

The Finite Element Method

Linear Static and Dynamic
Finite Element Analysis

Thomas J. R. Hughes

Professor of Mechanical Engineering
Chairman of the Division of Applied Mechanics
Stanford University

PRENTICE-HALL, INC., Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

HUGHES, THOMAS J. R.
The finite element method.

Bibliography.
Includes index.

1. Finite element method. 2. Boundary value
problems. I. Title.
TA347.F5H84 1987 620'.0042 86-22558
ISBN 0-13-317025-X

Editorial/production supervision

and interior design: Joan McCulley

Cover sketch: M. Igarashi and K. Shimomaki
Suzuki Motor Co., Ltd.

Cover design: 20/20 Services, Inc.

Manufacturing buyer: Rhett Conklin

©1987 by Prentice-Hall, Inc.

A Division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

*All rights reserved. No part of this book may be
reproduced, in any form or by any means,
without permission in writing from the publisher.*

Printed in the United States of America

10 9 8 7 6 5 4 3

ISBN 0-13-317025-X 025

PRENTICE-HALL INTERNATIONAL (UK) LIMITED, London

PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, Sydney

PRENTICE-HALL CANADA INC., Toronto

PRENTICE-HALL HISPANOAMERICANA, S. A., Mexico

PRENTICE-HALL OF INDIA PRIVATE LIMITED, New Delhi

PRENTICE-HALL OF JAPAN, INC., Tokyo

PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., Singapore

EDITORA PRENTICE-HALL DO BRASIL, LTDA., Rio de Janeiro

To my wife,
my mother,
and my children

Contents

| | |
|-------------------------------|------|
| PREFACE | XV |
| A BRIEF GLOSSARY OF NOTATIONS | XXII |

Part One Linear Static Analysis

| | |
|--|----|
| 1 FUNDAMENTAL CONCEPTS; A SIMPLE ONE-DIMENSIONAL BOUNDARY-VALUE PROBLEM | 1 |
| 1.1 Introductory Remarks and Preliminaries | 1 |
| 1.2 Strong, or Classical, Form of the Problem | 2 |
| 1.3 Weak, or Variational, Form of the Problem | 3 |
| 1.4 Equivalence of Strong and Weak Forms; Natural Boundary Conditions | 4 |
| 1.5 Galerkin's Approximation Method | 7 |
| 1.6 Matrix Equations; Stiffness Matrix K | 9 |
| 1.7 Examples: 1 and 2 Degrees of Freedom | 13 |
| 1.8 Piecewise Linear Finite Element Space | 20 |
| 1.9 Properties of K | 22 |
| 1.10 Mathematical Analysis | 24 |
| 1.11 Interlude: Gauss Elimination; Hand-calculation Version | 31 |
| 1.12 The Element Point of View | 37 |
| 1.13 Element Stiffness Matrix and Force Vector | 40 |
| 1.14 Assembly of Global Stiffness Matrix and Force Vector; LM Array | 42 |

| | | |
|--------------|--|------------|
| 1.15 | Explicit Computation of Element Stiffness Matrix and Force Vector | 44 |
| 1.16 | Exercise: Bernoulli-Euler Beam Theory and Hermite Cubics | 48 |
| Appendix 1.I | An Elementary Discussion of Continuity, Differentiability, and Smoothness | 52 |
| | References | 55 |
| 2 | FORMULATION OF TWO- AND THREE-DIMENSIONAL BOUNDARY-VALUE PROBLEMS | 57 |
| 2.1 | Introductory Remarks | 57 |
| 2.2 | Preliminaries | 57 |
| 2.3 | Classical Linear Heat Conduction: Strong and Weak Forms; Equivalence | 60 |
| 2.4 | Heat Conduction: Galerkin Formulation; Symmetry and Positive-definiteness of K | 64 |
| 2.5 | Heat Conduction: Element Stiffness Matrix and Force Vector | 69 |
| 2.6 | Heat Conduction: Data Processing Arrays ID, IEN, and LM | 71 |
| 2.7 | Classical Linear Elastostatics: Strong and Weak Forms; Equivalence | 75 |
| 2.8 | Elastostatics: Galerkin Formulation, Symmetry, and Positive-definiteness of K | 84 |
| 2.9 | Elastostatics: Element Stiffness Matrix and Force Vector | 90 |
| 2.10 | Elastostatics: Data Processing Arrays ID, IEN, and LM | 92 |
| 2.11 | Summary of Important Equations for Problems Considered in Chapters 1 and 2 | 98 |
| 2.12 | Axisymmetric Formulations and Additional Exercises | 101 |
| | References | 107 |
| 3 | ISOPARAMETRIC ELEMENTS AND ELEMENTARY PROGRAMMING CONCEPTS | 109 |
| 3.1 | Preliminary Concepts | 109 |
| 3.2 | Bilinear Quadrilateral Element | 112 |
| 3.3 | Isoparametric Elements | 118 |
| 3.4 | Linear Triangular Element; An Example of “Degeneration” | 120 |
| 3.5 | Trilinear Hexahedral Element | 123 |
| 3.6 | Higher-order Elements; Lagrange Polynomials | 126 |
| 3.7 | Elements with Variable Numbers of Nodes | 132 |

Contents

| | | |
|---------------|--|-----|
| 3.8 | Numerical Integration; Gaussian Quadrature | 137 |
| 3.9 | Derivatives of Shape Functions and Shape Function Subroutines | 146 |
| 3.10 | Element Stiffness Formulation | 151 |
| 3.11 | Additional Exercises | 156 |
| Appendix 3.I | Triangular and Tetrahedral Elements | 164 |
| Appendix 3.II | Methodology for Developing Special Shape Functions with Application to Singularities | 175 |
| | References | 182 |
| 4 | MIXED AND PENALTY METHODS, REDUCED AND SELECTIVE INTEGRATION, AND SUNDRY VARIATIONAL CRIMES | 185 |
| 4.1 | “Best Approximation” and Error Estimates: Why the standard FEM usually works and why sometimes it does not | 185 |
| 4.2 | Incompressible Elasticity and Stokes Flow | 192 |
| 4.2.1 | Prelude to Mixed and Penalty Methods | 194 |
| 4.3 | A Mixed Formulation of Compressible Elasticity Capable of Representing the Incompressible Limit | 197 |
| 4.3.1 | Strong Form | 198 |
| 4.3.2 | Weak Form | 198 |
| 4.3.3 | Galerkin Formulation | 200 |
| 4.3.4 | Matrix Problem | 200 |
| 4.3.5 | Definition of Element Arrays | 204 |
| 4.3.6 | Illustration of a Fundamental Difficulty | 207 |
| 4.3.7 | Constraint Counts | 209 |
| 4.3.8 | Discontinuous Pressure Elements | 210 |
| 4.3.9 | Continuous Pressure Elements | 215 |
| 4.4 | Penalty Formulation: Reduced and Selective Integration Techniques; Equivalence with Mixed Methods | 217 |
| 4.4.1 | Pressure Smoothing | 226 |
| 4.5 | An Extension of Reduced and Selective Integration Techniques | 232 |
| 4.5.1 | Axisymmetry and Anisotropy: Prelude to Nonlinear Analysis | 232 |
| 4.5.2 | Strain Projection: The \bar{B} -approach | 232 |
| 4.6 | The Patch Test; Rank Deficiency | 237 |
| 4.7 | Nonconforming Elements | 242 |
| 4.8 | Hougaard Stiffness | 251 |
| 4.9 | Additional Exercises and Projects | 254 |
| Appendix 4.I | Mathematical Preliminaries | 263 |
| 4.I.1 | Basic Properties of Linear Spaces | 263 |
| 4.I.2 | Sobolev Norms | 266 |
| 4.I.3 | Approximation Properties of Finite Element Spaces in Sobolev Norms | 268 |

| | | |
|---------------|---|-----|
| 4.I.4 | Hypotheses on $a(\cdot, \cdot)$ | 273 |
| Appendix 4.II | Advanced Topics in the Theory of Mixed and Penalty Methods: Pressure Modes and Error Estimates by David S. Malkus | 276 |
| 4.II.1 | Pressure Modes, Spurious and Otherwise | 276 |
| 4.II.2 | Existence and Uniqueness of Solutions in the Presence of Modes | 278 |
| 4.II.3 | Two Sides of Pressure Modes | 281 |
| 4.II.4 | Pressure Modes in the Penalty Formulation | 289 |
| 4.II.5 | The Big Picture | 292 |
| 4.II.6 | Error Estimates and Pressure Smoothing | 297 |
| | References | 303 |
| 5 | THE C ⁰ -APPROACH TO PLATES AND BEAMS | 310 |
| 5.1 | Introduction | 310 |
| 5.2 | Reissner-Mindlin Plate Theory | 310 |
| 5.2.1 | Main Assumptions | 310 |
| 5.2.2 | Constitutive Equation | 313 |
| 5.2.3 | Strain-displacement Equations | 313 |
| 5.2.4 | Summary of Plate Theory Notations | 314 |
| 5.2.5 | Variational Equation | 314 |
| 5.2.6 | Strong Form | 317 |
| 5.2.7 | Weak Form | 317 |
| 5.2.8 | Matrix Formulation | 319 |
| 5.2.9 | Finite Element Stiffness Matrix and Load Vector | 320 |
| 5.3 | Plate-bending Elements | 322 |
| 5.3.1 | Some Convergence Criteria | 322 |
| 5.3.2 | Shear Constraints and Locking | 323 |
| 5.3.3 | Boundary Conditions | 324 |
| 5.3.4 | Reduced and Selective Integration Lagrange Plate Elements | 327 |
| 5.3.5 | Equivalence with Mixed Methods | 330 |
| 5.3.6 | Rank Deficiency | 332 |
| 5.3.7 | The Heterosis Element | 335 |
| 5.3.8 | T1: A Correct-rank, Four-node Bilinear Element | 342 |
| 5.3.9 | The Linear Triangle | 355 |
| 5.3.10 | The Discrete Kirchhoff Approach | 359 |
| 5.3.11 | Discussion of Some Quadrilateral Bending Elements | 362 |
| 5.4 | Beams and Frames | 363 |
| 5.4.1 | Main Assumptions | 363 |
| 5.4.2 | Constitutive Equation | 365 |
| 5.4.3 | Strain-displacement Equations | 366 |

Contents

| | | |
|--------|---|-----|
| 5.4.4 | Definitions of Quantities Appearing in the Theory | 366 |
| 5.4.5 | Variational Equation | 368 |
| 5.4.6 | Strong Form | 371 |
| 5.4.7 | Weak Form | 372 |
| 5.4.8 | Matrix Formulation of the Variational Equation | 373 |
| 5.4.9 | Finite Element Stiffness Matrix and Load Vector | 374 |
| 5.4.10 | Representation of Stiffness and Load in Global Coordinates | 376 |
| 5.5 | Reduced Integration Beam Elements | 376 |
| | References | 379 |

6 THE C⁰-APPROACH TO CURVED STRUCTURAL ELEMENTS 383

| | | |
|--------|--|-----|
| 6.1 | Introduction | 383 |
| 6.2 | Doubly Curved Shells in Three Dimensions | 384 |
| 6.2.1 | Geometry | 384 |
| 6.2.2 | Lamina Coordinate Systems | 385 |
| 6.2.3 | Fiber Coordinate Systems | 387 |
| 6.2.4 | Kinematics | 388 |
| 6.2.5 | Reduced Constitutive Equation | 389 |
| 6.2.6 | Strain-displacement Matrix | 392 |
| 6.2.7 | Stiffness Matrix | 396 |
| 6.2.8 | External Force Vector | 396 |
| 6.2.9 | Fiber Numerical Integration | 398 |
| 6.2.10 | Stress Resultants | 399 |
| 6.2.11 | Shell Elements | 399 |
| 6.2.12 | Some References to the Recent Literature | 403 |
| 6.2.13 | Simplifications: Shells as an Assembly of Flat Elements | 404 |
| 6.3 | Shells of Revolution; Rings and Tubes in Two Dimensions | 405 |
| 6.3.1 | Geometric and Kinematic Descriptions | 405 |
| 6.3.2 | Reduced Constitutive Equations | 407 |
| 6.3.3 | Strain-displacement Matrix | 409 |
| 6.3.4 | Stiffness Matrix | 412 |
| 6.3.5 | External Force Vector | 412 |
| 6.3.6 | Stress Resultants | 413 |
| 6.3.7 | Boundary Conditions | 414 |
| 6.3.8 | Shell Elements | 414 |
| | References | 415 |

Part Two Linear Dynamic Analysis

| | |
|--|------------|
| 7 FORMULATION OF PARABOLIC, HYPERBOLIC, AND ELLIPTIC-EIGENVALUE PROBLEMS | 418 |
| 7.1 Parabolic Case: Heat Equation | 418 |
| 7.2 Hyperbolic Case: Elastodynamics and Structural Dynamics | 423 |
| 7.3 Eigenvalue Problems: Frequency Analysis and Buckling | 429 |
| 7.3.1 Standard Error Estimates | 433 |
| 7.3.2 Alternative Definitions of the Mass Matrix; Lumped and Higher-order Mass | 436 |
| 7.3.3 Estimation of Eigenvalues | 452 |
| Appendix 7.I Error Estimates for Semidiscrete Galerkin Approximations | 456 |
| References | 457 |
| 8 ALGORITHMS FOR PARABOLIC PROBLEMS | 459 |
| 8.1 One-step Algorithms for the Semidiscrete Heat Equation: Generalized Trapezoidal Method | 459 |
| 8.2 Analysis of the Generalized Trapezoidal Method | 462 |
| 8.2.1 Modal Reduction to SDOF Form | 462 |
| 8.2.2 Stability | 465 |
| 8.2.3 Convergence | 468 |
| 8.2.4 An Alternative Approach to Stability: The Energy Method | 471 |
| 8.2.5 Additional Exercises | 473 |
| 8.3 Elementary Finite Difference Equations for the One-dimensional Heat Equation; the von Neumann Method of Stability Analysis | 479 |
| 8.4 Element-by-element (EBE) Implicit Methods | 483 |
| 8.5 Modal Analysis | 487 |
| References | 488 |
| 9 ALGORITHMS FOR HYPERBOLIC AND PARABOLIC-HYPERBOLIC PROBLEMS | 490 |
| 9.1 One-step Algorithms for the Semidiscrete Equation of Motion | 490 |
| 9.1.1 The Newmark Method | 490 |
| 9.1.2 Analysis | 492 |
| 9.1.3 Measures of Accuracy: Numerical Dissipation and Dispersion | 504 |
| 9.1.4 Matched Methods | 505 |
| 9.1.5 Additional Exercises | 512 |

Contents

| | | |
|---------|---|-----|
| 9.2 | Summary of Time-step Estimates for Some Simple Finite Elements | 513 |
| 9.3 | Linear Multistep (LMS) Methods | 523 |
| 9.3.1 | LMS Methods for First-order Equations | 523 |
| 9.3.2 | LMS Methods for Second-order Equations | 526 |
| 9.3.3 | Survey of Some Commonly Used Algorithms in Structural Dynamics | 529 |
| 9.3.4 | Some Recently Developed Algorithms for Structural Dynamics | 550 |
| 9.4 | Algorithms Based upon Operator Splitting and Mesh Partitions | 552 |
| 9.4.1 | Stability via the Energy Method | 556 |
| 9.4.2 | Predictor/Multicorrector Algorithms | 562 |
| 9.5 | Mass Matrices for Shell Elements | 564 |
| | References | 567 |
| 10 | SOLUTION TECHNIQUES FOR EIGENVALUE PROBLEMS | 570 |
| 10.1 | The Generalized Eigenproblem | 570 |
| 10.2 | Static Condensation | 573 |
| 10.3 | Discrete Rayleigh-Ritz Reduction | 574 |
| 10.4 | Irons-Guyan Reduction | 576 |
| 10.5 | Subspace Iteration | 576 |
| 10.5.1 | Spectrum Slicing | 578 |
| 10.5.2 | Inverse Iteration | 579 |
| 10.6 | The Lanczos Algorithm for Solution of Large Generalized Eigenproblems | 582 |
| | by Bahram Nour-Omid | |
| 10.6.1 | Introduction | 582 |
| 10.6.2 | Spectral Transformation | 583 |
| 10.6.3 | Conditions for Real Eigenvalues | 584 |
| 10.6.4 | The Rayleigh-Ritz Approximation | 585 |
| 10.6.5 | Derivation of the Lanczos Algorithm | 586 |
| 10.6.6 | Reduction to Tridiagonal Form | 589 |
| 10.6.7 | Convergence Criterion for Eigenvalues | 592 |
| 10.6.8 | Loss of Orthogonality | 595 |
| 10.6.9 | Restoring Orthogonality | 598 |
| 10.6.10 | LANSEL Package | 600 |
| | References | 629 |
| 11 | DLEARN—A LINEAR STATIC AND DYNAMIC FINITE ELEMENT ANALYSIS PROGRAM | 631 |
| | by Thomas J. R. Hughes, Robert M. Ferencz, and Arthur M. Raefsky | |

| | | |
|--------|---|-----|
| 11.1 | Introduction | 631 |
| 11.2 | Description of Coding Techniques Used in DLEARN | 632 |
| 11.2.1 | Compacted Column Storage Scheme | 633 |
| 11.2.2 | Crout Elimination | 636 |
| 11.2.3 | Dynamic Storage Allocation | 644 |
| 11.3 | Program Structure | 650 |
| 11.3.1 | Global Control | 651 |
| 11.3.2 | Initialization Phase | 651 |
| 11.3.3 | Solution Phase | 653 |
| 11.4 | Adding an Element to DLEARN | 659 |
| 11.5 | DLEARN User's Manual | 662 |
| 11.5.1 | Remarks for the New User | 662 |
| 11.5.2 | Input Instructions | 663 |
| 11.5.3 | Examples | 691 |
| 1. | Planar Truss | 691 |
| 2. | Static Analysis of a Plane Strain Cantilever Beam | 705 |
| 3. | Dynamic Analysis of a Plane Strain Cantilever Beam | 705 |
| 4. | Implicit-explicit Dynamic Analysis of a Rod | 715 |
| 11.5.4 | Subroutine Index for Program Listing | 729 |
| 11.5.5 | Program Listing | 734 |
| | References | 796 |
| INDEX | | 797 |

Preface

This book is based on courses that I have taught at the California Institute of Technology and Stanford University during the last 10 years. It deals with the finite element method in linear static and dynamic analysis. It is intended primarily for engineering and physical science students who wish to develop comprehensive skills in finite element methodology, from fundamental concepts to practical computer implementations.

Some sections of this book touch upon the frontiers of research and have been used as lecture notes in a number of more advanced short courses I have taught in Europe, Japan, and the United States. Consequently, I believe it will also be of interest to more experienced analysts and researchers working in the finite element field.

SUBJECTS COVERED

The first chapter of the book introduces the finite element method in the context of simple one-dimensional model problems. Chapter 2 deals with variational formulations of two- and three-dimensional boundary-value problems in heat conduction (in fact, all problems governed by Laplace/Poisson equations, such as electrostatics, potential flow, elastic membranes, and flow in porous media) and elasticity theory. These serve as the basis for finite element discretization and the techniques developed illustrate the relationship between "strong," or classical, statements of boundary-value problems and their "weak," or variational, counterparts. The Galerkin method of approximate solution is emphasized in Chapter 2 and throughout the book, rather than "variational principles" due to the significantly greater generality of the former approach. In Chapter 3 a variety of finite element interpolatory schemes are developed.

These apply to triangles, quadrilaterals, tetrahedra, hexahedra, wedges, etc. The isoparametric concept is emphasized, and special-purpose interpolatory strategies are also developed (e.g., "singular elements"). Programming techniques are introduced for numerically integrated finite elements. Chapter 4 deals with basic error estimates for standard "displacement" finite element methods and introduces mixed methods for constrained media applications such as incompressible elasticity and Stokes flow. A variety of "variational crimes" are described: for example, incompatible elements, reduced and selective integration, strain projection (i.e., \bar{B} -) methods, etc. (Most of these have been decriminalized in recent years.) The mathematical analysis of finite element methods for incompressible media is rather complex. David Malkus, an authority on this subject, explains some of the subtleties in an appendix to Chapter 4. Chapter 5 is concerned with finite element methods for Reissner-Mindlin plates and elastic frameworks composed of straight beam elements accounting for axial, torsional, bending, and transverse shear deformations (i.e., Timoshenko beams). Chapter 6 deals with three-dimensional curved shell elements and two-dimensional special cases such as rings, tubes, and shells of revolution. The problem classes discussed in Chapters 1–6 give rise to associated eigenvalue problems and initial-value problems. The formulations of problems of these types are the subjects of Chapter 7. Chapter 8 presents time-stepping algorithms for first-order ordinary differential equation systems such as those arising from unsteady heat conduction ("parabolic case"). Chapter 9 deals with algorithms for second-order ordinary differential equation systems such as those emanating from elastodynamics and structural dynamics ("hyperbolic case"). Chapter 10 presents basic algorithmic strategies for symmetric elliptic eigenvalue problems such as those encountered in free vibration and structural stability. A very efficient major software package for matrix eigenvalue and eigenvector calculations based on the Lanczos method is also presented in Chapter 10. The documentation of the Lanczos algorithm and software were written by Bahram Nour-Omid, an expert on procedures of this type. Chapter 11 presents a linear static and dynamic analysis computer program based on the methods developed in the book. My student, Robert Ferencz, and colleague, Arthur Raefsky, collaborated with me in the writing of the program. The program is named DLEARN and it contains a very complete library of finite element software tools. This program is suitable for homework assignments, projects (e.g., programming additional elements), and research studies. DLEARN is highly structured for readability, maintainability, and extendability and has been written specifically to complement and enhance the procedures described in the remainder of the book.

LEVEL AND BACKGROUND

This book is primarily intended for the graduate level, although advanced undergraduates will find much of it accessible. An undergraduate degree in engineering, mathematics, computer science, or any of the physical sciences constitutes essential background. Courses in applied linear algebra and elementary ordinary and partial differential equations are desirable prerequisites. A working knowledge of FORTRAN

(the dominant language used in finite element programming) is also necessary for understanding the software presented in Chapters 10 and 11.

The book emphasizes heat conduction and elasticity as primary vehicles for developing finite element methods because of the widespread interest in these theories in the applied sciences. By virtue of the fact that the partial differential equation of heat conduction, namely, the Laplace/Poisson equation, appears under different names in virtually all branches of engineering and physics, most students have had some familiarity with it. Some exposure to the theory of elasticity is also desirable. This can be obtained, for example, in a good advanced course on strength of materials. Students at Caltech and Stanford frequently have taken courses in elasticity simultaneously with finite elements. Background in structural mechanics (i.e., the theory of beams, plates, and shells) is certainly an asset when it comes to studying this book but is not essential. Only Chapters 5 and 6 deal exclusively with this subject, and these chapters may be ignored by students whose primary interests lie elsewhere. It is worth noting that students who have taken this finite element course at Stanford and Caltech have had very diverse backgrounds (e.g., geophysics, chemical engineering, planetary sciences, coastal engineering, electrical engineering, computer science, mathematics, bioengineering, aeronautics and astronautics, material science, physics, earth sciences, environmental engineering, biomechanics, thermosciences, engineering design, applied mechanics, earthquake engineering, and, of course, mechanical and civil engineering). In this spirit the book emphasizes fundamental finite element concepts and techniques applicable to a very broad range of problems and thus constitutes a suitable text for most students in the physical sciences.

It is my experience that most finite element methodology can and should be taught initially within the confines of linear behavior before introducing nonlinear effects. A solid understanding of finite elements in linear analysis is, of course, very useful in its own right. In addition, it makes the subject of nonlinear finite element analysis much more accessible.

WHAT IS UNIQUE ABOUT THIS BOOK

Although this book deals with many standard aspects of the finite element method, there are many unique features, some of which are enumerated below.

- A comprehensive presentation and analysis of algorithms for time-dependent phenomena.
- Beam, plate, and shell theories are derived directly from three-dimensional elasticity theory.
- An extensive static and dynamic finite element analysis program, DLEARN, was specially prepared based on the text material. It is written using structured programming concepts and contains many advanced procedures.
- Although written for students without serious mathematical training, the book contains introductory material on the mathematical theory of finite elements and

many important mathematical results. It thus serves as a primer for more advanced works on this subject.

- A systematic treatment of “weak,” or variational, formulations for various classes of initial- and boundary-value problems.
- Many of the procedures described are presented in book form for the first time, for example, strain projection (i.e., \bar{B} -) methods, implicit-explicit finite element mesh partitions in transient analysis, element-by-element iterative solvers, complete computer implementation of predictor/multicorrector implicit-explicit time-stepping algorithms based upon the Hilber-Hughes-Taylor α -method, etc.
- A Lanczos software package for eigenvalue/eigenvector extraction.

NOTATIONAL CONVENTIONS

As far as possible, we have adopted fairly standard notations. Vectors and matrices are denoted by boldface italic characters. Scalars and components of vectors, tensors, and matrices are denoted by lightface italic characters. Cartesian tensor notation is used in the presentation of heat conduction, elasticity, plates, etc. The symbol ■ is used to denote the end of a proof. Unavoidably, in a book of this size and scope there are some minor conflicts of notation. In instances when this occurs, it is hoped that surrounding explanation and context will make the intended meaning clear. All notations are defined when they are introduced. Principal notations are summarized in a brief glossary following the Preface.

NUMBERING CONVENTIONS

The book is divided into two parts. Chapters 1–6 deal with static analysis, and Chapters 7–11 deal with dynamic analysis. The chapters are subdivided into sections and several of the chapters include appendices. Some of the sections are further subdivided into subsections. Equations, tables, and figures are numbered consecutively within each section, with each number consisting of chapter, section, and item number. Exercises are numbered consecutively beginning with 1 in each section. References are numbered consecutively at the end of each chapter. Equation numbers are enclosed in parentheses, and reference numbers appearing in the text are enclosed in square brackets. If, for example, we wish to refer to Eq. 8 of Section 3, Chapter 4, then we would write this as (4.3.8).

If we wish to draw the reader’s attention to a subsection, then the section number is followed by a decimal point and the subsection number. For example, subsection 2 of Section 3, Chapter 5, is written as Sec. 5.3.2.

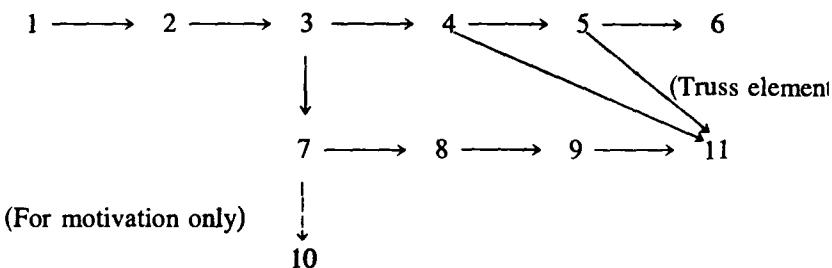
EXERCISES

Exercises appear throughout the text. In some cases answers or complete solutions are provided. Many of the exercises are placed in the proximity of the pertinent text material rather than at the ends of sections or chapters.

INTERDEPENDENCE OF CHAPTERS

Prior to consulting particular chapters it may be worthwhile to peruse the “reading paths,” which indicate the interdependence of the chapters. Of course, if one has sufficient background, the suggested reading paths may be bypassed. Knowledge of Chapters 1–3 is assumed for all subsequent chapters with the exception of Chapter 10, which is essentially self-contained.

Chapter Reading Paths



A NOTE FOR INSTRUCTORS REGARDING THE ORGANIZATION OF COURSES

Several courses can be developed based on the material in this book. Here are a few examples.

Almost the entire book can be covered in a one-year course:

Title: Linear Static Finite Element Analysis (1 semester)

Syllabus: Chapters 1–3, selected topics from Chapters 4–6, and computing assignments using DLEARN (Chapter 11).

Title: Linear Dynamic Finite Element Analysis (1 semester)

Syllabus: Chapters 7–10 and computing assignments using DLEARN (Chapter 11)

An abbreviated version of the above covering two quarters can be organized as follows:

Title: Linear Static Finite Element Analysis (1 quarter)

Syllabus: Chapters 1–3, selected topics from Chapter 4, and computing assignments using DLEARN (Chapter 11).

Title: Linear Dynamic Finite Element Analysis (1 quarter)

Syllabus: Sections 1–3 of Chapter 7; Sections 1, 2, and 5 of Chapter 8; Sections 1–4 of Chapter 9; Sections 1–5 of Chapter 10; and computing assignments using DLEARN (Chapter 11).

A course on finite element programming can be taught in one quarter or one semester:

Title: Finite Element Programming

Syllabus: Chapter 11 and selected sections from earlier chapters as background material. Lectures should describe overall program architecture (subroutines DLEARN and DRIVER), the structure of element routines (e.g., QUADC and subroutines called by QUADC), equation-solving (subroutines FACTOR and BACK), assembly routines (ADDLHS and ADDRHS), and important finite element utility routines (BC, COLHT, DCTNRY, DIAG, EQSET, FORMLM, LOCAL, and MPOINT). Programming assignments, such as adding new elements and extending capabilities, are essential.

ACKNOWLEDGMENTS

I would like to thank many former and current students, colleagues, and friends for their constructive comments and criticisms, especially Ted Belytschko, Alec Brooks, Marty Cohen, Leo Franca, Hans Hilber, Greg Hulbert, Itzhak Levit, Wing Kam Liu, Michel Mallet, Jerry Marsden, Isidoro Miranda, Arthur Muller, Juan Simo, Gary Stanley, Tayfun Tezduyar, Jim Winget, and Thomas Zimmermann.

I would also like to express my appreciation to those individuals who have made contributions to the writing of this book: Bob Ferencz, David Malkus, Bahram Nour-Omid, and Arthur Raefsky.

In addition, I would like to thank Alex Tessler who prepared notes for me on some recent developments in plate elements, which I used as the basis of comments included in Chapter 5. Wanda Gooden typed most of the manuscript. I wish to thank her for her patience, dedication, and good-naturedness in the face of a seemingly infinite number of corrections.

I would like to acknowledge the following organizations for permission to reprint various materials: the American Society of Mechanical Engineers for Figs. 5.3.21–5.3.31; John Wiley and Sons, Ltd., for Figs. 3.II.1, 3.II.2, 9.3.1–9.3.3, 9.3.5, and Table 3.I.1; Pergamon Journals, Ltd., for Figs. 5.3.11, 5.3.13–5.3.15, 5.3.18–5.3.20; and North-Holland Publishing Co. for portions of Chapters 8 and 9, which originally appeared in T. J. R. Hughes, “Analysis of Transient Algorithms with Particular Reference to Stability,” pp. 67–155, *Computational Methods for Transient Analysis* (eds. T. Belytschko and T. J. R. Hughes), 1983.

Finally, I would like to thank M. Igarashi and K. Shimomaki of the Suzuki Motor Co., Ltd., for providing the mesh for the cover, and Dave Benson and John Hallquist of Lawrence Livermore National Laboratory for the accompanying color graphics.

NOTE: The software in this book is available on diskette and magnetic tape for several computers. Further information and an order form may be obtained by writing to the author: Professor Thomas J. R. Hughes, Division of Applied Mechanics, Durand Building, Stanford University, Stanford, California 94305.

A Brief Glossary of Notations

| Sets | | n_{dof} | number of degrees of freedom |
|-------------------------|---|-----------------------|--|
| \mathbb{R} | real numbers | n_{ed} | number of element degrees of freedom |
| \mathbb{C} | complex numbers | | number of elements |
| \cup | set union | n_{el} | global node numbers $(1 \leq A, B \leq n_{np})$ |
| \cap | set intersection | A, B | element node numbers |
| \emptyset | empty set | | $(1 \leq a, b \leq n_{en})$ |
| \in | is a member of; belongs to | a, b | global equation numbers |
| \notin | is not a member of; does not belong to | P, Q | $(1 \leq P, Q \leq n_{eq})$ |
| \subset | is a subset of | | element equation numbers |
| $\not\subset$ | is not a subset of | | $(1 \leq p, q \leq n_{ee})$ |
| \forall | for all; for each | p, q | element number $(1 \leq e \leq n_{el})$ |
| Various Integers | | e | |
| n_{np} | number of nodal points | | |
| n_{en} | number of element nodes | Element Arrays | mass matrix of the e th element |
| n_{eq} | number of equations | m^e | damping matrix of the e th element |
| n_{ee} | number of element equations | c^e | |

| | | | |
|-------|---|--------------------------------|---|
| k^e | stiffness matrix of the e th element | Γ_k, Γ_{k_i} | part of the boundary where Neumann conditions are specified |
| f^e | force vector of the e th element | | Kronecker delta |
| a^e | acceleration vector of the e th element | δ_{ij} | set of node numbers |
| v^e | velocity vector of the e th element | η | set of node numbers at which Dirichlet conditions are specified |
| d^e | displacement vector of the e th element | $\eta_\alpha, \eta_{\alpha_i}$ | |

Global Arrays

| | | | |
|--------------|---|---|---|
| M | mass matrix | Variational Methods | |
| C | damping matrix | h | mesh parameter |
| K | stiffness matrix | $\mathcal{S}, \mathcal{S}_i, \mathcal{S}$ | collections of trial solutions |
| F | force vector | $\mathcal{V}, \mathcal{V}_i, \mathcal{V}$ | collections of weighting functions |
| a | acceleration vector | $\mathcal{S}^h, \mathcal{S}_i^h, \mathcal{S}^h$ | finite-dimensional collections of trial solutions |
| v | velocity vector | | |
| d | displacement vector | $\mathcal{O}^h, \mathcal{O}_i^h, \mathcal{O}^h$ | finite-dimensional collections of weighting functions |
| \mathbf{A} | finite element assembly operator | u, u_i, u | trial solutions |
| | $\left(\text{e.g., } K = \sum_{e=1}^{n_{el}} \mathbf{A} k^e \right)$ | w, w_i, w | weighting functions |
| | | u^h, u_i^h, u^h | finite-dimensional trial solutions |

Boundary-Value Problems

| | | |
|------------------------------------|---|--|
| n_{sd} | number of space dimensions | w^h, w_i^h, w^h |
| i, j, k, l | spatial indices $(1 \leq i, j, k, l \leq n_{sd})$ | w^h, w_i^h, w^h |
| $\mathbb{R}^{n_{sd}}$ | Euclidean n_{sd} -space | t, t_i, t |
| Ω | domain in $\mathbb{R}^{n_{sd}}$ | g, g_i, g |
| $\bar{\Omega}$ | closure of Ω | |
| x_i, x | point in $\bar{\Omega}$ | h, h_i, h |
| Γ | boundary of Ω | |
| n_i, n | unit outward normal vector to Γ | g^h, g_i^h, g^h |
| $\Gamma_\alpha, \Gamma_{\alpha_i}$ | part of the boundary where Dirichlet conditions are specified | e, e_i, e $L_2(\Omega)$ |
| | | errors |
| | | Hilbert space of square-integrable functions |

| | | | |
|-------------------------|---|--|---|
| $H^s(\Omega)$ | Sobolev space of functions with s square-integrable generalized derivatives | u_i $u_{(i,j)}$ | displacement vector symmetric part of the displacement gradients (equals ϵ_{ij}) |
| $\ \cdot\ $ | L_2 norm on Ω | f_i | prescribed body force vector |
| $\ \cdot\ _s$ | H^s norm on Ω | g_i | prescribed boundary displacement vector |
| $a(\cdot, \cdot)$ | symmetric bilinear form; strain-energy inner product | h_i | prescribed boundary traction vector |
| (\cdot, \cdot) | symmetric bilinear form; L_2 inner product on Ω | d_{ia}, d_{ia}^ϵ q_{ia}, q_{ia}^ϵ | nodal displacements prescribed nodal displacements |
| $(\cdot, \cdot)_\Gamma$ | symmetric bilinear form; L_2 inner product on Γ | B E ν | bulk modulus Young's modulus Poisson's ratio |

Data Processing Arrays

| | | | |
|------------------|----------------------|----------------|------------------------------|
| IEN (a, e) | element nodes matrix | G, μ | shear modulus |
| ID (i, A) | destination matrix | λ | Lamé modulus |
| LM (i, a, e) | location matrix | B, B_A, B_a | strain-displacement matrices |
| | | D, \tilde{D} | material moduli matrices |

Heat Conduction

| | | | |
|--------------|---------------------------------|---|--|
| q_i | heat flux vector | Isoparametric Elements | |
| K_{ij} | conductivities | Ω^e | element domain $\subset \Omega$ |
| u | temperature | Γ^e | boundary of Ω^e |
| $u_{,i}$ | temperature gradient | $x_1, x_2, x_3,$ x, y, z | Cartesian coordinates |
| f | prescribed heat supply | \square | parent domain of a quadrilateral or hexahedral element |
| q | prescribed boundary temperature | $x^e: \square \rightarrow \bar{\Omega}^e$ | isoparametric mapping |
| h | prescribed boundary heat flux | ξ, η | element natural coordinates for quadrilaterals |
| d_A, d_a^e | nodal temperature | ξ, η, ζ | element natural coordinates for hexahedra |
| q_A, q_a^e | prescribed nodal temperature | ξ | point in \square |

Elasticity

| | | | |
|-----------------|-----------------------------|-----|--|
| ϵ_{ij} | infinitesimal strain tensor | j | Jacobian determinant of the mapping $x^e(\xi)$ |
| σ_{ij} | Cauchy stress tensor | | |
| c_{ijkl} | elastic coefficients | | |

A Brief Glossary of Notations

| | | | |
|--|---|---|---|
| $C^0(\Omega)$ | the class of continuous functions | p | vector of nodal pressures |
| $C^k(\Omega)$ | the class of continuous functions possessing k continuous derivatives | B_a^{dev} | deviatoric strain-displacement matr |
| N_A, N_a^ϵ | shape functions | B_a^{dil} | dilatational strain-displacement matr |
| x_A, x_a^ϵ | nodal points | \bar{B}_a^{dil} | improved dilatatio: strain-displacemen matrix |
| r, s, t | triangular coordinates; area coordinates | \bar{B}_a | improved strain-displacement matr |
| r, s, t, u | tetrahedral coordinates; volume coordinates | | |
| $l_a(\xi)$ | Lagrange polynomial associated with the a th element node | Plates | tensor indices ($1 \leq \alpha, \beta, \gamma, \delta \leq$) |
| $\tilde{\xi}_l$ | location of the l th integration point | $e_{\alpha\beta}$ | alternator tensor |
| W_l | weight assigned to l th integration point | w | transverse displacement |
| $U1, U2, U3$ | uniform integration elements | θ_α | rotation vector |
| $S1, S2, S3$ | selective integration elements | $\kappa_{\alpha\beta} = \theta_{(\alpha, \beta)}$ | curvature tensor |
| Incompressible and Nearly-Incompressible Elasticity | | γ_α | shear strain vector |
| \mathcal{P} | pressure trial solution and weighting function space | $c_{\alpha\beta\gamma\delta}, c_{\alpha\beta}$ | elastic coefficients |
| \mathcal{D}^h | finite-dimensional pressure trial solution and weighting function space | $m_{\alpha\beta}$ | moment tensor |
| $p, q \in \mathcal{P}$ | trial pressure and weighting function | q_α | shear force vector |
| $p^h, q^h \in \mathcal{D}^h$ | finite-dimensional trial pressure and weighting function | W | prescribed boundary |
| $\mathbf{g}^\epsilon, \mathbf{G}$ | element and global gradient matrices | Θ_α | displacement |
| $(\mathbf{g}^\epsilon)^T, \mathbf{G}^T$ | element and global divergence matrices | \mathbf{F} | prescribed boundary rotations |
| | | C_α | total applied transverse force |
| | | M_α | total applied couple |
| | | Q | prescribed boundary moments |
| | | s | prescribed boundary shear force |
| | | $\hat{\theta}_\alpha = e_{\alpha\beta}\theta_\beta, \hat{\theta}_s, \hat{\theta}_n$ | arc-length paramete of plate boundary |
| | | $\mathbf{k}_b^\epsilon, \mathbf{k}_s^\epsilon$ | right-hand-rule rotations |
| | | | ϵ th element bending and shear stiffness |

| | | | |
|------------------------------|--|--|--|
| D^b, D^s | matrices of bending and shear moduli | $z_a(\zeta)$ | thickness function |
| B^b, B^s | bending and shear strain-displacement matrices | e_1^l, e_2^l, e_3^l | lamina basis vectors |
| | | e_1^f, e_2^f, e_3^f | fiber basis vectors |
| | | u | displacement vector |
| | | \bar{u}, \bar{u}_a | displacement of reference surface |
| Beams | | | |
| w_i | displacements | U, U_a, \hat{U}_a | fiber displacements |
| θ_i | rotations | $\tilde{\boldsymbol{\sigma}}^l = \tilde{\boldsymbol{D}}^l \tilde{\boldsymbol{\epsilon}}^l$ | reduced constitutive equation in the lamina basis |
| κ_α | curvatures | | moments |
| γ_α | shear strains | $m_{\alpha\beta}$ | membrane forces |
| ϵ | axial strain | n_α | transverse shear forces |
| ψ | twist | q_α | |
| m_i | moments | | |
| q_i | shear forces | | |
| W_i | prescribed end displacements | | |
| Θ_i | prescribed end rotations | t | time |
| F_i | total applied force per unit length | Δt | time step |
| C_i | total applied couple per unit length | n | time step number |
| M_i | prescribed end moments | a_n, v_n, d_n | approximations of $a(t_n), v(t_n)$, and $d(t_n)$, respectively |
| Q_i | prescribed end shears | $\tilde{a}_n, \tilde{v}_n, \tilde{d}_n$ | predictor values of acceleration, velocity, and displacement, respectively |
| A | cross-sectional area | | |
| A_α | shear areas | Δt_{crit} | critical time step |
| I_α | moments of inertia | A | amplification factor |
| J | polar moment of inertia | A | amplification matrix |
| $k_a^e, k_b^e, k_s^e, k_t^e$ | eth element axial, bending, shear, and torsional stiffness, respectively | $\rho(A)$ | spectral radius of amplification matrix |
| | | $\tau, \bar{\tau}$ | local truncation errors |
| | | $\lambda, \lambda^h, \bar{\lambda}^h$ | eigenvalues |
| | | ψ, ϕ | eigenvectors |
| | | $\omega, \omega^h, \bar{\omega}^h$ | frequencies |
| Shells | | | |
| x | position vector | ξ | algorithmic damping ratio |
| \bar{x}, \bar{x}_a | position vectors to references surface | $(\bar{T} - T)/T$ | relative period error |
| X, \hat{X}_a | fiber vectors | Ω_{crit} | critical sampling frequency |

A Brief Glossary of Notations

| | | | |
|----------------------|---|------------------------|--|
| M^I, C^I, K^I, F^I | implicit mesh partition of mass, damping, stiffness, and force | u_{0i}, \dot{u}_{0i} | initial displacement and velocities |
| M^E, C^E, K^E, F^E | explicit mesh partition of mass, damping, stiffness, and force | u_0 ρ c | initial temperature density capacity |

1

Fundamental Concepts; a Simple One-Dimensional Boundary-Value Problem

1.1 INTRODUCTORY REMARKS AND PRELIMINARIES

The main constituents of a finite element method for the solution of a boundary-value problem are

- i. The variational or weak statement of the problem; and
- ii. The approximate solution of the variational equations through the use of “finite element functions.”

To clarify concepts we shall begin with a simple example.

Suppose we want to solve the following differential equation for u :

$$u_{,xx} + f = 0 \quad (1.1.1)$$

where a comma stands for differentiation (i.e., $u_{,xx} = d^2u/dx^2$). We assume f is a given smooth, scalar-valued function defined on the unit interval. We write

$$f : [0, 1] \rightarrow \mathbb{R} \quad (1.1.2)$$

where $[0, 1]$ stands for the unit interval (i.e., the set of points x such that $0 \leq x \leq 1$) and \mathbb{R} stands for the real numbers. In words, (1.1.2) states that for a given x in $[0, 1]$, $f(x)$ is a real number. (Often we will use the notation \in to mean “in” or “a member of.” Thus for each $x \in [0, 1]$, $f(x) \in \mathbb{R}$.) Also, $[0, 1]$ is said to be the *domain* of f , and \mathbb{R} is its *range*.

We have described the given function f as being smooth. Intuitively, you probably know what this means. Roughly speaking, if we sketch the graph of the function f , we want it to be a smooth curve without discontinuities or kinks. We do this to avoid technical difficulties. Right now we do not wish to elaborate further as

this would divert us from the main theme. At some point prior to moving on to the next chapter, the reader may wish to consult Appendix 1.I, "An Elementary Discussion of Continuity, Differentiability and Smoothness," for further remarks on this important aspect of finite element work. The exercise in Sec. 1.16 already uses a little of the language described in Appendix 1.I. The terminology may be somewhat unfamiliar to engineering and physical science students, but it is now widely used in the finite element literature and therefore it is worthwhile to become accustomed to it.

Equation (1.1.1) is known to govern the transverse displacement of a string in tension and also the longitudinal displacement of an elastic rod. In these cases, physical parameters, such as the magnitude of tension in the string, or elastic modulus in the case of the rod, appear in (1.1.1). We have omitted these parameters to simplify subsequent developments.

Before going on, we introduce a few additional notations and terminologies. Let $]0, 1[$ denote the unit interval without end points (i.e., the set of points x such that $0 < x < 1$). $]0, 1[$ and $[0, 1]$ are referred to as *open and closed unit intervals*, respectively. To simplify subsequent writing and tie in with notation employed later on in multidimensional situations, we shall adopt the definitions

$$\Omega =]0, 1[\quad (\text{open}) \quad (1.1.3)$$

$$\bar{\Omega} = [0, 1] \quad (\text{closed}) \quad (1.1.4)$$

See Fig. 1.1.1.

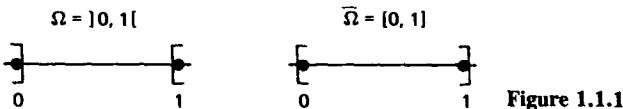


Figure 1.1.1

At this point, considerations such as these may seem pedantic. Our purpose, however, is to develop a language for the precise articulation of boundary-value problems, which is necessary for good finite element work.

1.2 STRONG, OR CLASSICAL, FORM OF THE PROBLEM

A boundary-value problem for (1.1.1) involves imposing *boundary conditions* on the function u . There are a variety of possibilities. We shall assume u is required to satisfy

$$u(1) = \varphi \quad (1.2.1)$$

$$-u_{,x}(0) = h \quad (1.2.2)$$

where φ and h are given constants. Equations (1.2.1) and (1.2.2) require that u take on the value φ at $x = 1$ and the derivative of u (i.e., slope) take on the value $-h$ at $x = 0$, respectively. This set of boundary conditions will later enable us to illustrate certain key features of variational formulations. For obvious reasons, boundary conditions of the type (1.2.1) and (1.2.2) lead to so-called *two-point boundary-value problems*.

Sec. 1.3 Weak, or Variational, Form of the Problem

The strong form of the boundary-value problem, (S) , is stated as follows:

$$(S) \left\{ \begin{array}{l} \text{Given } f : \bar{\Omega} \rightarrow \mathbb{R} \text{ and constants } g \text{ and } h, \text{ find } u : \bar{\Omega} \rightarrow \mathbb{R}, \text{ such that} \\ u_{xx} + f = 0 \quad \text{on } \Omega \\ u(1) = g \\ -u_{x}(0) = h \end{array} \right.$$

When we write $u_{xx} + f = 0$ on Ω we mean $u_{xx}(x) + f(x) = 0$ for all $x \in \Omega$. Of course, the exact solution of (S) is trivial to obtain, namely,

$$u(x) = g + (1 - x)h + \int_x^1 \left\{ \int_0^y f(z) dz \right\} dy \quad (1.2.3)$$

where y and z are used to denote dummy variables. However, this is not the main concern here. We are interested in developing schemes for obtaining approximate solutions to (S) that will be applicable to much more complex situations in which exact solutions are not possible.

Some methods of approximation begin directly with the strong statement of the problem. The most notable example is the finite difference method (e.g., see [1]). The finite element method requires a different formulation, which is treated in the next section.

1.3 WEAK, OR VARIATIONAL, FORM OF THE PROBLEM

To define the weak, or variational, counterpart of (S) , we need to characterize two classes of functions. The first is to be composed of candidate, or trial, solutions. From the outset, we shall require these possible solutions to satisfy the boundary condition $u(1) = g$. The other boundary condition will not be required in the definition. Furthermore, so that certain expressions to be employed make sense, we shall require that the derivatives of the trial solutions be square-integrable. That is, if u is a trial solution, then

$$\int_0^1 (u_x)^2 dx < \infty \quad (1.3.1)$$

Functions that satisfy (1.3.1) are called *H^1 -functions*; we write $u \in H^1$. Sometimes the domain is explicitly included, i.e., $u \in H^1([0, 1])$.

Thus the collection of *trial solutions*, denoted by \mathcal{S} , consists of all functions which have square-integrable derivatives and take on the value g at $x = 1$. This is written as follows:

$$\mathcal{S} = \{u \mid u \in H^1, u(1) = g\} \quad (\text{trial solutions}) \quad (1.3.2)$$

The fact that \mathcal{S} is a collection, or set, of objects is indicated by the curly brackets (called braces) in (1.3.2). The notation for the typical member of the set, in this case u , comes first inside the left-hand curly bracket. Following the vertical line ($|$) are the properties satisfied by members of the set.

The second collection of functions is called the *weighting functions*, or *variations*. This collection is very similar to the trial solutions except we require the homogeneous counterpart of the g -boundary condition. That is, we require weighting functions, w , to satisfy $w(1) = 0$. The collection is denoted by \mathcal{V} and defined by

$$\mathcal{V} = \{w \mid w \in H^1, w(1) = 0\} \quad (\text{weighting functions}) \quad (1.3.3)$$

It simplifies matters somewhat to continue to think of $f : \Omega \rightarrow \mathbb{R}$ as being smooth. (However, what follows holds for a considerably larger class of f 's.)

In terms of the preceding definitions, we may now state a suitable weak form, (W) , of the boundary-value problem.

$$(W) \left\{ \begin{array}{l} \text{Given } f, g, \text{ and } h, \text{ as before. Find } u \in \mathcal{S} \text{ such that for all } w \in \mathcal{V} \\ \boxed{\int_0^1 w_{,x} u_{,x} dx = \int_0^1 w f dx + w(0)h} \end{array} \right. \quad (1.3.4)$$

Formulations of this type are often called *virtual work*, or *virtual displacement, principles* in mechanics. The w 's are the *virtual displacements*.

Equation (1.3.4) is called the *variational equation*, or (especially in mechanics) the *equation of virtual work*.

The solution of (W) is called the *weak*, or *generalized, solution*. The definition given of a weak formulation is not the only one possible, but it is the most natural one for the problems we wish to consider.

1.4 EQUIVALENCE OF STRONG AND WEAK FORMS; NATURAL BOUNDARY CONDITIONS

Clearly, there must be some relationship between the strong and weak versions of the problem, or else there would be no point in introducing the weak form. It turns out that the weak and strong solutions are identical. We shall establish this assuming all functions are smooth. This will allow us to proceed expeditiously without invoking technical conditions with which the reader is assumed to be unfamiliar. "Proofs" of this kind are sometimes euphemistically referred to as "formal proofs." The intent is not to be completely rigorous but rather to make plausible the truth of the proposition. With this philosophy in mind, we shall "prove" the following.

Proposition

- a. Let u be a solution of (S) . Then u is also a solution of (W) .
- b. Let u be a solution of (W) . Then u is also a solution of (S) .

Another result, which we shall not bother to verify but is in fact easily established,

is that both (S) and (W) possess unique solutions. Thus, by (a) and (b), the strong and weak solutions are one and the same. Consequently, (W) is equivalent to (S) .

Formal Proof

- a. Since u is assumed to be a solution of (S) , we may write

$$0 = - \int_0^1 w(u_{,xx} + f) dx \quad (1.4.1)$$

for any $w \in \mathcal{V}$. Integrating (1.4.1) by parts results in

$$0 = \int_0^1 w_{,x} u_{,x} dx - \int_0^1 w f dx - w u_{,x} \Big|_0^1 \quad (1.4.2)$$

Rearranging and making use of the fact that $-u_{,x}(0) = h$ and $w(1) = 0$ results in

$$\int_0^1 w_{,x} u_{,x} dx = \int_0^1 w f dx + w(0)h \quad (1.4.3)$$

Furthermore, since u is a solution of (S) , it satisfies $u(1) = g$ and therefore is in \mathcal{S} . Finally, since u also satisfies (1.4.3) for all $w \in \mathcal{V}$, u satisfies the definition of a weak solution given by (W) .

b. Now u is assumed to be a weak solution. Thus $u \in \mathcal{S}$; consequently $u(1) = g$, and

$$\int_0^1 w_{,x} u_{,x} dx = \int_0^1 w f dx + w(0)h$$

for all $w \in \mathcal{V}$. Integrating by parts and making use of the fact $w(1) = 0$ results in

$$0 = \int_0^1 w(u_{,xx} + f) dx + w(0)[u_{,x}(0) + h] \quad (1.4.4)$$

To prove u is a solution of (S) it suffices to show that (1.4.4) implies¹

- i. $u_{,xx} + f = 0$ on Ω ; and
- ii. $u_{,x}(0) + h = 0$

First we shall prove (i). Define w in (1.4.4) by

$$w = \phi(u_{,xx} + f) \quad (1.4.5)$$

where ϕ is smooth; $\phi(x) > 0$ for all $x \in \Omega =]0, 1[$; and $\phi(0) = \phi(1) = 0$. For example, we can take $\phi(x) = x(1 - x)$, which satisfies all the stipulated requirements (see Figure 1.4.1). It follows that $w(1) = 0$ and thus $w \in \mathcal{V}$, so (1.4.5) defines a

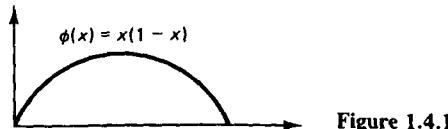


Figure 1.4.1

¹ These equations are sometimes called the *Euler-Lagrange equations* of the weak formulation.

legitimate member of \mathcal{V} . Substituting (1.4.5) into (1.4.4) results in

$$0 = \int_0^1 \phi \underbrace{(u_{,xx} + f)^2}_{\geq 0} dx + 0 \quad (1.4.6)$$

Since $\phi > 0$ on Ω , it follows from (1.4.6) that (i) must be satisfied.

Now that we have established (i), we may use it in (1.4.4) to prove (ii), namely,

$$0 = w(0)[u_{,x}(0) + h] \quad (1.4.7)$$

That $w \in \mathcal{V}$ puts no restriction whatsoever on its value at $x = 0$. Therefore, we may assume that the w in (1.4.7) is such that $w(0) \neq 0$. Thus (ii) is also shown to hold, which completes the proof of the proposition. ■

Remarks

1. The boundary condition $-u_{,x}(0) = h$ is not explicitly mentioned in the statement of (W). From the preceding proof, we see that this boundary condition is, however, implied by the satisfaction of the variational equation. Boundary conditions of this type are referred to as *natural boundary conditions*. On the other hand, trial solutions are explicitly required to satisfy the boundary condition $u(1) = \varrho$. Boundary conditions of this type are called *essential boundary conditions*. The fact that solutions of the variational equation satisfy natural boundary conditions is extremely important in more complicated situations which we will consider later on.

2. The method used to prove part (b) of the proposition goes under the name of the *fundamental lemma* in the literature of the calculus of variations. In essence, it is the methodology that enables us to deduce the differential equations and boundary conditions implied by the weak formulation. To develop correct weak forms for complex, multidimensional problems, it is essential to have a thorough understanding of these procedures.

Now we see that to obtain approximate solutions to the original boundary-value problem we have alternative starting points, i.e., the strong or weak statements of the problem. *Finite element methods are based upon the latter*. Roughly speaking, the basic idea is to approximate \mathcal{S} and \mathcal{V} by convenient, finite-dimensional collections of functions. (Clearly, \mathcal{S} and \mathcal{V} contain infinitely many functions.) The variational equations are then solved in this finite-dimensional context. An explicit example of how to go about this is the subject of the next section. However, we first introduce some additional notations to simplify subsequent writing.

Let

$$a(w, u) = \int_0^1 w_{,x} u_{,x} dx \quad (1.4.8)$$

$$(w, f) = \int_0^1 w f dx \quad (1.4.9)$$

In terms of (1.4.8) and (1.4.9), the variational equation takes the form

$$a(w, u) = (w, f) + w(0)/h \quad (1.4.10)$$

Here, $a(\cdot, \cdot)$ and (\cdot, \cdot) are examples of *symmetric, bilinear forms*. What this means is as follows: Let c_1 and c_2 be constants and let u , v , and w be functions. Then the *symmetry* property is

$$a(u, v) = a(v, u) \quad (1.4.11)$$

$$(u, v) = (v, u) \quad (1.4.12)$$

Bilinearity means linearity in each “slot”; for example,

$$a(c_1u + c_2v, w) = c_1a(u, w) + c_2a(v, w) \quad (1.4.13)$$

$$(c_1u + c_2v, w) = c_1(u, w) + c_2(v, w) \quad (1.4.14)$$

Exercise 1. Use the definitions of $a(\cdot, \cdot)$ and (\cdot, \cdot) to verify the properties of symmetry and bilinearity.

The above notations are very concise; at the same time they capture essential mathematical features and thus are conducive to a mathematical understanding of variational and finite element methods. Diverse classes of physical problems can be written in essentially similar fashion to (1.4.10). Thus ideas developed and results obtained are seen at once to have very broad applicability.

1.5 GALERKIN'S APPROXIMATION METHOD

We shall now describe a method of obtaining approximate solutions to boundary-value problems based upon weak formulations. Our introduction to this subject is somewhat of an abstract treatment. However, the meaning should be significantly reinforced by the remaining sections of the chapter. It may be worthwhile for the reader to consult this section again after completing the rest of the chapter to make sure a full comprehension of the material is attained.

The first step in developing the method is to construct finite-dimensional approximations of \mathcal{S} and \mathcal{V} . These collections of functions are denoted by \mathcal{S}^h and \mathcal{V}^h , respectively. The superscript refers to the association of \mathcal{S}^h and \mathcal{V}^h with a *mesh*, or *discretization*, of the domain Ω , which is parameterized by a characteristic length scale h . We wish to think of \mathcal{S}^h and \mathcal{V}^h as being subsets of \mathcal{S} and \mathcal{V} , respectively. This is written as

$$\mathcal{S}^h \subset \mathcal{S} \quad (\text{i.e., if } u^h \in \mathcal{S}^h, \text{ then } u^h \in \mathcal{S}) \quad (1.5.1)$$

$$\mathcal{V}^h \subset \mathcal{V} \quad (\text{i.e., if } w^h \in \mathcal{V}^h, \text{ then } w^h \in \mathcal{V}) \quad (1.5.2)$$

where the precise meaning is given in parentheses.² Consequences of (1.5.1) and (1.5.2) are (respectively) that if $u^h \in \mathcal{S}^h$ and $w^h \in \mathcal{V}^h$, then

$$u^h(1) = q \quad (1.5.3)$$

$$w^h(1) = 0 \quad (1.5.4)$$

The collections, \mathcal{S} , \mathcal{V} , \mathcal{S}^h , and \mathcal{V}^h , are often referred to as *function spaces*. The terminology *space* in mathematics usually connotes a linear structure. This has the following meaning: If c_1 and c_2 are constants and v and w are in \mathcal{V} , then $c_1v + c_2w$ is also in \mathcal{V} . Both \mathcal{V} and \mathcal{V}^h are thus seen to possess the property of a linear space. However, this property is clearly not shared by \mathcal{S} and \mathcal{S}^h due to the inhomogeneous boundary condition. For example, if u_1 and u_2 are members of \mathcal{S} , then $u_1 + u_2 \notin \mathcal{S}$, since $u_1(1) + u_2(1) = q + q = 2q$ in violation of the definition of \mathcal{S} . Nevertheless, the terminology function space is still (loosely) applied to \mathcal{S} and \mathcal{S}^h .

(Bubnov-) Galerkin Method

Assume the collection \mathcal{V}^h is given. Then, to each member $v^h \in \mathcal{V}^h$, we construct a function $u^h \in \mathcal{S}^h$ by

$$u^h = v^h + q^h \quad (1.5.5)$$

where q^h is a *given* function satisfying the essential boundary condition, i.e.,

$$q^h(1) = q \quad (1.5.6)$$

Note that (1.5.5) satisfies the requisite boundary condition also:

$$\begin{aligned} u^h(1) &= v^h(1) + q^h(1) \\ &= 0 + q \end{aligned} \quad (1.5.7)$$

Thus (1.5.5) constitutes a definition of \mathcal{S}^h ; that is, \mathcal{S}^h is all functions of the form (1.5.5). The key point to observe is that, up to the function q^h , \mathcal{S}^h and \mathcal{V}^h are composed of *identical* collections of functions. This property will be shown later on to have significant consequences for certain classes of problems.

We now write a variational equation, of the form of (1.4.10), in terms of $w^h \in \mathcal{V}^h$ and $u^h \in \mathcal{S}^h$:

$$a(w^h, u^h) = (w^h, f) + w^h(0)k \quad (1.5.8)$$

This equation is to be thought of as defining an approximate (weak) solution, u^h .

²This condition may be considered standard. However, it is often violated in practice. Strang [2] coined the terminology “variational crimes” to apply to this, and other, situations in which the classical rules of variational methods are violated. Many “variational crimes” have been given a rigorous mathematical basis (e.g., see [2]). We shall have more to say about this subject in subsequent chapters.

Substitution of (1.5.5) into (1.5.8), and the bilinearity of $a(\cdot, \cdot)$ enables us to write

$$a(w^h, v^h) = (w^h, f) + w^h(0)h - a(w^h, g^h) \quad (1.5.9)$$

The right-hand side consists of the totality of terms associated with given data (i.e., f , g , and h). Equation (1.5.9) is to be used to define v^h , the unknown part of u^h .

The (Bubnov-) Galerkin form of the problem, denoted by (G) , is stated as follows:

$$(G) \left\{ \begin{array}{l} \text{Given } f, g, \text{ and } h, \text{ as before, find } u^h = v^h + q^h, \text{ where } v^h \in \mathcal{V}^h, \\ \text{such that for all } w^h \in \mathcal{V}^h \\ a(w^h, v^h) = (w^h, f) + w^h(0)h - a(w^h, g^h) \end{array} \right.$$

Note that (G) is just a version of (W) posed in terms of a finite-dimensional collection of functions, namely, \mathcal{V}^h .

To make matters more specific, q^h and \mathcal{V}^h have to be explicitly defined. Before doing this, it is worthwhile to mention a larger class of approximation methods, called **Petrov-Galerkin methods**, in which v^h is contained in a collection of functions other than \mathcal{V}^h . Recent attention has been paid to methods of this type, especially in the context of fluid mechanics. For the time being, we will be exclusively concerned with the Bubnov-Galerkin method. The Bubnov-Galerkin method is commonly referred to as simply the Galerkin method, terminology we shall adopt henceforth. Equation (1.5.9) is sometimes referred to as the **Galerkin equation**.

Approximation methods of the type considered are examples of so-called **weighted residual methods**. The standard reference dealing with this subject is Finlayson [3]. For a more succinct presentation containing an interesting historical account, see Finlayson and Scriven [4].

1.6 MATRIX EQUATIONS; STIFFNESS MATRIX K

The Galerkin method leads to a coupled system of linear algebraic equations. To see this we need to give further structure to the definition of \mathcal{V}^h . Let \mathcal{V}^h consist of all linear combinations of given functions denoted by $N_A : \bar{\Omega} \rightarrow \mathbb{R}$, where $A = 1, 2, \dots, n$. By this we mean that if $w^h \in \mathcal{V}^h$, then there exist constants c_A , $A = 1, 2, \dots, n$, such that

$$\begin{aligned} w^h &= \sum_{A=1}^n c_A N_A \\ &= c_1 N_1 + c_2 N_2 + \dots + c_n N_n \end{aligned} \quad (1.6.1)$$

The N_A 's are referred to as **shape**, **basis**, or **interpolation** functions. We require that each N_A satisfies

$$N_A(1) = 0, \quad A = 1, 2, \dots, n \quad (1.6.2)$$

from which it follows by (1.6.1) that $w^h(1) = 0$, as is necessary. \mathcal{V}^h is said to have dimension n , for obvious reasons.

To define members of \mathcal{S}^h we need to specify φ^h . To this end, we introduce another shape function, $N_{n+1} : \bar{\Omega} \rightarrow \mathbb{R}$, which has the property

$$N_{n+1}(1) = 1 \quad (1.6.3)$$

(Note $N_{n+1} \notin \mathcal{V}^h$.) Then φ^h is given by

$$\varphi^h = \varphi N_{n+1} \quad (1.6.4)$$

and thus

$$\varphi^h(1) = \varphi \quad (1.6.5)$$

With these definitions, a typical $u^h \in \mathcal{S}^h$ may be written as

$$\begin{aligned} u^h &= v^h + \varphi^h \\ &= \sum_{A=1}^n d_A N_A + \varphi N_{n+1} \end{aligned} \quad (1.6.6)$$

where the d_A 's are constants and from which it is apparent that $u^h(1) = \varphi$.

Substitution of (1.6.1) and (1.6.6) into the Galerkin equation yields

$$\begin{aligned} a\left(\sum_{A=1}^n c_A N_A, \sum_{B=1}^n d_B N_B\right) &= \left(\sum_{A=1}^n c_A N_A, f\right) + \left[\sum_{A=1}^n c_A N_A(0)\right]_h \\ &\quad - a\left(\sum_{A=1}^n c_A N_A, \varphi N_{n+1}\right) \end{aligned} \quad (1.6.7)$$

By using the bilinearity of $a(\cdot, \cdot)$ and (\cdot, \cdot) , (1.6.7) becomes

$$0 = \sum_{A=1}^n c_A G_A \quad (1.6.8)$$

where

$$G_A = \sum_{B=1}^n a(N_A, N_B) d_B - (N_A, f) - N_A(0)_h + a(N_A, N_{n+1})\varphi \quad (1.6.9)$$

Now the Galerkin equation is to hold for all $w^h \in \mathcal{V}^h$. By (1.6.1), this means for all c_A 's, $A = 1, 2, \dots, n$. Since the c_A 's are arbitrary in (1.6.8), it necessarily follows that each G_A , $A = 1, 2, \dots, n$, must be identically zero, i.e., from (1.6.9)

$$\sum_{B=1}^n a(N_A, N_B) d_B = (N_A, f) + N_A(0)_h - a(N_A, N_{n+1})\varphi \quad (1.6.10)$$

Note that everything is known in (1.6.10) except the d_B 's. Thus (1.6.10) constitutes a system of n equations in n unknowns. This can be written in a more concise form as follows:

Let

$$K_{AB} = a(N_A, N_B) \quad (1.6.11)$$

$$F_A = (N_A, f) + N_A(0)t - a(N_A, N_{n+1})q. \quad (1.6.12)$$

Then (1.6.10) becomes

$$\sum_{B=1}^n K_{AB}d_B = F_A, \quad A = 1, 2, \dots, n \quad (1.6.13)$$

Further simplicity is gained by adopting a matrix notation. Let

$$K = [K_{AB}] = \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & & \vdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{bmatrix} \quad (1.6.14)$$

$$F = \{F_A\} = \begin{Bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{Bmatrix} \quad (1.6.15)$$

and

$$d = \{d_B\} = \begin{Bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{Bmatrix} \quad (1.6.16)$$

Now (1.6.13) may be written as

$$Kd = F \quad (1.6.17)$$

The following terminologies are frequently applied, especially when the problem under consideration pertains to a mechanical system:

K = stiffness matrix

F = force vector

d = displacement vector

A variety of physical interpretations are of course possible.

At this point, we may state the matrix equivalent, (M) , of the Galerkin problem.

$$(M) \left\{ \begin{array}{l} \text{Given the coefficient matrix } \mathbf{K} \text{ and vector } \mathbf{F}, \text{ find } \mathbf{d} \text{ such that} \\ \mathbf{Kd} = \mathbf{F} \end{array} \right.$$

The solution of (M) is, of course, just $\mathbf{d} = \mathbf{K}^{-1} \mathbf{F}$ (assuming the inverse of \mathbf{K} , \mathbf{K}^{-1} , exists). Once \mathbf{d} is known, the solution of (G) may be obtained at any point $x \in \bar{\Omega}$ by employing (1.6.6), viz.,

$$u^h(x) = \sum_{A=1}^n d_A N_A(x) + \varrho N_{n+1}(x) \quad (1.6.18)$$

Likewise, derivatives of u^h , if required, may be obtained by term-by-term differentiation. It should be emphasized, that the solution of (G) is an *approximate* solution of (W) . Consequently, the differential equation and natural boundary condition are only approximately satisfied. The quality of the approximation depends upon the specific choice of N_A 's and the number n .

Remarks

1. The matrix \mathbf{K} is symmetric. This follows from the symmetry of $a(\cdot, \cdot)$ and use of Galerkin's method (i.e., the same shape functions are used for the variations and trial solutions):

$$\begin{aligned} K_{AB} &= a(N_A, N_B) \\ &= a(N_B, N_A) \\ &= K_{BA} \end{aligned} \quad (1.6.19)$$

In matrix notation

$$\mathbf{K} = \mathbf{K}^T \quad (1.6.20)$$

where the superscript T denotes transpose. The symmetry of \mathbf{K} has important computational consequences.

2. Let us schematically retrace the steps leading to the matrix problem, as they are typical of the process one must go through in developing a finite element method for any given problem:

$$(S) \Leftrightarrow (W) \approx (G) \Leftrightarrow (M)$$

(1.6.21)

The only apparent approximation made thus far is in approximately solving (W) via (G) . In more complicated situations, encountered in practice, the number of approximations increases. For example, the data ℓ , ϱ , and h may be approximated, as well as the domain Ω , calculation of integrals, and so on. Convergence proofs and error analyses involve consideration of each approximation.

3. It is sometimes convenient to write

$$u^h(x) = \sum_{A=1}^{n+1} N_A(x) d_A \quad (1.6.22)$$

where $d_{n+1} = q$.

1.7 EXAMPLES: 1 AND 2 DEGREES OF FREEDOM

In this section we will carry out the detailed calculations involved in formulating and solving the Galerkin problem. The functions employed are extremely simple, thus expediting computations, but they are also primitive examples of typical finite element functions.

Example 1 (1 degree of freedom)

In this case $n = 1$. Thus $w^h = c_1 N_1$ and $u^h = v^h + q^h = d_1 N_1 + q N_2$. The only unknown is d_1 . The shape functions must satisfy $N_1(1) = 0$ and $N_2(1) = 1$ (see (1.6.2) and (1.6.3)). Let us take $N_1(x) = 1 - x$ and $N_2(x) = x$. These are illustrated in Fig. 1.7.1 and clearly satisfy the required conditions. Since we are dealing with only 1 degree of freedom, the matrix paraphernalia collapses as follows:

$$\mathbf{K} = [K_{11}] = K_{11} \quad (1.7.1)$$

$$\mathbf{F} = \{F_1\} = F_1 \quad (1.7.2)$$

$$\mathbf{d} = \{d_1\} = d_1 \quad (1.7.3)$$

$$K_{11} = a(N_1, N_1) = \int_0^1 \underbrace{N_{1,x}}_{-1} \underbrace{N_{1,x}}_{-1} dx = 1 \quad (1.7.4)$$

$$\begin{aligned} F_1 &= (N_1, f) + N_1(0)h - a(N_1, N_2)q \\ &= \int_0^1 (1-x)f(x) dx + h - \int_0^1 \underbrace{N_{1,x}}_{-1} \underbrace{N_{2,x}}_{+1} dx q \\ &= \int_0^1 (1-x)f(x) dx + h + q \end{aligned} \quad (1.7.5)$$

$$d_1 = K_{11}^{-1} F_1 = F_1 \quad (1.7.6)$$

Consequently

$$u^h(x) = \underbrace{\left[\int_0^1 (1-y)f(y) dy + h + q \right]}_{d_1} (1-x) + qx \quad (1.7.7)$$

In (1.7.7), y plays the role of a dummy variable. An illustration of (1.7.7) appears in Fig. 1.7.2. To get a feel for the nature of the approximation, let us compare (1.7.7) with the exact solution (see (1.2.3)). It is helpful to consider specific forms for f .

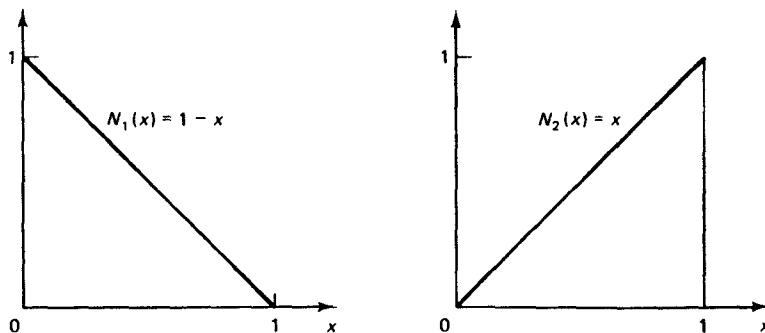


Figure 1.7.1 Functions for the 1 degree of freedom examples. (These functions are secretly the simplest finite element interpolation functions in a one-element context.)

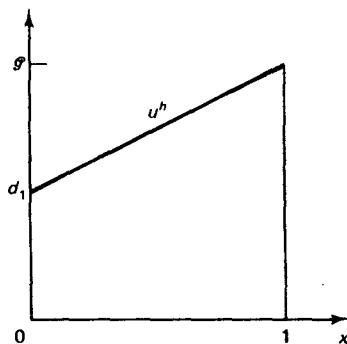


Figure 1.7.2 The Galerkin solution for the 1 degree of freedom example.

i. Let \$\ell = 0\$. Then

$$u^h(x) = u(x) = q + (1 - x)\ell \quad (1.7.8)$$

That is, the approximate solution is exact. In fact, it is clear by inspecting (1.7.7) and (1.2.3) that the homogeneous solution (i.e., the part of the solution corresponding to \$\ell = 0\$) is always exactly represented. The only approximation pertains to the particular solution (i.e., the part of the solution corresponding to \$\ell \neq 0\$).

ii. Now let us introduce a nonzero \$\ell\$. Assume \$\ell(x) = p\$, a constant. Then the particular solutions take the form

$$u_{\text{part}}(x) = \frac{p(1 - x^2)}{2} \quad (1.7.9)$$

and

$$u_{\text{part}}^h(x) = \frac{p(1 - x)}{2} \quad (1.7.10)$$

Equations (1.7.9) and (1.7.10) are compared in Fig. 1.7.3. Note that \$u_{\text{part}}^h\$ is exact at \$x = 0\$ and \$x = 1\$ and that \$u_{\text{part},x}^h\$ is exact at \$x = \frac{1}{2}\$ (It should be clear that it is impossible for \$u_{\text{part}}^h\$ to be exact at all \$x\$ in the present circumstances. The exact solution, (1.7.9), contains a quadratic term in \$x\$, whereas the approximate solution is restricted to linear variation in \$x\$ by the definitions of \$N_1\$ and \$N_2\$.)

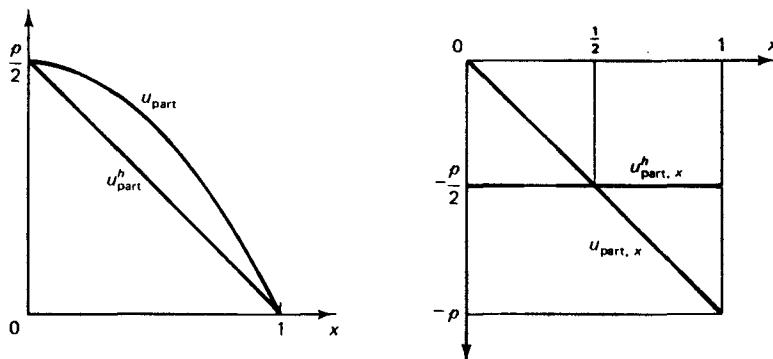


Figure 1.7.3 Comparison of exact and Galerkin particular solutions, Example 1, case (ii).

iii. This time let $f(x) = qx$, where q is a constant. This choice for f leads to

$$u_{\text{part}}(x) = \frac{q(1 - x^3)}{6} \quad (1.7.11)$$

and

$$u_{\text{part}}^h(x) = \frac{q(1 - x)}{6} \quad (1.7.12)$$

which are compared in Fig. 1.7.4. Again we note that the u_{part}^h is exact at $x = 0$ and $x = 1$. There is one point, $x = 1/\sqrt{3}$, at which $u_{\text{part},x}^h$ is exact.

Let us summarize what we have observed in this example:

- The homogeneous part of u^h is exact in all cases.
- In the presence of nonzero f , u^h is exact at $x = 0$ and $x = 1$.
- For each case, there is at least one point at which $u_{\text{part},x}^h$ is exact.

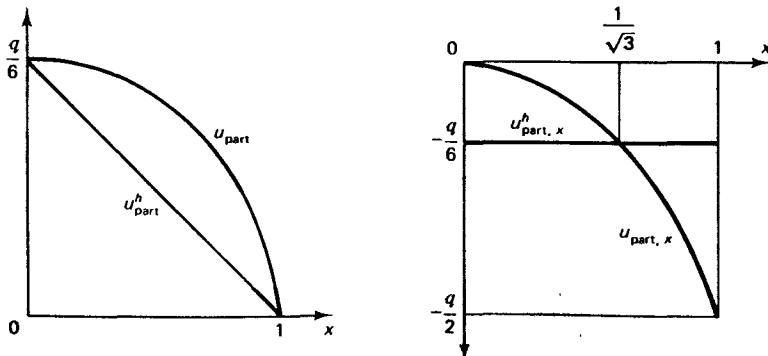


Figure 1.7.4 Comparison of exact and Galerkin particular solutions, Example 1, case (iii).

Example 2 (2 degrees of freedom)

In this case $n = 2$. Thus $w^h = c_1 N_1 + c_2 N_2$, where $N_1(1) = N_2(1) = 0$, and $u^h = d_1 N_1 + d_2 N_2 + \varrho N_3$, where $N_3(1) = 1$. Let us define the N_A 's as follows

$$N_1(x) = \begin{cases} 1 - 2x & 0 \leq x \leq \frac{1}{2} \\ 0 & \frac{1}{2} \leq x \leq 1 \end{cases} \quad (1.7.6)$$

$$N_2(x) = \begin{cases} 2x & 0 \leq x \leq \frac{1}{2} \\ 2(1-x) & \frac{1}{2} \leq x \leq 1 \end{cases} \quad (1.7.7)$$

$$N_3(x) = \begin{cases} 0 & 0 \leq x \leq \frac{1}{2} \\ 2x - 1 & \frac{1}{2} \leq x \leq 1 \end{cases} \quad (1.7.8)$$

The shape functions are illustrated in Fig. 1.7.5. Typical $w^h \in \mathcal{V}^h$ and $u^h \in \mathcal{S}^h$ and their derivatives are shown in Fig. 1.7.6. Since $n = 2$, the matrix paraphernalia takes the following form:

$$\mathbf{K} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \quad (1.7.9)$$

$$\mathbf{F} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix} \quad (1.7.10)$$

$$\mathbf{d} = \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} \quad (1.7.11)$$

$$K_{AB} = a(N_A, N_B) = \int_0^1 N_{A,x} N_{B,x} dx = \int_0^{1/2} N_{A,x} N_{B,x} dx + \int_{1/2}^1 N_{A,x} N_{B,x} dx \quad (1.7.12)$$

$$K_{11} = 2, \quad K_{12} = K_{21} = -2, \quad K_{22} = 4 \quad (1.7.13)$$

$$\mathbf{K} = 2 \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \quad (1.7.14)$$

$$\begin{aligned} F_A &= (N_A, f) + N_A(0)\kappa - a(N_A, N_3)\varrho \\ &= \int_0^1 N_A f dx + N_A(0)\kappa - \int_{1/2}^1 N_{A,x} N_{3,x} dx \varrho \end{aligned} \quad (1.7.15)$$

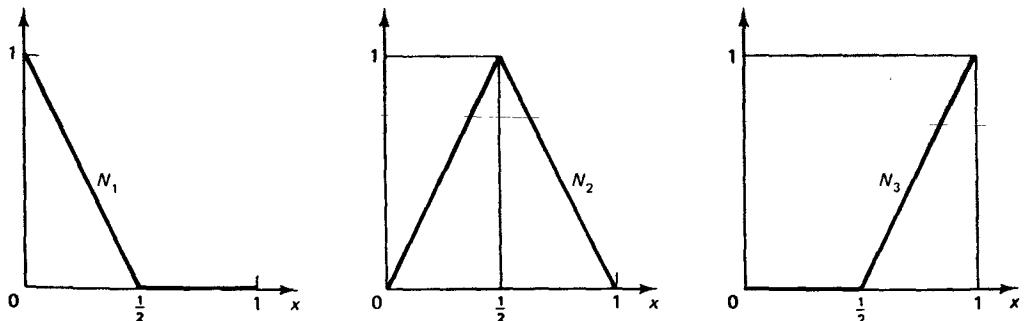


Figure 1.7.5 Functions for the 2 degree of freedom examples. (These functions are secretly the simplest finite element functions in a two-element context.)

