammed in Figure 6.1. Data was taken, in part, from the ADAMS Users Guide [47] and Orlandea [10]. Geometrical and inertial data are specified in Tables 6.1 through 6.3. The system of units is lbf-in-sec and the gravitational acceleration is -386.088 in/sec^2 in the Z direction.

Forces acting in the system are described by the following equations; refer to Figure 6.1 for their location. F/1 is the suspension spring.

$$f_1^* = -318(L_1 - 18.453) \tag{6.1.1}$$

F/2 is the force between the tire and the ground.

$$\begin{aligned} ZF(t) &= 1.5(\sin(20\pi t - \tfrac{\pi}{2}) + 1) \\ L_2^* &= L_2 - 14 - ZF(t) \\ f_2^* &= \begin{bmatrix} -1450L_2^* - 3\dot{L}_2^* & \text{if } L_2^* \le 0 \\ 0 & \text{if } L_2^* > 0 \end{bmatrix} \end{aligned} \tag{6.1.2}$$

F/3 is the shock absorber.

$$f_3^* = -10\dot{L}_3 \tag{6.1.3}$$

F/4 is the rebound bumper.

$$f_4^* = \begin{bmatrix} -33000(L_4 - 2) & \text{if } L_4 \le 2 \\ 0 & \text{if } L_4 > 2 \end{bmatrix} \tag{6.1.4}$$

F/5 is the jounce bumper.

$$f_5^* = \begin{bmatrix} -33000(L_5 - 1) & \text{if } L_5 \leq 1 \\ 0 & \text{if } L_5 > 1 \end{bmatrix} \qquad (6.1.5)$$

where

f_i^* is the i-th force magnitude.

L_i is the displacement between the nodes on which f_i^* acts.

TABLE 6.1

INERTIAL PROPERTIES FOR THE SUSPENSION SYSTEM

Part Name	Part Number	Weight (lbf)	J_x (lbf-in^2)	J_y (lbf-in^2)	J_z (lbf-in^2)
Ground	1	∞	∞	∞	∞
Lower Control Arm	2	15.83	389.95	772.18	463.31
Spindle and Tire Assembly	3	70.27	3420.74	2837.75	451.72
Upper Control Arm	4	6.99	108.10	173.74	92.66
Tie Rod	5	3.44	19.30	19.30	19.30

TABLE 6.2

NODAL COORDINATES FOR THE SUSPENSION SYSTEM

Node	X (inches)	Y (inches)	Z (inches)
1	15.25	-17.97	13.45
2	20.71	-19.15	14.37
3	21.02	-19.19	13.33
4	24.40	-18.39	13.99
5	20.53	-19.63	11.38
6	7.30	-21.60	3.41
7	25.29	-25.01	2.99
8	30.72	-18.57	6.30
9	31.04	-18.57	6.30
10	20.19	-20.87	6.41
11	17.34	-16.91	10.08
12	11.76	-12.40	3.14
13	17.81	-16.85	2.52
14	18.88	-17.84	-0.10
15	20.43	-21.02	5.20
16	26.23	-18.65	3.20
17	30.72	-18.57	-6.65
18	16.29	-23.31	3.20
19	17.88	-17.20	11.03
20	18.40	-17.64	-0.41

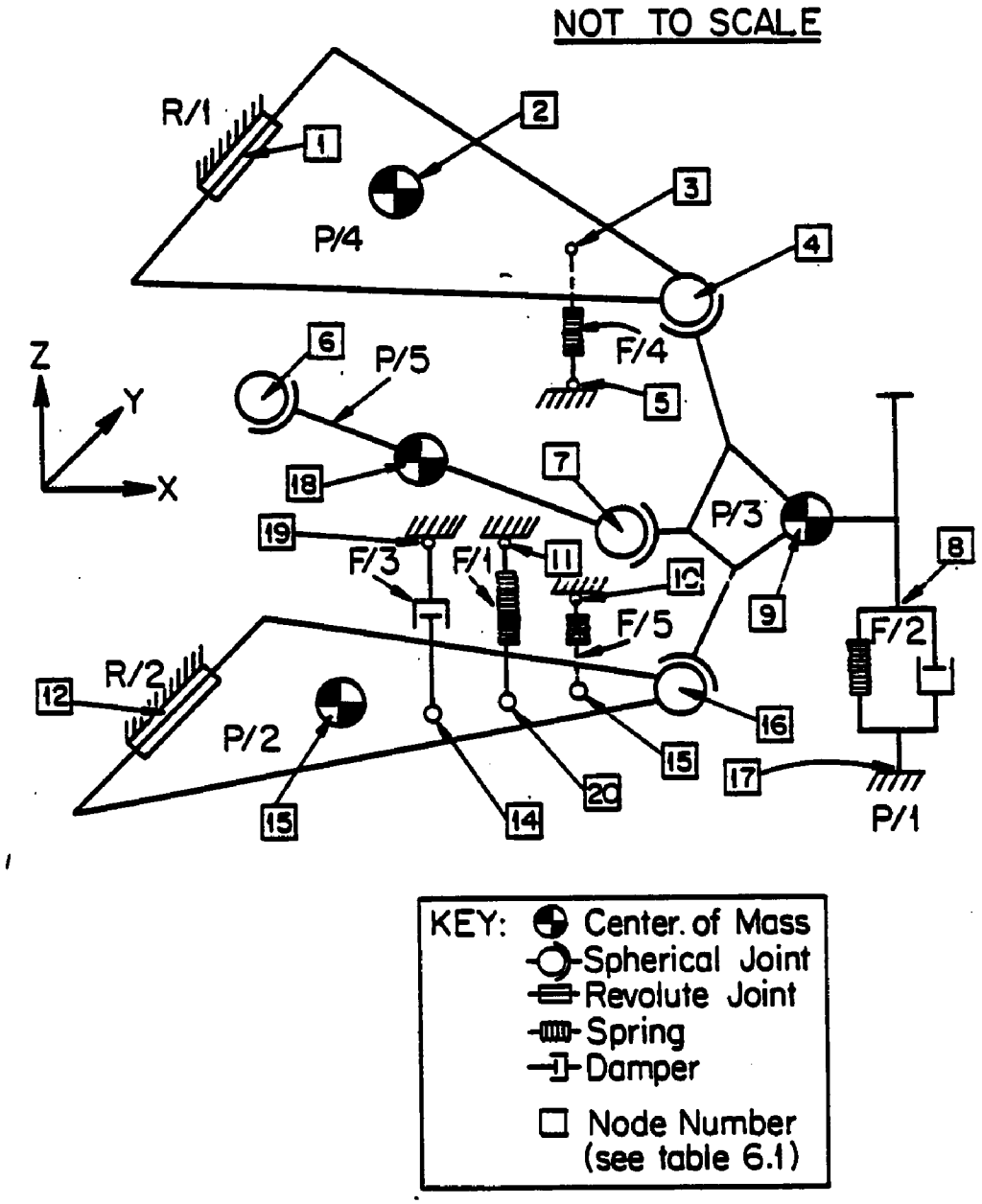


Fig. 6.1. Schematic of the automobile suspension system.

TABLE 6.3

THE DIRECTION COSINES
OF THE REVOLUTE ROTATION AXES
FOR THE SUSPENSION SYSTEM

Revolute	$\hat{i}$	$\hat{j}$	$\hat{k}$
R/1	.095390	.965819	-.241029
R/2	.362502	.931398	-.033023

The motion of the system consists primarily of translation in the Z direction and rotation about the Y axis. For comparing the output of DAVES and ADAMS, the displacement, velocity, and acceleration components in those directions for the center of mass of the lower arm are plotted in Figures 6.4 through 6.9 by using the ADAMS Postprocessor. Output from DAVES is plotted with a solid line, ADAMS, with a dashed line. The agreement is excellent.

The computational performance of the two programs is summarized in Table 6.4. An integration error criterion of 10^{-6} was used and the Gear integration package reported no corrector convergence failures for either program. The similarity between the performance of the programs is more significant than their minor differences.

TABLE 6.4

THE COMPUTATIONAL PERFORMANCE OF
DAVES AND ADAMS FOR A
SUSPENSION SYSTEM

Program	No. of Equations	No. of Integration Corrector Iterations	No. of Integration Timesteps	No. of Jacobian Evaluations	CPU (sec.)
DAVES	100	1180	357	405	11.255
ADAMS	122	1213	399	450(est.)	11.752

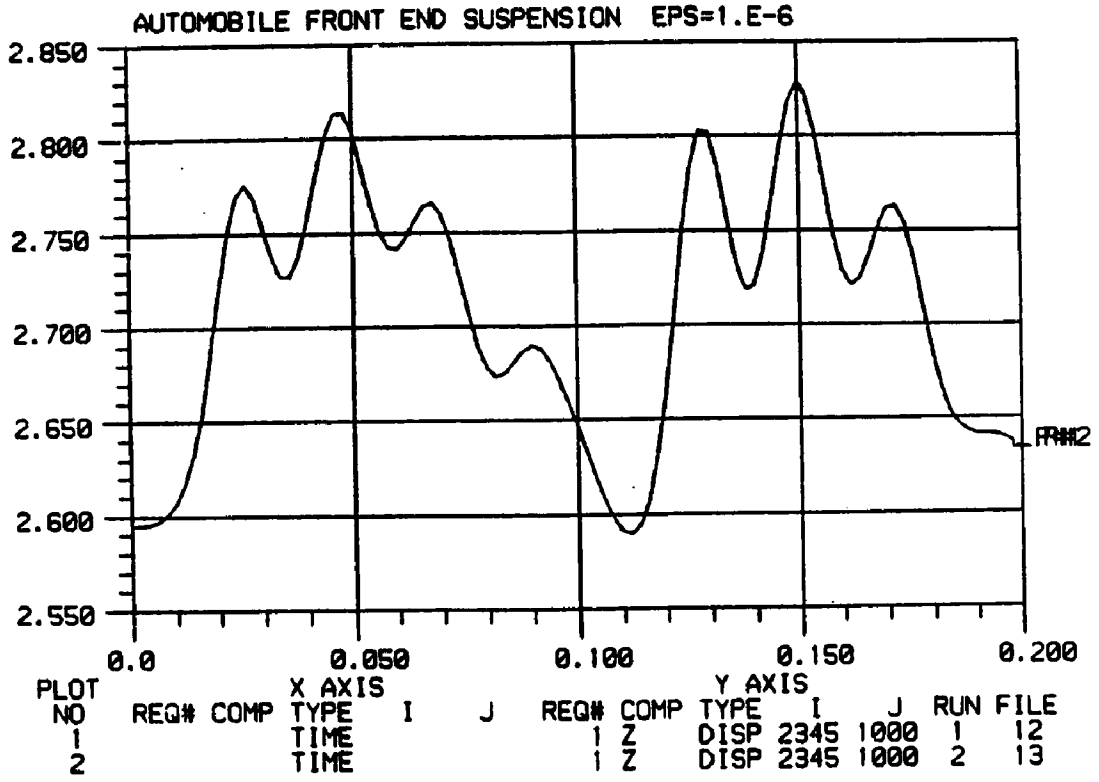


Fig. 6.2. Displacement of the lower arm in the Z direction.