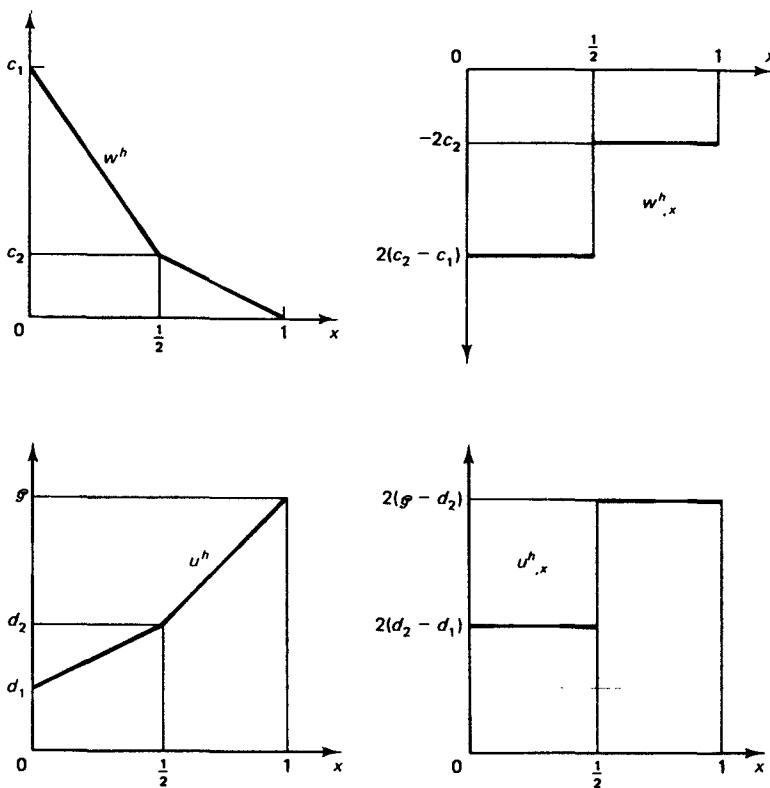


Figure 1.7.6 Typical weighting function and trial solution for the 2 degree of freedom example.

$$F_1 = \int_0^{1/2} (1 - 2x)\ell(x) dx + h \quad (1.7.16)$$

$$F_2 = 2 \int_0^{1/2} x\ell(x) dx + 2 \int_{1/2}^1 (1 - x)\ell(x) dx + 2g \quad (1.7.17)$$

Note that due to the shape functions' discontinuities in slope at $x = \frac{1}{2}$, it is convenient to express integrals over the subintervals $[0, \frac{1}{2}]$ and $[\frac{1}{2}, 1]$ (e.g., see (1.7.12) and (1.7.15)). We need not worry about the value of the derivative of N_A at $x = \frac{1}{2}$ (it suffers a discontinuity there and thus is not well-defined classically) since it has no effect on the integrals in (1.7.12). This amounts to employing the notion of a *generalized derivative*.

We shall again analyze the three cases considered in Example 1.
 i. $\ell = 0$.

$$\mathbf{F} = \begin{Bmatrix} h \\ 2g \end{Bmatrix} \quad (1.7.18)$$

$$\mathbf{d} = \mathbf{K}^{-1}\mathbf{F}$$

$$\begin{aligned}
 &= \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{Bmatrix} h \\ 2q \end{Bmatrix} \\
 &= \begin{Bmatrix} q + h \\ q + \frac{h}{2} \end{Bmatrix}
 \end{aligned} \tag{1.7.19}$$

This results in

$$\begin{aligned}
 u^h &= (q + h)N_1 + \left(q + \frac{h}{2}\right)N_2 + qN_3 \\
 &= q(N_1 + N_2 + N_3) + h\left(N_1 + \frac{N_2}{2}\right)
 \end{aligned} \tag{1.7.20}$$

$$u^h(x) = q + h(1 - x) \tag{1.7.21}$$

Again, the exact homogeneous solution is obtained. (The reason for this is that the exact solution is linear, and our trial solution is capable of exactly representing any linear function. Galerkin's method will give the exact answer whenever possible—that is, whenever the collection of trial solutions contains the exact solution among its members.)

ii. $f(x) = p = \text{constant}$.

$$F_1 = \frac{p}{4} + h \tag{1.7.22}$$

$$F_2 = \frac{p}{2} + 2q \tag{1.7.23}$$

$$d = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{Bmatrix} \frac{p}{4} + h \\ \frac{p}{2} + 2q \end{Bmatrix} = \begin{Bmatrix} \frac{p}{2} + q + h \\ \frac{3p}{8} + q + \frac{h}{2} \end{Bmatrix} \tag{1.7.24}$$

The solution takes the form

$$u^h(x) = q + h(1 - x) + u_{\text{part}}^h(x) \tag{1.7.25}$$

$$u_{\text{part}}^h = \frac{p}{2}N_1 + \frac{3p}{8}N_2 \tag{1.7.26}$$

The approximate particular solution is compared with the exact in Fig. 1.7.7, from which we see that agreement is achieved at $x = 0, \frac{1}{2}$ and 1, and derivatives coincide at $x = \frac{1}{4}$ and $\frac{3}{4}$.

iii. $f(x) = qx, q = \text{constant}$.

$$F_1 = \frac{q}{24} + h \tag{1.7.27}$$

$$F_2 = \frac{q}{4} + 2q \tag{1.7.28}$$

Sec. 1.7 Examples: 1 and 2 Degrees of Freedom

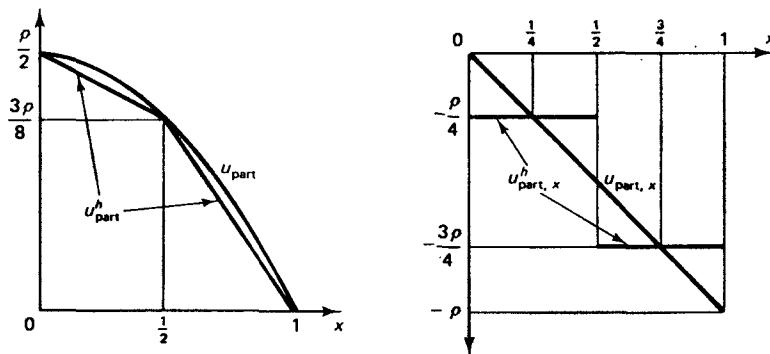


Figure 1.7.7 Comparison of exact and Galerkin particular solutions, Example 2, case (ii).

$$\mathbf{d} = \begin{Bmatrix} \frac{q}{6} + q + h \\ \frac{7q}{48} + q + \frac{h}{2} \end{Bmatrix}$$

Again u^h may be expressed in the form (1.7.25), where

$$u_{part}^h = \frac{q}{6} N_1 + \frac{7q}{48} N_2$$

A comparison is presented in Fig. 1.7.8. The Galerkin solution is seen to be exact again at $x = 0, \frac{1}{2}$, and 1, and the derivative is exact at two points.

Let us summarize the salient observations of Example 2:

- The homogeneous part of u^h is exact in all cases, as in Example 1. (A 1 for this is given after Equation (1.7.21).)

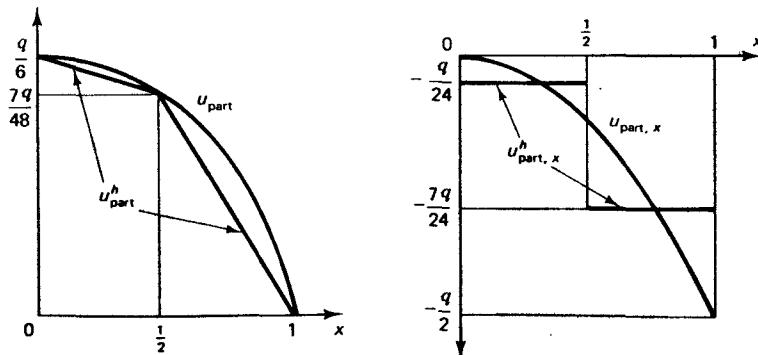


Figure 1.7.8 Comparison of exact and Galerkin particular solutions, Example 2, case (iii).

- b. The Galerkin solution is exact at the endpoints of each subinterval for all cases.
 c. In each case, there is at least one point in each subinterval at which u' is exact.

After generalizing to the case of n subintervals in the following section, we shall show in Sec. 1.10 that the above observations are not accidental.

Exercise 1. If the reader has not had experience with calculations of the type presented in this section, it would be worthwhile to reproduce all results, providing all omitted details.

1.8 PIECEWISE LINEAR FINITE ELEMENT SPACE

The examples of the preceding section employed definitions of \mathcal{V}^h and \mathcal{S}^h which were special cases of the so-called piecewise linear finite element space. To define the general case in which \mathcal{V}^h is n -dimensional, we partition the domain $[0, 1]$ into n nonoverlapping subintervals. The typical subinterval is denoted by $[x_A, x_{A+1}]$, where $x_A < x_{A+1}$ and $A = 1, 2, \dots, n$. We also require $x_1 = 0$ and $x_{n+1} = 1$. The x_A 's are called **nodal points**, or simply **nodes**. (The terminologies *joints* and *knots* are also used.) The subintervals are sometimes referred to as the **finite element domains**, or simply **elements**. Notice that the lengths of the elements, $h_A = x_{A+1} - x_A$, are *not* required to be equal. The mesh parameter, h , is generally taken to be the length of the maximum subinterval (i.e., $h = \max h_A, A = 1, 2, \dots, n$). The smaller h , the more "refined" is the partition, or mesh. If the subinterval lengths are equal, then $h = 1/n$.

The shape functions are defined as follows: Associated to a typical internal node (i.e., $2 \leq A \leq n$)

$$N_A(x) = \begin{cases} \frac{(x - x_{A-1})}{h_{A-1}}, & x_{A-1} \leq x \leq x_A \\ \frac{(x_{A+1} - x)}{h_A}, & x_A \leq x \leq x_{A+1} \\ 0, & \text{elsewhere} \end{cases} \quad (1.8.1)$$

whereas for the boundary nodes we have

$$N_1(x) = \frac{x_2 - x}{h_1}, \quad x_1 \leq x \leq x_2 \quad (1.8.2)$$

$$N_{n+1}(x) = \frac{x - x_n}{h_n}, \quad x_n \leq x \leq x_{n+1} \quad (1.8.3)$$

The shape functions are sketched in Fig. 1.8.1. For obvious reasons, they are referred to variously as "hat," "chapeau," and "roof" functions. Note that $N_A(x_B) = \delta_{AB}$, where

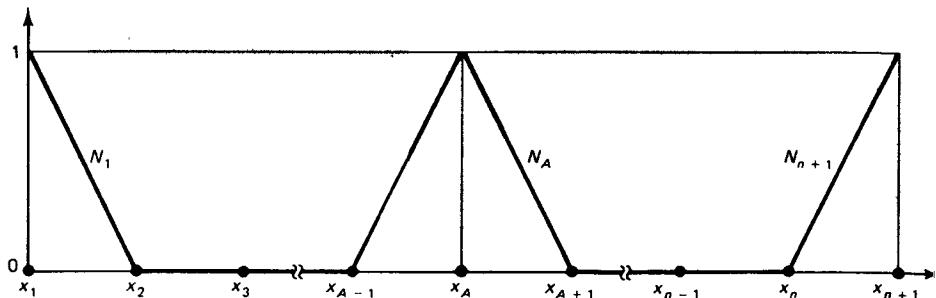
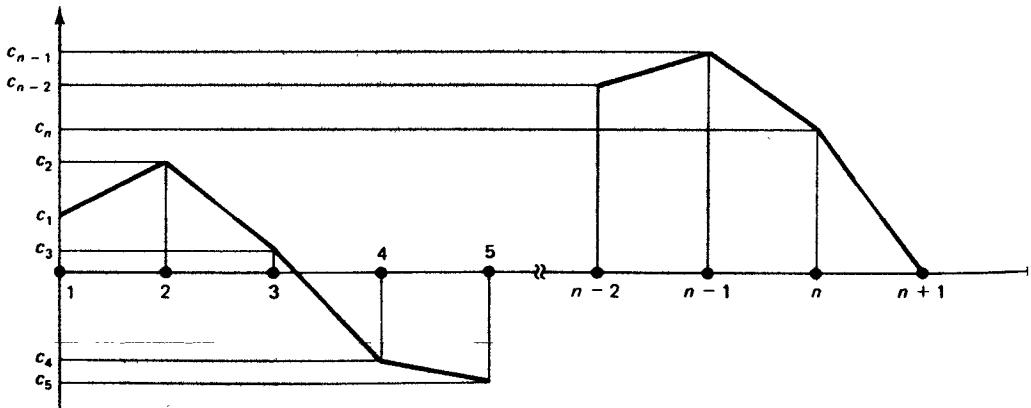


Figure 1.8.1 Basis functions for the piecewise linear finite element space.

δ_{AB} is the Kronecker delta (i.e., $\delta_{AB} = 1$ if $A = B$, whereas $\delta_{AB} = 0$ if $A \neq B$). In words, N_A takes on the value 1 at node A and is 0 at all other nodes. Furthermore, N_A is nonzero only in the subintervals that contain x_A .

A typical member $w^h \in \mathcal{V}^h$ has the form $\sum_{A=1}^n c_A N_A$ and appears as in Fig. 1.8.2. Note that w^h is continuous but has discontinuous slope across each element boundary. For this reason, w^h_x , the generalized derivative of w^h , will be piecewise constant, experiencing discontinuities across element boundaries. (Such a function is sometimes called a **generalized step function**.) Restricted to each element domain, w^h is a linear polynomial in x . In respect to the homogeneous essential boundary condition, $w^h(1) = 0$. Clearly, w^h is identically zero if and only if each $c_A = 0$, $A = 1, 2, \dots, n$.

Figure 1.8.2 A typical member $w^h \in \mathcal{V}^h$.

Typical members of \mathcal{S}^h are obtained by adding $q^h = q N_{n+1}$ to typical members of \mathcal{V}^h . This ensures that $u^h(1) = q$.

The piecewise linear finite element functions are the simplest and most widely used finite element functions for one-dimensional problems.

Exercise 1. Consider the weak formulation of the one-dimensional model problem:

$$\int_0^1 w_{,x} u_{,x} \, dx = \int_0^1 w f \, dx + w(0)h \quad (1.8.4)$$

where $w \in \mathcal{V}$ and $u \in \mathcal{S}$ are assumed to be smooth on element interiors (i.e., on $]x_A, x_{A+1}[$, $A = 1, 2, \dots, n$), but may suffer slope discontinuities across element boundaries. (Functions of this class contain the piecewise linear finite element space described earlier.) From (1.8.4) and the assumed continuity of the functions, show that:

$$0 = \sum_{A=1}^n \int_{x_A}^{x_{A+1}} w(u_{,xx} + f) \, dx + w(0)[u_{,x}(0^+) + h]$$

$$+ \sum_{A=2}^n w(x_A)[u_{,x}(x_A^+) - u_{,x}(x_A^-)] \quad (1.8.5)$$

Arguing as in Sec. 1.4, it may be concluded that the Euler-Lagrange conditions of (1.8.5) are

- i. $u_{,xx}(x) + f(x) = 0$, where $x \in]x_A, x_{A+1}[$ and $A = 1, 2, \dots, n$,
- ii. $-u_{,x}(0^+) = h$; and
- iii. $u_{,x}(x_A^-) = u_{,x}(x_A^+)$, where $A = 2, 3, \dots, n$.

Observe that (i) is the differential equation *restricted to element interiors*, and (iii) is a continuity condition across element boundaries. This may be contrasted with the case in which the solution is assumed smooth. In this case the continuity condition is identically satisfied and the summation of integrals over element interiors may be replaced by an integral over the entire domain (see Sec. 1.4).

In the Galerkin finite element formulation, an *approximate* solution of (i)–(iii) is obtained.

1.9 PROPERTIES OF \mathbf{K}

The shape functions N_A , $A = 1, 2, \dots, n + 1$, are zero outside a neighborhood of node A . As a result, many of the entries of \mathbf{K} are zero. This can be seen as follows. Let $B > A + 1$. Then (see Fig. 1.9.1)

$$K_{AB} = \int_0^1 \underbrace{N_{A,x} N_{B,x}}_0 \, dx = 0 \quad (1.9.1)$$

The symmetry of \mathbf{K} implies, in addition, that (1.9.1) holds for $A > B + 1$. One says that \mathbf{K} is **banded** (i.e., its nonzero entries are located in a band about the main diagonal). Figure 1.9.2 depicts this property. Banded matrices have significant advantages in that the zero elements outside the band neither have to be stored nor operated

Sec. 1.9 Properties of K

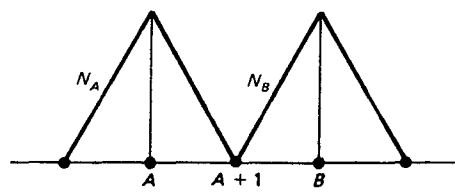


Figure 1.9.1 If $B > A + 1$, the non-zero portions of N_B and N_A do not overlap.

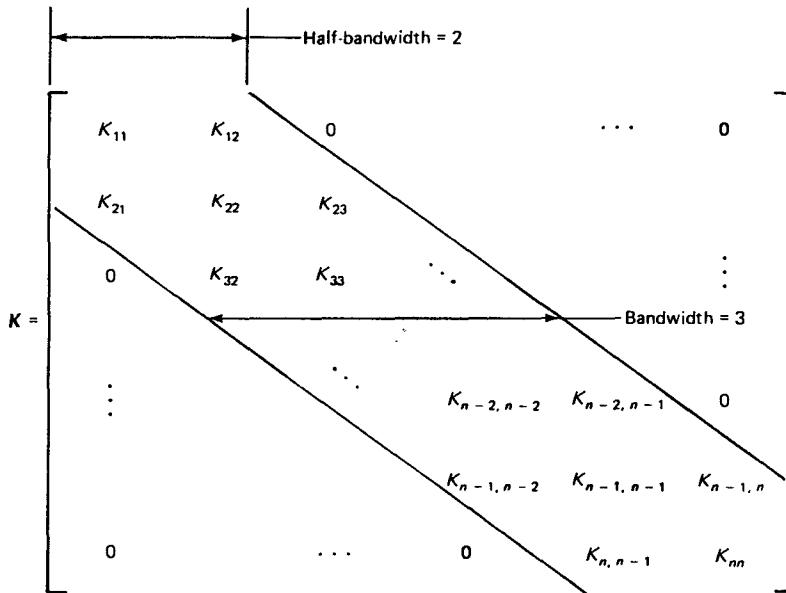


Figure 1.9.2 Band structure of K .

upon in the computer. The stiffness matrix arising in finite element analysis is, in general, narrowly banded, lending itself to economical formation and solution.

Definition. An $n \times n$ matrix A is said to be *positive definite* if

- $c^T A c \geq 0$ for all n -vectors c ; and
- $c^T A c = 0$ implies $c = 0$.

Remarks

1. A symmetric positive-definite matrix possesses a unique inverse.
2. The eigenvalues of a positive-definite matrix are real and positive.

Theorem. The $n \times n$ matrix K defined by (1.6.11) is positive definite.

Proof

- i. Let c_A , $A = 1, 2, \dots, n$, be the components of c (i.e., $c = \{c_A\}$), an arbitrary vector. Use these c_A 's to construct a member of \mathcal{V}^h , $w^h = \sum_{A=1}^n c_A N_A$, wh

the N_A 's are the basis functions for \mathcal{V}^h . Then

$$\begin{aligned}
 \mathbf{c}^T \mathbf{K} \mathbf{c} &= \sum_{A,B=1}^n c_A K_{AB} c_B \\
 &= \sum_{A,B=1}^n c_A a(N_A, N_B) c_B && (\text{definition of } K_{AB}) \\
 &= a\left(\sum_{A=1}^n c_A N_A, \sum_{B=1}^n c_B N_B\right) && (\text{bilinearity of } a(\cdot, \cdot)) \\
 &= a(w^h, w^h) && (\text{definition of } w^h) \\
 &= \int_0^1 \underbrace{(w^h_x)^2}_{\geq 0} dx && (\text{by (1.4.8)}) \\
 &\geq 0
 \end{aligned}$$

ii. Assume $\mathbf{c}^T \mathbf{K} \mathbf{c} = 0$. By the proof of part (i),

$$\int_0^1 (w^h_x)^2 dx = 0$$

and consequently w^h must be constant. Since $w^h \in \mathcal{V}^h$, $w^h(1) = 0$. Combining these facts, we conclude that $w^h(x) = 0$ for all $x \in [0, 1]$, which is possible only if each $c_A = 0$, $A = 1, 2, \dots, n$. Thus $\mathbf{c} = \mathbf{0}$. ■

Note that part (ii) depended upon the definition of \mathbf{K} and the zero essential boundary condition built into the definition of \mathcal{V}^h .

Summary. \mathbf{K} , defined by (1.6.11), is

- i. Symmetric
- ii. Banded
- iii. Positive-definite

The practical consequence of the above properties is that a very efficient computer solution of $\mathbf{Kd} = \mathbf{F}$ may be performed.

1.10 MATHEMATICAL ANALYSIS

In this section we will show that the observations made with reference to the example problems of Sec. 1.7 are, in fact, general results. To establish these facts rigorously requires only elementary mathematical techniques.

Our first objective is to establish that the Galerkin finite element solution u^h is exact at the nodes. To do this we must introduce the notion of a Green's function.

Let $\delta_y(x) = \delta(x - y)$ denote the *Dirac delta function*. The Dirac function is not a function in the classical sense but rather an operator defined by its action on

(continuous) functions. Let w be continuous on $[0, 1]$; then we write

$$\begin{aligned}(w, \delta_y) &= \int_0^1 w(x)\delta(x - y) dx \\ &= w(y)\end{aligned}\quad (1.10.1)$$

By (1.10.1), we see why attention is restricted to continuous functions— δ_y sifts off the value of w at y . If w were discontinuous at y , its value would be ambiguous. In mechanics, we think of δ_y visually as representing a concentrated force of unit amplitude located at point y .

The Green's function problem corresponding to (S) may be stated as follows. Find a function g (i.e., the *Green's function*) such that

$$g_{,xx} + \delta_y = 0 \quad \text{on } \Omega \quad (1.10.2)$$

$$g(1) = 0 \quad (1.10.3)$$

$$g_{,x}(0) = 0 \quad (1.10.4)$$

Note that (1.10.2)–(1.10.4) are simply (S) in which f is replaced by δ_y and g and φ are taken to be zero.

This problem may be solved by way of formal calculations with *distribution* or *generalized functions*, such as δ_y . (The theory of distributions is dealt with by Stakgold [5]. A good elementary account of formal calculations with distributions is presented in Popov [9]. This latter reference is recommended to readers having had no previous experience with this topic.) To this end we note that the (formal) integral of δ_y is the *Heaviside, or unit step, function*:

$$H_y(x) = H(x - y) = \begin{cases} 0, & x < y \\ 1, & x > y \end{cases} \quad (1.10.5)$$

The integral of H_y is the *Macaulay bracket*:

$$\langle x - y \rangle = \begin{cases} 0, & x \leq y \\ x - y, & x > y \end{cases} \quad (1.10.6)$$

The preceding functions are depicted in Fig. 1.10.1.

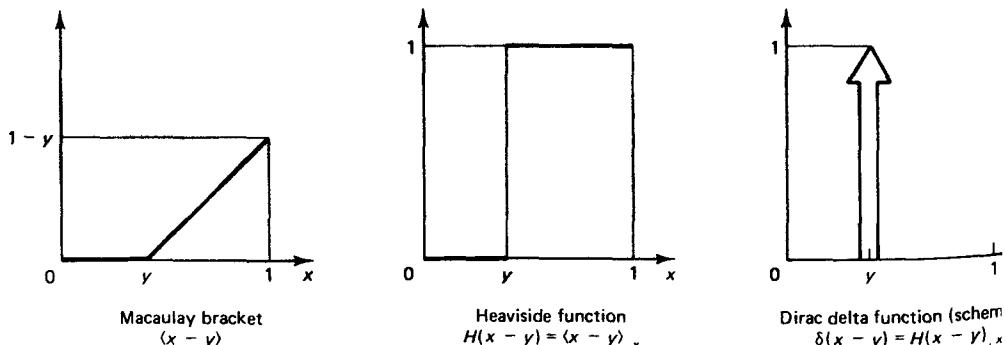


Figure 1.10.1 Elementary generalized functions (distributions).

To solve the Green's function problem, (1.10.2) is integrated, making use of (1.10.5), to obtain:

$$g_{,x} + H_y = c_1 \quad (1.10.7)$$

where c_1 is a constant of integration. A second integration and use of (1.10.6) yields

$$g(x) + \langle x - y \rangle = c_1 x + c_2 \quad (1.10.8)$$

where c_2 is another constant of integration. Evaluation of c_1 and c_2 is performed by requiring (1.10.7) and (1.10.8) to satisfy the boundary conditions. This results in (see Fig. 1.10.2)

$$g(x) = (1 - y) - \langle x - y \rangle \quad (1.10.9)$$

Observe that g is piecewise linear. Thus if $y = x_A$ (i.e., if y is a node), $g \in \mathcal{V}^h$.

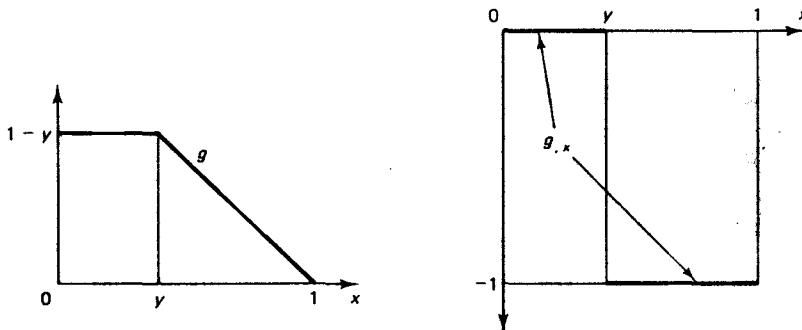


Figure 1.10.2 Green's function.

In the ensuing analysis we will need the variational equation corresponding to the Green's function problem. This can be deduced from (W) by replacing u by g , f by δ_y , and q and h by 0, viz.,

$$a(w, g) = (w, \delta_y) = w(y) \quad (1.10.10)$$

Equation (1.10.10) holds for all continuous $w \in \mathcal{V}$. The square-integrability of derivatives of functions $w \in \mathcal{V}$ actually implies the continuity of all $w \in \mathcal{V}$ by a well-known theorem in analysis due to Sobolev. (This result is true only in one dimension. The square-integrability of second derivatives is also required to ensure the continuity of functions defined on two- and three-dimensional domains.)

Theorem. $u^h(x_A) = u(x_A)$, $A = 1, 2, \dots, n + 1$ (i.e., u^h is exact at the nodes). To prove the theorem, we need to establish two preliminary results.

Lemma 1. $a(u - u^h, w^h) = 0$ for all $w^h \in \mathcal{V}^h$.

Proof. We have observed previously that $\mathcal{V}^h \subset \mathcal{V}$, so we may replace w by w^h in the variational equation:

$$a(w^h, u) = (w^h, f) + w^h(0)h \quad (1.10.11)$$

Equation (1.10.11) holds for all $w^h \in \mathcal{V}^h$. Recall that the Galerkin equation is identical to (1.10.11) except that u^h appears instead of u . Subtracting the Galerkin equation

from (1.10.11) and using the bilinearity and symmetry of $a(\cdot, \cdot)$ yields the required result.

Lemma 2. $u(y) - u^h(y) = a(u - u^h, g)$, where g is the Green's function

Proof

$$\begin{aligned} u(y) - u^h(y) &= (u - u^h, \delta_y) && \text{(definition of } \delta_y\text{)} \\ &= a(u - u^h, g) && \text{(by (1.10.10))} \end{aligned}$$

Note that line 2 is true since $u - u^h$ is in \mathcal{V} .

Proof of Theorem. As we have remarked previously, if $y = x_A$, a node $g \in \mathcal{V}^h$. Let us take this to be the case. Then

$$\begin{aligned} u(x_A) - u^h(x_A) &= a(u - u^h, g) && \text{(Lemma 2)} \\ &= 0 && \text{(Lemma 1)} \end{aligned}$$

The theorem is valid for $A = 1, 2, \dots, n + 1$. Strang and Fix [6] attribute this argument to Douglas and Dupont. Results of this kind, embodying exceptional accuracy characteristics, are often referred to as *superconvergence* phenomena. However, the reader should appreciate that, in more complicated situations, we will not be able in practice, to guarantee nodal exactness. Nevertheless, as we shall see later, weighted residual procedures provide a framework within which optimal accuracy properties of some sort may often be guaranteed.

Accuracy of the Derivatives

In considering the convergence properties of the derivatives, certain elementary notions of numerical analysis arise. The reader should make sure that he or she has complete understanding of these ideas as they subsequently arise in other contexts. We begin by introducing some preliminary mathematical results.

Taylor's Formula with Remainder

Let $f: [0, 1] \rightarrow \mathbb{R}$ possess k continuous derivatives and let y and z be two points in $[0, 1]$. Then there is a point c between y and z such that

$$\begin{aligned} f(z) &= f(y) + (z - y)f_{,x}(y) + \frac{1}{2}(z - y)^2 f_{,xx}(y) \\ &\quad + \frac{1}{3!}(z - y)^3 f_{,xxx}(y) + \cdots + \\ &\quad + \underbrace{\frac{1}{k!}(z - y)^k f_{,x \dots x}(c)}_{k \text{ times}} \end{aligned} \tag{1.10.12}$$

The proof of this formula may be found in [7]. Equation (1.10.12) is sometimes called a *finite Taylor expansion*.

Mean-Value Theorem

The mean-value theorem is a special case of (1.10.12) which is valid as long as $k \geq 1$ (i.e., f is continuously differentiable):

$$f(z) = f(y) + (z - y)f_{,x}(c) \quad (1.10.13)$$

Consider a typical subinterval $[x_A, x_{A+1}]$. We have already shown that u^h is exact at the endpoints (see Fig. 1.10.3). The derivative of u^h in $]x_A, x_{A+1}[$ is constant:

$$u_{,x}^h(x) = \frac{u^h(x_{A+1}) - u^h(x_A)}{h_A}, \quad x \in]x_A, x_{A+1}[\quad (1.10.14)$$

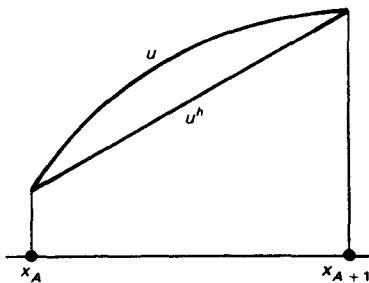


Figure 1.10.3

Theorem. Assume u is continuously differentiable. Then there exists at least one point in $]x_A, x_{A+1}[$ at which (1.10.14) is exact.

Proof. By the mean value theorem, there exists a point $c \in]x_A, x_{A+1}[$ such that

$$\frac{u(x_{A+1}) - u(x_A)}{h_A} = u_{,x}(c) \quad (1.10.15)$$

(We have used (1.10.13) with u , x_A , and x_{A+1} , in place of f , y , and z , respectively.) Since $u(x_A) = u^h(x_A)$ and $u(x_{A+1}) = u^h(x_{A+1})$, we may rewrite (1.10.15) as

$$\frac{u^h(x_{A+1}) - u^h(x_A)}{h_A} = u_{,x}(c) \quad (1.10.16)$$

Comparison of (1.10.16) with (1.10.14) yields the desired result. ■

Remarks

1. This result means that the constant value of $u_{,x}^h$ must coincide with $u_{,x}$ somewhere on $]x_A, x_{A+1}[$; see Fig. 1.10.4.
2. Without knowledge of u we have no way of determining the locations at which the derivatives are exact. The following results are more useful in that they tell us that the midpoints are, in a sense, optimally accurate, independent of u .

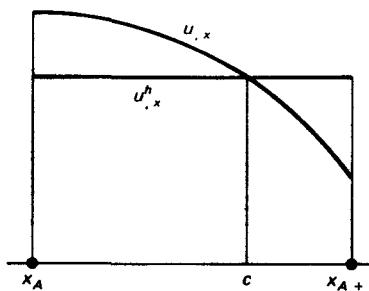


Figure 1.10.4

Let

$$e_{,x}(\alpha) \stackrel{\text{def.}}{=} u_{,x}^h(\alpha) - u_{,x}(\alpha) = \frac{u^h(x_{A+1}) - u^h(x_A)}{h_A} - u_{,x}(\alpha)$$

the *error in the derivative* at $\alpha \in [x_A, x_{A+1}]$. To establish the superiority of the midpoints in evaluating the derivatives, we need a preliminary result.

Lemma. Assume u is three times continuously differentiable. Then

$$\begin{aligned} e_{,x}(\alpha) &= \left(\frac{x_{A+1} + x_A}{2} - \alpha \right) u_{,xx}(\alpha) \\ &\quad + \frac{1}{3! h_A} [(x_{A+1} - \alpha)^3 u_{,xxx}(c_1) - (x_A - \alpha)^3 u_{,xxx}(c_2)] \end{aligned} \tag{1.10.17}$$

where c_1 and c_2 are in $[x_A, x_{A+1}]$.

Proof. Expand $u(x_{A+1})$ and $u(x_A)$ in finite Taylor expansions about $\alpha \in [x_A, x_{A+1}]$, viz.,

$$u(x_{A+1}) = u(\alpha) + (x_{A+1} - \alpha)u_{,x}(\alpha) + \frac{1}{2}(x_{A+1} - \alpha)^2 u_{,xx}(\alpha)$$

$$+ \frac{1}{3!}(x_{A+1} - \alpha)^3 u_{,xxx}(c_1), \quad c_1 \in [\alpha, x_{A+1}]$$

$$u(x_A) = u(\alpha) + (x_A - \alpha)u_{,x}(\alpha) + \frac{1}{2}(x_A - \alpha)^2 u_{,xx}(\alpha)$$

$$+ \frac{1}{3!}(x_A - \alpha)^3 u_{,xxx}(c_2), \quad c_2 \in [x_A, \alpha]$$

Subtracting and dividing through by h_A yields

$$\begin{aligned} \frac{u(x_{A+1}) - u(x_A)}{h_A} &= u_{,x}(\alpha) + \left(\frac{x_{A+1} + x_A}{2} - \alpha \right) u_{,xx}(\alpha) \\ &\quad + \frac{1}{3! h_A} [(x_{A+1} - \alpha)^3 u_{,xxx}(c_1) - (x_A - \alpha)^3 u_{,xxx}(c_2)] \end{aligned}$$

Replacing $u(x_{A+1})$ by $u^h(x_{A+1})$ and $u(x_A)$ by $u^h(x_A)$ in the left-hand side and rearranging terms completes the proof. ■

Discussion

To determine what (1.10.17) tells us about the accuracy of the derivatives, we wish to think of the situation in which the mesh is being systematically refined (i.e., we let h_A approach zero). In this case h_A^2 will be much smaller than h_A . Thus, for a given u , if the right-hand side of (1.10.17) is $O(h_A^2)$,³ the error in the derivatives will be much smaller than if the right-hand side is only $O(h_A)$. The exponent of h_A is called the *order of convergence* or *order of accuracy*. In the former case we would have second-order convergence of the derivative, whereas in the latter case we would have only first-order convergence.

As an example, assume $\alpha \rightarrow x_A$. Then

$$e_{,x}(x_A) = \frac{h_A}{2} u_{,xx}(x_A) + \frac{h_A^2}{3!} u_{,xxx}(c_1) = O(h_A)$$

As $h_A \rightarrow 0$, the first term dominates. (We have seen from the example calculations in Sec. 1.8 that the endpoints of the subintervals are not very accurate for the derivatives.)

Clearly any point $\alpha \in [x_A, x_{A+1}]$ achieves first-order accuracy. We are thus naturally led to asking the question, are there any values of α at which higher-order accuracy is achieved?

Corollary. Let $x_{A+1/2} \equiv (x_A + x_{A+1})/2$ (i.e., the midpoint). Then

$$\begin{aligned} e_{,x}(x_{A+1/2}) &= \frac{h_A^2}{24} u_{,xxx}(c), \quad c \in [x_A, x_{A+1}] \\ &= O(h_A^2) \end{aligned}$$

Proof. By (1.10.17)

$$e_{,x}(x_{A+1/2}) = \frac{h_A^2}{48} [u_{,xxx}(c_1) + u_{,xxx}(c_2)]$$

By the continuity of $u_{,xxx}$, there is at least one point c between c_1 and c_2 such that

$$u_{,xxx}(c) = \frac{1}{2} [u_{,xxx}(c_1) + u_{,xxx}(c_2)]$$

Combining these facts completes the proof. ■

Remarks

- From the corollary we see that the derivatives are second-order accurate at the midpoints.

³A function $f(x)$ is said to be $O(x^k)$ (i.e., order x^k) if $f(x)/x^k \rightarrow$ a constant as $x \rightarrow 0$. For example, $f(x) = x^k$ is $O(x^k)$, as is $f(x) = \sum_{j=k}^{k+l} x^j$, $l \geq 0$. But neither is $O(x^{k+1})$. (Verify.)

2. If the exact solution is quadratic (i.e., consists of a linear combination of the monomials $1, x, x^2$), then $u_{xx} = 0$ and—by (1.10.17)—the derivative is exact at the midpoints. This is the case when $f(x) = \rho = \text{constant}$.

3. In linear elastic rod theory, the derivatives are proportional to the stresses. The midpoints of linear “elements” are sometimes called the ***Barlow stress points***, after Barlow [8], who first noted that points of optimal accuracy existed within elements.

Exercise 1. Assume the mesh length is constant (i.e., $h_A = h, A = 1, 2, \dots, n$). Consider the standard finite difference “stencil” for $u_{xx} + f = 0$ at a typical internal node, namely,

$$\frac{u_{A+1} - 2u_A + u_{A-1}}{h^2} + f_A = 0 \quad (1.10.18)$$

Assuming f varies in piecewise linear fashion and so can be expanded as

$$f = \sum_{A=1}^{n+1} f_A N_A \quad (1.10.19)$$

where the f_A 's are the nodal values of f , set up the finite element equation associated with node A and contrast it with (1.10.18). Deduce when (1.10.18) will also be capable of exhibiting superconvergence phenomena. (That is, what is the restriction on f ? Set up the finite element equation associated with node 1, accounting for nonzero h . Discuss this equation from the point of view of finite differences. (For further comparisons along these lines, the interested reader is urged to consult [6], Chapter 1.)

Summary. The Galerkin finite element solution u^h , of the problem (S) , possesses the following properties:

- i. It is exact at the nodes.
- ii. There exists at least one point in each element at which the derivative is exact.
- iii. The derivative is second-order accurate at the midpoints of the elements.

1.11 INTERLUDE: GAUSS ELIMINATION; HAND-CALCULATION VERSION

It is important for anyone who wishes to do finite element analysis to become familiar with the efficient and sophisticated computer schemes that arise in the finite element method. It is felt that the best way to do this is to begin with the simplest scheme, perform some hand calculations, and gradually increase the sophistication as time goes on.

To do some of the problems we will need a fairly efficient method of solving matrix equations by hand. The following scheme is applicable to systems of equations

$Kd = F$ in which *no pivoting* (i.e., reordering) is necessary. For example, symmetric, positive-definite coefficient matrices never require pivoting. The procedure is as follows:

Gauss Elimination

- Solve the first equation for d_1 and eliminate d_1 from the remaining $n - 1$ equations.
- Solve the second equation for d_2 and eliminate d_2 from the remaining $n - 2$ equations.
- \vdots
- \vdots
- Solve the $n - 1$ st equation for d_{n-1} and eliminate d_{n-1} from the n th equation.
- Solve the n -th equation for d_n .

The preceding steps are called *forward reduction*. The original matrix is reduced to upper triangular form. For example, suppose we began with a system of four equations as follows:

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \\ K_{41} & K_{42} & K_{43} & K_{44} \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix}$$

The *augmented matrix* corresponding to this system is

$$\left[\begin{array}{cccc|c} K_{11} & K_{12} & K_{13} & K_{14} & F_1 \\ K_{21} & K_{22} & K_{23} & K_{24} & F_2 \\ K_{31} & K_{32} & K_{33} & K_{34} & F_3 \\ \underbrace{K_{41} & K_{42} & K_{43} & K_{44}}_K & & & & F_4 \end{array} \right]$$

After the forward reduction, the augmented matrix becomes

$$\left[\begin{array}{cccc|c} 1 & K'_{12} & K'_{13} & K'_{14} & F'_1 \\ 0 & 1 & K'_{23} & K'_{24} & F'_2 \\ 0 & 0 & 1 & K'_{34} & F'_3 \\ 0 & 0 & 0 & 1 & d_4 \end{array} \right] \quad (1.11.1)$$

corresponding to the upper triangular system $Ud = F'$.⁴ It is a simply verified fact that if K is banded, then U will be also.

Employing the reduced augmented matrix, proceed as follows:

- Eliminate d_n from equations $n - 1, n - 2, \dots, 1$.

⁴Primes will be used to denote intermediate quantities throughout this section.

- Eliminate d_{n-1} from equations $n = 2, n = 3, \dots, 1$.

.

- Eliminate d_2 from the first equation.

This procedure is called ***back substitution***. For example, in the example just given, after back substitution we obtain

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & d_1 \\ 0 & 1 & 0 & 0 & d_2 \\ 0 & 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 & d_4 \end{array} \right] \quad (1.11.2)$$

\underbrace{I}_{d}

corresponding to the identity $Id = d$. The solution winds up in the last column.

Hand-Calculation Algorithm

In a hand calculation, Gauss elimination can be performed on the augmented matrix as follows.

Forward reduction

- Divide row 1 by K_{11} .
 - Subtract $K_{21} \times$ row 1 from row 2.
 - Subtract $K_{31} \times$ row 1 from row 3.
- .
- .
- Subtract $K_{n1} \times$ row 1 from row n .

Consider the example of four equations. The preceding steps reduce the first column to the form

$$\left[\begin{array}{cccc|c} 1 & K'_{12} & K'_{13} & K'_{14} & F'_1 \\ 0 & K''_{22} & K''_{23} & K''_{24} & F''_2 \\ 0 & K''_{32} & K''_{33} & K''_{34} & F''_3 \\ 0 & K''_{42} & K''_{43} & K''_{44} & F''_4 \end{array} \right]$$

Note that if $K_{A1} = 0$, then the computation for the A th row can be ignored.

Now reduce the second column

- Divide row 2 by K''_{22} .
- Subtract $K''_{32} \times$ row 2 from row 3.
- Subtract $K''_{42} \times$ row 2 from row 4.

- Subtract $K''_{n2} \times$ row 2 from row n .

The result for the example will look like

$$\left[\begin{array}{cccc|c} 1 & K'_{12} & K'_{13} & K'_{14} & F'_1 \\ 0 & 1 & K'''_{23} & K'''_{24} & F'''_2 \\ 0 & 0 & K'''_{33} & K'''_{34} & F'''_3 \\ 0 & 0 & K'''_{43} & K'''_{44} & F'''_4 \end{array} \right]$$

Note that only the submatrix enclosed in dashed lines is affected in this procedure.

Repeat until columns 3 to n are reduced and the upper triangular form (1.11.1) is obtained.

Back substitution

- Subtract $K'_{n-1,n} \times$ row n from row $n - 1$.
- Subtract $K'_{n-2,n} \times$ row n from row $n - 2$.

- Subtract $K'_{1,n} \times$ row n from row 1.

After these steps the augmented matrix, for this example, will look like

$$\left[\begin{array}{ccc|c|c} 1 & K'_{12} & K'_{13} & 0 & F''''_1 \\ 0 & 1 & K'_{23} & 0 & F''''_2 \\ 0 & 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 & d_4 \end{array} \right]$$

Note that the submatrix enclosed in dashed lines is unaffected by these steps, and, aside from zeroing the appropriate elements of the last column of the coefficient matrix, only the vector F' is altered.

Now clear the second-to-last column in the coefficient matrix:

- Subtract $K'_{n-2,n-1} \times$ row $n - 1$ from row $n - 2$.
- Subtract $K'_{n-3,n-1} \times$ row $n - 1$ from row $n - 3$.

- Subtract $K'_{1,n-1} \times$ row $n - 1$ from row 1.

Again we mention that the only nontrivial calculations are being performed on the last column (i.e., on F).

Repeat as above until columns $n - 2, n - 3, \dots, 2$ are cleared. The result is (1.11.2).

Remarks

1. In passing we note that the above procedure is *not* the same as the way one would implement Gauss elimination on a computer, which we shall treat later. In a computer program for Gauss elimination of symmetric matrices we would want all intermediate results to retain symmetry and thus save storage. This can be done by a small change in the procedure. However, it is felt that the given scheme is the clearest for hand calculations.

2. The numerical example with which we close this section illustrates the preceding elimination scheme. Note that the band is maintained (i.e., the zeros in the upper right-hand corner of the coefficient matrix remain zero throughout the calculations). The reader is urged to perform the calculations.

Example of Gauss elimination

$$\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

Augmented matrix

$$\left[\begin{array}{cccc|c} 1 & -1 & 0 & 0 & 1 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & 0 \end{array} \right]$$

Forward reduction

$$\left[\begin{array}{cccc|c} 1 & -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & 0 \end{array} \right]$$

$$\left[\begin{array}{cccc|c} 1 & -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & -1 & 2 & 0 \end{array} \right]$$

$$\left[\begin{array}{cccc|c} 1 & -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{array} \right]$$

Back substitution

$$\left[\begin{array}{cccc|c} 1 & -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 1 \end{array} \right]$$

$$\left[\begin{array}{cccc|c} 1 & -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 1 \end{array} \right]$$

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 4 \\ 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 1 \end{array} \right]$$

$$\left\{ \begin{array}{l} d_1 \\ d_2 \\ d_3 \\ d_4 \end{array} \right\} = \left\{ \begin{array}{l} 4 \\ 3 \\ 2 \\ 1 \end{array} \right\}$$

Exercise 1. Consider the boundary-value problem discussed in the previous sections:

$$u_{xx}(x) + f(x) = 0 \quad x \in]0, 1[$$

$$u(1) = q$$

$$-u_x(0) = h$$

Assume $f = qx$, where q is constant, and $q = h = 0$.

- Employing the linear finite element space with equally spaced nodes, set up and solve the Galerkin finite element equations for $n = 4$ ($h = \text{mesh parameter} = \frac{1}{4}$). Recall that in Sec. 1.7 this was carried out for $n = 1$ and $n = 2$ ($h = 1$ and $h = \frac{1}{2}$, respectively). Do *not* invert the stiffness matrix K ; use Gauss elimination to solve $Kd = F$ or a more sophisticated direct factorization scheme if you know one. You can check your answers since they must be exact at the nodes.
- Let $re_x = |u_x^h - u_x|/(q/2)$, the **relative error** in u_x . Compute re_x at the midpoints of the four elements. They should all be equal. (This was also the case for $n = 2$.)
- Employing the data for $h = 1, \frac{1}{2}$, and $\frac{1}{4}$, plot $\ln re_x$ versus $\ln h$.
- Using the error analysis for re_x at the midpoints presented in Sec. 1.10, answer the following questions:
 - What is the significance of the slope of the graph in part (c)?
 - What is the significance of the y-intercept?

1.12 THE ELEMENT POINT OF VIEW

So far we have viewed the finite element method simply as a particular Galerkin approximation procedure applied to the weak statement of the problem in question. What makes what we have done a finite element procedure is the character of the selected basis functions; particularly their piecewise smoothness and “local support” (i.e., $N_A \equiv 0$ outside a neighborhood of node A). This is the mathematical point of view; it is a *global* point of view in that the basis functions are considered to be defined everywhere on the domain of the boundary-value problem. The global viewpoint is useful in establishing the mathematical properties of the finite element method. This can be seen in Sec. 1.10 and will be made more apparent later on.

Now we wish to discuss another point of view called the *local*, or *element*, point of view. This viewpoint is the traditional one in engineering and is useful in the computer implementation of the finite element method and in the development of finite elements.

We begin our treatment of the local point of view with a question: What is a finite element?

We shall attempt to give the answer in terms of the piecewise linear finite element space that we defined previously. An individual element consists of the following quantities.

Linear finite element (global description)

- (g 1) Domain: $[x_A, x_{A+1}]$
- (g 2) Nodes: $\{x_A, x_{A+1}\}$
- (g 3) Degrees of freedom: $\{d_A, d_{A+1}\}$
- (g 4) Shape functions:⁵ $\{N_A, N_{A+1}\}$
- (g 5) Interpolation function:

$$u^h(x) = N_A(x)d_A + N_{A+1}(x)d_{A+1}, \quad x \in [x_A, x_{A+1}]$$

(Recall $d_A = u^h(x_A)$.) In words, a linear finite element is just the totality of paraphernalia associated with the globally defined function u^h restricted to the *element* domain. The above quantities are in terms of *global* parameters—namely, the global coordinates, global shape functions, global node ordering, and so on. It is fruitful to introduce a *local* set of quantities, corresponding to the global ones, so that calculations for a typical element may be standardized. These are given as follows:

Linear finite element (local description)

- (l 1) Domain: $[\xi_1, \xi_2]$

⁵In weighted residual methods in which δ^h and \mathcal{V}^h are built up from different classes of functions (i.e., Petrov-Galerkin methods), we would also have to specify a set of *weighting functions*, say $\{\tilde{N}_A, \tilde{N}_{A+1}\}$; the entire set of \tilde{N}_A 's would then constitute a basis for \mathcal{V}^h . In Galerkin's method $\tilde{N}_A = N_A$.

- (I2) Nodes: $\{\xi_1, \xi_2\}$
 (I3) Degrees of freedom: $\{d_1, d_2\}$
 (I4) Shape functions: $\{N_1, N_2\}$
 (I5) Interpolation function:

$$u^h(\xi) = N_1(\xi)d_1 + N_2(\xi)d_2$$

Note that in the local description, the nodal numbering begins with 1.

We shall relate the domains of the global and local descriptions by an “affine” transformation $\xi : [x_A, x_{A+1}] \rightarrow [\xi_1, \xi_2]$, such that $\xi(x_A) = \xi_1$ and $\xi(x_{A+1}) = \xi_2$. It is standard practice to take $\xi_1 = -1$ and $\xi_2 = +1$. Thus ξ may be represented by the expression

$$\xi(x) = c_1 + c_2 x \quad (1.12.1)$$

where c_1 and c_2 are constants which are determined by

$$\begin{aligned} -1 &= c_1 + x_A c_2 \\ 1 &= c_1 + x_{A+1} c_2 \end{aligned} \quad \left. \right\} \quad (1.12.2)$$

Solving this system yields

$$\xi(x) = \frac{2x - x_A - x_{A+1}}{h_A} \quad (1.12.3)$$

(Recall $h_A = x_{A+1} - x_A$.) The inverse of ξ is obtained by solving for x :

$$x(\xi) = \frac{h_A \xi + x_A + x_{A+1}}{2} \quad (1.12.4)$$

In (1.12.1), ξ is a mapping and x is a point, whereas in (1.12.4), x is a mapping and ξ is a point.

In the sequel, we adopt the notational convention that subscripts a, b, c, \dots pertain to the local numbering system. The subscripts A, B, C, \dots will always pertain to the global numbering system. To control the proliferation of notations, we will frequently use the same notation for the local and global systems (e.g., d_a and d_A or N_a and N_A). This generally should not cause confusion as the context will make clear which point of view is being adopted. If there is danger of confusion, a superscript e will be introduced to denote a quantity in the local description associated with element number e (e.g., $d_a^e = d_A$, $N_a^e(\xi) = N_A(x^e(\xi))$, where $x^e : [\xi_1, \xi_2] \rightarrow [x_1^e, x_2^e] = [x_A, x_{A+1}]$, etc.).

In terms of ξ , the shape functions in the local description take on a standard form

$$N_a(\xi) = \frac{1}{2}(1 + \xi_a \xi), \quad a = 1, 2 \quad (1.12.5)$$

Note also that (1.12.4) may be written in terms of (1.12.5):

$$x^\epsilon(\xi) = \sum_{a=1}^2 N_a(\xi) x_a^\epsilon. \quad (1.12.6)$$

This has the same form as the interpolation function (cf. 15).

For future reference, we note the following results:

$$N_{a,\xi} = \frac{\xi_a}{2} = \frac{(-1)^a}{2} \quad (1.12.7)$$

$$x_{,\xi}^\epsilon = \frac{h^\epsilon}{2} \quad (1.12.8)$$

where $h^\epsilon = x_2^\epsilon - x_1^\epsilon$ and

$$\xi_{,x}^\epsilon = (x_{,\xi}^\epsilon)^{-1} = \frac{2}{h^\epsilon} \quad (1.12.9)$$

The local and global descriptions of the ϵ th element are depicted in Fig. 1.12.1.

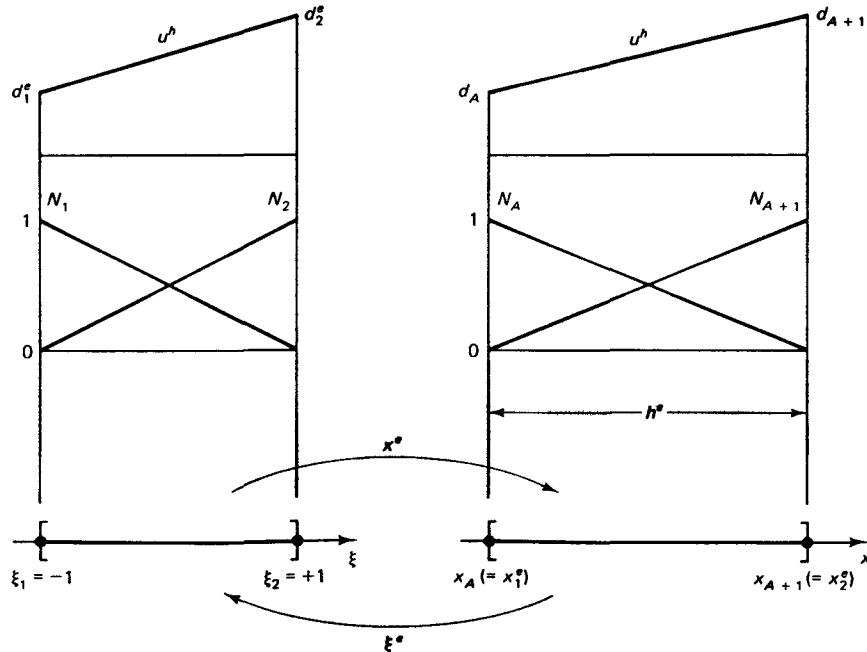


Figure 1.12.1 Local and global descriptions of the ϵ th element.

1.13 ELEMENT STIFFNESS MATRIX AND FORCE VECTOR

To develop the element point of view further, let us assume that our model consists of n_{el} elements, numbered as shown in Figure 1.13.1. Clearly $n_{el} = n$ for this case. Let us take e to be the variable index for the elements; thus $1 \leq e \leq n_{el}$.

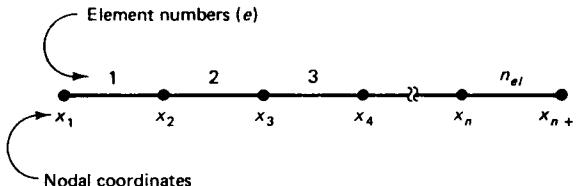


Figure 1.13.1

Now recall the definitions of the (global) stiffness matrix and force vector

$$\mathbf{K} = \underbrace{[K_{AB}]}_{n \times n}, \quad \mathbf{F} = \underbrace{\{F_A\}}_{n \times 1} \quad (1.13.1)$$

where

$$K_{AB} = a(N_A, N_B) = \int_0^1 N_{A,x} N_{B,x} dx \quad (1.13.2)$$

$$\begin{aligned} F_A &= (N_A, f) + \delta_{A1} h - a(N_A, N_{n+1}) q \\ &= \int_0^1 N_A f dx + \delta_{A1} h - \int_0^1 N_{A,x} N_{n+1,x} dx q \end{aligned} \quad (1.13.3)$$

(In (1.13.3) we have assumed $N_A(x_1) = \delta_{A1}$, as for the piecewise linear finite element space.) The integrals over $[0, 1]$ may be written as sums of integrals over the element domains. Thus

$$\mathbf{K} = \sum_{e=1}^{n_{el}} \mathbf{K}^e, \quad \mathbf{K}^e = [K_{AB}^e] \quad (1.13.4)$$

$$\mathbf{F} = \sum_{e=1}^{n_{el}} \mathbf{F}^e, \quad \mathbf{F}^e = \{F_A^e\} \quad (1.13.5)$$

where

$$K_{AB}^e = a(N_A, N_B)^e = \int_{\Omega^e} N_{A,x} N_{B,x} dx \quad (1.13.6)$$

$$\begin{aligned} F_A^e &= (N_A, f)^e + \delta_{e1} \delta_{A1} h - a(N_A, N_{n+1})^e q \\ &= \int_{\Omega^e} N_A f dx + \delta_{e1} \delta_{A1} h - \int_{\Omega^e} N_{A,x} N_{n+1,x} dx q \end{aligned} \quad (1.13.7)$$

and $\Omega^e = [x_1^e, x_2^e]$, the domain of the e th element.

The important observation to make is that \mathbf{K} and \mathbf{F} can be constructed by

summing the contributions of elemental matrices and vectors, respectively. In the literature, this procedure is sometimes called the *direct stiffness method* [10].

By the definitions of the N_A 's, we have that

$$K_{AB}^e = 0, \quad \text{if } A \neq e \text{ or } e + 1 \text{ or } B \neq e \text{ or } e + 1 \quad (1.13.8)$$

and

$$F_A^e = 0, \quad \text{if } A \neq e \text{ or } e + 1 \quad (1.13.9)$$

The situation for a typical element, e , is shown in Fig. 1.13.2. In practice we would not, of course, add in the zeros but merely add in the nonzero terms to the appropriate locations. For this purpose it is useful to define the *eth element stiffness matrix* k^e and *element force vector* f^e as follows:

$$\boxed{\begin{aligned} k^e &= \underbrace{[k_{ab}^e]}_{2 \times 2}, & f^e &= \underbrace{\{f_a^e\}}_{2 \times 1} \\ &&& \end{aligned}} \quad (1.13.10)$$

$$k_{ab}^e = a(N_a, N_b)^e = \int_{\Omega^e} N_{a,x} N_{b,x} dx \quad (1.13.11)$$

$$f_a^e = \int_{\Omega^e} N_a f dx + \begin{cases} \delta_{a1} h & e = 1 \\ 0 & e = 2, 3, \dots, n_{el} - 1 \\ -k_{a2}^e g & e = n_{el} \end{cases} \quad (1.13.12)$$

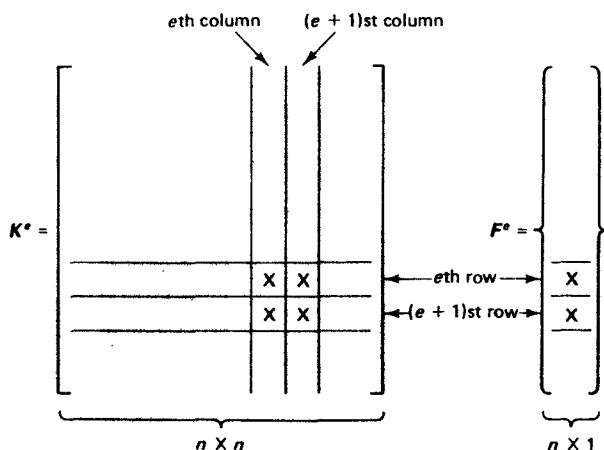


Figure 1.13.2 X 's indicate nonzero terms; all other terms are zero.

Here k^e and f^e are defined with respect to the *local* ordering, whereas K^e and F^e are defined with respect to the *global* ordering. To determine where the components of k^e and f^e "go" in K and F , respectively, requires keeping additional information. This is discussed in the following section.

1.14 ASSEMBLY OF GLOBAL STIFFNESS MATRIX AND FORCE VECTOR; LM ARRAY

In a finite element computer program, it is the task of a “finite element subroutine” to produce k^e and f^e , $e = 1, 2, \dots, n_{el}$, from given data and to provide an “assembly subroutine” enough information so that the terms in k^e and f^e can be added to the appropriate locations in K and F , respectively. This assembly information is stored in an array named LM, the *location matrix*.

Let us construct the LM array for the problem under consideration. The dimensions of LM are n_{en} , *the number of element nodes*, by the number of elements; in the present case, the numbers are 2 and n_{el} , respectively. Given a particular degree of freedom number and an element number (say a and e , respectively), the value returned by the LM array is the corresponding global equation number, A , viz.,

$$A = LM(a, e) = \begin{cases} e & \text{if } a = 1 \\ e + 1 & \text{if } a = 2 \end{cases} \quad (1.14.1)$$

The complete LM array is depicted in Fig. 1.14.1. This is the way we envision it stored in the computer. Note that $LM(2, n_{el}) = 0$. This indicates that degree of freedom 2 of element number n_{el} is prescribed and is not an unknown in the global matrix equation. Hence the terms $k_{12}^{n_{el}}$, $k_{21}^{n_{el}}$, $k_{22}^{n_{el}}$, and $f_2^{n_{el}}$ are *not* assembled into K and F , respectively. (There are no places for them to go!)

| Element numbers $1 \leq e \leq n_{el}$ | | | | | | | | | |
|--|---|---|---|---------|---------|---------|------------|----------|-----|
| Local node number | 1 | 2 | 3 | \dots | e | \dots | n_{el-1} | n_{el} | |
| (Local node number 1) | 1 | 1 | 2 | 3 | \dots | e | \dots | $n - 1$ | n |
| | 2 | 2 | 3 | 4 | \dots | $e + 1$ | \dots | n | 0 |

$(n_{en} = 2)$ $n_{en} \times n_{el}$

Figure 1.14.1 LM array for example problem.

As an example, assume we want to add the e th elemental contributions, where $1 \leq e \leq n_{el-1}$, to the partially assembled K and F . From the LM array, we deduce the following assembly procedure:

$$K_{ee} \leftarrow K_{ee} + k_{11}^e \quad (1.14.2)$$

$$K_{e,e+1} \leftarrow K_{e,e+1} + k_{12}^e \quad (1.14.3)$$

$$K_{e+1,e} \leftarrow K_{e+1,e} + k_{21}^e \quad (1.14.4)$$

$$K_{e+1,e+1} \leftarrow K_{e+1,e+1} + k_{22}^e \quad (1.14.5)$$

⁶ Due to symmetry k_{11}^e would not actually be assembled in practice.

$$F_e \leftarrow F_e + f_1^e \quad (1.14.6)$$

$$F_{e+1} \leftarrow F_{e+1} + f_2^e \quad (1.14.7)$$

where the arrow (\leftarrow) is read “is replaced by.”

For element n_{el} we have only that

$$K_{nn} \leftarrow K_{nn} + k_{11}^{n_{el}} \quad (1.14.8)$$

$$F_n \leftarrow F_n + f_1^{n_{el}} \quad (1.14.9)$$

With these ideas, we may construct, in sketchy fashion, an algorithm for the assembly of K and F ; see Fig. 1.14.2.

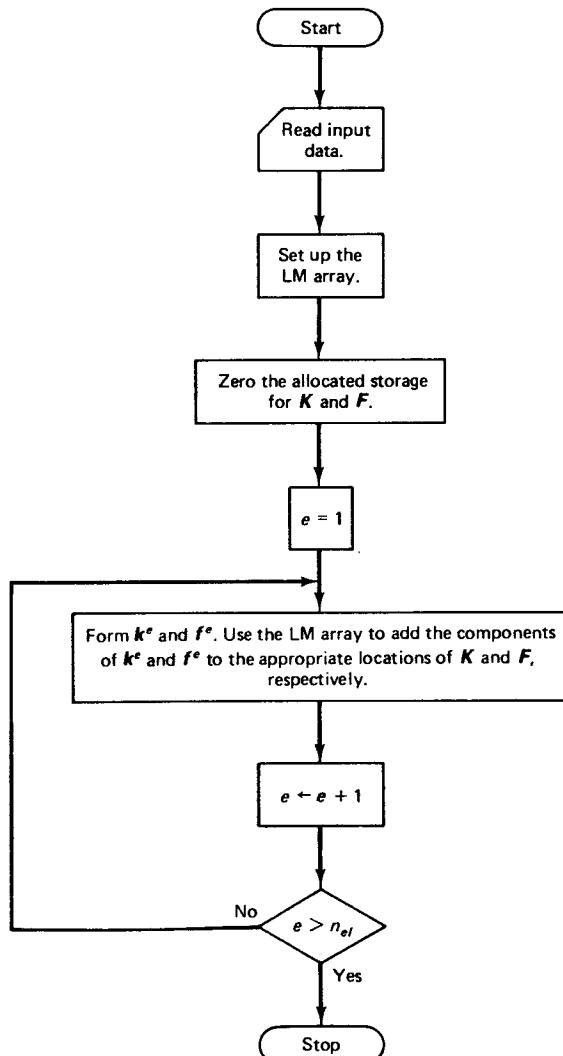


Figure 1.14.2 Flowchart of a finite element assembly algorithm.

The action of the assembly algorithm is denoted throughout by \mathbf{A} , the *assembly operator*, vis.,

$$\mathbf{K} = \sum_{e=1}^{n_{el}} (\mathbf{k}^e), \quad \mathbf{F} = \sum_{e=1}^{n_{el}} (\mathbf{f}^e) \quad (1.14.10)$$

1.15 EXPLICIT COMPUTATION OF ELEMENT STIFFNESS MATRIX AND FORCE VECTOR

The explicit computation of \mathbf{k}^e and \mathbf{f}^e , for the problem under consideration, provides some preliminary insight into the type of calculations that must be performed in a finite element subroutine. Some preliminary results are required.

Change of Variables Formula (One-Dimensional Version)

Let $f: [x_1, x_2] \rightarrow \mathbb{R}$ be an integrable function and let $x: [\xi_1, \xi_2] \rightarrow [x_1, x_2]$ be continuously differentiable, with $x(\xi_1) = x_1$ and $x(\xi_2) = x_2$. Then

$$\int_{x_1}^{x_2} f(x) dx = \int_{\xi_1}^{\xi_2} f(x(\xi))x_{,\xi}(\xi) d\xi \quad (1.15.1)$$

Chain Rule

Let f and x be as above, and, in addition, assume f is differentiable. Then

$$\frac{\partial}{\partial \xi} f(x(\xi)) = f_{,x}(x(\xi))x_{,\xi}(\xi) \quad (1.15.2)$$

Proofs of these results may be found in [11].

The computation of \mathbf{k}^e proceeds as follows:

$$\begin{aligned} k_{ab}^e &= \int_{\Omega^e} N_{a,x}(x)N_{b,x}(x) dx \quad (\text{by definition}) \\ &= \int_{-1}^{+1} N_{a,x}(x(\xi))N_{b,x}(x(\xi))x_{,\xi}(\xi) d\xi \\ &\quad (\text{Change of variables, where } x(\xi) \text{ is defined by (1.12.6)}) \\ &= \int_{-1}^{+1} N_{a,\xi}(\xi)N_{b,\xi}(\xi)(x_{,\xi}(\xi))^{-1} d\xi \end{aligned}$$

$$\begin{aligned} & \text{(Chain rule; } N_{a,\xi}(\xi) = (\partial/\partial\xi)N_a(x(\xi)) = N_{a,x}(x(\xi))x_{,\xi}(\xi)) \\ & = (-1)^{a+b}/h^e \quad \text{(by (1.12.7)–(1.12.9))} \end{aligned}$$

Thus

$$\boxed{k^e = \frac{1}{h^e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}} \quad (1.15.3)$$

Observe that $N_{a,\xi}$ (see (1.12.7)) does not depend upon the particular element data, as $N_a = N_a(\xi)$. We shall see that this is generally true, and hence these computations may be done once and for all.

The derivatives $x_{,\xi}$ and $\xi_{,x}$ do depend on the particular element data (in the present case h^e), and subroutines will be necessary to compute the analogs of these quantities in more general cases.

Now we wish to compute f^e . However, this cannot be done without explicitly knowing what $f = f(x)$ is. In practice, it would be inconvenient to reprogram every time we wanted to solve a problem involving a different function f . Generally a convenient approximation is made. For example, we might replace f by its linear interpolate over each element, namely,

$$f^e = \sum_{a=1}^2 f_a N_a \quad (1.15.4)$$

where $f_a = f(x(\xi_a))$; see Fig. 1.15.1. The notation f^e is used to indicate that the approximation depends upon the mesh. This represents an approximation that is sufficient for most practical applications. (It is, of course, exact for constant or linear “loading” of the element.) Now standardization of input to the program may be facilitated; that is, the nodal values of f are the required data. Let us employ this approximation in the explicit calculation of an element force vector:

$$\begin{aligned} \int_{\Omega^e} N_a(x) f^e(x) dx &= \int_{-1}^{+1} N_a(x(\xi)) f^e(x(\xi)) x_{,\xi}(\xi) d\xi \quad \text{(change of variables)} \\ &= \frac{h^e}{2} \sum_{b=1}^2 \int_{-1}^{+1} N_a(\xi) N_b(\xi) d\xi f_b \quad \text{(by (1.12.8))} \end{aligned} \quad (1.15.5)$$

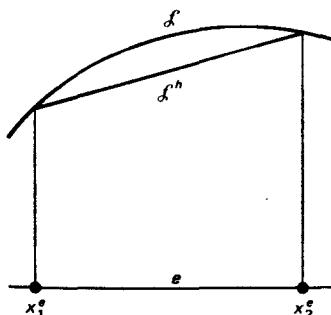


Figure 1.15.1 Approximation of f by piecewise linear interpolation of nodal values.

Carrying out the integrations ($\int_{-1}^{+1} N_a N_b d\xi = (1 + \delta_{ab})/3$) yields

$$\begin{aligned} f^e &= \frac{h^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \quad (+ \text{ boundary terms,} \\ &\quad \text{cf. (1.13.12))} \\ &= \frac{h^e}{6} \begin{Bmatrix} 2f_1 + f_2 \\ f_1 + 2f_2 \end{Bmatrix} \quad (+ \text{ boundary terms}) \end{aligned} \tag{1.15.6}$$

Remark. It can be shown that, under suitable hypotheses, piecewise linear nodal interpolation produces $O(h^2)$ errors in the data; in this case, f . (See [12], pp. 56–57, for basic estimates of interpolation errors.) It can be shown that, in appropriate measures of the error, this produces at worst $O(h^2)$ errors in u^h and u_x^h .

The following exercise indicates that there may be better ways to approximate given data.

Exercise 1. Suppose $f(x)$ is quadratic (i.e., consists of a linear combination of the monomials $1, x, x^2$). Determine a piecewise linear approximation—not necessarily continuous—to f over each element which results in exact nodal values. Hint: The analysis may be performed with respect to one element.

Exercise 2. The equation of a string on an elastic foundation is given by:

$$u_{xx} - \lambda u + f = 0 \quad \text{on } \Omega =]0, 1[$$

where λ , a positive constant, is a measure of the foundation stiffness. Assuming the same boundary conditions as for the problem discussed previously in this chapter, it can be shown that an equivalent weak formulation is:

$$\int_{\Omega} (w_{xx} u_{xx} + w \lambda u) dx = \int_{\Omega} w f dx + w(0)h$$

where $u \in \mathcal{S}$, $w \in \mathcal{V}$, and so on. This can also be written as

$$a(w, u) + (w, \lambda u) = (w, f) + w(0)h$$

i. Let $u^h = v^h + q^h$. Write the Galerkin counterpart of the weak formulation:

$$a(w^h, v^h) + \boxed{} =$$

$$(w^h, f) + w^h(0)h - a(w^h, q^h)$$

$$- \boxed{}$$

ii. Define $K_{AB} = a(N_A, N_B) +$

and

$$k_{ab}^\epsilon = a(N_a, N_b)^\epsilon +$$

iii. Determine k^ϵ explicitly:

$$k^\epsilon = [k_{ab}^\epsilon] = \begin{bmatrix} & \\ & \\ & \end{bmatrix}$$

iv. Show that K is symmetric.

v. Show that K is positive definite. Is it necessary to employ the boundary condition $w^h(1) = 0$? Why?

vi. The Green's function for this problem satisfies

$$g_{xx} - \lambda g + \delta_y = 0$$

and can be written as

$$g(x) = \begin{cases} c_1 e^{px} + c_2 e^{-px}, & 0 \leq x \leq y \\ c_3 e^{px} + c_4 e^{-px}, & y \leq x \leq 1 \end{cases}$$

where $p = \lambda^{1/2}$ and the c 's are determined from the following four boundary and continuity conditions:

$$g(1) = 0$$

$$g_{xx}(0) = 0$$

$$g(y^+) = g(y^-)$$

$$g_{xx}(y^+) = g_{xx}(y^-) - 1$$

Why is the piecewise linear finite element space incapable of attaining nodally exact solutions in this case?

vii. Construct exponential element shape functions $N_1(x)$ and $N_2(x)$ such that

$$u^h(x) = d_1^\epsilon N_1(x) + d_2^\epsilon N_2(x), \quad x \in \Omega^\epsilon$$

where

$$u^h(x) = c_1 e^{px} + c_2 e^{-px}$$

and the c 's are determined from

$$d_a^\epsilon = u^h(x_a^\epsilon), \quad a = 1, 2$$

What is the attribute which this choice of functions attains?

1.16 EXERCISE: BERNOULLI-EULER BEAM THEORY AND HERMITE CUBICS

This problem develops basic finite element results for Bernoulli-Euler beam theory. The strong form of a boundary-value problem for a thin beam (Bernoulli-Euler theory) fixed at one end and subjected to a shear force and moment at the other end, may be stated as follows:

Let the beam occupy the unit interval (i.e., $\Omega =]0, 1[$, $\bar{\Omega} = [0, 1]$).

$$\left. \begin{array}{l} \text{Given } \ell: \Omega \rightarrow \mathbb{R} \text{ and constants } M \text{ and } Q, \text{ find } u: \bar{\Omega} \rightarrow \mathbb{R} \text{ such that} \\ EI u_{,xxxx} = \ell \text{ on } \Omega \quad (\text{transverse equilibrium}) \\ u(1) = 0 \quad (\text{zero transverse displacement}) \\ u_{,x}(1) = 0 \quad (\text{zero slope}) \\ EI u_{,xx}(0) = M \quad (\text{prescribed moment}) \\ EI u_{,xxx}(0) = Q \quad (\text{prescribed shear}) \end{array} \right\} \quad (S)$$

where E is Young's modulus and I is the moment of inertia, both of which are assumed to be constant.

The setup is shown in Fig. 1.16.1.

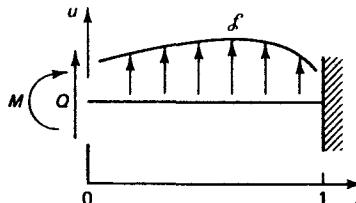


Figure 1.16.1

Let $\mathcal{S} = \mathcal{V} = \{w \mid w \in H^2(\Omega), w(1) = w_{,x}(1) = 0\}$ ⁷. Then a corresponding weak form of the problem is:

$$\left. \begin{array}{l} \text{Given } \ell, M, \text{ and } Q, \text{ find } u \in \mathcal{S} \text{ such that for all } w \in \mathcal{V} \\ a(w, u) = (w, \ell) - w_{,x}(0)M + w(0)Q \end{array} \right\} \quad (W)$$

where

$$a(w, u) = \int_0^1 w_{,xx} EI u_{,xx} dx$$

$$(w, \ell) = \int_0^1 w \ell dx$$

⁷ $w \in H^2(\Omega)$ essentially means that $w_{,xx}$ is square-integrable (i.e., $\int_0^1 (w_{,xx})^2 dx < \infty$).

The collection of functions, \mathcal{V} , may be thought of as the space of finite strain-energy configurations of the beam, satisfying the kinematic (essential) boundary conditions at $x = 1$. It is a consequence of Sobolev's theorem that each $w \in \mathcal{V}$ is continuously differentiable. For reasonable ℓ , these problems possess unique solutions.

Let $\mathcal{S}^h = \mathcal{V}^h$ be a finite-dimensional approximation of \mathcal{S} . In particular, we assume $w^h \in \mathcal{V}^h$ satisfies $w^h(1) = w_{,x}(1) = 0$.

The Galerkin statement of the problem goes as follows:

$$(G) \left\{ \begin{array}{l} \text{Given } \ell, M, \text{ and } Q, \text{ find } u^h \in \mathcal{S}^h \text{ such that for all } w^h \in \mathcal{V}^h \\ \boxed{a(w^h, u^h) = (w^h, \ell) - w_{,x}(0)M + w^h(0)Q} \end{array} \right.$$

a. Assuming all functions are smooth and bounded, show that the solutions of (S) and (W) are identical. What are the natural boundary conditions?

b. Assume $0 = x_1 < x_2 < \dots < x_{n+1} = 1$ and $\mathcal{V}^h = \{w^h \mid w^h \in C^1(\bar{\Omega}), w^h(1) = w_{,x}(1) = 0, \text{ and } w^h \text{ restricted to } [x_A, x_{A+1}] \text{ is a cubic polynomial (i.e., consists of a linear combination of } 1, x, x^2, x^3)\}^8$. This is a space of *piecewise cubic Hermite shape functions*. Observe that $w^h \in \mathcal{V}^h$ need not have continuous second derivatives at the nodes.

On each subinterval, show that w^h may be written as

$$w^h(x) = N_1(x)w^h(x_1) + N_3(x)w^h(x_2) + N_2(x)w_{,x}(x_1) + N_4(x)w_{,x}(x_2)$$

where

$$N_1(x) = \frac{-(x - x_2)^2[-h + 2(x_1 - x)]}{h^3}$$

$$N_2(x) = \frac{(x - x_1)(x - x_2)^2}{h^2}$$

$$N_3(x) = \frac{(x - x_1)^2[h + 2(x_2 - x)]}{h^3}$$

$$N_4(x) = \frac{(x - x_1)^2(x - x_2)}{h^2}$$

Hint: Let $w^h(x) = c_1 + c_2x + c_3x^2 + c_4x^3$, where the c 's are constants. Determine them by requiring the following four conditions hold:

$$w^h(x_1) = c_1 + c_2x_1 + c_3x_1^2 + c_4x_1^3$$

$$w^h(x_2) = c_1 + c_2x_2 + c_3x_2^2 + c_4x_2^3$$

$$w_{,x}(x_1) = c_2 + 2c_3x_1 + 3c_4x_1^2$$

$$w_{,x}(x_2) = c_2 + 2c_3x_2 + 3c_4x_2^2$$

⁸The notation $w^h \in C^1$ means w^h is continuously differentiable.