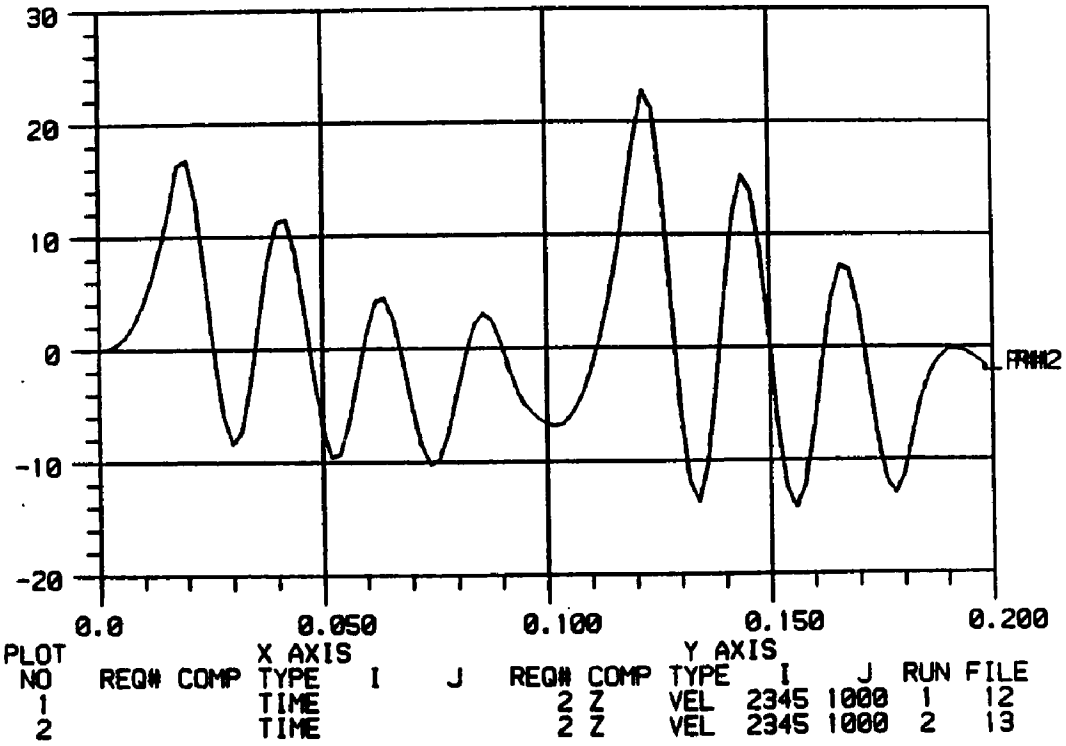


Fig. 6.3. Velocity of the lower arm in the Z direction.

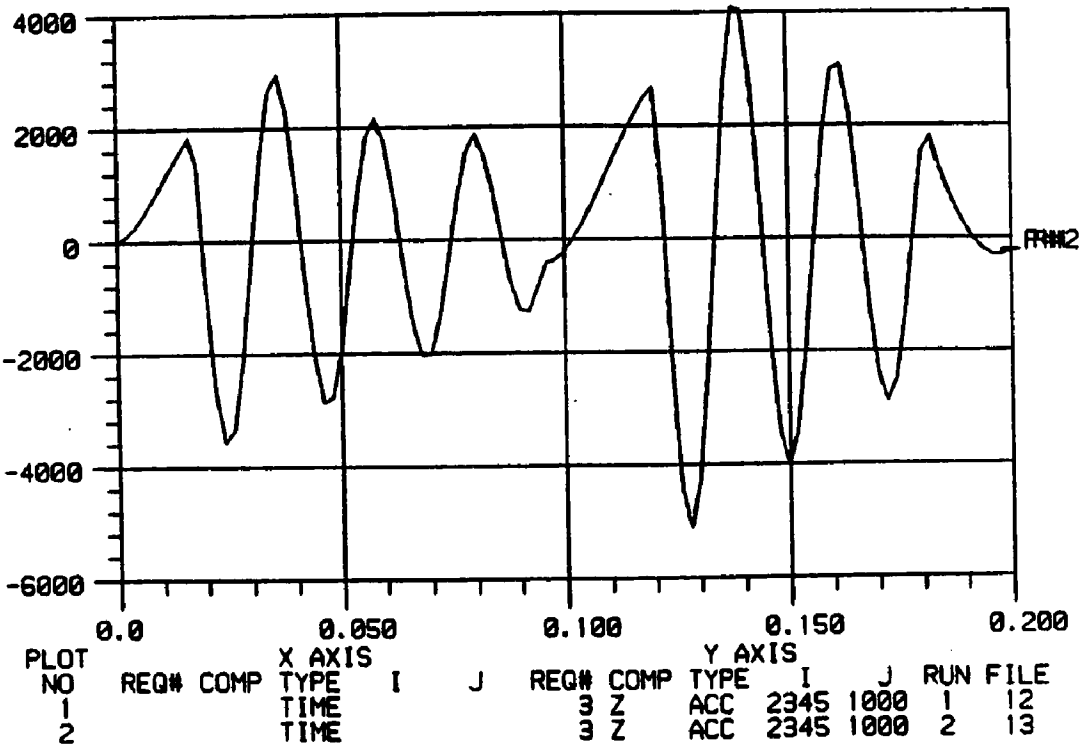


Fig. 6.4. Acceleration of the lower arm in the Z direction.

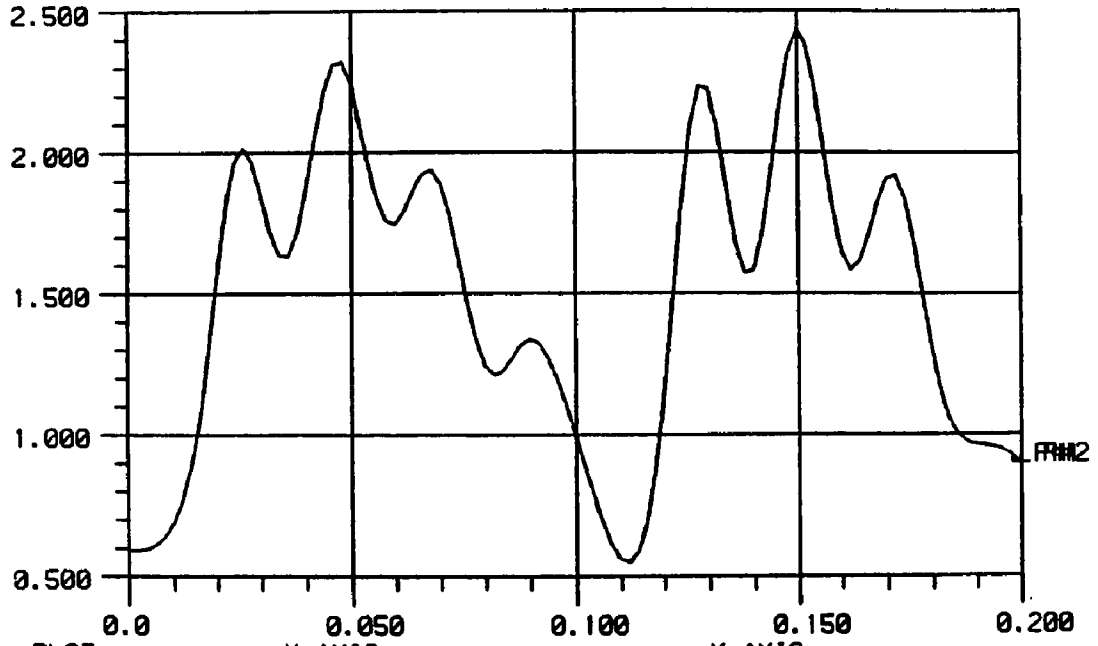


Fig. 6.5. Theta, measured in degrees, of the lower arm.

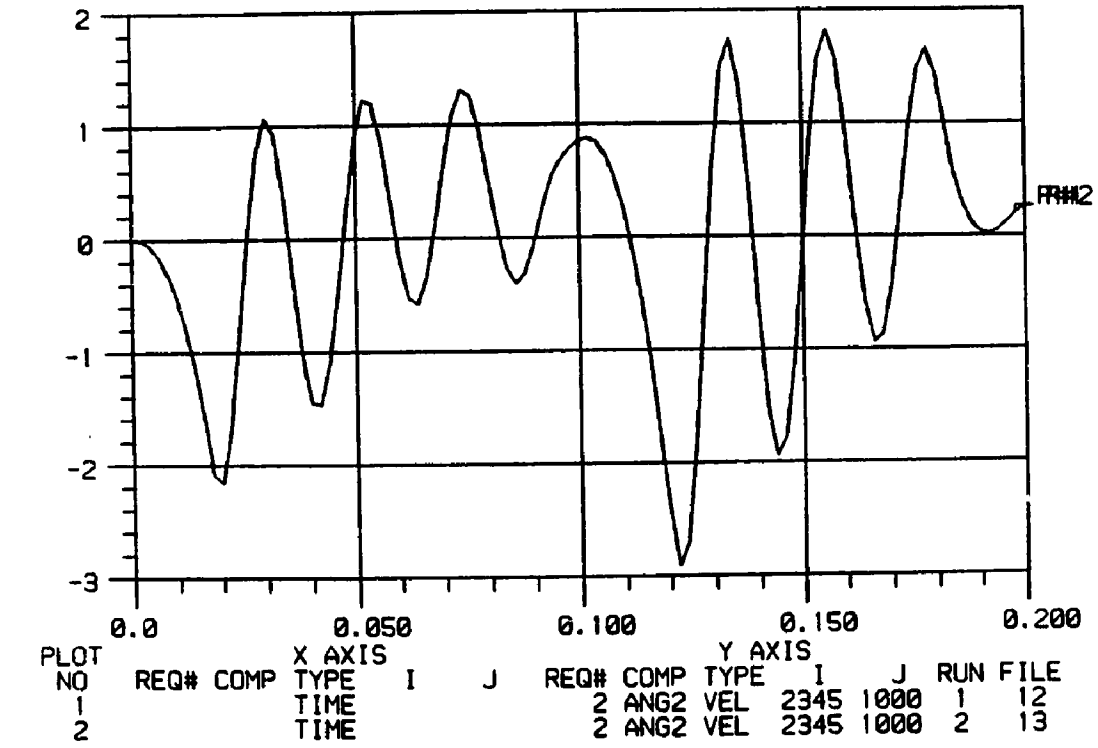


Fig. 6.6. Angular velocity of the lower arm about the Y axis.