Sketch the element functions N_1 , N_2 , N_3 , and N_4 , and their typical global counterparts.

The finite element space described in part (b) results in *exact* nodal displacements and slopes (first derivatives), analogous to the case presented in Sec. 1.10. In part (g), you are asked to prove this. In problems of beam bending we are generally interested in curvatures (second derivatives) for bending moment calculations.

c. Locate the optimal curvature points in the sense of Barlow. *Warning:* The algebraic manipulations can be tiresome unless certain simplifications are observed. If we work in the ξ -element coordinate system introduced in Sec. 1.12 (recall $\xi = (2x - x_A - x_{A+1})/h_A$), the location of the Barlow curvature points may be expressed as $\xi = \pm 1/\sqrt{3}$. That is, there are two symmetrically spaced optimal locations to compute curvature.

d. What is the rate of convergence of curvature at these points? (*Ans.* $O(h^3)$).

e. If the segment of the beam $[x_A, x_{A+1}]$ is unloaded (i.e., $u_{,xxxx} = 0$, where u is the exact solution), which points are optimal?

f. Assume $n_{el} = 1$ (i.e., one element) and $f(x) = c = \text{constant}$. Set up and solve the Galerkin-finite element equations. Plot u^h and u ; $u_{,x}^h$ and $u_{,x}$; and $u_{,xx}^h$ and $u_{,xx}$. Indicate the locations of the Barlow curvature points.

g. Prove that

$$u^h(x_A) = u(x_A)$$

$$u_{,x}^h(x_A) = u_{,x}(x_A)$$

where x_A is a typical node (i.e., prove *the displacements and slopes are exact at the nodes*). To do the second part you will have to be familiar with the *dipole*, $\delta_{,x}(x - x_A)$, which is the generalized derivative of the delta function.

h. Show that the Barlow curvature points are exact when $f(x) = c = \text{constant}$.

i. Why do we require that the functions in \mathcal{V}^h have continuous first derivatives?

j. Calculate the 4×4 element stiffness matrix,

$$k_{pq}^e = \int_{x_1^e}^{x_2^e} N_{z,xx} EI N_{q,zz} dx \quad 1 \leq p, q \leq 4$$

where $h^e = x_2^e - x_1^e$.

k. (See the exercise in Sec. 1.8.) Consider the weak formulation. Assume $w \in \mathcal{V}$ and $u \in \mathcal{S}$ are smooth on element interiors (i.e., on $[x_A, x_{A+1}]$) but may exhibit discontinuities in second, and higher, derivatives across element boundaries. (Functions of this type contain the piecewise-cubic Hermite functions.) Show that

$$\begin{aligned}
 0 = & \sum_{A=1}^n \int_{x_A}^{x_{A+1}} w(EI u_{,xxxx} - f) dx \\
 & - w_{,x}(0)(EI u_{,xx}(0^+) - M) \\
 & + w(0)(EI u_{,xxx}(0^+) - Q) \\
 & - \sum_{A=2}^n w_{,x}(x_A) EI (u_{,xx}(x_A^+) - u_{,xx}(x_A^-)) \\
 & + \sum_{A=2}^n w(x_A) EI (u_{,xxx}(x_A^+) - u_{,xxx}(x_A^-))
 \end{aligned}$$

from which it may be concluded that the Euler-Lagrange conditions are

- i. $EI u_{,xxxx}(x) = f(x)$, where $x \in]x_A, x_{A+1}[$ and $A = 1, 2, \dots, n$
- ii. $EI u_{,xx}(0^+) = M$
- iii. $EI u_{,xxx}(0^+) = Q$
- iv. $EI u_{,xx}(x_A^+) = EI u_{,xx}(x_A^-)$, where $A = 2, 3, \dots, n$
- v. $EI u_{,xxx}(x_A^+) = EI u_{,xxx}(x_A^-)$, where $A = 2, 3, \dots, n$

Note that (i) is the equilibrium equation *restricted to the element interiors*, and (iv) and (v) are continuity conditions across element boundaries of moment and shear, respectively. Contrast these results with those obtained for functions w and u , which are *globally* smooth.

The Galerkin finite element formulation yields a solution that *approximates* (i) through (v).

An Elementary Discussion of Continuity, Differentiability, and Smoothness

Throughout Chapter 1 we have introduced mathematical terminologies and ideas in a gradual, as-needed format. Many of these ideas had to do with the continuity and differentiability of functions. The presentation was, admittedly, somewhat vague on these points in order that the main ideas would not be overencumbered. Careful characterization of the properties of functions is an essential ingredient in the development and analysis of finite element methods. However, to pursue this subject deeply would take us into the realm of serious mathematical analysis, which is outside the scope of this book. Nevertheless, we feel compelled to say a few additional words on the subject to round out the presentation in Chapter 1 and to expose the reader to notations and ideas that will probably be encountered if he or she attempts to read published papers on finite elements.

The discussion here will be restricted to one dimension. In Chapter 1 we spoke of continuously differentiable functions. If we have a grasp of the notion of a continuous function, then continuously differentiable functions pose no problem.

Definition: A function $f: \Omega \rightarrow \mathbb{R}$ (recall $\Omega =]0, 1[$) is said to be *k-times continuously differentiable*, or *of class $C^k = C^k(\Omega)$* , if its derivatives of order j , where $0 \leq j \leq k$, exist and are continuous functions.

A C^0 function is simply a continuous function. A C^∞ function is one that possesses a continuous derivative of any order (i.e., $j = 0, 1, \dots, \infty$).

Definition: A function f is said to be *of class C_b^k* if it is C^k and bounded (i.e., $|f(x)| < c$, where c is a constant, for all $x \in \Omega$).

Example 1

The functions defined by monomials (i.e., $f(x) = 1, x, x^2$, etc.) are C_∞ .

Example 2

The function $f(x) = 1/x$ is continuous on Ω , as are all its derivatives; hence it is C^∞ , but it is not bounded (i.e., there does not exist a constant c such that $|1/z| < c$ for all $x \in \Omega$; see Fig. 1.I.1). Consequently this function is not of class C_b^k for any $k \geq 0$.

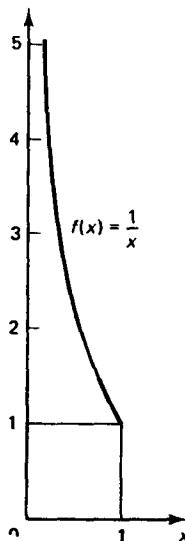


Figure 1.I.1 A continuous function that is not bounded.

Example 3

The function

$$f(x) = \begin{cases} x, & x \leq \frac{1}{2} \\ 1/2, & x > \frac{1}{2} \end{cases} \quad (1.I.1)$$

is continuous but not continuously differentiable (i.e., it is C_b^0 but not C_b^1). Functions in C_b^k , $k \geq 1$, but not in C_b^{k+1} may be constructed by integrating (1.I.1) k times. For example,

$$f(x) = \begin{cases} \frac{x^2}{2}, & x \leq \frac{1}{2} \\ \frac{(x - \frac{1}{2})}{2}, & x > \frac{1}{2} \end{cases} \quad (1.I.2)$$

is in C_b^1 but not C_b^2 . (The reader may wish to verify this.)

There is no universally accepted definition of what is meant by a "smooth" function. However, it is generally taken to mean that at least one derivative exists and is continuous (i.e., either C^1 or C_b^1) and sometimes means $k > 1$, even ∞ .

The C^k and C_b^k functions employ the classical notion of a derivative in their definitions. If we employ the closed unit interval, $\bar{\Omega} = [0, 1]$, instead of $\Omega =]0, 1[$, the difference between C^k and C_b^k disappears. This is because if f is $C^k([0, 1])$, $f(0)$ and $f(1)$ are real numbers and are not allowed to be ∞ . Thus unboundedness, as in the example above, is precluded. Very often, we think of C^k functions in this light. However, in some situations the differences between $C^k(\Omega)$ and $C_b^k(\Omega)$ must be kept in mind.

Generally, finite element functions are smooth on element interiors (there are exceptions, however) but possess only low-order continuity across element boundaries. One might be tempted to characterize them as locally smooth but globally "rough." The piecewise linear finite element functions discussed in Sec. 1.8 are of class C_b^0 . The Hermite cubics employed in Sec. 1.16 are C_b^1 . To calculate derivatives of such functions we need to employ the notion of a "generalized derivative," as was used in solving the Green's function problem of Sec. 1.10. For example, the first derivative of a piecewise linear finite element function is a generalized step function; the second derivative is a generalized Dirac delta function (i.e., delta functions, of various amplitudes, acting at the nodes). In the case of the Hermite cubics, the first derivative is continuous, the second a generalized step function, and so on. We have seen in Sec. 1.16 that other generalized functions also arise in the analysis of finite element behavior (namely, the dipole). The useful examples of generalized functions are by no means exhausted by what we have seen thus far. However, the ones we have introduced are perhaps the most basic.

In the mathematical analysis of boundary-value problems, and consequently in finite element analysis, we need to introduce classes of functions that possess generalized derivatives and, in addition, certain integrability properties. We have encountered such functions in the statements of weak formulations in Sec. 1.3 and 1.16. These are particular examples of *Sobolev spaces of functions* defined as follows:

$$H^k = H^k(\Omega) = \{w \mid w \in L_2; w_{,x} \in L_2; \dots; w_{,\underbrace{x}_{k \text{ times}}} \in L_2\} \quad (1.1.3)$$

where

$$L_2 = L_2(\Omega) = \{w \mid \int_0^1 w^2 dx < \infty\} \quad (1.1.4)$$

In words, the Sobolev space of degree k , denoted by H^k , consists of functions that possess square-integrable generalized derivatives through order k . A square-integrable function is called an L_2 -function, by virtue of (1.1.4). From (1.1.3), we see that $H^0 = L_2$ and that $H^{k+1} \subset H^k$. The Sobolev spaces are the most important for studying elliptic boundary-value problems.

The question naturally arises as to the relation between Sobolev spaces and the classical spaces of differentiable functions introduced previously. In particular, when is an H^k -function smooth in the classical sense? The answer is provided by *Sobolev's theorem*, which states that, in one dimension, $H^{k+1} \subset C_b^k$. That is, if a function is of class H^{k+1} , then it is actually a C_b^k function. For example, in Sec. 1.3 we required H^1 functions. By Sobolev's theorem, such functions are, additionally, continuous and bounded. In Sec. 1.16, we employed H^2 functions. These are C_b^1 by Sobolev's theorem and thus possess bounded, continuous, classical derivatives.

Certain “singularities” are precluded by square-integrability. For example, $x^{-1/4}$ is in L_2 , but $x^{-1/2}$ is not. (Verify!) Such considerations become important in many physical circumstances (e.g., in fracture mechanics).

The number of other types of function spaces that arise in mathematical analysis is large, and many are difficult to comprehend without serious training in “functional analysis.” These topics are outside the scope of this book. The reader who wishes to delve further may consult [13, 14, 15] and references therein.

REFERENCES

Section 1.2

1. A. R. Mitchell and D. F. Griffiths, *The Finite Difference Method in Partial Differential Equations*. New York: John Wiley, 1980.

Section 1.5

2. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*. Englewood Cliffs, N. J.: Prentice-Hall, 1973.
3. B. A. Finlayson, *The Method of Weighted Residuals and Variational Principles*. New York: Academic Press, 1972.
4. B. A. Finlayson and L. E. Scriven, “The Method of Weighted Residuals—A Review,” *Applied Mechanics Reviews*, 19, (1966), 735–738.

Section 1.10

5. I. Stakgold, *Boundary-Value Problems of Mathematical Physics*. Vols. I and II, New York: Macmillan, 1968.
6. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*. Englewood Cliffs, N. J.: Prentice-Hall, 1973.
7. J. E. Marsden, *Elementary Classical Analysis*. San Francisco: W. H. Freeman, 1974.
8. J. Barlow, “Optimal Stress Locations in Finite Element Models,” *International Journal for Numerical Methods in Engineering*, 10 (1976), 243–251.
9. E. Popov, *Introduction to Mechanics of Solids*. Englewood Cliffs, N. J.: Prentice-Hall, 1968.

Section 1.13

10. M. J. Turner, R. W. Clough, H. C. Martin, and L. J. Topp, “Stiffness and deflection analysis of complex structures,” *Journal of Aeronautical Sciences*, 23 (1956), 805–823.

Section 1.15

11. J. E. Marsden, *Elementary Classical Analysis*. San Francisco: W. H. Freeman, 1974.
12. P. J. Davis, *Interpolation and Approximation*. New York: Blaisdell, 1963.

Appendix 1.I

13. P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*. New York: North-Holland, 1978.
14. J. T. Oden and J. N. Reddy, *An Introduction to the Mathematical Theory of Finite Elements*. New York: Academic Press, 1978.
15. J. T. Oden, *Applied Functional Analysis*. Englewood Cliffs, N. J.: Prentice-Hall, 1979.

2

Formulation of Two- and Three-dimensional Boundary-value Problems

2.1 INTRODUCTORY REMARKS

It makes no sense to attempt to “solve” a boundary-value problem without a precise knowledge of what the problem is. The truth of this statement seems self-evident. Unfortunately, attempts are often made to solve vaguely defined problems, creating considerable confusion and, sometimes, totally erroneous results. In this chapter we present precise statements of multidimensional boundary-value problems in classical linear heat conduction and elastostatics. The presentation is similar in many respects to that for the one-dimensional model problem of Chapter 1. In particular, we discuss strong and weak forms, their equivalence, corresponding Galerkin formulations, the definitions of element arrays, and pertinent data processing concepts. In multi-dimensions, the data processing ideas necessarily become more involved. The reader is urged to study them carefully as they are necessary in order to understand the computer implementation of finite element techniques.

2.2 PRELIMINARIES

Let n_{sd} ($= 2$ or 3) denote the *number of space dimensions* of the problem under consideration. Let $\Omega \subset \mathbb{R}^{n_{sd}}$ be an open set¹ with piecewise smooth boundary Γ . A general point in $\mathbb{R}^{n_{sd}}$ is denoted by x . We will identify the point x with its position vector emanating from the origin of $\mathbb{R}^{n_{sd}}$. The *unit outward normal vector to Γ* is denoted by n .

¹ For our purposes, it is sufficient to think of an open set as one without its boundary.

We shall employ the following alternative representations for \mathbf{x} and \mathbf{n} :

$$(n_{sd} = 2): \quad \mathbf{x} = \{x_i\} = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} x \\ y \end{Bmatrix} \quad \mathbf{n} = \{n_i\} = \begin{Bmatrix} n_1 \\ n_2 \end{Bmatrix} = \begin{Bmatrix} n_x \\ n_y \end{Bmatrix} \quad (2.2.1)$$

$$(n_{sd} = 3): \quad \mathbf{x} = \{x_i\} = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} \quad \mathbf{n} = \{n_i\} = \begin{Bmatrix} n_1 \\ n_2 \\ n_3 \end{Bmatrix} = \begin{Bmatrix} n_x \\ n_y \\ n_z \end{Bmatrix} \quad (2.2.2)$$

where x_i and n_i , $1 \leq i \leq n_{sd}$, are the Cartesian components of \mathbf{x} and \mathbf{n} , respectively; see Figure 2.2.1. Unless otherwise specified we shall work in terms of Cartesian components of vectors and tensors.

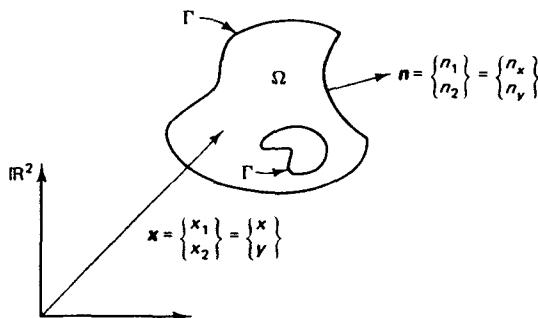


Figure 2.2.1

We assume that Γ admits the decomposition

$$\Gamma = \overline{\Gamma_q \cup \Gamma_k} \quad (2.2.3)$$

where

$$\Gamma_q \cap \Gamma_k = \emptyset \quad (2.2.4)$$

and Γ_q and Γ_k are open sets in Γ . The notations are defined as follows: \cup is the *set union* symbol. Thus $\Gamma_q \cup \Gamma_k$ means the set of all points \mathbf{x} contained in either Γ_q or Γ_k . Also, \cap is the *set intersection* symbol. Thus $\Gamma_q \cap \Gamma_k$ means the set of all points contained in both Γ_q and Γ_k . The *empty set* is denoted by \emptyset . Thus (2.2.4) means that there is no point \mathbf{x} contained in both Γ_q and Γ_k (i.e., Γ_q and Γ_k do not intersect or overlap). A bar above a set means *set closure*, i.e., the union of the set with its boundary. Thus

$$\overline{\Omega} = \Omega \cup \Gamma \quad (2.2.5)$$

To understand the meaning of $\overline{\Gamma_q \cup \Gamma_k}$, we must define the boundaries of Γ_q and Γ_k in Γ . We shall do this with the aid of an example.

Example 1

Let $\Omega = \{\mathbf{x} \in \mathbb{R}^2 \mid x^2 + y^2 < 1\}$, i.e., the interior of the unit disc. The boundary of Ω is $\Gamma = \{\mathbf{x} \in \mathbb{R}^2 \mid x^2 + y^2 = 1\}$, i.e., the unit circle. Let

$$\Gamma_q = \Gamma \cap \{\mathbf{x} \in \mathbb{R}^2 \mid y > 0\} \quad (2.2.6)$$

The “boundary of Γ_a ” consists of the endpoints of the upper semicircle, i.e., $\{x \in \mathbb{R}^2 \mid x = -1, y = 0, \text{ and } x = +1, y = 0\}$. Thus

$$\overline{\Gamma_a} = \Gamma \cap \{x \in \mathbb{R}^2 \mid y \geq 0\} \quad (2.2.7)$$

Similarly, let

$$\Gamma_k = \Gamma \cap \{x \in \mathbb{R}^2 \mid y < 0\} \quad (2.2.8)$$

Thus

$$\overline{\Gamma_k} = \Gamma \cap \{x \in \mathbb{R}^2 \mid y \leq 0\} \quad (2.2.9)$$

Clearly

$$\overline{\Gamma_a \cup \Gamma_k} = \overline{\Gamma_a} \cup \overline{\Gamma_k} = \overline{\Gamma} \quad (2.2.10)$$

These sets are depicted in Fig. 2.2.2.

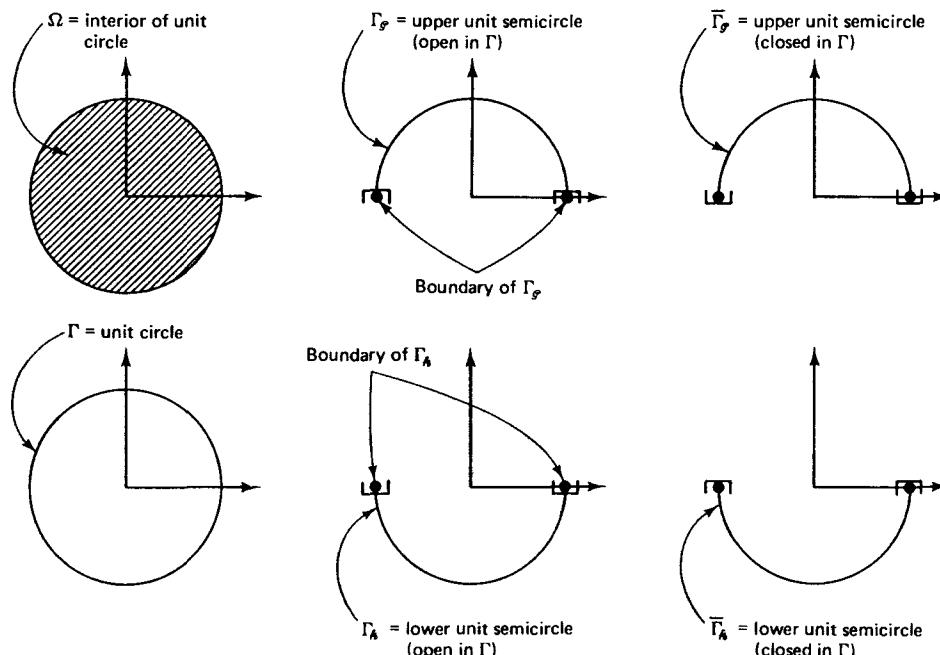


Figure 2.2.2

We shall assume throughout that $\Gamma_a \neq \emptyset$ but allow for the case $\Gamma_k = \emptyset$.

Let the indices i, j, k, l , run over the values $1, \dots, n_{sd}$. Differentiation is denoted by a comma (e.g., $u_{,i} = u_{,xi} = \partial u / \partial x_i$) and repeated indices imply summation (e.g., in \mathbb{R}^3 , $u_{,ii} = u_{,11} + u_{,22} + u_{,33} = \partial^2 u / \partial x^2 + \partial^2 u / \partial y^2 + \partial^2 u / \partial z^2$). The summation convention only applies to the indices i, j, k , and l and only to two repeated indices. If there are three, or more, repeated indices in an expression, then

the summation convention is *not* in effect. (This is in keeping with the usual convention.)

Divergence theorem. Let $f : \bar{\Omega} \rightarrow \mathbb{R}$ be C^1 . Then

$$\int_{\Omega} f_{,i} d\Omega = \int_{\Gamma} f n_i d\Gamma \quad (2.2.11)$$

The proof may be found in [1].

Integration by parts. Let f be as above and also let $g : \bar{\Omega} \rightarrow \mathbb{R}$ be C^1 . Then

$$\int_{\Omega} f_{,i} g d\Omega = - \int_{\Omega} f g_{,i} d\Omega + \int_{\Gamma} f g n_i d\Gamma \quad (2.2.12)$$

Proof. We integrate the identity (i.e., “product rules of differentiation,” see [1])

$$(fg)_{,i} = f_{,i}g + fg_{,i}$$

to get

$$\int_{\Omega} (fg)_{,i} d\Omega = \int_{\Omega} f_{,i}g d\Omega + \int_{\Omega} fg_{,i} d\Omega$$

and then use the divergence theorem to convert the left-hand side into a boundary integral. ■

2.3 CLASSICAL LINEAR HEAT CONDUCTION: STRONG AND WEAK FORMS; EQUIVALENCE

Let q_i denote (Cartesian components of) the **heat flux vector**, let u be the **temperature**, and let ϕ be the **heat supply per unit volume**. Assume the heat flux vector is defined in terms of the temperature gradient by the **generalized Fourier law**²:

$$q_i = -\kappa_{ij} u_{,j}, \quad \kappa_{ij} = \kappa_{ji} \quad (\text{symmetry}) \quad (2.3.1)$$

where the **conductivities**, κ_{ij} 's, are given functions of x . (If the κ_{ij} 's are constant throughout Ω , the body is said to be **homogeneous**.) The **conductivity matrix**, $\kappa = [\kappa_{ij}]$, is assumed positive definite (see the definition in Sec. 1.9). The most common situation in practice is the **isotropic case** in which $\kappa_{ij}(x) = \kappa(x)\delta_{ij}$, where δ_{ij} is the Kronecker delta.

²The generalized Fourier law is a **constitutive equation**, or **equation of state**, which reflects the heat conduction properties of the body (i.e., Ω) under consideration.

A formal statement³ of the strong form of the boundary-value problem is as follows:

$$(S) \left\{ \begin{array}{l} \text{Given } f : \Omega \rightarrow \mathbb{R}, g : \Gamma_g \rightarrow \mathbb{R} \text{ and } h : \Gamma_h \rightarrow \mathbb{R}, \text{ find } u : \bar{\Omega} \rightarrow \mathbb{R} \text{ such that} \\ q_{i,i} = f \quad \text{in } \Omega \quad (\text{heat equation}) \quad (2.3.2) \\ u = g \quad \text{on } \Gamma_g \quad (2.3.3)^4 \\ -q_i n_i = h \quad \text{on } \Gamma_h \quad (2.3.4) \end{array} \right.$$

where q_i is defined by (2.3.1).

The functions g and h are the *prescribed boundary temperature and heat flux*, respectively. This problem possesses a unique solution for appropriate restrictions on the given data.

In more mathematical terminology, (2.3.2) is a generalized Poisson equation, (2.3.3) is a Dirichlet boundary condition, and (2.3.4) is a Neumann boundary condition.

We shall now construct a weak formulation of the boundary-value problem analogous to that for the one-dimensional problem of Chapter 1. In particular, (2.3.3) and (2.3.4) will be treated as essential and natural boundary conditions, respectively. As before, let \mathcal{S} denote the trial solution space and \mathcal{V} the variation space. This time \mathcal{S} and \mathcal{V} consist of real-valued functions defined on $\bar{\Omega}$ satisfying certain smoothness requirements, such that all members of \mathcal{S} satisfy (2.3.3), whereas if $w \in \mathcal{V}$, then

$$w = 0 \quad \text{on } \Gamma_g \quad (2.3.5)$$

The weak formulation of the problem goes as follows:

$$(W) \left\{ \begin{array}{l} \text{Given } f : \Omega \rightarrow \mathbb{R}, g : \Gamma_g \rightarrow \mathbb{R} \text{ and } h : \Gamma_h \rightarrow \mathbb{R}, \text{ find } u \in \mathcal{S} \text{ such that} \\ \text{for all } w \in \mathcal{V} \\ \boxed{- \int_{\Omega} w_{,i} q_i d\Omega = \int_{\Omega} w f d\Omega + \int_{\Gamma_h} w h d\Gamma} \quad (2.3.6) \\ \text{where } q_i \text{ is defined by (2.3.1).} \end{array} \right.$$

Theorem. Assume all functions involved are smooth enough to justify the manipulations. Then a solution of (S) is a solution of (W) and vice versa.

Proof. 1. Assume u is the solution of (S) . By virtue of (2.3.3), $u \in \mathcal{S}$. Pick any $w \in \mathcal{V}$ and proceed as follows:

³By a formal statement, we mean one in which we do not precisely delineate the spaces to which the functions involved belong.

⁴The statement $u = g$ on Γ_g means $u(x) = g(x)$ for all $x \in \Gamma_g$, and so on.

$$\begin{aligned}
 0 &= \int_{\Omega} w(\underbrace{q_{i,i} - \ell}_{0}) d\Omega && \text{(heat equation, i.e., (2.3.2))} \\
 &= - \int_{\Omega} w_{,i} q_i d\Omega + \int_{\Gamma} w q_i n_i d\Gamma - \int_{\Omega} w \ell d\Omega && \text{(integration by parts)} \\
 &= - \int_{\Omega} w_{,i} q_i d\Omega - \int_{\Gamma_k} w h d\Gamma - \int_{\Omega} w \ell d\Omega && \text{(generalized Fourier law, } w = 0 \text{ on } \Gamma_u \text{, and heat flux boundary condition, i.e. (2.3.4))} \\
 \end{aligned}$$

Therefore (2.3.6) is satisfied and so u is a solution of (W) .

2. Assume u is the solution of (W) . Then $u = \varphi$ on Γ_u , and for all $w \in \mathcal{V}$

$$\begin{aligned}
 0 &= \int_{\Omega} w_{,i} q_i d\Omega + \int_{\Omega} w \ell d\Omega + \int_{\Gamma_k} w h d\Gamma && \text{(by (2.3.6))} \\
 &= \int_{\Omega} w(-q_{i,i} + \ell) d\Omega + \int_{\Gamma_k} w(n_i q_i + h) d\Gamma && \text{(2.3.7)}
 \end{aligned}$$

Let

$$\alpha = -q_{i,i} + \ell$$

$$\beta = q_i n_i + h$$

To show (2.3.2) and (2.3.4) are satisfied and thus complete the proof, we must prove that

$$\alpha = 0 \quad \text{on } \Omega$$

$$\beta = 0 \quad \text{on } \Gamma_k$$

First pick $w = \alpha \phi$ where

- i. $\phi > 0$ on Ω ;
- ii. $\phi = 0$ on Γ ; and
- iii. ϕ is smooth.

(These conditions insure that $w \in \mathcal{V}$) With this choice for w , (2.3.7) becomes

$$0 = \int_{\Omega} \alpha^2 \phi d\Omega$$

which implies $\alpha = 0$ on Ω .

Now pick $w = \beta \psi$, where

- i'. $\psi > 0$ on Γ_k ;
- ii'. $\psi = 0$ on Γ_u ; and
- iii'. ψ is smooth.

(These conditions insure that $w \in \mathcal{V}$) With this choice for w , (2.3.7) becomes

(making use of $\alpha = 0$):

$$0 = \int_{\Gamma_k} \beta^2 \psi d\Gamma$$

from which it follows that $\beta = 0$ on Γ_k . Thus u is a solution of (S) . ■

It is convenient to introduce an abstract version of (2.3.6). Let

$$a(w, u) = \int_{\Omega} w_{,i} \kappa_{ij} u_{,j} d\Omega \quad (2.3.8)$$

$$(w, f) = \int_{\Omega} w f d\Omega \quad (2.3.9)$$

$$(w, h)_{\Gamma} = \int_{\Gamma_k} w h d\Gamma \quad (2.3.10)$$

Then (2.3.6) may be written as

$$a(w, u) = (w, f) + (w, h)_{\Gamma} \quad (2.3.11)$$

Exercise 1. Verify that $a(\cdot, \cdot)$, (\cdot, \cdot) and $(\cdot, \cdot)_{\Gamma}$, as just defined, are symmetric bilinear forms. (Note that the symmetry of $a(\cdot, \cdot)$ follows from the symmetry of the conductivities.)

In manipulating terms in theories involving vector and tensor quantities, the indicial notation used is very explicit and convenient. However, when we come to the Galerkin formulation analogous to (2.3.6), additional indices necessarily appear. The situation becomes very complicated due to the greater number of indices involved and due to the ranges of the various indices being different. When we come to elasticity theory, the situation is even worse as the corresponding terms have an even greater number of indices. For these reasons it is useful at this point to adopt an *index-free* notation for (2.3.6). Aside from stemming the proliferation of indices, we shall find later on that this formulation is conducive to the computer implementation of the element arrays, especially in more complicated situations such as elasticity.

In introducing our index-free notation we shall assume for definiteness that $n_{sd} = 2$. Let ∇ denote the gradient operator; thus

$$\nabla u = \{u_{,i}\} = \begin{Bmatrix} u_{,1} \\ u_{,2} \end{Bmatrix} \quad (2.3.12)$$

$$\nabla w = \{w_{,i}\} = \begin{Bmatrix} w_{,1} \\ w_{,2} \end{Bmatrix} \quad (2.3.13)$$

In the case of two space dimensions, the conductivity matrix may be written as

$$\kappa = [\kappa_{ij}] = \begin{bmatrix} \kappa_{11} & \kappa_{12} \\ \kappa_{21} & \kappa_{22} \end{bmatrix} \quad (\text{symmetric}) \quad (2.3.14)$$

In the isotropic case, (2.3.14) simplifies to

$$\kappa = \kappa[\delta_{ij}] = \kappa \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (2.3.15)$$

In terms of the above expressions, the integrand of (2.3.8) may be written in index-free fashion:

$$w_{,i} \kappa_{ij} u_{,j} = (\nabla w)^T \kappa (\nabla u) \quad (2.3.16)$$

Thus in place of (2.3.8) we may write

$$a(w, u) = \int_{\Omega} (\nabla w)^T \kappa (\nabla u) d\Omega \quad (2.3.17)$$

Exercise 2. Verify (2.3.16) for the cases $n_{sd} = 2$ and 3.

2.4 HEAT CONDUCTION: GALERKIN FORMULATION; SYMMETRY AND POSITIVE-DEFINITENESS OF K

Let \mathcal{S}^h and \mathcal{V}^h be finite-dimensional approximations to \mathcal{S} and \mathcal{V} , respectively. We assume all members of \mathcal{V}^h vanish, or vanish approximately, on Γ_g and that each member of \mathcal{S}^h admits the representation

$$u^h = v^h + q^h \quad (2.4.1)$$

where $v^h \in \mathcal{V}^h$ and q^h results in satisfaction, or at least approximate satisfaction, of the boundary condition $u = q$ on Γ_g .

The Galerkin formulation is given as follows:

$$(G) \left\{ \begin{array}{l} \text{Given } \ell, q, \text{ and } h \text{ [as in (W)], find } u^h = v^h + q^h \in \mathcal{S}^h \text{ such that for all} \\ w^h \in \mathcal{V}^h \text{ (cf. Sec. 1.5):} \\ \boxed{a(w^h, v^h) = (w^h, \ell) + (w^h, h)_\Gamma - a(w^h, q^h)} \end{array} \right. \quad (2.4.2)$$

We now view our domain as “discretized” into element domains Ω^e , $1 \leq e \leq n_{el}$. In two dimensions the element domains might be simply triangles and quadrilaterals; see Fig. 2.4.1. Nodal points may exist anywhere on the domain but most frequently appear at the element vertices and interelement boundaries and less often in the interiors.

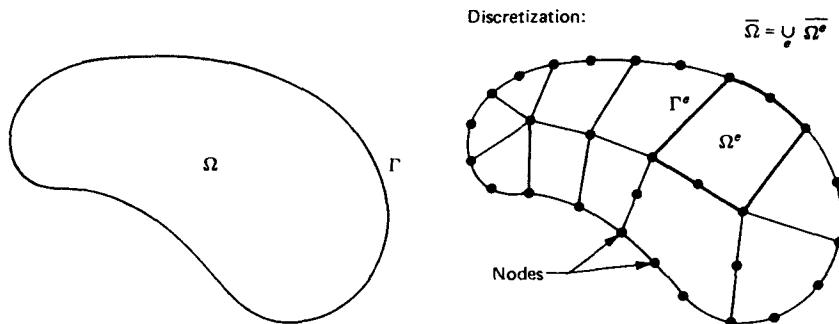


Figure 2.4.1

In Sec. 1.9 the global nodal ordering and ordering of equations in the matrix system coincided. In multidimensional applications this would prove to be an inconvenient restriction with regard to data preparation.

In what follows, a more flexible scheme is described. Let $\eta = \{1, 2, \dots, n_{np}\}$, the *set of global node numbers* where n_{np} is the *number of nodal points*. By the terminology q -node we shall mean a node, A , at which it is prescribed that $u^h = q$. Let $\eta_q \subset \eta$ be the set of “ q -nodes.” The *complement of η_q in η* , denoted by $\eta - \eta_q$, is the set of nodes at which u^h is to be determined. The number of nodes in $\eta - \eta_q$ equals n_{eq} , the *number of equations*.

A typical member of \mathcal{V}^h is assumed to have the form

$$w^h(x) = \sum_{A \in \eta - \eta_q} N_A(x) c_A \quad (2.4.3)$$

where N_A is the shape function associated with node number A and c_A is a constant. We assume throughout that $w^h = 0$ if and only if $c_A = 0$ for each $A \in \eta - \eta_q$. Likewise

$$v^h(x) = \sum_{A \in \eta - \eta_q} N_A(x) d_A \quad (2.4.4)$$

where d_A is the unknown at node A (i.e., temperature) and

$$\varphi^h(x) = \sum_{A \in \eta_q} N_A(x) q_A, \quad q_A = q(x_A) \quad (2.4.5)$$

From (2.4.5), we see that φ^h has been *defined* to be the nodal interpolate of q by way of the shape functions.⁵ Consequently, φ^h will be, generally, only an approximation of q . See Fig. 2.4.2. Additional sources of error are (1) the use of approximations ℓ^h and h^h in place of ℓ and h , respectively; and (2) domain approximations in which the element boundaries do not exactly coincide with Γ . Analyses of these approximations are presented in Strang and Fix [2].

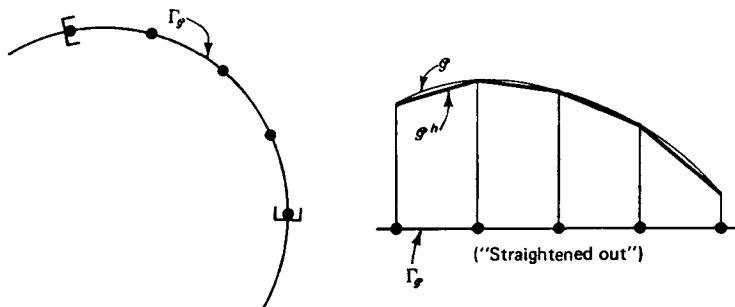


Figure 2.4.2 Piecewise linear approximation of boundary data (schematic).

Substituting (2.4.3)–(2.4.5) into (2.4.2) and arguing as in Sec. 1.6, results in

$$\sum_{B \in \eta - \eta_q} a(N_A, N_B) d_B = (N_A, \ell) + (N_A, h)_\Gamma - \sum_{B \in \eta_q} a(N_A, N_B) q_B, \quad A \in \eta - \eta_q \quad (2.4.6)$$

To define the global stiffness matrix and force vector, we need to first specify the global ordering of equations. For this purpose we introduce the ID array, sometimes called the *destination array*, which assigns to node A the corresponding global equation number, viz.,

$$\text{ID}(A) = \begin{cases} P & \text{if } A \in \eta - \eta_q \\ 0 & \text{if } A \in \eta_q \end{cases} \quad \begin{matrix} \text{Global equation} \\ \text{number} \end{matrix} \quad (2.4.7)$$

⁵This is not the only possibility, nor the best from the standpoint of accuracy. However, in practice it is generally the most convenient.

where $1 \leq P \leq n_{eq}$. The dimension of ID is n_{np} . As may be seen from (2.4.7), nodes at which q is prescribed are assigned “equation number” zero. An example of the setup of ID and other important data-processing arrays is presented in Sec. 2.6.

The matrix equivalent of (2.4.6) is given as follows:

$$\mathbf{Kd} = \mathbf{F} \quad (2.4.8)$$

$$\mathbf{K} = [K_{PQ}], \quad d = \{d_Q\}, \quad \mathbf{F} = \{F_P\}, \quad 1 \leq P, Q \leq n_{eq} \quad (2.4.9)$$

$$K_{PQ} = a(N_A, N_B), \quad P = \text{ID}(A), \quad Q = \text{ID}(B) \quad (2.4.10)$$

$$F_P = (N_A, f) + (N_A, h)_\Gamma - \sum_{B \in \eta_q} a(N_A, N_B) q_B \quad (2.4.11)$$

The main properties of \mathbf{K} are established in the following theorem.

Theorem

1. \mathbf{K} is symmetric.
2. \mathbf{K} is positive definite.

Proof

1. The symmetry of \mathbf{K} follows directly from the symmetry of $a(\cdot, \cdot)$, viz.,

$$\begin{aligned} K_{PQ} &= a(N_A, N_B) && (\text{by definition}) \\ &= a(N_B, N_A) && (\text{symmetry of } a(\cdot, \cdot)) \\ &= K_{QP} && (\text{by definition}) \end{aligned}$$

2. (Recall that we must show (i) $c^T \mathbf{K} c \geq 0$ and (ii) $c^T \mathbf{K} c = 0$ implies $c = \mathbf{0}$.)

To each n_{eq} -vector $c = \{c_P\}$, we may associate a member $w^h \in \mathcal{V}^h$ by the expression $w^h = \sum_{A \in \eta - \eta_q} N_A \bar{c}_A$, where $\bar{c}_A = c_P$, $P = \text{ID}(A)$.

i.

$$\begin{aligned} c^T \mathbf{K} c &= \sum_{P, Q=1}^{n_{eq}} c_P K_{PQ} c_Q \\ &= \sum_{A, B \in \eta - \eta_q} \bar{c}_A a(N_A, N_B) \bar{c}_B \\ &= a\left(\sum_{A \in \eta - \eta_q} N_A \bar{c}_A, \sum_{B \in \eta - \eta_q} N_B \bar{c}_B\right) && (\text{bilinearity of } a(\cdot, \cdot)) \\ &= a(w^h, w^h) && (\text{definition of } w^h) \\ &= \underbrace{\int_{\Omega} w_{,i}^h K_{ij} w_{,j}^h d\Omega}_{\geq 0} && (\text{positive-definiteness of conductivities}) \\ &\geq 0 \end{aligned}$$

ii. Assume $\mathbf{c}^T \mathbf{K} \mathbf{c} = 0$. By the proof of part (i),

$$\int_{\Omega} \underbrace{w_{,i}^h \kappa_{ij} w_{,j}^h}_{\geq 0} d\Omega = 0$$

and thus it follows that

$$w_{,i}^h \kappa_{ij} w_{,j}^h = 0$$

By the positive-definiteness hypothesis on the conductivities, this requires $w_{,i}^h = 0$ and so w^h must be constant. However $w^h = 0$ on Γ_e (which is not empty) and so w^h must be zero throughout Ω . By the definition of w^h , it follows that each $c_p = 0$; that is $\mathbf{c} = \mathbf{0}$, which was to be proved. ■

Remarks

1. Observe that it is the positive-definiteness hypothesis on the constitutive coefficients (i.e., κ_{ij} 's) *and* the boundary condition incorporated in the definition of \mathcal{V}^h which together result in the positive-definiteness of \mathbf{K} and thus ensure its invertibility.

2. The explicit structure of the shape functions, which will be delineated in Chapter 3, will also result in \mathbf{K} being banded.

Exercise 1. (This exercise is a multidimensional analog of the one contained in Sec. 1.8.) Let

$$\Gamma_{\text{int}} = \left(\bigcup_{e=1}^{n_{el}} \Gamma^e \right) - \Gamma \quad (\text{interior element boundaries})$$

One side of Γ_{int} is (arbitrarily) designated to be the “+ side” and the other is the “- side.” Let \mathbf{n}^+ and \mathbf{n}^- be unit normals to Γ_{int} which point in the plus and minus directions, respectively. Clearly $\mathbf{n}^+ = -\mathbf{n}^-$. Let q_i^+ and q_i^- denote the values of q_i obtained by approaching $\mathbf{x} \in \Gamma_{\text{int}}$ from + and - sides, respectively. The “jump” in $q_n = q_i n_i$ at \mathbf{x} is defined to be

$$\begin{aligned} [q_n] &= (q_i^+ - q_i^-) n_i^+ \\ &= q_i^+ n_i^+ + q_i^- n_i^- \end{aligned}$$

As may be easily verified, the jump is invariant with respect to reversing the + and - designations.

Consider the weak formulation (i.e., (2.3.6)) and assume w and u are smooth on the element interiors but may experience discontinuities in gradient across element boundaries. (Functions of this type contain the standard C^0 finite element interpolations; see Chapter 3.) Show that

$$0 = \sum_{e=1}^{n_{el}} \int_{\Omega^e} w(q_{i,i} - t) d\Omega - \int_{\Gamma_k} w(q_n + h) d\Gamma + \int_{\Gamma_{\text{int}}} w[q_n] d\Gamma$$

from which the Euler-Lagrange conditions may be readily deduced:

$$\text{i. } q_{i,i} = f \text{ in } \bigcup_{e=1}^{n_{el}} \Omega^e$$

$$\text{ii. } -q_n = h \text{ on } \Gamma_k$$

$$\text{iii. } [q_n] = 0 \text{ on } \Gamma_{\text{int}}$$

As may be seen, (i) is the heat equation on the *element interiors* and (iii) is a continuity condition across element boundaries on the heat flux. Contrast the present results with those obtained assuming w and u are *globally* smooth.

The Galerkin finite element formulation obtains an *approximate* solution to (i) through (iii).

2.5 HEAT CONDUCTION: ELEMENT STIFFNESS MATRIX AND FORCE VECTOR

As before, we can break up the global arrays into sums of elemental contributions:

$$\mathbf{K} = \sum_{e=1}^{n_{el}} \mathbf{K}^e, \quad \mathbf{K}^e = [\mathbf{K}_{PQ}^e] \quad (2.5.1)$$

$$\mathbf{F} = \sum_{e=1}^{n_{el}} \mathbf{F}^e, \quad \mathbf{F}^e = \{\mathbf{F}_P^e\} \quad (2.5.2)$$

where

$$\mathbf{K}_{PQ}^e = a(N_A, N_B)^e = \int_{\Omega^e} (\nabla N_A)^T \kappa (\nabla N_B) d\Omega \quad (2.5.3)$$

$$\begin{aligned} \mathbf{F}_P^e &= (N_A, f)^e + (N_A, h)_f - \sum_{B \in \eta_q} a(N_A, N_B)^e q_B \\ &= \int_{\Omega^e} N_A f d\Omega + \int_{\Gamma_k^e} N_A h d\Gamma - \sum_{B \in \eta_q} a(N_A, N_B)^e q_B \end{aligned} \quad (2.5.4)$$

$$\Gamma_k^e = \Gamma_k \cap \Gamma^e, \quad P = \text{ID}(A), \quad Q = \text{ID}(B) \quad (2.5.5)$$

See Fig. 2.5.1 for an illustration of Γ_k^e .

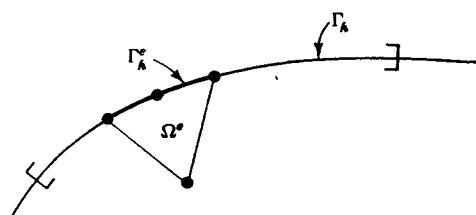


Figure 2.5.1

The element stiffness, k^e , and element force vector, f^e , may be deduced from these equations:

$$k^e = [k_{ab}^e], \quad f^e = \{f_a^e\}, \quad 1 \leq a, b \leq n_{en} \quad (2.5.6)$$

$$k_{ab}^e = a(N_a, N_b)^e = \int_{\Omega^e} (\nabla N_a)^T \kappa (\nabla N_b) d\Omega \quad (2.5.7)$$

$$f_a^e = \int_{\Omega^e} N_a q d\Omega + \int_{\Gamma_e^i} N_a q d\Gamma - \sum_{b=1}^{n_{en}} k_{ab}^e q_b^e \quad (2.5.8)$$

where (recall) n_{en} is the number of element nodes, and $q_b^e = q(x_b^e)$ if q is prescribed at node number b and equals zero otherwise.⁶

The global arrays, K and F may be formed from the element arrays k^e and f^e , respectively, by way of an assembly algorithm as described in Sec. 1.14.

The element stiffness matrix can be written in a standard form convenient for programming:

$$k^e = \int_{\Omega^e} B^T D B d\Omega \quad (2.5.9)$$

where, in the present case,

$$\underbrace{D}_{n_{sd} \times n_{sd}} = \kappa \quad (2.5.10)$$

$$\underbrace{B}_{n_{sd} \times n_{en}} = [B_1, B_2, \dots, B_{n_{en}}] \quad (2.5.11)$$

$$\underbrace{B_a}_{n_{sd} \times 1} = \nabla N_a \quad (2.5.12)$$

The component version of (2.5.9) is

$$k_{ab}^e = \int_{\Omega^e} B_a^T D B_b d\Omega \quad (2.5.13)$$

⁶ An implicit assumption in localizing the q -term is that if x_A is not a node attached to element e , then $N_A(x) = 0$ for all $x \in \bar{\Omega}^e$. Otherwise, the last term in (2.5.4) may involve q -data of nodes not attached to element e , which is not accounted for in (2.5.8).

Exercise 1. Let

$$\underbrace{\mathbf{d}^e}_{n_{en} \times 1} = \{d_a^e\} = \begin{Bmatrix} d_1^e \\ d_2^e \\ \vdots \\ d_{n_{en}}^e \end{Bmatrix} \quad (2.5.14)$$

where

$$d_a^e = u^h(\mathbf{x}_a^e) \quad (2.5.15)^7$$

\mathbf{d}^e is called the *element temperature vector*. Show that the heat flux vector at point $x \in \Omega^e$ can be calculated from the formula

$$\mathbf{q}(x) = -D(x)\mathbf{B}(x)\mathbf{d}^e = -D(x) \sum_{a=1}^{n_{en}} B_a d_a^e \quad (2.5.16)$$

Exercise 2. Consider the strong statement of the boundary-value problem in classical linear heat conduction in which the h -type boundary condition (i.e., eq. (2.3.4)) is replaced by the following expression:

$$\lambda u - q_i n_i = h \quad \text{on } \Gamma_i \quad (2.5.17)$$

where $\lambda \geq 0$ is a given function of $x \in \Gamma_i$. Generalize the weak formulation to include (2.5.17) as a natural boundary condition. Obtain an expression for the additional contribution to k_{ab}^e (cf. (2.5.13)) arising from (2.5.17). Show that K is positive-definite.

The boundary condition (2.5.17) is equivalent to what is often called *Newton's law of heat transfer*; λ is called the *coefficient of heat transfer*. This boundary condition applies to the case in which the heat flux is proportional to the difference of the surface temperatures of the body and surrounding medium, the latter formally represented by h/λ in (2.5.17).

2.6 HEAT CONDUCTION: DATA PROCESSING ARRAYS; ID, IEN, AND LM

The element nodal data is stored in the array IEN, the *element nodes array*, which relates local node numbers to global node numbers, viz.,

| | | | |
|-------------------------|-------------------|--------------------------|-----------|
| $IEN(a, e)$ | $=$ | \underbrace{A} | $(2.6.1)$ |
| Local node number | Element number | Global node number | |

The relationship between global node numbers and global equation numbers as well as nodal boundary condition information is stored in the ID array (see 2.4.7). In

⁷The g -boundary conditions are accounted for in this definition.

practice, the IEN and ID arrays are set up from input data. The LM array, which was described in the context of the one-dimensional model problem in Sec. 1.14, may then be constructed from the relation

$$\text{LM}(a, e) = \text{ID}(\text{IEN}(a, e)) \quad (2.6.2)$$

Because of the previous relationship, we often think of LM as the element “localization” of ID. Strictly speaking the LM array is redundant. However, it is generally convenient in computing to set up LM once and for all, rather than make use of (2.6.2) repeatedly.

Example 1 illustrates the structure of the ID, IEN, and LM arrays.

Example 1

Consider the mesh of four-node, rectangular elements shown in Fig. 2.6.1. We assume that the local node numbering begins at the lower left-hand node of each element and proceeds in counterclockwise fashion. This is illustrated in Fig. 2.6.1 for element 2, which is typical. We also assume that essential boundary conditions (i.e., “ φ -type”) are specified at nodes 1, 4, 7, and 10. Thus there will only be eight equations in the global system $Kd = F$. We adopt the usual convention that the global equation numbers run in ascending order with respect to the ascending order of global node numbers. The ID, IEN, and LM arrays are given in Fig. 2.6.2. The reader is urged to verify the details.

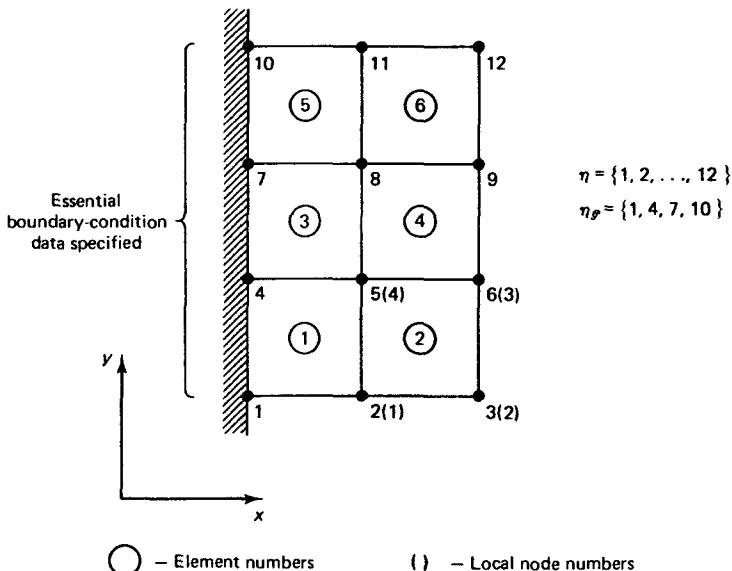


Figure 2.6.1 Mesh of four-node, rectangular elements; global and local node numbers, element numbers, and essential boundary condition nodes.

Sec. 2.6 Heat Conduction: Data Processing Arrays

ID array:

 Global node numbers (A)

| | | | | | | | | | | | | $(n_{np} = 12)$ |
|----|---|---|---|---|---|---|---|---|----|----|----|-----------------|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | |
| 0* | 1 | 2 | 0 | 3 | 4 | 0 | 5 | 6 | 0 | 7 | 8 | |

$$(n_{eq} = 8)$$

$$P = \text{ID}(A)$$

IEN array:

 Element numbers (e)

| | | | | | | | | | | | | $(n_{el} = 6)$ |
|-------------------------------------|---|---|---|---|---|----|----|--|--|--|--|----------------|
| | 1 | 2 | 3 | 4 | 5 | 6 | | | | | | |
| Local node numbers (a) | 1 | 1 | 2 | 4 | 5 | 7 | 8 | | | | | |
| | 2 | 2 | 3 | 5 | 6 | 8 | 9 | | | | | |
| | 3 | 5 | 6 | 8 | 9 | 11 | 12 | | | | | |
| | 4 | 4 | 5 | 7 | 8 | 10 | 11 | | | | | |

$$(n_{en} = 4)$$

$$A = \text{IEN}(a, e)$$

LM array:

 Element numbers (e)

| | | | | | | | | | | | | $(n_{el} = 6)$ |
|-------------------------------------|---|----|---|---|---|---|---|--|--|--|--|----------------|
| | 1 | 2 | 3 | 4 | 5 | 6 | | | | | | |
| Local node numbers (a) | 1 | 0* | 1 | 0 | 3 | 0 | 5 | | | | | |
| | 2 | 1 | 2 | 3 | 4 | 5 | 6 | | | | | |
| | 3 | 3 | 4 | 5 | 6 | 7 | 8 | | | | | |
| | 4 | 0 | 3 | 0 | 5 | 0 | 7 | | | | | |

$$(n_{en} = 4)$$

$$P = \text{LM}(a, e) = \text{ID}(\text{IEN}(a, e))$$

* Temperature boundary conditions ("q-type") denoted by zeros.

Figure 2.6.2. ID, IEN, and LM arrays for the mesh of Fig. 2.6.1.

In terms of the IEN and LM arrays, a precise definition of the q_a^e 's may be given (see (2.5.8)):

$$q_a^e = \begin{cases} 0 & \text{if } \text{LM}(a, e) \neq 0 \\ q_A & \text{if } \text{LM}(a, e) = 0, \text{ where } A = \text{IEN}(a, e) \end{cases} \quad (2.6.3)$$

This definition may be easily programmed.

In our final example of this section we shall illustrate the assembly procedure for a typical element subjected to essential boundary conditions.

Example 2

Consider a typical, four-noded element e . Assume values of the LM array, for this element, are given as follows:

$$\left. \begin{array}{l} \text{LM}(1, e) = 5 \\ \text{LM}(2, e) = 0 \\ \text{LM}(3, e) = 0 \\ \text{LM}(4, e) = 9 \end{array} \right\} \quad (2.6.4)$$

We deduce from (2.6.4) that the contributions to the global arrays are given as follows:⁸

$$\left. \begin{array}{l} K_{55} \leftarrow K_{55} + k_{11} \\ K_{59} \leftarrow K_{59} + k_{14} \\ K_{95} \leftarrow K_{95} + k_{41} \\ K_{99} \leftarrow K_{99} + k_{44} \end{array} \right\} \quad (2.6.5)$$

$$\left. \begin{array}{l} F_5 \leftarrow F_5 + f_1^e \\ F_9 \leftarrow F_9 + f_4^e \end{array} \right\} \quad (2.6.6)$$

Note that all terms in the second and third rows and columns of k^e do *not* contribute to K . However, they may contribute to F via f_1^e and f_4^e , since, by (2.5.8) we have

$$f_1^e = \dots - k_{12}^e q_2^e - k_{13}^e q_3^e \quad (2.6.7)$$

$$f_4^e = \dots - k_{42}^e q_2^e - k_{43}^e q_3^e \quad (2.6.8)$$

in which, for clarity, we have omitted the first two terms of the right-hand side of (2.5.8).

Special subroutines are easily programmed to carry out the operations indicated in (2.6.5)–(2.6.8).

It is instructive to visualize the contributions of the e th element to the global stiffness and force. These contributions are depicted in Fig. 2.6.3.

We note that all necessary element assembly information is provided by the LM array.

⁸ Due to symmetry, k_{11}^e is not actually assembled in practice.

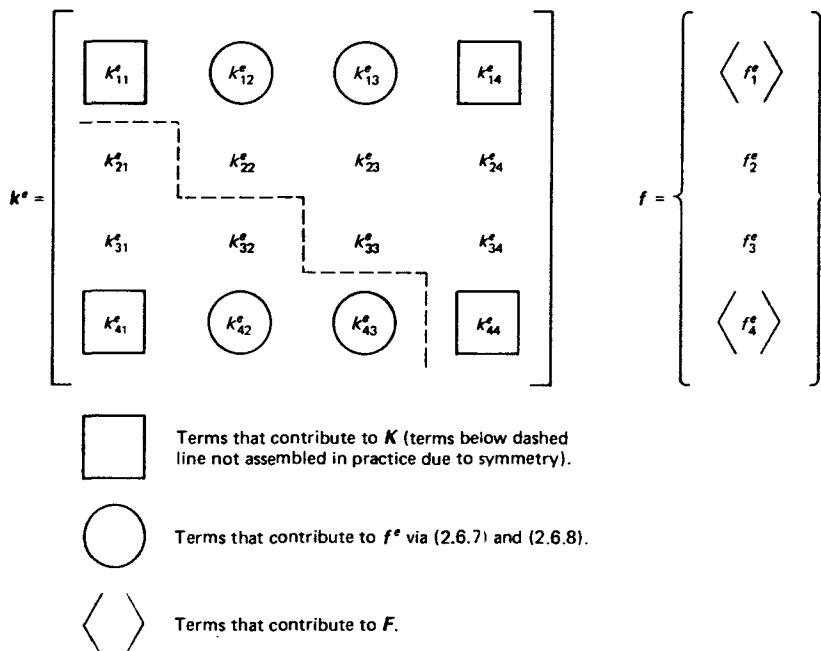
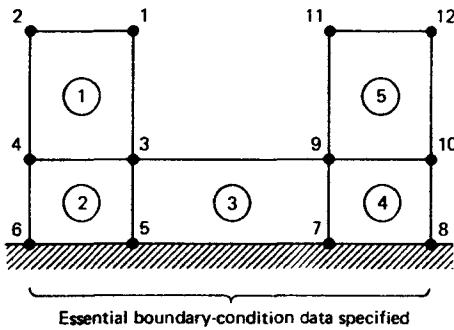


Figure 2.6.3 Contributions of heat conduction element in Example 2 to global arrays.

Exercise 1. Consider the accompanying mesh. Set up the ID, IEN, and LM arrays.



2.7 CLASSICAL LINEAR ELASTOSTATICS: STRONG AND WEAK FORMS; EQUIVALENCE

Classical elastostatics is a rich subject in its own right. See [3–7] for background and references to the literature. These references range from the very physical to the very

mathematical. The most physical book is the one by Timoshenko and Goodier. In ascending order of mathematical content are Sokolnikoff, Gurtin, Duvaut-Lions, and Fichera.

The reader is reminded that indices i, j, k , and l take on values $1, \dots, n_{sd}$, where n_{sd} is the number of spatial dimensions, and the summation convention applies to repeated indices i, j, k , and l only.

Let σ_{ij} denote (Cartesian components of) the (Cauchy) *stress tensor*, let u_i denote the *displacement vector*, and let f_i be the *prescribed body force per unit volume*. The (*infinitesimal*) *strain tensor*, ϵ_{ij} , is defined to be the symmetric part of the displacement gradients, viz.,

$$\epsilon_{ij} = u_{(i,j)} \stackrel{\text{def.}}{=} \frac{u_{i,j} + u_{j,i}}{2} \quad (\text{strain-displacement equations}) \quad (2.7.1)$$

The stress tensor is defined in terms of the strain tensor by the *generalized Hooke's law*:⁹

$$\sigma_{ij} = c_{ijkl} \epsilon_{kl} \quad (2.7.2)$$

where the c_{ijkl} 's, the *elastic coefficients*, are given functions of x . (If the c_{ijkl} 's are constants throughout, the body is called *homogeneous*.) The elastic coefficients are assumed to satisfy the following properties:

Symmetry

$$c_{ijkl} = c_{klji} \quad (\text{major symmetry}) \quad (2.7.3)$$

$$\left. \begin{aligned} c_{ijkl} &= c_{jikl} \\ c_{ijkl} &= c_{ijlk} \end{aligned} \right\} \quad (\text{minor symmetries}) \quad (2.7.4)$$

Positive-definiteness

$$c_{ijkl}(x) \psi_{ij} \psi_{kl} \geq 0 \quad (2.7.5)$$

$$c_{ijkl}(x) \psi_{ij} \psi_{kl} = 0 \quad \text{implies} \quad \psi_{ij} = 0 \quad (2.7.6)$$

for all $x \in \bar{\Omega}$ and all $\psi_{ij} = \psi_{ji}$.

Note. The positive-definiteness condition is in terms of symmetric arrays, ψ_{ij} .

We shall see in Sec. 2.8 that a consequence of the major symmetry (2.7.3) is that K is symmetric. The first minor symmetry implies the symmetry of the stress tensor (i.e., $\sigma_{ij} = \sigma_{ji}$).¹⁰ The positive-definiteness condition, when combined with appropriate boundary conditions on the displacement, leads to the positive-definiteness of K .

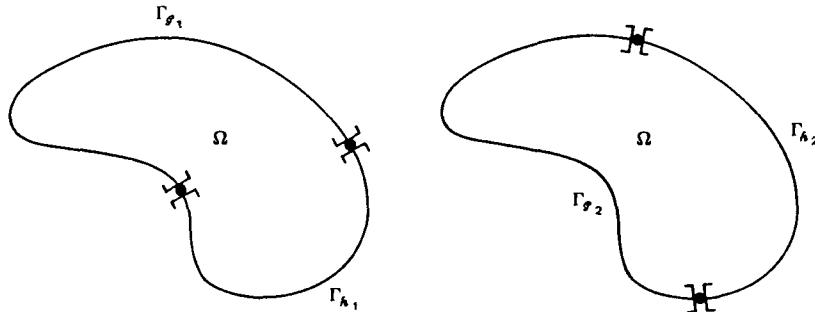
⁹This is another *constitutive equation*, which reflects the elastic properties of the body under consideration.

¹⁰From a fundamental continuum mechanics standpoint, the symmetry of the Cauchy stress tensor emanates from the balance of angular momentum.

In the present theory, the unknown is a vector (i.e., the displacement vector). Consequently, a generalization of the boundary conditions considered previously is necessitated. We shall assume that Γ admits decompositions

$$\left. \begin{aligned} \Gamma &= \overline{\Gamma_{g_i} \cup \Gamma_{h_i}} \\ \emptyset &= \Gamma_{g_i} \cap \Gamma_{h_i} \end{aligned} \right\} \quad i = 1, \dots, n_{sd} \quad (2.7.7)$$

For example, in two dimensions the situation might appear as in Fig. 2.7.1. As can be seen there can be a different decomposition for each $i = 1, 2, \dots, n_{sd}$.



$$\left. \begin{aligned} \Gamma &= \overline{\Gamma_{g_1} \cup \Gamma_{h_1}} \\ \emptyset &= \Gamma_{g_1} \cap \Gamma_{h_1} \end{aligned} \right.$$

$$\left. \begin{aligned} \Gamma &= \overline{\Gamma_{g_2} \cup \Gamma_{h_2}} \\ \emptyset &= \Gamma_{g_2} \cap \Gamma_{h_2} \end{aligned} \right.$$

Figure 2.7.1

A formal statement of the strong form of the boundary-value problem goes as follows:

$$(S) \left\{ \begin{array}{l} \text{Given } f_i : \Omega \rightarrow \mathbb{R}, g_i : \Gamma_{g_i} \rightarrow \mathbb{R}, \text{ and } h_i : \Gamma_{h_i} \rightarrow \mathbb{R}, \text{ find } u_i : \bar{\Omega} \rightarrow \mathbb{R} \text{ such that} \\ \sigma_{ij,j} + f_i = 0 \quad \text{in } \Omega \quad (\text{equilibrium equations}) \quad (2.7.8) \\ u_i = g_i \quad \text{on } \Gamma_{g_i} \quad (2.7.9) \\ \sigma_{ij}n_j = h_i \quad \text{on } \Gamma_{h_i} \quad (2.7.10) \end{array} \right.$$

where σ_{ij} is defined in terms of u_i by (2.7.1) and (2.7.2).

Remarks

1. The functions g_i and h_i are called the *prescribed boundary displacements* and *tractions*, respectively.
2. (S) is sometimes referred to as the *mixed boundary-value problem of linear elastostatics*. Under appropriate hypotheses on the data, (S) possesses a unique solution (see Fichera [4]).
3. The additional complexity of the present theory, when compared with heat conduction, is that the unknown (i.e., $\mathbf{u} = \{u_i\}$) is vector-valued rather than scalar-valued.

4. In practice, it is important to be able to deal with somewhat more complicated boundary-condition specification. In order not to encumber the present exposition, this generalization will be considered later on in the form of an exercise (see Exercise 5 in Sec. 2.12).

Let \mathcal{S}_i denote the trial solution space and \mathcal{V}_i the variation space. Each member $u_i \in \mathcal{S}_i$ satisfies $u_i = q_i$ on Γ_{q_i} , whereas each $w_i \in \mathcal{V}_i$ satisfies $w_i = 0$ on Γ_{q_i} . Equation (2.7.10) will be a natural boundary condition.

The weak formulation goes as follows:

$$(W) \left\{ \begin{array}{l} \text{Given } f_i : \Omega \rightarrow \mathbb{R}, q_i : \Gamma_{q_i} \rightarrow \mathbb{R} \text{ and } h_i : \Gamma_{h_i} \rightarrow \mathbb{R}, \text{ find } u_i \in \mathcal{S}_i \text{ such that for all } w_i \in \mathcal{V}_i, \\ \boxed{\int_{\Omega} w_{(i,j)} \sigma_{ij} d\Omega = \int_{\Omega} w_i f_i d\Omega + \sum_{i=1}^{n_{sd}} \left(\int_{\Gamma_{h_i}} w_i h_i d\Gamma \right)} \end{array} \right. \quad (2.7.11)$$

where σ_{ij} is defined in terms of u_i by (2.7.1) and (2.7.2).

Remarks

5. In the solid mechanics literature, (W) is sometimes referred to as the *principle of virtual work*, or *principle of virtual displacements*, w_i being the *virtual displacements*.

6. The existence and uniqueness of weak solutions is discussed in Duvaut-Lions [3].

Note. The boundary integral in (2.7.11) takes on the explicit form:

$$\sum_{i=1}^{n_{sd}} \left(\int_{\Gamma_{h_i}} w_i h_i d\Gamma \right) = \int_{\Gamma_{h_1}} w_1 h_1 d\Gamma + \cdots + \int_{\Gamma_{h_{n_{sd}}}} w_{n_{sd}} h_{n_{sd}} d\Gamma \quad (2.7.12)$$

Theorem. Assume all functions are smooth enough to justify the manipulations. Then a solution of (S) is a solution of (W), and vice versa.

Remark

7. The proof of the equivalence theorem requires some preliminary results, which we shall establish in the following lemmas.

Lemma 1. (*Euclidean decomposition of a second-rank tensor.*) Let s_{ij} denote a general nonsymmetric, second-rank tensor. Then $s_{ij} = s_{(ij)} + s_{[ij]}$, where $s_{(ij)}$ is symmetric (i.e., $s_{(ij)} = s_{(ji)}$) and $s_{[ij]}$ is skew-symmetric (i.e., $s_{[ij]} = -s_{[ji]}$).

Proof. Define

$$s_{(ij)} = \frac{s_{ij} + s_{ji}}{2} \quad (2.7.13)$$

$$s_{[ij]} = \frac{s_{ij} - s_{ji}}{2} \quad (2.7.14)$$

It is easily verified that $s_{(ij)}$ is symmetric and $s_{[ij]}$ is skew-symmetric. ■

Remark

8. $s_{(ij)}$ and $s_{[ij]}$ are called the *symmetric* and *skew-symmetric parts* of s_{ij} , respectively.

Lemma 2. Let s_{ij} be a nonsymmetric tensor and let t_{ij} be a symmetric tensor. Then

$$s_{ij}t_{ij} = s_{(ij)}t_{ij} \quad (2.7.15)$$

Proof. By Lemma 1, $s_{ij} = s_{(ij)} + s_{[ij]}$. Since

$$s_{ij}t_{ij} = s_{(ij)}t_{ij} + s_{[ij]}t_{ij}$$

the lemma will be established if we can show that $s_{[ij]}t_{ij} = 0$. We proceed as follows:

$$\begin{aligned} s_{[ij]}t_{ij} &= -s_{[ji]}t_{ij} && \text{(skew-symmetry of } s_{[ij]} \text{)} \\ &= -s_{[ji]}t_{ji} && \text{(symmetry of } t_{ij} \text{)} \\ &= -s_{[ij]}t_{ij} && \text{(redefinition of dummy indices)} \end{aligned}$$

from which the result follows. ■

Proof of Theorem

1. Let u_i be a solution of (S) . Thus $u_i \in \mathcal{S}_i$. Multiply (2.7.8) by $w_i \in \mathcal{V}_i$ and integrate over Ω , viz.,

$$0 = \int_{\Omega} w_i(\sigma_{ij,j} + \ell_i) d\Omega = - \int_{\Omega} w_{i,j}\sigma_{ij} d\Omega + \int_{\Gamma} w_i\sigma_{ij}n_j d\Gamma + \int_{\Omega} w_i\ell_i d\Omega \quad (\text{integration by parts})$$

$$= - \int_{\Omega} w_{(i,j)}\sigma_{ij} d\Omega + \sum_{i=1}^{n_d} \left(\int_{\Gamma_{k_i}} w_i h_i d\Gamma \right) + \int_{\Omega} w_i \ell_i d\Omega \quad (\text{symmetry of } \sigma_{ij}, \text{ Lemma 2, } w_i = 0 \text{ on } \Gamma_{q_i}, \text{ and (2.7.10)})$$

Therefore, u_i is a solution of (W) .

2. Assume u_i is a solution of (W) . Since $u_i \in \mathcal{S}_i$, $u_i = q_i$ on Γ_{q_i} . From (2.7.11)

$$0 = - \int_{\Omega} \underbrace{w_{(i,j)}\sigma_{ij}}_{(= w_{i,j}\sigma_{ij} \text{ by Lemma 2})} d\Omega + \int_{\Omega} w_i \ell_i d\Omega + \sum_{i=1}^{n_d} \int_{\Gamma_{k_i}} w_i h_i d\Gamma$$

$$= \int_{\Omega} w_i (\sigma_{ij,j} + f_i) d\Omega - \sum_{i=1}^{n_d} \int_{\Gamma_{k_i}} w_i (\sigma_{ij} n_j - h_i) d\Gamma \quad (\text{integration by parts, } w_i = 0 \text{ on } \Gamma_{q_i}) \quad (2.7.16)$$

Let

$$\alpha_i = \sigma_{ij,j} + f_i$$

$$\beta_i = \sigma_{ij} n_j - h_i$$

Thus to complete the proof we must show that

$$\alpha_i = 0 \quad \text{on } \Omega$$

$$\beta_i = 0 \quad \text{on } \Gamma_{k_i}$$

These conditions can be proved by the techniques used in Sec. 2.3. Let $w_i = \alpha_i \phi$, where

- i. $\phi > 0$ on Ω ;
- ii. $\phi = 0$ on Γ ; and
- iii. ϕ is smooth.

(These conditions insure that $w_i \in \mathcal{V}_i$.) Substituting this w_i into (2.7.16) yields

$$0 = \int_{\Omega} \underbrace{\alpha_i \alpha_i}_{\geq 0} \underbrace{\phi}_{> 0} d\Omega$$

which implies $\alpha_i = 0$ on Ω .

Now take $w_i = \delta_{i1} \beta_1 \psi$, where

- i'. $\psi > 0$ on Γ_{k_1} ;
- ii'. $\psi = 0$ on Γ_{q_1} ; and
- iii'. ψ is smooth.

(Again, $w_i \in \mathcal{V}_i$.) Substituting this w_i into (2.7.16) and making use of $\alpha_i = 0$ results in

$$0 = \int_{\Gamma_{k_1}} \beta_1^2 \psi d\Gamma$$

from which it follows that $\beta_1 = 0$ on Γ_{k_1} .

We may proceed analogously to show $\beta_2 = 0$, and so on. Consequently, u_i is a solution of (S). ■

The abstract notation for the present case is

$$a(w, u) = \int_{\Omega} w_{(i,j)} c_{ijkl} u_{(k,l)} d\Omega \quad (2.7.17)$$

$$(w, \ell) = \int_{\Omega} w_i \ell_i d\Omega \quad (2.7.18)$$

$$(w, h)_\Gamma = \sum_{i=1}^{n_{sd}} \left(\int_{\Gamma_{k_i}} w_i h_i d\Gamma \right) \quad (2.7.19)$$

Exercise 1. Verify that $a(\cdot, \cdot)$, (\cdot, \cdot) and $(\cdot, \cdot)_\Gamma$, as just defined, are symmetric, bilinear forms (cf. Sec. 1.4).

Let $\mathcal{S} = \{u \mid u_i \in \mathcal{S}_i\}$ and let $\mathcal{V} = \{w \mid w_i \in \mathcal{V}_i\}$. Then the weak form can be concisely written in terms of (2.7.17–2.7.19) as follows:

$$(W) \left\{ \begin{array}{l} \text{Given } \ell, g, \text{ and } h \text{ (in which the components are defined in } (W)) \text{, find } u \in \mathcal{S} \\ \text{such that for all } w \in \mathcal{V} \end{array} \right. \quad \boxed{a(w, u) = (w, \ell) + (w, h)_\Gamma} \quad (2.7.20)$$

As discussed in Sec. 2.3, it is desirable to construct index-free counterparts of the expressions on the right-hand sides of (2.7.17)–(2.7.19). For concreteness we shall assume that $n_{sd} = 2$; thus $1 \leq i, j, k, l \leq 2$.

Let¹¹

$$\epsilon(u) = \{\epsilon_I(u)\} = \begin{Bmatrix} u_{1,1} \\ u_{2,2} \\ u_{1,2} + u_{2,1} \end{Bmatrix} \quad (2.7.21)$$

$$\epsilon(w) = \{\epsilon_I(w)\} = \begin{Bmatrix} w_{1,1} \\ w_{2,2} \\ w_{1,2} + w_{2,1} \end{Bmatrix} \quad (2.7.22)$$

$$D = [D_{IJ}] = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ & D_{22} & D_{23} \\ & & \text{symmetric } D_{33} \end{bmatrix} \quad (2.7.23)$$

¹¹ According to our previous notational conventions, $\epsilon = [\epsilon_{ij}]$, the matrix of strain components. However, we will have no need for this matrix, and consequently we reserve ϵ for the “strain vector” defined in (2.7.21). A similar notational conflict occurs with respect to the “stress vector,” σ , defined in Exercise 4. Note that factors of one-half have been eliminated from the *shearing components* (i.e., last components) of (2.7.21) and (2.7.22). (Compare (2.7.21) and (2.7.22) with (2.7.1).) This will considerably simplify subsequent writing.

where

$$D_{IJ} = c_{ijkl} \quad (2.7.24)$$

in which the indices are related by the following table:

TABLE 2.7.1

| I | J | i | k | j | l |
|-----|-----|-----|-----|-----|-----|
| 1 | | 1 | | 1 | |
| 3 | | 1 | | 2 | |
| 3 | | 2 | | 1 | |
| 2 | | 2 | | 2 | |

As should be clear, we have “collapsed” pairs of indices (i, j , and k, l) into single indices (I and J , respectively) taking account of the symmetries of c_{ijkl} , $u_{(k,l)}$ and $w_{(i,j)}$. Observe that the indices I and J take on values 1, 2, and 3. In n_{sd} dimensions, the I and J indices will take on values 1, 2, . . . , $n_{sd}(n_{sd} + 1)/2$.

It can be shown that

$$w_{(i,j)}c_{ijkl}u_{(k,l)} = \epsilon(w)^T D \epsilon(u) \quad (2.7.25)$$

and so

$$a(w, u) = \int_{\Omega} \epsilon(w)^T D \epsilon(u) d\Omega \quad (2.7.26)$$

Exercise 2. Verify (2.7.25) for $n_{sd} = 2$.

Exercise 3. Construct the analog of Table 2.7.1 for the case $n_{sd} = 3$. For definiteness of the ordering, take

$$\epsilon(u) = \begin{Bmatrix} u_{1,1} \\ u_{2,2} \\ u_{3,3} \\ u_{2,3} + u_{3,2} \\ u_{1,3} + u_{3,1} \\ u_{1,2} + u_{2,1} \end{Bmatrix} \quad (2.7.27)$$

Exercise 4. Show that

$$\sigma = D \epsilon(u) \quad (2.7.28)$$

where

$$\boldsymbol{\sigma} = \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{Bmatrix}, \quad n_{sd} = 2 \quad (2.7.29)$$

$$\boldsymbol{\sigma} = \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{Bmatrix}, \quad n_{sd} = 3 \quad (2.7.30)$$

Exercise 5. If the body in question is *isotropic*, then

$$c_{ijkl}(x) = \mu(x)(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) + \lambda(x)\delta_{ij}\delta_{kl} \quad (2.7.31)$$

where λ and μ are the *Lamé parameters*; μ is often referred to as the shear modulus and denoted by G . The relationships of λ and μ to E , *Young's modulus*, and ν , *Poisson's ratio*, are given by

$$\lambda = \frac{\nu E}{(1 + \nu)(1 - 2\nu)} \quad (2.7.32)$$

$$\mu = \frac{E}{2(1 + \nu)} \quad (2.7.33)$$

(See Sokolnikoff [6], p. 71, for further relations with other equivalent moduli.) If (2.7.31) is not satisfied, the body is said to be *anisotropic*. Using (2.7.31) set up D for $n_{sd} = 2$ and $n_{sd} = 3$. Hint: The answer for $n_{sd} = 2$ is

$$D = \begin{bmatrix} \lambda + 2\mu & \lambda & 0 \\ \lambda & \lambda + 2\mu & 0 \\ \text{Symmetric} & & \mu \end{bmatrix} \quad (2.7.34)$$

This matrix manifests the *plane strain* hypothesis. (See Sokolnikoff [6] for elaboration.)

Remark

9. The case of isotropic *plane stress* may be determined from (2.7.34) by replacing λ by $\bar{\lambda}$, where

$$\bar{\lambda} = \frac{2\lambda\mu}{\lambda + 2\mu} \quad (2.7.35)$$

(See Sokolnikoff [6] or Timoshenko and Goodier [7] for elaboration on the physical ideas.)

Exercise 6. (See Exercise 1, Sec. 2.4 for background and an analogous result.)

Let the “jump” in $\sigma_{in} = \sigma_{ij}n_j$ be denoted by $[\sigma_{in}]$. Consider the weak formulation (i.e., (2.7.11)) and assume w_i and u_i are smooth on element interiors, but experience gradient discontinuities across element boundaries. Show that

$$0 = \sum_{e=1}^{n_{el}} \int_{\Omega^e} w_i(\sigma_{ij,j} + f_i) d\Omega - \sum_{i=1}^{n_{sd}} \int_{\Gamma_{k_i}} w_i(\sigma_{in} - h_i) d\Gamma - \int_{\Gamma_{int}} w_i [\sigma_{in}] d\Gamma$$

from which the Euler-Lagrange conditions may be read:

- i. $\sigma_{ij,j} + f_i = 0$ in $\bigcup_{e=1}^{n_{el}} \Omega^e$
- ii. $\sigma_{in} = h_i$ on Γ_{k_i}
- iii. $[\sigma_{in}] = 0$ on Γ_{int}

Here (i) is the equilibrium equation on *element interiors*, and (iii) is a traction continuity condition across element boundaries. Compare these results with those obtained assuming w_i and u_i are *globally* smooth.