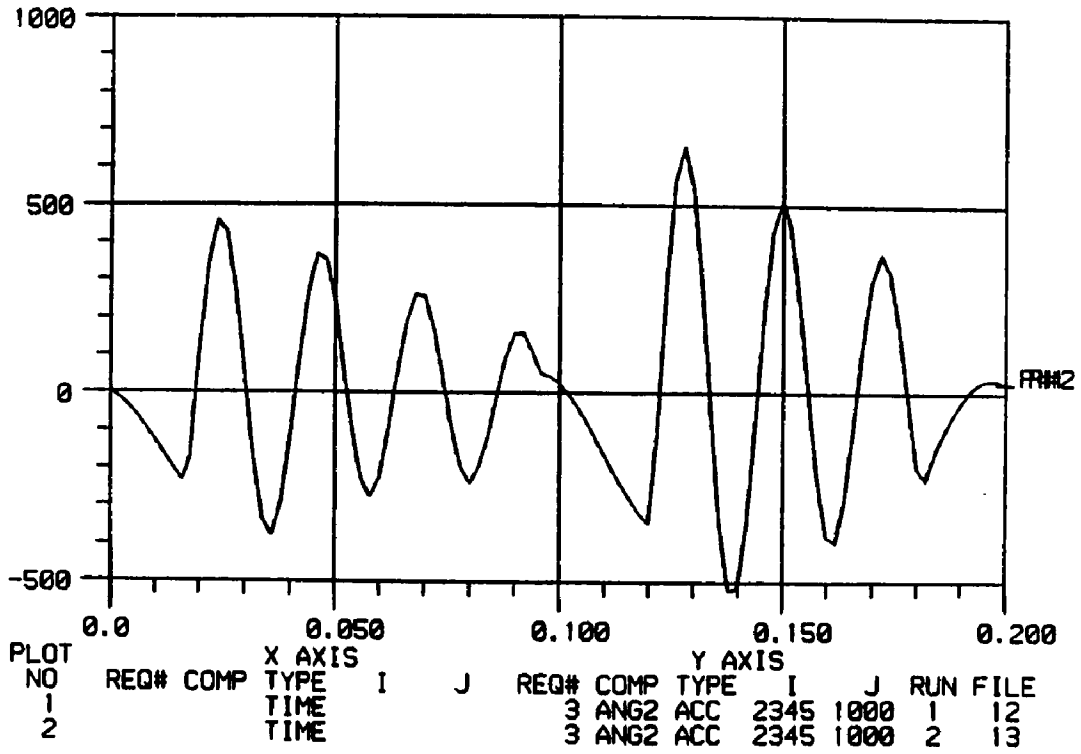


Fig. 6.7. Angular acceleration of the lower arm about the Y axis.

6.2 A Finite Element with Selective Reduced Integration

This section documents the eight node hexahedral element that is used in the remaining examples. The discussion of a particular finite element is delayed to this chapter because the general formulation in Chapter 4 does not asume a particular element and the mode shapes are calculated outside of the program described in Chapter 5. A program named FEM, written by Professor Kikuchi, uses the hexahedral element and a lumped mass matrix, which is calculated by the fourth method discussed in section 4.3.4, with Jacobi iteration to calculate the modes subject to the appropriate displacement boundary conditions, if any.

From Figure 6.8, the basis functions are intuitively obvious.

$$N_\alpha = \frac{1}{8}(1 + \xi_1^*\xi_{\alpha 1}^*)(1 + \xi_2^*\xi_{\alpha 2}^*)(1 + \xi_3^*\xi_{\alpha 3}^*) \qquad (6.2.1)$$

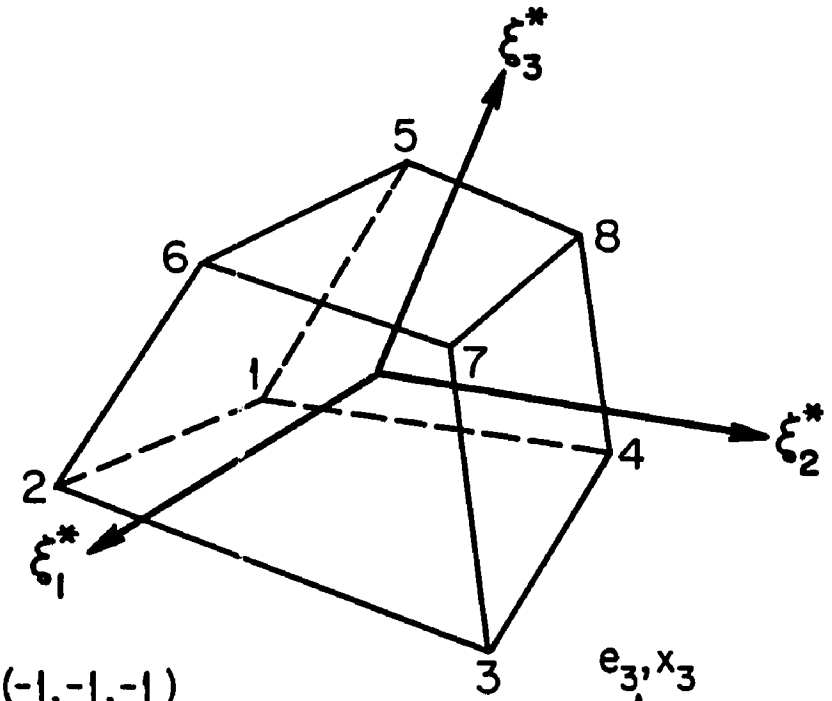
where there is no summing over α and

ξ_i^* are the model element coordinates.

The integrals in Chapter 4 are transformed from the material domain to the model element domain by applying equation 4.3.2.3.

$$\int_{\Omega_o} f d\Omega_o = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} f|J| d\xi_1^* d\xi_2^* d\xi_3^*$$

$$J = \det\left(\frac{\partial \xi_i}{\partial \xi_j^*}\right) \qquad (6.2.2)$$



NODE	
1	$\xi^* = (-1,-1,-1)$
2	$\xi^* = (1,-1,-1)$
3	$\xi^* = (1, 1,-1)$
4	$\xi^* = (-1, 1,-1)$
5	$\xi^* = (-1,-1, 1)$
6	$\xi^* = (1,-1, 1)$
7	$\xi^* = (1, 1, 1)$
8	$\xi^* = (-1, 1, 1)$

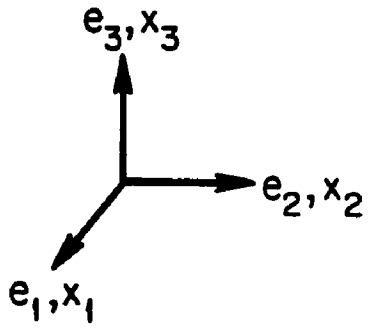


Fig. 6.8. Eight node hexahedral element.

Numerical integration is performed using Gaussian quadrature.

$$\int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} f|J|d\xi_1^* d\xi_2^* d\xi_3^* = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} f_{ijk}|J_{ijk}|w_i w_j w_k \quad (6.2.3)$$

where

f_{ijk} is $f(\bar{\xi}_i^*, \bar{\xi}_j^*, \bar{\xi}_k^*)$.

J_{ijk} is $J(\bar{\xi}_i^*, \bar{\xi}_j^*, \bar{\xi}_k^*)$.

w_i are the Gauss weights.

$\bar{\xi}_i^*$ are the Gauss sampling points.

n is the number of Gauss points.

Assuming small strains, the Cauchy and the two Piola-Kirchhoff stress tensors are almost equal, and, assuming material isotropy, equation 4.3.3.1 is rewritten using the Lame coefficients.

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{23}\\ \sigma_{31}\\ \sigma_{32}\end{bmatrix} = \begin{bmatrix}\lambda+2\mu & \lambda & \lambda & 0 & 0 & 0\\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0\\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0\\ 0 & 0 & 0 & \mu & 0 & 0\\ 0 & 0 & 0 & 0 & \mu & 0\\ 0 & 0 & 0 & 0 & 0 & \mu\end{bmatrix} \begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{33}\\ \gamma_{23}\\ \gamma_{31}\\ \gamma_{32}\end{bmatrix} \quad (6.2.4)$$

or

$$\left[\sigma\right] = \left[D\right]\left[\varepsilon\right] \tag{6.2.5}$$

where

λ and μ are the Lame coefficients.

γ_{ij} are the shear strains, $2\varepsilon_{ij}$.

The relationships between the Lame coefficients and the more commonly used modulus of elasticity, Poisson's ratio, and the shear modulus are given below. Note that λ is infinite when Poisson's ratio is 0.5.

$$\lambda = \frac{E\nu}{(1 + \nu)(1 - 2\nu)}$$

$$\mu = G = \frac{E}{2(1 + \nu)} \tag{6.2.6}$$

where

E is the modulus of elasticity.

ν is Poisson's ratio.

G is the shear modulus.

Traditionally, the eight node tetrahedral element stiffness matrix is derived by substituting equations 6.2.1 and 6.2.4 into equation 4.3.3.5, and evaluating the integrals with 2x2x2 integration. The resulting matrix is too stiff, but by using selective reduced integration (one point integration in this case), the accuracy of the element is significantly increased.

The D matrix is decomposed into three matrices, one associated with the volumetric strain energy, and the other two, shear strain energy [32].

$$\left[D\right] = \left[D^{V}\right] + \left[D_1^{S}\right] + \left[D_2^{S}\right] \tag{6.2.7}$$

$$\left[D^{V}\right] = \lambda \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.2.8}$$

$$\left[D_1^{S}\right] = \mu \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.2.9}$$

$$\left[D_2^S\right] = \mu \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (6.2.10)$$

where

$\left[D^V\right]$ is associated with the volumetric strain energy.

$\left[D_i^S\right]$ is associated with the shear strain energy.

Stiffness contributions from $[D_2^S]$ are evaluated with 2x2x2 integration but the $[D^V]$ and $[D_1^S]$ contributions are evaluated using one point integration. There are three important advantages gained by this procedure:

1) No "locking" for materials that are nearly incompressible [32].
2) No "locking" of the shear deformations [53]. The inclusion of the additional degrees of freedom in the element derived in the reference is equivalent to selective reduced integration.
3) "Hourglass" modes, occuring when reduced integration is applied to all of the terms, are eliminated.

"Locking" refers to obtaining no deformations as a solution for a well-posed problem because the numerical methods (such as a poor choice of a quadrature method) overconstrain the problem. Zienkiewicz explains the locking phenomena for incompressible materials by decomposing the stiffness matrix in a manner analogous to equation 6.2.6 and examining the behaviour of the solution as Poisson's ratio approaches 0.5.

$$K_{\alpha i \beta j} = \lambda K^{v}_{\alpha i \beta j} + \mu K^{s}_{\alpha i \beta j} \qquad (6.2.11)$$

where

$\lambda K^{v}_{\alpha i \beta j}$ is the $[D^{v}]$ contribution to $K_{\alpha i \beta j}$.

$\mu K^{s}_{\alpha i \beta j}$ is the $[D^{s}_{1}]$ and $[D^{s}_{2}]$ contribution to $K_{\alpha i \beta j}$.

As ν approaches 0.5, λ approaches infinity, and equation 4.3.3.5 is dominated by $[K^{v}]$.

$$K_{\alpha i \beta j} u_{\beta j} = f_{\alpha i} \simeq \lambda K^{v}_{\alpha i \beta j} u_{\beta j} \qquad (6.2.12)$$

giving

$$K^{v}_{\alpha i \beta j} u_{\beta j} = \frac{1}{\lambda} f_{\alpha i} \qquad (6.2.13)$$

For a nonsingular $[K^{v}]$ the solution must therefore approach zero as λ approaches infinity. To eliminate the locking, a singularity is imposed on $[K^{v}]$ by using reduced integration with $[D^{v}]$, resulting in a nontrivial solution for equation 6.2.13.

Shear locking is more complex; the reader is referred to the previously cited reference for an anlysis of shear locking in planar quadrilateral elements. For beam elements Zienkiewicz gives an analysis similar to the preceding one.

Hourglass modes are displacement modes that, like rigid body modes, do not contribute to the strain energy of the element. Caused by reduced integration, they are named after their shapes. Their control is the subject of continuing research [21], [54]; the use of selective reduced integration [55] is but one method for their control.

Although selective reduced integration is discussed in terms of the eight node hexahedron, the discussion is valid for twenty node and twenty-seven node hexahedrons (which use 3x3x3 and 2x2x2 Gauss quadrature for full and reduced integration respectively) as well as other element types.

6.3 The Vibration of a Cantilevered Beam

The classical structural dynamics problem of a cantilevered beam is analyzed in this section to demonstrate that DAVES correctly formulates and solves the equations of motion for the deformations of the body. The data for the beam follows:

Length (X direction) . .	10.0 inches
Depth (Y direction) . . .	1.5 inches
Height (Z direction) . .	1.0 inches
Mesh (X x Y x Z)	3 x 2 x 1

Young's modulus . . . . . 8.0×10^4 psi

Poisson's ratio 0.3

Density . . . . . 1.0 lbf-sec^2/in.^{4}

6.3.1 The Natural Frequencies

The natural frequencies of a cantilevered beam are the solutions to an eigenvalue problem with generates a characteristic equation which is solved numerically [56].

$$\cos(\beta_i L)\cosh(\beta_i L) = -1 \text{ (no sum on i)} \qquad (6.3.1.1)$$

$$\beta_i^4 \equiv \frac{\omega_i^2 m_L}{EI} \qquad (6.3.1.2)$$

where

L is the length of the beam.

m_L is the mass per unit length.

I is the cross-sectional area moment.

Substitution of the specified material properties gives a simple expression for the natural frequencies for the example problem.

$$\omega_i = \frac{\sqrt{2}}{\sqrt{3}}\beta_i^2 \qquad (6.3.1.3)$$

A crude mesh is used to demonstrate the substantial difference between a hexahedral element with full integraton and one with selective reduced integration. A comparison of the theoretical values of the bending modes in the X-Z plane with the calculated values shows that the beam model using

the selectively integrated elements approaches the solution from below and is more accurate than the model using full integration.

TABLE 6.5

COMPARISON OF NATURAL FREQUENCIES

Bending Mode	Theoretical Value of ω_i	Selective Reduced Integration		Full Integraton	
		Calc. ω_i	Percent Error	Calc. ω_i	Percent Error
1	2.87	2.42	-16.	6.32	120.
2	17.99	14.83	-18.	33.91	88.
3	50.37	50.06	-1.	79.89	59.

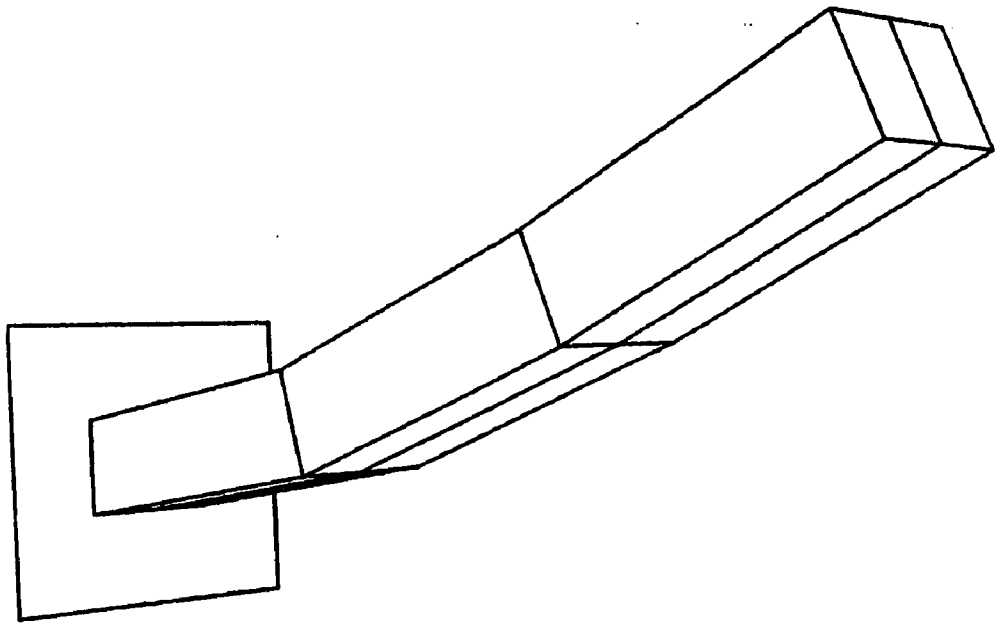


Fig. 6.9. First bending mode.

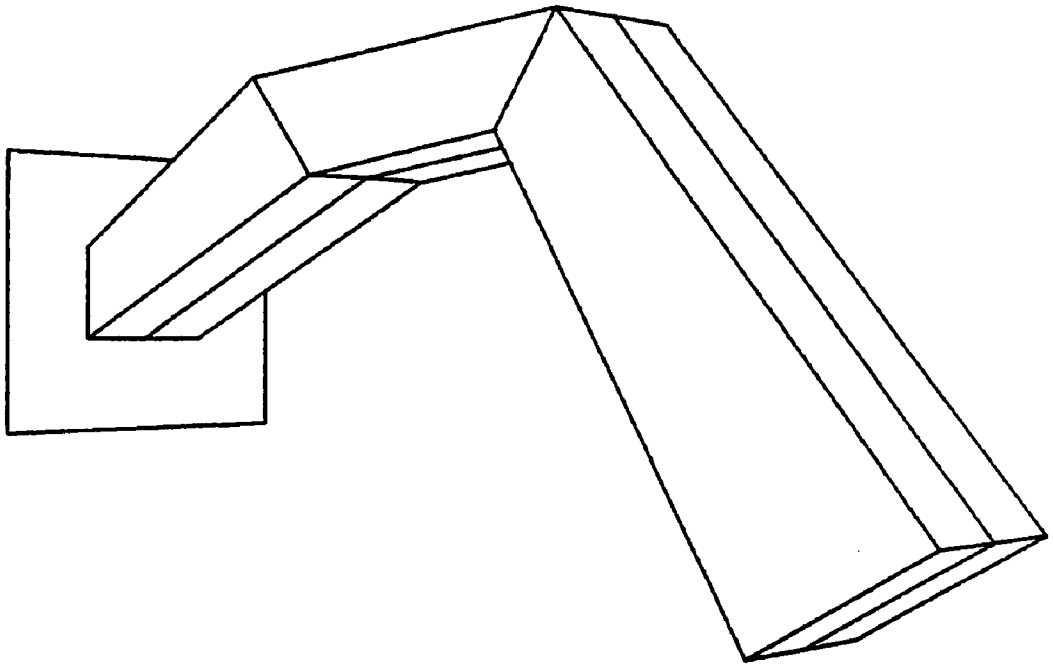


Fig. 6.10. Second bending mode.

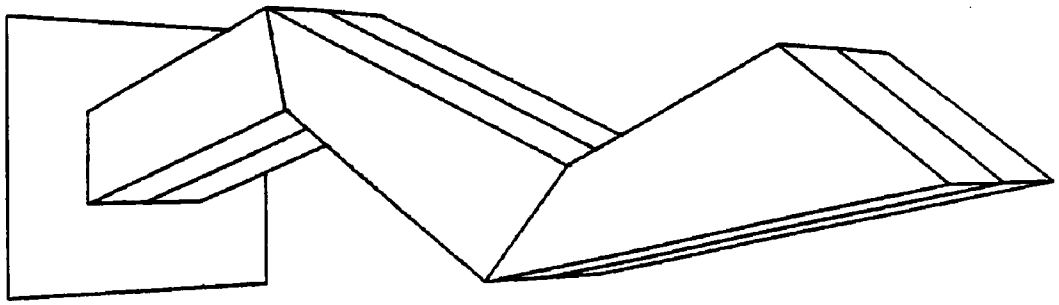


Fig. 6.11. Third bending mode.

6.3.2 Critical Excitation of a Bending Mode

For a cantilevered beam, represented by the first bending mode, the equation of motion is very simple.

$$\ddot{\eta}_1 + \xi_1\dot{\eta}_1 + \omega_1^2\eta_1 = H_{\alpha i 1}f^*_{\alpha i} \qquad (6.3.2.1)$$

The response of the system is unbounded when the applied forces have the same period as the beam and the system has no damping. To demonstrate this phenomena with DAVES, a point force is applied to the end of the beam in the Z direction. Following the general expressions are the numerical expressions that apply to the example.

$$\begin{aligned} f^* &= A\cdot\cos(\omega_1 t) \\ &= 2.0\cos(2.42t) \end{aligned} \qquad (6.3.2.2)$$

$$\begin{aligned} \eta_1 &= \frac{HA}{2\omega_1}t\cdot\sin(\omega_1 t) \\ &= 0.204t\cdot\sin(2.42t) \end{aligned} \qquad (6.3.2.3)$$

$$\begin{aligned} Z_e &= \frac{H^2A}{2\omega_1}t\cdot\sin(\omega_1 t) \\ &= 0.101t\cdot\sin(2.42t) \end{aligned} \qquad (6.3.2.4)$$

where

H is the appropriate $H_{\alpha 31}$.

A is the amplitude of the applied force.

Z_e is the deflection of the tip of the beam.

Output from the program agrees with the theoretical calculations very closely; Figures 6.12 through 6.14 show the response of the system.