2.8 ELASTOSTATICS: GALERKIN FORMULATION, SYMMETRY, AND POSITIVE-DEFINITENESS OF K

Let \mathcal{S}^h and \mathcal{V}^h be finite-dimensional approximations to \mathcal{S} and \mathcal{V} , respectively. We assume members $w^h \in \mathcal{V}^h$ result in satisfaction, or approximate satisfaction, of the boundary condition $w_i = 0$ on Γ_{g_i} , and members of \mathcal{S}^h admit the decomposition

$$\mathbf{u}^h = \mathbf{v}^h + \mathbf{q}^h \quad (2.8.1)$$

where $\mathbf{v}^h \in \mathcal{V}^h$ and \mathbf{q}^h results in satisfaction, or approximate satisfaction, of the boundary condition $u_i = q_i$ on Γ_{g_i} .

The Galerkin formulation of our problem is given as follows:

$$(G) \left\{ \begin{array}{l} \text{Given } \mathbf{f}, \mathbf{g}, \text{ and } \mathbf{h} \text{ (as in (W)), find } \mathbf{u}^h = \mathbf{v}^h + \mathbf{q}^h \in \mathcal{S}^h \text{ such that for all } \mathbf{w}^h \in \mathcal{V}^h \\ \boxed{a(\mathbf{w}^h, \mathbf{v}^h) = (\mathbf{w}^h, \mathbf{f}) + (\mathbf{w}^h, \mathbf{h})_\Gamma - a(\mathbf{w}^h, \mathbf{q}^h)} \end{array} \right. \quad (2.8.2)$$

To define the global stiffness matrix and force vector for elasticity, it is necessary to introduce the ID array. This entails a generalization of the definition given in Sec. 2.6, since in the present case there will be more than 1 degree of freedom per node. For elasticity there are n_{sd} degrees of freedom per node, but in order to include in our

definition cases such as heat conduction, we shall take the fully general situation in which it is assumed that there are n_{dof} degrees of freedom per node.¹² In this case

$$\boxed{\begin{array}{c} \text{Global equation number} \\ \downarrow \\ \text{ID}(i, A) = \begin{cases} P & \text{if } A \in \eta - \eta_{q_i} \\ 0 & \text{if } A \in \eta_{q_i} \end{cases} \\ \begin{array}{l} \text{Degree of} \\ \text{freedom} \\ \text{number} \end{array} \quad \begin{array}{l} \text{Global node} \\ \text{number} \end{array} \end{array}} \quad (2.8.3)$$

where $1 \leq i \leq n_{\text{dof}}$. Thus ID has dimensions $n_{\text{dof}} \times n_{\text{np}}$. If $n_{\text{dof}} = 1$, we reduce to the case considered previously in Sec. 2.6 (i.e., $\text{ID}(i, A) = \text{ID}(A)$).

Recall that $\eta = \{1, 2, \dots, n_{\text{np}}\}$ denotes the set of global node numbers. Let $\eta_{q_i} \subset \eta$ be the set of nodes at which $u_i^h = q_i$ and let $\eta - \eta_{q_i}$ be the complement of η_{q_i} . For each node in $\eta - \eta_{q_i}$, the nodal value of u_i^h is to be determined.

The explicit representations of v_i^h and q_i^h , in terms of the shape functions and nodal values are

$$\boxed{\begin{array}{c} \text{Degree of freedom number} \\ v_i^h = \sum_{A \in \eta - \eta_{q_i}} N_A d_{iA} \quad (\text{no sum on } i) \\ \text{Global node number} \\ q_i^h = \sum_{A \in \eta_{q_i}} N_A q_{iA} \quad (\text{no sum on } i) \end{array}} \quad (2.8.4) \quad (2.8.5)$$

Let e_i denote the i th Euclidean basis vector for $\mathbb{R}^{n_{\text{sd}}}$; e_i has a 1 in slot i and zeros elsewhere. For example

$$(n_{\text{sd}} = 2) \quad e_1 = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}, \quad e_2 = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \quad (2.8.6)$$

¹²In general, this is taken to mean the maximum number of degrees of freedom per node in the global model. It is possible in practice to have elements with fewer degrees of freedom per node contributing to the model.

$$(n_{sd} = 3) \quad \mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad (2.8.7)$$

The vector versions of (2.8.4) and (2.8.5) may be defined with the aid of \mathbf{e}_i , viz.,

$$\mathbf{v}^h = v_i^h \mathbf{e}_i \quad (2.8.8)$$

$$\mathbf{q}^h = q_i^h \mathbf{e}_i \quad (2.8.9)$$

Likewise, a typical member $\mathbf{w}^h \in \mathcal{V}^h$ has the representation

$$\mathbf{w}^h = w_i^h \mathbf{e}_i, \quad w_i^h = \sum_{A \in \eta - \eta_{q_i}} N_A c_{iA} \quad (\text{no sum on } i) \quad (2.8.10)$$

Substituting (2.8.4), (2.8.5), and (2.8.8)–(2.8.10) into (2.8.2) and arguing along the lines of Sec. 1.6 results in (verify!)

$$\sum_{j=1}^{n_{dof}} \left(\sum_{B \in \eta - \eta_{q_j}} a(N_A \mathbf{e}_i, N_B \mathbf{e}_j) d_{jB} \right) = (N_A \mathbf{e}_i, \mathbf{f}) + (N_A \mathbf{e}_i, \mathbf{h})_\Gamma$$

$$- \sum_{j=1}^{n_{dof}} \left(\sum_{B \in \eta_{q_j}} a(N_A \mathbf{e}_i, N_B \mathbf{e}_j) q_{jB} \right), \quad A \in \eta - \eta_{q_i}, \quad 1 \leq i \leq n_{sd}$$

(2.8.11)¹³

This is equivalent to the matrix equation

¹³For correct interpretation of the meaning of these equations, the sum on j should be taken first. For example, in two dimensions

$$\sum_{j=1}^2 \left(\sum_{B \in \eta - \eta_{q_j}} a(N_A \mathbf{e}_i, N_B \mathbf{e}_j) d_{jB} \right) = \sum_{B \in \eta - \eta_{q_1}} a(N_A \mathbf{e}_i, N_B \mathbf{e}_1) d_{1B}$$

$$+ \sum_{B \in \eta - \eta_{q_2}} a(N_A \mathbf{e}_i, N_B \mathbf{e}_2) d_{2B}$$

and

$$\sum_{j=1}^2 \left(\sum_{B \in \eta_{q_j}} a(N_A \mathbf{e}_i, N_B \mathbf{e}_j) q_{jB} \right) = \sum_{B \in \eta_{q_1}} a(N_A \mathbf{e}_i, N_B \mathbf{e}_1) q_{1B}$$

$$+ \sum_{B \in \eta_{q_2}} a(N_A \mathbf{e}_i, N_B \mathbf{e}_2) q_{2B}$$

$$\mathbf{Kd} = \mathbf{F} \quad (2.8.12)$$

where

$$\mathbf{K} = [K_{PQ}] \quad (2.8.13)$$

$$\mathbf{d} = \{d_Q\} \quad (2.8.14)$$

$$\mathbf{F} = \{F_P\} \quad (2.8.15)$$

$$K_{PQ} = a(N_A e_i, N_B e_j) \quad (2.8.16)$$

$$F_P = (N_A e_i, \mathbf{f}) + (N_A e_i, \mathbf{k})_\Gamma - \sum_{j=1}^{n_{\text{dof}}} \left(\sum_{B \in \eta_{q_j}} a(N_A e_i, N_B e_j) q_{jB} \right) \quad (2.8.17)^{13}$$

in which

$$P = \text{ID}(i, A), \quad Q = \text{ID}(j, B) \quad (2.8.18)$$

Equation (2.8.16) may be written in more explicit form by using (2.7.26) and noting that (see (2.7.21) and (2.7.22)):

$$\epsilon(N_A e_i) = \mathbf{B}_A e_i \quad (2.8.19)$$

where

$$(n_{sd} = 2) \quad \mathbf{B}_A = \begin{bmatrix} N_{A,1} & 0 \\ 0 & N_{A,2} \\ N_{A,2} & N_{A,1} \end{bmatrix} \quad (2.8.20)$$

$$(n_{sd} = 3) \quad \mathbf{B}_A = \begin{bmatrix} N_{A,1} & 0 & 0 \\ 0 & N_{A,2} & 0 \\ 0 & 0 & N_{A,3} \\ 0 & N_{A,3} & N_{A,2} \\ N_{A,3} & 0 & N_{A,1} \\ N_{A,2} & N_{A,1} & 0 \end{bmatrix} \quad (2.8.21)$$

Exercise 1. Verify (2.8.19)–(2.8.21).

With these, (2.8.16) becomes

$$\underline{K_{PQ}} = \mathbf{e}_i^T \int_{\Omega} \mathbf{B}_A^T \mathbf{D} \mathbf{B}_B d\Omega \mathbf{e}_j$$

Global
equation
numbers Global node numbers Degree of freedom numbers

(2.8.22)

and the indices are related by (2.8.18).

Equation (2.8.17) is also amenable to explication. Note that, by (2.7.18)

$$(N_A \mathbf{e}_i, \boldsymbol{\epsilon}) = \int_{\Omega} N_A \epsilon_i \, d\Omega \quad (2.8.23)$$

and likewise by (2.7.19)

$$(N_A \mathbf{e}_i, \boldsymbol{\kappa})_{\Gamma} = \int_{\Gamma_{k_i}} N_A \kappa_i \, d\Gamma \quad (\text{no sum}) \quad (2.8.24)$$

Thus (2.8.17) may be written as

$$F_P = \int_{\Omega} N_A \epsilon_i \, d\Omega + \int_{\Gamma_{k_i}} N_A \kappa_i \, d\Gamma - \sum_{j=1}^{n_{\text{dof}}} \left(\sum_{B \in \eta_{q_j}} a(N_A \mathbf{e}_i, N_B \mathbf{e}_j) q_{jB} \right) \quad (2.8.25)^{14}$$

Now that we have defined \mathbf{K} , we can establish its fundamental properties. We shall need the following preliminary results.

Let $n_{sd} = 2$ or 3 and let $\mathbf{w} : \Omega \rightarrow \mathbb{R}^{n_{sd}}$. If $w_{(i,j)} = 0$ ("zero strains"), then \mathbf{w} admits the representations:

$$(n_{sd} = 2) \quad \mathbf{w}(\mathbf{x}) = \underbrace{\mathbf{c}}_{\text{Translation}} + \underbrace{\mathbf{c}_3(\mathbf{x}_1 \mathbf{e}_2 - \mathbf{x}_2 \mathbf{e}_1)}_{\text{Rotation}} \quad (2.8.26)$$

$$(n_{sd} = 3) \quad \mathbf{w}(\mathbf{x}) = \underbrace{\mathbf{c}_1}_{\text{Translation}} + \underbrace{\mathbf{c}_2 \times \mathbf{x}}_{\text{Rotation}} \quad (2.8.27)$$

where

$$\mathbf{c} = \begin{Bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{Bmatrix} \quad \mathbf{c}_1 = \begin{Bmatrix} c_{11} \\ c_{12} \\ c_{13} \end{Bmatrix} \quad \mathbf{c}_2 = \begin{Bmatrix} c_{21} \\ c_{22} \\ c_{23} \end{Bmatrix} \quad (2.8.28)$$

and c_3 are constants; and \times denotes the vector cross product. Equations (2.8.26) and (2.8.27) define *infinitesimal rigid-body motions*.

Assumption R

We assume that the homogeneous boundary conditions incorporated into the definition of \mathcal{V}^h preclude nontrivial infinitesimal rigid-body motions. In other words, we assume that if $\mathbf{w}^h \in \mathcal{V}^h$ is a rigid-body motion, then \mathbf{w}^h is identically zero.

Theorem

1. \mathbf{K} is symmetric.
2. If Assumption R holds, then \mathbf{K} is also positive definite.

Proof of 1. Symmetry We may note that symmetry of \mathbf{K} follows from (2.8.16) and the symmetry of $a(\cdot, \cdot)$. However, we shall provide an alternative proof in terms of (2.8.22).

¹⁴See footnote 13 of this chapter.

$$\begin{aligned}
K_{PQ} &= \mathbf{e}_i^T \int_{\Omega} \mathbf{B}_A^T \mathbf{D} \mathbf{B}_B d\Omega \mathbf{e}_j \\
&= \mathbf{e}_j^T \int_{\Omega} \mathbf{B}_B^T \mathbf{D}^T \mathbf{B}_A d\Omega \mathbf{e}_i \\
&= \mathbf{e}_j^T \int_{\Omega} \mathbf{B}_B^T \mathbf{D} \mathbf{B}_A d\Omega \mathbf{e}_i \quad (\text{symmetry of } \mathbf{D}) \\
&= K_{QP}
\end{aligned}$$

■

Remark

Note that the symmetry of \mathbf{K} followed from the symmetry of \mathbf{D} , which was a consequence of the major symmetry of the c_{ijkl} 's (see (2.7.3)).

Proof of 2. Positive definite (Recall from Sec. 1.9 that we must show (i) $\mathbf{c}^T \mathbf{K} \mathbf{c} \geq 0$ and (ii) $\mathbf{c}^T \mathbf{K} \mathbf{c} = 0$ implies $\mathbf{c} = \mathbf{0}$.)

Let $\mathbf{w}^h = \sum_{A \in \eta - \eta_{eq}} N_A c_{iA}$ be a member of \mathcal{V}_i^h . Then $c_P = c_{iA}$, where $P = \text{ID}(i, A)$, $1 \leq P \leq n_{eq}$, defines the components of an n_{eq} -vector \mathbf{c} .

i.

$$\begin{aligned}
\mathbf{c}^T \mathbf{K} \mathbf{c} &= \sum_{P, Q=1}^{n_{eq}} c_P K_{PQ} c_Q \\
&= \sum_{i, j=1}^{n_{\text{dof}}} \left(\sum_{\substack{A \in \eta - \eta_{eq} \\ B \in \eta - \eta_{eq}}} c_{iA} \mathbf{a}(N_A \mathbf{e}_i, N_B \mathbf{e}_j) c_{jB} \right) \quad (\text{definition of } K_{PQ}) \\
&= \mathbf{a} \left(\sum_{i=1}^{n_{\text{dof}}} \left(\sum_{A \in \eta - \eta_{eq}} c_{iA} N_A \mathbf{e}_i \right), \sum_{j=1}^{n_{\text{dof}}} \left(\sum_{B \in \eta - \eta_{eq}} c_{jB} N_B \mathbf{e}_j \right) \right) \quad (\text{bilinearity of } \mathbf{a}(\cdot, \cdot)) \\
&= \mathbf{a}(\mathbf{w}^h, \mathbf{w}^h) \quad (\text{definition of } \mathbf{w}^h) \\
&= \int_{\Omega} \underbrace{\mathbf{w}_{(i,j)}^h c_{ijkl} \mathbf{w}_{(k,l)}^h}_{\geq 0} d\Omega \quad (\text{by (2.7.5) and (2.7.17)}) \\
&\geq 0
\end{aligned}$$

ii.

Assume $\mathbf{c}^T \mathbf{K} \mathbf{c} = 0$. By the proof of part (i), we deduce that

$$\mathbf{w}_{(i,j)}^h c_{ijkl} \mathbf{w}_{(k,l)}^h = 0$$

From (2.7.6), this means that $w_{(i,j)}^h = 0$, and so \mathbf{w}^h is an infinitesimal rigid motion. By Assumption R, $\mathbf{w}^h = \mathbf{0}$, from which it follows that $c_p = 0$; hence $\mathbf{c} = \mathbf{0}$. ■

Remark

Positive definiteness of \mathbf{K} is based upon two requirements: a positive-definiteness condition on the constitutive coefficients and suitable boundary conditions being incorporated into \mathcal{V}^h .

2.9 ELASTOSTATICS: ELEMENT STIFFNESS MATRIX AND FORCE VECTOR

As usual, K and \mathbf{F} may be decomposed into sums of elemental contributions. These results will be omitted here as the reader should now be familiar with the ideas involved (cf. Sec. 2.5). We will proceed directly to the definitions of \mathbf{k}^ϵ and \mathbf{f}^ϵ :

$$\mathbf{k}^\epsilon = [k_{pq}^\epsilon], \quad \mathbf{f}^\epsilon = \{f_p^\epsilon\}, \quad 1 \leq p, q \leq n_{ee} = n_{ed}n_{en} \quad (2.9.1)^{15}$$

$$k_{pq}^\epsilon = e_i^T \int_{\Omega^\epsilon} \mathbf{B}_a^T \mathbf{D} \mathbf{B}_b d\Omega e_j, \quad p = n_{ed}(a-1) + i, \\ q = n_{ed}(b-1) + j \quad (2.9.2)$$

$$(n_{sd} = 2) \quad \mathbf{B}_a = \begin{bmatrix} N_{a,1} & 0 \\ 0 & N_{a,2} \\ N_{a,2} & N_{a,1} \end{bmatrix} \quad (2.9.3)$$

$$(n_{sd} = 3) \quad \mathbf{B}_a = \begin{bmatrix} N_{a,1} & 0 & 0 \\ 0 & N_{a,2} & 0 \\ 0 & 0 & N_{a,3} \\ 0 & N_{a,3} & N_{a,2} \\ N_{a,3} & 0 & N_{a,1} \\ N_{a,2} & N_{a,1} & 0 \end{bmatrix} \quad (2.9.4)$$

and

$$f_p^\epsilon = \int_{\Omega^\epsilon} N_a h_i d\Omega + \int_{\Gamma_{k_i}^\epsilon} N_a h_i d\Gamma - \sum_{q=1}^{n_{ee}} k_{pq} g_q^\epsilon, \quad \Gamma_{k_i}^\epsilon = \Gamma_{k_i} \cap \Gamma^\epsilon \\ (\text{no sum on } i) \quad (2.9.5)$$

where $g_q^\epsilon = g_{jb}^\epsilon = g_j(\mathbf{x}_b^\epsilon)$ if g_j is prescribed at node b , and equals zero otherwise.

It is useful for programming purposes to define the *nodal submatrix*

$$\underbrace{k_{ab}^\epsilon}_{n_{ed} \times n_{ed}} = \int_{\Omega^\epsilon} \mathbf{B}_a^T \mathbf{D} \mathbf{B}_b d\Omega \quad (2.9.6a)$$

From (2.9.2) we see that

¹⁵ n_{ee} stands for the *number of element equations* and n_{ed} is the *number of element degrees of freedom* (per node). It is possible in practice to have $n_{ed} < n_{dof}$, although they are usually equal.

$$k_{pq}^e = \mathbf{e}_i^T \mathbf{k}_{ab}^e \mathbf{e}_j \quad (2.9.e)$$

This means, "the pq -component of \mathbf{k}^e is the ij -component of the submatrix \mathbf{k}_{ab}^e ."

By (2.9.1) through (2.9.4), we see that \mathbf{k}^e may be written as

$$\mathbf{k}^e = \int_{\Omega^e} \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega \quad (2.9.e)$$

where

$$\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_{n_{en}}] \quad (2.9.e)$$

For example, in the case of a two-dimensional (i.e., $n_{sd}=n_{ed}=2$), four-noded element \mathbf{k}^e takes the form

$$\underbrace{\mathbf{k}^e}_{8 \times 8} = \begin{bmatrix} k_{11}^e & k_{12}^e & k_{13}^e & k_{14}^e \\ \vdots & \vdots & \vdots & \vdots \\ k_{21}^e & k_{22}^e & k_{23}^e & k_{24}^e \\ \vdots & \vdots & \vdots & \vdots \\ k_{31}^e & k_{32}^e & k_{33}^e & k_{34}^e \\ \vdots & \vdots & \vdots & \vdots \\ k_{41}^e & k_{42}^e & k_{43}^e & k_{44}^e \end{bmatrix} \quad (2.9.e)$$

In practice, the submatrices above the dashed line are computed and those below, required, are determined by symmetry.

The global arrays \mathbf{K} and \mathbf{F} may be formed from the element arrays \mathbf{k}^e and \mathbf{f}^e , respectively, by way of an assembly algorithm as outlined in Sec. 1.14.

Exercise 1. Let

$$\underbrace{\mathbf{d}^e}_{n_{ed} \times 1} = \{\mathbf{d}_a^e\} = \begin{Bmatrix} d_1^e \\ d_2^e \\ \vdots \\ d_{n_{en}}^e \end{Bmatrix} \quad (2.9.1)$$

$$(n_{ed} = 2) \quad \mathbf{d}_a^e = \begin{Bmatrix} d_{1a}^e \\ d_{2a}^e \end{Bmatrix} \quad (2.9.1)$$

$$(n_{ed} = 3) \quad \mathbf{d}_a^e = \begin{Bmatrix} d_{1a}^e \\ d_{2a}^e \\ d_{3a}^e \end{Bmatrix} \quad (2.9.1)$$

where

$$d^e = u_i^h(x^e) \quad (2.9.13)^{16}$$

d^e is called the **element displacement vector**. Show that the stress vector (see Exercise 4, Sec. 2.7.) at point $x \in \Omega^e$ can be calculated from the formula

$$\sigma(x) = D(x)B(x)d^e = D(x) \sum_{a=1}^{n_{en}} B_a(x)d_a^e \quad (2.9.14)$$

2.10 ELASTOSTATICS: DATA PROCESSING ARRAYS ID, IEN, AND LM

We have already noted that the definition of the ID array must be generalized for the present case as indicated in Sec. 2.8. We must also generalize our definition of the LM array. However, the IEN array remains the same as before.

In the present and fully general cases, the LM array is three-dimensional, with dimensions $n_{ed} \times n_{en} \times n_{el}$, and is defined by

| | |
|----------------------------------|----------|
| $LM(i, a, e) = ID(i, IEN(a, e))$ | (2.10.1) |
|----------------------------------|----------|

Degrees of freedom number Local node number Element number

Alternatively, it is sometimes convenient to think of LM as two-dimensional, with dimensions $n_{ee} \times n_{el}$, viz.,¹⁷

| | |
|---|----------|
| $LM(p, e) = LM(i, a, e), \quad p = n_{ed}(a - 1) + i$ | (2.10.2) |
|---|----------|

Local equation number Element number

To see how everything works in practice, it is helpful to run through a simple example.

Example 1

Consider the mesh of four-node, rectangular elements illustrated in Fig. 2.10.1. We assume that the local node numbering begins in the lower left-hand corner for each element and proceeds in counterclockwise fashion.

¹⁶The φ -boundary conditions are accounted for in this definition.

¹⁷The reader knowledgeable in FORTRAN will realize that the internal computer storage of (2.10.1) and (2.10.2) is identical.

Sec. 2.10 Elastostatics: Data Processing Arrays

This is shown for element 4, which is typical. Four displacement (i.e., “ g -type”) boundary conditions are specified; namely, the horizontal displacement is specified at nodes 1 and 10, and the vertical displacement is specified at nodes 1 and 3. Since $n_{np} = 12$, $n_{dof} = n_{ed} = 2$, and 4 displacement degrees of freedom are specified, we have $n_{eq} = 20$. As is usual, we adopt the convention that the global equation numbers run in ascending order with respect to the ascending order of global node numbers.¹⁸ The IEN, and LM arrays are given in Figure 2.10.2. The reader is urged to verify the results.

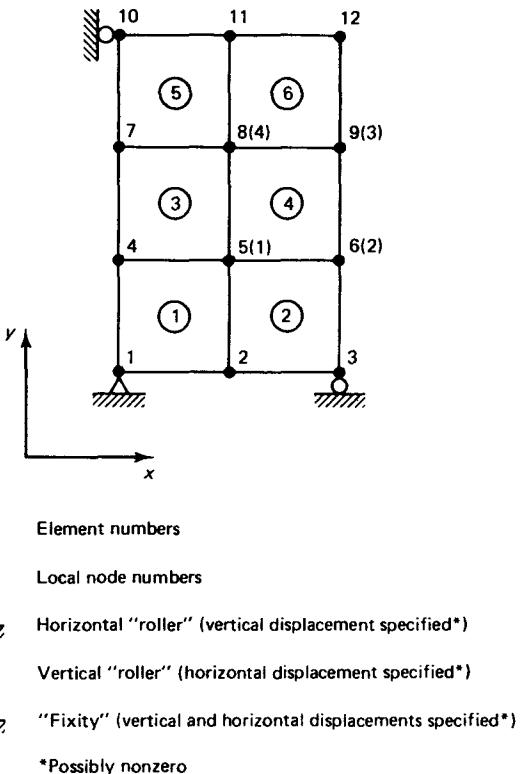


Figure 2.10.1 Mesh of four-node, rectangular, elasticity elements; global and local node numbers, element numbers, and displacement boundary conditions.

In terms of the IEN and LM arrays, a precise definition of the g_p^e ’s may be given (see (2.9.5)):

$$g_p^e = g_{ia}^e = \begin{cases} 0, & \text{if } LM(i, a, e) \neq 0 \\ g_{ia}, & \text{where } A = IEN(a, e), \text{ if } LM(i, a, e) = 0 \end{cases} \quad (2.10.1)$$

This definition may be easily programmed.

¹⁸ In practice, equation numbers are often renumbered internally to minimize the bandwidth of and thus decrease storage and solution effort. This is especially important in analyzing large-scale systems involving tens of thousands of equations. An algorithm for reducing bandwidth is presented in [8].

ID array:

| Global node numbers (A) | | | | | | | | | | | | |
|-----------------------------------|----|---|---|---|---|---|----|----|----|----|----|------------------------|
| Global degrees of freedom (i) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 0* | 1 | 3 | 4 | 6 | 8 | 10 | 12 | 14 | 0 | 17 | 19 |
| | 0 | 2 | 0 | 5 | 7 | 9 | 11 | 13 | 15 | 16 | 18 | 20 |
| $(n_{\text{dof}} = 2)$ | | | | | | | | | | | | |
| $P = \text{ID}(i, A)$ | | | | | | | | | | | | $(n_{\text{eq}} = 20)$ |

IEN array:

| Element numbers (e) | | | | | | | | | | | | |
|----------------------------|---|---|---|---|----|----|---|---|---|----|----|----|
| Local node numbers (a) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 1 | 2 | 4 | 5 | 7 | 8 | | | | | | |
| | 2 | 3 | 5 | 6 | 8 | 9 | | | | | | |
| | 5 | 6 | 8 | 9 | 11 | 12 | | | | | | |
| | 4 | 5 | 7 | 8 | 10 | 11 | | | | | | |
| $(n_{\text{en}} = 4)$ | | | | | | | | | | | | |
| $A = \text{IEN}(a, e)$ | | | | | | | | | | | | |

LM array:

| Element numbers (e) | | | | | | | | | | | | | |
|--|---|----|---|----|----|----|----|---|---|----|----|----|--|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | |
| 1 | 1 | 0* | 1 | 4 | 6 | 10 | 12 | | | | | | |
| | | 0 | 2 | 5 | 7 | 11 | 13 | | | | | | |
| | 2 | 1 | 3 | 6 | 8 | 12 | 14 | | | | | | |
| | | 2 | 0 | 7 | 9 | 13 | 15 | | | | | | |
| | 3 | 6 | 8 | 12 | 14 | 17 | 19 | | | | | | |
| | | 7 | 9 | 13 | 15 | 18 | 20 | | | | | | |
| | 4 | 4 | 6 | 10 | 12 | 0 | 17 | | | | | | |
| | | 5 | 7 | 11 | 13 | 16 | 18 | | | | | | |
| $(n_{\text{ee}} = 8)$ | | | | | | | | | | | | | |
| Element degree of freedom numbers (i , where $1 \leq i \leq n_{\text{ed}}$ and here $n_{\text{ed}} = 2$) | | | | | | | | | | | | | |
| Local node numbers (a) | | | | | | | | | | | | | |
| Local equation numbers ($p = n_{\text{ed}}(a - 1) + i$) | | | | | | | | | | | | | |

$$P = \text{LM}(p, e) = \text{LM}(i, a, e) = \text{ID}(i, \text{IEN}(a, e))$$

*Displacement boundary conditions are denoted by zeros.

Figure 2.10.2 ID, IEN, and LM arrays for the mesh of Figure 2.10.1.

Example 2

As a final example, we consider a typical four-node, elasticity element in some large mesh; see Fig. 2.10.3. We assume the pertinent entries of the ID array are given follows:

$$\left. \begin{array}{l} \text{ID}(1, 32) = 0 \\ \text{ID}(2, 32) = 0 \\ \text{ID}(1, 59) = 115 \\ \text{ID}(2, 59) = 116 \\ \text{ID}(1, 164) = 0 \\ \text{ID}(2, 164) = 325 \\ \text{ID}(1, 168) = 332 \\ \text{ID}(2, 168) = 333 \end{array} \right\} \quad (2.10.)$$

The entries of IEN follow from Fig. 2.10.3:

$$\left. \begin{array}{l} \text{IEN}(1, e) = 164 \\ \text{IEN}(2, e) = 32 \\ \text{IEN}(3, e) = 168 \\ \text{IEN}(4, e) = 59 \end{array} \right\} \quad (2.10.)$$

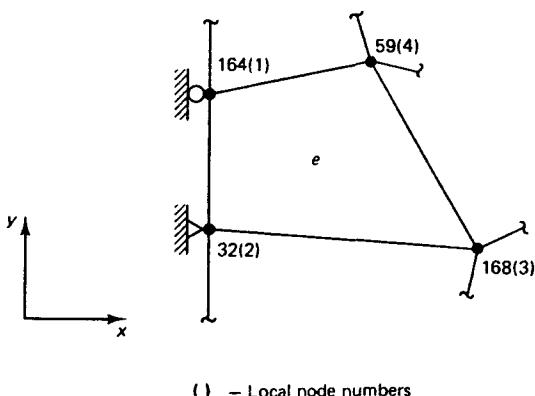


Figure 2.10.3 Typical four-node elasticity element; global and local node numbers.

Combining (2.10.4) and (2.10.5), by way of (2.10.1), yields entries of the LM array:

$$\left. \begin{array}{l} \text{LM}(1, 1, e) = 0 \\ \text{LM}(2, 1, e) = 325 \\ \text{LM}(1, 2, e) = 0 \\ \text{LM}(2, 2, e) = 0 \\ \text{LM}(1, 3, e) = 332 \\ \text{LM}(2, 3, e) = 333 \\ \text{LM}(1, 4, e) = 115 \\ \text{LM}(2, 4, e) = 116 \end{array} \right\} \quad (2.10.6)$$

The contribution to the global arrays may be deduced from LM:

Stiffness (due to symmetry, only the upper triangular portion need be assembled.)

$$\left. \begin{array}{l} K_{115, 115} \leftarrow K_{115, 115} + k_{77}^e \\ K_{115, 116} \leftarrow K_{115, 116} + k_{78}^e \\ K_{115, 325} \leftarrow K_{115, 325} + k_{72}^e \\ K_{115, 332} \leftarrow K_{115, 332} + k_{75}^e \\ K_{115, 333} \leftarrow K_{115, 333} + k_{76}^e \\ \\ K_{116, 116} \leftarrow K_{116, 116} + k_{88}^e \\ K_{116, 325} \leftarrow K_{116, 325} + k_{82}^e \\ K_{116, 332} \leftarrow K_{116, 332} + k_{85}^e \\ K_{116, 333} \leftarrow K_{116, 333} + k_{86}^e \\ \\ K_{325, 325} \leftarrow K_{325, 325} + k_{22}^e \\ K_{325, 332} \leftarrow K_{325, 332} + k_{25}^e \\ K_{325, 333} \leftarrow K_{325, 333} + k_{26}^e \\ \\ K_{332, 332} \leftarrow K_{332, 332} + k_{55}^e \\ K_{332, 333} \leftarrow K_{332, 333} + k_{56}^e \\ K_{333, 333} \leftarrow K_{333, 333} + k_{66}^e \end{array} \right\} \quad (2.10.7)$$

Force

$$\left. \begin{array}{l} F_{115} \leftarrow F_{115} + f_7^e \\ F_{116} \leftarrow F_{116} + f_8^e \\ F_{325} \leftarrow F_{325} + f_2^e \\ F_{332} \leftarrow F_{332} + f_5^e \\ F_{333} \leftarrow F_{333} + f_6^e \end{array} \right\} \quad (2.10.8)$$

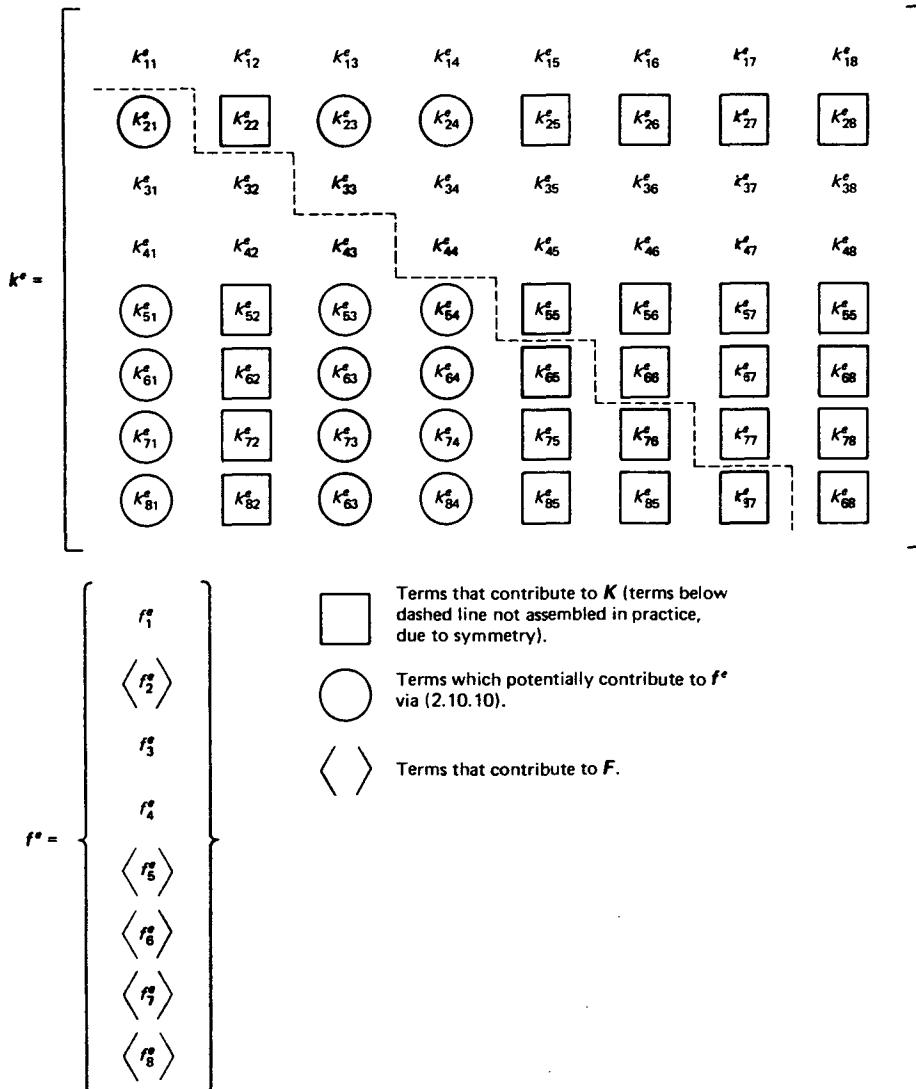
where

$$f_p^e = \dots - \sum_{q=1}^{n_e} k_{pq}^e q_q^e \quad (2.10.9)$$

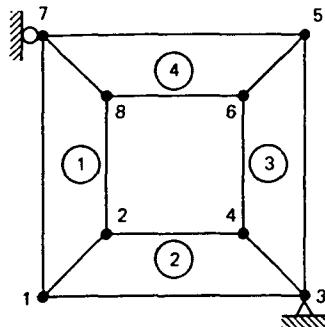
(We have omitted the first two terms in the right-hand side of (2.9.5) in writing (2.10.9).) In the present example, only q_1^e , q_3^e and q_4^e may be nonzero. Therefore (2.10.9) may be simplified to

$$f_p^e = \dots - k_{p1}^e q_1^e - k_{p3}^e q_3^e - k_{p4}^e q_4^e \quad (2.10.10)$$

The multiplications indicated in (2.10.10) are only performed in practice if the q_p^e 's are nonzero. A schematic representation of the contributions of k^e and f^e to K and F is shown in Figure 2.10.4.



Exercise 1. Consider a two-dimensional elastostatic boundary-value problem. Set up the ID, IEN, and LM arrays for the following mesh of four-node quadrilaterals:



2.11 SUMMARY OF IMPORTANT EQUATIONS FOR PROBLEMS CONSIDERED IN CHAPTERS 1 AND 2

1. Classical Linear Elastostatics

$$(S) \quad \begin{aligned} \sigma_{ij,j} + f_i &= 0 && \text{on } \Omega \\ u_i = q_i & && \text{on } \Gamma_{q_i} \\ \sigma_{ij} n_j = h_i & && \text{on } \Gamma_h \end{aligned}$$

where $\sigma_{ij} = c_{ijkl} \epsilon_{kl} = c_{ijkl} u_{(k,l)}$

(W)¹⁹ Find $u \in \mathcal{S}$, such that $\forall w \in \mathcal{V}$

$$a(w, u) = (w, \ell) + (w, h)_\Gamma$$

where

$$a(w, u) = \int_{\Omega} w_{(i,j)} c_{ijkl} u_{(k,l)} d\Omega$$

$$(w, \ell) = \int_{\Omega} w_i \ell_i d\Omega$$

$$(w, h)_\Gamma = \sum_{i=1}^{n_\Gamma} \left(\int_{\Gamma_{h_i}} w_i h_i d\Omega \right)$$

(G) Find $v^h \in \mathcal{V}^h$, such that $\forall w^h \in \mathcal{V}^h$

$$a(w^h, v^h) = (w^h, \ell) + (w^h, h)_\Gamma - a(w^h, q^h)$$

¹⁹The notation \forall means "for all."

$$(M) \quad Kd = F, \text{ where } K = \sum_{e=1}^{n_{el}} \mathbf{A}(k^e), F = F_{\text{nodal}} + \sum_{e=1}^{n_{el}} \mathbf{A}(f^e)^{20}$$

$$k_{pq}^e = \mathbf{e}_i^T \mathbf{k}_{ab}^e \mathbf{e}_j, \quad \mathbf{k}_{ab}^e = \int_{\Omega^e} \mathbf{B}_a^T \mathbf{D} \mathbf{B}_b \, d\Omega$$

$$f_p^e = \int_{\Omega^e} N_a \ell_i \, d\Omega + \int_{\Gamma_{N_e}^e} N_a h_i \, d\Gamma - \sum_{q=1}^{n_{el}} k_{pq}^e q_q^e \quad (\text{no sum on } i)$$

$$p = n_{ed}(a - 1) + i$$

$$q = n_{ed}(b - 1) + j$$

$$\text{Stress at a point: } \boldsymbol{\sigma}(x) = \mathbf{D}(x) \sum_{a=1}^{n_{en}} \mathbf{B}_a(x) d_a^e$$

2. Classical Linear Heat Conduction

$$(S) \quad \begin{aligned} q_{i,i} &= f && \text{in } \Omega \\ u &= g && \text{on } \Gamma_g \\ -q_i n_i &= h && \text{on } \Gamma_h \end{aligned}$$

where $q_i = \kappa_{ij} u_{,j}$

(W) Find $u \in \mathcal{S}$, such that $\forall w \in \mathcal{V}$

$$a(w, u) = (w, f) + (w, h)_\Gamma$$

where

$$a(w, u) = \int_{\Omega} w_{,i} \kappa_{ij} u_{,j} \, d\Omega$$

$$(w, f) = \int_{\Omega} w f \, d\Omega$$

$$(w, h)_\Gamma = \int_{\Gamma} w h \, d\Gamma$$

(G) Find $v^h \in \mathcal{S}^h$, such that $\forall w^h \in \mathcal{V}^h$

$$a(w^h, v^h) = (w^h, f) + (w^h, h)_\Gamma - a(w^h, q^h)$$

²⁰In defining F we have added to the element contributions the term F_{nodal} , which is a vector of nodal applied forces. The reason for this is that it is often easier in practice to directly input concentrated forces at nodes rather than go through the element-by-element form and assemble procedure. The expression for F then emphasizes that both modes of constructing F are to be accommodated in the computer implementation of problems of this type.

$$(M) \quad Kd = F, \text{ where } K = \sum_{e=1}^{n_{el}} \mathbf{A}(k^e), \quad F = F_{\text{nodal}} + \sum_{e=1}^{n_{el}} \mathbf{A}(f^e)^{20}$$

$$k_{ab}^e = \int_{\Omega^e} \mathbf{B}_a^T \mathbf{D} \mathbf{B}_b \, d\Omega$$

$$f_a^e = \int_{\Omega^e} N_a \ell \, d\Omega + \int_{\Gamma_k^e} N_a h \, d\Gamma - \sum_{b=1}^{n_{el}} k_{ab}^e q_b^e$$

Heat flux vector at a point: $\mathbf{q}(x) = -D(x) \sum_{a=1}^{n_{en}} \mathbf{B}_a(x) d_a^e$

3. One-Dimensional Model Problem

$$(S) \quad \begin{aligned} u_{,xx} + \ell &= 0 && \text{on } \Omega =]0, 1[\\ u(1) &= q && (\Gamma_q = \{1\}) \\ -u_{,x}(0) &= h && (\Gamma_h = \{0\}) \end{aligned}$$

(W) Find $u \in \mathcal{S}$, such that $\forall w \in \mathcal{V}$

$$a(w, u) = (w, \ell) + wh(0)$$

where

$$a(w, u) = \int_0^1 w_{,x} u_{,x} \, dx$$

$$(w, \ell) = \int_0^1 w \ell \, dx$$

(G) Find $v^h \in \mathcal{V}^h$, such that $\forall w^h \in \mathcal{V}^h$

$$a(w^h, v^h) = (w^h, \ell) + w^h(0)h - a(w^h, q^h)$$

$$(M) \quad Kd = F, \text{ where } K = \sum_{e=1}^{n_{el}} \mathbf{A}(k^e), \quad F = F_{\text{nodal}} + \sum_{e=1}^{n_{el}} \mathbf{A}(f^e)^{20}$$

$$k_{ab}^e = \int_{\Omega^e} N_{a,x} N_{b,x} \, dx$$

$$f_a^e = \int_{\Omega^e} N_a \ell \, dx + \begin{cases} h \delta_{a1} & e = 1 \\ 0 & e = 2, \dots, n_{el} - 1 \\ -q k_{2a}^e & e = n_{el} \end{cases}$$

2.12 AXISYMMETRIC FORMULATIONS AND ADDITIONAL EXERCISES

Axisymmetric formulations are expressed in terms of cylindrical coordinates:

$x_1 = r$ = the radial coordinate

$x_2 = z$ = the axial coordinate

$x_3 = \theta$ = the circumferential coordinate

The basic hypothesis of axisymmetry is that all functions under consideration are independent of θ . That is, they are functions of r and z only. Thus three-dimensional problem classes are reduced to two-dimensional ones.

Heat Conduction

The axisymmetric formulation for heat conduction is almost identical to the two-dimensional Cartesian case considered previously. The only difference is that a factor of $2\pi r$ needs to be included in each integrand of the variational equation to account for the correct volumetric weighting (e.g., $d\Omega = 2\pi r dr dz$ replaces $d\Omega = dx_1 dx_2$). Since the constant 2π is common to all terms, it may be cancelled throughout if desired.

Elasticity

The displacement components in cylindrical coordinates are:

$u_1 = u_r$ = the radial displacement

$u_2 = u_z$ = the axial displacement

$u_3 = u_\theta$ = the circumferential displacement

In addition to the basic hypothesis of axisymmetry, we further assume that $u_\theta = 0$, and thus

$$\epsilon_{r\theta} = \epsilon_{z\theta} = 0 \quad (2.12.1)$$

Note that $\epsilon_{\theta\theta} = u_r/r$ and therefore it is generally not zero. The constitutive moduli are assumed to be such that the preceding kinematical hypotheses result in

$$\sigma_{r\theta} = \sigma_{z\theta} = 0 \quad (2.12.2)$$

The preceding assumptions lead to what is called the *torsionless axisymmetric case*. This formulation is similar to but somewhat more complicated than the two-dimensional cases of plane strain and plane stress. Here there are four nonzero components of stress and strain:

$$\sigma = \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \\ \sigma_{33} \end{Bmatrix} = \begin{Bmatrix} \sigma_{rr} \\ \sigma_{zz} \\ \sigma_{rz} \\ \sigma_{\theta\theta} \end{Bmatrix} \quad (2.12.3)$$

$$\epsilon = \begin{Bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ 2\epsilon_{12} \\ \epsilon_{33} \end{Bmatrix} = \begin{Bmatrix} \epsilon_{rr} \\ \epsilon_{zz} \\ 2\epsilon_{rz} \\ \epsilon_{\theta\theta} \end{Bmatrix} \quad (2.12.4)$$

The ordering emanates from the following generalization of Table 2.7.1:

| <i>I</i> | <i>i</i> | <i>j</i> |
|----------|----------|----------|
| <i>J</i> | <i>k</i> | <i>l</i> |
| 1 | 1 | 1 |
| 3 | 1 | 2 |
| 3 | 2 | 1 |
| 2 | 2 | 2 |
| 4 | 3 | 3 |

The *D* array takes on the following form:

$$D = [D_{IJ}] = \boxed{\begin{bmatrix} D_{33} & D_3 \\ D_3^T & D_{44} \end{bmatrix}} \quad 4 \times 4 \quad (2.12.5)$$

$$D_{33} = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ & D_{22} & D_{23} \\ & & D_{33} \end{bmatrix} \quad \text{symmetric} \quad (2.12.6)$$

$$D_3 = \begin{Bmatrix} D_{14} \\ D_{24} \\ D_{34} \end{Bmatrix} \quad (2.12.7)$$

where $D_{IJ} = c_{ijkl}$, in which the indices are related by the table. The B_a -matrix takes on the form

$$B_a = \boxed{\begin{bmatrix} N_{a,1} & 0 \\ 0 & N_{a,2} \\ N_{a,2} & N_{a,1} \\ \hline & \\ \frac{N_a}{r} & 0 \end{bmatrix}} \quad (2.12.8)$$

Again, a factor of $2\pi r$ needs to be included in all integrands.

Remark

The *plane strain* case may be obtained from the axisymmetric formulation by

- i. Ignoring the $2\pi r$ factors; and
- ii. Ignoring the fourth row of B_a and the fourth row and column of D .

Furthermore, the *plane stress* case may be similarly obtained if, in addition, D_{33} is replaced by

$$D_{33} = D_3 D_{44}^{-1} D_3^T \quad (2.12.9)$$

which directly follows from the plane stress condition, $\sigma_{44} = 0$. (References [6] and [7] may be consulted for further elaboration on the physical ideas.) Sometimes (2.12.9) is referred to as the *statically condensed* elastic coefficient matrix.

Consequently, in programming the axisymmetric case, for a small amount of additional effort both plane strain and plane stress may also be included.

Exercise 1. Under the assumption of isotropy, show that D_{33} is the same as the D -matrix in (2.7.34). Furthermore, show that $D_{44} = \lambda + 2\mu$ and

$$D_3 = \begin{Bmatrix} \lambda \\ \lambda \\ 0 \end{Bmatrix} \quad (2.12.10)$$

Exercise 2. Verify that for the isotropic case, (2.12.9) achieves a similar end to the procedure described in Remark 9 of Sec. 2.7.

Exercise 3. Consider the one-dimensional model problem discussed previously. Obtain exact expressions for $f = \{f_a^e\}$, $a = 1, 2$, for the following cases (ignore q and h contributions):

- i. $f = \text{constant}$.
- ii. $f = \delta(x - \bar{x})$, the delta function, where $x_1^e \leq \bar{x} \leq x_2^e$. Specialize for the cases $\bar{x} = x_1^e$ and $\bar{x} = (x_1^e + x_2^e)/2$.

Solution

$$f_a^e = \int_{x_1^e}^{x_2^e} N_a(x) f(x) dx$$

$$\text{i. } f_a^e = f \int_{x_1^e}^{x_2^e} N_a(x) dx = \frac{fh^e}{2} \underbrace{\int_{-1}^{+1} N_a(\xi) d\xi}_{1}$$

$$f^e = \frac{fh^e}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

ii. $f_a^\epsilon = \int_{x_1^\epsilon}^{x_2^\epsilon} N_a(x) \delta(x - \bar{x}) dx = N_a(\bar{x})$
For $x = x_b^\epsilon$,

$$f_a^\epsilon = N_a(\bar{x}) = N_a(x_b^\epsilon) = \delta_{ab} \quad (\text{Kronecker delta})$$

$$f^\epsilon = \begin{cases} \delta_{1b} \\ \delta_{2b} \end{cases}$$

For $\bar{x} = (x_1^\epsilon + x_2^\epsilon)/2$,

$$f_a^\epsilon = N_a(\bar{x}) = N_a\left(\frac{x_1^\epsilon + x_2^\epsilon}{2}\right) = \frac{1}{2}$$

Therefore,

$$f^\epsilon = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Exercise 4. Consider the boundary-value problem for classical linear elastostatics discussed previously. In the linearized theory of *small displacements superposed upon large*, the stiffness term in the variational equation,

$$\int_{\Omega} w_{(i,j)} c_{ijkl} u_{(k,l)} d\Omega$$

is replaced by

$$\int_{\Omega} w_{i,j} d_{ijkl} u_{k,l} d\Omega$$

where

$$d_{ijkl} = c_{ijkl} + \delta_{ik} \sigma_{jl}^0$$

$$\sigma_{jl}^0 = \sigma_{jl}^0$$

and the σ_{jl}^0 's (i.e., *initial stresses*) are given functions of $x \in \Omega$. It follows from the symmetries of c_{ijkl} and σ_{jl}^0 that

$$d_{ijkl} = d_{klji}$$

Assume $n_{sd} = 2$. An index-free formulation of the stiffness term is given by

$$\int_{\Omega} \begin{pmatrix} w_{1,1} \\ w_{2,2} \\ w_{1,2} + w_{2,1} \\ w_{1,2} - w_{2,1} \end{pmatrix}^T \underbrace{\begin{matrix} D \\ 4 \times 4 \end{matrix}}_{\begin{matrix} u_{1,1} \\ u_{2,2} \\ u_{1,2} + u_{2,1} \\ u_{1,2} - u_{2,1} \end{matrix}} d\Omega$$

which leads to the following definition of the element stiffness matrix:

$$k_{pq}^\epsilon = e_i^T \underbrace{\int_{\Omega^\epsilon} \frac{B_a^T}{2 \times 4} \frac{D}{4 \times 4} \frac{B_b}{4 \times 2} d\Omega}_{e_j} e_j$$

Set up B_a in terms of the shape function N_a . Define the components of D in terms of the d_{ijkl} 's. (The σ_{jl}^0 -contribution to the stiffness is sometimes called the *initial-stress stiffness matrix*. It is important to account for it in the solution of many nonlinear problems.)

Exercise 5. Let Ω be a region in \mathbb{R}^2 and let its boundary $\Gamma = \overline{\Gamma}_1 \cup \overline{\Gamma}_2 \cup \overline{\Gamma}_3 \cup \overline{\Gamma}_4$ where $\Gamma_1, \dots, \Gamma_4$ are nonoverlapping subregions of Γ . Let n be the unit outward normal vector to Γ such that s and n form a right-hand rule basis; see Fig. 2.12.1. Consider the following boundary-value problem in classical linear elastostatics: Given $f_i : \Omega \rightarrow \mathbb{R}$; $g_i : \Gamma_1 \rightarrow \mathbb{R}$; $h_i : \Gamma_2 \rightarrow \mathbb{R}$; g_n and $h_s : \Gamma_3 \rightarrow \mathbb{R}$; and g_s and $h_n : \Gamma_4 \rightarrow \mathbb{R}$; find $u_i : \overline{\Omega} \rightarrow \mathbb{R}$ such that

$$\begin{aligned} \sigma_{ij,j} + f_i &= 0 && \text{in } \Omega \\ u_i &= g_i && \text{on } \Gamma_1 \\ \sigma_{ij}n_j &= h_i && \text{on } \Gamma_2 \\ \left. \begin{aligned} u_i n_i &= g_n \\ \sigma_{ij}n_j s_i &= h_s \end{aligned} \right\} &= \left. \begin{aligned} &\text{on } \Gamma_3 && \left(\begin{array}{l} \text{normal displacement} \\ \text{tangential traction} \end{array} \right) \\ u_i s_i &= g_s \\ \sigma_{ij}n_j n_i &= h_n \end{aligned} \right\} &= \left. \begin{aligned} &\text{on } \Gamma_4 && \left(\begin{array}{l} \text{tangential displacement} \\ \text{normal traction} \end{array} \right) \end{aligned} \right. \end{aligned}$$

where $\sigma_{ij} = c_{ijkl} u_{(k,l)}$. Establish a weak formulation for this problem in which all “ g -type” boundary conditions are essential and all “ h -type” boundary conditions are natural. State all requirements on the spaces \mathcal{S} and \mathcal{V} . Hint: $w = w_n n + w_s s$; i.e., $w_i = w_n n_i + w_s s_i$.

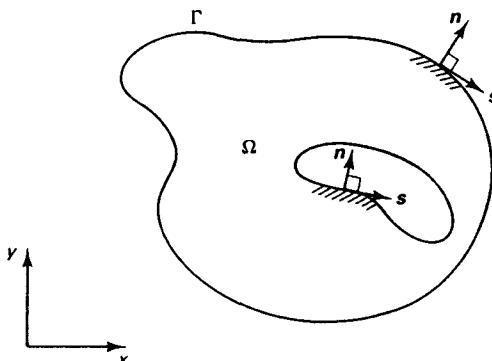


Figure 2.12.1

Exercise 6. In practice, it is often useful to generalize the constitutive equation of classical elasticity to the form

$$\sigma_{ij} = c_{ijkl}(\epsilon_{kl} - \epsilon_{kl}^0) + \sigma_{ij}^0 \quad (2.12.11)$$

where ϵ_{ij}^0 and σ_{ij}^0 are the *initial strain* and *initial stress*, both given functions of x . The initial strain term may be used to represent thermal expansion effects by way of

$$\epsilon_{kl}^0 = -\theta c_{kl} \quad (2.12.12)$$

where θ is the *temperature* and the c_{kl} 's are the *thermal expansion coefficients* (both given functions). Clearly, (2.12.11) will in no way change the stiffness matrix. However there will be additional contributions to f_p^e . Generalize the definition of f_p^e to account for these additional terms.

Solution We begin with the weak form

$$\int_{\Omega} w_{(i,j)} \sigma_{ij} d\Omega = \int_{\Omega} w_i f_i d\Omega + \sum_{i=1}^{n_s} \left(\int_{\Gamma_{h_i}} w_i h_i d\Gamma \right)$$

Substituting (2.12.11) and (2.12.12) into the weak form leads to

$$\int_{\Omega} w_{(i,j)} c_{ijkl} \epsilon_{kl} d\Omega = \int_{\Omega} w_i f_i d\Omega + \sum_{i=1}^{n_{sd}} \left(\int_{\Gamma_{ki}} w_i h_i d\Gamma \right)$$

↑ As before
 ↓ New contributions
 to right-hand side:
 $+ \int_{\Omega} w_{(i,j)} c_{ijkl} \epsilon_{kl}^0 d\Omega$
 $- \int_{\Omega} w_{(i,j)} \sigma_{ij}^0 d\Omega$

from which the additional terms in f_p^e may be deduced:

$$f_p^e = \dots + e_i^T \int_{\Omega^e} B_a^T D \theta c d\Omega - e_i^T \int_{\Omega^e} B_a^T \sigma^0 d\Omega$$

↑ Anywhere
 inside integral

where

$$c = \begin{Bmatrix} c_{11} \\ c_{22} \\ 2c_{12} \end{Bmatrix}; \quad \sigma^0 = \begin{Bmatrix} \sigma_{11}^0 \\ \sigma_{22}^0 \\ \sigma_{12}^0 \end{Bmatrix}; \dots$$

↑ Without loss of generality, we
 may assume symmetry, i.e., $c_{12} + c_{21} = 2c_{12}$

Exercise 7. Consider the following boundary-value problem:

$$u_{,xx} - p(p-1)x^{p-2} = 0, \quad 0 < x < 1$$

$$-u_{,x}(0) = 0$$

$$u(1) = 1$$

where p is a given constant.

- Obtain the exact solution to this problem for $p = 5$. Sketch.
- State the weak formulation of the problem.
- State the Galerkin formulation.
- State the matrix formulation.
- Solve the matrix problem assuming $p = 5$ and using the piecewise linear finite element space for the following cases:
 - one element
 - two equal-length elements
- Compare the exact value of $u_{,x}(1)$ with the approximate values computed in part v. Explain why it is impossible for these results to compare favorably.

Exercise 8. In heat conduction, it is often of interest to accurately calculate the boundary heat flux over a portion of the boundary where temperature is specified. Suppose we use the usual Galerkin finite element formulation to calculate the temperature. However, instead of calculating the heat flux in the usual way (i.e., by differentiating the temperature), we introduce a *post-processing* which derives from the following weak formulation:

Find $u \in \mathcal{S}$ and $h \in L_2(\Gamma_e)$ such that for all $w \in \mathcal{V}$,

$$-\int_{\Omega} w_i q_i d\Omega = \int_{\Omega} w \ell d\Omega + \int_{\Gamma_h} wh d\Gamma + \int_{\Gamma_e} wh d\Gamma$$

where h is the *unknown* heat flux on Γ_e .

(Note: In this formulation, it is *not* assumed that $w = 0$ on Γ_e !)

- i. Show, in addition to the usual differential equations and boundary conditions, that

$$h = -q_i n_i \quad \text{on } \Gamma_e$$

arises naturally from the new weak formulation.

- ii. State the Galerkin and matrix formulations corresponding to the new weak formulation assuming h is approximated in the usual way, namely

$$h^h(x) = \sum_{A \in \mathcal{N}_e} N_A(x) h_A$$

(Hint: The equations governing the temperature are unchanged.)

- iii. Specialize this formulation to the one-dimensional problem described in Exercise 7 and calculate the boundary flux at $x = 1$ by the new procedure (cf. parts v and vi of Exercise 7).

(Hint: The new method should produce exact results for these cases.)

- iv. Develop a counterpart of the new formulation for elasticity. That is, introduce the i th component of traction as an independent unknown on Γ_{ei} and carefully state the weak formulation.
- v. Prove that the new method is exact for the one-dimensional model problem of Chapter 1.

REFERENCES

Section 2.2

1. J. E. Marsden and A. J. Tromba, *Vector Calculus*. San Francisco: W. H. Freeman, 1976.

Section 2.4

2. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*. Englewood Cliffs, N.J.: Prentice-Hall, 1973.

Section 2.7

3. G. Duvaut and J. L. Lions, *Les Inéquations en Mécanique et en Physique*. Paris: Dunod, 1972.

4. G. Fichera, "Existence Theorems in Elasticity," in *Handbuch der Physik, Volume VIa/2, Mechanics of Solids II*, ed. C. Truesdell. New York: Springer-Verlag, 1972.
5. M. Gurtin, "The Linear Theory of Elasticity," in *Handbuch der Physik, Volume VIa/2, Mechanics of Solids II*, ed. C. Truesdell. New York: Springer-Verlag, 1972.
6. I. S. Sokolnikoff, *Mathematical Theory of Elasticity* (2nd ed.). New York: McGraw-Hill, 1956.
7. S. Timoshenko and J. N. Goodier, *Theory of Elasticity* (3rd ed.). New York: McGraw-Hill, 1969.

Section 2.10

8. N. E. Gibbs, W. G. Poole, Jr., and P. K. Stockmeyer, "An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix," *SIAM Journal of Numerical Analysis*, 13 (1976), 236–250.

3

Isoparametric Elements and Elementary Programming Concepts

3.1 PRELIMINARY CONCEPTS

We wish to define the shape functions in such a way that, as the finite element mesh is refined, the approximate Galerkin solution converges to the exact solution. The following question arises: What conditions must the shape functions satisfy so that this property is guaranteed? We shall be content, for the time being, to state sufficient conditions for convergence. These conditions are possessed by the most prevalent and important finite element shape functions. However, we note that convergent elements can be constructed from shape functions which do not satisfy all these requirements. Nevertheless these conditions may be considered basic in that they provide the simplest criteria to ensure convergence for a wide class of problems.

The basic convergence requirements are that the shape functions be

- C1. smooth (i.e., at least C^1) on each element interior, Ω^e ;
- C2. continuous across each element boundary Γ^e ; and
- C3. complete.

Remarks

1. Conditions C1 and C2 guarantee that first derivatives of the shape functions have, at worst, finite jumps across the element interfaces; see Fig. 3.1.1. This ensures that all integrals necessary for the computation of element arrays (see Sec. 2.11) are well defined, since at most first derivatives appear in the integrands. If we permit finite discontinuities in the shape functions on element boundaries, the derivatives possess delta functions (cf. Sec. 1.10) and we are unable to make sense out of the squares of these quantities that would appear in the stiffness integrands.

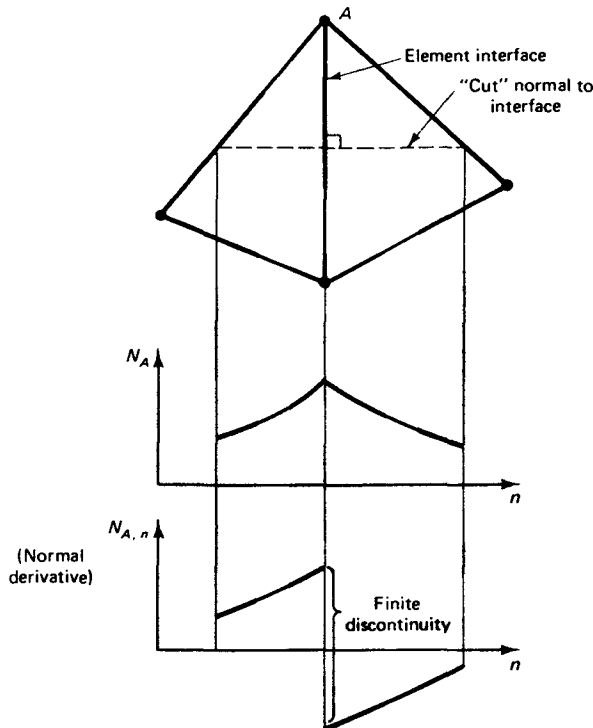


Figure 3.1.1 Example of a shape function that satisfies conditions C1 and C2.

2. Shape functions that satisfy C1 and C2 are of class $C^0(\bar{\Omega})$. Finite elements constructed from $C^0(\bar{\Omega})$ shape functions are often referred to as C^0 -elements.

3. Theories such as Bernoulli-Euler beam theory, which require higher-order derivatives in the stiffness integrands, necessitate strengthening condition C1 to C^2 -continuity on Ω^e and condition C2 to C^1 -continuity across Γ^e . Shape functions of this type are of class $C^1(\bar{\Omega})$ and lead to so-called C^1 -elements. Examples of $C^1(\bar{\Omega})$ shape functions are the Hermite cubics discussed in Sec. 1.16.

4. If the stiffness integrands involve derivatives of order m , Condition C1 should be strengthened to C^m -continuity on Ω^e and Condition C2 should be strengthened to C^{m-1} -continuity across Γ^e . Finite elements that satisfy this property are called *conforming*, or *compatible*. (The use of elements that violate this property, *non-conforming* or *incompatible elements*, is, however, common. We say more about this later.)

5. It may be noted that C2 is equivalent to requiring that each function $u^e \in \mathcal{S}^e$ be continuous across Γ^e . In specific cases it is often easier to establish this fact directly rather than prove it for individual shape functions.

Completeness

To illustrate the completeness requirement, let us take the case of heat conduction in which the e th element interpolation function may be written

$$u^h = \sum_{a=1}^{n_{sd}} N_a d_a^e \quad (3.1.1)$$

where $d_a^e = u^h(x_a^e)$. Let $n_{sd} = 3$. In this case, the shape functions are said to be *complete* if

$$d_a^e = c_0 + c_1 x_a^e + c_2 y_a^e + c_3 z_a^e \quad (3.1.2)$$

implies that

$$u^h(x) = c_0 + c_1 x + c_2 y + c_3 z \quad (3.1.3)$$

where c_0, \dots, c_3 are arbitrary constants. In two dimensions, we omit the z -terms in (3.1.2) and (3.1.3). In words, *completeness requires that the element interpolation function is capable of exactly representing an arbitrary linear polynomial when the nodal degrees of freedom are assigned values in accordance with it.*

Completeness is a plausible requirement as the following argument indicates: As the finite element mesh is further and further refined, the exact solution and its derivatives approach constant values over each element domain. To ensure that these constant values are representable, the shape functions must contain all constant and linear monomials. This argument was originally given in [1] and has subsequently been proved to be the key mathematical idea for proving convergence theorems for finite element approximations (e.g., see [2]).

For elasticity, the requirement is essentially the same. In this case we may write

$$u_i^h = \sum_{a=1}^{n_{sd}} N_a d_{ia}^e \quad (3.1.4)$$

where $d_{ia} = u_i^h(x_a)$. The N_a are complete if

$$d_{ia}^e = c_0 + c_1 x_a^e + c_2 y_a^e + c_3 z_a^e \quad (3.1.5)$$

implies

$$u_i^h(x) = c_0 + c_1 x + c_2 y + c_3 z \quad (3.1.6)$$

Remarks

6. In elasticity, the presence of all monomials through linear terms means that an element may exactly represent all rigid motions (i.e., rigid translations and rotations) and constant strain states. We thus often speak of completeness in terms of the ability to represent rigid motions and constant strains (e.g., see [3], p. 33).

7. For theories involving m th derivatives in the stiffness integrands, completeness must be strengthened to m th-order polynomials.

Example 1

The piecewise linear finite element space introduced in Chapter 1 satisfies the convergence conditions C1–C3. C1 and C2 follow from the definition (see Sec. 1.8). For C3 we proceed as follows: Assume $d_a^e = c_0 + c_1 x_a^e$ and substitute in $u^h = \sum_{a=1}^2 N_a d_a^e$, viz.,

$$\begin{aligned}
 u^h &= \sum_{a=1}^2 N_a d_a^\epsilon \\
 &= \sum_{a=1}^2 N_a (c_0 + c_1 x_a^\epsilon) \\
 &= \left(\sum_{a=1}^2 N_a \right) c_0 + \left(\sum_{a=1}^2 N_a x_a^\epsilon \right) c_1
 \end{aligned} \tag{3.1.7}$$

Thus to establish completeness we must show that

$$\sum_{a=1}^2 N_a = 1 \tag{3.1.8}$$

and

$$\sum_{a=1}^2 N_a x_a^\epsilon = x \tag{3.1.9}$$

By (1.12.5), $N_a(\xi) = (1 + (-1)^a \xi)/2$, from which (3.1.8) follows. Equation (3.1.9) was established previously (see (1.12.6)).

3.2 BILINEAR QUADRILATERAL ELEMENT

This element is attributed to Taig [4]. The rectangular version was proposed earlier by Argyris in [5].

The domain of a straight-edged quadrilateral element is defined by the locations of its four nodal points x_a^ϵ , $a = 1, \dots, 4$ in the \mathbb{R}^2 -plane. We assume the nodal points are labeled in ascending order corresponding to the counterclockwise direction (see Fig. 3.2.1). We seek a change of coordinates which maps the given quadrilateral into the biunit square, as depicted in Fig. 3.2.1. The biunit square is sometimes called the *parent domain*. The coordinates of a point

$$\begin{Bmatrix} \xi \\ \eta \end{Bmatrix} \tag{3.2.1}$$

in the biunit square are to be related to the coordinates of a point

$$\begin{Bmatrix} x \\ y \end{Bmatrix} \tag{3.2.2}$$

in Ω^ϵ by mappings of the form

$$x(\xi, \eta) = \sum_{a=1}^4 N_a(\xi, \eta) x_a^\epsilon \tag{3.2.3}$$

$$y(\xi, \eta) = \sum_{a=1}^4 N_a(\xi, \eta) y_a^\epsilon \tag{3.2.4}$$

ξ and η are sometimes called the *natural coordinates*. A more succinct representation of the above formulas is

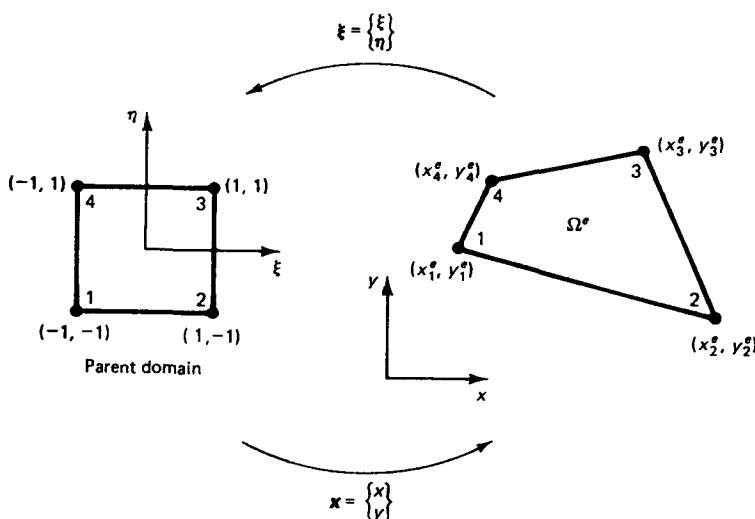


Figure 3.2.1 Bilinear quadrilateral element domain and local node ordering.

$$\mathbf{x}(\xi) = \sum_{a=1}^4 N_a(\xi) \mathbf{x}_a^e \quad (3.2.5)$$

We obtain the functions N_a by first *assuming* the "bilinear" expansions

$$x(\xi, \eta) = \alpha_0 + \alpha_1 \xi + \alpha_2 \eta + \alpha_3 \xi \eta \quad (3.2.6)$$

$$y(\xi, \eta) = \beta_0 + \beta_1 \xi + \beta_2 \eta + \beta_3 \xi \eta \quad (3.2.7)$$

where the α 's and β 's are parameters to be determined; and second, by stipulating that (3.2.6) and (3.2.7) satisfy the (respective) conditions

$$x(\xi_a, \eta_a) = x_a^e \quad (3.2.8)$$

$$y(\xi_a, \eta_a) = y_a^e \quad (3.2.9)$$

where ξ_a and η_a are defined in Table 3.2.1.

TABLE 3.2.1 Nodal coordinates in ξ -space

| a | ξ_a | η_a |
|-----|---------|----------|
| 1 | -1 | -1 |
| 2 | 1 | -1 |
| 3 | 1 | 1 |
| 4 | -1 | 1 |

Conditions (3.2.8) and (3.2.9) impose restrictions on the functions N_a ; namely,

$$N_a(\xi_b) = \delta_{ab} \quad (3.2.10)^1$$

To see this we can combine (3.2.3) and (3.2.8), viz.,

$$x_b^\epsilon = x(\xi_b, \eta_b) = \sum_{a=1}^4 N_a(\xi_b, \eta_b) x_a^\epsilon \quad (3.2.11)$$

which holds only if $N_a(\xi_b, \eta_b) = \delta_{ab}$. This same restriction may be deduced by combining (3.2.4) and (3.2.9). Equations (3.2.6) through (3.2.9) lead to the following matrix equations:

$$\begin{Bmatrix} x_1^\epsilon \\ x_2^\epsilon \\ x_3^\epsilon \\ x_4^\epsilon \end{Bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{Bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{Bmatrix} \quad (3.2.12)$$

$$\begin{Bmatrix} y_1^\epsilon \\ y_2^\epsilon \\ y_3^\epsilon \\ y_4^\epsilon \end{Bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{Bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{Bmatrix} \quad (3.2.13)$$

In each case the coefficient matrix is the same. Solving these for the α 's and β 's, substituting the results into (3.2.6) and (3.2.7), and comparing with (3.2.3) and (3.2.4) reveals that

$$N_a(\xi) = N_a(\xi, \eta) = \frac{1}{4}(1 + \xi_a \xi)(1 + \eta_a \eta) \quad (3.2.14)$$

Note that this is precisely the product of one-dimensional shape functions derived previously (see (1.12.5)).

Exercise 1. Verify (3.2.14).

Observe that coordinate lines in ξ -space (i.e., lines such that $\xi = \text{constant}$ or $\eta = \text{constant}$) are mapped into straight lines in x -space. In particular, boundary lines are mapped into boundary lines (see Fig. 3.2.2).

Now we shall *assume* that the element functions are given by similar expansions in terms of the shape functions.

Heat conduction

$$u^h(\xi) = \sum_{a=1}^4 N_a(\xi) d_a^\epsilon \quad (3.2.15)$$

¹Equation (3.2.10) is sometimes referred to as the *interpolation property* since it implies that the assumed expansions (e.g., (3.2.6) and (3.2.7)) interpolate the nodal data (e.g., (3.2.8) and (3.2.9)).

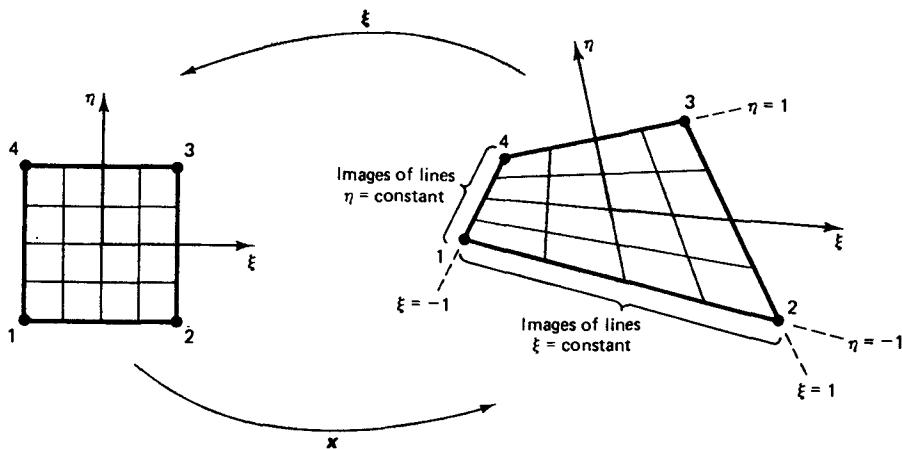


Figure 3.2.2

Elastostatics

$$u_i^h(\xi) = \sum_{a=1}^4 N_a(\xi) d_{ia}^e \quad (3.2.16)$$

Henceforce, it will be sufficient to consider only (3.2.15). The case of elastostatics may be deduced by viewing (3.2.15) as any one of the n_{sd} component functions of (3.2.16).

(C1) Smoothness on Ω^o . It can be shown that N_a is a smooth function of x and y , if all interior angles formed by two adjacent edges are less than 180° ; see Fig. 3.2.3. (Note that N_a is always a smooth function of ξ and η ; see (3.2.14). When N_a is viewed as a function of x and y (i.e., $N_a(\xi(x, y), \eta(x, y))$), to ascertain smoothness we must consider the inverse functions $\xi(x, y)$ and $\eta(x, y)$, which may not be well defined if the element is too distorted. Figure 3.2.3 illustrates when this occurs for the four-node element. This issue is discussed from a more general perspective in the following section.)

(C2) Continuity across Γ^o . Let us examine a typical shape function N_a . For the moment, assume $a = 1$ or 2 . The behavior of N_a along the edge connecting nodes 1 and 2 may be determined by substituting $\eta = -1$ into (3.2.14). The result is

$$N_a(\xi, -1) = \frac{1 + \xi_a \xi}{2}, \quad a = 1, 2 \quad (3.2.17)$$

From Table 3.2.1, $\xi_1 = -1$ and $\xi_2 = +1$; thus the right-hand side of (3.2.17) is the familiar linear shape function of the one-dimensional theory (see Fig. 3.2.4). By virtue of the fact that (3.2.17) is typical of all four edges and by (3.2.10), we conclude

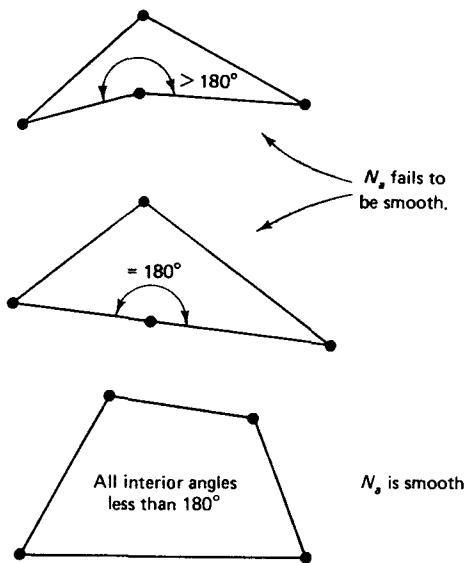


Figure 3.2.3 Smoothness criterion for bilinear shape functions.

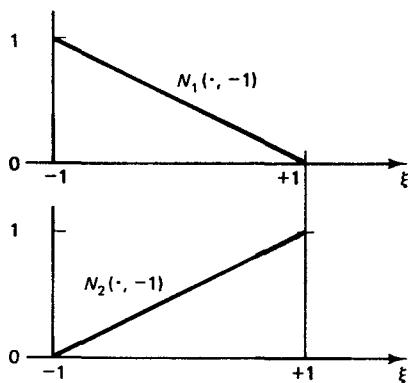


Figure 3.2.4 Bilinear shape functions along edge connecting nodes 1 and 2.

that N_a appears as in Fig. 3.2.5. N_a is said to be a *hyperbolic paraboloid*. Similar considerations for adjacent elements indicate that N_A is continuous and appears as a “tent” whose “pole” emanates from node A ; see Fig. 3.2.6 for a typical situation at an internal node. Consequently, \mathcal{S}^h consists of continuous functions as each member is a linear combination of the N_A ’s (cf. (2.4.3)–(2.4.5)). It is instructive to argue this fact from a slightly different point of view.

Consider the behavior of (3.2.15) along the boundary segment connecting x_1^ϵ to x_2^ϵ . By (3.2.17), this may be written

$$u^h(\xi, -1) = \sum_{a=1}^2 \frac{1}{2}(1 + \xi_a \xi) d_a^\epsilon \quad (3.2.18)$$

There are two important observations to be made about this result:

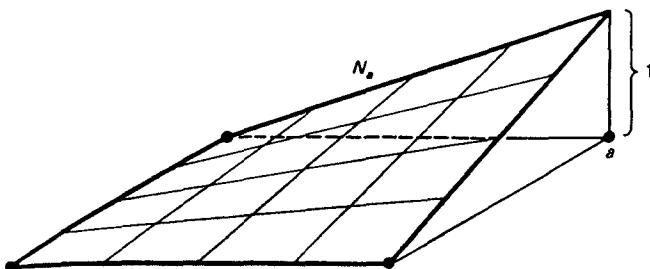


Figure 3.2.5 Bilinear shape function.

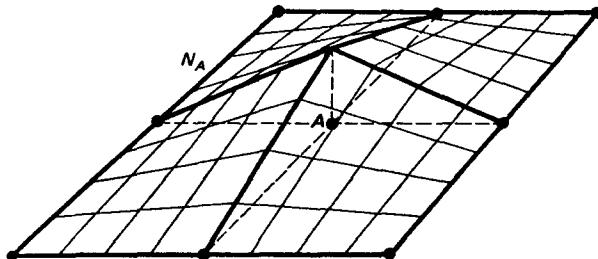
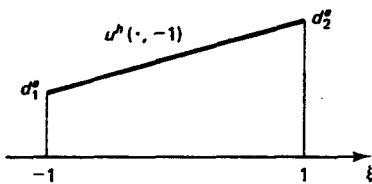


Figure 3.2.6 Global shape function constructed from element bilinear shape functions.

1. The behavior of u^h along $[x_1^\epsilon, x_2^\epsilon]$ is determined solely by the nodal values of u^h at the nodes x_1^ϵ and x_2^ϵ .
2. The variation of u^h is linear in the natural coordinate ξ (see Fig. 3.2.7).