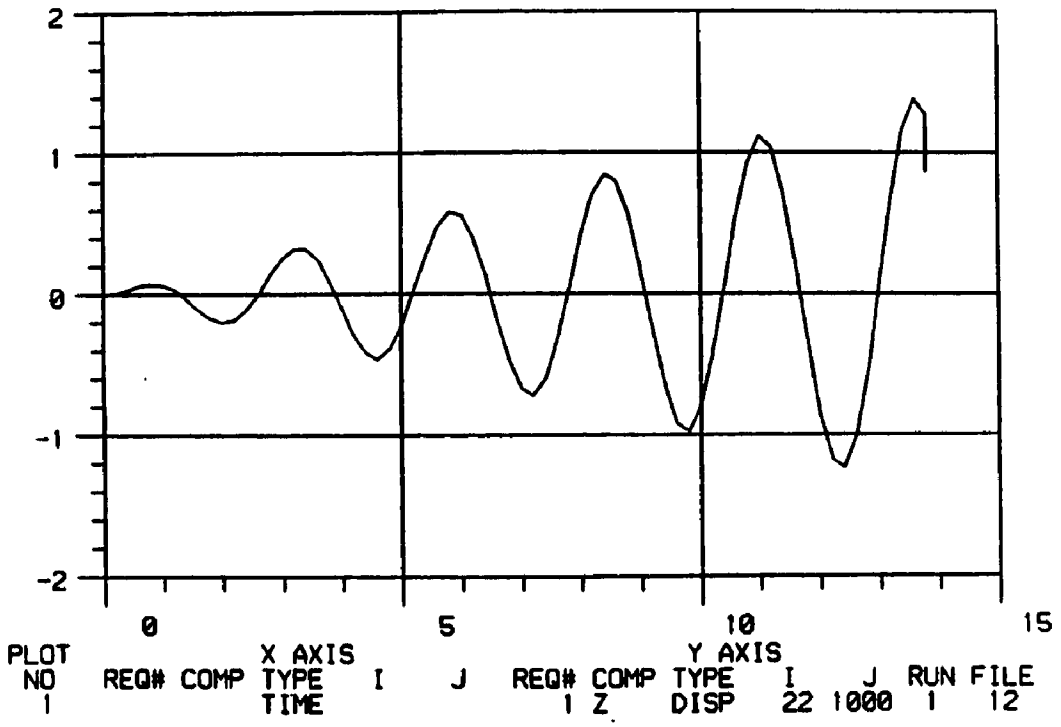


Fig. 6.12. The deflection of the tip of the beam.

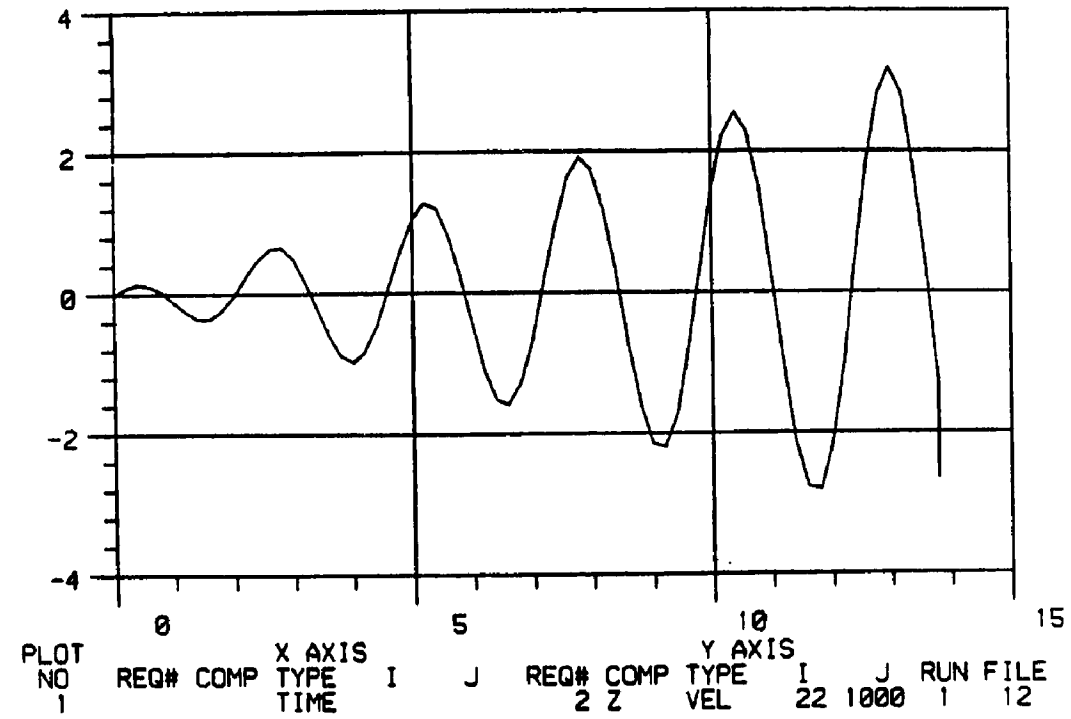


Fig. 6.13. The velocity of the tip of the beam.

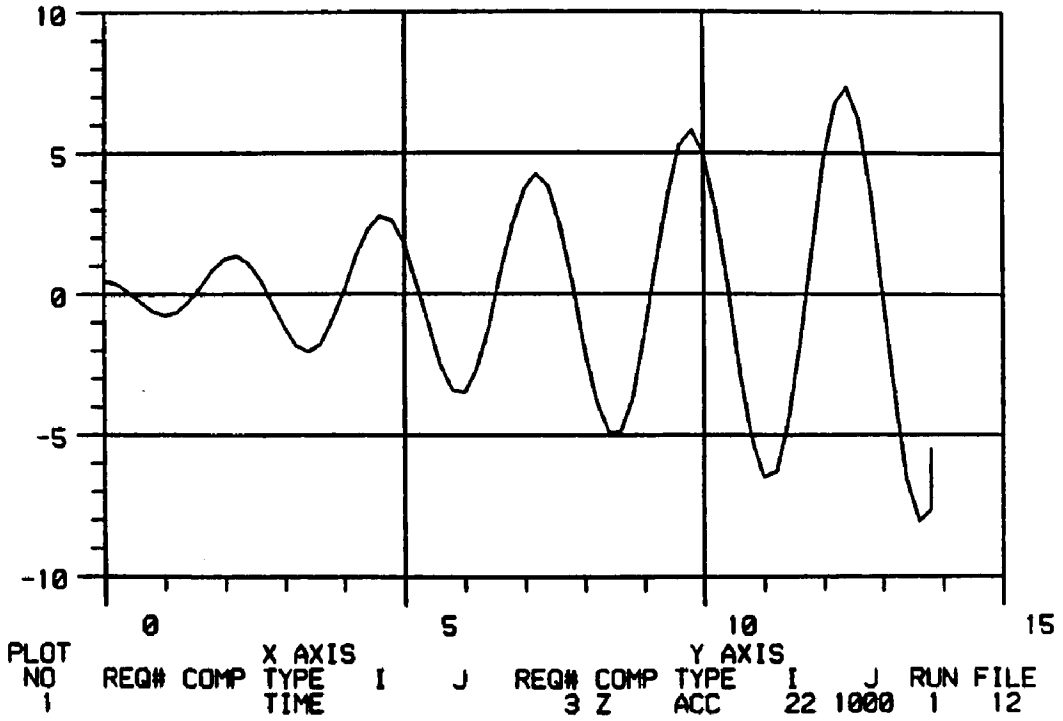


Fig. 6.14. The acceleration of the tip of the beam.

6.3.3 Computational Cost as a Function of the Beam Model

The performance of DAVES for several combinations of bending modes and damping, with the forcing function described in equation 6.3.2.2, is summarized in Table 6.6. Each case was run for five seconds of simulation time with ten output steps and an integration error tolerance of 10^{-6}. Two important generalities are apparent from the table:

1) The highest frequency roughly determines the computational cost.
2) Adding damping decreases the computational cost. As the damping increases so does the mathematical stiffness, and a stiff integration method becomes increasingly necessary.

TABLE 6.6

COMPUTATIONAL COST
AS A FUNCTION OF THE BEAM MODEL

Problem	Modes	Damping	No. of Corrector Iterations	No. of Timesteps	No. of Jacobian Evaluations
1	1	$\xi_1=0$	322	101	120
2	1,2	$\xi_1=0$ $\xi_2=0$	384	121	141
3	1,2,3	$\xi_1=0$ $\xi_2=0$ $\xi_3=0$	2043	596	698
4	3	$\xi_3=0$	2112	611	721
5	1,2,3	$\xi_1=0$ $\xi_2=1$ $\xi_3=1$	1634	486	562
6	1,2,3	$\xi_1=0$ $\xi_2=10$ $\xi_3=10$	1059	320	369

The differences in the response of the models that include the first modes are, for this particular example, only significant in the accelerations. In the following figures, the response of problems 1, 3, and 5 are plotted using solid lines, dash-dot-dash lines, and dashed lines, respectively.