Figure 3.2.7 Graph of u^h along boundary segment joining nodes x_1^ϵ and x_2^ϵ .

The same conclusion can be drawn for any other element of this type whose boundary contains $[x_1^\epsilon, x_2^\epsilon]$. Thus we see that continuity of u^h is automatic across element interfaces.

(C3) Completeness

$$\begin{aligned}
 u^h &= \sum_{a=1}^{n_{eq}} N_a d_a^\epsilon \\
 &= \sum_{a=1}^{n_{eq}} N_a (c_0 + c_1 x_a^\epsilon + c_2 y_a^\epsilon) \\
 &= \left(\sum_{a=1}^{n_{eq}} N_a \right) c_0 + \underbrace{\left(\sum_{a=1}^{n_{eq}} N_a x_a^\epsilon \right)}_{= x \text{ by (3.2.3)}} c_1 + \underbrace{\left(\sum_{a=1}^{n_{eq}} N_a y_a^\epsilon \right)}_{= y \text{ by (3.2.4)}} c_2
 \end{aligned}$$

Thus it remains only to prove that the shape functions sum to value 1 at each point. This may be done by explicit calculation, viz.,

$$\begin{aligned} \sum_{a=1}^{n_{en}} N_a(\xi, \eta) &= \frac{1}{4}(1 - \xi)(1 - \eta) + \frac{1}{4}(1 + \xi)(1 - \eta) \\ &\quad + \frac{1}{4}(1 + \xi)(1 + \eta) + \frac{1}{4}(1 - \xi)(1 + \eta), \quad (\text{by (3.2.14)}) \\ &= 1 \end{aligned} \quad (3.2.20)$$

3.3 ISOPARAMETRIC ELEMENTS

The isoparametric concept is generally attributed to Taig [6] and Irons [7].

Let \square denote a parent domain in ξ -space (e.g., the closed, biunit square in \mathbb{R}^2).

Definition 1. Let $x: \square \rightarrow \bar{\Omega}^e$ be of the form

$$x(\xi) = \sum_{a=1}^{n_{en}} N_a(\xi) x_a^e \quad (3.3.1)$$

If the element interpolation function u^h can be written as

$$u^h(\xi) = \sum_{a=1}^{n_{en}} N_a(\xi) d_a^e \quad (3.3.2)$$

the element is said to be *isoparametric*.

The key point to observe in the definition is that the shape functions which define (3.3.1) also serve to define (3.3.2).

The bilinear quadrilateral, derived in the previous section, is an example of an isoparametric element.

We shall now discuss some fundamental mathematical properties of mappings. Based on this, we shall argue that convergence condition C1 is generally achieved for isoparametric elements. Furthermore, a criterion, which may be used in a computer program to determine whether or not individual elements satisfy C1, emanates from the discussion.

Convergence Condition C1

Definition 2. A mapping $x: \square \rightarrow \bar{\Omega}^e \subset \mathbb{R}^{n_{sd}}$ is said to be *one-to-one* if for each pair of points $\xi^{(1)}, \xi^{(2)} \in \square$ such that $\xi^{(1)} \neq \xi^{(2)}$, then $x(\xi^{(1)}) \neq x(\xi^{(2)})$.

In words, this statement means that two different points of \square do not get mapped into the same point in $\bar{\Omega}^e$.

Definition 3. $x: \square \rightarrow \bar{\Omega}^e$ is said to be *onto* if $\bar{\Omega}^e = x(\square)$ (i.e., each point in $\bar{\Omega}^e$ is the image of a point in \square under the mapping x).

Definition 4. Let $x : \square \rightarrow \bar{\Omega}^e$ be a differentiable mapping. The determinant of the derivative, denoted by $j = \det(\partial x / \partial \xi)$, is called the *Jacobian determinant*.

The Jacobian determinants in two and three dimensions, respectively, are given explicitly by

$$j = \det \begin{bmatrix} x_{,\xi} & x_{,\eta} \\ y_{,\xi} & y_{,\eta} \end{bmatrix} \quad (3.3.3)$$

$$j = \det \begin{bmatrix} x_{,\xi} & x_{,\eta} & x_{,\zeta} \\ y_{,\xi} & y_{,\eta} & y_{,\zeta} \\ z_{,\xi} & z_{,\eta} & z_{,\zeta} \end{bmatrix} \quad (3.3.4)$$

Remark

1. It is a consequence of the inverse function theorem (e.g., see [8]) that if x is

- i. one-to-one;
- ii. onto;
- iii. C^k , $k \geq 1$; and if
- iv. $j(\xi) > 0$ for all $\xi \in \square$;

then the inverse mapping $\xi = x^{-1} : \bar{\Omega}^e \rightarrow \square$ exists and is C^k .

Proposition 1. Let the mapping defined by (3.3.1) satisfy (i) through (iv). Then the smoothness requirement (C1) is satisfied.

Proof. By virtue of the hypotheses, $N_a = N_a(\xi)$ is also a C^1 function. By Remark 1, $\xi = \xi(x)$ is also C^1 . Thus $N_a(x) = N_a(\xi(x))$ is a C^1 function of x . (This last fact may be proved with the aid of the chain rule.) ■

Remarks

2. In practice, the mappings $x : \square \rightarrow \bar{\Omega}^e$ usually satisfy (i) through (iv). However, there is one exception of practical importance. It is concerned with the technique of element “degeneration,” in which nodes are coalesced. The simplest example of this procedure, in which two nodes of the bilinear quadrilateral are coalesced to form a triangle, is presented in the next section. Further examples are presented in subsequent sections. When degeneration is performed, the Jacobian determinant vanishes at certain nodal points within the element. Away from these points it is positive, and the mapping $\xi = \xi(x)$ remains smooth (i.e., C1 is satisfied). For reasons that will be apparent later on, it is not usually required to calculate derivatives at these points.

3. In actual computations the sign of the Jacobian determinant is monitored at special points to ensure condition (iv) is satisfied. If a zero or negative value is encountered, computations are terminated. Generally this is an indication of an input-data error or an overly distorted element.

Convergence Condition (C3)

Proposition 2. If $\sum_{a=1}^{n_{en}} N_a = 1$, then completeness condition C3 is satisfied for isoparametric elements.

Proof. (We shall prove the assertion for the three-dimensional case.)

$$\begin{aligned} u^h &= \sum_{a=1}^{n_{en}} N_a d_a^\epsilon && \text{(by (3.3.2))} \\ &= \sum_{a=1}^{n_{en}} N_a (c_0 + c_1 x_a^\epsilon + c_2 y_a^\epsilon + c_3 z_a^\epsilon) \\ &= c_0 \left(\sum_{a=1}^{n_{en}} N_a \right) + c_1 \left(\sum_{a=1}^{n_{en}} N_a x_a^\epsilon \right) + c_2 \left(\sum_{a=1}^{n_{en}} N_a y_a^\epsilon \right) + c_3 \left(\sum_{a=1}^{n_{en}} N_a z_a^\epsilon \right) \\ &= c_0 \left(\sum_{a=1}^{n_{en}} N_a \right) + c_1 x + c_2 y + c_3 z && \text{(by (3.3.1))} \end{aligned}$$

The condition that the shape functions sum to one, assumed in Proposition 2, is easily checked on a case-by-case basis. ■

The only remaining convergence condition is C2, the continuity requirement on Γ^ϵ . This condition can be verified once the construction of the global shape functions from the element shape functions is explicated. It happens that if this procedure is done in the “obvious” way, continuity is achieved. In the sequel we shall consider this issue on a case-by-case basis.

Summary. The importance of the isoparametric concept is that the three basic convergence conditions are virtually automatic. In addition, isoparametric elements may be designed to take on convenient shapes for practical analysis, including curved boundaries, and lend themselves to concise computer implementation.

3.4 LINEAR TRIANGULAR ELEMENT; AN EXAMPLE OF “DEGENERATION”

The linear triangle was one of the first finite elements conceived (see Courant [9] and Turner et al. [10]) and is still widely used. In the structural mechanics literature it is often referred to as the constant stress/strain triangle, or simply the CST. We shall derive it from the bilinear quadrilateral by coalescing nodes 3 and 4; see Fig. 3.4.1. Specifically, we set $x_4^\epsilon = x_3^\epsilon$ in (3.2.5) and define new shape functions for the triangle N'_a , $a = 1, 2, 3$, as follows:

$$\begin{aligned} x &= \sum_{a=1}^4 N_a x_a^\epsilon \\ &= N_1 x_1^\epsilon + N_2 x_2^\epsilon + (N_3 + N_4) x_3^\epsilon = \sum_{a=1}^3 N'_a x_a^\epsilon && \text{(3.4.1)} \end{aligned}$$

Jacobian determinant is zero at node 3

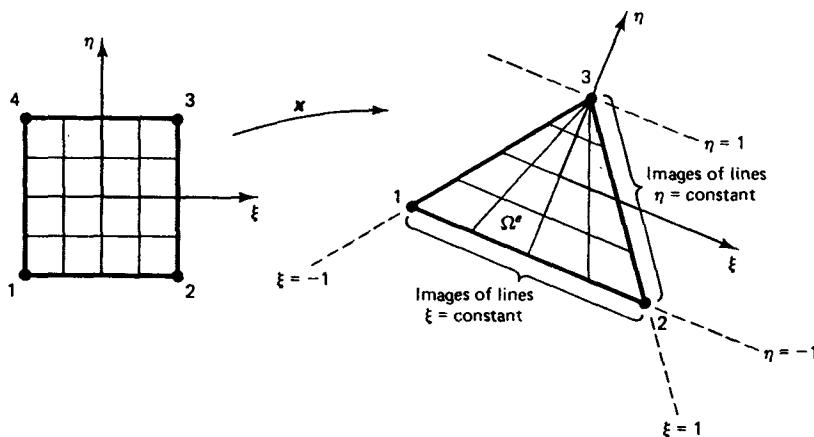


Figure 3.4.1

where

$$N'_a = \begin{cases} N_a = \frac{1}{4}[1 + (-1)^a \xi](1 - \eta) & a = 1, 2 \\ N_3 + N_4 = \frac{1}{2}(1 + \eta) & a = 3 \end{cases} \quad (3.4.2)$$

The shape functions are illustrated in Fig. 3.4.2; they are "planes" above Ω' , and so the derivatives with respect to coordinates x and y are constants. In this case note that (3.4.1) is not one-to-one, since the boundary segment $\xi \in [-1, 1]$, $\eta = 1$ in ξ -space is mapped into the point x_3 in x -space. In fact, the Jacobian determinant is zero at x_3 . However, as we have already seen, the derivatives with respect to x and y are well behaved. Thus condition C1 is satisfied.

The continuity condition C2 may be inferred from Fig. 3.4.2. A typical global shape function is illustrated in Fig. 3.4.3. Sometimes these functions are called *pyramids*, for obvious reasons. An alternative verification of continuity may be attained by evaluating the element interpolation function

$$u^h = \sum_{a=1}^3 N'_a d_a^x \quad (3.4.3)$$

along the boundaries, viz.,

(Side 1, 2)

$$u^h(\xi, -1) = \frac{1}{2}(1 - \xi)d_1^x + \frac{1}{2}(1 + \xi)d_2^x \quad (3.4.4)$$

(Side 2, 3)

$$u^h(1, \eta) = \frac{1}{2}(1 - \eta)d_2^x + \frac{1}{2}(1 + \eta)d_3^x \quad (3.4.5)$$

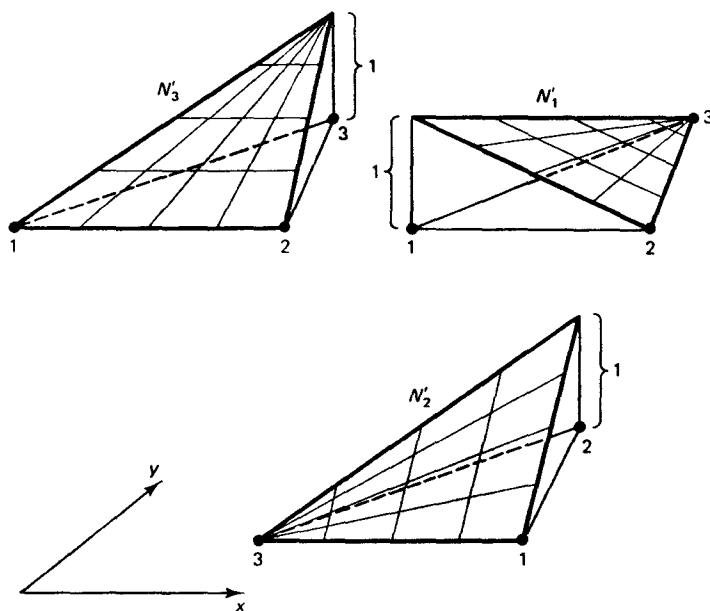


Figure 3.4.2 Element shape functions for the linear triangle.

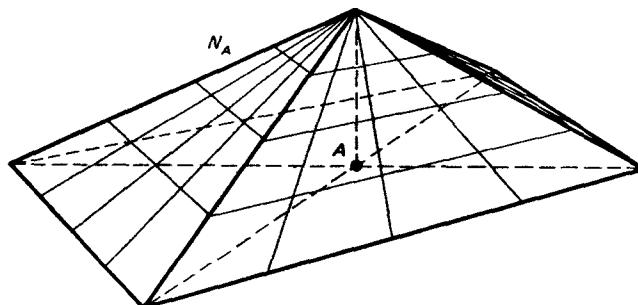


Figure 3.4.3 Global shape function constructed from linear triangular element shape functions.

(Side 3, 1)

$$u^h(-1, \eta) = \frac{1}{2}(1 + \eta)d_1^e + \frac{1}{2}(1 - \eta)d_2^e \quad (3.4.6)$$

As is clear from (3.4.4)–(3.4.6), the behavior is linear along each edge. As this is the same for all contiguous elements, continuity is assured. We may also freely mix linear triangles and bilinear quadrilaterals in the same mesh and achieve continuity, since their edge behavior is identical.

The completeness condition C3 is automatic since we have employed the isoparametric concept in defining (3.4.3).

Exercise 1. Compute the Jacobian determinant at $\xi = \eta = 0$ for the element shown in Fig. 3.4.4. Plot the result as a function of $x_1^e \in [-2, 2]$. Comment.

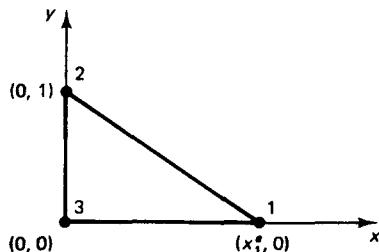


Figure 3.4.4

Summary. The linear triangular element may be obtained from the bilinear quadrilateral by *degeneration*. Computer implementation may be performed along similar lines once the bilinear quadrilateral is programmed.

3.5 TRILINEAR HEXAHEDRAL ELEMENT

The trilinear hexahedral element is the basic element for three-dimensional analysis. It is a straightforward generalization of the bilinear quadrilateral presented in Sec. 3.2. The domain Ω^e (see Fig. 3.5.1) is the image of the biunit cube in ξ -space under the trilinear mapping

$$x(\xi) = \alpha_0 + \alpha_1\xi + \alpha_2\eta + \alpha_3\zeta + \alpha_4\xi\eta + \alpha_5\eta\zeta + \alpha_6\xi\zeta + \alpha_7\xi\eta\zeta \quad (3.5.1)$$

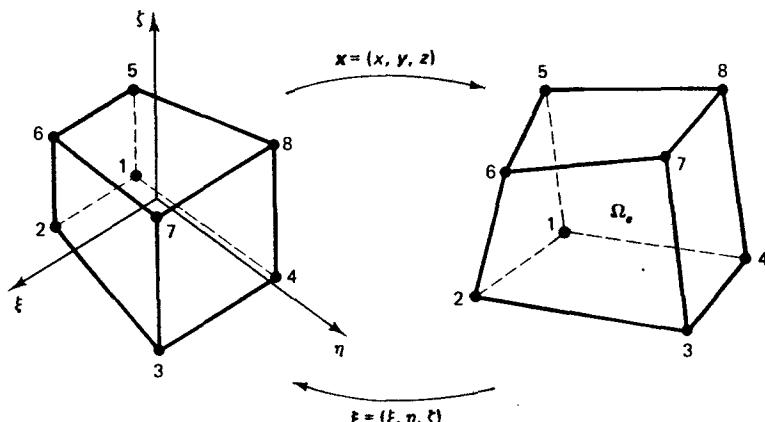


Figure 3.5.1 Trilinear hexahedral element domain and local node ordering.

with corresponding expressions for $y(\xi)$ and $z(\xi)$. The coefficients $\alpha_0, \dots, \alpha_7$ are determined by the conditions

$$x(\xi_a) = x_a^\epsilon, \quad a = 1, \dots, 8 \quad (3.5.2)$$

(The nodal coordinates in ξ -space are defined in Table 3.5.1.) This gives rise to a system of linear algebraic equations, viz.,

$$\begin{bmatrix} 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 \end{bmatrix} \begin{Bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \end{Bmatrix} = \begin{Bmatrix} x_1^\epsilon \\ x_2^\epsilon \\ x_3^\epsilon \\ x_4^\epsilon \\ x_5^\epsilon \\ x_6^\epsilon \\ x_7^\epsilon \\ x_8^\epsilon \end{Bmatrix} \quad (3.5.3)$$

TABLE 3.5.1 Nodal coordinates in ξ -space

| a | ξ_a | η_a | ζ_a |
|-----|---------|----------|-----------|
| 1 | -1 | -1 | -1 |
| 2 | 1 | -1 | -1 |
| 3 | 1 | 1 | -1 |
| 4 | -1 | 1 | -1 |
| 5 | -1 | -1 | -1 |
| 6 | 1 | -1 | 1 |
| 7 | 1 | 1 | 1 |
| 8 | -1 | 1 | 1 |

The matrix may be inverted with the aid of the computer. Solving for the α 's and substituting in (3.5.3) yields

$$x(\xi) = \sum_{a=1}^8 N_a(\xi) x_a^\epsilon \quad (3.5.4)$$

where

$$N_a(\xi, \eta, \zeta) = \frac{1}{8}(1 + \xi_a \xi)(1 + \eta_a \eta)(1 + \zeta_a \zeta) \quad (3.5.5)$$

with similar expressions for $y(\xi)$ and $z(\xi)$. Observe that the right-hand side of (3.5.5) consists of the product of the one-dimensional shape functions derived in Sec. 1.12.

We define u^h by invoking the isoparametric concept, i.e.,

$$u^h(\xi) = \sum_{a=1}^8 N_a(\xi) d_a^e \quad (3.5.6)$$

The continuity of the interpolation functions may be seen by observing the behavior of (3.5.6) along a face of Ω' . For example, if $\zeta = -1$, (3.5.6) becomes

$$u^h(\xi, \eta, -1) = \sum_{a=1}^4 \frac{1}{4}(1 + \xi_a \xi)(1 + \eta_a \eta) d_a^e \quad (3.5.7)$$

As can be seen from (3.5.7), u^h has hyperbolic paraboloidal variation on the face $\zeta = -1$ and is uniquely defined by the four nodal values associated with that face. The same conclusion can be drawn for any other element of this type which shares this face, and so the continuity of u^h is assured. If the element domain, Ω' , is not too distorted, the shape functions will be smooth functions of x . The completeness condition is guaranteed by virtue of the isoparametric hypothesis.

The hexahedron, or “brick” as it is often referred to, is the most generally useful shape in modeling three-dimensional domains. However, it is often convenient to have a wedge-shaped element at our disposal. The technique of degeneration may be employed to derive a wedge-shaped element from the hexahedron. Specifically, we coalesce nodes 3 and 4, and 7 and 8 (see Fig. 3.5.2). In this case (3.5.4) degenerates to

$$x = \sum_{a=1}^3 N'_a x_a^e + \sum_{a=5}^7 N'_a x_a^e \quad (3.5.8)$$

where

$$N'_a = \begin{cases} N_a & a = 1, 2, 5, 6 \\ N_a + N_{a+1} & a = 3, 7 \end{cases} \quad (3.5.9)$$

The Jacobian determinant becomes zero along the edge connecting nodes 3 and 7, but the x -derivatives of the shape functions are well behaved (cf. Sec. 3.4).

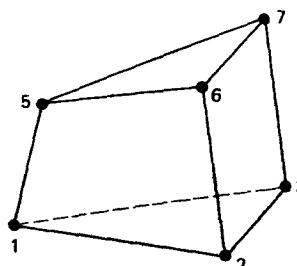


Figure 3.5.2 Wedge-shaped element formed by degenerating the eight-node hexahedral element.

A further coalescing of nodes 5, 6 and 7 yields the well-known linear tetrahedral element (see Fig. 3.5.3). Equation (3.5.8) becomes

$$x = \sum_{a=1}^3 N'_a x_a' + N''_5 x_5' \quad (3.5.10)$$

where

$$N''_5 = N'_5 + N'_6 + N'_7 \quad (3.5.11)$$

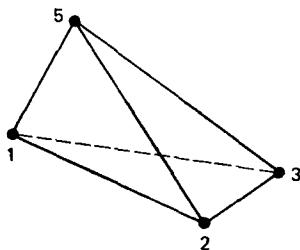


Figure 3.5.3 The linear tetrahedral element formed by degenerating the wedge-shaped element.

It can be shown that the x -derivatives of the shape functions are constants in this case (cf. Sec. 3.4). The linear tetrahedron was first proposed by Gallagher et al. [11].

3.6 HIGHER-ORDER ELEMENTS; LAGRANGE POLYNOMIALS

Thus far we have considered elements for which the shape functions are constructed from products of the linear one-dimensional shape functions. In this section we begin consideration of *higher-order elements*. These elements are often capable of more accurate representations than the *linear* elements considered previously and also allow more faithful representations of element domains, as the boundary edges and surfaces may be curved. At the same time they are generally more expensive than the basic linear elements. Thus the cost-effectiveness of the various elements is often in dispute. It appears that the optimal choice is very problem-dependent, so no single element is to be preferred exclusively.

Systematic derivation of certain higher-order isoparametric elements may be performed with the aid of the one-dimensional Lagrange polynomials. We first present the definition of the Lagrange polynomials and then, by way of examples, indicate the construction for multidimensional elements.

Lagrange Polynomials

The Lagrange polynomials are defined by

Order of the polynomial

$$\begin{aligned}
 l_a^{n_{en}-1}(\xi) &= \frac{\prod_{\substack{b=1 \\ b \neq a}}^{n_{en}} (\xi - \xi_b)}{\prod_{\substack{b=1 \\ b \neq a}}^{n_{en}} (\xi_a - \xi_b)} \\
 &= \frac{(\xi - \xi_1)(\xi - \xi_2) \cdots (\xi - \xi_{a-1}) \underset{a\text{-term omitted}}{\cancel{(\xi - \xi_a)}} \cdots (\xi - \xi_{n_{en}})}{(\xi_a - \xi_1)(\xi_a - \xi_2) \cdots (\xi_a - \xi_{a-1}) \underset{a\text{-term omitted}}{\cancel{(\xi_a - \xi_a)}} \cdots (\xi_a - \xi_{n_{en}})}
 \end{aligned}
 \tag{3.6.1}$$

If the order of the polynomial is not important we omit the superscript and simply write l_a . It is simple to see from (3.6.1) that $l_a(\xi_a) = 1$ and, if $b \neq a$, $l_a(\xi_b) = 0$. In other words

$$l_a(\xi_b) = \delta_{ab} \tag{3.6.2}$$

We shall define the shape functions of an n_{en} -noded element in one dimension by the relation

$$N_a = l_a^{n_{en}-1} \tag{3.6.3}$$

The ξ_a 's, as usual, denote the locations of the nodes in ξ -space. Equation (3.6.2) guarantees satisfaction of the interpolation property.

Example 1

Consider the two-noded, one-dimensional element presented in Sec. 1.12. We shall derive the shape functions by way of (3.6.3). From (3.6.3), we have that

$$l_1^1(\xi) = \frac{(\xi - \xi_2)}{(\xi_1 - \xi_2)} = \frac{(\xi - 1)}{-1 - (+1)} = \frac{1}{2}(1 - \xi) \tag{3.6.4}$$

$$l_2^1(\xi) = \frac{(\xi - \xi_1)}{(\xi_2 - \xi_1)} = \frac{(\xi + 1)}{1 - (-1)} = \frac{1}{2}(1 + \xi) \tag{3.6.5}$$

As the reader may see, (3.6.4) and (3.6.5) are the familiar shape functions of the one-dimensional, linear element.

Example 2

We shall now use (3.6.3) to derive the shape functions for a *quadratic* three-node element. We assume that the nodes are located at -1 , 0 , and $+1$ in ξ -space (see Fig.

3.6.1). The shape functions are given by

$$N_1(\xi) = l_1^2(\xi) = \frac{(\xi - \xi_2)(\xi - \xi_3)}{(\xi_1 - \xi_2)(\xi_1 - \xi_3)} = \frac{\xi(\xi - 1)}{(-1)(-2)} = \frac{1}{2}\xi(\xi - 1) \quad (3.6.6)$$

$$N_2(\xi) = l_2^2(\xi) = \frac{(\xi - \xi_1)(\xi - \xi_3)}{(\xi_2 - \xi_1)(\xi_2 - \xi_3)} = \frac{(\xi + 1)(\xi - 1)}{(1)(-1)} = 1 - \xi^2 \quad (3.6.7)$$

$$N_3(\xi) = l_3^2(\xi) = \frac{(\xi - \xi_1)(\xi - \xi_2)}{(\xi_3 - \xi_1)(\xi_3 - \xi_2)} = \frac{(\xi + 1)\xi}{(2)(1)} = \frac{1}{2}\xi(\xi + 1) \quad (3.6.8)$$

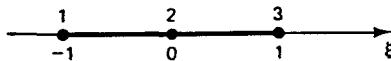


Figure 3.6.1 Node locations in ξ -space for the quadratic, three-node element.

These functions are depicted in Fig. 3.6.2.

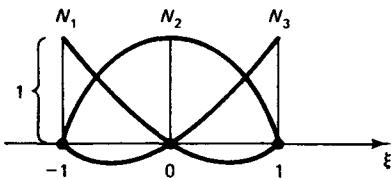


Figure 3.6.2 Shape functions for the quadratic, three-node element.

Exercise 1. Employ (3.6.1) to obtain the shape functions for the *cubic* four-noded (i.e., $n_{en} = 4$) element shown in Fig. 3.6.3. Sketch the results.



Figure 3.6.3 Node locations in ξ -space for the cubic, four-node element.

Example 3

The shape functions for the bilinear quadrilateral may be set up by taking products of the first order, Lagrange polynomials. The formula is

$$N_a(\xi, \eta) = l_b^1(\xi)l_c^1(\eta) \quad (3.6.9)$$

where the relationship among the indices is given in Table 3.6.1.

TABLE 3.6.1

| <i>a</i> | <i>b</i> | <i>c</i> |
|----------|----------|----------|
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 2 | 2 |
| 4 | 1 | 2 |

The procedure is schematically illustrated in Fig. 3.6.4.

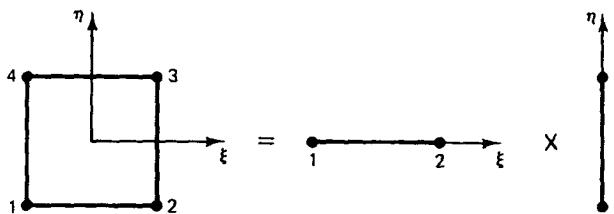


Figure 3.6.4 Schematic relationship between the linear, one-dimensional shape functions and the bilinear, two-dimensional shape functions.

Explicating (3.6.9), we have that

$$N_1(\xi, \eta) = l_1^1(\xi)l_1^1(\eta) = \frac{1}{4}(1 - \xi)(1 - \eta) \quad (3.6.10)$$

$$N_2(\xi, \eta) = l_2^1(\xi)l_1^1(\eta) = \frac{1}{4}(1 + \xi)(1 - \eta) \quad (3.6.11)$$

$$N_3(\xi, \eta) = l_1^1(\xi)l_2^1(\eta) = \frac{1}{4}(1 - \xi)(1 + \eta) \quad (3.6.12)$$

$$N_4(\xi, \eta) = l_2^1(\xi)l_2^1(\eta) = \frac{1}{4}(1 + \xi)(1 + \eta) \quad (3.6.13)$$

An analogous procedure may be employed to obtain the trilinear, three-dimensional shape functions of the previous section.

Example 4

Higher-order, two- and three-dimensional elements may be derived by taking products of Lagrange polynomials. We shall illustrate this idea by setting up the nine-node, two-dimensional Lagrange element. The element shape functions are the products of quadratic Lagrange polynomials. The setup is illustrated in Fig. 3.6.5. Note that the element domain Ω^e may have curved edges. The formula for the shape functions is

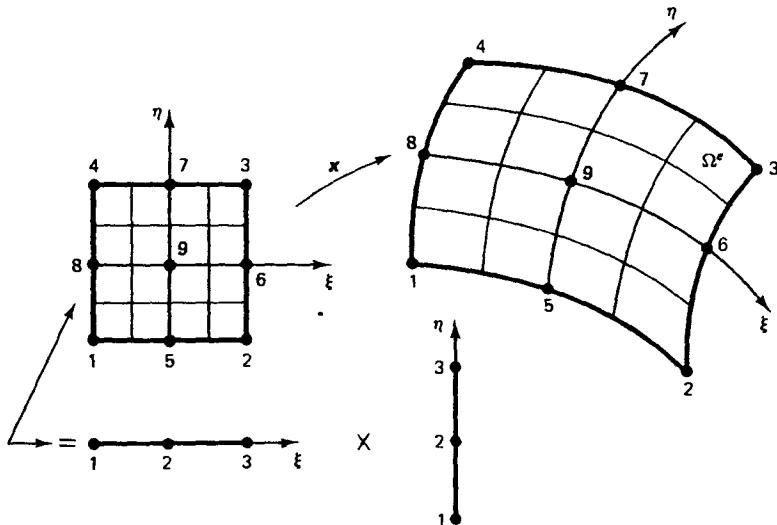


Figure 3.6.5 Nine-node, two-dimensional Lagrange element.

$$N_a(\xi, \eta) = l_b^2(\xi)l_c^2(\eta) \quad (3.6.14)$$

where the relationship among the indices is given in Table 3.6.2.

TABLE 3.6.2

| <i>a</i> | <i>b</i> | <i>c</i> |
|----------|----------|----------|
| 1 | 1 | 1 |
| 2 | 3 | 1 |
| 3 | 3 | 3 |
| 4 | 1 | 3 |
| 5 | 2 | 1 |
| 6 | 3 | 2 |
| 7 | 2 | 3 |
| 8 | 1 | 2 |
| 9 | 2 | 2 |

Typical amongst (3.6.14) are

$$\begin{pmatrix} \text{Corner} \\ \text{node} \end{pmatrix} \quad N_1(\xi, \eta) = l_1^2(\xi)l_1^2(\eta) = \frac{1}{4}\xi\eta(\xi - 1)(\eta - 1) \quad (3.6.15)$$

$$\begin{pmatrix} \text{Midside} \\ \text{node} \end{pmatrix} \quad N_5(\xi, \eta) = l_2^2(\xi)l_1^2(\eta) = \frac{1}{2}\eta(1 - \xi^2)(\eta - 1) \quad (3.6.16)$$

$$\begin{pmatrix} \text{Middle} \\ \text{node} \end{pmatrix} \quad N_9(\xi, \eta) = l_2^2(\xi)l_2^2(\eta) = (1 - \xi^2)(1 - \eta^2) \quad (3.6.17)$$

These functions are shown in Fig. 3.6.6; N_9 is frequently referred to as the *bubble function*, for obvious reasons.

Remark

Elements for which the shape functions are formed from products of one-dimensional Lagrange polynomials are called *Lagrange elements*.

Exercise 2. Use the results of Exercise 1 to derive the shape functions for the 16-node, two-dimensional Lagrange element.

Exercise 3. Use the quadratic one-dimensional shape functions to derive the shape functions for the triquadratic 27-node three-dimensional element.

Exercise 4. Use linear and quadratic shape functions to derive shape functions for the six-node quadrilateral depicted in Fig. 3.6.7.

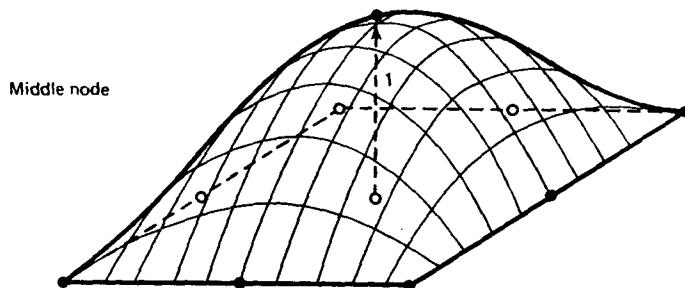
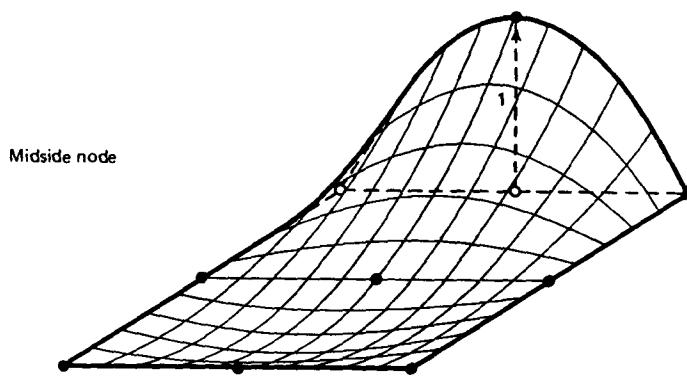
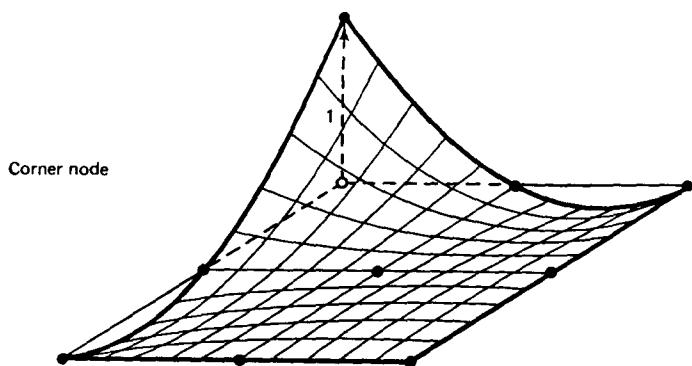


Figure 3.6.6 Typical Lagrange shape functions for the nine-node element.

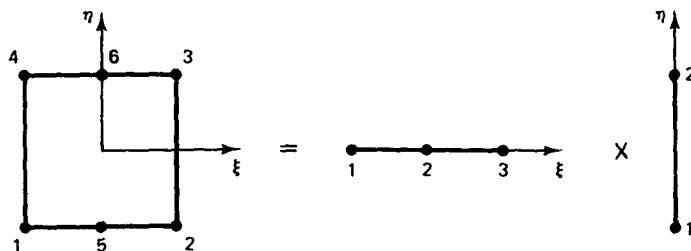


Figure 3.6.7

3.7 ELEMENTS WITH VARIABLE NUMBERS OF NODES

In this section we shall derive the shape functions of a two-dimensional "quadrilateral" element, which possesses four through nine nodes. When all nine nodes are present, the element is the Lagrange element considered in the previous section. When only four nodes are present, the element degenerates to the basic bilinear quadrilateral of Sec. 3.2. Any of nodes 5 through 9 (see Fig. 3.6.5) may be added or omitted. Thus either linear or quadratic behavior may be accommodated along any edge, which permits the "mixing" of basic linear and higher-order elements in one mesh.

We begin with the bilinear shape functions of the four-node quadrilateral element:

$$N_a(\xi, \eta) = \frac{1}{4}(1 + \xi_a \xi)(1 + \eta_a \eta), \quad a = 1, 2, 3, 4 \quad (3.7.1)$$

Our first objective is to add the fifth node, permitting a curved edge and quadratic behavior along the edge connecting nodes 1 and 2 (see Fig. 3.7.1). Note that N_1 and N_2 do not vanish at node 5 (i.e., $N_a(\xi_5, \eta_5) = \frac{1}{2}$, $a = 1, 2$), an essential requirement of the shape functions of the five-node element. Shape functions 3 and 4 are satisfactory in this regard. Thus shape functions 1 and 2 will need to be modified. Let us first introduce a shape function for node 5. Let

$$\begin{aligned} N_5(\xi, \eta) &= \underbrace{l_2^2(\xi)}_{\substack{\text{Middle-node} \\ \text{quadratic} \\ \text{Lagrange} \\ \text{polynomial}}} \underbrace{l_1^1(\eta)}_{\substack{\text{Left-node} \\ \text{linear} \\ \text{Lagrange} \\ \text{polynomial}}} \\ &= \frac{1}{2}(1 - \xi^2)(1 - \eta) \end{aligned} \quad (3.7.2)$$

Note that

$$N_5(\xi_a, \eta_a) = \delta_{a5}, \quad a = 1, 2, \dots, 5 \quad (3.7.3)$$

This shape function is illustrated in Fig. 3.7.2 and satisfies all requisite properties. In

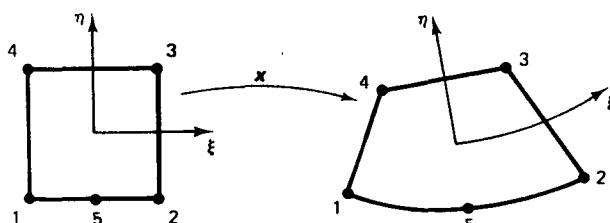


Figure 3.7.1 Five-node element.

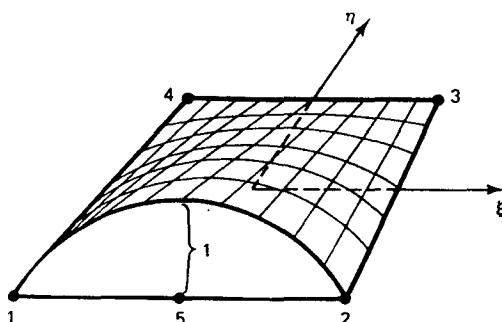


Figure 3.7.2

addition, we may employ it to correct the behavior of bilinear shape functions at node 5. To do this we subtract a multiple of N_5 from N_1 and N_2 so that the resulting functions vanish identically at node 5. Specifically, we define the new shape functions

$$N_a = \frac{1}{2}N_5, \quad a = 1, 2 \quad (3.7.4)$$

It is easily seen that

$$N_a(\xi_b, \eta_b) - \frac{1}{2}N_5(\xi_b, \eta_b) = \delta_{ab}, \quad a = 1, 2, \dots, b = 1, 2, \dots, 5 \quad (3.7.5)$$

and so (3.7.4) are appropriate shape functions for nodes 1 and 2 of the five-node element. These functions are illustrated in Fig. 3.7.3. In summary, the shape functions of the five-node element are given by (3.7.2), (3.7.4), and the original bilinear shape functions for nodes 3 and 4. Observe that in addition to introducing N_5 , only the bilinear shape functions associated with the nodes along the edge containing node 5 needed to be modified. A similar situation occurs if we wish to add any of the other midside nodes. We may construct shape functions for midside nodes 6 through 8 in analogous fashion to the construction of N_5 (see (3.7.2)). These are

$$N_6(\xi, \eta) = \frac{1}{2}(1 - \eta^2)(1 + \xi) \quad (3.7.6)$$

$$N_7(\xi, \eta) = \frac{1}{2}(1 - \xi^2)(1 + \eta) \quad (3.7.7)$$

$$N_8(\xi, \eta) = \frac{1}{2}(1 - \eta^2)(1 - \xi) \quad (3.7.8)$$

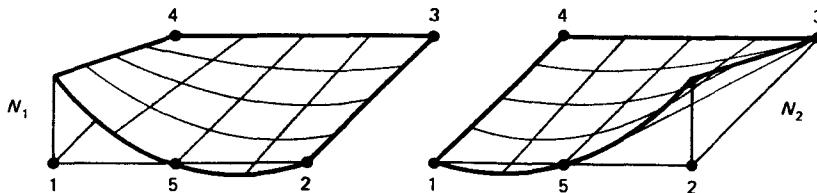


Figure 3.7.3

If any of nodes 5, 6, 7 and 8 are absent, we may formally define N_5 , N_6 , N_7 , and N_8 to be identically zero, respectively. Then the modified shape functions for nodes 1 through 4 are

$$N_1 \leftarrow N_1 - \frac{1}{2}(N_5 + N_8) \quad (3.7.9)$$

$$N_2 \leftarrow N_2 - \frac{1}{2}(N_5 + N_6) \quad (3.7.10)$$

$$N_3 \leftarrow N_3 - \frac{1}{2}(N_6 + N_7) \quad (3.7.11)$$

$$N_4 \leftarrow N_4 - \frac{1}{2}(N_7 + N_8) \quad (3.7.12)$$

The reader may wish to verify that the modified shape functions, (3.7.9) through (3.7.12), satisfy

$$N_a(\xi_b, \eta_b) = \delta_{ab} \quad (3.7.13)$$

for all b corresponding to nodes present.

To include node 9 we introduce the *bubble function*:

$$N_9(\xi, \eta) = (1 - \xi^2)(1 - \eta^2) \quad (3.7.14)$$

N_9 vanishes at nodes 1 through 8. However, N_1, N_2, \dots, N_8 do not in general vanish at node 9, and therefore they must be modified. The procedure is carried out most expeditiously if the original bilinear shape functions and N_5, N_6, N_7, N_8 are first modified to account for the presence of node 9. This is accomplished as follows:

$a = 1, 2, 3, 4$:

$$N_a \leftarrow \underbrace{N_a}_{\text{Bilinear}} - \frac{1}{4}N_9 \quad (3.7.15)$$

shape functions; (3.7.1)

$a = 5, 6, 7, 8$:

Defined by (3.7.2), (3.7.6)–(3.7.8)

$$N_a = \begin{cases} \overbrace{N_a}^{N_a - \frac{1}{2}N_9} & \text{(if node } a \text{ is present)} \\ 0 & \text{(if node } a \text{ is absent)} \end{cases} \quad (3.7.16)$$

Following this modification, the corrections indicated in (3.7.9) through (3.7.12) need be performed. The procedure is most succinctly summarized in a flowchart; see Fig. 3.7.4.

Care must also be taken that high-order elements are not too distorted. For example, in the case of the eight-node serendipity quadrilateral (see Fig. 3.7.5), all interior angles must be less than 180° (as for the four-node quadrilateral) and, in addition, the midside nodes should not migrate too far off the center of the sides. It is recommended in [12], p. 186, that the safe zone is confined to the middle third of the side. Although rules of thumb are useful for initiatory considerations, in general, we need rely on numerical checks of the sign of the Jacobian determinant to determine if the element geometry is “too distorted.”

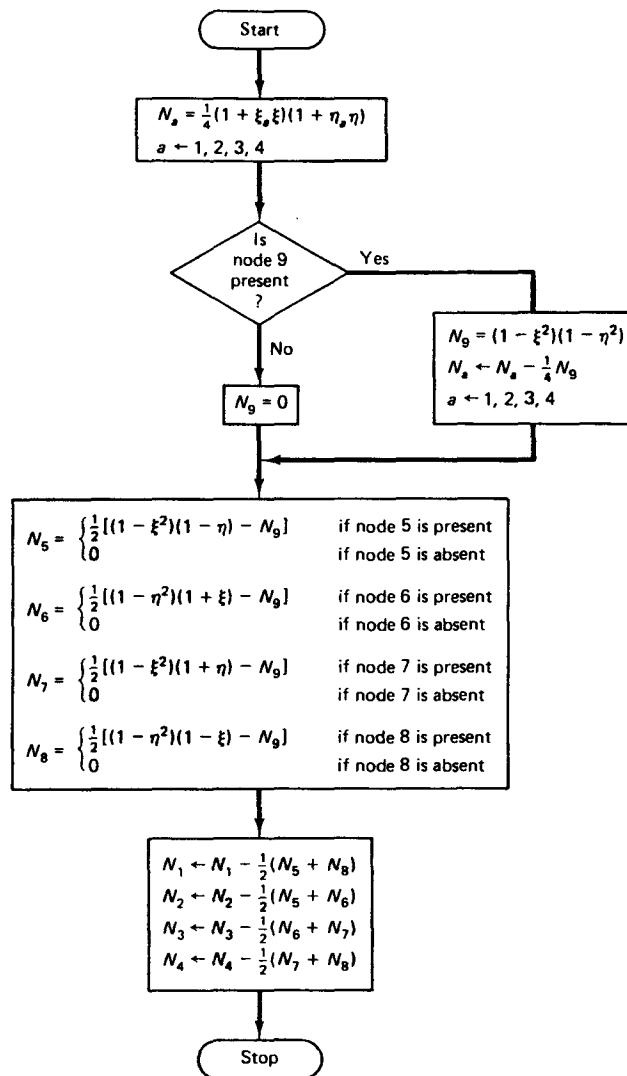


Figure 3.7.4 Calculation of a shape function for a four- through nine-node element.

Exercise 1. Use the flowchart in Fig. 3.7.4 to derive explicitly the shape functions of the eight-node *serendipity quadrilateral* for which the internal node (node 9) is omitted.

Exercise 2. Use the results of Exercise 1 and the degeneration technique to derive the shape functions of the six-node quadratic triangle; see Fig. 3.7.5.

Exercise 3. Generalize the flowchart of Fig. 3.7.4 so that the four- through nine-node element may be degenerated to a triangle.

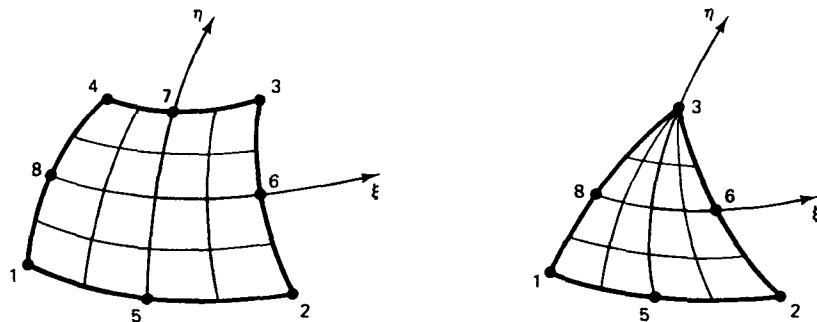


Figure 3.7.5 Degeneration of the eight-node serendipity element to the six-node triangle.

Exercise 4. Develop a flowchart analogous to the one in Fig. 3.7.4 for an 8- through 27-variable-number-of-nodes three-dimensional “brick” element. The node numbering is indicated in Fig. 3.7.6.

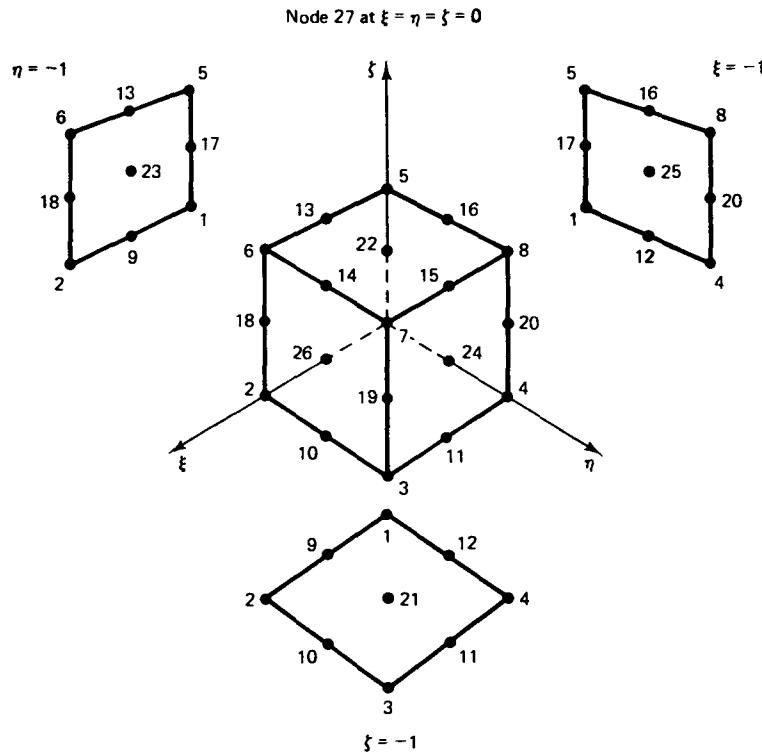
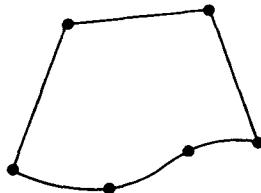


Figure 3.7.6 Nodal numbering for the 8- through 27-variable-number-of-nodes element.

Exercise 5. Generalize the flowchart of Exercise 4 so that the 8- through 27-node element may be degenerated to a wedge-shaped element.

Exercise 6. Develop a flowchart for a 4- through 16-variable-number-of-nodes quadrilateral element. (The 16-node element is the bicubic Lagrange element of Exercise 2, Sec. 3.6.)

Exercise 7. Develop shape functions for a six-node quadrilateral, which exhibits cubic behavior along one edge and linear behavior along the other three (see the accompanying figure).



Standard Element Families

In Fig. 3.7.7, some members of standard two-dimensional element families are shown. (Brick, wedge, and tetrahedral three-dimension analogs may easily be envisioned.) Isoparametric shape functions for all these elements may easily be derived by the techniques described in this section.

We may note from Fig. 3.7.7 that the serendipity elements have nodeless interiors. Beyond cubic level, the serendipity elements may be enhanced by the inclusion of some internal nodes to achieve higher degrees of polynomial completeness. This may be understood by considering the *Pascal triangles* shown in Fig. 3.7.8. As may be seen, for quartic and higher-order serendipity elements, the degree of polynomial completeness is fixed at cubic. The degree of completeness is generally considered to be the most important measure of accuracy of an element. Thus it is seen to be necessary to include functions associated with internal nodes in order to achieve higher-order accurate elements.

3.8 NUMERICAL INTEGRATION; GAUSSIAN QUADRATURE

Let $f : \Omega^e \subset \mathbb{R}^{n_{sd}} \rightarrow \mathbb{R}$ be a given function. We are interested in computing

$$\int_{\Omega^e} f(x) d\Omega \quad (3.8.1)$$

for purposes of constructing element arrays. (f may be thought of as any term in an integrand that we have encountered so far. For example, $f = \mathbf{B}_a^T \mathbf{D} \mathbf{B}_b$, the integrand of the stiffness matrix in heat conduction; see Sec. 2.5.) Throughout we shall assume f is smooth and integrable, so there is no ambiguity as to the meaning of (3.8.1). We

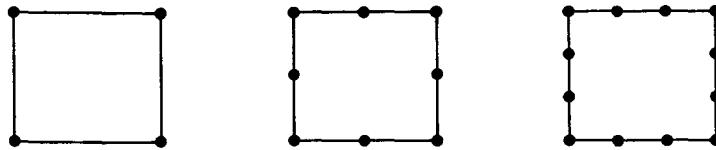
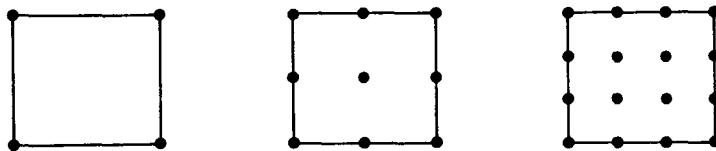
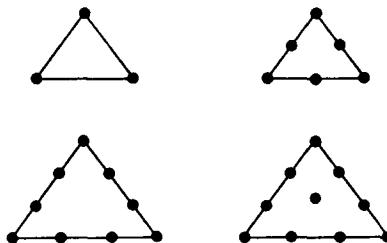
Serendipity family of quadrilateral elementsLagrange family of quadrilateral elementsStandard triangular elements

Figure 3.7.7 Standard two-dimensional element families. Note that the first members of the Lagrange and serendipity families are identical.

will need to make use of the change of variables formula. In each case we will pull back the integral over the element domain in x -space to one over the parent domain (i.e., the biunit n_{sd} -cube). For completeness, we begin by recalling the one-dimensional case (see Sec. 1.15):

$$n_{sd} = 1:$$

$$\int_{\Omega^e} f(x) dx = \int_{-1}^1 f(x(\xi)) x_{,\xi}(\xi) d\xi \quad (3.8.2)$$

The two- and three-dimensional cases are given as follows:

$$n_{sd} = 2:$$

$$\int_{\Omega^e} f(x, y) d\Omega = \int_{-1}^1 \int_{-1}^1 f(x(\xi, \eta), y(\xi, \eta)) j(\xi, \eta) d\xi d\eta \quad (3.8.3)$$

Lagrange quadrilaterals

$$\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & & \xi & \eta & & & \\
 & & \xi^2 & & \xi\eta & & \eta^2 & & \\
 & & \xi^3 & \xi^2\eta & \xi\eta^2 & & \eta^3 & & \\
 & & \xi^4 & \xi^3\eta & \xi^2\eta^2 & \xi\eta^3 & & \eta^4 & \\
 & \ddots & \xi^4\eta & \xi^3\eta^2 & \xi^2\eta^3 & \xi\eta^4 & & \ddots & \\
 \xi^m & & \xi^4\eta^2 & \xi^3\eta^3 & \xi^2\eta^4 & & & & \eta^m \\
 \xi^m\eta & & \xi^4\eta^3 & \xi^4\eta^4 & \xi^3\eta^4 & & & & \xi\eta^m \\
 \xi^m\eta^2 & & & & & & & & \\
 \xi^m\eta^3 & & & & & & \xi^3\eta^m & & \\
 \xi^m\eta^4 & & & & & & \xi^4\eta^m & & \\
 & & & & & & \ddots & & \\
 & & & & & & \xi^m\eta^m & &
 \end{array}$$

Serendipity quadrilaterals

$$\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & & \xi & \eta & & & \\
 & & \xi^2 & & \xi\eta & & \eta^2 & & \\
 & & \xi^3 & \xi^2\eta & \xi\eta^2 & & \eta^3 & & \\
 & & \xi^4 & \xi^3\eta & \xi\eta^3 & \xi\eta^4 & & \ddots & \\
 & \ddots & \xi^4\eta & & & & & & \\
 \xi^m & \ddots & & & & & & & \eta^m \\
 \xi^m\eta & & & & & & & & \xi\eta^m \\
 & & & & & & & &
 \end{array}$$

Triangles (internal nodal functions included)

$$\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & & \xi & \eta & & & \\
 & & \xi^2 & & \xi\eta & & \eta^2 & & \\
 & & \xi^3 & \xi^2\eta & \xi\eta^2 & & \eta^3 & & \\
 & & \xi^4 & \xi^3\eta & \xi^2\eta^2 & \xi\eta^3 & & \eta^4 & \\
 & \ddots & \ddots & \ddots & \ddots & & \ddots & & \\
 \xi^m & \xi^{m-1}\eta & \xi^{m-2}\eta^2 & \dots & \xi^2\eta^{m-2} & \xi\eta^{m-1} & & \eta^m &
 \end{array}$$

Figure 3.7.8. Pascal triangles for standard two-dimensional element families.

$n_{sd} = 3$:

$$\int_{\Omega^e} f(x, y, z) d\Omega$$

$$= \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 f(x(\xi, \eta, \zeta), y(\xi, \eta, \zeta), z(\xi, \eta, \zeta)) j(\xi, \eta, \zeta) d\xi d\eta d\zeta$$

(3.8.4)

Recall that $j = \det(\partial x / \partial \xi)$, the Jacobian determinant.

In each case we have to evaluate an integral of the form:

$$\underbrace{\int_{-1}^1 \cdots \int_{-1}^1}_{n_{sd} \text{ times}} g(\xi, \dots) \underbrace{d\xi \cdots}_{n_{sd} \text{ differentials}} \quad (3.8.5)$$

Consider the one-dimensional case, in which we wish to evaluate

$$\int_{-1}^1 g(\xi) d\xi \quad (3.8.6)$$

This can be approximately computed by way of a numerical integration (quadrature) formula as follows:

$$\int_{-1}^1 g(\xi) d\xi = \sum_{l=1}^{n_{int}} g(\tilde{\xi}_l) W_l + R \cong \sum_{l=1}^{n_{int}} g(\tilde{\xi}_l) W_l \quad (3.8.7)$$

where n_{int} is the number of integration (quadrature) points, $\tilde{\xi}_l$ is the coordinate of the l th integration point, W_l is the “weight” of the l th integration point, and R is the remainder. The reader is probably familiar with some of the classical numerical integration formulas, such as the trapezoidal and Simpson’s rules. We review these now as examples.

Example 1 (Trapezoidal rule)

$$n_{int} = 2$$

$$\tilde{\xi}_1 = -1$$

$$\tilde{\xi}_2 = 1$$

$$W_l = 1, \quad l = 1, 2$$

$$R = -\frac{2}{3} g_{,\xi\xi}(\bar{\xi})$$

$\bar{\xi}$ denotes some point in the interval $[-1, 1]$. The trapezoidal rule is exact for constants

and linear polynomials but only approximate for quadratic and higher-order polynomials. It is said to be “second-order accurate.”

Example 2 (Simpson’s rule)

$$n_{\text{int}} = 3$$

$$\tilde{\xi}_1 = -1$$

$$\tilde{\xi}_2 = 0$$

$$\tilde{\xi}_3 = 1$$

$$W_1 = W_3 = \frac{1}{3}$$

$$W_2 = \frac{4}{3}$$

$$R = \frac{-g^{(4)}(\tilde{\xi})}{90}$$

where $g^{(4)} = g_{xxxx}$. Simpson’s rule integrates general cubic polynomials exactly and is thus said to be “fourth-order accurate.”

Although the above rules are widely used they are inefficient in the sense that there exist integration rules that are just as accurate but involve fewer integration points. *This issue is of great importance in practice since the fewer the integration points, the less the cost.* Considerable savings can be engendered by choosing an appropriate numerical integration rule.

Gaussian Quadrature

In one dimension the Gauss quadrature formulas are optimal. Accuracy of order $2n_{\text{int}}$ is achieved by n_{int} integration points. The locations of the quadrature points and values of associated weights are determined to attain maximum accuracy. The theory is thoroughly discussed in [13]. We give the first three Gaussian rules in the following:

Gaussian quadrature rules.

1. $n_{\text{int}} = 1$

$$\tilde{\xi}_1 = 0$$

$$W_1 = 2$$

$$R = \frac{g_{xx}(\tilde{\xi})}{3}$$

2. $n_{\text{int}} = 2$

$$\tilde{\xi}_1 = \frac{-1}{\sqrt{3}}$$

$$\tilde{\xi}_2 = \frac{1}{\sqrt{3}}$$

$$W_1 = W_2 = 1$$

$$R = \frac{g^{(4)}(\tilde{\xi})}{135}$$

3. $n_{int} = 3$

$$\tilde{\xi}_1 = -\sqrt{\frac{3}{5}}$$

$$\tilde{\xi}_2 = 0$$

$$\tilde{\xi}_3 = \sqrt{\frac{3}{5}}$$

$$W_1 = W_3 = \frac{5}{9}$$

$$W_2 = \frac{8}{9}$$

$$R = \frac{g^{(6)}(\tilde{\xi})}{15,750}$$

Exercise 1 Verify the accuracy of the preceding Gaussian rules by integrating the monomials $1, \xi, \xi^2, \dots$ in turn.

The derivation of Gaussian quadrature formulas is illustrated by the following example.

Example 3 (Derivation of the Gaussian quadrature formula for $n_{int} = 2$)

We seek a rule involving two integration points that is exact for a general cubic polynomial. Let $g(\xi) = \alpha_0 + \alpha_1 \xi + \alpha_2 \xi^2 + \alpha_3 \xi^3$, where the α 's are arbitrary constants, and assume the integration points are equally weighted and symmetrically spaced (i.e., $W_1 = W_2$ and $\tilde{\xi}_1 = -\tilde{\xi}_2$, respectively). The exact integral of

$$\int_{-1}^1 g(\xi) d\xi = 2\alpha_0 + \frac{2}{3}\alpha_2 \quad (3.8.8)$$

and this is to be equal to

$$\sum_{l=1}^2 g(\tilde{\xi}_l) W_l = 2W_2(\alpha_0 + \alpha_2 \tilde{\xi}_2^2) \quad (3.8.9)$$

This is to hold for arbitrary values of α_0 and α_2 , and it thus follows that $W_2 = 1$ and $\tilde{\xi}_2 = 1/\sqrt{3}$.

Exercise 2. Derive the Gauss quadrature rule for $n_{\text{int}} = 3$. Hint: From considerations of symmetry, assume $\tilde{\xi}_1 = -\tilde{\xi}_3$, $\tilde{\xi}_2 = 0$ and $W_1 = W_3$.

General Gaussian Quadrature Rule

The general Gaussian quadrature rule is defined by the following:

$$W_l = \frac{2}{[(1 - \tilde{\xi}_l^2)(P'_{n_{\text{int}}}(\tilde{\xi}_l)^2)]}, \quad 1 \leq l \leq n_{\text{int}} \quad (3.8.10)$$

$$R = \frac{2^{2n_{\text{int}}+1}(n_{\text{int}}!)^4}{(2n_{\text{int}} + 1)[(2n_{\text{int}})!]^3} g \underbrace{,\xi \dots \xi}_{2n_{\text{int}} \text{ times}}(\tilde{\xi}) \quad (3.8.11)$$

where $\tilde{\xi}_l$ is the l th zero of the Legendre polynomial $P_{n_{\text{int}}}(\xi)$, and $P'_{n_{\text{int}}}$ denotes the derivative of $P_{n_{\text{int}}}$. The Legendre polynomials are defined by

$$P_{n_{\text{int}}}(\xi) = \frac{1}{2^{n_{\text{int}}} n_{\text{int}}!} \frac{d^{n_{\text{int}}}}{d\xi^{n_{\text{int}}}} (\xi^2 - 1)^{n_{\text{int}}} \quad (3.8.12)$$

See [13] for tabulated values of $\tilde{\xi}_l$ and W_l .

Gaussian Quadrature in Several Dimensions

Gaussian rules for integrals in several dimensions are constructed by employing one-dimensional Gaussian rules on each coordinate separately. For example, in two dimensions

$$\begin{aligned} \int_{-1}^1 \int_{-1}^1 g(\xi, \eta) d\xi d\eta &\cong \int_{-1}^1 \left\{ \sum_{l^{(1)}=1}^{n_{\text{int}}^{(1)}} g(\tilde{\xi}_{l^{(1)}}, \eta) W_{l^{(1)}}^{(1)} \right\} d\eta && \text{(Gaussian quadrature} \\ &\cong \sum_{l^{(1)}=1}^{n_{\text{int}}^{(1)}} \sum_{l^{(2)}=1}^{n_{\text{int}}^{(2)}} g(\tilde{\xi}_{l^{(1)}}, \tilde{\eta}_{l^{(2)}}) W_{l^{(1)}}^{(1)} W_{l^{(2)}}^{(2)} && \text{rule 1 applied to} \\ &&& \text{\xi-coordinate}) \\ &&& \text{(Gaussian quadrature} \\ &&& \text{rule 2 applied to} \\ &&& \eta\text{-coordinate}) \end{aligned}$$

(3.8.13)

Example 4

If the one-point rule is used in each direction (3.8.13) specializes to

$$\int_{-1}^1 \int_{-1}^1 g(\xi, \eta) d\xi d\eta = 4g(0, 0) \quad (3.8.14)$$

Example 5

If the two-point rule is used in each direction, (3.8.13) specializes to

$$\int_{-1}^1 \int_{-1}^1 g(\xi, \eta) d\xi d\eta = g\left(\frac{-1}{\sqrt{3}}, \frac{-1}{\sqrt{3}}\right) + g\left(\frac{1}{\sqrt{3}}, \frac{-1}{\sqrt{3}}\right) \\ + g\left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right) + g\left(\frac{-1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right) \quad (3.8.15)$$

It is frequently preferable to tabulate multidimensional rules in terms of a single index so that in place of (3.8.13) we would have

$$\int_{-1}^1 \int_{-1}^1 g(\xi, \eta) d\xi d\eta \approx \sum_{l=1}^{n_{\text{int}}} g(\tilde{\xi}_l, \tilde{\eta}_l) W_l \quad (3.8.16)$$

Comparison with (3.8.13) indicates that

$$n_{\text{int}} = n_{\text{int}}^{(1)} n_{\text{int}}^{(2)}, \quad 1 \leq l \leq n_{\text{int}} \quad (3.8.17)$$

$$\tilde{\xi}_l = \tilde{\xi}_{l^{(1)}}^{(1)} \quad (3.8.18)$$

$$\tilde{\eta}_l = \tilde{\eta}_{l^{(2)}}^{(2)} \quad (3.8.19)$$

$$W_l = W_{l^{(1)}}^{(1)} W_{l^{(2)}}^{(2)} \quad (3.8.20)$$

where a tabular relationship among the indices must be established (e.g., see Table 3.8.1)

Example 6

If the one-point rule is used in each direction, corresponding to (3.8.17) through (3.8.20), we have

$$n_{\text{int}} = 1, \quad \tilde{\xi}_1 = \tilde{\eta}_1 = 0, \quad W_1 = 2 \cdot 2 = 4 \quad (3.8.21)$$

Thus (3.8.14), (3.8.16), and (3.8.21) are consistent.

Example 7

Consider use of the two-point rule in each direction. We assume the relationship among the indices is as given in Table 3.8.1.

TABLE 3.8.1

| l | $l^{(1)}$ | $l^{(2)}$ |
|-----|-----------|-----------|
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 2 | 2 |
| 4 | 1 | 2 |

In this case (3.8.17) through (3.8.20) yield

$$n_{\text{int}} = 2 \cdot 2 = 4 \quad (3.8.22)$$

$$\tilde{\xi}_1 = \frac{-1}{\sqrt{3}}, \quad \tilde{\xi}_2 = \frac{1}{\sqrt{3}}, \quad \tilde{\xi}_3 = \frac{1}{\sqrt{3}}, \quad \tilde{\xi}_4 = \frac{-1}{\sqrt{3}} \quad (3.8.23)$$

$$\tilde{\eta}_1 = \frac{-1}{\sqrt{3}}, \quad \tilde{\eta}_2 = \frac{1}{\sqrt{3}}, \quad \tilde{\eta}_3 = \frac{1}{\sqrt{3}}, \quad \tilde{\eta}_4 = \frac{-1}{\sqrt{3}} \quad (3.8.24)$$

$$W_1 = W_2 = W_3 = W_4 = 1 \quad (3.8.25)$$

Clearly (3.8.15), (3.8.16), and (3.8.22) through (3.8.25) are consistent.

Exercise 3. Consider use of the three-point Gauss rule in each direction. Define a relationship between the indices l , $l^{(1)}$, and $l^{(2)}$ by constructing a table similar to Table 3.8.1. Determine $\tilde{\xi}_l$, $\tilde{\eta}_l$, and W_l , $1 \leq l \leq 9$.

Locations of the quadrature points for the one-point, 2×2 , and 3×3 Gaussian rules are illustrated in Fig. 3.8.1.

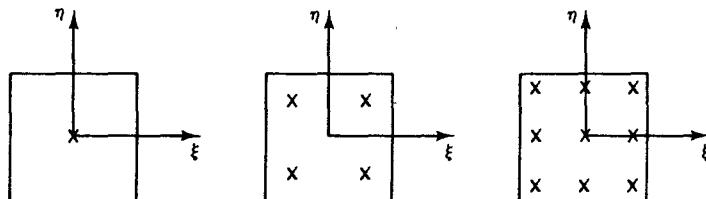


Figure 3.8.1 Locations of Gauss quadrature points for one-point, 2×2 , and 3×3 rules.

Analogous arguments lead to the form of rules in three dimensions, viz.,

$$\begin{aligned}
 & \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 g(\xi, \eta, \zeta) d\xi d\eta d\zeta \\
 & \cong \sum_{l^{(1)}=1}^{n_{\text{int}}^{(1)}} \sum_{l^{(2)}=1}^{n_{\text{int}}^{(2)}} \sum_{l^{(3)}=1}^{n_{\text{int}}^{(3)}} g(\tilde{\xi}_{l^{(1)}}, \tilde{\eta}_{l^{(2)}}, \tilde{\zeta}_{l^{(3)}}) W_{l^{(1)}}^{(1)} W_{l^{(2)}}^{(2)} W_{l^{(3)}}^{(3)} \\
 & = \sum_{l=1}^{n_{\text{int}}} g(\tilde{\xi}_l, \tilde{\eta}_l, \tilde{\zeta}_l) W_l
 \end{aligned} \quad (3.8.26)$$

Exercise 4. Determine $\tilde{\xi}_l$, $\tilde{\eta}_l$, $\tilde{\zeta}_l$, and W_l for the one-point and $2 \times 2 \times 2$ Gaussian rules in three dimensions.

In multidimensional cases the Gaussian quadrature rules are no longer necessarily optimal. For example, in three dimensions the six-point (non-Gaussian) rule given in Table 3.8.2 is fourth-order accurate [14]. The quadrature points are located at the centers of the six faces of the cube. (This means that all monomials of the form

$\xi^i \eta^j \zeta^k$, $0 \leq i, j, k \leq 3$, $i + j + k \leq 3$, are exactly integrated.) The fourth-order Gaussian rule requires eight points ($2 \times 2 \times 2$ rule) and is thus more expensive to use.

TABLE 3.8.2 Fourth-Order Accurate, Six-Point Quadrature Rule in Three Dimensions

| l | $\tilde{\xi}_l$ | $\tilde{\eta}_l$ | $\tilde{\zeta}_l$ | W_l |
|-----|-----------------|------------------|-------------------|-------|
| 1 | 1 | 0 | 0 | 4/3 |
| 2 | -1 | 0 | 0 | 4/3 |
| 3 | 0 | 1 | 0 | 4/3 |
| 4 | 0 | -1 | 0 | 4/3 |
| 5 | 0 | 0 | 1 | 4/3 |
| 6 | 0 | 0 | -1 | 4/3 |

A 14-point rule for three dimensions that is sixth-order accurate is described in [15, 16]. This rule represents a considerable savings when compared with the 27-point ($3 \times 3 \times 3$) Gaussian rule [17].

Unfortunately, a general theory of optimal integration formulas for even such simple shapes as squares and cubes does not yet seem to be known. Despite some measure of inefficiency in multidimensional applications, Gaussian quadrature formulas are still widely used and will be sufficient for our present needs. Recent progress toward the development of improved rules is reported in [18].

Exercise 5. Consider the following four-point rule in two dimensions:

| l | $\tilde{\xi}_l$ | $\tilde{\eta}_l$ | W_l |
|-----|-----------------|------------------|-------|
| 1 | 0 | + a | 1 |
| 2 | 0 | - a | 1 |
| 3 | + a | 0 | 1 |
| 4 | - a | 0 | 1 |

Determine a such that fourth-order accuracy is attained.

3.9 DERIVATIVES OF SHAPE FUNCTIONS AND SHAPE FUNCTION SUBROUTINES

We need to calculate explicitly the derivatives of the shape functions to construct the element stiffness matrices. For simplicity let us take the case $n_{sd} = 2$. The derivatives

of N_a with respect to x and y may be evaluated with the aid of the chain rule:

$$N_{a,x} = N_{a,\xi}\xi_{,x} + N_{a,\eta}\eta_{,x} \quad (3.9.1)$$

$$N_{a,y} = N_{a,\xi}\xi_{,y} + N_{a,\eta}\eta_{,y} \quad (3.9.2)$$

It is worthwhile to recast these relations in the following matrix form:

$$\langle N_{a,x} N_{a,y} \rangle = \langle N_{a,\xi} N_{a,\eta} \rangle \begin{bmatrix} \xi_{,x} & \xi_{,y} \\ \eta_{,x} & \eta_{,y} \end{bmatrix} \quad (3.9.3)$$

The derivatives $N_{a,\xi}$ and $N_{a,\eta}$ may be explicitly computed. However, the terms in the matrix cannot be directly computed since we do not have explicit expressions $\xi = \xi(x, y)$ and $\eta = \eta(x, y)$. On the other hand, we do have the inverse relations

$$x(\xi, \eta) = \sum_{a=1}^{n_{en}} N_a(\xi, \eta)x_a^\epsilon \quad (3.9.4)$$

$$y(\xi, \eta) = \sum_{a=1}^{n_{en}} N_a(\xi, \eta)y_a^\epsilon \quad (3.9.5)$$

which enables us to calculate the matrix

$$x_{,\xi} = \begin{bmatrix} x_{,\xi} & x_{,\eta} \\ y_{,\xi} & y_{,\eta} \end{bmatrix} \quad (3.9.6)$$

The components are:

$$x_{,\xi} = \sum_{a=1}^{n_{en}} N_{a,\xi}x_a^\epsilon, \quad x_{,\eta} = \sum_{a=1}^{n_{en}} N_{a,\eta}x_a^\epsilon \quad (3.9.7)$$

$$y_{,\xi} = \sum_{a=1}^{n_{en}} N_{a,\xi}y_a^\epsilon, \quad y_{,\eta} = \sum_{a=1}^{n_{en}} N_{a,\eta}y_a^\epsilon \quad (3.9.8)$$

The matrix (3.9.6) is the inverse of the matrix in (3.9.3), i.e.,

$$\begin{bmatrix} \xi_{,x} & \xi_{,y} \\ \eta_{,x} & \eta_{,y} \end{bmatrix} = (x_{,\xi})^{-1} = \frac{1}{j} \begin{bmatrix} y_{,\eta} & -x_{,\eta} \\ -y_{,\xi} & x_{,\xi} \end{bmatrix} \quad (3.9.9)$$

where

$$j = \det(x_{,\xi}) = x_{,\xi}y_{,\eta} - x_{,\eta}y_{,\xi} \quad (3.9.10)$$

All the preceding relations are typically evaluated in **shape function subroutines**. It is convenient to segregate the calculations into those which can be performed once for

all elements of the same type (i.e., an *element group*) and those which need to be repeated for each element.

For the element group, calculate integration-rule weights, shape functions, and local derivatives:

For $l = 1, \dots, n_{\text{int}}$

Determine: $W_l, \tilde{\xi}_l, \tilde{\eta}_l$

For $a = 1, \dots, n_{\text{en}}$

Calculate: $N_a(\tilde{\xi}_l, \tilde{\eta}_l), N_{a,\xi}(\tilde{\xi}_l, \tilde{\eta}_l), N_{a,\eta}(\tilde{\xi}_l, \tilde{\eta}_l)$

Subroutine QDCSHL performs these calculations for the four-node bilinear quadrilateral element in program DLEARN (see Chapter 11).

Given the e th element's coordinates (x_a^e, y_a^e where $1 \leq a \leq n_{\text{en}}$), calculate global derivatives of the shape functions and Jacobian determinants:

For $l = 1, \dots, n_{\text{int}}$

Calculate:

$$x_{,\xi}(\tilde{\xi}_l, \tilde{\eta}_l) = \sum_{a=1}^{n_{\text{en}}} N_{a,\xi}(\tilde{\xi}_l, \tilde{\eta}_l) x_a^e$$

$$x_{,\eta}(\tilde{\xi}_l, \tilde{\eta}_l) = \sum_{a=1}^{n_{\text{en}}} N_{a,\eta}(\tilde{\xi}_l, \tilde{\eta}_l) x_a^e$$

$$y_{,\xi}(\tilde{\xi}_l, \tilde{\eta}_l) = \sum_{a=1}^{n_{\text{en}}} N_{a,\xi}(\tilde{\xi}_l, \tilde{\eta}_l) y_a^e$$

$$y_{,\eta}(\tilde{\xi}_l, \tilde{\eta}_l) = \sum_{a=1}^{n_{\text{en}}} N_{a,\eta}(\tilde{\xi}_l, \tilde{\eta}_l) y_a^e$$

$$j(\tilde{\xi}_l, \tilde{\eta}_l) = x_{,\xi}(\tilde{\xi}_l, \tilde{\eta}_l) y_{,\eta}(\tilde{\xi}_l, \tilde{\eta}_l) - x_{,\eta}(\tilde{\xi}_l, \tilde{\eta}_l) y_{,\xi}(\tilde{\xi}_l, \tilde{\eta}_l)$$

For $a = 1, \dots, n_{\text{en}}$

Calculate:

$$N_{a,x}(\tilde{\xi}_l, \tilde{\eta}_l) = \frac{N_{a,\xi}(\tilde{\xi}_l, \tilde{\eta}_l) y_{,\eta}(\tilde{\xi}_l, \tilde{\eta}_l) - N_{a,\eta}(\tilde{\xi}_l, \tilde{\eta}_l) y_{,\xi}(\tilde{\xi}_l, \tilde{\eta}_l)}{j(\tilde{\xi}_l, \tilde{\eta}_l)}$$

$$N_{a,y}(\tilde{\xi}_l, \tilde{\eta}_l) = \frac{-[N_{a,\xi}(\tilde{\xi}_l, \tilde{\eta}_l) x_{,\eta}(\tilde{\xi}_l, \tilde{\eta}_l) - N_{a,\eta}(\tilde{\xi}_l, \tilde{\eta}_l) x_{,\xi}(\tilde{\xi}_l, \tilde{\eta}_l)]}{j(\tilde{\xi}_l, \tilde{\eta}_l)}$$

In obtaining the formulas for $N_{a,x}$ and $N_{a,y}$, we have combined (3.9.3) and (3.9.9). Subroutine QDCSHG performs these calculations in DLEARN for both the four-node bilinear quadrilateral and three-node linear triangle (using the “degeneration” technique of Sec. 3.4). The logical variable QUAD is used to indicate whether the element under consideration is a quadrilateral or triangle:

QUAD = .TRUE. (quadrilateral)

QUAD = .FALSE. (triangle)

Table 3.9.1 translates the text notation into the FORTRAN names used in QDCSHL and QDCSHG. The reader should study these routines carefully.

TABLE 3.9.1 Notation for Shape Function Subroutines QDCSHL and QDCSHG

| Notation | FORTRAN name | Array dimensions |
|--|--------------|--------------------------------------|
| n_{int} | NINT (= 4) | |
| n_{sd} | NSD (= 2) | |
| n_{en} | NEN (= 4) | |
| W_l | W(L) | NINT |
| $\tilde{\xi}_l$ | R | |
| $\tilde{\eta}_l$ | S | |
| $N_a(\tilde{\xi}_l, \tilde{\eta}_l)$ | SHL(3,I,L) | |
| $N_{a,\xi}(\tilde{\xi}_l, \tilde{\eta}_l)$ | SHL(1,I,L) | |
| $N_{a,\eta}(\tilde{\xi}_l, \tilde{\eta}_l)$ | SHL(2,I,L) | NROWSH × NEN × NINT (= 3 × 4 × 4) |
| x_a^e | XL(1,I) | |
| y_a^e | XL(2,I) | NSD × NEN |
| $N_a(\tilde{\xi}_l, \tilde{\eta}_l)$ | SHG(3,I,L) | |
| $N_{a,\xi}(\tilde{\xi}_l, \tilde{\eta}_l)$ | SHG(1,I,L) | |
| $N_{a,\eta}(\tilde{\xi}_l, \tilde{\eta}_l)$ | SHG(2,I,L) | NROWSH × NEN × NINT |
| $\frac{\xi_a}{2}$ | RA(I) | NEN |
| $\frac{\eta_a}{2}$ | SA(I) | NEN |
| $\begin{bmatrix} x_{,\xi} & x_{,\eta} \\ y_{,\xi} & y_{,\eta} \end{bmatrix}$ | XS(I,J) | NSD × NSD |
| $j(\tilde{\xi}_l, \tilde{\eta}_l)$ | DET(L) | NINT |

The case $n_{sd} = 3$ proceeds analogously. The main results are summarized as follows:

$$\text{cof}_{11} = y_{,\eta} z_{,\xi} - y_{,\xi} z_{,\eta} \quad (3.9.11)$$

$$\text{cof}_{12} = y_{,\xi} z_{,\xi} - y_{,\xi} z_{,\eta} \quad (3.9.12)$$

$$\text{cof}_{13} = y_{,\xi}z_{,\eta} - y_{,\eta}z_{,\xi} \quad (3.9.13)$$

$$\text{cof}_{21} = z_{,\eta}x_{,\xi} - z_{,\xi}x_{,\eta} \quad (3.9.14)$$

$$\text{cof}_{22} = z_{,\xi}x_{,\xi} - z_{,\eta}x_{,\eta} \quad (3.9.15)$$

$$\text{cof}_{23} = z_{,\xi}x_{,\eta} - z_{,\eta}x_{,\xi} \quad (3.9.16)$$

$$\text{cof}_{31} = x_{,\eta}y_{,\xi} - x_{,\xi}y_{,\eta} \quad (3.9.17)$$

$$\text{cof}_{32} = x_{,\xi}y_{,\xi} - x_{,\eta}y_{,\eta} \quad (3.9.18)$$

$$\text{cof}_{33} = x_{,\xi}y_{,\eta} - x_{,\eta}y_{,\xi} \quad (3.9.19)$$

$$j = x_{,\xi}\text{cof}_{11} + x_{,\eta}\text{cof}_{12} + x_{,\xi}\text{cof}_{13} \quad (3.9.20)$$

$$N_{a,x} = \frac{N_{a,\xi}\text{cof}_{11} + N_{a,\eta}\text{cof}_{12} + N_{a,\xi}\text{cof}_{13}}{j} \quad (3.9.21)$$

$$N_{a,y} = \frac{N_{a,\xi}\text{cof}_{21} + N_{a,\eta}\text{cof}_{22} + N_{a,\xi}\text{cof}_{23}}{j} \quad (3.9.22)$$

$$N_{a,z} = \frac{N_{a,\xi}\text{cof}_{31} + N_{a,\eta}\text{cof}_{32} + N_{a,\xi}\text{cof}_{33}}{j} \quad (3.9.23)$$

The cof_{ij} 's are the cofactors of the matrix $x_{,\xi}$. (Recall the definition of matrix inverse: $(x_{,\xi})^{-1} = (\text{cof})^T/j$.)

Exercise 1. Program a shape function subroutine for the eight-node trilinear brick. (To do this it will be helpful first to generalize explicitly the notation table (Table 3.9.1) to the three-dimensional case. Mimic the style of routines QDCSHL and QDCSHG.)

Exercise 2. Program shape function subroutines for the four- through nine-node element described in Fig. 3.7.4. Assume the IEN array is made available to the routines through the argument list and that if $\text{IEN}(a) = 0$, then node a is absent for the element under consideration. The IEN array may be taken to have dimension 9 within the routines. (Note that all operations performed to calculate the shape functions need also be performed to calculate the derivatives.)

Exercise 3. Generalize the subroutine of Exercise 1 to allow for the possibility of degeneration to a wedge-shaped element. Assume that the logical variable BRICK is used where

BRICK = .TRUE. (brick)

BRICK = .FALSE. (wedge)

The reader may set many additional shape-function-subroutine exercises by considering the different possibilities discussed in previous sections of this chapter.

3.10 ELEMENT STIFFNESS FORMULATION

Programming an element stiffness matrix is an essential step in learning the finite element method. There are several ways to go about this. In this section we describe three of the most important implementational styles of stiffness coding. Each has attributes and it is recommended that the reader fully understand each of these procedures.

Implementation 1. The first implementation employs the following form of the stiffness:

$$\mathbf{k}^e = \int_{\Omega^e} \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega = \int_{\square} \mathbf{B}^T \mathbf{D} \mathbf{B} j d\square \quad (3.10.1)$$

Applying numerical quadrature to (3.10.1) enables us to write

$$\mathbf{k}^e \cong \sum_{l=1}^{n_{int}} (\mathbf{B}^T \mathbf{D} \mathbf{B} j) \Big|_{\tilde{\xi}_l} W_l \quad (3.10.2)$$

It simplifies the coding to combine j and W_l with \mathbf{D} . In this case we write

$$\widetilde{\mathbf{D}} = j(\tilde{\xi}_l) W_l \mathbf{D} \quad (3.10.3)$$

and thus

$$\mathbf{k}^e \cong \sum_{l=1}^{n_{int}} (\mathbf{B}^T \widetilde{\mathbf{D}} \mathbf{B})_l \quad (3.10.4)$$

In the actual FORTRAN programming $\widetilde{\mathbf{D}}$ is written over \mathbf{D} . Thus no additional storage for $\widetilde{\mathbf{D}}$ is required. This implementation is used in subroutines QDCK (quadrilateral continuum \mathbf{k}^e) and TRUSK (truss \mathbf{k}^e) in DLEARN (see Chapter 11). The main steps are summarized next.

For $l = 1, \dots, n_{int}$

Set up the strain-displacement matrix \mathbf{B} .

Set up the constitutive matrix $\widetilde{\mathbf{D}}$.

Multiply $\widetilde{\mathbf{D}} * \mathbf{B}$.

Multiply $\mathbf{B}^T * (\widetilde{\mathbf{D}} \mathbf{B})$, taking account of symmetry, and accumulate in \mathbf{k}^e .

Table 3.10.1 translates the text notation into the FORTRAN names used in QDCK and TRUSK (see also Table 3.9.1 for names of shape function variables).

TABLE 3.10.1 Notation for Element Stiffness Routines QDCK and TRUSK

| Notation | FORTRAN name | Array dimensions |
|--------------|--------------|--------------------------|
| n_{en} | NEE | |
| B | B | NROWB ^a × NEE |
| D | C | NROWB × NROWB |
| \tilde{D} | DMAT | NROWB × NROWB |
| \tilde{DB} | DB | NROWB × NEE |
| k' | ELSTIF | NEE × NEE |

^aNROWB is a variable used to dimension the arrays. In subroutine QDCK, NROWB = 4. This dimension is needed, for example, in the axisymmetric case. In plane stress, however, only the first three rows are needed. To save calculations, another variable is introduced, namely, NSTR (3 in plane stress; 4 in the axisymmetric case), which is used to define the limits of the matrix multiplication loops. To appreciate this point, the reader should examine the calling statement to subroutine MULTAB in QDCK and subroutine MULTAB itself.

Exercise 1. Determine formulas for the numbers of multiplications and additions required to form an element stiffness by the above procedure.

Implementation 2. In the opinion of the author the method of coding just described is the most important because it is completely general and does not assume any special structure of B or D . However, if special structure is present, then greater efficiency can be gained. For example, if there are zero entries in B and/or D , then operations can be saved in the multiplications $\tilde{D} * B$ and $B^T * (\tilde{DB})$. This requires programming special subroutines for these calculations.

Example 1

Consider two-dimensional, isotropic, plane-stress elasticity theory. The matrices B and D take the form:

$$B = [B_1, \dots, B_{n_{en}}] \quad (3.10.5)$$

$$B_a = \begin{bmatrix} N_{a,x} & 0 \\ 0 & N_{a,y} \\ N_{a,y} & N_{a,x} \end{bmatrix}, \quad 1 \leq a \leq n_{en} \quad (3.10.6)$$

$$D = \begin{bmatrix} D_{11} & D_{12} & 0 \\ D_{21} & D_{22} & 0 \\ \text{symmetric} & & D_{33} \end{bmatrix} \quad (3.10.7)$$

Recall that k' can be written in terms of $n_{en} \times n_{en}$ nodal submatrices, k'_{ab} , where (see (2.9.6) through (2.9.9))

$$\underbrace{k_{ab}^e}_{n_{ed} \times n_{ed}} = \int_{\Omega^e} \mathbf{B}_a^T \mathbf{D} \mathbf{B}_b \, d\Omega \quad (3.10.8)$$

We can write (3.10.8) in numerical-integration form as follows:

$$k_{ab}^e \approx \sum_{l=1}^{n_{int}} (\mathbf{B}_a^T \tilde{\mathbf{D}} \mathbf{B}_b)_l \quad (3.10.9)$$

The nodal submatrix k_{ab}^e has dimension 2×2 in two dimensions, viz.,

$$k_{ab}^e = \begin{bmatrix} k_{2a-1, 2b-1} & k_{2a-1, 2b} \\ k_{2a, 2b-1} & k_{2a, 2b} \end{bmatrix} \quad (3.10.10)$$

In the computer implementation, k^e may be computed by calculating the nodal submatrices, namely, (3.10.10), for each a and b such that $1 \leq a, b \leq n_{en}$. Only the nodal submatrices on or above the diagonal need be computed because the lower part is determined by symmetry. The procedure is given by the algorithm in the box.

For $l = 1, \dots, n_{int}$

Set up $\tilde{\mathbf{D}}$.

For $b = 1, \dots, n_{en}$

Let $B_1 = N_{b,x}$ and $B_2 = N_{b,y}$.

Multiply $\tilde{\mathbf{D}} * B_b$, taking account of zeros in $\tilde{\mathbf{D}}$ and B_b :

$$\tilde{\mathbf{D}} \mathbf{B}_b = \begin{bmatrix} \tilde{\mathbf{D}} B_{11} & \tilde{\mathbf{D}} B_{12} \\ \tilde{\mathbf{D}} B_{21} & \tilde{\mathbf{D}} B_{22} \\ \tilde{\mathbf{D}} B_{31} & \tilde{\mathbf{D}} B_{32} \end{bmatrix} = \begin{bmatrix} \tilde{D}_{11} B_1 & \tilde{D}_{12} B_2 \\ \tilde{D}_{12} B_1 & \tilde{D}_{22} B_2 \\ \tilde{D}_{33} B_2 & \tilde{D}_{33} B_1 \end{bmatrix}$$

For $a = 1, \dots, b$

Let $B_1 = N_{a,x}$ and $B_2 = N_{a,y}$.

Multiply $B_a^T * (\tilde{\mathbf{D}} \mathbf{B}_b)$, taking account of zeros in B_a , and accumulate in k^e :

$$\begin{bmatrix} k_{2a-1, 2b-1} & k_{2a-1, 2b} \\ k_{2a, 2b-1} & k_{2a, 2b} \end{bmatrix} \leftarrow \begin{bmatrix} k_{2a-1, 2b-1} & k_{2a-1, 2b} \\ k_{2a, 2b-1} & k_{2a, 2b} \end{bmatrix} + \begin{bmatrix} (B_1 \tilde{D} B_{11} + B_2 \tilde{D} B_{31}) & (B_1 \tilde{D} B_{12} + B_2 \tilde{D} B_{32}) \\ (B_2 \tilde{D} B_{21} + B_1 \tilde{D} B_{31}) & (B_2 \tilde{D} B_{22} + B_1 \tilde{D} B_{32}) \end{bmatrix}$$

Exercise 2. Determine the numbers of multiplications and additions required to form a four-node element stiffness by the boxed procedure. Specialize the results of Exercise 1

to the present case and compare. (The savings in three dimensions are even more impressive!)

Implementation 3. This implementation was first proposed by Gupta and Mohraz [19]. It emanates from the following development. Consider two- and three-dimensional elasticity theory (see Secs. 2.7 through 2.11). Note that for these cases $n_{ed} = n_{sd}$ (i.e., the number of element degrees of freedom equals the number of space dimensions). Let

$$w_i^h = \sum_{a=1}^{n_{ed}} c_{ia} N_a, \quad u_i^h = \sum_{a=1}^{n_{ed}} d_{ia} N_a \quad (3.10.11)$$

Recall the equivalent single-index representations of the coefficients,

$$c_p = c_{ia}, \quad d_q = d_{jb} \quad (3.10.12)$$

where

$$p = n_{ed}(a - 1) + i, \quad q = n_{ed}(b - 1) + j \quad (3.10.13)$$

With these we may write²

$$\begin{aligned} \int_{\Omega'} w_{(i,k)}^h c_{ikl} u_{(j,l)}^h d\Omega &= \int_{\Omega'} w_{i,k}^h c_{ikl} u_{j,l}^h d\Omega \quad (\text{by the minor symmetries of the } c_{ijkl} \text{'s and Lemma 2 of Sec. 2.7}) \\ &= \sum_{a,b=1}^{n_{ed}} c_{ia} \left(\int_{\Omega'} N_{a,k} c_{ikl} N_{b,l} d\Omega \right) d_{jb} \\ &= \sum_{a,b=1}^{n_{ed}} c_{ia} k_{iajb}^e d_{jb} = \sum_{p,q=1}^{n_{ed}} c_p k_{pq}^e d_q \end{aligned} \quad (3.10.14)$$

from which it follows that

$$k_{pq}^e = k_{iajb}^e = \int_{\Omega'} N_{a,k} c_{ikl} N_{b,l} d\Omega \quad (3.10.15)$$

It is convenient here to work with the four-indexed version of the element stiffness, namely, k_{iajb}^e . In FORTRAN, the two-indexed version and four-indexed version are stored in identical fashion in consecutive addresses in memory if the indices are arranged in the order given. This enables us to deal with the array ELSTIF (i.e., k^e) as a two-dimensional array in one routine and as a four-dimensional array in another.

| Two-dimensional version | Four-dimensional version |
|----------------------------|--------------------------------------|
| DIMENSION ELSTIF(NEE, NEE) | DIMENSION ELSTIF(NED, NEN, NED, NEN) |
| ELSTIF(IP, JQ) | ELSTIF(I, IA, J, JB) |
| where | |
| IP = NED*(IA - 1) + I | |
| JQ = NED*(JB - 1) + J | (3.10.16) |

²Recall that the summation over repeated spatial indices (i.e., i, j, k, l) is in force.

Assume that the material occupying Ω^e is *isotropic*, that is,

$$c_{ijkl} = \mu(\delta_{ij}\delta_{kl} + \delta_{il}\delta_{kj}) + \lambda\delta_{ik}\delta_{jl} \quad (3.10.17)$$

and *homogeneous*, that is, λ and μ are constants. With these assumptions, (3.10.15) can be written as:

$$\begin{aligned} k_{iabj}^e &= (\mu(\delta_{ij}\delta_{kl} + \delta_{il}\delta_{kj}) + \lambda\delta_{ik}\delta_{jl}) \int_{\Omega^e} N_{a,k}N_{b,l} d\Omega \\ &= \mu \left(\delta_{ij} \int_{\Omega^e} N_{a,k}N_{b,k} d\Omega + \int_{\Omega^e} N_{a,j}N_{b,i} d\Omega \right) \\ &\quad + \lambda \int_{\Omega^e} N_{a,i}N_{b,j} d\Omega \end{aligned} \quad (3.10.18)$$

Thus to calculate the element stiffness, the expression

$$\int_{\Omega^e} N_{a,i}N_{b,j} d\Omega \cong \sum_{l=1}^{n_{int}} (N_{a,i}N_{b,j}) \Big|_{\tilde{\xi}_l} W_l \quad (3.10.19)$$

must be evaluated first. Observe that (3.10.19) is symmetric under the interchange of indices $(i, a) \leftrightarrow (j, b)$. This fact reduces the number of calculations involved. The algorithm in the following box calculates the element stiffness based upon (3.10.18) and (3.10.19). Note that the integrals, (3.10.19), are first stored in k_{iabj}^e , which is then overwritten by the stiffness. This procedure leads to additional savings compared with Implementation 2.

```

 $c_1 := \lambda + \mu$ 
 $c_2 := \mu$ 
 $c_3 := \lambda$ 
For  $l = 1, \dots, n_{int}$ 
  const =  $j(\tilde{\xi}_l)W_l$ 
  For  $b = 1, \dots, n_{en}$ 
    For  $j = 1, \dots, n_{ed}$ 
      temp = const  $\cdot N_{b,j}(\tilde{\xi}_l)$ 
      For  $a = 1, \dots, b$ 
        For  $i = 1, \dots, j$ 
           $k_{iabj}^e \leftarrow k_{iabj}^e + \text{temp} \cdot N_{a,i}(\tilde{\xi}_l)$ 

```

```

For  $b = 1, \dots, n_{en}$ 
  For  $a = 1, \dots, b$ 
    temp =  $\sum_{k=1}^{n_{ed}} k_{akb}^e$ 

```

```

For  $j = 1, \dots, n_{ed}$ 
  For  $i = 1, \dots, j$ 
    if  $i = j$ , then
       $k_{iaib}^e \leftarrow c_1 k_{iaib}^e + c_2 \text{temp}$ 
    else
      if  $a = b$ , then
         $k_{iaja}^e \leftarrow c_1 k_{iaja}^e$ 
      else
         $k_{iajb}^e \leftarrow c_3 k_{iajb}^e + c_2 k_{jaib}^e$ 
      endif
    endif
  
```

Exercise 3. Determine the number of multiplications and additions for a four-node element stiffness and compare with results of Exercise 2.

Exercise 4. Determine the numbers of multiplications and additions for an eight-node brick element for each of the three implementations. Compare.

Discussion

The first implementation is the most general and easiest to program. The second is somewhat more specialized but registers a significant gain in efficiency. The third is the most specialized and most efficient. In anticipation of extending the present ideas to nonlinear situations, we may note that the matrix D is typically full. Furthermore, a popular method for improving the behavior of elements in both linear and nonlinear analysis engenders a full B . (This method is described in Sec. 4.5.) Under these circumstances the first implementation is still applicable, whereas the others are not. However, the latter two implementations are preferable for the cases to which they pertain. The “best” implementation clearly depends upon what our goals are. For additional information on finite element programming see [20–24].

3.11 ADDITIONAL EXERCISES

Exercise 1. Consider a boundary-value problem in which

$$\Omega = \{(x, y) \mid x \in]a, b[, y \in]a, b[\}$$

A mesh of isoparametric elements is proposed for obtaining an approximate solution to the boundary-value problem (see Figure 3.11.1), in which elements 1 and 2 are eight-node quadrilaterals and elements 3 and 4 are four-node quadrilaterals. What fundamental requirement of the finite element spaces is not met by this mesh? Without changing the number of nodes, how would you fix the mesh?

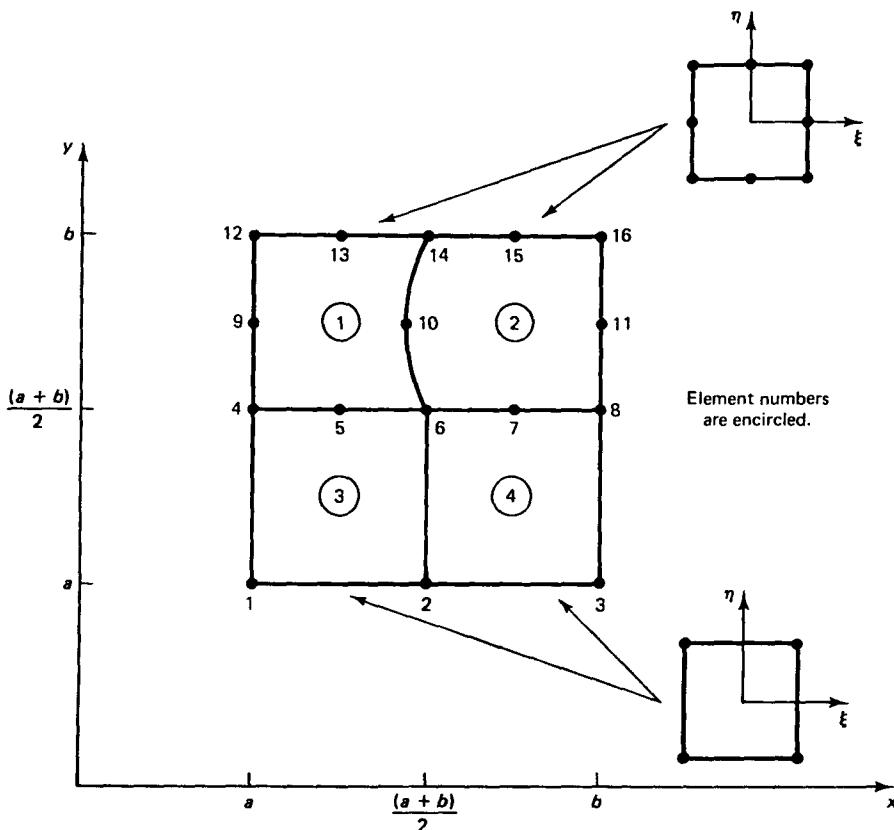


Figure 3.11.1

Exercise 2. Consider a one-dimensional, quadratic, three-node element shown in Fig. 3.11.2. The following relations hold:

$$x(\xi) = \sum_{a=1}^3 N_a(\xi)x_a^e \quad h^e = x_3^e - x_1^e$$

$$u^h(\xi) = \sum_{a=1}^3 N_a(\xi)d_a^e \quad f^e = \{f_a^e\}$$

$$N_1(\xi) = \frac{1}{2}\xi(\xi - 1) \quad f_a^e = \int_{x_1^e}^{x_3^e} N_a \ell \, dx$$

$$N_2(\xi) = 1 - \xi^2 \quad k^e = [k_{ab}^e]$$

$$N_3(\xi) = \frac{1}{2}\xi(\xi + 1) \quad k_{ab}^e = \int_{x_1^e}^{x_3^e} N_{a,x} N_{b,x} \, dx, \quad 1 \leq a, b \leq 3$$

Given a “loading” ℓ , the elements of the consistently derived “force” vector f^e are sometimes surprising, especially for higher-order elements. This problem establishes

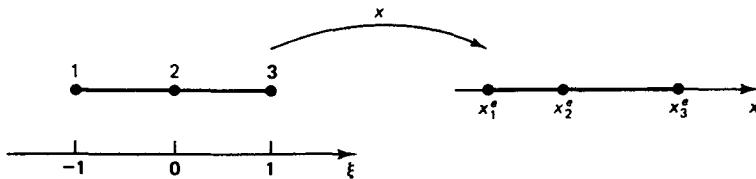


Figure 3.11.2

some basic results concerning the calculation of f^e , appropriate quadrature formulas, and Barlow stress points for the three-node element.

- Assume $f = \text{constant}$ and let $x_2^* = (x_1^* + x_3^*)/2$. Determine exact expressions for f_a^* , $a = 1, 2, 3$.
- ³ Assume $f = \delta(x - \bar{x})$, $x_1^* \leq \bar{x} \leq x_3^*$, (the delta function) and let $x_2^* = (x_1^* + x_3^*)/2$. Obtain an exact expression for f_a^* , $a = 1, 2, 3$, and specialize for the cases $\bar{x} = x_1^*$ and $\bar{x} = x_3^*$.
- Assume $f = \text{constant}$, but make no assumption on the location of x_2^* other than $x_1^* < x_2^* < x_3^*$. Determine the lowest-order Gaussian quadrature formula ($n_{\text{int}} = ?$) which exactly integrates f^e . Justify your answer.
- Assume $x_2^* = (x_1^* + x_3^*)/2$. Determine the lowest-order Gaussian quadrature formula ($n_{\text{int}} = ?$) which exactly integrates k^e . Justify your answer.
- An analysis of the quadratic element is being performed to locate the Barlow stress points (i.e., the points at which $e_{,x} = u_{,x}^h - u_{,x}$ optimally converges). It is assumed in the analysis that $x_2^* = (x_1^* + x_3^*)/2$ and the nodal values are exact, that is, $d_a^* = u(x_a)$. Determine the Barlow stress points.

Solution of part (e) We present a simple solution to illustrate that we do not have to perform a lot of algebraic manipulations to solve problems of this type.

Note that the position of node 2 implies that

³We need to be careful in changing variables when delta functions are present. Consider the multidimensional case in which

$$d\Omega = j d\Box$$

The delta function transforms by multiplication with the *inverse* Jacobian determinant:

$$\delta_{\bar{x}} = \delta(x - \bar{x}) = j^{-1} \delta(\xi - \bar{\xi}) = j^{-1} \delta_{\xi}$$

where

$$\bar{x} = x(\xi)$$

In one dimension, this specializes as follows:

$$dx = x_{,\xi}(\xi) d\xi$$

$$\delta_{\bar{x}} = \delta(x - \bar{x}) = (x_{,\xi})^{-1} \delta(\xi - \bar{\xi}) = (x_{,\xi})^{-1} \delta_{\xi}$$

$$\bar{x} = x(\xi)$$

Further details concerning transformation rules for generalized functions may be found in Chapter 5 of I. Stakgold, *Boundary Value Problems of Mathematical Physics*, vol. II. New York: Macmillan, 1968.

$$e_{,\epsilon} = e_{,\xi}\xi_{,\epsilon} = \frac{2e_{,\xi}}{h^\epsilon}$$

Observe that if $u(\xi)$ is a quadratic polynomial, then $u^h(\xi) = u(\xi)$ by the assumption that the nodal values are exact. Therefore, take $u(\xi) = c\xi^3$, where c is an arbitrary constant. Compute as follows:

$$\begin{aligned} e_{,\epsilon} &= u_{,\xi}^h - u_{,\xi} \\ &= c(-\xi + \frac{1}{2} + \xi + \frac{1}{2} - 3\xi^2) \end{aligned}$$

Thus $e_{,\epsilon} = 0$ at $\xi = \pm 1/\sqrt{3}$, the Gauss points of the two-point rule. If a formal Taylor-expansion approach is adopted, the coefficient of the $u_{,\epsilon\epsilon\epsilon}$ term leads to the same result after considerably more work!

f. Let

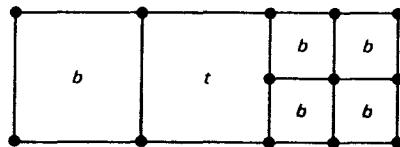
$$x_2^\epsilon = x_1^\epsilon + \frac{h^\epsilon}{4} \quad (\text{quarter point})$$

$$r = \frac{x - x_1^\epsilon}{h^\epsilon}$$

$$h^\epsilon = x_3^\epsilon - x_1^\epsilon$$

Determine an expression for $u_{,r}^h(r)$ and indicate the order of the singularity at $r = 0$ [i.e., determine α , where $u_{,r}^h = O(r^{-\alpha})$]. (This result is useful for the construction of *crack elements*.)

Exercise 3. It is desired to develop a *transition element*, which will facilitate abrupt changes in mesh refinement between domains consisting of bilinear quadrilateral elements. The idea is illustrated in Fig. 3.11.3. The transition element is to be designed to maintain continuity across all interfaces. A shape function associated with node 5 which achieves this end is illustrated in Fig. 3.11.4. Starting with the usual bilinear shape functions, determine the appropriate modifications. Sketch the modified shape function associated with node 1.



Key: b = bilinear quadrilateral
 t = transition element

Figure 3.11.3

Exercise 4. Let

$$\begin{Bmatrix} r(\xi, \eta) \\ \theta(\xi, \eta) \end{Bmatrix} = \sum_{a=1}^4 N_a(\xi, \eta) \begin{Bmatrix} r_a^\epsilon \\ \theta_a^\epsilon \end{Bmatrix}$$

Determine the N_a 's for a *sector element* (see Fig. 3.11.5), so that the following conditions are met:

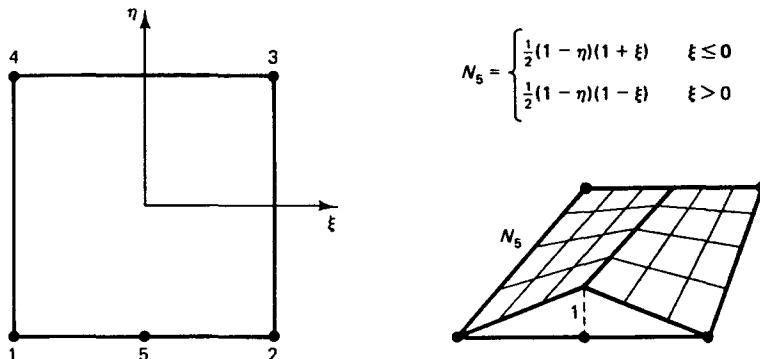


Figure 3.11.4

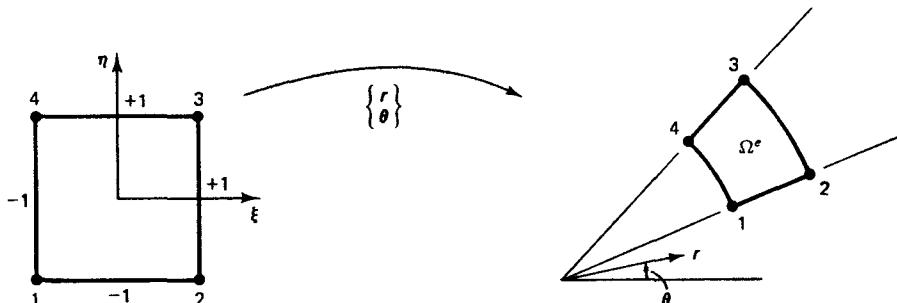


Figure 3.11.5

- i. If $r_4 = r_1$, then $r(-1, \eta) = r_1$.
- ii. If $r_3 = r_2$, then $r(+1, \eta) = r_2$.
- iii. If $\theta_2 = \theta_1$, then $\theta(\xi, -1) = \theta_1$.
- iv. If $\theta_4 = \theta_3$, then $\theta(\xi, +1) = \theta_3$.

Exercise 5. (See the solution of Exercise 2, part (e).) Determine the Barlow stress points for the four-node element in one dimension. Assume that

$$x_2^\epsilon = x_1^\epsilon + \frac{h^\epsilon}{3}$$

$$x_3^\epsilon = x_1^\epsilon + \frac{2h^\epsilon}{3}$$

$$h^\epsilon = x_4^\epsilon - x_1^\epsilon$$

$$d_a^\epsilon = u(x_a^\epsilon)$$

The regular positioning of the internal nodes implies $x_{\cdot \epsilon} = h^\epsilon/2$. (Ans.: 0, $\pm (\frac{5}{9})^{1/2}$. The three Barlow points in this example do *not* coincide with the Gauss points of the three-point rule!)

Exercise 6. A boundary-value problem for the *convection-diffusion equation* is as follows:

Given $f : \Omega \rightarrow \mathbb{R}$, $g : \Gamma_\sigma \rightarrow \mathbb{R}$, and $h : \Gamma_t \rightarrow \mathbb{R}$, find $u : \bar{\Omega} \rightarrow \mathbb{R}$ such that

$$\begin{aligned} b_i u_{,i} &= (\kappa_{ij} u_{,j})_{,i} + f && \text{in } \Omega \\ u &= g && \text{on } \Gamma_\sigma \\ \kappa_{ij} u_{,j} n_i &= h && \text{on } \Gamma_t \end{aligned}$$

- Establish a weak formulation for this problem in which the boundary condition on Γ_σ is essential and the boundary condition on Γ_t is natural.
- Establish the Galerkin formulation and obtain expressions for K_{PQ} and F_P .
- Show that if $b_{j,j} = 0$ on Ω and $b_j n_j = 0$ on Γ_t , then the b -term contribution to K_{PQ} is skew-symmetric (i.e., $K_{PQ}^b = -K_{QP}^b$).

Exercise 7.

- Consider the four-node bilinear element. Prove that the one-point Gauss quadrature formula in two dimensions is sufficient to exactly integrate the area

$$\int_{\Omega^\epsilon} d\Omega$$

b. Let

$$u_i^h(\xi, \eta) = \sum_{a=1}^4 N_a(\xi, \eta) d_{ia}^\epsilon, \quad i = 1, 2$$

where the N_a 's are the bilinear shape functions. Show that if

$$u_{i,i}^h(0, 0) = 0$$

then

$$\int_{\Omega^\epsilon} u_{i,i}^h d\Omega = 0 \quad (\text{"mean incompressibility"})$$

(Part (b) requires some work.)

Exercise 8. Consider a bilinear quadrilateral element. Assume that the prescribed surface traction (i.e., h_i) along one edge is constant. What is the lowest-order, one-dimensional Gauss quadrature formula which will exactly integrate the prescribed traction contribution to f^ϵ ? Justify your answer.

Solution The setup is shown in Fig. 3.11.6. Note that

$$s = N_a(\xi) s_a + N_b(\xi) s_b$$

Piecewise linear shape functions

Calculate as follows:

$$\int_{\Gamma_{h_i}} N_a h_i d\Gamma = \int_0^L N_a h_i ds \quad (\text{continued})$$

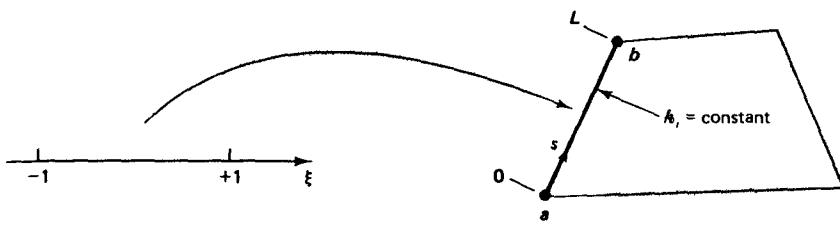


Figure 3.11.6

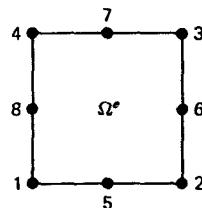
$$= h_i \int_{-1}^{+1} N_a(\xi) \frac{L}{2} d\xi$$

$$= \frac{L}{2} h_i \int_{-1}^{+1} \underbrace{N_a(\xi)}_{\text{Linear}} d\xi$$

$$= \frac{L}{2} h_i$$

Because the integral involves a linear polynomial in ξ , one-point Gauss quadrature suffices.

Exercise 9. Consider the eight-node isoparametric element shown:



Assume the nodal coordinates are as follows:

| a | x_a | y_a |
|-----|----------------|----------------|
| 1 | $-\frac{h}{2}$ | $-\frac{h}{2}$ |
| 2 | $\frac{h}{2}$ | $-\frac{h}{2}$ |
| 3 | $\frac{h}{2}$ | $\frac{h}{2}$ |
| 4 | $-\frac{h}{2}$ | $\frac{h}{2}$ |
| 5 | 0 | $-\frac{h}{2}$ |
| 6 | $\frac{h}{2}$ | 0 |
| 7 | 0 | $\frac{h}{2}$ |
| 8 | $-\frac{h}{2}$ | 0 |

Consider the following body force:

$$\boldsymbol{f} = \begin{Bmatrix} 0 \\ -g \end{Bmatrix}; \quad g \text{ constant}$$

Compute (by hand) the nodal forces for nodes 1 and 5, i.e.,

$$f_{21}^e = \int_{\Omega^e} N_1 f_2 \, dx$$

$$f_{25}^e = \int_{\Omega^e} N_5 f_2 \, dx$$

(Note that $f_{1a} = 0$, $1 \leq a \leq 8$, and, by symmetry, $f_{21}^e = f_{22}^e = f_{23}^e = f_{24}^e$ and $f_{25}^e = f_{26}^e = f_{27}^e = f_{28}^e$.)

The results of this simple computation should be somewhat surprising. Comment.

Exercise 10. Generalize Exercise 8 to the case in which the quadrilateral is of biquadratic type. Assume that the three nodes along an edge define a curved edge (i.e., they do not lie along a straight line). [Ans.: Two-point Gauss.]

Exercise 11. Generalize Exercise 10 to the case in which h_i represents normal pressure. That is,

$$h_i = -pn_i$$

where p , the applied pressure, is assumed constant and n_i is the unit outward normal to the boundary. [Ans.: Three-point Gauss.]

Exercise 12. Consider a trilinear brick element subjected to normal pressure along one surface. What is the lowest-order, two-dimensional Gauss quadrature formula which will exactly integrate the prescribed traction contribution to \boldsymbol{f}^e ? Justify your answer. Hint: The following change-of-variables formula is useful in situations like these:

$$\int_{\Gamma_{h_i}} N_a h_i \, d\Gamma = \int_{-1}^{+1} \int_{-1}^{+1} N_a h_i \| \mathbf{x}_{,\xi} \times \mathbf{x}_{,\eta} \| \, d\xi \, d\eta$$

where \times denotes cross product and $\| \cdot \|$ is the Euclidean length.

Exercise 13. Generalize Exercise 12 to the case of the triquadratic brick element.

Appendix 3.1

Triangular and Tetrahedral Elements

It is often stated that triangular and tetrahedral elements are responsible for the geometric flexibility of the finite element method. This is perhaps somewhat of an exaggeration as triangular and tetrahedral shapes are often not needed in practice. Most regions are conveniently discretized by arbitrary quadrilateral and brick-shaped elements, as described earlier.

It is also often stated that triangles and tetrahedra enable modeling of particularly intricate geometries and that these shapes facilitate transition from coarsely meshed zones of a grid to finely meshed zones. This is, of course, true, but quadrilaterals and bricks are capable of doing the same thing at least to some degree. To illustrate this point, consider Fig. 3.I.1 in which a triangular zone is discretized into three quadrilaterals. Thus we see that any triangular element could be replaced by quadrilaterals. (Mesh generation for triangular regions via quadrilaterals may be handled similarly; see Fig. 3.I.2.) Quadrilaterals may also be used to perform mesh transition as illustrated in Fig. 3.I.3.

Nevertheless, if a few triangles are desired (or tetrahedra, or wedges) in any particular situation, they may be easily obtained from quadrilaterals (respectively, bricks) by way of the element degeneration technique described previously. Consequently, no special finite element subroutines are required.

However, because many individuals are fond of particular triangular and tetrahedral elements and do not perceive them as being degenerated from quadrilaterals and bricks and because there are a number of useful triangular and tetrahedral elements for which there are no quadrilateral and brick counterparts, we shall describe the machinery necessary for the direct development of triangular and tetrahedral shape functions. The following techniques should also be employed if extensive use of triangles or tetrahedra is envisioned, because, in this case, the degeneration procedure proves inefficient.

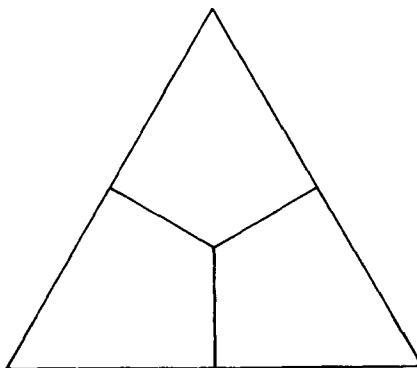


Figure 3.I.1

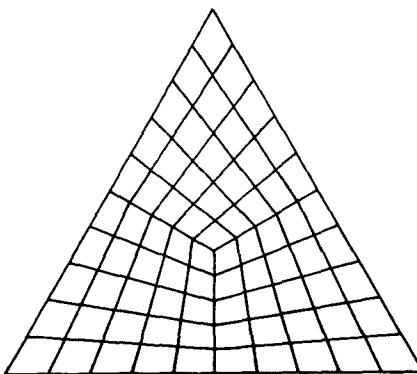


Figure 3.I.2

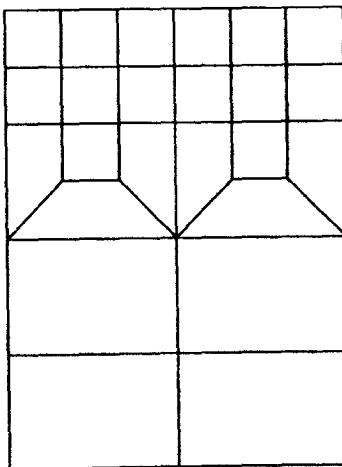


Figure 3.I.3

Shape Functions for Triangles

Consider the *parent triangle* illustrated in Fig. 3.I.4. This serves as the counterpart of the biunit square in ξ, η -space which was described in Sec. 3.2. We take r and s as the independent natural coordinates. Note that the equation of the inclined edge is

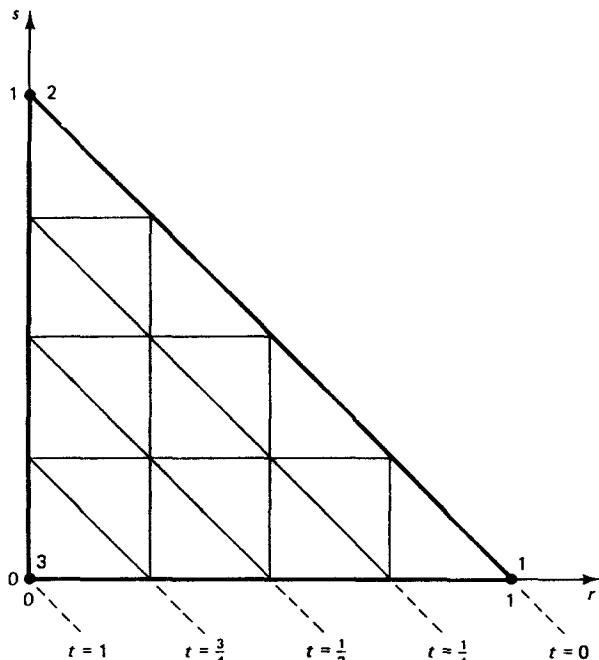


Figure 3.I.4

$1 - r - s = 0$. If we define

$$t = t(r, s) = 1 - r - s \quad (3.I.1)$$

then the family of $t = \text{constant}$ lines are parallel to the inclined edge. The triple r, s, t defines **triangular coordinates**. Each is zero along one edge and takes on the value 1 at the opposite vertex. The shape functions of the piecewise linear triangle may be concisely expressed with the aid of the triangular coordinates, viz.,

$$N_1(r, s) = r \quad (3.I.2)$$

$$N_2(r, s) = s \quad (3.I.3)$$

$$N_3(r, s) = t(r, s) = 1 - r - s \quad (3.I.4)$$

These functions are equivalent to those computed by the degeneration technique in Sec. 3.4. Higher-order shape functions for triangles may be systematically defined through the use of triangular coordinates. The general formula for **Lagrange-type interpolation** over triangles is given by

$$N_a(r, s, t) = T_l(r)T_j(s)T_k(t) \quad (3.I.5)$$

where

$$T_l(r) = \begin{cases} l!^{-1} \left(\frac{2r}{r_l - 1} \right), & l \neq 1 \\ 1, & l = 1 \end{cases} \quad (3.I.6)$$

and $a = a(I, J, K)$ is the formula defining the single nodal index and I in (3.I.6) stand for the Lagrange interpolation formula (see Sec. 3.6). Illustrations of the use of (3.I.6) are given in the following examples.

Example 1

Consider the three-node triangle depicted in Fig. 3.I.5. The shape functions are:

$$\begin{aligned} N_1(r, s, t) &= T_2(r)T_1(s)T_1(t) \\ &= l_2^1 \left(\frac{2r}{r_2 - 1} \right) \\ &= r \end{aligned} \quad (3.I.7)$$

$$\begin{aligned} N_2(r, s, t) &= T_1(r)T_2(s)T_1(t) \\ &= l_2^1 \left(\frac{2s}{s_2 - 1} \right) \\ &= s \end{aligned} \quad (3.I.8)$$

$$\begin{aligned} N_3(r, s, t) &= T_1(r)T_1(s)T_2(t) \\ &= l_2^1 \left(\frac{2t}{t_2 - 1} \right) \\ &= t \end{aligned} \quad (3.I.9)$$

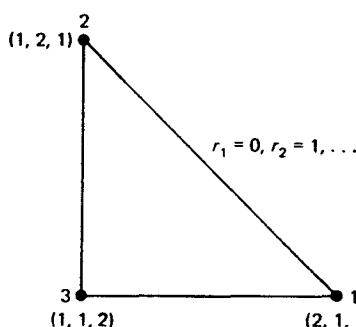


Figure 3.I.5

Example 2

Consider the six-node triangle shown in Fig. 3.I.6.

$$\begin{aligned} N_1(r, s, t) &= T_3(r)T_1(s)T_1(t) \\ &= l_3^1 \left(\frac{2r}{r_3 - 1} \right) \\ &= r(2r - 1) \end{aligned} \quad (3.I.10)$$

$$\begin{aligned} N_2(r, s, t) &= T_1(r)T_3(s)T_1(t) \\ &= l_3^1 \left(\frac{2s}{s_3 - 1} \right) \quad (\text{continued}) \end{aligned}$$

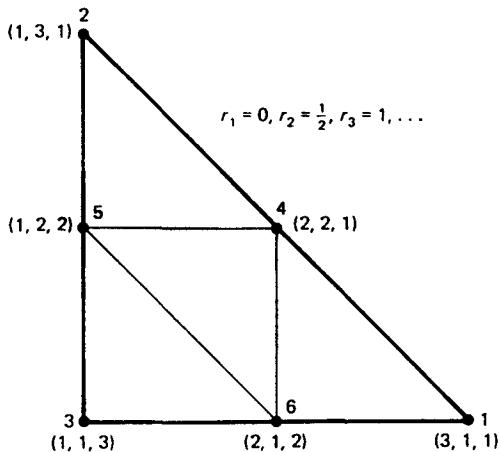


Figure 3.I.6

$$= s(2s - 1) \quad (3.I.11)$$

$$\begin{aligned} N_3(r, s, t) &= T_1(r)T_1(s)T_3(t) \\ &= l_3^2\left(\frac{2t}{t_3 - 1}\right) \\ &= t(2t - 1) \end{aligned} \quad (3.I.12)$$

$$\begin{aligned} N_4(r, s, t) &= T_2(r)T_2(s)T_1(t) \\ &= l_2^1\left(\frac{2r}{r_2 - 1}\right)l_2^1\left(\frac{2s}{s_2 - 1}\right) \\ &= 4rs \end{aligned} \quad (3.I.13)$$

$$\begin{aligned} N_5(r, s, t) &= T_1(r)T_2(s)T_2(t) \\ &= l_2^1\left(\frac{2s}{s_2 - 1}\right)l_2^1\left(\frac{2t}{t_2 - 1}\right) \\ &= 4st \end{aligned} \quad (3.I.14)$$

$$\begin{aligned} N_6(r, s, t) &= T_2(r)T_1(s)T_2(t) \\ &= l_2^1\left(\frac{2r}{r_2 - 1}\right)l_2^1\left(\frac{2t}{t_2 - 1}\right) \\ &= 4rt \end{aligned} \quad (3.I.15)$$

Example 3

Consider the 10-node triangle depicted in Fig. 3.I.7.

$$\begin{aligned} N_1(r, s, t) &= T_4(r)T_1(s)T_1(t) \\ &= l_4^3\left(\frac{2r}{r_4 - 1}\right) \quad (\text{continued}) \end{aligned}$$

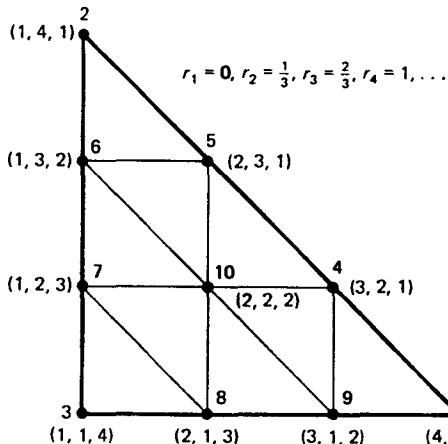


Figure 3.I.7

$$= \frac{9}{2}r\left(r^2 - r + \frac{2}{9}\right) \quad (3.I.16)$$

$$\begin{aligned} N_4(r, s, t) &= T_3(r)T_2(s)T_1(t) \\ &= l_3^1\left(\frac{2r}{r_3 - 1}\right)l_2^1\left(\frac{2s}{s_2 - 1}\right) \\ &= \frac{9}{2}sr(3r - 1) \end{aligned} \quad (3.I.17)$$

$$\begin{aligned} N_{10}(r, s, t) &= T_2(r)T_2(s)T_2(t) \\ &= l_2^1\left(\frac{2r}{r_2 - 1}\right)l_2^1\left(\frac{2s}{s_2 - 1}\right)l_2^1\left(\frac{2t}{t_2 - 1}\right) \\ &= 27rst \end{aligned} \quad (3.I.18)$$

The reader may wish to calculate the remaining shape functions.

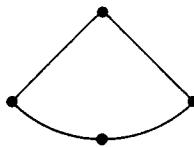
Exercise 1. Determine the shape functions for the 15-node quartic Lagrange-type triangle.

Remark

Note that the Lagrange-type shape functions for triangles result in complete polynomials without any extraneous monomials (see the Pascal triangle, Fig. 3.7.8).

Serendipity-type triangular elements may also be derived with the aid of triangular coordinates. However, there does not seem to be much practical interest in elements of this type.

Exercise 2. Use triangular coordinates to develop shape functions for a four-node triangle which exhibits quadratic behavior along one edge and linear behavior along the other two (see the accompanying figure on page 170).



Shape Functions for Tetrahedra

The *parent tetrahedron* is illustrated in Fig. 3.I.8. The *tetrahedral coordinates* are r , s , t . Each is zero on one surface of the tetrahedron and takes on the value 1 at the opposite vertex. The shape functions of the linear tetrahedron are simply

$$N_1(r, s, t) = r \quad (3.I.19)$$

$$N_2(r, s, t) = s \quad (3.I.20)$$

$$N_3(r, s, t) = t \quad (3.I.21)$$

$$N_4(r, s, t) = u(r, s, t) = 1 - r - s - t \quad (3.I.22)$$

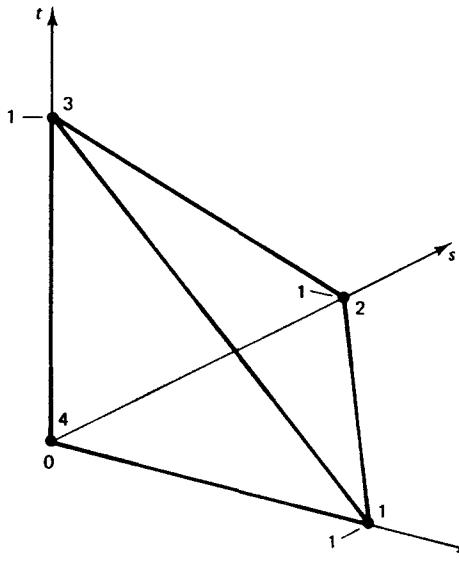


Figure 3.I.8

As in the case of triangles, shape functions for Lagrange-type tetrahedra may be derived with the aid of a simple formula:

$$N_a(r, s, t, u) = T_I(r)T_J(s)T_K(t)T_L(u) \quad (3.I.23)$$

where the T 's are again defined by (3.I.6) and $a = a(I, J, K, L)$.

Exercise 3. Consider the 10-node quadratic tetrahedron shown in Fig. 3.I.9. Using (3.I.23) derive the shape functions and verify that they satisfy the interpolation property (i.e., $N_a(r_b, s_b, t_b, u_b) = \delta_{ab}$).

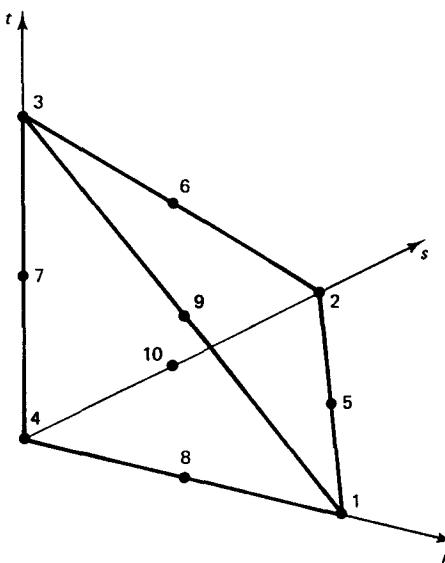


Figure 3.I.9

Shape Functions for Wedge-Shaped Elements

Shape functions of Lagrange-type for wedge-shaped elements may be derived by taking products of one-dimensional and triangular shape functions.

Example 4

Consider the basic six-node wedge element illustrated in Fig. 3.I.10. Shape functions for this element derive from the linear one-dimensional element and linear triangle, viz.,

$$N_1(\xi, r, s, t) = \frac{1}{2}(1 - \xi)r \quad (3.I.24)$$

$$N_2(\xi, r, s, t) = \frac{1}{2}(1 - \xi)s \quad (3.I.25)$$

$$N_3(\xi, r, s, t) = \frac{1}{2}(1 - \xi)t \quad (3.I.26)$$

$$N_4(\xi, r, s, t) = \frac{1}{2}(1 + \xi)r \quad (3.I.27)$$

$$N_5(\xi, r, s, t) = \frac{1}{2}(1 + \xi)s \quad (3.I.28)$$

$$N_6(\xi, r, s, t) = \frac{1}{2}(1 + \xi)t \quad (3.I.29)$$

Higher-order wedges may be similarly derived.

Integration

The integration of monomials over straight-edged triangles and flat-surfaced tetrahedra may be performed with the aid of the following exact expressions:

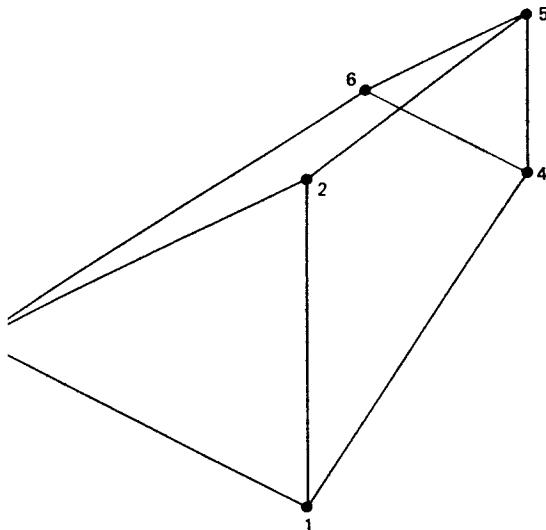


Figure 3.I.10

Triangles

$$\int_{\Omega} r^{\alpha} s^{\beta} t^{\gamma} d\Omega = \frac{\alpha! \beta! \gamma!}{(\alpha + \beta + \gamma + 2)!} 2A \quad (3.I.30)$$

where A , the area of Ω , is given by

$$2A = \det \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \quad (3.I.31)$$

in which x_a and y_a are the coordinates of vertex a .

Tetrahedra

$$\int_{\Omega} r^{\alpha} s^{\beta} t^{\gamma} u^{\delta} d\Omega = \frac{\alpha! \beta! \gamma! \delta!}{(\alpha + \beta + \gamma + \delta + 3)!} 6V \quad (3.I.32)$$

where V , the volume of Ω , is given by

$$6V = \det \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix} \quad (3.I.33)$$

in which x_a , y_a , and z_a are the coordinates of vertex a .

In the case of distorted elements, integrands will not consist simply of monomials. Consequently, numerical integration needs to be used. Formulas that are symmetrical with respect to the tetrahedral coordinates have been derived. References [25–27] may be consulted for results of this kind. A sampling of results is presented in Tables 3.I.1 and 3.I.2. Higher-order rules for tetrahedra are presented in [28].

App. 3.I Triangular and Tetrahedral Elements

TABLE 3.I.1 Integration Rules for Triangles [26, 27]

| w_i | \tilde{r} | \tilde{s} | \tilde{t} |
|---|---------------------|---------------------|---------------------|
| <i>3-point formula</i> <i>degree of precision 2⁽²⁾</i> | | | |
| 0.33333 33333 33333 | 0.66666 66666 66667 | 0.16666 66666 66667 | 0.16666 66666 66667 |
| <i>3-point formula</i> <i>degree of precision 2</i> | | | |
| 0.33333 33333 33333 | 0.50000 00000 00000 | 0.50000 00000 00000 | 0.00000 00000 00000 |
| <i>4-point formula</i> <i>degree of precision 3</i> | | | |
| -0.56250 00000 00000 | 0.33333 33333 33333 | 0.33333 33333 33333 | 0.33333 33333 33333 |
| 0.52083 33333 33333 | 0.60000 00000 00000 | 0.20000 00000 00000 | 0.20000 00000 00000 |
| <i>6-point formula</i> <i>degree of precision 3</i> | | | |
| 0.16666 66666 66667 | 0.65902 76223 74092 | 0.23193 33685 53031 | 0.10903 90090 72877 |
| <i>6-point formula</i> <i>degree of precision 4</i> | | | |
| 0.10995 17436 55322 | 0.81684 75729 80459 | 0.09157 62135 09771 | 0.09157 62135 09771 |
| 0.22338 15896 78011 | 0.10810 30181 68070 | 0.44594 84909 15965 | 0.44594 84909 15965 |
| <i>7-point formula</i> <i>degree of precision 4</i> | | | |
| 0.37500 00000 00000 | 0.33333 33333 33333 | 0.33333 33333 33333 | 0.33333 33333 33333 |
| 0.10416 66666 66667 | 0.73671 24989 68435 | 0.23793 23664 72434 | 0.02535 51345 51932 |
| <i>7-point formula</i> <i>degree of precision 5</i> | | | |
| 0.22503 00003 00000 | 0.33333 33333 33333 | 0.33333 33333 33333 | 0.33333 33333 33333 |
| 0.12593 91805 44827 | 0.79742 69853 53087 | 0.10128 65073 23456 | 0.10128 65073 23456 |
| 0.13239 41527 88506 | 0.47014 20641 05115 | 0.47014 20641 05115 | 0.05971 58717 89770 |
| <i>9-point formula</i> <i>degree of precision 5</i> | | | |
| 0.20595 05047 60887 | 0.12494 95032 33232 | 0.43752 52483 83384 | 0.43752 52483 83384 |
| 0.06369 14142 86223 | 0.79711 26518 60071 | 0.16540 99273 89841 | 0.03747 74207 50088 |
| <i>12-point formula</i> <i>degree of precision 6</i> | | | |
| 0.05084 49063 70207 | 0.87382 19710 16996 | 0.06308 90144 91502 | 0.06308 90144 91502 |
| 0.11678 62757 26379 | 0.50142 65096 58179 | 0.24928 67451 70910 | 0.24928 67451 70911 |
| 0.08285 10756 18374 | 0.63650 24991 21399 | 0.31035 24510 33785 | 0.05314 50498 44816 |

⁽¹⁾The formulas are symmetric in the triangular coordinates. All permutations of the quadrature points accounted for.

⁽²⁾The degree of precision refers to the order of the remainder. (One-point, centroidal quadrature achieves precision equal to two.)

E 3.I.1 Integration Rules for Triangles [26, 27] (cont.)

| W_i | \tilde{r} | \tilde{s} | \tilde{t} | Multiplicity ⁽¹⁾ |
|-------------------------|---------------------|------------------------------|---------------------|-----------------------------|
| <i>13-point formula</i> | | <i>degree of precision 7</i> | | |
| 17 00444 67670 | 0.33333 33333 33333 | 0.33333 33333 33333 | 0.33333 33333 33333 | 1 |
| 1 52574 33204 | 0.47930 80678 41923 | 0.26034 59660 79038 | 0.26034 59660 79038 | 3 |
| 4 72356 08839 | 0.86973 97941 95568 | 0.06513 01029 02216 | 0.06513 01029 02216 | 3 |
| 1 37608 90257 | 0.63844 41885 69809 | 0.31286 54960 04875 | 0.48690 31542 53160 | 6 |

TABLE 3.I.2 Integration Rules for Tetrahedra [25]

| W_i | \tilde{r} | \tilde{s} | \tilde{t} | \tilde{u} | Multiplicity |
|------------------------|---------------|------------------------------|---------------|---------------|--------------|
| <i>1-point formula</i> | | <i>degree of precision 2</i> | | | |
| 1 | $\frac{1}{4}$ | $\frac{1}{4}$ | $\frac{1}{4}$ | $\frac{1}{4}$ | 1 |
| <i>4-point formula</i> | | <i>degree of precision 3</i> | | | |
| $\frac{1}{4}$ | 0.58541020 | 0.13819660 | 0.13819660 | 0.13819660 | 4 |
| <i>5-point formula</i> | | <i>degree of precision 4</i> | | | |
| $-\frac{4}{3}$ | $\frac{1}{4}$ | $\frac{1}{4}$ | $\frac{1}{4}$ | $\frac{1}{4}$ | 1 |
| $\frac{9}{20}$ | $\frac{1}{3}$ | $\frac{1}{6}$ | $\frac{1}{6}$ | $\frac{1}{6}$ | 4 |

It is traditional to tabulate quadrature rules for triangles and tetrahedra such that the weights sum to 1. Because the area of the parent triangle equals $\frac{1}{2}$ and the volume of the parent tetrahedron equals $\frac{1}{6}$, the proper form for a quadrature rule in these cases is

$$\int_{\Omega} f d\Omega \cong c \sum_{l=1}^{n_{\text{int}}} f_l j_l W_l \quad (3.I.34)$$

where $c = \frac{1}{2}$ for triangles and $c = \frac{1}{6}$ for tetrahedra.

Derivatives of Shape Functions

To calculate derivatives of shape functions with respect to global coordinates we may employ the techniques described in Sec. 3.9. All formulas of Sec. 3.9 may be employed with the following substitutions.

Triangles

Replace ξ and η by r and s , respectively; set $t = 1 - r - s$ throughout.

Tetrahedra

Replace ξ , η , and ζ by r , s , and t , respectively; set $u = 1 - r - s - t$ throughout.

Appendix 3.II

Methodology for Developing Special Shape Functions with Application to Singularities

In many areas of finite element analysis, elements with special properties are required to achieve maximal accuracy. As examples, we may mention infinite elements for the representation of spatial domains that extend to infinity [29, 30] and singular elements for modeling point and line singularities engendered by geometric features such as reentrant corners and cracks [31–33]. In this appendix we are concerned with techniques for developing shape functions for special elements in general but with particular emphasis on singularities.

Geometric features of the spatial domain may give rise to highly singular response in the solution of a field problem [34]. Eigenfunction analyses are frequently available for purposes of characterizing the asymptotic strength of the singularity [35]. However, the actual solution depends upon the nature of the loading and there are situations when the singularity does not contribute significantly [36]. For this reason it has been suggested that, ideally, singular finite elements should strike some balance between being able to represent smooth behavior and potential singular behavior [36]. In this way the element will remain effective under all loading conditions. Translated into analytical terms, the finite element shape functions should be designed to represent the standard low-order polynomials as well as the singular functions emanating from asymptotic analysis. To construct shape functions of this type gives rise to an interpolation problem, which involves a system of linear simultaneous equations. In special circumstances explicit representations are available for the solution of the interpolation problem. The most well known example is the Lagrange interpolation formula, which is the solution of the interpolation problem for polynomials passing through distinct points. It appears that no such representation exists when different classes of functions are merged, as seems appropriate for singular elements. Although, in principle, the interpolation problem could be solved numerically for each singular element during formation, this would be highly inefficient.

of computations required. Furthermore, numerical sensitivity may manifest itself in certain configurations. Consequently, it is of practical interest to develop techniques for systematically defining shape functions for singularity modeling (and for developing special elements in general), which circumvent the interpolation problem. In what follows, a highly concise algorithm for generating shape functions from an arbitrary starting set of independent functions is presented [37]. A novel feature of the procedure is that *no* coefficient matrix is involved (i.e., no system of simultaneous equations need be solved). Some useful one-dimensional interpolatory schemes for modeling singularities are developed by employing the interpolation algorithm. These schemes are the basis for constructiong two- and three-dimensional elements for modeling point and line singularities.

Let $r = r(\xi) = (1 + \xi)/2$ and $r_a = r(\xi_a)$. Note that $r(\xi) \in [0, 1]$ for all $\xi \in [-1, 1]$. The r -coordinate description is convenient for modeling singularities. In multidimensional applications, we use boldfaced notations analogous to the preceding ones. For example, in two dimensions $\mathbf{r} = (r, s)$, where $s = (1 + \eta)/2$, $\eta \in [-1, 1]$.

The *Lagrange polynomials* are defined by

$$l_a(f) = \frac{\prod_{\substack{b=1 \\ b \neq a}}^m (f - f_b)}{\prod_{\substack{b=1 \\ b \neq a}}^m (f_a - f_b)}, \quad a = 1, 2, \dots, m \quad (3.II.1)$$

where the f_a 's are distinct points in \mathbb{R} . It may be seen from (3.II.1) that $l_a(f)$ is an $(m - 1)$ st order polynomial in f that satisfies the “interpolation property”(i.e., $l_a(f_b) = \delta_{ab}$, the Kronecker delta). Note that f may be taken to be ξ, r, r^α , or any other convenient function of ξ .

Algorithm for Constructing Interpolation Functions

Consider an n -node finite element. We assume we are given a set of “preliminary” shape functions, N_a , $a = 1, 2, \dots, n$, which satisfy the interpolation property on the first m nodes only; viz., $N_a(r_b) = \delta_{ab}$, where $a, b = 1, 2, \dots, m < n$. The idea is to modify systematically the N_a 's such that the interpolation property is satisfied on all n nodes. The procedure is given as follows:

$$\text{Step 1} \quad N_{m+1}(\mathbf{r}) \leftarrow \frac{N_{m+1}(\mathbf{r}) - \sum_{a=1}^m N_{m+1}(r_a)N_a(\mathbf{r})}{N_{m+1}(r_{m+1}) - \sum_{a=1}^m N_{m+1}(r_a)N_a(r_{m+1})} \quad (3.II.2)$$

$$\text{Step 2} \quad N_a(\mathbf{r}) \leftarrow N_a(\mathbf{r}) - N_a(r_{m+1})N_{m+1}(\mathbf{r}), \quad a = 1, 2, \dots, m \quad (3.II.3)$$

Step 3 If $m + 1 < n$, replace m by $m + 1$ and repeat Steps 1 to 3.
 If $m + 1 = n$, stop.

Remarks

- After completing Steps 1 and 2, the shape functions satisfy the interpolation property on the first $m + 1$ nodes. Clearly, after completing the entire procedure, the desired properties are achieved.
- In many cases, the algorithm enables the explicit hand calculation of shape functions. *However, there is no need to do this in practice, as the algorithm itself may be programmed as part of a shape function subroutine.*
- The derivatives of shape functions may be constructed in similar fashion. The analogs of Steps 1 and 2 are respectively

$$\partial N_{m+1}(\mathbf{r}) \leftarrow \frac{\partial N_{m+1}(\mathbf{r}) - \sum_{a=1}^m N_{m+1}(\mathbf{r}_a) \partial N_a(\mathbf{r})}{N_{m+1}(\mathbf{r}_{m+1}) - \sum_{a=1}^m N_{m+1}(\mathbf{r}_a) N_a(\mathbf{r}_{m+1})} \quad (3.II.4)$$

and

$$\partial N_a(\mathbf{r}) \leftarrow \partial N_a(\mathbf{r}) - N_a(\mathbf{r}_{m+1}) \partial N_{m+1}(\mathbf{r}) \quad (3.II.5)$$

In (3.II.4) and (3.II.5), ∂ represents the partial differentiation operator with respect to any of the coordinates (e.g., in two dimensions, ∂ may represent $\partial/\partial r$ or $\partial/\partial s$).

4. Interpolation problems are generally formulated in terms of a system of simultaneous linear equations. It may be noted that, if $m = 1$, the algorithm encompassed by Steps 1–3 solves the interpolation problem *without* engendering any coefficient matrix. Thus no “storage” is required (beyond that for the N_a 's) in programming the procedure. Another nice feature of the algorithm is that the interpolation property possessed by the preliminary shape functions is fully exploited in that only $n - m$ passes through Steps 1–3 need be performed.

In practice, the Lagrange polynomial formula may be employed to generate a partial set of preliminary shape functions, which satisfies the interpolation property on the first m nodes. This is illustrated in the examples to follow.

Example 1

Consider a one-dimensional three-node element with nodes given by $r_1 = 0$, $r_2 = \frac{1}{2}$, and $r_3 = 1$. We wish to construct shape functions capable of exactly representing 1, r , and r^α . (If $\alpha = 2$, we have the usual three-node quadratic element. Other values of α enable the modeling of singularities.) We begin with the following preliminary shape functions:

$$N_1(r) = 1 - 2r, \quad N_2(r) = 2r, \quad N_3(r) = r^\alpha \quad (3.II.6)$$

Observe that $N_a(r_b) = \delta_{ab}$, $1 \leq a, b \leq 2$. In terms of our algorithm, $m = 2$ and $n = 3$, so only one pass through Steps 1 and 2 is required:

$$N_3(r) \leftarrow \frac{N_3(r) - N_3(\frac{1}{2})N_2(r)}{N_3(1) - N_3(\frac{1}{2})N_2(1)} = \frac{r^\alpha - 2(\frac{1}{2})^\alpha r}{1 - 2(\frac{1}{2})^\alpha} \quad (3.II.7)$$

$$N_1(r) \leftarrow N_1(r) - N_1(1)N_3(r) = 1 - 2r + \left[\frac{r^\alpha - 2(\frac{1}{2})^\alpha r}{1 - 2(\frac{1}{2})^\alpha} \right] \quad (3.II.8)$$

$$N_2(r) \leftarrow N_2(r) - N_2(1)N_3(r) = 2r - 2 \left[\frac{r^\alpha - 2(\frac{1}{2})^\alpha r}{1 - 2(\frac{1}{2})^\alpha} \right] \quad (3.II.9)$$

Likewise, the derivatives are given by:

$$\partial N_3(r) \leftarrow \frac{\partial N_3(r) - N_3(\frac{1}{2})\partial N_2(r)}{N_3(1) - N_3(\frac{1}{2})N_2(1)} = \frac{\alpha r^{\alpha-1} - 2(\frac{1}{2})^\alpha}{1 - 2(\frac{1}{2})^\alpha} \quad (3.II.10)$$

$$\partial N_1(r) \leftarrow \partial N_1(r) - N_1(1)\partial N_3(r) = -2 + \left[\frac{\alpha r^{\alpha-1} - 2(\frac{1}{2})^\alpha}{1 - 2(\frac{1}{2})^\alpha} \right] \quad (3.II.11)$$

$$\partial N_2(r) \leftarrow \partial N_2(r) - N_2(1)\partial N_3(r) = 2 - 2 \left[\frac{\alpha r^{\alpha-1} - 2(\frac{1}{2})^\alpha}{1 - 2(\frac{1}{2})^\alpha} \right] \quad (3.II.12)$$

(In this case ∂N is taken to mean $\partial N/\partial r$)

Example 2

Consider a one-dimensional four-node element for which the nodes are given by $r_1 = 0$, $r_2 = \frac{1}{3}$, $r_3 = \frac{2}{3}$, and $r_4 = 1$. We wish to construct shape functions capable of representing 1 , r^α , $r^{2\alpha}$, and r^β . The following special cases are of practical importance:

$\alpha = 1, \beta = 3$: $1, r, r^2, r^3$ cubic polynomial in r

$\alpha = 1, \beta$ arbitrary: $1, r, r^2, r^\beta$ quadratic polynomial in r plus linear polynomial in r^β

α arbitrary, $\beta = 1$: $1, r, r^\alpha, r^{2\alpha}$ linear polynomial in r plus quadratic polynomial in r^α

α arbitrary, $\beta = 3\alpha$: $1, r^\alpha, r^{2\alpha}, r^{3\alpha}$ cubic polynomial in r^α

To obtain a starting set of shape functions, we may employ the Lagrange interpolation formula with $f = r^\alpha$ and $m = 3$, viz.,

$$N_1(r) = l_1(r^\alpha) = \frac{(r^\alpha - r_2^\alpha)(r^\alpha - r_3^\alpha)}{(r_1^\alpha - r_2^\alpha)(r_1^\alpha - r_3^\alpha)} = \frac{(3^\alpha r^\alpha - 1)(3^\alpha r^\alpha - 2^\alpha)}{2^\alpha} \quad (3.II.13)$$

$$N_2(r) = l_2(r^\alpha) = \frac{(r^\alpha - r_1^\alpha)(r^\alpha - r_3^\alpha)}{(r_2^\alpha - r_1^\alpha)(r_2^\alpha - r_3^\alpha)} = \frac{3^\alpha r^\alpha(3^\alpha r^\alpha - 2^\alpha)}{(1 - 2^\alpha)} \quad (3.II.14)$$

$$N_3(r) = l_3(r^\alpha) = \frac{(r^\alpha - r_1^\alpha)(r^\alpha - r_2^\alpha)}{(r_3^\alpha - r_1^\alpha)(r_3^\alpha - r_2^\alpha)} = \frac{3^\alpha r^\alpha(3^\alpha r^\alpha - 1)}{2^\alpha(2^\alpha - 1)} \quad (3.II.15)$$

It may be verified that $N_a(r_b) = \delta_{ab}$ for all $a, b = 1, 2, 3$. The last preliminary shape function is taken to be

$$N_4(r) = r^\beta \quad (3.II.16)$$

With regard to the algorithm, $m = 3$ and $n = 4$, so only one pass through Steps 1 and 2 is needed, viz.,

$$N_4(r) \leftarrow \frac{N_4(r) - \sum_{a=2}^3 N_4(r_a)N_a(r)}{1 - \sum_{a=2}^3 N_4(r_a)N_a(1)} \quad (3.II.17)$$

$$N_a(r) \leftarrow N_a(r) - N_a(1)N_4(r), \quad a = 1, 2, 3 \quad (3.II.18)$$

Equations (3.II.17) and (3.II.18) may be programmed in a shape function subroutine along with corresponding expressions for derivatives.

Remarks

5. The preceding one-dimensional examples cover most cases of practical interest. If the need arises, higher-order shape functions may be constructed along similar lines. The basic philosophy is to maximize m in order to minimize calculations in the shape function routines.

6. For completeness, we wish to mention the simplest one-dimensional shape functions that allow the modeling of singularities; namely, the two-node element with $r_1 = 0$, $r_2 = 1$, $N_1(r) = 1 - r^\alpha$, and $N_2(r) = r^\alpha$.

7. Let $u(r) = \sum_{a=1}^n N_a(r)d_a$, where (by construction) $d_a = u(r_a)$. The function u is capable of exactly representing the powers of r used in deriving the N_a 's. (The d_a 's need only be set accordingly.) In practice, it is usually desired that the same powers of the physical coordinate, $x = x(r)$, be representable. This will be attained if $x = x(r)$ is affine (i.e., of the form $x(r) = c_1 + c_2r$). There are two ways of achieving this in practice.

The *first* depends upon 1 and r being present among the powers of r used in deriving the shape functions. In this case the isoparametric concept may be invoked; that is, we take $x(r) = \sum_{a=1}^n N_a(r)x_a$ and equally space the nodal coordinates (i.e., x_a 's in x -space).

The *second* procedure does not require that 1 and r be present, but it involves an additional set of shape function calculations, so it is less efficient. In this case, we take $x(r) = \sum_{a=1}^n P_a(r)x_a$, where the P_a 's are the standard polynomial shape functions, and the x_a 's are, again, equally spaced.

To see the necessity of adopting the second procedure when 1 and r are absent from the N_a 's, we need only consider the two-node element of Remark 6. If the isoparametric concept is adopted, then u varies linearly with x (rather than x^α). To see this we note that if $x_2 \neq x_1$, then we can write $d_a = c_1 + c_2x_a$, where the c 's are constants. Consequently

$$\begin{aligned} u &= \sum_{a=1}^2 N_a d_a \\ &= \sum_{a=1}^2 N_a (c_1 + c_2 x_a) \\ &= c_1 \left(\sum_{a=1}^2 N_a \right) + c_2 \left(\sum_{a=1}^2 N_a x_a \right) \\ &= c_1 + c_2 x \end{aligned} \tag{3.II.19}$$

We shall now employ the above results in deriving two-dimensional finite element shape functions for modeling line and point singularities.

Example 3

Consider the four-node quadrilateral illustrated in Fig. 3.II.1(a). The shape functions

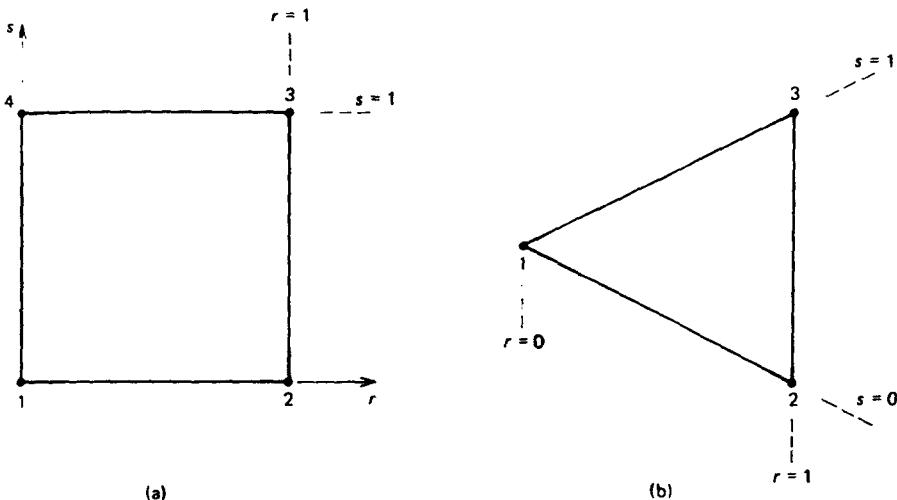


Figure 3.II.1 (a) Four-node quadrilateral element. (b) Three-node triangular element formed by degenerating the four-node quadrilateral.

may be constructed from products of linear polynomial shape functions in the s -direction and the shape functions of Remark 6 in the r -direction, viz.,

$$N_1(r, s) = (1 - r^{\alpha})(1 - s) \quad (3.II.20)$$

$$N_2(r, s) = r^{\alpha}(1 - s) \quad (3.II.21)$$

$$N_3(r, s) = r^{\alpha}s \quad (3.II.22)$$

$$N_4(r, s) = (1 - r^{\alpha})s \quad (3.II.23)$$

Thus $u(r, s) = \sum_{a=1}^4 N_a(r, s)d_a$ is capable of exactly representing a line singularity of order r^{α} along edge 14. By virtue of Remark 7, the standard polynomial shape functions must be employed to define the geometry. That is $x(r, s) = \sum_{a=1}^4 P_a(r, s)x_a$, where

$$P_1(r, s) = (1 - r)(1 - s) \quad (3.II.24)$$

$$P_2(r, s) = r(1 - s) \quad (3.II.25)$$

$$P_3(r, s) = rs \quad (3.II.26)$$

$$P_4(r, s) = (1 - r)s \quad (3.II.27)$$

This element was first proposed by Akin [38].

Example 4

A three-node triangular element (see Fig. 3.II.1(b)) may be developed by combining the shape functions associated with nodes 1 and 4 in the previous example, viz.,

$$N_1(r, s) \leftarrow N_1(r, s) + N_4(r, s) = 1 - r^{\alpha} \quad (3.II.28)$$

$$P_1(r, s) \leftarrow P_1(r, s) + P_4(r, s) = 1 - r \quad (3.II.29)$$

The shape functions of nodes 2 and 3 remain the same. This element is capable of exactly

representing a point singularity of order r^α . Similar interpolations were used in [33] and [39].

Example 5

Consider the nine-node Lagrange-type quadrilateral illustrated in Fig. 3.II.2(a). The shape functions shall be constructed from products of the three-node one-dimensional shape functions of Example 1. That is, we define $N_a(r)$, $a = 1, 2, 3$, by (3.II.7)—(3.II.9) and $P_a(s)$ by the same expressions, with r replaced by s and $\alpha = 2$. The two-dimensional shape functions may then be defined as follows:

$$N_1(r, s) = N_1(r)P_1(s) \quad (3.II.30)$$

$$N_2(r, s) = N_3(r)P_1(s) \quad (3.II.31)$$

$$N_3(r, s) = N_3(r)P_3(s) \quad (3.II.32)$$

$$N_4(r, s) = N_1(r)P_3(s) \quad (3.II.33)$$

$$N_5(r, s) = N_2(r)P_1(s) \quad (3.II.34)$$

$$N_6(r, s) = N_3(r)P_2(s) \quad (3.II.35)$$

$$N_7(r, s) = N_2(r)P_3(s) \quad (3.II.36)$$

$$N_8(r, s) = N_1(r)P_2(s) \quad (3.II.37)$$

$$N_9(r, s) = N_2(r)P_2(s) \quad (3.II.38)$$

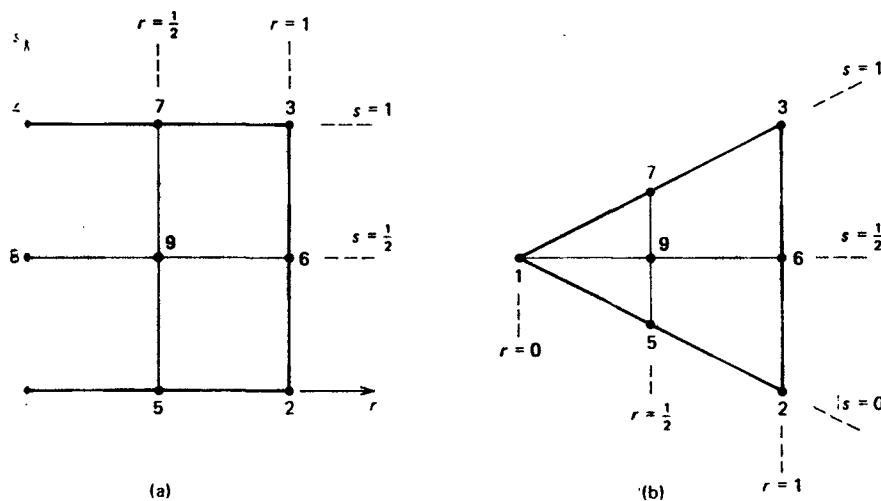


Figure 3.II.2 (a) Nine-node Lagrange quadrilateral element. (b) Seven-node triangular element formed by degenerating the nine-node Lagrange quadrilateral.