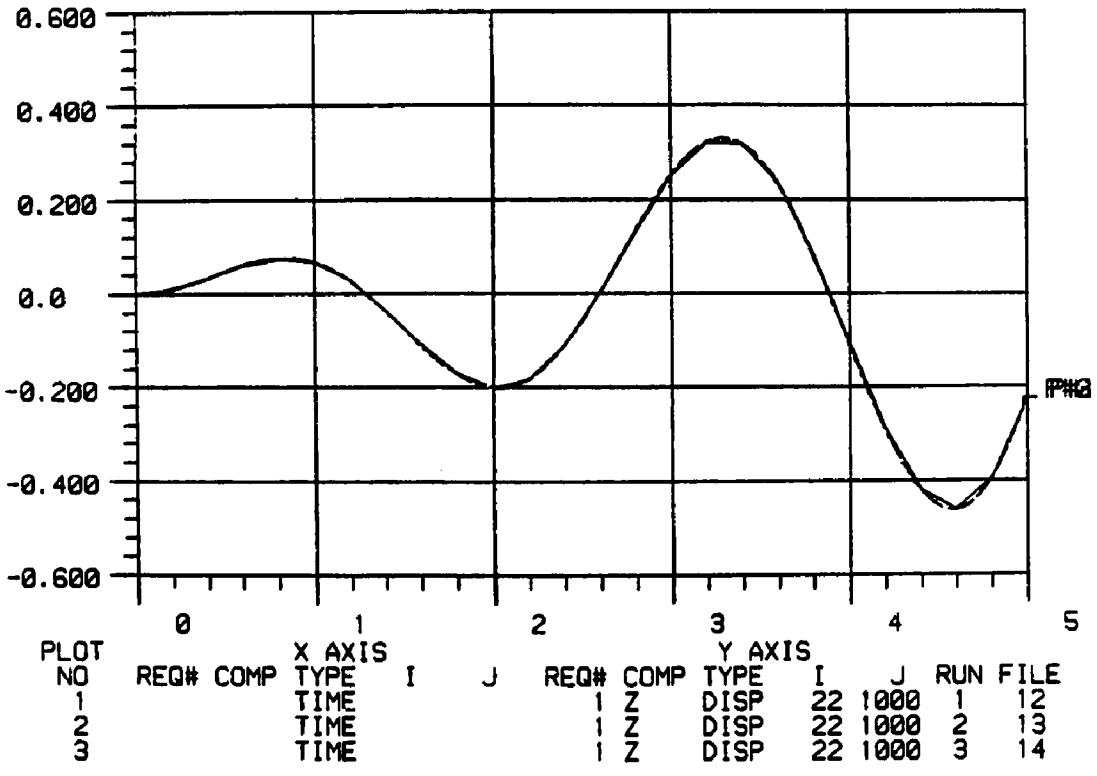
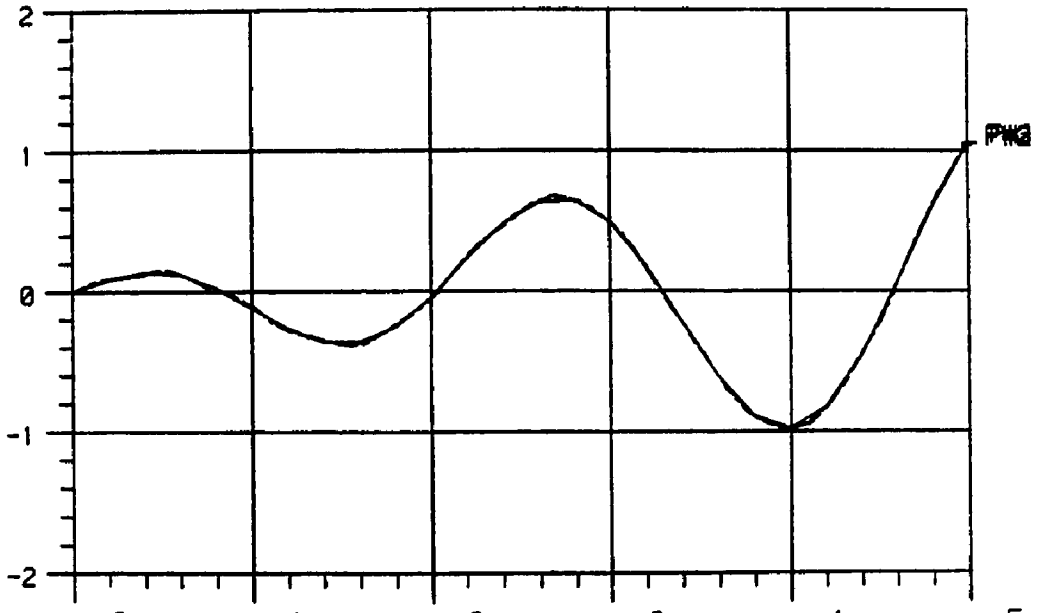
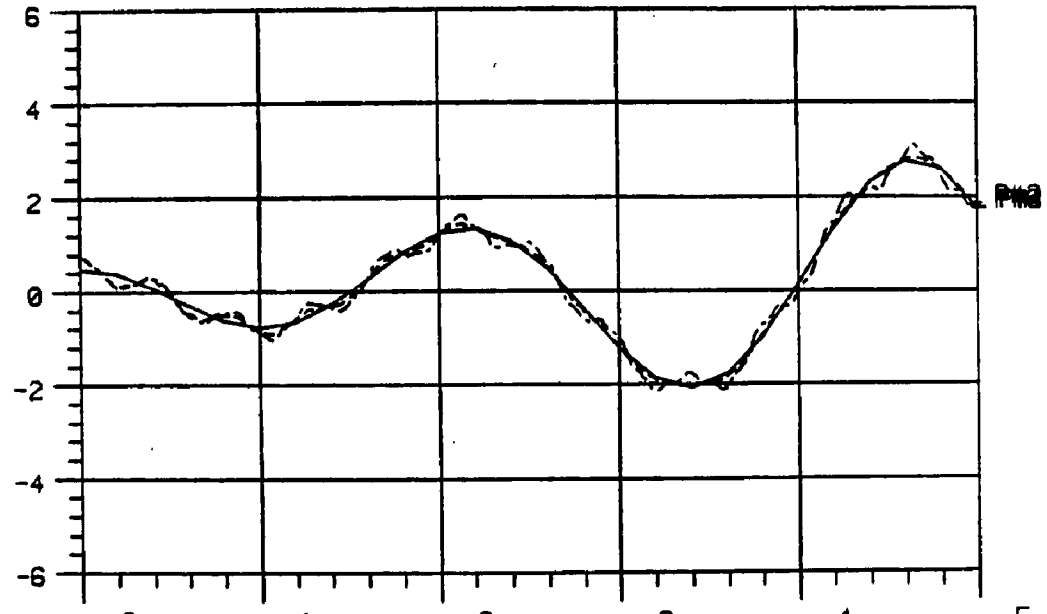


Fig. 6.15. The deflection of three beam models at the tip.



PLOT NO	X AXIS REQ#	COMP	TYPE	I	J	Y AXIS REQ#	COMP	TYPE	I	J	RUN	FILE
1			TIME			2	Z	VEL	22	1000	1	12
2			TIME			2	Z	VEL	22	1000	2	13
3			TIME			2	Z	VEL	22	1000	3	14

Fig. 6.16. The velocity of three beam models at the tip.



PLOT NO	X AXIS REQ#	COMP	TYPE	I	J	Y AXIS REQ#	COMP	TYPE	I	J	RUN	FILE
1			TIME			3	Z	ACC	22	1000	1	12
2			TIME			3	Z	ACC	22	1000	2	13
3			TIME			3	Z	ACC	22	1000	3	14

Fig. 6.17. The acceleration of three beam models at the tip.

6.4 Analysis of a Flexible Slider-Crank Mechanism

The analysis of a flexible slider-crank mechanism demonstrates not only the ability of DAVES to accurately and efficiently simulate a mechanical system with both linear deformations of the bodies and large rigid body motion, but also the importance of properly choosing the modes to represent the elastic body.

6.4.1 Problem Description

Although there is no analytical solution available for the flexible slider-crank mechanism, illustrated in Figure 6.18, it was simulated numerically by two independent groups using different methods [57],[6], with good correlation so that the solution is known to within a small envelope. The essential data follows:

Crank length . . . . . 6.0 inches

Connecting rod length. . 12.0 inches

Dia. of connecting rod . 0.25 inches

Crank speed . . . . . 124.8 radians/sec.

Young's modulus . . . . 30.0×10^{6} psi

Density . . . . . 7.331×10^{-4} lbf-sec^2/in^4.

Poisson's ratio 0.29

All parts except for the connecting rod are rigid, and the connecting rod is undamped. The slider is massless.

To convert the problem to three dimensions for DAVES, several changes were made:

1) The beam was modelled with a rectangular cross section chosen to have the same bending stiffness and mass/length as the circular beam so that the brick element described in section 6.2 could be used.

   Length (Z direction). . . 12.0 inches

   Depth (Y direction) . . . 0.2267 inches

   Height (X direction). . . 0.2165 inches

   Mesh (X x Y x Z). . . . . 2 x 1 x 2

2) A heavy flywheel, with an inertia of 10^3 lbf-in^2, replaces the crank since DAVES does not have the rotational displacement constraint, equation 4.9.3.9, implemented in it.

3) Additional parts and joints are introduced, Figure 6.19, to prevent the system from being overconstrained in the Y direction.

The initial configuration of the system has the crank alligned with the horizontal axis and the elastic deformations are zero.

6.4.2 Comparing the Solutions

Chu and Pan [57], and Song and Haug [6] use the Euler-Bernoulli beam model to represent the connecting rod, which is more than justified given the 48:1 length to height ratio of the beam and the lack of surface tractions.

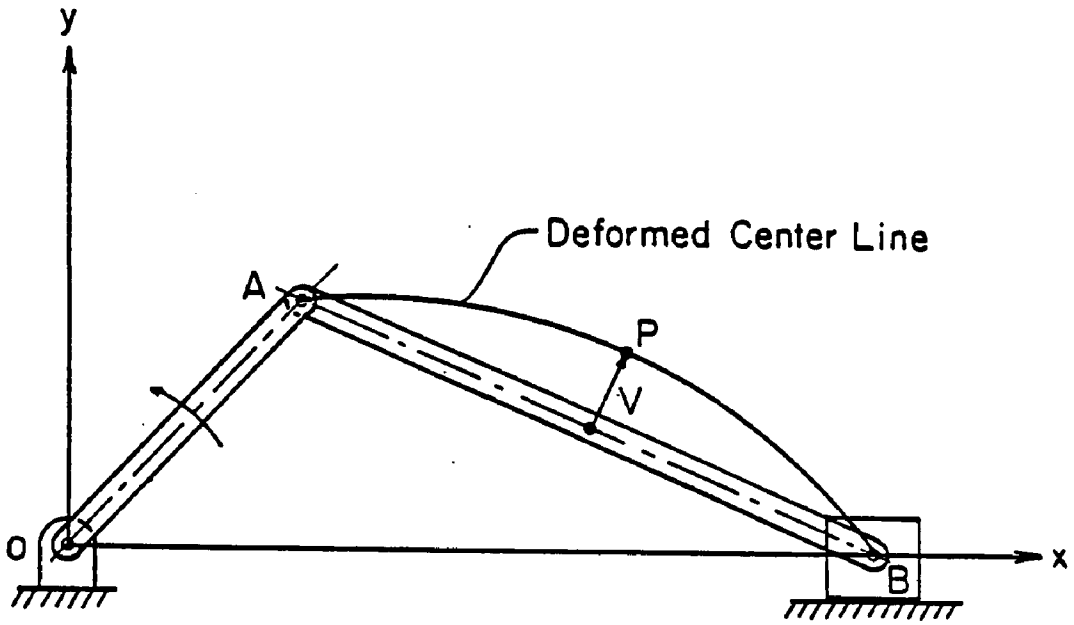


Fig. 6.18. Slider-crank mechanism, reprinted from [6].

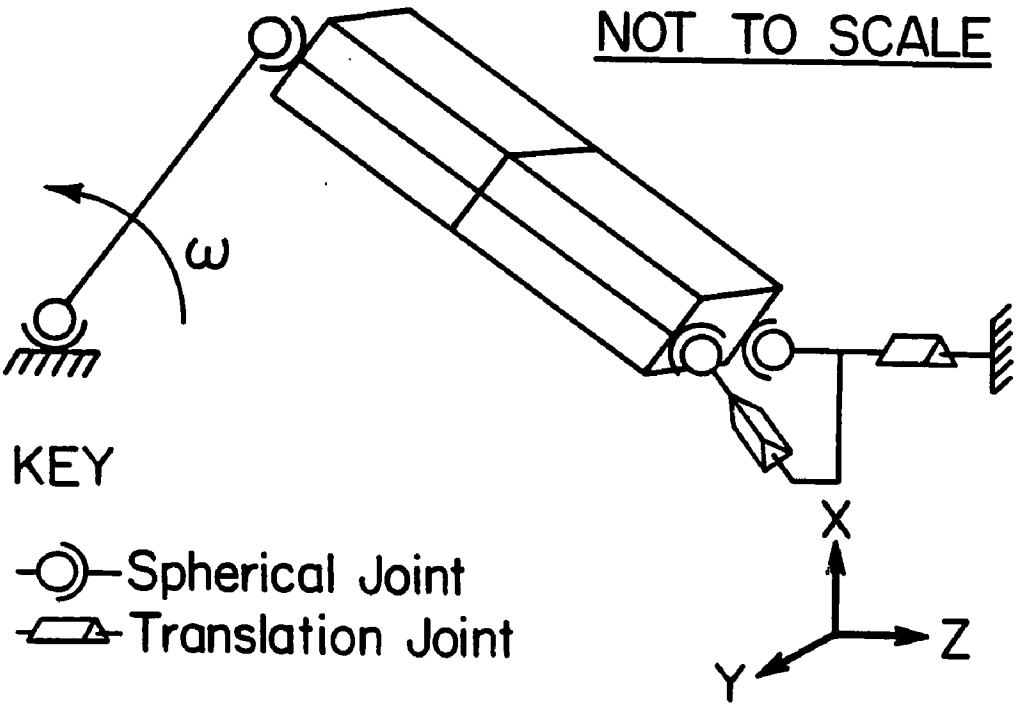


Fig. 6.19. Three dimensional slider crank model for DAVES.