In this case $u(r, s) = \sum_{a=1}^9 N_a(r, s)d_a$ is capable of exactly representing a line singularity of order r^α along edge 14. Furthermore, the following monomials may be exactly represented: 1, r , s , r^α , rs , s^2 , $r^\alpha s$, $s^2 r$, and $r^\alpha s^2$. By virtue of the presence of 1, r and s , either the isoparametric concept may be employed to define the geometry (i.e., $x(r, s) = \sum_{a=1}^9 N_a(r, s)x_a$), or recourse may be made to the standard Lagrange poly-

nomials (i.e., $\mathbf{x}(r, s) = \sum_{a=1}^9 P_a(r, s)x_a$, where the $P_a(r, s)$'s may be obtained from (3.II.30) through (3.II.38) by replacing $N_a(r)$ by $P_a(r)$, $a = 1, 2, 3$).

Remark

More complicated two- and three-dimensional elements developed along the lines sketched here are presented in [37].

REFERENCES

Section 3.1

1. G.P. Bazeley, Y.K. Cheung, B.M. Irons, and O.C. Zienkiewicz, "Triangular Elements in Bending—Conforming and Non-conforming Solutions," *Proceedings of the Conference on Matrix Methods in Structural Mechanics*, Wright-Patterson Air Force Base, Ohio, 1965.
2. G. Strang and G.J. Fix, *An Analysis of the Finite Element Method*. Englewood Cliffs, N.J.: Prentice-Hall, 1973.
3. O. C. Zienkiewicz, *The Finite Element Method*. London: McGraw-Hill, 1977.

Section 3.2

4. I.C. Taig, *Structural Analysis by the Matrix Displacement Method*, English Electric Aviation Report No. S017, 1961.
5. J.H. Argyris and S. Kelsey, *Energy Theorems and Structural Analysis*. London: Butterworths, 1960 (originally published in a series of articles in *Aircraft Engineering*, 1954–55).

Section 3.3

6. I.C. Taig, *Structural Analysis by the Matrix Displacement Method*, English Electric Aviation Report No. S017, 1961.
7. B. M. Irons, "Engineering Application of Numerical Integration in Stiffness Method," *Journal of the American Institute of Aeronautics and Astronautics*, 14 (1966), 2035–2037.
8. S. Lang, *Real Analysis*. Reading Mass.: Addison-Wesley, 1969.

Section 3.4

9. R. Courant, "Variational Methods for the Solution of Problems of Equilibrium and Vibration," *Bulletin of the American Mathematical Society*, 49 (1943), 1–23.
10. M. J. Turner, R. W. Clough, H. C. Martin, and L. P. Topp, "Stiffness and Deflection Analysis of Complex Structures," *Journal of Aeronautical Sciences*, 23, no. 9 (1956), 805–823.

Section 3.5

11. R. H. Gallagher, J. Padlog, and P. P. Bijlaard, "Stress Analysis of Heated Complex Shapes," *American Rocket Society Journal*, 32, no. 5 (1962), 700–707.

Section 3.7

12. O. C. Zienkiewicz, *The Finite Element Method*. London: McGraw-Hill, 1977.

Section 3.8

13. A. H. Stroud and D. Secrest, *Gaussian Quadrature Formulas*. Englewood Cliffs, N.J.: Prentice-Hall, 1966.
14. M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*. Washington, D.C.: National Bureau of Standards, 1964.
15. P. C. Hammer and A. H. Stroud, "Numerical Evaluation of Multiple Integrals II," *Mathematical Tables and Aids of Computation*, 12, no. 64 (1958), 272–280.
16. B. M. Irons, "Quadrature Rules for Brick Based Finite Elements," *International Journal of Numerical Methods in Engineering*, 3, no. 2 (1971), 293–294.
17. T. K. Hellen, "Effective Quadrature Rules for Quadratic Solid Isoparametric Finite Elements," *International Journal for Numerical Methods in Engineering*, 4, no. 4 (1972), 597–599.
18. C. Maforano, S. Odorizzi, and R. Vitaliani, "Shortened Quadrature Rules for Finite Elements," *Advances in Engineering Software*, 4, no. 2 (1982).

Section 3.10

19. A. K. Gupta and B. Mohraz, "A Method of Computing Numerically Integrated Stiffness Matrices," *International Journal for Numerical Methods in Engineering*, 5 (1972), 83–89.
20. R. L. Taylor, "Computer Procedures for Finite Element Analysis," Chapter 24 in O. C. Zienkiewicz, *The Finite Element Method*. London: McGraw-Hill, 1977.
21. E. Hinton and D. R. J. Owen, *Finite Element Programming*. London: Academic Press, 1977.
22. K. J. Bathe and E. L. Wilson, *Numerical Methods in Finite Element Analysis*. Englewood Cliffs, N.J.: Prentice-Hall, 1976.
23. D. R. J. Owen and E. Hinton, *Finite Elements in Plasticity: Theory and Practice*. Swansea, U.K.: Pineridge Press, 1980.
24. J. E. Akin, *Application and Implementation of Finite Element Methods*. London: Academic Press, 1982.

Appendix 3.I

25. P. C. Hammer, O. P. Marlowe, and A. H. Stroud, "Numerical Integration over Simplices and Cones," *Mathematical Tables and Aids to Computation*, 10 (1956), 130–137.
26. G. R. Cowper, "Gaussian Quadrature Formulas for Triangles," *International Journal for Numerical Methods in Engineering*, 7 (1973), 405–408.
27. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*. Englewood Cliffs, N.J.: Prentice-Hall, 1973.
28. Y. Jinyun, "Symmetric Gaussian Quadrature Formulae for Tetrahedral Regions," *Computer Methods in Applied Mechanics and Engineering*, 43, no. 3 (1984), 349–353.

Appendix 3.II

29. D. Gartling and E. B. Becker, "Computationally Efficient Finite Element Analysis of

- Viscous Flow Problems," *Computational Methods in Nonlinear Mechanics*, (ed. J. T. Oden). Austin: Texas Institute for Computational Mechanics, 1974, 603–614.
- 30. P. Bettess, "Infinite Elements," *International Journal for Numerical Methods in Engineering*, 11 (1977), 53–64.
 - 31. W. S. Blackburn, "Calculation of Stress Intensity Factors at Crack Tips Using Special Finite Elements," *Mathematics of Finite Elements and Applications*, ed. J. R. Whiteman. London: Academic Press, (1973), 327–336.
 - 32. R. D. Henshell and K. G. Shaw, "Crack Tip Elements are Unnecessary," *International Journal for Numerical Methods in Engineering*, 9 (1975), 495–507.
 - 33. J. E. Akin, "Generation of Elements with Singularities," *International Journal for Numerical Methods in Engineering*, 10 (1976), 1249–1259.
 - 34. R. E. Barnhill and J. R. Whiteman, "Error Analysis of Finite Element Methods with Triangles for Elliptic Boundary Value Problems," *Mathematics of Finite Element Methods and Applications*, ed. J. R. Whiteman. London: Academic Press, (1973), 83–112.
 - 35. A. K. Rao, "Stress Concentrations and Singularities at Interface Corners," *ZAMM*, 51 (1971), 395–406.
 - 36. R. S. Barsoum, "On the Use of Isoparametric Finite Elements in Linear Fracture Mechanics," *International Journal for Numerical Methods in Engineering*, 10 (1976), 25–37.
 - 37. T. J. R. Hughes and J. E. Akin, "Techniques for Developing 'Special' Finite Element Shape Functions with Particular Reference to Singularities," *International Journal for Numerical Methods in Engineering*, 15 (1980), 733–751.
 - 38. J. E. Akin, "Elements for Problems with Line Singularities," *Mathematics of Finite Elements and Applications III*, ed. J. R. Whiteman. London: Academic Press, 1978.
 - 39. D. M. Tracey and T. S. Cook, "Analysis of Power Type Singularities Using Finite Elements," *International Journal for Numerical Methods in Engineering*, 10 (1976), 1249–1259.

4

Mixed and Penalty Methods, Reduced and Selective Integration, and Sundry Variational Crimes

4.1 "BEST APPROXIMATION" AND ERROR ESTIMATES: WHY THE STANDARD FEM USUALLY WORKS AND WHY SOMETIMES IT DOES NOT

In this section we shall establish the convergence of the finite element methodology considered so far and show, in addition, that in a sense it is an optimal technique. The main results require some mathematical preliminaries that are presented in Appendix 4.I for the uninitiated reader. We conclude the section with a caution regarding the application of these ideas to certain classes of problems.

We assume the *exact problem* can be written in the following abstract variational, or weak, form:¹ Find $u \in \mathcal{S}$ such that for all $w \in \mathcal{V}$,

$$a(w, u) = (w, f) + (w, h)_\Gamma \quad (4.1.1)$$

Recall from Chapter 2 that functions in \mathcal{S} satisfy the given essential boundary conditions, and functions in \mathcal{V} satisfy corresponding *homogeneous* essential boundary conditions.

The *approximate finite element problem* can be stated analogously: Find $u^h \in \mathcal{S}^h$ such that for all $w^h \in \mathcal{V}^h$,

$$a(w^h, u^h) = (w^h, f) + (w^h, h)_\Gamma \quad (4.1.2)$$

Throughout we shall assume that

- i. $\mathcal{S}^h \subset \mathcal{S}$ and $\mathcal{V}^h \subset \mathcal{V}$.
- ii. $a(\cdot, \cdot)$, (\cdot, \cdot) and $(\cdot, \cdot)_\Gamma$ are symmetric and bilinear.

¹With appropriate specialization, this includes all problems considered so far.

iii. $a(\cdot, \cdot)$ and $\|\cdot\|_m$ define *equivalent norms* on \mathcal{V} , that is,

$$c_1 \|w\|_m \leq a(w, w)^{1/2} \leq c_2 \|w\|_m \quad (4.1.3)^2$$

where

$$\|w\|_m = \left[\int_{\Omega} \underbrace{(w_i w_i + w_{i,j} w_{i,j} + \cdots + w_{i,k \dots i} w_{i,k \dots i})}_{m \text{ indices}} d\Omega \right]^{1/2} \quad (4.1.4)$$

and c_1 and c_2 are constants independent of w . In (4.1.3), m is the order of derivatives appearing in $a(\cdot, \cdot)$. For example, in heat conduction and elasticity, $m = 1$, whereas in Bernoulli-Euler beam theory $m = 2$. Equation (4.1.4) defines the *m th Sobolev norm of w* (i.e., H^m norm). Also, $a(w, w)^{1/2}$ is called the (*strain*) *energy norm* and $a(\cdot, \cdot)$ the (*strain*) *energy inner product*.

Theorem. Let $e = u^h - u$ denote the *error* in the finite element approximation.

- a. $a(w^h, e) = 0$ for all $w^h \in \mathcal{V}^h$.
- b. $a(e, e) \leq a(U^h - u, U^h - u)$ for all $U^h \in \mathcal{S}^h$.

Part (a) means that the error is orthogonal to the subspace $\mathcal{V}^h \subset \mathcal{V}$. Another way of putting this is to say “ u^h is the projection of u onto \mathcal{S}^h with respect to $a(\cdot, \cdot)$.”

Part (b) means that there is no member of \mathcal{S}^h that is a better approximation to u (with respect to the energy norm) than u^h , the solution of the Galerkin finite element problem. This is referred to as the *best approximation property*. It may be interpreted as meaning that the approximate solution is a least-squares best fit of the exact solution in terms of $a(\cdot, \cdot)$. For the problem classes emphasized in previous chapters, this means that the m th derivatives of u^h best fit the m th derivatives of u in a weighted sense. (In elasticity, this means the accuracy of strains, or stresses, is optimized.)

Proof

- a. Because $\mathcal{V}^h \subset \mathcal{V}$, we may write the exact equation as

$$a(w^h, u) = (w^h, f) + (w^h, b)_\Gamma$$

for all $w^h \in \mathcal{V}^h$. Subtracting this equation from (4.1.2), and using the bilinearity of $a(\cdot, \cdot)$, results in

$$0 = a(w^h, u^h) - a(w^h, u) = a(w^h, e)$$

- b. Let $w^h \in \mathcal{V}^h$. Then, by the symmetry and bilinearity of $a(\cdot, \cdot)$,

$$a(e + w^h, e + w^h) = a(e, e) + \underbrace{2a(w^h, e)}_{0 \text{ by part (a)}} + a(w^h, w^h)$$

²This condition is verified for the cases considered so far in Appendix 4.I.

which, by virtue of $a(\mathbf{w}^h, \mathbf{w}^h) \geq 0$, implies

$$a(\mathbf{e}, \mathbf{e}) \leq a(\mathbf{e} + \mathbf{w}^h, \mathbf{e} + \mathbf{w}^h)$$

Any $\mathbf{U}^h \in \mathcal{S}^h$ can be written as

$$\mathbf{U}^h = \mathbf{u}^h + \mathbf{w}^h$$

for some $\mathbf{w}^h \in \mathcal{V}^h$. Because

$$\begin{aligned}\mathbf{e} + \mathbf{w}^h &= \mathbf{u}^h - \mathbf{u} + \mathbf{w}^h \\ &= \mathbf{U}^h - \mathbf{u}\end{aligned}$$

we immediately obtain the desired result:

$$\begin{aligned}a(\mathbf{e}, \mathbf{e}) &\leq a(\mathbf{e} + \mathbf{w}^h, \mathbf{e} + \mathbf{w}^h) \\ &= a(\mathbf{U}^h - \mathbf{u}, \mathbf{U}^h - \mathbf{u})\end{aligned}\blacksquare$$

Corollary. ("Pythagorean Theorem")

Assume $\mathcal{S}^h = \mathcal{V}^h$ (i.e., the essential boundary conditions are homogeneous).

Then

$$a(\mathbf{u}, \mathbf{u}) = a(\mathbf{u}^h, \mathbf{u}^h) + a(\mathbf{e}, \mathbf{e})$$

Proof. The hypothesis and part (a) imply

$$a(\mathbf{u}^h, \mathbf{e}) = 0$$

Thus

$$\begin{aligned}a(\mathbf{u}, \mathbf{u}) &= a(\mathbf{u}^h - \mathbf{e}, \mathbf{u}^h - \mathbf{e}) \\ &= a(\mathbf{u}^h, \mathbf{u}^h) - \underbrace{2a(\mathbf{u}^h, \mathbf{e})}_{0} + a(\mathbf{e}, \mathbf{e})\end{aligned}\blacksquare$$

Remarks

- Rearranging this result in the form $a(\mathbf{e}, \mathbf{e}) = a(\mathbf{u}, \mathbf{u}) - a(\mathbf{u}^h, \mathbf{u}^h)$ reveals that *the energy of the error equals (minus) the error of the energy*.
- It is a direct consequence of the corollary that

$$a(\mathbf{u}^h, \mathbf{u}^h) \leq a(\mathbf{u}, \mathbf{u})$$

that is, *the approximate solution underestimates the strain energy*.

Exercise 1. Assume $\mathbf{g} = \mathbf{k} = \mathbf{0}$ and $f(x) = \delta(x - \bar{x}_i)e_i$, i.e., a Dirac delta function in the direction e_i located at \bar{x}_i . Use Remark 2, above, to show that

$$u_i^h(\bar{x}) \leq u_i(\bar{x})$$

(In the parlance of structural mechanics, the “displacement under the load” is underestimated by the Galerkin finite element solution.)

Exercise 2. (The Principle of Minimum Potential Energy)

Let

$$U_\epsilon = \mathbf{u} + \epsilon \mathbf{w} \quad (4.1.5)$$

where $\epsilon \in \mathbb{R}$. Note every member of \mathcal{S} can be represented in the form (4.1.5) for some $\mathbf{w} \in \mathcal{V}$ and $\epsilon \in \mathbb{R}$. Define the *potential energy* by

$$I(\mathbf{U}_\epsilon) = \frac{1}{2} a(\mathbf{U}_\epsilon, \mathbf{U}_\epsilon) - (\mathbf{U}_\epsilon, \mathbf{f}) - (\mathbf{U}_\epsilon, \mathbf{h})_\Gamma$$

Establish the following results:

- a. The potential energy is *stationary* (i.e., $(dI(\mathbf{U}_\epsilon)/d\epsilon)|_{\epsilon=0} = 0$) if and only if the variational equation (4.1.1) is satisfied.
- b. The potential energy is *minimized* at \mathbf{u} , that is, $I(\mathbf{u}) \leq I(\mathbf{u} + \epsilon \mathbf{w})$ for all $\mathbf{w} \in \mathcal{V}$ and $\epsilon \in \mathbb{R}$. (*Hint:* Use part (a) and show that $(d^2 I(\mathbf{U}_\epsilon)/d\epsilon^2)|_{\epsilon=0} = a(\mathbf{w}, \mathbf{w}) \geq 0$.)

(Note that counterparts of (a) and (b) can be established for the Galerkin formulation (4.1.2) if \mathbf{u} , \mathbf{w} , \mathbf{U}_ϵ , \mathcal{S} , and \mathcal{V} are replaced by \mathbf{u}^h , \mathbf{w}^h , \mathbf{U}_ϵ^h , \mathcal{S}^h , and \mathcal{V}^h , respectively.)

- c. *The approximate solution overestimates the potential energy*, i.e.,

$$I(\mathbf{u}^h) \geq I(\mathbf{u})$$

Hint: This follows immediately from $\mathcal{S}^h \subset \mathcal{S}$.

Part (a) of the problem indicates that the variational equations are derivable from the derivative of a potential (i.e., the potential energy). When this is the case, one says that a *variational principle* exists for the equations. Vainberg's theorem [1] asserts that *the existence of a variational principle is equivalent to the symmetry of the bilinear form $a(\cdot, \cdot)$* . The subjects of variational principles and methods are fascinating ones, which permeate the fields of solid and structural mechanics (e.g., see [2–4]). It must be emphasized, however, that the *finite element methodology presented is in no way dependent upon the existence of a variational principle*. On the other hand, the best approximation property is intimately linked with the fact that the weak forms considered so far emanate from the minimization of a potential.

Standard Error Estimates in Sobolev Norms

The finite element function spaces discussed in Chapter 3 are endowed with an approximation property that may be stated as follows: Given a function \mathbf{u} possessing r square integrable generalized derivatives³, i.e.,

³One says that \mathbf{u} is of class H^r , or $\mathbf{u} \in H^r$.

$$\int_{\Omega} (u_i u_i + \underbrace{u_{i,j} u_{i,j}}_{r \text{ indices}} + \cdots + \underbrace{u_{i,jk\dots l} u_{i,jk\dots l}}_{r \text{ indices}}) d\Omega < \infty$$

then there exists a function $U^h \in \mathcal{S}^h$ (sometimes called the *interpolate*) such that

$$\|u - U^h\|_m \leq c h^\alpha \|u\|_r \quad (4.1.6)$$

where c is a constant *independent of u and h* , $\alpha = \min(k+1-m, r-m)$, k is the degree of complete polynomial appearing in the element shape functions, and h is the *mesh parameter*, a scalar characterizing the refinement of the finite element mesh. The mesh parameter may be taken to be the diameter of the largest element in the mesh (see Fig. 4.1.1). A collection of finite element spaces $\{\mathcal{S}^h\}$ (i.e., meshes parameterized by h) possessing the approximation property (4.1.6) is called *k, m -regular*. See Appendix 4.I for elaboration.

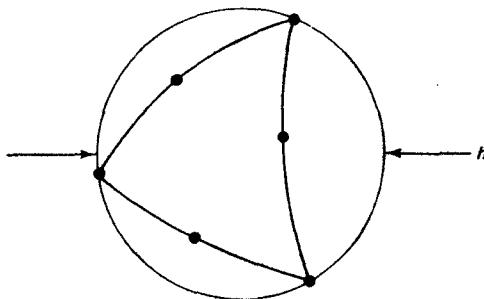


Figure 4.1.1 Schematic of the mesh parameter; h is the diameter of the smallest circle that contains the largest element of the mesh.

Theorem. The approximation result (4.1.6) and the best approximation property (part (b) of the last theorem) enable us to establish *the fundamental error estimate for the elliptic boundary-value problem*:

$$\|e\|_m \leq \bar{c} h^\alpha \|u\|_r$$

where \bar{c} is a constant independent of u and h .

Proof

$$\begin{aligned} \|e\|_m &\leq \frac{1}{c_1} a(e, e)^{1/2} \\ &\leq \frac{1}{c_1} a(u - U^h, u - U^h)^{1/2} \\ &\leq \frac{c_2}{c_1} \|u - U^h\|_m \end{aligned}$$

(continued)

$$\leq \bar{c} h^\alpha \|u\|_r$$

where $\bar{c} = cc_2/c_1$.

Lines 1 and 3 follow from the assumption that $a(\cdot, \cdot)^{1/2}$ and $\|\cdot\|_m$ define equivalent norms, i.e., (4.1.3). Line 2 follows from part (b) of the previous theorem. Line 4 follows from (4.1.6) by choosing U^h to be the interpolate. (The choice of U^h is arbitrary by virtue of part (b) of the previous theorem.) ■

Remarks

1. As long as $k + 1$ and r are greater than m , we have optimal convergence in the H^m norm.

2. Assume u is smooth in the sense that $u \in H^{k+1}$. Then the error satisfies

$$\|e\|_m \leq c h^{k+1-m} \|u\|_{k+1}$$

This is sometimes referred to as the *standard error estimate*. (Remember that k is the degree of polynomial completeness of the element shape functions and m is the order of the highest derivatives appearing in the energy expression.)

3. Error estimates in lower H^s -norms, $0 \leq s \leq m$, can be developed by the *Aubin-Nitsche method* (see [5–7] for descriptions). Assuming $u \in H^{k+1}$, the main result is

$$\|e\|_s \leq c h^\beta \|u\|_{k+1}$$

where c is a constant, independent of u and h , and $\beta = \min(k + 1 - s, 2(k + 1 - m))$. Thus the finite element approximation is optimal in the H^s -norm for all s such that $0 \leq s \leq m$. For example, let $k = m = 1$. Then

$$\|e\|_0 \leq c h^2 \|u\|_2$$

$$\|e\|_1 \leq c h \|u\|_2$$

Using mechanics terminology, the former relation gives the convergence rate in the L_2 -norm for “displacements,” and the latter gives the convergence rate in the L_2 -norm for “displacement gradients” (i.e., strains or stresses). Increasing k increases the order of convergence, whereas increasing m decreases the order of convergence.

Exercise 3. Use the result of Remark 3 to determine the rates of convergence of e in L_2 and H^1 when $m = 1$ and $k = 2$ (i.e., quadratic elements).

Exercise 4. Determine the convergence rates of e in L_2 , H^1 , and H^2 for Bernoulli-Euler beam theory ($m = 2$) and Hermite cubic shape functions ($k = 3$).

Effect of Numerical Quadrature

The previously developed theory assumes that we rigorously adhere to the Galerkin recipe. In particular, this means that all integrals need to be calculated exactly. As indicated in Chapter 3, this would be virtually impossible for isoparametric elements in all but the simplest configurations. Because numerically integrated stiffnesses represent the rule rather than the exception, it is important to understand the accuracy implications of approximate numerical integration. A theory has been presented in [5] that provides *sufficient* conditions for quadrature rules to maintain the full rate of convergence of the exactly integrated formulation. A summary of the main results follows: Let \bar{k} denote the order of the highest-order monomial present in the element shape functions—for example, for the bilinear quadrilateral $\bar{k} = 2$, due to the presence of $\xi\eta$; for the biquadratic quadrilateral $\bar{k} = 4$, due to $\xi^2\eta^2$; for the trilinear brick $\bar{k} = 3$, due to $\xi\eta\zeta$, and so on. The Pascal triangle can be consulted to determine \bar{k} for standard elements. The full rate of convergence in the energy norm of the exactly integrated procedure is attained if the quadrature rule is capable of exactly integrating all monomials through degree $\bar{k} + k - 2m$. First-order convergence is maintained if the rule is exact for monomials through degree $\bar{k} - m$.

Caution. Sufficiently accurate quadrature rules turn out to be of lower order than we might expect. Low-order rules can have a deleterious side effect: *rank deficiency*. This issue is discussed further toward the end of this chapter. For now it is sufficient to note that rank deficiency is an independent issue, which must also be considered in selecting a suitable quadrature rule.

Example 1

For the bilinear quadrilateral in heat conduction, or elasticity, full rate of convergence is maintained if the rule exactly integrates polynomials through degree $\bar{k} + k - 2m = 2 + 1 - 2 = 1$. So, for example, the one-point Gaussian rule is adequate from the standpoint of accuracy (but see Sec. 4.6 concerning rank deficiency!). Likewise, for the eight-node serendipity quadrilateral, $\bar{k} + k - 2m = 3 + 2 - 2 = 3$; thus 2×2 Gauss is required. Because $\bar{k} - m = 3 - 1 = 2$, the 2×2 Gauss rule is needed to maintain first-order convergence.

Example 2

One-dimensional elements, triangles, and tetrahedra typically employ complete polynomials. Thus $\bar{k} = k$, and so the rule must be capable of integrating polynomials of degree $2(k - m)$ to maintain full rate of convergence.

Remark

The quadrature rule determined by the above considerations can be applied to other element integrals (e.g., the body force integral) without altering the conclusions. This becomes important later on when we introduce mass matrices in dynamics.

Discussion

The preceding mathematical results should provide a degree of confidence in the theoretical soundness of the finite element method. In fact, in practice, for typical problems, the results obtained for the Galerkin finite element method generally are very good. The best approximation property is a significant guarantee of reasonable behavior even when very crude meshes are employed.

What could possibly go wrong? Well, suppose the best approximation is not very good. That is, there is no function in \mathcal{S}^h that is a good approximation to \mathbf{u} . This can happen and, for typical finite elements which we have considered so far, does for **constrained media problems** such as ones involving incompressible, or nearly incompressible, behavior. In these cases the approximation may be so poor that it is practically useless. Fortunately, an alternative finite element formulation accommodates successful approximations. This is dealt with in the following sections.

INCOMPRESSIBLE ELASTICITY AND STOKES FLOW

Many problems of physical importance involve motions that essentially preserve volumes locally. That is, after deformation each small portion of the medium has the same volume as before deformation. Media that behave in this fashion are termed **incompressible**. Rubber is often modeled as an incompressible elastic material, and many fluid flows are assumed incompressible.

In isotropic linear elasticity the condition of incompressibility may be expressed in terms of Poisson's ratio ν . As ν approaches $\frac{1}{2}$, resistance to volume change is greatly increased assuming resistance to shearing remains constant. This may be seen by calculating the ratio of bulk modulus, B , to shearing modulus, μ ([8], p. 71):

$$\frac{B}{\mu} = \frac{2(1 + \nu)}{3(1 - 2\nu)} \quad (4.2.1)$$

Clearly, as $\nu \rightarrow \frac{1}{2}$, the ratio approaches infinity. The limiting value $\nu = \frac{1}{2}$ thus represents incompressibility.

This limit creates problems in the equations of (compressible) elasticity. Recall that the constitutive equation in the isotropic case may be written as (see eqs. (2.7.1), (2.7.2), and (2.7.31))

$$\sigma_{ij} = \lambda u_{k,k} \delta_{ij} + 2\mu u_{(i,j)} \quad (4.2.2)$$

where

$$\lambda = \frac{2\nu\mu}{1 - 2\nu} \quad (4.2.3)$$

Thus the Lamé parameter, λ , also becomes unbounded in the incompressible limit and an alternative formulation of the theory is therefore necessary. In this formulation the constitutive equation is written as

$$\sigma_{ij} = -p\delta_{ij} + 2\mu u_{(i,j)} \quad (4.2.4)$$

where $p = p(\mathbf{x})$ is the *hydrostatic pressure*. The pressure must be determined as part of the solution to the boundary-value problem and thus represents an additional unknown. The additional equation that needs to be introduced is the *kinematic condition of incompressibility*, namely,

$$\operatorname{div} \mathbf{u} = u_{i,i} = 0 \quad (4.2.5)$$

The boundary-value problem may then be stated as follows:

Incompressible Isotropic Elasticity

Given ℓ_i , q_i , and h_i (as in Sec. 2.7), find the displacement, u_i , and pressure, p , such that

$$\left. \begin{aligned} \sigma_{ij,j} + \ell_i &= 0 \\ u_{i,i} &= 0 \end{aligned} \right\} \quad \text{in } \Omega \quad (4.2.6)$$

$$u_i = q_i \quad \text{on } \Gamma_{q_i} \quad (4.2.8)$$

$$\sigma_{ij}n_j = h_i \quad \text{on } \Gamma_{h_i} \quad (4.2.9)$$

where σ_{ij} is given by (4.2.4).

Remark

In the case of the displacement boundary-value problem, in which $\Gamma = \Gamma_{q_i}$ and $\Gamma_{h_i} = \emptyset$, a consistency condition on q_i follows from incompressibility:

$$\begin{aligned} 0 &= \int_{\Omega} u_{i,i} d\Omega \quad (\text{by (4.2.7)}) \\ &= \int_{\Gamma} u_i n_i d\Gamma \quad (\text{divergence theorem, (2.2.11)}) \\ &= \int_{\Gamma} q_i n_i d\Gamma \quad (\text{by (4.2.8)}) \end{aligned} \quad (4.2.10)$$

The given q_i must satisfy (4.2.10) or else no solution to the boundary-value problem can exist. In the displacement boundary-value problem, the pressure is determinable only up to an arbitrary constant.

Stokes Flow

The equations of Stokes flow are identical to the equations of isotropic incompressible elasticity. Only the physical interpretation of the variables is different. In Stokes flow \mathbf{u} is the velocity of the fluid and μ is the dynamic viscosity. Stokes flow governs highly viscous phenomena, often referred to as *creeping flow*.

Exercise 1. Use (4.2.3) and (2.7.35) to show that the coefficients of the isotropic plane stress constitutive equation remain bounded as $\nu \rightarrow \frac{1}{2}$. Thus a special formulation is *not* required for the incompressible plane stress case.

4.2.1 Prelude to Mixed and Penalty Methods

The subject of finite element approximations to incompressible elasticity problems is replete with so-called mixed and penalty formulations. In order to introduce the essential aspects of these methods in a context that is as simple as possible, we shall consider the example of appending a constraint to a linear algebraic system. Suppose we wish to solve our standard matrix problem

$$\mathbf{K}\mathbf{d} = \mathbf{F} \quad (4.2.11)$$

where, as usual, \mathbf{K} is symmetric and positive-definite, subject to a constraint on one of the degrees of freedom, namely,

$$d_Q = q \quad (4.2.12)$$

where the subscript Q represents the equation number in the global ordering and q is a given constant. Physically, we can think of (4.2.12) as perhaps a modification to an original design. For example, the position of a boundary node may be altered to stiffen a structure for some purpose. The modified problem consisting of (4.2.11) and (4.2.12) may be formulated as a *constrained variational problem*. The essential character of mixed methods is exhibited in this framework. To develop this idea, it is helpful to think of $\mathbf{K}\mathbf{d} = \mathbf{F}$ as arising from the minimization of a function:

$$\mathcal{F}(\mathbf{d}) = \frac{\mathbf{d}^T \mathbf{K} \mathbf{d}}{2} - \mathbf{d}^T \mathbf{F} \quad (4.2.13)$$

which is called the *total potential energy function*. The vector that minimizes \mathcal{F} satisfies (4.2.11). This can be seen as follows: Let ϵ be a real parameter. Form the one-parameter family of displacement vectors

$$\mathbf{d} + \epsilon \mathbf{c} \quad (4.2.14)$$

where \mathbf{c} is arbitrary. \mathcal{F} is minimized by \mathbf{d} if

$$0 = \left(\frac{d}{d\epsilon} \mathcal{F}(\mathbf{d} + \epsilon \mathbf{c}) \right)_{\epsilon=0} \quad (4.2.15)$$

for all vectors \mathbf{c} . This calculation is carried out as follows:

$$\begin{aligned} \left(\frac{d}{d\epsilon} \mathcal{F}(\mathbf{d} + \epsilon \mathbf{c}) \right)_{\epsilon=0} &= \left(\frac{d}{d\epsilon} ((\mathbf{d} + \epsilon \mathbf{c})^T \mathbf{K} (\mathbf{d} + \epsilon \mathbf{c}) / 2 - (\mathbf{d} + \epsilon \mathbf{c})^T \mathbf{F}) \right)_{\epsilon=0} \\ &= \mathbf{c}^T \mathbf{K} \mathbf{d} / 2 + \mathbf{d}^T \mathbf{K} \mathbf{c} / 2 - \mathbf{c}^T \mathbf{F} \\ &= \mathbf{c}^T (\mathbf{K} \mathbf{d} - \mathbf{F}) \quad (\text{symmetry of } \mathbf{K}) \end{aligned} \quad (4.2.16)$$

Thus we see that (4.2.16) must be zero for arbitrary \mathbf{c} , which implies $\mathbf{K} \mathbf{d} = \mathbf{F}$. Thus we have equivalent alternative characterizations of \mathbf{d} : It is the solution of problem (4.2.11) and also the minimizer of function (4.2.13). The fact that \mathbf{d} minimizes \mathcal{F} (i.e., solves a “variational problem”) allows us to introduce standard calculus-of-variations methodology in order to formulate the modified problem consisting of (4.2.11) and (4.2.12). For this purpose it is convenient to rephrase (4.2.12) as

$$0 = \mathcal{Q}(\mathbf{d}) = \mathbf{1}_Q^T \mathbf{d} - q \quad (4.2.17)$$

where

$$\mathbf{1}_Q^T = \langle 0 \dots 0 \underset{\substack{\uparrow \\ Q\text{th term}}}{1} 0 \dots 0 \rangle \quad (4.2.18)$$

The angle brackets signify a row vector. Clearly, (4.2.17) is equivalent to (4.2.12).

Lagrange-Multiplier Method

Consider the following function:

$$\mathcal{H}(\mathbf{d}, m) = \mathcal{F}(\mathbf{d}) + m \mathcal{Q}(\mathbf{d}) \quad (4.2.19)$$

where m is a scalar parameter called the *Lagrange multiplier*. Rendering (4.2.19) “stationary” is equivalent to satisfaction of the constrained problem (i.e., eqs. (4.2.11) and (4.2.12)). The condition of stationarity is that

$$0 = \left(\frac{d}{d\epsilon} \mathcal{H}(\mathbf{d} + \epsilon \mathbf{c}, m + \epsilon l) \right)_{\epsilon=0} \quad (4.2.20)$$

for all values of the vector \mathbf{c} and scalar l . The multiplier m plays the role of the force that maintains the constraint (4.2.12). It is an additional unknown corresponding to the additional equation that must be satisfied, namely (4.2.12). Let us carry out the calculation indicated in (4.2.20):

$$\begin{aligned} 0 &= \left(\frac{d}{d\epsilon} (\mathcal{F}(\mathbf{d} + \epsilon \mathbf{c}) + (m + \epsilon l) \mathcal{Q}(\mathbf{d} + \epsilon \mathbf{c})) \right)_{\epsilon=0} \\ &= \mathbf{c}^T (\mathbf{K}\mathbf{d} - \mathbf{F}) + l \mathcal{Q}(\mathbf{d}) + m \mathbf{1}_Q^T \mathbf{c} \\ &= \mathbf{c}^T (\mathbf{K}\mathbf{d} + m \mathbf{1}_Q - \mathbf{F}) + l (\mathbf{1}_Q^T \mathbf{d} - q) \end{aligned} \quad (4.2.21)$$

Due to the arbitrariness of \mathbf{c} and l , (4.2.21) implies

$$\mathbf{K}\mathbf{d} + m \mathbf{1}_Q = \mathbf{F} \quad (4.2.22)$$

and

$$\mathbf{1}_Q^T \mathbf{d} = q \quad (4.2.23)$$

or, equivalently,

$$\begin{bmatrix} \mathbf{K} & \mathbf{1}_Q \\ \mathbf{1}_Q^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{d} \\ m \end{Bmatrix} = \begin{Bmatrix} \mathbf{F} \\ q \end{Bmatrix} \quad (4.2.24)$$

This is the equation system which determines \mathbf{d} and m , the solution of the modified problem. To account for the constraint, the original system, (4.2.11), needed to be modified (see (4.2.22)). This is physically reasonable.

The equation system (4.2.24) is a prototype of a *mixed method*, i.e., one in which there are both displacements and forces as unknowns. Its importance in regard

to problems of incompressibility—a constraint—is that we have from the outset both displacements and pressure—the force-like variable—as unknowns. Consequently, we may ultimately anticipate algebraic systems having features in common with (4.2.24). Note that the coefficient matrix in (4.2.24) is symmetric but not positive-definite. Symmetry follows from the definition of a transposed partitioned matrix:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^T = \begin{bmatrix} \mathbf{A}^T & \mathbf{C}^T \\ \mathbf{B}^T & \mathbf{D}^T \end{bmatrix} \quad (4.2.25)$$

Failure of the positive-definiteness condition (see (ii) of the definition in Sec. 1.9) follows from:

$$\begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}^T \begin{bmatrix} \mathbf{K} & \mathbf{1}_Q \\ \mathbf{1}_Q^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}^T \begin{bmatrix} \mathbf{1}_Q \\ 0 \end{bmatrix} = 0 \quad (4.2.26)$$

Penalty Method

The penalty method of formulating the constrained problem may be viewed as an approximation to the Lagrange-multiplier method. In the penalty formulation, the Lagrange multiplier is approximated as follows:

$$m \equiv k \mathcal{Q}(\mathbf{d}) \quad (4.2.27)$$

where k is a large positive number having the physical interpretation of a stiff spring constant. Note that k is *not* an unknown. With this approximation we may define a new function,

$$\mathcal{J}(\mathbf{d}) = \mathcal{F}(\mathbf{d}) + \frac{k}{2} \mathcal{Q}(\mathbf{d})^2 \quad (4.2.28)$$

whose minimum defines an approximate solution to the constrained problem. Calculating, as before,

$$\begin{aligned} 0 &= \left(\frac{d}{d\epsilon} \mathcal{J}(\mathbf{d} + \epsilon \mathbf{c}) \right)_{\epsilon=0} \\ &= \left(\frac{d}{d\epsilon} \left(\mathcal{F}(\mathbf{d} + \epsilon \mathbf{c}) + \frac{k}{2} \mathcal{Q}(\mathbf{d} + \epsilon \mathbf{c})^2 \right) \right)_{\epsilon=0} \\ &= \mathbf{c}^T (\mathbf{K}\mathbf{d} - \mathbf{F}) + k \mathcal{Q}(\mathbf{d}) \mathbf{1}_Q^T \mathbf{c} \\ &= \mathbf{c}^T \left((\mathbf{K} + k \mathbf{1}_Q \mathbf{1}_Q^T) \mathbf{d} - (\mathbf{F} + k \mathbf{q} \mathbf{1}_Q) \right) \end{aligned} \quad (4.2.29)$$

which implies

$$(\mathbf{K} + k \mathbf{1}_Q \mathbf{1}_Q^T) \mathbf{d} = \mathbf{F} + k \mathbf{q} \mathbf{1}_Q \quad (4.2.30)$$

Explicating (4.2.30) yields

$$\underbrace{\left(\mathbf{K} + \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & k & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \right)}_{k \text{ appears in the } Q\text{th diagonal entry}} \mathbf{d} = \mathbf{F} + \underbrace{\begin{Bmatrix} 0 \\ \vdots \\ 0 \\ kq \\ 0 \\ \vdots \\ 0 \end{Bmatrix}}_{kq \text{ appears in the } Q\text{th row}} \quad (4.2.31)$$

Thus it is clear that as $k \rightarrow \infty$, $d_Q \rightarrow q$ (i.e., the constraint is satisfied) and thus the right-hand side of (4.2.27) is an approximation to the constraining force (Lagrange multiplier). The larger the k , the better the approximation. This formulation is suggestive of general ways of approximating constrained problems. In the context of elasticity, one would interpret ideas like this as approximating the incompressible case by a slightly compressible formulation. That is, one for which the ratio B/μ , or equivalently λ/μ , is very large (see (4.2.1) and (4.2.3)).

4.3 A MIXED FORMULATION OF COMPRESSIBLE ELASTICITY CAPABLE OF REPRESENTING THE INCOMPRESSIBLE LIMIT

It is desirable to have a formulation of isotropic elasticity that is valid for both compressible and incompressible behavior. To this end we may introduce a pair of constitutive equations

$$\sigma_{ij} = -p\delta_{ij} + 2\mu u_{(i,j)} \quad (4.3.1)$$

$$0 = u_{i,i} + \frac{p}{\lambda} \quad (4.3.2)$$

where the pressure parameter, p , is viewed as an independent unknown. If $\nu = \frac{1}{2}$, (4.3.2) becomes the incompressibility condition and p is the hydrostatic pressure as in the previous section. If $\nu < \frac{1}{2}$, p may be eliminated from (4.3.1) by way of (4.3.2) to obtain the constitutive equation of the compressible case, namely,

$$\sigma_{ij} = \lambda u_{k,k} \delta_{ij} + 2\mu u_{(i,j)} \quad (4.3.3)$$

Thus we see that (4.3.1) and (4.3.2) are valid in both the compressible and incompressible cases.

Remark

Note that p may be interpreted as the hydrostatic pressure *only* in the incompressible case. In general, the hydrostatic pressure is $-\sigma_{ii}/3$. Thus in the compressible case, (4.3.3) yields

$$-\sigma_{ii}/3 = -\underbrace{(\lambda + 2\mu/3)}_B u_{i,i} \quad (4.3.4)$$

whereas (4.3.2) gives

$$p = -\lambda u_{i,i} \quad (4.3.5)$$

If $\mu \ll \lambda$ (nearly incompressible case), (4.3.5) is a good approximation to (4.3.4). On the other hand, in the incompressible limit

$$p = \frac{-\sigma_{ii}}{3} \quad (4.3.6)$$

follows directly from (4.3.1).

4.3.1 Strong Form

$$(S) \left\{ \begin{array}{l} \text{With } f_i, q_i, \text{ and } h_i \text{ given as before, we wish to find } u_i \text{ and } p \text{ such that} \\ \left. \begin{array}{l} \sigma_{ij,j} + f_i = 0 \\ u_{i,i} + p/\lambda = 0 \end{array} \right\} \quad \text{in } \Omega \\ u_i = q_i \quad \text{on } \Gamma_{q_i} \\ \sigma_{ij} n_j = h_i \quad \text{on } \Gamma_{h_i} \end{array} \right. \begin{array}{l} (4.3.7) \\ (4.3.8) \\ (4.3.9) \\ (4.3.10) \end{array}$$

where σ_{ij} is defined by (4.3.1).

Remark

The formulation presented in this section was first proposed by Herrmann [9]. Generalizations to anisotropic cases were proposed by Taylor, Pister, and Herrmann [10] and Key [11]. The subject of anisotropy and incompressibility is taken up in Sec. 4.6.

4.3.2 Weak Form

The weak formulation of the problem is similar to the one for compressible elasticity (see (2.7.11)) except we need to introduce a term that implies satisfaction of (4.3.8). In addition to the displacement weighting and trial solution spaces (\mathcal{O}_i and \mathcal{S}_i , respectively) a *space of pressures*, \mathcal{P} , is required. The functions in \mathcal{P} are required to be square-integrable (i.e., “ L_2 -functions”; see Appendix 4.I). Because there are no explicit boundary conditions on the pressures, \mathcal{P} suffices as both a trial solution space and a weighting function space.

The weak formulation may then be stated as follows.

Given ℓ_i , g_i , and h_i , as before, find $u_i \in \mathcal{S}_i$ and $p \in \mathcal{P}$, such that for all $w_i \in \mathcal{V}_i$ and $q \in \mathcal{D}$ (pressure weighting function)

$$(W) \quad \left\{ \begin{aligned} & \int_{\Omega} w_{(i,j)} \sigma_{ij} d\Omega - \int_{\Omega} q(u_{i,i} + p/\lambda) d\Omega \\ &= \int_{\Omega} w_i \ell_i d\Omega + \sum_{i=1}^{n_{sd}} \int_{\Gamma_{k_i}} w_i h_i d\Gamma \end{aligned} \right. \quad (4.3.11)$$

where σ_{ij} is given by (4.3.1).

To see what equations are implied by satisfaction of (4.3.11), we need to integrate (4.3.11) by parts (we assume all functions are smooth):

$$\begin{aligned} 0 &= \int_{\Omega} w_i \underbrace{(\sigma_{ij,j} + \ell_i)}_{\text{equilibrium}} d\Omega \\ &+ \int_{\Omega} q \underbrace{(u_{i,i} + p/\lambda)}_{\text{eq. (4.3.8)}} d\Omega \\ &+ \sum_{i=1}^{n_{sd}} \int_{\Gamma_{k_i}} w_i \underbrace{(h_i - \sigma_{ij} n_j)}_{\substack{\text{traction boundary} \\ \text{condition}}} d\Gamma \end{aligned} \quad (4.3.12)$$

The usual arguments enable us to establish that the terms in parentheses in (4.3.12) vanish identically on their respective domains of definition (see Sec. 2.7 for detailed arguments of this type).

The variational equation (4.3.11) may be written in the abstract form:

$$\bar{a}(\mathbf{w}, \mathbf{u}) - (\operatorname{div} \mathbf{w}, p) - \left(q, \operatorname{div} \mathbf{u} + \frac{p}{\lambda} \right) = (\mathbf{w}, \boldsymbol{\ell}) + (\mathbf{w}, \mathbf{h})_{\Gamma} \quad (4.3.13)$$

where $\bar{a}(\cdot, \cdot)$ is the symmetric bilinear form defined by

$$\bar{a}(\mathbf{w}, \mathbf{u}) = \int_{\Omega} w_{(i,j)} \bar{c}_{ijkl} u_{(k,l)} d\Omega \quad (4.3.14)$$

in which

$$\bar{c}_{ijkl} = \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) \quad (4.3.15)$$

Exercise 1. Verify that (4.3.11) and (4.3.13) are equivalent.

Exercise 2. Show that $\bar{a}(\cdot, \cdot)$ satisfies the definition of a symmetric bilinear form (see (1.4.11) and (1.4.13)).

Exercise 3. Show that if $\mu > 0$, then

$$\bar{c}_{ijkl}\psi_{ij}\psi_{kl} \geq 0 \quad (4.3.16)$$

for all symmetric ψ_{ij} and furthermore

$$\bar{c}_{ijkl}\psi_{ij}\psi_{kl} = 0 \quad (4.3.17)$$

if and only if $\psi_{ij} = 0$ (positive definiteness, see (2.7.5) and (2.7.6)).

4.3.3 Galerkin Formulation

Recall that \mathcal{S}^h and \mathcal{V}^h are the finite-dimensional approximations to \mathcal{S} and \mathcal{V} , respectively. Likewise, let \mathcal{P}^h be the finite-dimensional space that approximates \mathcal{P} . The Galerkin formulation may then be stated as follows.

$$(G) \left\{ \begin{array}{l} \text{Given } \mathbf{f}, \mathbf{g}, \text{ and } \mathbf{h}, \text{ as in } (W), \text{ find } \mathbf{u}^h = \mathbf{v}^h + \mathbf{q}^h \in \mathcal{S}^h \text{ and } p^h \in \mathcal{P}^h \text{ such} \\ \text{that for all } \mathbf{w}^h \in \mathcal{V}^h \text{ and } q^h \in \mathcal{P}^h, \\ \boxed{\begin{aligned} \bar{a}(\mathbf{w}^h, \mathbf{v}^h) - (\operatorname{div} \mathbf{w}^h, p^h) - (q^h, \operatorname{div} \mathbf{v}^h + p^h/\lambda) \\ = (\mathbf{w}^h, \mathbf{f}) + (\mathbf{w}^h, \mathbf{h})_\Gamma - \bar{a}(\mathbf{w}^h, \mathbf{q}^h) + (q^h, \operatorname{div} \mathbf{q}^h) \end{aligned}} \end{array} \right. \quad (4.3.18)$$

4.3.4 Matrix Problem

In order to develop the matrix form of the problem we need to introduce interpolatory expansions for p^h . Since p^h just needs to be square-integrable, it may be discontinuous across element boundaries. (Note that no derivatives of p^h , or q^h , appear in the variational equation; see (4.3.18).) Thus a wider range of interpolations is permissible for pressure than for displacements. Possible combinations are illustrated in Fig. 4.3.1. *It must, however, be emphasized that arbitrary combinations of interpolations may lead to poor numerical performance and even nonconvergence.* It is difficult to give intuitive guidelines because seemingly “natural” combinations may fail in practice. The subject of appropriate combinations is dealt with in Sec. 4.5.

Let us denote the pressure interpolation by

$$p^h(\mathbf{x}) = \sum_{\tilde{\lambda} \in \tilde{\eta}} \tilde{N}_{\tilde{\lambda}}(\mathbf{x}) p_{\tilde{\lambda}} \quad (4.3.19)$$

where $\tilde{\eta}$ is the set of pressure node numbers, $\tilde{N}_{\tilde{\lambda}}$ is the pressure shape function associated with pressure node number $\tilde{\lambda}$, and $p_{\tilde{\lambda}}$ is the value of pressure at node number $\tilde{\lambda}$.

Similarly, the pressure weighting function may be expressed as

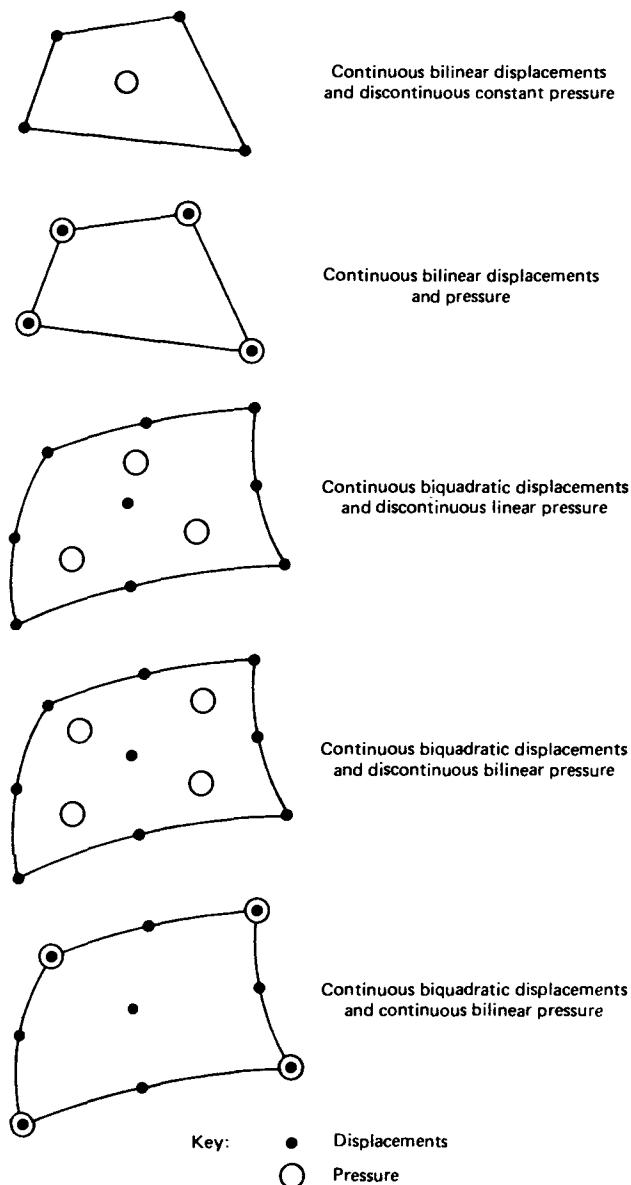


Figure 4.3.1 Examples of possible displacement and pressure interpolations in two dimensions. *Warning:* Not all are effective in practice.

$$q^h(x) = \sum_{\tilde{\lambda} \in \tilde{\eta}} \tilde{N}_{\tilde{\lambda}}(x) q_{\tilde{\lambda}} \quad (4.3.20)$$

Substitution of (4.3.19) and (4.3.20), along with the expressions for v^h , w^h , and g^h (see eqs. (2.8.4) through (2.8.10)) into (4.3.18) leads to the global matrix equation, which can be written in the following partitioned form.

Segregated d , p -form of the matrix equation

$$\boxed{\begin{bmatrix} \bar{K} & G \\ G^T & M \end{bmatrix} \begin{Bmatrix} d \\ p \end{Bmatrix} = \begin{Bmatrix} \bar{F} \\ H \end{Bmatrix}} \quad (4.3.21)$$

The arrays in (4.3.21) and corresponding terms in the variational equations are identified in Table 4.3.1. The matrix G is the discrete gradient operator, G^T is the discrete divergence operator, \bar{K} and M are both symmetric, \bar{K} is positive-definite, and M is negative-definite, except in the case $\nu = \frac{1}{2}$ in which $M = \mathbf{0}$.

TABLE 4.3.1

| Global array | Term in Galerkin equation from which global array emanates |
|--------------|--|
| \bar{K} | $\bar{a}(w^h, v^h)$ |
| G | $-(\operatorname{div} w^h, p^h)$ |
| G^T | $-(q^h, \operatorname{div} v^h)$ |
| M | $-(q^h, p^h/\lambda)$ |
| \bar{F} | $(w^h, \varphi) + (w^h, h)_\Gamma - \bar{a}(w^h, q^h)$ |
| H | $(q^h, \operatorname{div} \varphi^h)$ |

There are several possible ways of solving Eq. (4.3.21). Some of these procedures are described below:

Procedure 1

If $M \neq \mathbf{0}$ (compressible case), then p can be eliminated by expanding (4.3.21),

$$\bar{K}d + Gp = \bar{F} \quad (4.3.22)$$

$$G^T d + Mp = H \quad (4.3.23)$$

Solving (4.3.23) for p ,

$$p = M^{-1}(H - G^T d) \quad (4.3.24)$$

and substituting into (4.3.22):

$$\boxed{\underbrace{(\bar{K} - GM^{-1}G^T)}_K d = \underbrace{\bar{F} - GM^{-1}H}_F} \quad (4.3.25)$$

Equation (4.3.25) has the usual format. Observe that K is symmetric and positive definite. After solving (4.3.25) for d , p may be calculated from (4.3.24).

Procedure 2

If $M = \mathbf{0}$ (incompressible case) the preceding elimination cannot be performed. However, the following procedure may be employed. Solve (4.3.22) for \mathbf{d} , premultiply by \mathbf{G}^T , and employ (4.3.23) to obtain the *discrete Poisson equation for pressure*:

$$\underbrace{(\mathbf{G}^T \bar{\mathbf{K}}^{-1} \mathbf{G})}_{\hat{\mathbf{K}}} \mathbf{p} = \underbrace{\mathbf{G}^T \bar{\mathbf{K}}^{-1} \bar{\mathbf{F}} - \mathbf{H}}_{\hat{\mathbf{F}}} \quad (4.3.26)$$

After \mathbf{p} is obtained by solving (4.3.26), (4.3.22) may be used to obtain \mathbf{d} .

Exercise 4. Generalize (4.3.26) to the case in which $M \neq \mathbf{0}$.

Procedure 3

Both Procedures 1 and 2 are global in nature and are valid whether p^h is continuous or discontinuous. If pressure is *discontinuous* between elements then, in the compressible case, the pressure degrees of freedom may be eliminated on the element level. The element arrays that correspond to the global arrays are notationally defined in Table 4.3.2. In this case we may write the global system in the usual way as an assembly of element arrays, i.e.,

$$\mathbf{K}\mathbf{d} = \mathbf{F} \quad (4.3.27)$$

where

$$\mathbf{K} = \sum_{e=1}^{n_{el}} \mathbf{A}_e (\mathbf{k}^e) \quad (4.3.28)$$

$$\mathbf{F} = \sum_{e=1}^{n_{el}} \mathbf{A}_e (\mathbf{f}^e) \quad (4.3.29)$$

and

$$\mathbf{k}^e = \bar{\mathbf{k}}^e - \mathbf{g}^e (\mathbf{m}^e)^{-1} (\mathbf{g}^e)^T \quad (4.3.30)$$

$$\mathbf{f}^e = \bar{\mathbf{f}}^e - \mathbf{g}^e (\mathbf{m}^e)^{-1} \mathbf{h}^e \quad (4.3.31)$$

Note that (4.3.30) and (4.3.31) are the element analogs of the arrays that appear in (4.3.25). The above, although equivalent to Procedure 1, is much more convenient from a practical standpoint in that only operations on the element level are involved. Likewise, the element vector of nodal pressures, \mathbf{p}^e , can be calculated from the element nodal displacements once the latter are determined. This can be written as

$$\mathbf{p}^e = -(\mathbf{m}^e)^{-1} (\mathbf{g}^e)^T \mathbf{d}^e \quad (4.3.32)$$

where \mathbf{d}^e is the element displacement vector. Recall that \mathbf{d}^e includes specified displacement degrees of freedom (see eqs. (2.9.10) through (2.9.13)). This is the reason why no \mathbf{h}^e -term appears (compare the global counterpart of (4.3.32), i.e., (4.3.24)). By consulting Table 4.3.1, the rationale behind (4.3.32) may be verified.

TABLE 4.3.2

| Global array | Corresponding element array |
|--------------|-----------------------------|
| \bar{K} | \bar{k}^e |
| G | g^e |
| M | m^e |
| F | \bar{f}^e |
| H | h^e |

4.3.5 Definition of Element Arrays

The components of the element arrays introduced in the previous section are given in this section.

$$\bar{k}^e = [\bar{k}_{pq}^e], \quad 1 \leq p, q \leq n_{ee} \quad (4.3.33)$$

$$\bar{k}_{pq}^e = e_i^T \bar{k}_{ab}^e e_j, \quad 1 \leq a, b \leq n_{en} \quad (4.3.34)$$

$$p = n_{ed}(a - 1) + i, \quad q = n_{ed}(b - 1) + j \quad (4.3.35)$$

$$\bar{k}_{ab}^e = \int_{\Omega^e} B_a^T \bar{D} B_b d\Omega \quad (4.3.36)$$

$$\bar{f}^e = \{\bar{f}_p^e\} \quad (4.3.37)$$

$$\bar{f}_p^e = \int_{\Omega^e} N_a f_i d\Omega + \int_{\Gamma_k} N_a h_i d\Gamma - \sum_{q=1}^{n_{ee}} \bar{k}_{pq}^e g_q^e \quad (4.3.38)$$

The preceding formulas are close analogs of those given for elasticity in Chapter 2. The only difference involves the appearance of the matrix \bar{D} instead of D . From (4.3.15), it may be concluded that \bar{D} is the part of D in which λ -terms are omitted. Therefore, we have the following (e.g., see eq. (2.7.34)).

Three dimensions

$$\bar{D} = \mu \begin{bmatrix} 2 & & & \\ & 2 & & \\ & & 2 & \\ & & & 1 \\ & & & & 1 \\ & & & & & 1 \end{bmatrix} \quad (4.3.39)$$

Plane strain

$$\bar{D} = \mu \begin{bmatrix} 2 & & \\ & 2 & \\ & & 1 \end{bmatrix} \quad (4.3.40)$$

Axisymmetry

$$\bar{D} = \mu \begin{bmatrix} 2 & & & \\ & 2 & & \\ & & 1 & \\ & & & 2 \end{bmatrix} \quad (4.3.41)$$

All omitted terms in (4.3.39) through (4.3.41) are zero.

In (4.3.33) through (4.3.38) the indexing pertains to displacement degrees of freedom only. We assume that the element in question possesses \tilde{n}_{en} pressure nodes and that $1 \leq \tilde{a}, \tilde{b} \leq \tilde{n}_{en}$, where \tilde{a} and \tilde{b} are element pressure node numbers. With these we may write

$$\mathbf{m}^\epsilon = [m_{\tilde{a}\tilde{b}}^\epsilon] \quad (4.3.42)$$

$$m_{\tilde{a}\tilde{b}}^\epsilon = \int_{\Omega^\epsilon} \frac{1}{\lambda} \tilde{N}_{\tilde{a}} \tilde{N}_{\tilde{b}} d\Omega \quad (4.3.43)$$

$$\mathbf{g}^\epsilon = [g_{p\tilde{a}}^\epsilon] \quad (4.3.44)$$

$$g_{p\tilde{a}}^\epsilon = - \int_{\Omega^\epsilon} \operatorname{div}(N_a \mathbf{e}_i) \tilde{N}_{\tilde{a}} d\Omega \quad (4.3.45)$$

$$\mathbf{h}^\epsilon = \{h_{\tilde{a}}^\epsilon\} \quad (4.3.46)$$

$$h_{\tilde{a}}^\epsilon = - \sum_{p=1}^{\tilde{n}_{en}} g_{p\tilde{a}}^\epsilon q_p^\epsilon \quad (4.3.47)$$

In (4.3.45), $\operatorname{div}(N_a \mathbf{e}_i)$ is given by the following.

Three dimensions and plane strain

$$\operatorname{div}(N_a \mathbf{e}_i) = N_{a,i} \quad (4.3.48)$$

Axisymmetry (see Sec. 2.12)

$$\operatorname{div}(N_a \mathbf{e}_i) = \begin{cases} N_{a,1} + \frac{N_a}{r} & i = 1 \\ N_{a,2} & i = 2 \end{cases} \quad (4.3.49)$$

The stress “vector” in an element may be computed from the formula (compare eq. (2.9.14)):

$$\boldsymbol{\sigma}(\mathbf{x}) = - \left(\sum_{\bar{a}=1}^{\bar{n}_{en}} \tilde{N}_{\bar{a}}(\mathbf{x}) p_{\bar{a}}^e \right) \mathbf{V} + \overline{\mathbf{D}}(\mathbf{x}) \sum_{a=1}^{n_{en}} B_a(\mathbf{x}) d_a^e \quad (4.3.50)$$

where \mathbf{V} is defined by

$$\mathbf{V} = \begin{Bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \quad (\text{three dimensions}) \quad (4.3.51)$$

$$\mathbf{V} = \begin{Bmatrix} 1 \\ 1 \\ 0 \end{Bmatrix} \quad (\text{plane strain}) \quad (4.3.52)$$

$$\mathbf{V} = \begin{Bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{Bmatrix} \quad (\text{axisymmetry}) \quad (4.3.53)$$

Remark

In practical computing, the segregated \mathbf{d} , \mathbf{p} -form of the global matrix equation, (4.3.21), is rarely employed. The reason for this is that the band-profile structure of the coefficient matrix is lost unless the displacement and pressure degrees of freedom associated with an element are grouped together in the overall equation-number ordering. The following exercises are useful in understanding the data processing aspects of the preceding formulation.

Exercise 5. Consider the mesh shown in Fig. 2.10.1. Assume that the present mixed formulation of elasticity is being employed and that both displacement and pressure are interpolated in continuous bilinear fashion over each element. Thus there are three degrees of freedom per node, less displacement boundary conditions. Assume the pressure degree of freedom at a node directly follows the displacement degrees of freedom. Set up the ID, IEN, and LM arrays.

Sketch the band-profile structure of the global coefficient matrix. Calculate the half-bandwidth (see Fig. 1.9.2). Reorder the rows and columns of the coefficient matrix so that the pressure degrees of freedom come last. This ordering puts the coefficient matrix into the segregated \mathbf{d} , \mathbf{p} -form, eq. (4.3.21). Calculate the new half-bandwidth and compare the result with the previous ordering.

Exercise 6. Repeat Exercise 5, but assume that pressure is piecewise constant on each element. Again assume three degrees of freedom per node and associate the single element pressure degree of freedom with the last node of the element in the local

ordering. The third degree of freedom at each of the first three element nodes is a dummy degree of freedom and may be eliminated as if it were “prescribed” (i.e., set a zero in the appropriate position of the ID array).

Remark

It is important to realize that the matrix equation of the present formulation is somewhat different than the form considered heretofore. The present coefficient matrix is symmetric but not positive-definite. It possesses both positive and negative eigenvalues. In fact, in the incompressible case, improper interpolatory combinations may also lead to spurious zero eigenvalues, in which case the coefficient matrix is rendered singular and solution is impossible. These are frequently referred to as *pressure modes* in the literature; e.g., see Sani et al. [12]. (Equal-order interpolations, such as in Exercise 5, generally create this pathology. The interpolations of Exercise 6 do too under certain circumstances! See Sec. 4.5 for elaboration.)

In well-set cases, in which there are no spurious pressure modes, typical symmetric band-profile equation solvers are also capable of solving systems of the present type. However, some precautions must be taken. For example, an equation with a zero diagonal element must not appear first in the global ordering. Because the global ordering is arbitrary, this can always be accomplished. Due to the fact that some eigenvalues will be negative, equation-solving techniques that take square roots are inapplicable (e.g., the Cholesky decomposition). Alternatives that do not take square roots, such as the Crout algorithm, are acceptable however (see R.L. Taylor’s chapter on computing in [13] and Sec. 11.2.2).

4.3.6 Illustration of a Fundamental Difficulty

We have already given some forewarning that arbitrary combinations of displacement and pressure interpolations may prove ineffective in incompressible cases. An example of one of the difficulties is illustrated by the mesh in Fig. 4.3.2. Suppose linear displacement–constant pressure triangular elements are being employed. Furthermore, assume the left-hand side and bottom edges of the mesh are fixed (i.e., the displacements are identically zero).

It may be concluded from the Galerkin equation, (4.3.18), that the condition of incompressibility is

$$(q^h, \operatorname{div} \mathbf{u}^h) = 0 \quad (4.3.54)$$

Because

$$(q^h, \operatorname{div} \mathbf{u}^h) = \sum_{e=1}^{n_e} \int_{\Omega^e} q^h \operatorname{div} \mathbf{u}^h d\Omega \quad (4.3.55)$$

and q^h is an arbitrary constant on each triangle, we infer from (4.3.54) and (4.3.55) that incompressibility is satisfied in the mean, that is

$$\int_{\Omega^e} \operatorname{div} \mathbf{u}^h d\Omega = 0, \quad 1 \leq e \leq n_e \quad (4.3.56)$$

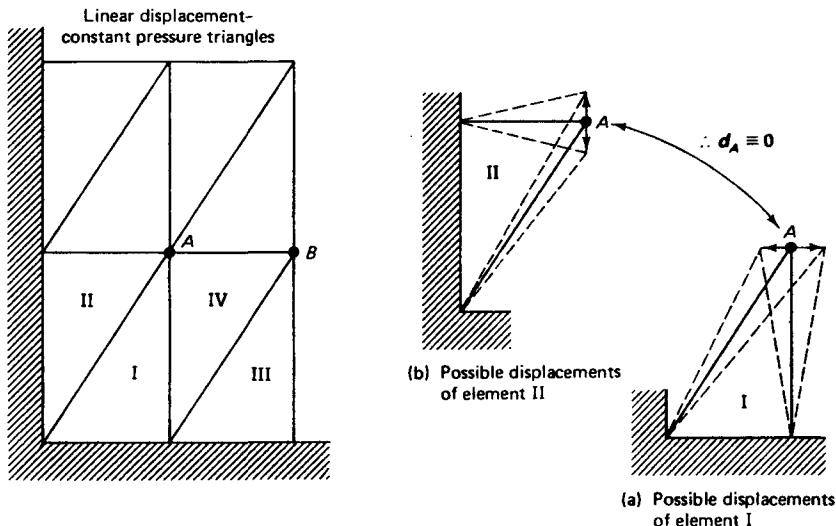


Figure 4.3.2 Mesh for which incompressibility dictates zero displacements.

Thus the area of each triangle must necessarily remain constant. (Due to the fact that \mathbf{u}^h is a linear polynomial over each triangle and thus $\operatorname{div} \mathbf{u}^h$ is constant, (4.3.56) actually implies the stronger pointwise condition

$$\operatorname{div} \mathbf{u}^h = 0 \quad (4.3.57)$$

on each Ω^e . However, this result is not needed to illustrate the present difficulty.)

Let us examine what conditions (4.3.56) enforces on the kinematics of the mesh in Fig. 4.3.2. Consider element I. We see from Fig. 4.3.2(a) that constant volume prevents the displacement at node A , d_A , from having a nonzero vertical component. Now consider element II. For this element the constant volume condition precludes horizontal motion of node A (see Fig. 4.3.2(b)). Taken together, d_A must be identically zero. We can now repeat the argument for elements III and IV to conclude d_B must also be zero. In fact, identical reasoning may be used to conclude that every node in the entire mesh must have zero displacement. Thus the only possible incompressible displacement is $\mathbf{u}^h \equiv \mathbf{0}$. This result holds no matter how many elements are present in each direction. Clearly, this type of mesh offers no approximation power whatsoever. This phenomenon is often referred to as *mesh locking*. It is but one of the difficulties afflicting problems of incompressibility.

In the nearly incompressible case, the same phenomenon occurs, only this time $\mathbf{u}^h \approx \mathbf{0}$. Thus introducing slight compressibility does not make the problem go away. To varying degrees, a tendency to lock afflicts many standard elements.

A mathematical convergence theory for mixed finite element methods of the type under consideration has been established. The key technical ingredient is the celebrated *Babuška-Brezzi, or LBB, stability condition*. To establish whether or not this condition is satisfied for elements of interest is *not* a trivial task. The interested reader

is urged to consult Oden and Carey [30] for a detailed presentation of the mathematics. For elements that satisfy the Babuška-Brezzi condition, error estimates of the following form may be established:

$$\|u^h - u\|_1 + \|p^h - p\|_0 = O(h^{\min(k, l+1)})$$

where k and l are the orders of the displacement and pressure interpolations, respectively. If $k = \min\{k, l+1\}$, the rate of convergence is said to be “optimal.” Clearly, elements that satisfy the Babuška-Brezzi condition will not lock.

The mathematics of mixed methods and, in particular, the Babuška-Brezzi condition are beyond the scope of this book. Thus it is desirable to have a simple procedure for assessing whether or not an element will lock. For this purpose the method of constraint counting proves quite effective [14–16].

4.3.7 Constraint Counts

This method is a heuristic approach for determining the ability of an element to perform well in incompressible and nearly incompressible applications. It should be emphasized that this is not a precise mathematical method for assessing elements but rather a quick and simple tool for obtaining an indication of element potential. However, it does seem to be able to predict a propensity for locking. There are, of course, other issues that need to be considered in an overall evaluation of element performance.

Let us introduce a standard mesh, which is illustrated in Fig. 4.3.3 for two-dimensional problems. Let n_{eq} represent the total number of displacement equations after boundary conditions have been imposed (i.e., the length of the vector d in (4.3.21)) and let n_c represent the total number of incompressibility constraints. As long as the pressure equations are linearly independent, n_c will equal \tilde{n}_{eq} , the number of pressure equations (i.e., length of the vector p in eq. (4.3.21)). We shall define the *constraint ratio*, r , by

$$r = \frac{n_{eq}}{n_c} \quad (4.3.58)$$

We are interested in values of r as the number of elements per side, n_{es} , approaches infinity. The conjecture is that r should mimic the behavior of the number of equilibrium equations divided by the number of incompressibility conditions for the governing system of partial differential equations. These are n_{sd} , the number of space dimensions, and 1, respectively. So in two dimensions, the ideal value of r would be 2. A value of r less than 2 would indicate a tendency to lock. If $r \leq 1$, there are more constraints on d than there are displacement degrees of freedom available, and thus severe locking would be anticipated, such as was seen for the linear displacements-constant pressure triangle. A value of r much greater than 2 indicates that not enough incompressibility conditions are present, so the incompressibility condition may be poorly approximated in some problems. A summary of these ideas follows:

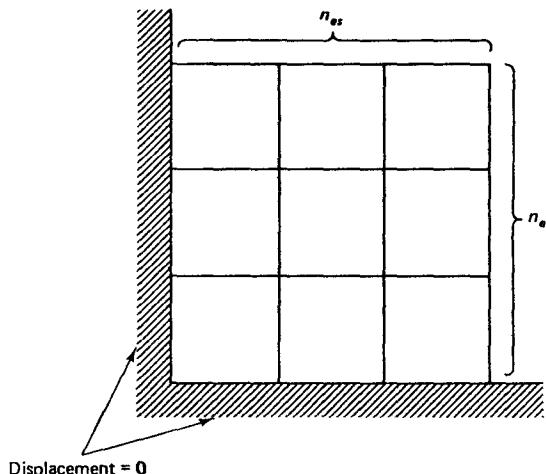


Figure 4.3.3 Standard mesh.

- $r > 2$ too few incompressibility constraints
- $r = 2$ optimal
- $r < 2$ too many incompressibility constraints
- $r \leq 1$ locking

We shall begin by calculating r for some two-dimensional elements in which **pressure is discontinuous**. In this case r is constant as a function of n_{es} , so r may be determined by considering a single element ($n_{es} = 1$). (The reader may wish to verify this statement for some of the cases considered.)

4.3.8 Discontinuous Pressure Elements

Example 1

Consider the case of the linear displacements–constant pressure triangle. See Fig. 4.3.4(a). In this case

$$r = \frac{n_{eq}}{n_c} = \frac{n_{eq}}{\tilde{n}_{eq}} = \frac{2}{2} = 1$$

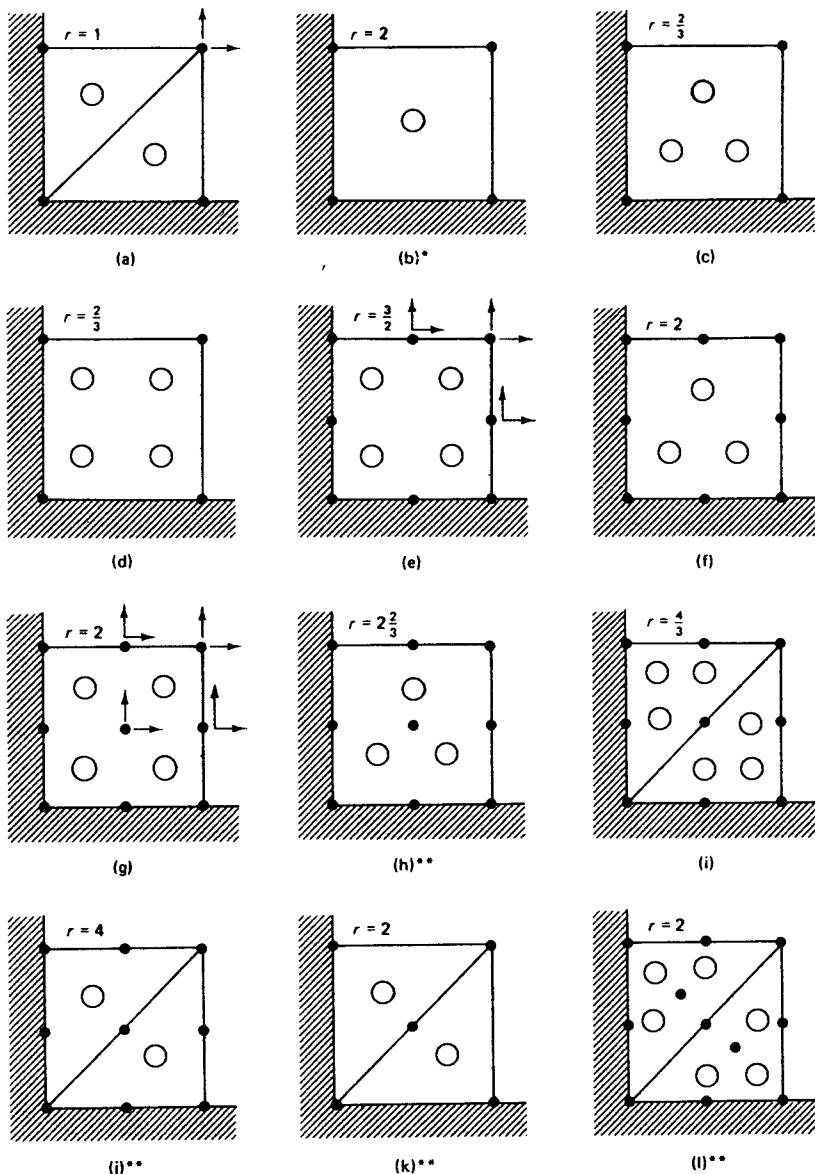
which is consistent with the behavior previously deduced. This element was first studied by Hughes and Allik [17].

Example 2

Consider the bilinear displacements–constant pressure element (see Fig. 4.3.4(b)). For this element, incompressibility is achieved in the mean, i.e.,

$$\int_{\Omega^e} \operatorname{div} u^h d\Omega = 0 \quad (4.3.59)$$

This may be shown by way of the same reasoning which led to (4.3.56). In this case we have



● Displacement node ○ Pressure node

* Despite the fact that this element violates the Babuška-Brezzi condition, optimal rate of convergence (i.e., 1) can be proven under suitable assumptions, namely, the mesh must be composed of straight-edged quadrilateral macroelements consisting of four bilinear displacements-constant pressure elements; the pressure degrees of freedom need to be eliminated by way of regularization (i.e., the "nearly incompressible" approximation); and the pressure needs to be smoothed in an appropriate way (see [33] for further details).

** These elements satisfy the Babuška-Brezzi condition. The rate of convergence is 1 for the elements in (j) and (k) and 2 for the elements in (h) and (l). The remaining elements do not satisfy the Babuška-Brezzi condition.

Figure 4.3.4 Discontinuous pressure-field elements.

$$r = \frac{n_{eq}}{n_c} = \frac{n_{eq}}{\tilde{n}_{eq}} = \frac{2}{1} = 2$$

Optimal behavior is indicated and this element is widely used. This element was first proposed by Hughes and Allik [17].

From Exercise 7 of Sec. 3.11 we recall that the mean value of $\operatorname{div} \mathbf{u}^h$ occurs at the origin of the isoparametric coordinate system (i.e., $\xi = \eta = 0$). This is the only point in the element at which incompressibility is identically satisfied. The mean-value point shifts somewhat for the case of the axisymmetric version of this element.

Example 3

Consider the bilinear displacements-linear pressure element shown in Fig. 4.3.4(c). In this case,

$$r = \frac{n_{eq}}{n_c} = \frac{n_{eq}}{\tilde{n}_{eq}} = \frac{2}{3}$$

which indicates severe locking.

Although no improvement could be expected by increasing the pressure interpolation to bilinear, we wish to consider this element (see Fig. 4.3.4(d)) since it illustrates a point. In this case⁴

$$\int_{\Omega^h} \underbrace{q^h}_{\text{bilinear}} \underbrace{\operatorname{div} \mathbf{u}^h}_{\text{linear}} d\Omega = 0 \quad (4.3.60)$$

Thus there are more pressure weighting functions, and—consequently—incompressibility conditions, than there are independent monomials in $\operatorname{div} \mathbf{u}^h$. As a result the four incompressibility conditions are linearly dependent and so

$$r = \frac{n_{eq}}{n_c} = \frac{n_{eq}}{\tilde{n}_{eq} - 1} = \frac{2}{4 - 1} = \frac{2}{3}$$

as for the preceding case. Increasing the order of pressure interpolation further does not change r but increases the order of singularity of the matrix system. Clearly an approximation of this kind is useless, since the global matrix could not be inverted in the incompressible case.