The Song and Haug beam element is derived using the standard assumption of a cubic displacement function with its four coefficients determined by the deflections and rotations at the ends of the beam. Cases 1, 2, and 3 in Figure 6.20 are the solutions obtained by representing the connecting rod with one, two, and three beam elements, respectively. Note that the deformation of the midpoint of the beam is scaled by the length of the beam. As Song and Haug increased the number of beam elements, their solution approached the one given by Chu and Pan, and therefore, for the purpose of comparision, the solution of Chu and Pan is considered to be the exact solution.

Three cases were solved using DAVES with the beam represented with only a single mode; the cases differ only in the choice of the natural frequency and the mode shape. The first two cases assume that the ends of the beam are pinned. Case 1, the solid line in Figure 6.21, uses the theoretical first bending mode frequency of 867 radians/sec, while Case 2, the dash-dot-dash line, uses the frequency calculated by FEM, 971 radians/sec. Both cases use the eigenvector calculated by FEM, which agrees with the theoretical eigenvector for the particular lumped mass representation of the beam used in this analysis. The first free-free bending mode, with a frequency calculated by FEM to be 1372 radians/sec, was used in Case 3, the dashed line in Figure 6.2.2.

All of the solutions from DAVES have time signatures almost identical to those from the other solutions. Using the peak amplitudes as an accuracy criterion, Case 1 is obviously the best, agreeing very closely with the solution of Chu and Pan. Obviously the free-free mode is too stiff, and the theoretical value of the first bending mode, 1941 radians/sec, would result in a poor solution, indicating the danger of blindly relying only on the free-free modes to represent elastic bodies.

6.4.3 Computational Efficiency

The CPU time required for DAVES was relatively insensitive to the frequency of the modes. Each case was run for 0.070 seconds of simulation time with a corrector error criterion of 10^{-3}. A typical run required 4.7 CPU seconds for 430 corrector iterations, 144 time steps, and 160 Jacobian evaluations. Eliminating the output operations reduced the cost to 2.3 CPU seconds for 300 corrector iterations, 70 time steps, and 96 Jacobian evaluations.

6.5 Summary

DAVES was shown to be an accurate and efficient program for solving rigid body dynamics problems, linear vibration problems, and for problems combining rigid body motion and linearly elastic bodies by solving a problem from each category and comparing the solution to a known solution. The equations developed in Chapter 4 are therefore correct since DAVES is based on them.

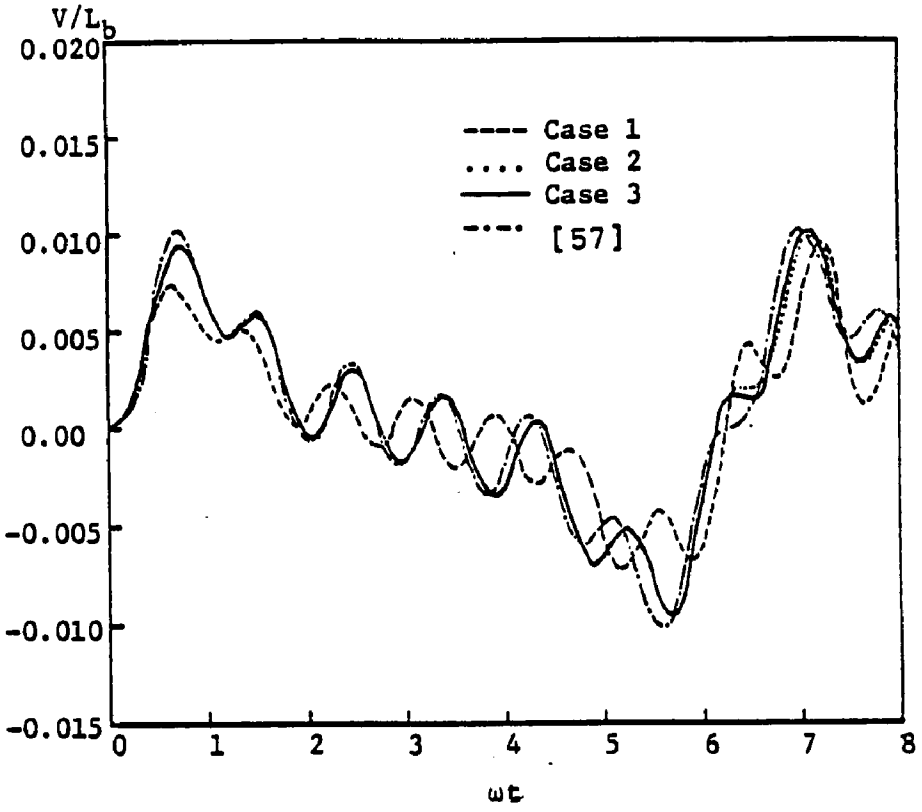


Fig. 6.20. Midpoint connecting rod deflection from [6].

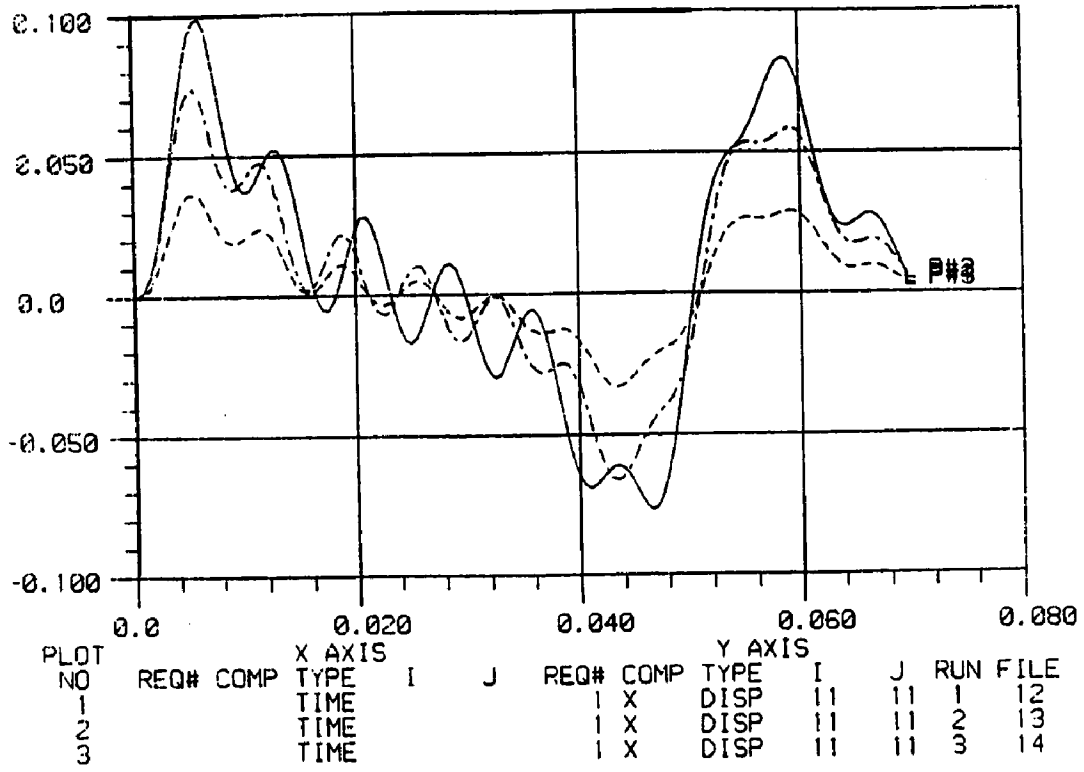


Fig. 6.21. Midpoint connecting rod deflection from DAVES

This chapter also showed the benefits of selective reduced integration with an eight node brick element. Solutions obtained with a crude mesh compare favorably with analytical solutions.

The last example in this chapter showed that very good accuracy is obtainable for some problems with only a single mode representing an elastic body provided that the mode is carefully chosen. Conversely, there are situations where the free-free modes, often used in the literature on deformable mechanisms, may not adequately represent the deformations of a body, indicating that programs and formulations requiring free-free modes are restricted in their application.

CHAPTER 7

CONCLUSIONS

7.1 A Summary of the Research

The equations of motion in three dimensions for constrained mechanical systems undergoing both large, nonlinear deformations and rigid body motion were developed from first principles by introducing quasi-coordinates and using the finite element method. After introducing the appropriate simplifications, the general formulation was specialized to rigid bodies and linearly elastic bodies. The equations for several types of applied forces and joints were also presented.

Realizing that simulating mechanical systems is computationally demanding, the equations of motion were carefully derived and structured to ensure that they could take full advantage of the computational speed of vector processors (Class VI computers), such as the Cray-1 or Cyber-205. This attention to vectorization is unique in the literature on mechanisms, although structural dynamics codes have exploited vectorization for several years.

A generalized, self-formulating program for three-dimensional systems, named DAVES, was developed and tested to demonstrate the utility of the general formulation.

Although it is limited to rigid and linearly elastic bodies, the program successfully demonstrates the general formulation since the specialized formulation for elastic bodies differs from the general one only by a linear transformation of coordinates. The program is unique among mechanisms programs for two reasons: 1) it is vectorized, and 2) it is the first program to incorporate Gear's stiff integration method for flexible, three-dimensional mechanical systems.

7.2 Conclusions

Removing the restriction that the deformations of a body are small and linear greatly increases the complexity of the equations of motion. Researchers such as de Veubeke, who take full advantage of the simplifying assumptions of linear elasticity, have already published formulations that should take no more than thirty percent longer (as opposed to the factor of 1.5~2 for DAVES) per integration corrector iteration than the equivalent rigid body formulation. When the deformations of a body are large and nonlinear, however, the many terms coupling the rigid body motion and the deformations can not be eliminated. Linear coordinate reduction, which is, of course, the single most effective method for reducing the problem size for linearly elastic problems, is inappropriate for finite strain problems.

Quasi-coordinates are a useful tool for formulating a comparatively simple set of equations of motion for deformable mechanical systems. Their implementation in a

generalized mechanisms program was simple; no numerical difficulties were encountered with them in any problem that ran on DAVES.

Formulating equations so that they are vectorizable forces the researcher to delve deeply into the underlying structure of a problem, and therefore vectorization has benefits other than just computational efficiency. Vectorized code is in many ways simpler and more structured than typical scalar code, making it simpler to understand and maintain.

7.3 Future Research and Development

Incorporating the general formulation into DAVES is the next logical step in the development of the program. Naturally, in addition to the finite element code necessary for handling large strains and nonlinear materials, the sparse matrix equation solver must be replaced with a highly efficient, vectorized equation solver, the block-oriented sparse solver probably being the most appropriate. The existing code could also benefit from some restructuring since it was developed without any access to a vector processor.

The preceding issues are more development than research. Certainly any research that benefits structural dynamics is beneficial to the work presented in this dissertation. Some of the more important areas are:

1) A better understanding of hourglass modes and reduced integration.

2) Much work remains in the area of inequality constraints and friction.
3) Improved accuracy is needed in highly distorted elements to alleviate the necessity of remeshing during the simulation.
4) Better nonlinear equation solution algorithms are necessary for solving problems with nonlinear materials, e.g., the post-yield unloading of a plastic material.
5) Last, but definitely not least, bigger and faster computers are necessary if the research in the above areas fails, and especially if it succeeds.

REFERENCES

REFERENCES

1. Benson, D.J. "The Simulation of Anelastic Mechanisms Using Vector Processors." Proc. ASME Second International Computer Conference, 2, Aug. 15-19, 1982.

2. Erdman, A. G. and Sandor, G. N. "Kineto-Elastodynamics -- A Review of the State of the Art and Trends." Mechanisms and Machine Theory 7 (1972): 19-33.

3. Winfrey, R. C. "Elastic Link Mechanism Dynamics." J. Eng. Ind. ASME B93 (1971): 268-272.

4. Erdman A. G., Sandor, G. N., Oakberg, R. G.. "A General Method fro Kineto-Elastodynamic Analysis and Synthesis of Mechanism." J. Eng. Ind. ASME B93 (1971): 1193-1205.

5. Baghat, B. M., Willmert, K. D. "Finite Element Vibrational Analysis of Planar Mechanisms." Mech. Mach. Theory 8 (1973): 497-516.

6. Song, J. C., Haug, E. J. "Dynamic Analysis of Planar Flexible Mechanisms." Computer Methods in Applied Mechanics and Engineering 24 (1980): 359-381.

7. Likins, P. W. "Dynamic Analysis of a System of Hinge Connected Rigid Bodies With Nonrigid Appendages." NASA Technical Report 32-1576 (Feb. 1974).

8. Boland, P., Samin, J. C., Willems, P. Y. "Stability Analysis of Interconnected Deformable Bodies With Closed Loop Configuration." AIAA Journal 13 (1975): 864-867.

9. de Veubeke, B. J. "The Dynamics of Flexible Bodies." Intl. J. of Eng. Science 14 (1976): 895-913.

10. Orlandea, N. "Node-Analogous Sparsity-Oriented Methods for the Simulation of Mechanical System." Diss. The University of Michigan, 1973.

11. Sunada, W., Dubowsky, S. "The Application of Finite Element Methods to the Dynamic Analysis of Flexible Spatial and Co-Planar Linkage Systems." ASME paper 80-DET-87, 1980.

12. Sheth P. N. "A Digital Computer Based Simulation Procedure for Multiple Degree of Freedom Mechanical Systems With Geometrical Constraints." Diss. University of Wisconsin, 1972.

13. Hurty, W. C. "Dynamic Analysis of Structural Systems Using Component Modes." AIAA Journal 3 (April 1965): 678-685.

14. Wehage, R. A., Haug, E. J. "Generalized Coordinate Partitioning for Dimension Reduction in Analysis of Constrained Dynamic Systems." Journal of Mechanical Design 104 (Jan. 1982): 247-255.

15. Shabana, A., Wehage, R. A. "Variable Degree-of-Freedom Component Mode Analysis of Inertia Variant Flexible Mechanical Systems." ASME paper 82-DET-93, 1982.

16. Hallquist, J. O. "Theoretical Manual for DYNA3D." Lawerence Livermore Laboratory report UCID-19401, June 1982.

17. Hallquist, J. O. "NIKE2D -- A Vectorized, Implicit, Finite Deformation, Finite Element Code for Analyzing the Static and Dynamic Response of 2-D Solids." Lawerence Livermore Laboratory report UCID-19677, Feb. 1983.

18. Calahan D. A. "Vector Algorithm Organization and Evaluation." The University of Michigan, 1980.

19. Calahan D. A. "High-Performance Banded and Profile Equation Solvers for the Cray-1, I. The Unsymmetric Case." Systems Engineering Laboratory Report No. 160, The University of Michigan, Feb. 1, 1982.

20. Calahan D. A. "Randomly Sparse Equation Solution by Loopless Code Generation on the Cray-1." Systems Engineering Laboratory Report No. 162, The University of Michigan, May 1, 1982.

21. Hallquist, J. O. "Recent Developments In Large-Scale Finite Element Lagrangian Hydrocode Technology." Computer Methods in Applied Mechanics and Engineering 33 (1982): 725-757.

22. Gear, C. W. Numerical Initial Value Problems in Ordinary Differential Equations. New York: Prentice-Hall, 1971.

23. Carnahan, B., Luther, H. A., Wilkes, J. O. Applied Numerical Methods. New York: John Wiley and Son, 1969.

24. Gear, C. W. "Simultaneous Numerical Solution of Differential-Algebraic Equations." IEEE Transactions on Circuit Theory CT-18, No. 1 (Jan. 1971): 89-95.

25. Petzold, L. R., Gear, C. W. "ODE Methods for the Solution of Differential/Algebraic Equations." Sandia Report SAND82-8051, Oct. 1982.

26. Benson D. "Performance Analysis of Four Computatonal Dynamics Algorithms." Proc. ASME Second International Computer Conference 1, Aug. 15-19, 1982.

27. Calahan, D. A. Computer-Aided Network Design. New York: McGraw-Hill, 1972.

28. Calahan, D. A. "Vectorized General Sparsity Algorithms with Backing Store." Systems Engineering Laboratory Report No. 96, The University of Michigan, Jan. 15, 1977.

29. Calahan, D. A. "Multi-Level Vectorized Sparse Solution of LSI Circuits." Proc. 1980 IEEE Conference on Circuits and Computers. 1980.

30. Calahan, D. A. "A Block-Oriented Equation Solver for the Cray-1." Systems Engineering Laboratory Report No. 136, The University of Michigan, Dec. 1, 1980.

31. Calahan, D. A. "A Vectorized General Sparsity Solver." Systems Engineering Laboratory Report No. 168., The University of Michigan, Oct. 1, 1982.

32. Zienkiewicz, O. C. The Finite Element Method. London: McGraw-Hill, 1979.

33. Turner, M. J., Clough, R. W., Martin, H. C., Topp, L. J. "Stiffness and Deflection Analysis of Complex Structures." Journal of Aeronautical Science 23 (1956): 805-823.

34. Rektorys, K. Variational Methods in Mathematical Science and Engineering. Boston: D. Reidel Publishing Company, 1980.

35. Malvern, L. E. Introduction to the Mechanics of a Continuous Medium. Englewood Cliffs: Prentice-Hall, Inc., 1969.

36. Nečas, J., Hlaváček, I. Mathematical Theory of Elastic and Elasto-Plastic Bodies: An Introduction. New York: Elsevier, 1981.

37. Bathe, K. Finite Element Procedures in Engineering Analysis. Englewood-Cliffs: Prentice-Hall, 1982.

38. Craig, R. R., Bampton, M. C. C. "Coupling of Substructures for Dynamic Analyses." AIAA Journal 6, No. 7 (July 1968): 1313-1319.

39. Kane, T. R., Levinson, D. A. "Formulation of Equations of Motion for Complex Spacecraft." Journal of Guidance and Control 3, No. 2 (March-April 1980): 99-112.

40. Canavin, J. R., Likins, P. W. "Floating Reference Frames for Flexible Spacecraft." Journal of Spacecraft 14, No. 12 (Dec. 1977): 724-732.

41. Buckens, F. "The Influence of Elastic Components on the Attitude Stability of a Satellite." Proc. Fifth International Symposium of Space Technology and Science, 1963, 193-203.

42. Ciarlet, P. G. The Finite Element Method for Elliptic Boundary Value Problems. New York: North-Holland, 1980.

43. Oden J. T. Finite Elements of Nonlinear Continua. New York: McGraw-Hill, 1972.

44. Truesdell, C., Noll, W. "The Nonlinear Field Theories of Mechanics." Handbuch der Physik III/3. Berlin: Springer-Verlag, 1965.

45. Meirovitch L., Methods of Analytical Dynamics. New York: McGraw-Hill, 1970.

46. Kikuchi, N., Oden, J. T. Contact Problems in Elasticity. Philadelphia: SIAM, 1984.

47. The ADAMS Users Guide. Ann Arbor: Mechanical Dynamics, Inc., 1979.

48. Wittenberg, J. Dynamics of Systems of Rigid Bodies. Stuttgart: B. G. Teubner, 1977.

49. Venkayya, V. B., Calahan, D. A., Summers, P. M. "Structural Optimization On Vector Processors." to be published.

50. Chace, M. A., Smith, D. A., "DAMN -- A Digital Computer Program for the Dynamic Analysis of Generalized Mechanical Systems." SAE Paper 710244, Jan. 1971.

51. Bathe, K. J., Wilson, E. L. "NONSAP -- A General Finite Element Program for Nonlinear Dynamic Analysis of Complex Structures." Proceedings, 2nd Int. Conf. on Structural Mechanics in Reactor Tech. Paper M3-1. Berlin, Sept. 1973.

52. Belytschko, T., Chiapetta, R. L., Bartel, H. D. "Efficient Large Scale Non-linear Transient Analysis by Finite Elements." Int. J. Num. Meth. in Eng. 10 (1976): 579-596.

53. Taylor, R. L., Beresford, P. J., Wilson, E. "A Non-conforming Element for Stress Analysis." The Mathematics of Finite Elements and Applications. New York: Academic Press, 1977.

54. Wilkins, M. L., Blum, R. E. "A Method for Computer Simulation of Problems in Solid Mechanics and Gas Dynamics in Three Dimensions and Time." University of California, Lawrence Livermore National Laboratory, Report UCRL-51574 Rev. 1 (1975).

55. Belytschko, T., Liu, W. K., Kennedy, J. M. "Hourglass Control in Linear and Nonlinear Problems." Recent Developments in Computer Methods for Nonlinear Solid and Structural Mechanics, ed. Alturi, S., Perrone, N., Houston: ASME/AMD, June 1983.

56. Meirovitch, L. Elements of Vibration Analysis. New York: McGraw-Hill, 1975.

57. Chu, S. C., Pan, K. C. "Dynamic Response of a High-Speed Slider-Crank Mechanism with an Elastic Connecting Rod." J. of Eng. for Ind., ASME B97, No. 2 (May 1975): 542-550.

58. Baumgarte, J. "Stabilization of Constraints and Integrals of Motion in Dynamic Systems." Computer Methods in Applied Mechanics and Engineering 1 (1972): 1-16.