Singularities of this type affect many elements in incompressible applications. It is not always obvious that this can occur for an element. For example, bilinear displacements-constant pressure elements exhibit a singularity in the global pressure equations for the mesh shown in Fig. 4.3.5(a) as long as n_{es} is even. This pathology is referred to as the *checkerboard mode* since the pressure degrees of freedom of the eigenvector of the global equations corresponding to the zero eigenvalue take the form +1 on the “red” squares and -1 on the “black” squares (Fig. 4.3.5(b)). When we come to the penalty method, we will show that this mode can be removed resulting in an

⁴For the standard mesh we can write $\xi = \xi(x)$ and $\eta = \eta(y)$, where each of these functions is linear. Thus a bilinear expansion in ξ and η can also be written as a bilinear expansion in x and y . For example, $u_i^h = \alpha_0 + \alpha_1 \xi + \alpha_2 \eta + \alpha_3 \xi \eta = \beta_0 + \beta_1 x - \beta_2 y + \beta_3 xy$ where the α 's and β 's are parameters which depend on the nodal values of u_i^h . The β 's also depend on the lengths of the element edges. Clearly, $\operatorname{div} \mathbf{u}^h = (\beta_{11} + \beta_{22}) + \beta_{32}x + \beta_{31}y$, which is identically zero pointwise if and only if $0 = \beta_{11} + \beta_{22} = \beta_{32} = \beta_{31}$, i.e., there are three independent constraints on \mathbf{u}^h .

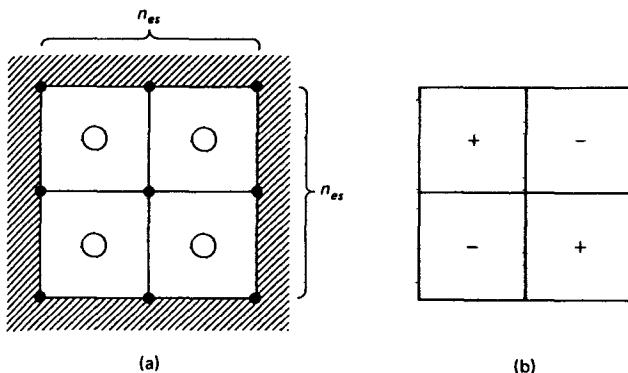


Figure 4.3.5 Checkerboard pressure mode.

effective formulation for this element. See [31–33] for mathematical results employing this idea.

In a sense all elements are subject to the problem of singularity in the incompressible limit. This can easily be seen from (4.3.21). If enough displacement degrees of freedom have been specified such that $n_{eq} < \tilde{n}_{eq}$, then the matrix *must* be singular at least to degree $\tilde{n}_{eq} - n_{eq}$.

Example 4

Consider the eight-node serendipity displacement–bilinear pressure quadrilateral shown in Fig. 4.3.4(e). A calculation of $\operatorname{div} \mathbf{u}^h$ reveals that it can be expressed as a full quadratic polynomial in x and y and thus involves six independent coefficients, which must vanish for incompressibility to be satisfied pointwise. For a bilinear pressure expansion, the incompressibility condition takes the form

$$\int_{\Omega^e} \underbrace{q^h}_{\text{bilinear}} \underbrace{\operatorname{div} u^h}_{\text{quadratic}} d\Omega = 0 \quad (4.3.61)$$

and thus four conditions emanate from (4.3.61). The constraint ratio is, therefore,

$$r = \frac{n_{eq}}{n_c} = \frac{n_{eq}}{\tilde{n}_{eq}} = \frac{6}{4} = \frac{3}{2}$$

which indicates that there are too many incompressibility constraints.

Some ostensible improvement can be made by reducing the pressure interpolation to linear (see Fig. 4.3.4(f)). In this case

$$r = \frac{n_{eq}}{\tilde{n}_{eq}} = \frac{6}{3} = 2$$

the optimal value. However, neither of these elements (i.e., Fig. 4.3.4(e) and (f)) is currently favored.

Example 5

Consider the biquadratic displacements–bilinear pressure quadrilateral element shown in Fig. 4.3.4(g) for which

$$r = \frac{n_{eq}}{n_c} = \frac{n_{eq}}{\tilde{n}_{eq}} = \frac{8}{4} = 2$$

Thus from a constraint ratio point of view this element appears ideal. However, it also gives rise to a pressure mode such as that for the bilinear displacements–constant pressure element.

Reducing the pressure interpolation to linear (Fig. 4.3.4(h)) removes this pressure mode and the constraint ratio increases to $r = \frac{8}{3} = 2\frac{2}{3}$. This element is currently felt to be one of the most effective quadrilateral elements for incompressible analysis.

Example 6

Consider the quadratic displacements–linear pressure triangle shown in Fig. 4.3.4(i). The constraint ratio is

$$r = \frac{n_{eq}}{n_c} = \frac{n_{eq}}{\tilde{n}_{eq}} = \frac{8}{6} = \frac{4}{3}$$

Thus this element possesses too many incompressibility constraints. (Observe that pointwise satisfaction of the incompressibility constraint is attained.)

To reduce the number of incompressibility constraints, constant pressure may be employed (see Fig. 4.3.4(j)), resulting in incompressibility in the mean and a relatively high constraint ratio of $r = \frac{8}{2} = 4$. This element is, nevertheless, favored by some analysts [18] because it does not possess pressure modes of the kind described previously. The drawback, however, is the crude approximation of incompressibility (i.e., piecewise constant) relative to displacements (i.e., piecewise quadratic), which results in suboptimal convergence. (For optimal rate of convergence, the complete polynomial in the pressure field should be one order lower than the complete polynomial in the displacements.) The convergence of this element has been established in [19].

A more balanced approximation employing constant pressure is shown in Fig. 4.3.4(k) [20]. Each quadrilateral macroelement is composed of linear displacement–constant pressure triangles in which quadratic displacement modes are added along the diagonal edge. The constraint ratio is $r = \frac{4}{2} = 2$, which is optimal. Additionally, this element exhibits no spurious pressure modes.

Another way to “fix” the quadratic triangle of Fig. 4.3.4(i) is to add an internal displacement node, as in Fig. 4.3.4(l). (The displacement interpolations may be constructed using techniques described in Chapter 3.) This element possesses an optimal constraint ratio:

$$r = \frac{n_{eq}}{n_c} = \frac{n_{eq}}{\tilde{n}_{eq}} = \frac{12}{6} = 2$$

Convergence and error estimates have been established by Crouzeix and Raviart [19].

The remaining examples involve *continuous pressure fields*. The first to study continuous pressure-field elements were Hughes and Allik [17]. In these cases r varies with n_{es} . However, it may be argued that

$$\lim_{n_{es} \rightarrow \infty} r$$

may be obtained by again considering the single corner element of the standard mesh

and ignoring all degrees of freedom—pressure in addition to displacement—on the left and bottom boundaries. (The reader may wish to verify this statement as an exercise for some of the following elements.)

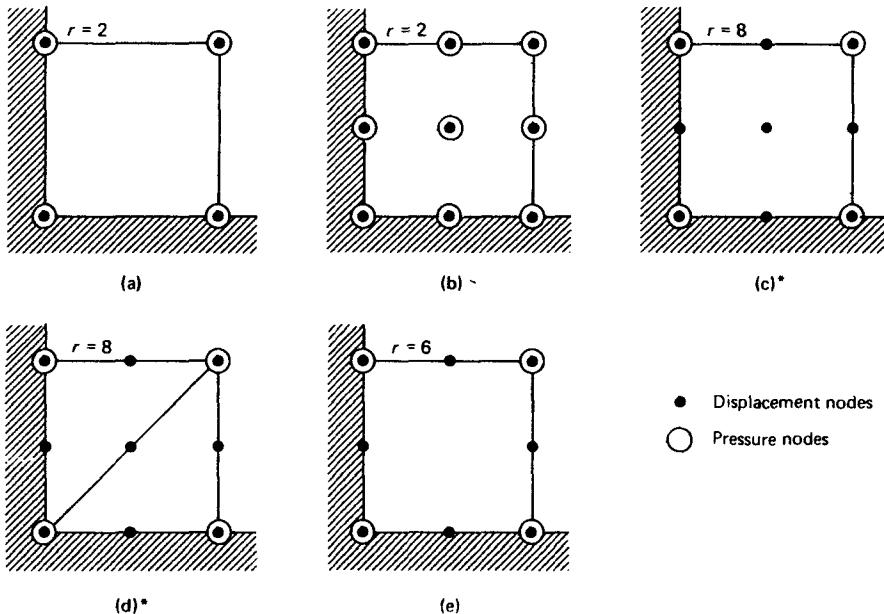
4.3.9 Continuous Pressure Elements

Example 7

Consider the bilinear displacements—(continuous!) bilinear pressure quadrilateral shown in Fig. 4.3.6(a). The constraint ratio is

$$\lim_{n_{es} \rightarrow \infty} r = 2$$

which is optimal. However, the convergence is from below and this element may exhibit spurious pressure modes. This appears to be a general fact for typical elements possessing identical displacement and pressure interpolations (e.g., see Fig. 4.3.6(b)).



* These elements satisfy the Babuška-Brezzi condition. The convergence rate is 2. The elements in (a) and (b) violate the Babuška-Brezzi condition, whereas the issue is still open regarding the element in (e).

Figure 4.3.6 Continuous pressure-field elements.

Example 8

The deficiencies noted in the previous example can be corrected by lowering the pressure interpolation. Fig. 4.3.6(c) and (d) depicts elements of this type. For both these elements

$$\lim_{n_{es} \rightarrow \infty} r = 8$$

which is very high (i.e., there are too few incompressibility constraints). Although no pressure modes are exhibited by these elements and the convergence rate is theoretically optimal [21]⁵, very poor approximations of the incompressibility condition are frequently noted. Nevertheless, these elements are widely used in incompressible analysis.

The constraint ratio can be reduced somewhat by removing internal displacement degrees of freedom as, for example, in Fig. 4.3.6(e). For this element,

$$\lim_{n_{es} \rightarrow \infty} r = 6$$

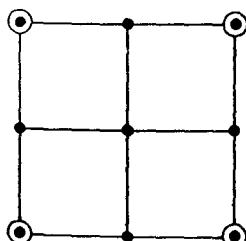
which is still rather high. This element has also been widely used in incompressible analysis although, at the time of writing, the question of convergence is still open.

Remarks

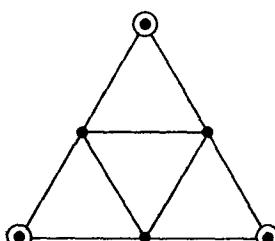
1. It is an empirical observation that all elements which are effective in incompressible analysis possess constraint ratios greater than or equal to n_{sd} . We wish to reiterate, however, that some of these elements may give rise to so-called pressure modes (i.e., singularities in the global equations). Thus the analysis of incompressible media is a delicate matter and care must be exercised. We consider the matter further in subsequent sections.

2. A number of other interesting elements for incompressible media are described in Ruas [20], Thomasset [22], Fortin [23], Griffiths [24–27], Oden and Jacquotte [28], and Boland and Nicolaides [29].

3. The continuous-pressure macroelements illustrated in Fig. 4.3.7 have also been shown to converge. Note that the nodal patterns are identical to Fig. 4.3.6(c) and (d) and thus so are the constraint ratios (i.e., $\lim_{n_{es} \rightarrow \infty} r = 8$). These elements represent a trade-off compared with those of Fig. 4.3.6(c) and (d): The rate of convergence is one order higher for the elements of Fig. 4.3.6(c) and (d), but the elements of Fig. 4.3.7 are amenable to more efficient implementation. See Thomasset [22] and references therein for further details.



Four bilinear displacement quadrilaterals covered with a bilinear pressure field



Four linear displacement triangles covered with a linear pressure field

Figure 4.3.7 Some convergent, continuous pressure-field macroelements.

⁵Certain cautions must be respected in specifying displacement boundary conditions. For the quadrilateral in Fig. 4.3.6(c), displacement should be specified on at most two edges, whereas for the triangle of Fig. 4.3.6(d), displacement should be specified on at most one edge.

4. In some recent works, convergence of equal-order interpolations is established by adopting modified formulations of the incompressible problem. Brezzi and Pitkäranta [34] propose a formulation for linear triangles and Hughes et al. [35] develop a formulation for general classes of elements.

4.4 PENALTY FORMULATION: REDUCED AND SELECTIVE INTEGRATION TECHNIQUES; EQUIVALENCE WITH MIXED METHODS

Let us recall the formulations we introduced previously for modeling a single constraint (Sec. 4.2). These were the *Lagrange multiplier method* and the *penalty function formulation*. In the context of the incompressibility constraint, the mixed formulation of Sec. 4.3 is a Lagrange-multiplier-type of formulation in which the pressure field plays the role of the Lagrange multiplier.

We also recall from Sec. 4.2 that in the penalty formulation, we simply approximated the single constraint by introducing a stiff elastic spring. In the context of incompressibility, this amounts to allowing for slight compressibility. That is, λ is taken finite, but large with respect to μ .

Nearly incompressible case

$$\frac{\lambda}{\mu} \gg 1 \quad (4.4.1)$$

This case can be easily done within the framework of Sec. 4.3 and has some advantages. For example, in the case of discontinuous pressure fields, the pressure degrees of freedom can be eliminated on the element level. The idea is to select λ sufficiently large so that compressibility errors are negligible, but not so large that numerical problems arise. The ratio λ/μ thus depends on the floating-point word length of the computer being utilized. In our experience with words of length 60–64 bits, we have found that the range

$$10^7 \leq \frac{\lambda}{\mu} \leq 10^9 \quad (4.4.2)$$

is effective. As noted in Sec. 4.3, allowing for slight compressibility does not change the situation with regard to elements. One must employ only those elements that are also effective in the incompressible limit, as slight compressibility does not eliminate the fundamental difficulties.

Once we are willing to introduce some compressibility, the theory of Chapter 2 is also applicable. The question naturally arises then as to the performance of the standard “displacement-only” elements of Chapter 3 in these circumstances.⁶ The

⁶The finite element methodology of Chapters 2 and 3 is generally referred to as the *displacement formulation*.

answer, as might be anticipated, is that these elements tend to perform poorly in nearly incompressible applications, frequently exhibiting a tendency to lock. However, a slight modification of the usual formulation enables the construction of elements that are identical to many of the discontinuous pressure field elements generated by the mixed formulation. The basic tools in this process are *reduced and selective integration* procedures, which are described as follows.

Reduced and Selective Integration

The expression for element stiffness in the displacement formulation is given by (we omit the element number superscript, e , for simplicity):

$$\mathbf{k} = [k_{pq}], \quad 1 \leq p, q \leq n_{ee} \quad (4.4.3)$$

$$k_{pq} = \mathbf{e}_i^T \mathbf{k}_{ab} \mathbf{e}_j \quad (4.4.4)$$

$$p = n_{ed}(a - 1) + i, \quad q = n_{ed}(b - 1) + j \quad (4.4.5)$$

$$\mathbf{k}_{ab} = \int_{\Omega} \mathbf{B}_a^T D \mathbf{B}_b \, d\Omega \quad (4.4.6)$$

The material properties matrix D may be written as

$$\mathbf{D} = \bar{\mathbf{D}} + \bar{\bar{\mathbf{D}}} \quad (4.4.7)$$

where $\bar{\mathbf{D}}$ is the μ -part of \mathbf{D} , defined in Sec. 4.3, and $\bar{\bar{\mathbf{D}}}$, the remainder, is the λ -part. The reader may easily verify the following formulas.

Plane strain

$$\bar{\bar{\mathbf{D}}} = \lambda \begin{bmatrix} 1 & 1 & 0 \\ & 1 & 0 \\ \text{symm.} & & 0 \end{bmatrix} \quad (4.4.8)$$

Axisymmetry

$$\bar{\bar{\mathbf{D}}} = \lambda \begin{bmatrix} 1 & 1 & 0 & 1 \\ & 1 & 0 & 1 \\ \text{symm.} & 0 & 0 & \\ & & & 1 \end{bmatrix} \quad (4.4.9)$$

Three dimensions

$$\bar{\bar{\mathbf{D}}} = \lambda \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ & 1 & 1 & 0 & 0 & 0 \\ & & 1 & 0 & 0 & 0 \\ & & & 0 & 0 & 0 \\ \text{symm.} & & & 0 & 0 & \\ & & & & & 0 \end{bmatrix} \quad (4.4.10)$$

Employing (4.4.7) in (4.4.6) enables us to write

$$k_{ab} = \bar{k}_{ab} + \bar{\bar{k}}_{ab} \quad (4.4.11)$$

where

$$\bar{k}_{ab} = \int_{\Omega'} \mathbf{B}_a^T \bar{\mathbf{D}} \mathbf{B}_b \, d\Omega \quad (4.4.12)$$

$$\bar{\bar{k}}_{ab} = \int_{\Omega'} \mathbf{B}_a^T \bar{\bar{\mathbf{D}}} \mathbf{B}_b \, d\Omega \quad (4.4.13)$$

Note that \bar{k}_{ab} is the part of the stiffness that also appears in the mixed formulation (Sec. 4.3). Due to the fact that $\lambda/\mu \gg 1$ and $\bar{\bar{k}}$ is proportional to λ , whereas \bar{k} is proportional to μ , the numerical values of terms in \bar{k} tend to be very large compared with those of $\bar{\bar{k}}$. The $\bar{\bar{k}}$ -term is the part of the stiffness that attempts to maintain the volumetrically stiff behavior. Because typical finite elements tend to lock (i.e., there are proportionally too many incompressibility-type conditions), special treatment of $\bar{\bar{k}}$ is required to alleviate this tendency. One simple and practically important way of going about this is to reduce the order of numerical quadrature employed to evaluate $\bar{\bar{k}}$ below that “normally” used. The basic idea is illustrated in the following example.

Example 1

Consider the four-node bilinear displacement element in plane strain. “Normal” quadrature for this element is considered to be the 2×2 Gauss rule. The stiffness matrix turns out to be *identical* to that obtained in the mixed formulation for bilinear displacements and (discontinuous) linear pressures in which the pressure degrees of freedom are eliminated on the element level, as indicated in (4.3.30). This fact is known from an equivalence theorem due to Malkus and Hughes [36]. The constraint ratio of this element was calculated to be $\frac{2}{3}$ in Sec. 4.3, and thus locking-type behavior would be anticipated.

Recall also from Sec. 4.3 that by employing constant pressure over each element, the bilinear element attains an optimal constraint ratio of 2. The equivalent displacement model may be obtained by reducing the quadrature of the $\bar{\bar{k}}$ -term to one-point Gauss [36, 37]. Thus, as $\lambda/\mu \rightarrow \infty$, incompressibility in the mean is attained. The performance of this element is illustrated by the following numerical example.

Consider the equations of two-dimensional linear isotropic elasticity theory on the domain illustrated in Fig. 4.4.1. The boundary conditions are given as follows:

Displacement

$$u_1(0, 0) = u_2(0, 0) = 0$$

$$u_1(0, \pm c) = 0$$

Traction

$$h_1(x_1, \pm c) = h_2(x_1, \pm c) = 0, \quad x_1 \in]0, L[$$

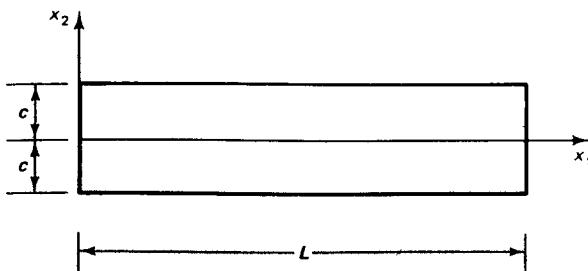


Figure 4.4.1 Domain for plane strain elasticity problem

$$\left. \begin{array}{l} h_1(L, x_2) = 0 \\ h_2(L, x_2) = \frac{P}{2I}(c^2 - x_2^2) \\ h_1(0, x_2) = \frac{PLx_2}{I} \\ h_2(0, x_2) = -\frac{P}{2I}(c^2 - x_2^2) \end{array} \right\} \begin{array}{l} x_2 \in]-c, c[\\ x_2 \in]-c, 0[\cup]0, c[\end{array}$$

where P is a given constant and $I = 2c^3/3$.

The traction boundary conditions are those encountered in simple bending theory for a cantilever beam with root section at $x = 0$ —i.e., parabolically varying end shear and linearly varying bending stress at the root. The displacement boundary conditions allow the root section to warp. The following data were employed in the numerical calculations:

$$L = 16, \quad c = 2$$

The mesh used is depicted in Fig. 4.4.2. (Only half the domain need be modeled since the x_1 -axis is a line of antisymmetry.)

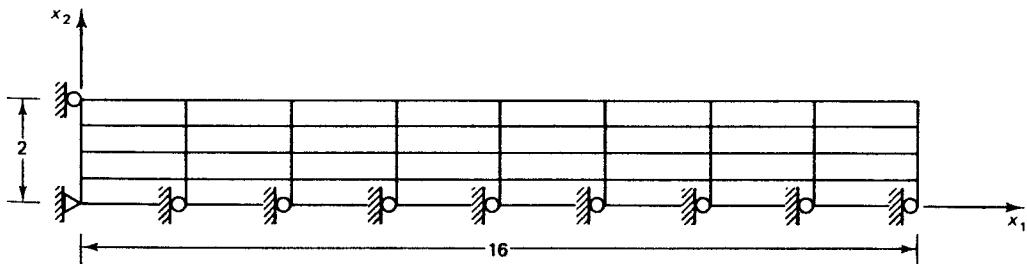


Figure 4.4.2 Element mesh and boundary conditions for plane strain elasticity problem.

Plane strain conditions were assumed and four-node quadrilateral elements were employed.

Vertical tip displacements [i.e., $u_2(16, 0)$] are compared in Table 4.4.1. Both the “standard” 2×2 Gauss quadrature element and the selective reduced element provide adequate results for $\nu = 0.3$. However, for the nearly incompressible case, the standard quadrilateral degenerates, whereas the selective integration element retains accuracy.

TABLE 4.4.1 Normalized Vertical
Tip Displacements of Plane Strain
Beam

| ν | U_2 | S_1 |
|-------|-------|-------|
| 0.3 | 0.904 | 0.912 |
| 0.499 | 0.334 | 0.937 |

Key:

U_2 2×2 uniform integration (= exact
in present case)

S_1 selective integration (2×2 on
 μ -term, one-point on λ -term)

Definitions. *Selective reduced integration* refers to the case in which reduced integration is used only on the λ -term, whereas normal integration is used on the μ -term. *Uniform reduced integration* refers to the case when both terms are integrated with a reduced rule.

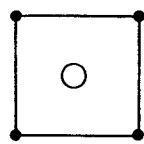
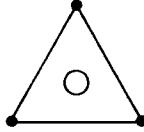
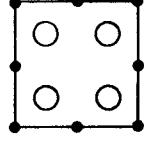
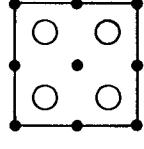
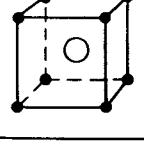
The ostensible advantage of uniform reduced integration is the economy of element formulation. The disadvantage is that the rank of the element stiffness may be reduced, resulting in singularity of the global matrix. Selective integration retains the correct rank of the element stiffness, and therefore the global stiffness also possesses correct rank. This follows from the fact that $\bar{a}(\cdot, \cdot)$ is positive definite. (The $\bar{\bar{a}}(\cdot, \cdot)$ term may be entirely ignored without affecting rank.)

Equivalence Theorem. The general theorem presented in Malkus and Hughes [36] established the equivalence of many mixed and reduced and selective integration elements. Some examples are presented in Fig. 4.4.3. It may thus be concluded that the reduced and selective integration procedures are very simple ways of attaining the performance of the mixed formulation without engendering the additional complications.

The equivalence theorem states that the element stiffness of the mixed method, namely (4.3.30), is identical to the element stiffness of the reduced or selective integration approach and consequently the displacements are also identical. At the Gauss points of the *reduced integration rule* in the reduced and selective integration approaches,

$$p^h = -\lambda \operatorname{div} u^h \quad (4.4.14)$$

agrees with the pressure field of the mixed method. Elsewhere in the element, the pressure may be determined from the displacements by interpolating the values of (4.4.14) with the pressure shape functions. The equivalence theorem thus provides the interpretation rule by which the pressure should be defined within the reduced and selective integration approaches.

| | Mixed element | Equivalent selective integration element | | |
|---|----------------------------|--|--|--|
| | Displacement interpolation | Pressure interpolation | Normal Gaussian quadrature* (\bar{k}) | Reduced Gaussian quadrature ($\bar{\bar{k}}$) |
|  | Bilinear | Constant | 2 X 2 | One-point |
|  | Linear | Constant | One-point | One-point |
|  | Serendipity | Bilinear | 3 X 3 | 2 X 2 |
|  | Biquadratic | Bilinear | 3 X 3 | 2 X 2 |
|  | Trilinear | Constant | 2 X 2 X 2 | One-point |

* Also used on all terms of mixed formulation.

Figure 4.4.3 Some equivalent elements.

For example, in the case of the selectively integrated four-node bilinear element, the constant element pressure of the mixed method agrees with the value computed from (4.4.14) at the origin of isoparametric coordinates (i.e., $\xi = \eta = 0$). As a second example, consider the selectively integrated nine-node element. In this case,

the values of (4.4.14) computed at the 2×2 Gauss points need to be interpolated via bilinear shape functions. The resulting formula is

$$p^h(\xi, \eta) = \sum_{\tilde{\alpha}=1}^4 \tilde{N}_{\tilde{\alpha}}(\xi, \eta) p_{\tilde{\alpha}} \quad (4.4.15)$$

where

$$p_{\tilde{\alpha}} = -\lambda (\operatorname{div} \mathbf{u}^h)(\tilde{\xi}_{\tilde{\alpha}}, \tilde{\eta}_{\tilde{\alpha}}) \quad (4.4.16)$$

and $\tilde{\xi}_{\tilde{\alpha}}$, $\tilde{\eta}_{\tilde{\alpha}}$ are the coordinates of the $\tilde{\alpha}$ th Gauss point. The shape functions in (4.4.15) are defined by

$$\tilde{N}_{\tilde{\alpha}}(\xi, \eta) = \frac{1}{4}(1 + 3\tilde{\xi}_{\tilde{\alpha}}\xi)(1 + 3\tilde{\eta}_{\tilde{\alpha}}\eta) \quad (4.4.17)$$

Exercise 1. Verify that (4.4.17) satisfies the interpolation property at the reduced Gauss points, i.e.,

$$\tilde{N}_{\tilde{\alpha}}(\tilde{\xi}_{\tilde{\beta}}, \tilde{\eta}_{\tilde{\beta}}) = \delta_{\tilde{\alpha}\tilde{\beta}} \quad (4.4.18)$$

Exercise 2. Describe how to program the selective integration procedure.

Exercise 3. Show that for the isotropic case, the symmetric bilinear form $a(\cdot, \cdot)$ can be written as

$$a(\mathbf{w}, \mathbf{u}) = \bar{a}(\mathbf{w}, \mathbf{u}) + \bar{\bar{a}}(\mathbf{w}, \mathbf{u}) \quad (4.4.19)$$

where

$$\bar{\bar{a}}(\mathbf{w}, \mathbf{u}) = (\operatorname{div} \mathbf{w}, \lambda \operatorname{div} \mathbf{u}) \quad (4.4.20)$$

This result should reinforce the assertion that it is the $\bar{\bar{k}}$ -stiffness which is responsible for enforcing the volumetrically stiff behavior.

Remarks

1. Constraint ratios may be determined for reduced and selective integration elements as follows: Consider the expression that leads to $\bar{\bar{k}}$, namely, (4.4.20). The maximum number of constraints possible is given by the number of independent monomials in $\operatorname{div} \mathbf{u}^h$. Likewise, the maximum can be no greater than the number of quadrature points used to evaluate (4.4.20). Thus

$$n_c = \min \{ \text{number of independent monomials present in } \operatorname{div} \mathbf{u}^h; \text{ number of quadrature points used to evaluate (4.4.20)} \}$$

As an example, consider the four-node bilinear element. Normal quadrature results in $n_c = 3$ (i.e., the number of independent monomials in $\operatorname{div} \mathbf{u}^h$) and so $r = \frac{3}{4}$. On the other hand, if reduced one-point quadrature is used on (4.4.20), $n_c = 1$ (i.e., number of quadrature points) and so $r = 2$. These values of the constraint ratio agree with those computed for the equivalent mixed elements.

2. Fried [38] argued in a somewhat different way in favor of reducing the integration rule of the λ -term. Consider the case in which λ and μ are constants and let

$$\bar{\mathbf{K}} = \mu \mathbf{K}_1 \quad (4.4.21)$$

$$\bar{\bar{\mathbf{K}}} = \lambda \mathbf{K}_2 \quad (4.4.22)$$

Then

$$\begin{aligned} \mathbf{F} &= \mathbf{Kd} \\ &= (\bar{\mathbf{K}} + \bar{\bar{\mathbf{K}}})\mathbf{d} \\ &= (\mu \mathbf{K}_1 + \lambda \mathbf{K}_2)\mathbf{d} \end{aligned} \quad (4.4.23)$$

Fried noted that for typical elements \mathbf{K}_2 tended to have too great a rank (i.e., too many incompressibility constraints). In fact, in some situations \mathbf{K}_2 is nonsingular and therefore for $\lambda/\mu \gg 1$,

$$\begin{aligned} \mathbf{d} &= (\mu \mathbf{K}_1 + \lambda \mathbf{K}_2)^{-1}\mathbf{F} \\ &\cong \frac{1}{\lambda} \mathbf{K}_2^{-1} \mathbf{F} \end{aligned} \quad (4.4.24)$$

From (4.4.24), it can be seen that as $\lambda \rightarrow \infty$, $\mathbf{d} \rightarrow 0$. This situation may be seen to be equivalent to the locking phenomenon noted in the example of Fig. 4.3.2. Thus Fried argued that for a formulation to be successful in nearly incompressible applications, \mathbf{K}_2 **must be singular** so that (4.4.24) does not hold. Fried further argued that one way of achieving this end was to reduce the order of quadrature on the element contributions to $\bar{\bar{\mathbf{K}}}$.

3. The linear triangular displacement element, which is equivalent to the linear displacement-constant pressure triangular mixed element (Hughes and Allik [39]), exhibits pathological locking on the standard mesh (Fig. 4.3.3). However, Nagtegaal et al. [40] found that in the cross-diagonal pattern (see Fig. 4.4.4(a)) this element improves. Ostensibly, the constraint ratio for this pattern is still 1. However, what occurs is that the incompressibility conditions exhibit a linear dependency and thus there are only three incompressibility constraints per quadrilateral macroelement. Thus $r = \frac{4}{3}$.

Mercier [41] provided an elegant argument, which established similar improvement for quadratic triangles in the cross-diagonal pattern (see Fig. 4.4.4(b)).

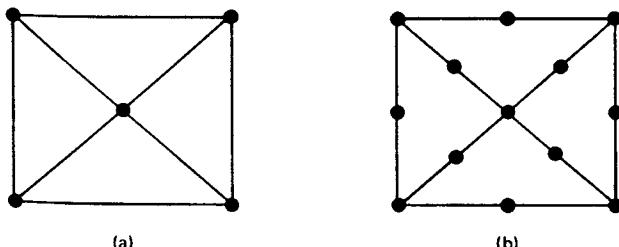


Figure 4.4.4 Quadrilateral macro-elements formed from triangles in the cross-diagonal pattern.

At this point, however, there appear to be several disadvantages in adopting the macroelement approach. First of all, since there is a necessity of assembling into quadrilateral macroelements, no additional flexibility is gained by the use of the triangular elements over standard quadrilateral elements. Second, when compared to the selectively integrated Lagrange quadrilaterals of equal interpolatory order, less accuracy is attained by the triangular macroelements despite the fact that approximately two times as many unknowns are involved.

On the other hand, a potential advantage of the triangular macroelements is that in the plane strain case, if the edges are straight, pointwise incompressibility may be obtained. Incompressibility, in general, occurs only at the quadrature points of the reduced integration rule for the selectively integrated Lagrange elements.

Exercise 4. Consider the Lagrange family of quadrilaterals and bricks in which normal integration is used for the entire stiffness. Determine expressions for the constraint ratio as functions of the number of nodes along an element side, say n . Deduce that the constraint ratio is poorer for lower-order members of the family than for higher-order members. Show that as $n \rightarrow \infty$, $r \rightarrow n_{sd}$ (optimal). (This result is consistent with the observation that fully integrated higher-order elements are more successful in nearly incompressible applications than fully integrated lower-order elements.) Contrast the results with those obtained using selective reduced integration.

Partial solution We shall do the two-dimensional case and leave the three-dimensional case for the reader.

$$\begin{aligned}n_{eq} &= 2(n - 1)^2 \\n_c &= n^2 - 1 \quad (\text{normal integration; exact}) \\r &= \frac{2(n - 1)^2}{n^2 - 1} = 2 \frac{(n - 1)}{(n + 1)}\end{aligned}$$

For normal integration, r improves as the order of the element is increased. For example,

| n | r |
|----------|------------------|
| 2 | $\frac{2}{3}$ |
| 3 | 1 |
| 4 | $\frac{6}{5}$ |
| . | . |
| . | . |
| ∞ | 2 ($= n_{sd}$) |

Thus the necessity of reduced integration decreases as the order of the element increases. In the case of reduced integration of the λ -term,

$$\begin{aligned}n_c &= (n - 1)^2 \\r &= 2 (= n_{sd}) \quad \text{independent of } n\end{aligned}$$

Some Historical Remarks on Mixed and Reduced and Selective Integration Methods

Mixed finite element formulations were first discussed by Fraeijs de Veubeke [42] and Herrmann [43]. Herrmann developed a reduced form of Reissner's variational principle particularly suited to problems of incompressible and nearly incompressible elasticity and, based upon this principle, established the first effective finite elements for such cases. This is the formulation given in Sec. 4.3. Prior to this development many displacement models were applied to these problems, and poor behavior was typically observed. The reasons for this were not understood at the time. Certain elements derived from Herrmann's formulation also failed. Hughes and Allik [39] traced this failure to a correspondence between mixed and displacement models, contained within Fraeijs de Veubeke's *limitation principle* [42].

The first example of a uniform reduced integration element was apparently the plate-shell element presented by Zienkiewicz et al. [44]. This element, among others, is discussed in Chapter 5. The same concept was employed in other areas by Zienkiewicz and colleagues. In particular, Naylor [45] and Zienkiewicz and Godbole [46] advocated the use of the eight-node serendipity element in problems involving incompressibility. The procedure, however, was viewed by many as more a "trick" than a method and some bad experiences were subsequently noted for the serendipity element.

The concept of selective integration was first employed by Doherty et al. [47] to obtain improved bending behavior in simple four-node elasticity elements. One-point Gauss quadrature was used on the shear-strain term, and 2×2 Gauss quadrature was used to integrate the remaining terms. Although improved behavior was noted in some configurations, lack of invariance opened the approach to criticism.

Studies performed by Fried [38], Nagtegaal et al. [40], and Argyris et al. [48] provided fresh insights into why the displacement approach failed in constrained problems. Malkus [49, 50] proved the equivalence of a class of mixed models with reduced selective integration single-field elements in linear elasticity theory. The equivalence results of Malkus and Hughes [36] elevated the reduced and selective integration approaches from the realm of tricks to a legitimate methodology. Considerable research on the behavior of mixed and reduced and selective integration elements has taken place in recent years. A summary of more recent developments is contained in the following sections.

4.4.1 Pressure Smoothing

The pressure field in the reduced and selective integration penalty function formulation is to be viewed as discontinuous from element to element. In fact, all displacement derivatives for C^0 isoparametric elements are, in general, discontinuous across element boundaries. Thus, for plotting purposes, it is desirable to employ a smoothing procedure, which redefines the field under consideration in terms of the displacement shape functions N_A .

With specific reference to the pressure, there is at least one other reason for employing a smoothing procedure. It was mentioned earlier that, in certain situations,

discontinuous-pressure, mixed-method finite elements exhibit a rank-deficiency in the assembled pressure equations. By the equivalence theorem, “problems” are also to be expected with the pressure field of the penalty function formulation. These problems typically manifest themselves as pressure oscillations. For example, if four-node, quadrilateral elements are employed in a square mesh, with an even number of square elements in each direction, subjected to all velocity boundary conditions, then a checkerboard pressure oscillation is produced. Despite the pressure oscillations, the velocity field remains good.

Fortunately, smoothing procedures of a *least squares* type [51] seem to perform the necessary filtering as a byproduct. A comprehensive study of such techniques has been performed by Lee et al. [52]. The methods we prefer for constant-pressure elements [53], which involve slight modifications of schemes proposed in [52], are described next.

Let the discontinuous pressure field be written as

$$p^h = \sum_{e=1}^{n_{el}} \psi^e p^e \quad (4.4.25)$$

where p^e is the element mean pressure and ψ^e is the e th element “characteristic function,” i.e.,

$$\psi^e(x) = \begin{cases} 1 & \text{if } x \in \Omega^e \\ 0 & \text{if } x \notin \Omega^e \end{cases} \quad (4.4.26)$$

The smoothed pressure is written

$$\tilde{p} = \sum_{A=1}^{n_{np}} N_A \tilde{p}_A \quad (4.4.27)$$

The standard least squares procedure gives rise to the following matrix problem:⁷

$$Y \tilde{p} = P \quad (4.4.28)$$

where

$$Y = [Y_{AB}] \quad (4.4.29)$$

$$\tilde{p} = \{\tilde{p}_B\} \quad (4.4.30)$$

and

$$P = \{P_A\} \quad (4.4.31)$$

⁷The least squares procedure defines \tilde{p} by minimizing

$$\int_{\Omega} (\tilde{p} - p^h)^2 d\Omega$$

with respect to the \tilde{p}_A 's. The resulting equations emanate from

$$\frac{\partial}{\partial \tilde{p}_A} \int_{\Omega} (\tilde{p} - p^h)^2 d\Omega = 0$$

for $A = 1, 2, \dots, n_{np}$.

The indices A, B take on the values $1, 2, \dots, n_{np}$. The construction of \mathbf{Y} and \mathbf{P} is performed in the usual element-by-element fashion, viz.⁸

$$\mathbf{Y} = \sum_{e=1}^{n_{el}} (\mathbf{y}^e), \quad \mathbf{P} = \sum_{e=1}^{n_{el}} (\mathbf{p}^e) \quad (4.4.32)$$

in which

$$\mathbf{y}^e = [y_{ab}^e], \quad \mathbf{p}^e = \{p_a^e\}, \quad 1 \leq a, b \leq n_{en} \quad (4.4.33)$$

$$y_{ab}^e = \int_{\Omega^e} N_a^e N_b^e d\Omega, \quad p_a^e = p^e \int_{\Omega^e} N_a^e d\Omega \quad (4.4.34)$$

As it stands, the matrix \mathbf{Y} is symmetric and positive-definite and possesses a band-profile structure. Additional simplification may be engendered by replacing \mathbf{Y} by an associated diagonal matrix.⁹ This is done by approximating the first of (4.4.34); effective procedures are summarized as follows:

$n = 2$; rectilinear case. The 2×2 product, trapezoidal integration rule may be used to diagonalize \mathbf{y}^e , i.e.,

$$y_{ab}^e = \delta_{ab} j^e(\xi_a, \eta_a) \quad (\text{no sum on } a) \quad (4.4.35)$$

where

$$j^e = \det \begin{bmatrix} x_1^e, \xi & x_1^e, \eta \\ x_2^e, \xi & x_2^e, \eta \end{bmatrix} \quad (\text{Jacobian determinant}) \quad (4.4.36)$$

$$\mathbf{x}^e = \sum_{a=1}^{n_{en}} N_a^e \mathbf{x}_a^e \quad (4.4.37)$$

and ξ_a and η_a are the coordinates of node a in the element “natural” coordinate system. Applying the same integration scheme to the second of (4.4.34) yields

$$p_a^e = p^e j^e(\xi_a, \eta_a) \quad (4.4.38)$$

Further simplification may be achieved by approximating $j^e(\xi_a, \eta_a)$ in (4.4.35) and (4.4.38) by $j^e(0, 0)$. (When Ω^e is a parallelogram, j^e is constant and no loss of accuracy is incurred by this procedure.)

The three-dimensional case is the straightforward generalization of the above, so we omit the details.

$n = 2$; axisymmetric case. If we attempt to apply the above procedure in the axisymmetric case, we encounter a difficulty due to the factor x_1 (i.e., r) in the integrands. Along the x_2 -axis, $x_1 = 0$; hence the trapezoidal integration technique will

⁸The “assembly operators” in (4.4.32) are not the same as those used previously. Here, no boundary conditions are taken account of and there is only one degree of freedom per node.

⁹Lee et al. [52] have also found that higher accuracy is attained when \mathbf{Y} is diagonal!

produce a zero diagonal entry in \mathbf{Y} . In this case we employ a “row-sum” diagonalization technique in which

$$y_{ab}^e = \delta_{ab} \int_{\Omega'} N_a^e d\Omega \quad (\text{no sum on } a) \quad (4.4.39)$$

The above integration, which also suffices for the second of (4.4.34), may be performed by either one-point or 2×2 Gauss-Legendre integration—the latter scheme being exact.

The procedures just described render the formation, storage, and solution of the matrix equation (4.4.28) very efficient. The results produced tend to be very good at interior nodes but leave something to be desired at boundary nodes. To improve upon the results, a “correction” at each boundary node is performed. The procedure used for four-node elements may be described with the aid of an example.

Consider the mesh illustrated in Fig. 4.4.5(a). The nodes are segregated into four groups. The boundary node corrections are carried out in the following steps in order:

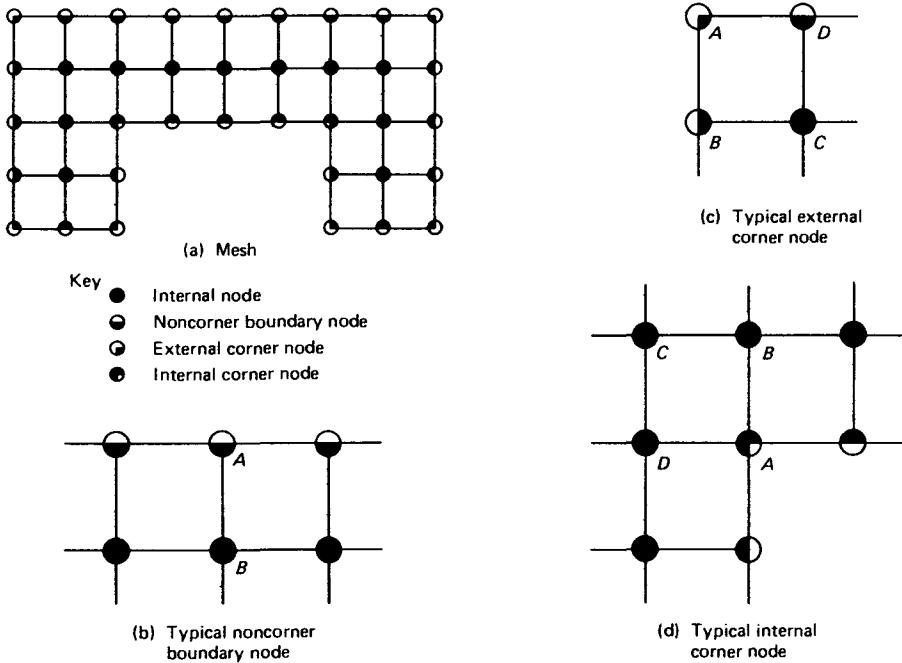


Figure 4.4.5 Example mesh for four-node element, pressure-smoothing algorithm.

Step 1: Noncorner, Boundary Nodes

A typical case of a noncorner, boundary node is depicted in Fig. 4.4.5(b). It may be observed that the unaltered value of \tilde{p}_A is actually a higher-order approximation to the

pressure at the midpoint of the line joining nodes A and B ; see Barlow [54]. Thus we redefine the \tilde{p}_A by way of linear extrapolation, i.e.,

$$\tilde{p}_A \leftarrow 2\tilde{p}_A - \tilde{p}_B \quad (4.4.40)$$

Step 2: External Corner Nodes

A typical situation is depicted in Fig. 4.4.5(c). The unaltered value of \tilde{p}_A is precisely the constant pressure p^* , because the above procedures reduce to “do-nothing” calculations at external corners. (If checkerboarding was occurring in the p^* ’s, the value of \tilde{p}_A would be grossly in error.) In this case linear extrapolation is employed through nodes B , C , and D , i.e.,

$$\tilde{p}_A \leftarrow \frac{\tilde{L}_B \tilde{p}_B + \tilde{L}_C \tilde{p}_C + \tilde{L}_D \tilde{p}_D}{L} \quad (4.4.41)$$

where

$$\tilde{L}_B = L_B + (x_{2C} - x_{2D})x_{1A} + (x_{1D} - x_{1C})x_{2A} \quad (4.4.42)$$

$$\tilde{L}_C = L_C + (x_{2D} - x_{2B})x_{1A} + (x_{1B} - x_{1D})x_{2A} \quad (4.4.43)$$

$$\tilde{L}_D = L_D + (x_{2B} - x_{2C})x_{1A} + (x_{1C} - x_{1B})x_{2A} \quad (4.4.44)$$

$$L_B = x_{1C}x_{2D} - x_{1D}x_{2C} \quad (4.4.45)$$

$$L_C = x_{1D}x_{2B} - x_{1B}x_{2D} \quad (4.4.46)$$

$$L_D = x_{1B}x_{2C} - x_{1C}x_{2B} \quad (4.4.47)$$

$$L = L_B + L_C + L_D \quad (4.4.48)$$

Step 3: Internal Corner Nodes

A typical configuration is shown in Fig. 4.4.5(d). In this case the unaltered \tilde{p}_A is essentially a weighted average of the p^* ’s associated with the three elements which have node A in common. As in Step 2, if checkerboarding has occurred, the unaltered \tilde{p}_A would be significantly in error. Again linear extrapolation is used; namely (4.4.41)–(4.4.48).

Generalizations of the above procedure may be used for smoothing pressures in some higher-order elements.

Example (Driven Cavity Flow)

A problem description is shown in Fig. 4.4.6. This problem is a much studied example of Stokes flow. Note that the boundary conditions are discontinuous at the upper corner. In the example problem the corner node velocity is set as illustrated in Fig. 4.4.7. For further discussion of the significance of the manner in which the corner discontinuity is modeled, see [53]. The calculation was performed in double precision (64 bits/floating-point word). The penalty parameter was defined by $\lambda/\mu = 10^7$. A 10×10 mesh of bilinear elements was employed with the $S1$ integration scheme. The unsmoothed pressures exhibit significant oscillations, which are removed by the method described above; see Fig. 4.4.8.

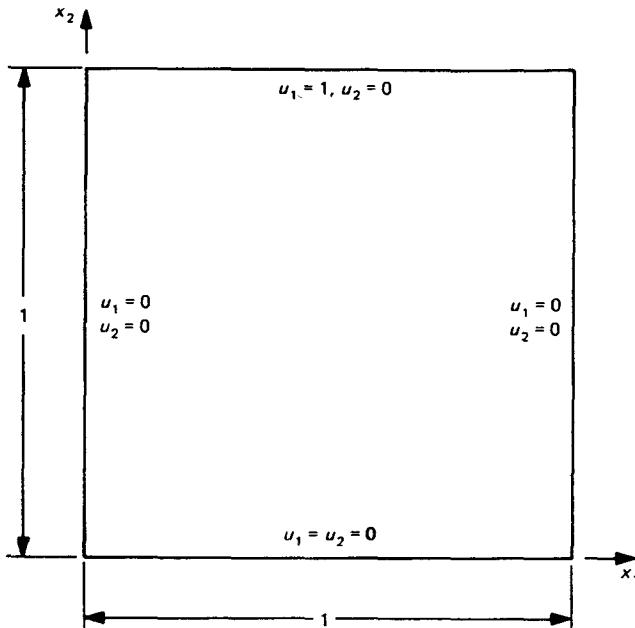


Figure 4.4.6 Driven cavity flow: problem description.

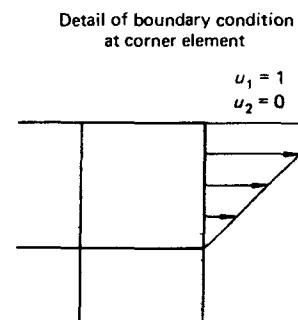


Figure 4.4.7

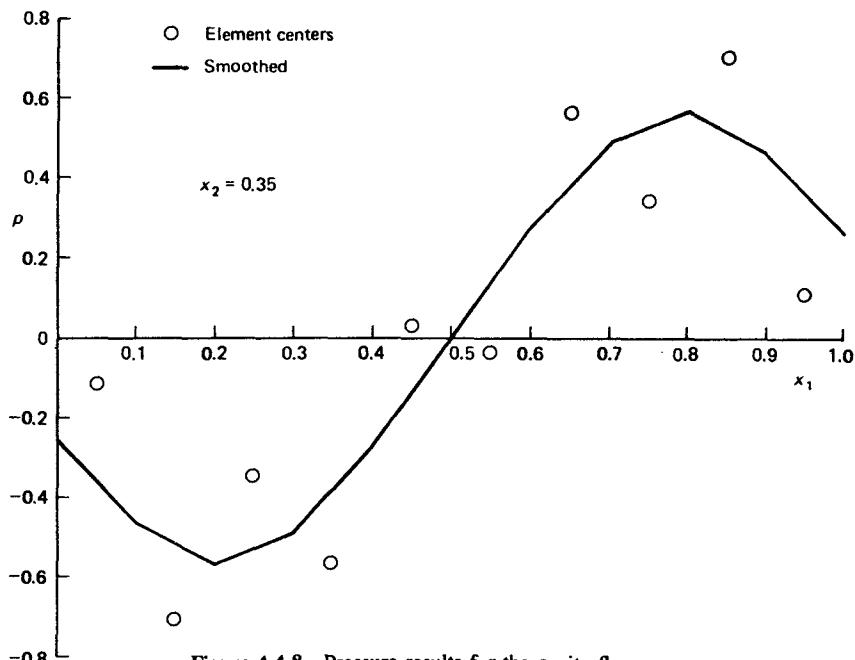


Figure 4.4.8 Pressure results for the cavity flow.

4.5. AN EXTENSION OF REDUCED AND SELECTIVE INTEGRATION TECHNIQUES

4.5.1 Axisymmetry and Anisotropy: Prelude to Nonlinear Analysis

As we pointed out in Sec. 4.4, reduced and selective integration procedures are in certain cases equivalent to the mixed formulations of Sec. 4.3. The equivalence typically holds in plane strain and three-dimensional analysis; however, it breaks down in the axisymmetric case. It is known in this case that the mixed formulation is superior, and thus it would be desirable to develop a pure displacement method, which achieves commensurate results. An approach achieving this end appeared initially in the paper of Nagtegaal et al. [55] (it is sometimes referred to as the *mean-dilatation approach*).

Another deficiency of the selective integration procedure presented in Sec. 4.4 is that it is limited to the *isotropic case*. Extension to orthotropic and anisotropic cases is ambiguous and computationally inconvenient. Generalization of the mixed formulation to these cases [56, 57] also tends to be somewhat complicated. It thus would also be desirable to develop a simple, pure displacement approach, which simultaneously extends the selective integration procedure to anisotropic cases while attaining the theoretical coherency of the mixed formulation. This is also important for eventual application to nonlinear problems because in these cases the tangent moduli (i.e., the analogs of the c_{ijkl} 's) always exhibit anisotropic character.

A procedure of the kind desired has been presented by Hughes [58] and is described next. The approach may be simply implemented by a small change of the standard technique and is shown to specialize to the selective integration and mean-dilatation formulations under appropriate hypotheses.

4.5.2 Strain Projection: The \bar{B} -approach.

We begin by recalling the displacement-method definitions of the element arrays in which the strain-displacement matrix (i.e., B) appears:

$$k^e = \int_{\Omega^e} B^T D B \, d\Omega \quad (\text{element stiffness}) \quad (4.5.1)$$

$$f^e = \int_{\Omega^e} B^T \sigma \, d\Omega \quad (\text{element internal force}) \quad (4.5.2)$$

So far we have not had much occasion to use the element internal force vector; however, its importance increases in dynamic and nonlinear problems. We recall further that the strain-displacement matrix may be expanded in terms of nodal submatrices as follows:

$$\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_{n_{en}}] \quad (4.5.3)$$

where n_{en} is the number of element nodes. In three-dimensional analysis, a typical submatrix, \mathbf{B}_a , $1 \leq a \leq n_{en}$, may be written as

$$\mathbf{B}_a = \begin{bmatrix} B_1 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & B_3 \\ \hline 0 & B_3 & B_2 \\ B_3 & 0 & B_1 \\ B_2 & B_1 & 0 \end{bmatrix} \quad (4.5.4)$$

in which

$$B_i = \partial N_a / \partial x_i, \quad 1 \leq i \leq 3 \quad (4.5.5)$$

where N_a is the shape function associated with node a , and x_i is the i th Cartesian coordinate.

These expressions are standard but must be modified to be successful in application to nearly incompressible cases. Let $\mathbf{B}_a^{\text{dil}}$ denote the dilatational part of \mathbf{B}_a , i.e.,

$$\mathbf{B}_a^{\text{dil}} = \frac{1}{3} \begin{bmatrix} B_1 & B_2 & B_3 \\ B_1 & B_2 & B_3 \\ B_1 & B_2 & B_3 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (4.5.6)$$

Exercise 1. Derive (4.5.6). Hint: Employ the definition of *dilatational components*, $\frac{1}{3} \delta_{ij} u_{k,k}^h = \frac{1}{3} \delta_{ij} \sum_{a=1}^{n_{en}} N_{a,k} d_{ka}^h$, and consult Chapter 2 for necessary background.

The deviatoric part of \mathbf{B}_a is then defined by

$$\mathbf{B}^{\text{dev}} = \mathbf{B}_a - \mathbf{B}_a^{\text{dil}} \quad (4.5.7)$$

(This follows from the definition of *deviatoric components*, $u_{(i,j)}^h = \frac{1}{3} \delta_{ij} u_{k,k}^h$.) To achieve an effective formulation for nearly incompressible applications, $\mathbf{B}_a^{\text{dil}}$ needs to be replaced by an “improved” dilatational contribution, which we shall denote by $\bar{\mathbf{B}}_a^{\text{dil}}$:

$$\bar{\mathbf{B}}_a^{\text{dil}} = \frac{1}{3} \begin{bmatrix} \bar{B}_1 & \bar{B}_2 & \bar{B}_3 \\ \bar{B}_1 & \bar{B}_2 & \bar{B}_3 \\ \bar{B}_1 & \bar{B}_2 & \bar{B}_3 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (4.5.8)$$

In place of B_a we now employ

$$\bar{B}_a = B_a^{\text{dev}} + \bar{B}_a^{\text{dil}} \quad (4.5.9)$$

which is given explicitly by

$$\bar{B}_a = \begin{bmatrix} B_5 & B_6 & B_8 \\ B_4 & B_7 & B_8 \\ B_4 & B_6 & B_9 \\ \hline 0 & B_3 & B_2 \\ B_3 & 0 & B_1 \\ B_2 & B_1 & 0 \end{bmatrix} \quad (4.5.10)$$

where

$$B_4 = \frac{\bar{B}_1 - B_1}{3} \quad (4.5.11)$$

$$B_5 = B_1 + B_4 \quad (4.5.12)$$

$$B_6 = \frac{\bar{B}_2 - B_2}{3} \quad (4.5.13)$$

$$B_7 = B_2 + B_6 \quad (4.5.14)$$

$$B_8 = \frac{\bar{B}_3 - B_3}{3} \quad (4.5.15)$$

$$B_9 = B_3 + B_8 \quad (4.5.16)$$

Clearly, the whole approach reduces to appropriate definitions of the \bar{B}_i 's. We shall present some examples after establishing some preliminary results.

We assume a quadrature rule is specified to integrate the element stiffness matrix and internal force vector. We wish to think of this rule as the "normal" one for the element.

In our first two examples, another quadrature rule is introduced, which may be thought of as a "reduced" rule. For this rule, \tilde{n}_{int} and $\tilde{\xi}_{\tilde{a}}$ denote the number of points and locations, respectively. A special set of shape functions $\tilde{N}_{\tilde{a}}$'s, are defined with nodal points at the $\tilde{\xi}_{\tilde{a}}$'s (i.e., $\tilde{N}_{\tilde{a}}(\tilde{\xi}_{\tilde{b}}) = \delta_{\tilde{a}\tilde{b}}$, $1 \leq \tilde{a}, \tilde{b} \leq \tilde{n}_{\text{int}}$).

For example, if the element under consideration was a quadrilateral and the reduced rule was the 2×2 Gauss rule, then the $\tilde{N}_{\tilde{a}}$'s would be bilinear functions interpolating the 2×2 Gauss points. (See (4.4.17) for an explicit representation.)

The general form of the \bar{B}_i 's is given by

$$\bar{B}_i(\xi) = \sum_{\tilde{\alpha}=1}^{\tilde{n}_{\text{int}}} \tilde{N}_{\tilde{\alpha}}(\xi) B_{i\tilde{\alpha}} \quad (4.5.17)$$

where ξ represents the element natural coordinates and the $B_{i\tilde{\alpha}}$'s are defined in the following examples.

Example 1 (A Generalization of Selective Integration)

The equivalent of selective integration may be attained by taking

$$B_{i\tilde{\alpha}} = B_i(\tilde{\xi}_{\tilde{\alpha}}) \quad (4.5.18)$$

As an example, consider the four-node bilinear quadrilateral. The normal rule is the 2×2 Gauss rule. Take as the reduced rule the one-point Gauss rule, so that $\tilde{n}_{\text{int}} = 1$, $\tilde{N}_1 = 1$, and $\tilde{\xi}_1 = \mathbf{0}$ (i.e., the element "center"). Then (4.5.17) reduces to

$$\bar{B}_i(\xi) = B_i(\mathbf{0}) \quad (4.5.19)$$

That is, the value at the center of the element is used to compute the dilatational contribution.

Example 2 (A Generalization of the Mean-dilatation Formulation)

An approach that generalizes the mean-dilatation formulation of Nagtegaal et al. [55] is defined by

$$B_{i\tilde{\alpha}} = \sum_{\tilde{b}=1}^{\tilde{n}_{\text{int}}} (m^{-1})_{\tilde{\alpha}\tilde{b}} \int_{\Omega^e} \tilde{N}_{\tilde{b}} B_i d\Omega \quad (4.5.20)$$

where

$$m = [m_{\tilde{\alpha}\tilde{b}}] \quad (4.5.21)$$

$$m_{\tilde{\alpha}\tilde{b}} = \int_{\Omega^e} \tilde{N}_{\tilde{\alpha}} \tilde{N}_{\tilde{b}} d\Omega, \quad 1 \leq \tilde{\alpha}, \tilde{b} \leq \tilde{n}_{\text{int}} \quad (4.5.22)$$

The $\tilde{\xi}_{\tilde{\alpha}}$'s play no role in this formulation. Note that (4.5.20) through (4.5.22) arise from the Galerkin equation $(\tilde{N}_{\tilde{b}}, \bar{B}_i - B_i) = 0$ and (4.5.17).

To see that (4.5.20) through (4.5.22) specialize to the mean-dilatation element of Nagtegaal, consider the same setup as in the previous example. In this case (4.5.17) becomes

$$\bar{B}_i(\xi) = \frac{\int_{\Omega^e} B_i d\Omega}{\int_{\Omega^e} d\Omega} \quad (4.5.23)$$

Thus the mean value of B_i is used to compute the dilatational contribution.

In passing, we may note that this generalization of the mean-dilatation formulation [i.e., (4.5.20) through (4.5.22)] is somewhat more involved than the generalization of the selective integration formulation defined by (4.5.18).

Example 3 (Another Generalization of the Mean-dilatation Formulation)

In this example we show how the present formulation includes the higher-order generalization of the mean-dilatation formulation originally suggested in [55].

For this case there is no need to introduce a reduced quadrature rule. However, we may still employ (4.5.20) through (4.5.22), where the $\tilde{N}_{\bar{\alpha}}$'s are here interpreted as an arbitrary set of \tilde{n}_{int} functions.

As an example of how this procedure might be used, consider the nine-node Lagrange quadrilateral. Assume that the 3×3 rule is to play the role of the normal quadrature rule. Take $\tilde{n}_{int} = 3$ and assume the $\tilde{N}_{\bar{\alpha}}$'s are 1 , $x_1 - x_1(0)$, and $x_2 - x_2(0)$. This assumption was proposed in [55]. This element is an analog of the nine-node, linear pressure quadrilateral of Fig. 4.3.4(h).

The main significance of the preceding ideas is that the number of functions is not restricted to be the number of points of a quadrature rule. This restriction has limited the success of the selective integration procedure to specific elements.

Remark

Formulations of the type described in Examples 2 and 3 may be described as *strain projections* in that the dilatational strain is projected onto a simple set of functions by way of the Galerkin method. Example 1 is not generally a projection but is rather an interpolation procedure. In specific cases, however, it may be equivalent to a projection. This issue is discussed further in Hughes and Malkus [59].

Torsionless Axisymmetric Analysis

The preceding ideas may be extended to torsionless axisymmetry in a straightforward manner (compare Sec. 2.12). It is convenient in this case to introduce a cylindrical coordinate system in which

x_1 = the radial coordinate

x_2 = the axial coordinate

x_3 = the circumferential coordinate

Proceeding as in the development of (4.5.10), we are led to

$$\bar{B}_a = \begin{bmatrix} B_{12} & B_6 \\ B_{10} & B_7 \\ \hline B_2 & B_1 \\ \hline \cdots & \cdots \\ B_{11} & B_6 \end{bmatrix} \quad (4.5.24)$$

where

$$B_0 = \frac{N_a}{x_1} \quad (4.5.25)$$

$$B_{10} = B_4 + \frac{\bar{B}_0 - B_0}{3} \quad (4.5.26)$$

$$B_{11} = B_0 + B_{10} \quad (4.5.27)$$

$$B_{12} = B_1 + B_{10} \quad (4.5.28)$$

In (4.5.26) \bar{B}_0 is defined by (4.5.17).

Plane Strain Analysis

The plane strain case may be obtained from (4.5.24) through (4.5.28) by setting $\bar{B}_0 = B_0 = 0$ and interpreting the x_i 's as Cartesian coordinates. It is important to note that for plane strain \bar{B}_a has dimension 4×3 , whereas the standard B_a has dimension 3×3 (compare (2.9.3)).

Remarks

1. The preceding ideas, first proposed in Hughes [58, 60], apply to the nonlinear case if we employ an “updated” formulation for element array calculation.
2. Note that in all cases the \bar{B} -formulation involves only one numerical integration “do loop” for element array calculations, and thus implementation is not significantly different from the standard uniform integration case (see Sec. 3.10).
3. The relationship of the present procedures with mixed formulations has been explored in Hughes and Malkus [59]. Simo and Hughes [61] and Hughes et al. [62] have established the equivalence of the \bar{B} -approach with finite element methods derived from the Hu-Washizu variational formulation.
4. The \bar{B} -formulation is now widely used in large-scale linear and nonlinear finite element systems. Due to the importance of the method, it was chosen for inclusion in program DLEARN (see Chapter 11). Specifically, the mean-dilatational four-node element is available for plane strain and axisymmetric analysis. The element mean values of the shape functions and their global derivatives are calculated in subroutine MEANSH and stored in an array SHGBAR. The setup of (4.5.24) for each node and integration point is performed in subroutine QDCB. Note that (4.5.24) is a full matrix, in contrast to the usual \bar{B}_a . The reader may wish to study QDCB and MEANSH in order to understand fully the implementation of the mean-dilatational element.

4.6 THE PATCH TEST; RANK DEFICIENCY

In the previous two sections we have presented methods of displacement-type which involve “violations” of the classical Galerkin finite element formulation described in Chapters 1 to 3 and analyzed in Sec. 4.1. The motivation for changing the standard

formulation is clear: It simply does not always work for problems of interest. In subsequent sections we will consider even more drastic modifications to the standard formulations. In particular, so-called incompatible, or nonconforming, elements. Strang has aptly termed all these violations of the Galerkin code *variational crimes*. The theory of Sec. 4.1 simply does not apply to these cases. A more general theory is thus required, but this would take us beyond the scope of this book. It turns out that an essential ingredient in the mathematical analysis of nonstandard elements is based on an idea of Irons called the "patch test". Irons' original presentation [63] was very practically motivated and nonmathematical in nature. In fact, the test was described in terms of computational experiments, which would simultaneously assess the correctness of the formulation and its computer implementation. We shall refer to this form of the patch test as the "engineering version." A mathematical version was described and popularized in Strang and Fix [64]. Subsequently, the patch test has generated some mathematical controversy (see Stummel [65]) and undergone rumination (see Irons and Loikkanen [66] and Taylor et al. [67]). In addition, in the context of complicated theories, it is not always even clear how to pose patch tests. For these reasons faith in the patch test has eroded in some quarters. *This is unfortunate, for we firmly believe that, within the realm of problems dealt with so far in this book, the patch test is the most practically useful technique for assessing element behavior.* Thus we wish to avoid altogether the mathematically controversial facets of this subject and return to the spirit of Irons' original conception.

The Patch Test ("Engineering Version")

Basically, the patch test enables us to determine whether or not an element satisfies the completeness condition. The patch test may be applied to elements employing nonstandard features (e.g., selective integration, incompatible interpolations, etc.).

We shall say that the patch test is passed for a two-dimensional element if the states x , y , and z are exactly representable by an arbitrary "patch" of elements whenever the exact solution behaves accordingly. What does this mean? We shall use an example to clarify the idea.

Example

Consider a two-dimensional elasticity problem in which the elastic moduli are constant and there are no body forces present. Consider the arbitrary patch of elements sketched in Fig. 4.6.1. We assume the boundary nodal displacements are set in accord with linear monomials as follows:

| Test number | u_{1a}^* | u_{2a}^* |
|-------------|------------|------------|
| 1 | 1 | 0 |
| 2 | 0 | 1 |
| 3 | x_a | 0 |
| 4 | 0 | x_a |
| 5 | y_a | 0 |
| 6 | 0 | y_a |

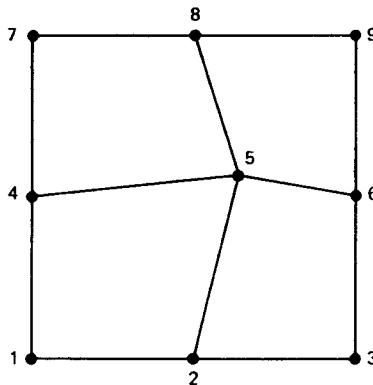


Figure 4.6.1 Arbitrary patch of two-dimensional elements.

Thus we wish to solve six displacement boundary-value problems. The patch test will be satisfied if in all cases: (1) the solution at node $a = 5$ is *exact*, i.e., takes on the value given in the above table, and (2) the displacement gradients, and consequently strains and stresses, are *exact* within each element.

In constructing an arbitrary patch, it is important to use an irregular geometry, because some elements pass the patch test in certain special configurations but not others. An example of this is described in Sec. 4.7.

The reader should have no trouble generalizing the above idea to three dimensions and specializing it to the scalar heat-conduction case.

Rank Deficiency

Consider a two-dimensional bilinear elasticity element. The element stiffness may be expressed as

$$\underbrace{\mathbf{k}^e}_{8 \times 8} = \int_{\Omega^e} \underbrace{\mathbf{B}^T}_{8 \times 3} \underbrace{\mathbf{D}}_{3 \times 3} \underbrace{\mathbf{B}}_{3 \times 8} d\Omega \quad (4.6.1)$$

If exact integration, or 2×2 Gaussian integration, is performed on the integral in (4.6.1), the resulting stiffness will have rank 5. This may be seen to be correct as follows: The rank will equal the number of degrees of freedom, namely, eight, minus the number of rigid-body modes, which in this case is three (i.e., two translations and one rotation). However, if instead of 2×2 Gaussian integration (or higher), we use the one-point Gauss rule, the rank will be reduced to three. This can be seen algebraically by examining the one-point stiffness matrix:

$$\mathbf{k}_{1\text{-pt.}}^e = 4(j\mathbf{B}^T \mathbf{D} \mathbf{B}) \Big|_{\tilde{\xi}=\tilde{\eta}=0} \quad (4.6.2)$$

(Recall that j denotes the Jacobian determinant and 4 is the weight of the one-point rule.) By virtue of the fact that \mathbf{D} is a 3×3 matrix, the rank of $\mathbf{k}_{1\text{-pt.}}^e$ can be no greater than three. It can be verified that it is, in fact, exactly three. This means that there are two additional (spurious) zero-energy modes. These are the *hourglass modes* depicted in Fig. 4.6.2. In a mesh formed from such elements, the hourglass modes can communicate to form a singular, or nearly singular, global stiffness \mathbf{K} . In the singular case, no solution is possible; in the nearly singular case, hourglass “instabilities” may

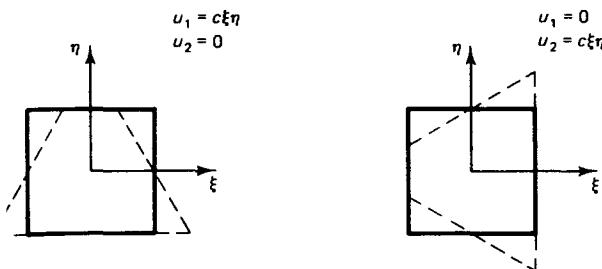


Figure 4.6.2 Hourglass modes; c is an arbitrary constant.

pollute the displacement field. Our initial reaction is, of course, to prudently avoid elements of this type. However, with appropriate modifications, one-point integration elements have proven to be very effective. We shall return to this theme in Sec. 4.8.

When reduced integration is used, we must be aware of the possibility of generating spurious zero-energy modes. (These are sometimes referred to in the literature as *mechanisms*.) A way of numerically assessing the presence of zero-energy modes is to calculate the eigenvalues of the element stiffness, which are defined by

$$\det(k^e - \lambda I) = 0 \quad (4.6.3)$$

The number of zero eigenvalues equals the number of zero-energy modes. Another way is to perform a Crout factorization of k^e and count the number of zero “pivots.” This procedure is employed in the DLEARN program and is described in Exercise 3 of Sec. 4.9.

Examples of Elements with Spurious Zero-energy Modes

Example 1

The correct rank for the eight-node brick element in three dimensions is 18 (i.e., 24 degrees of freedom minus 6 rigid-body modes). The $2 \times 2 \times 2$ Gaussian rule is sufficient to assure correct rank. One-point Gaussian integration produces a rank of 6. That is, there are 12 spurious zero-energy modes.

Example 2

The eight-node serendipity quadrilateral with reduced 2×2 quadrature possesses one spurious zero-energy mode; see Fig. 4.6.3. This mode is often described as “non-

$$u_1 = c\xi(\eta^2 - \frac{1}{3})$$

$$u_2 = -c\eta(\xi^2 - \frac{1}{3})$$

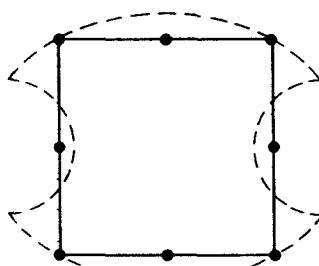


Figure 4.6.3 The spurious zero-energy mode of the reduced 2×2 Gaussian integration eight-node serendipity quadrilateral; c is an arbitrary constant.

"communicable" because in an assembly of two or more elements no zero-energy modes are present. However, in certain situations problems can still occur. For example, consider the configuration illustrated in Fig. 4.6.4. The type of loading applied to the steel base can arouse the spurious zero-energy mode. The contiguous soil elements may be too soft to adequately resist it.

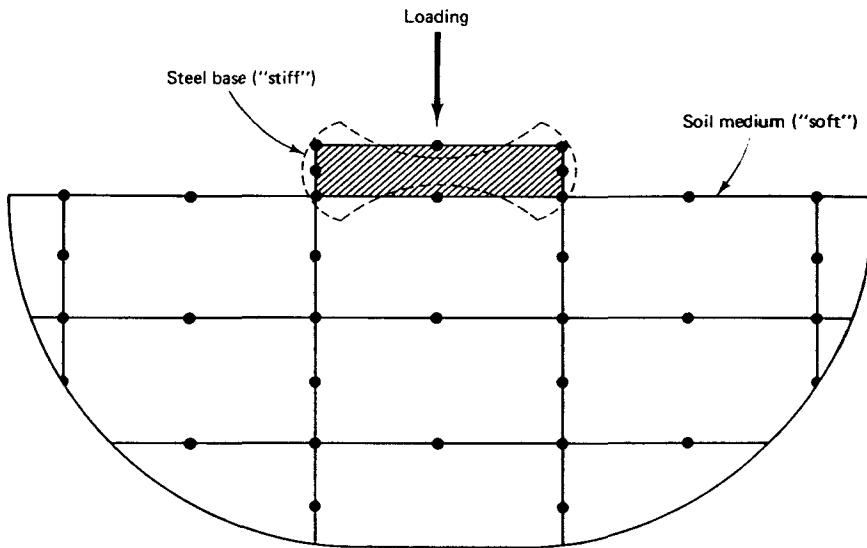


Figure 4.6.4 Stiff steel base on a soft soil medium. Illustration of potential modeling difficulties with the 2×2 reduced integration serendipity element.

Exercise. Consider a square bilinear two-dimensional element. Assume that the global coordinates are coincident with the local coordinates and differ only in length-scale. Thus we may write $x = h\xi/2$ and $y = h\eta/2$. All derivatives of the hourglass patterns (see Fig. 4.6.2) vanish identically at $\xi = \eta = 0$. This fact and the expression for strain energy,

$$\frac{1}{2}(\mathbf{d}^e)^T \mathbf{k}_{1\text{-pt.}}^e \mathbf{d}^e = 2(j u_{(i,j)}^h c_{ijkl} u_{(k,:)}^h) \Big|_{\bar{\xi} = \bar{\eta} = 0}$$

illustrate that hourglass modes possess zero strain energy when one-point quadrature is used to obtain the stiffness matrix. The following will enable the reader to verify this and related facts. Consider eight linearly independent deformation patterns for the square element:

1. $\begin{cases} u_1 = c \\ u_2 = 0 \end{cases}$ (x-translation)
2. $\begin{cases} u_1 = 0 \\ u_2 = c \end{cases}$ (y-translation)
3. $\begin{cases} u_1 = c\eta \\ u_2 = -c\xi \end{cases}$ (infinitesimal rotation)

4. $\begin{cases} u_1 = c\xi\eta \\ u_2 = 0 \end{cases}$ (x-hourglass)
5. $\begin{cases} u_1 = 0 \\ u_2 = c\xi\eta \end{cases}$ (y-hourglass)
6. $\begin{cases} u_1 = c\xi \\ u_2 = 0 \end{cases}$ (uniform x-extension)
7. $\begin{cases} u_1 = 0 \\ u_2 = c\eta \end{cases}$ (uniform y-extension)
8. $\begin{cases} u_1 = c\eta \\ u_2 = c\xi \end{cases}$ (uniform shear)

Calculate the strain energy in each pattern for an exactly integrated stiffness and a one-point quadrature stiffness. Deduce the following facts:

- a. For both stiffnesses the rigid-body modes (i.e., 1 to 3) produce zero strain energy.
 - b. For both stiffnesses the homogeneous strain states (i.e., 6 to 8) produce nonzero strain energy.
 - c. One-point quadrature produces zero strain energy in the hourglass modes, whereas exact integration produces nonzero strain energy.
-

4.7 NONCONFORMING ELEMENTS

Thus far we have always constructed our finite element spaces such that they are proper subsets of the corresponding spaces employed in the exact weak statement of the problem. For example, this is written as follows:

$$\mathcal{S}^h \subset \mathcal{S} \quad (4.7.1)$$

$$\mathcal{V}^h \subset \mathcal{V} \quad (4.7.2)$$

Recall the meaning of (4.7.1) and (4.7.2): If $\mathbf{u}^h \in \mathcal{S}^h$, then $\mathbf{u}^h \in \mathcal{S}$, and so \mathbf{u}^h inherits the properties defining \mathcal{S} . That is, if members of \mathcal{S} satisfy certain boundary conditions and possess some degree of “regularity” (e.g., have square-integrable first derivatives), then members of \mathcal{S}^h also possess these properties, and likewise for \mathcal{V} and \mathcal{V}^h .

In Chapters 1 and 3 we constructed finite element shape functions, which were *continuous* across element boundaries. Thus, although derivatives are typically *discontinuous* across element boundaries, these functions still possess square-integrable first derivatives. This is the class of C^0 finite element interpolations and is appropriate for problems in which the expression for $a(\cdot, \cdot)$ contains products of first, but no higher, derivatives (such as heat conduction, or classical elasticity).

It is important to note that (4.7.1) and (4.7.2) are explicitly used in establishing convergence in Sec. 4.1. Thus there is no guarantee that convergence will be achieved if (4.7.1) and (4.7.2) are not respected. Nevertheless, finite elements have been

proposed, and in fact are commonly used, in which C^0 continuity is violated, resulting in functions that do not possess square-integrable first derivatives. (Dirac delta distributions exist in the derivatives of such functions, and the product of these is not well defined.) Elements of this type are referred to as *incompatible*, or *nonconforming*. We shall illustrate the ideas involved by considering the *incompatible modes element* originally proposed by Wilson et al. [68].

Incompatible Modes Element

Version 1, Wilson et al. It may be recalled that the standard four-node quadrilaterals employed in the beam-bending elasticity example of Sec. 4.4 (see Figs. 4.4.1 and 4.4.2 and Table 4.4.1) attained a level of accuracy of approximately 90% in the case of Poisson's ratio = 0.3. In this example, 32 elements were used to model the upper half of the beam (see Fig. 4.4.2). For "bending" cases like this, the accuracy level attained by simple elements tends to be somewhat disappointing. If only one element is used through the thickness, the accuracy degrades further, leading to worthless results. (The reason for this will be made clear in a moment.) It thus would appear worthwhile to attempt to modify the basic four-node element so that improved behavior is attained in bending situations. This is the motivation that prompted Wilson et al. to propose the incompatible modes approach. They noted that if a bending moment is applied to a single element, the element responds in shear rather than bending, and this spurious shearing is responsible for the overly stiff behavior (see Fig. 4.7.1). They further observed that one way of producing correct behavior in this situation was to add quadratic modes of deformation corresponding to the exact pure bending solution.¹⁰ This is the essence of the incompatible modes approach. Starting with the standard expansion in terms of bilinear shape functions for a typical element, quadratic modes are added as follows:

$$u^h(\xi, \eta) = \sum_{a=1}^4 N_a(\xi, \eta) d_a^e + \sum_{a=5}^6 N_a(\xi, \eta) \alpha_a^e \quad (4.7.3)$$

where the first four N_a 's are the standard bilinear shape functions (see Sec. 3.2),

$$\begin{aligned} N_5(\xi, \eta) &= 1 - \xi^2 \\ N_6(\xi, \eta) &= 1 - \eta^2 \end{aligned} \quad \text{(incompatible modes)} \quad (4.7.4)$$

and the α_a 's are *generalized displacements* (as distinguished from nodal displacements). The incompatible modes are sketched in Fig. 4.7.2. There are no nodal points associated with the additional modes and the α_a 's may be thought of as "internal" element degrees of freedom. The incompatible modes result in the element displacements being discontinuous between nodes (see Fig. 4.7.2).

Equation (4.7.3) is used in the definition of the element stiffness, or wherever the strain-displacement matrix, B , appears, but not elsewhere, i.e., not in the

¹⁰Clearly, another possibility would be simply to employ higher-order elements, such as the eight-node serendipity, or nine-node Lagrange, quadrilateral. As might be anticipated, excellent results may be obtained in this fashion. Here, however, the objective is to achieve good behavior *without* introducing additional nodes to the basic four-node pattern.

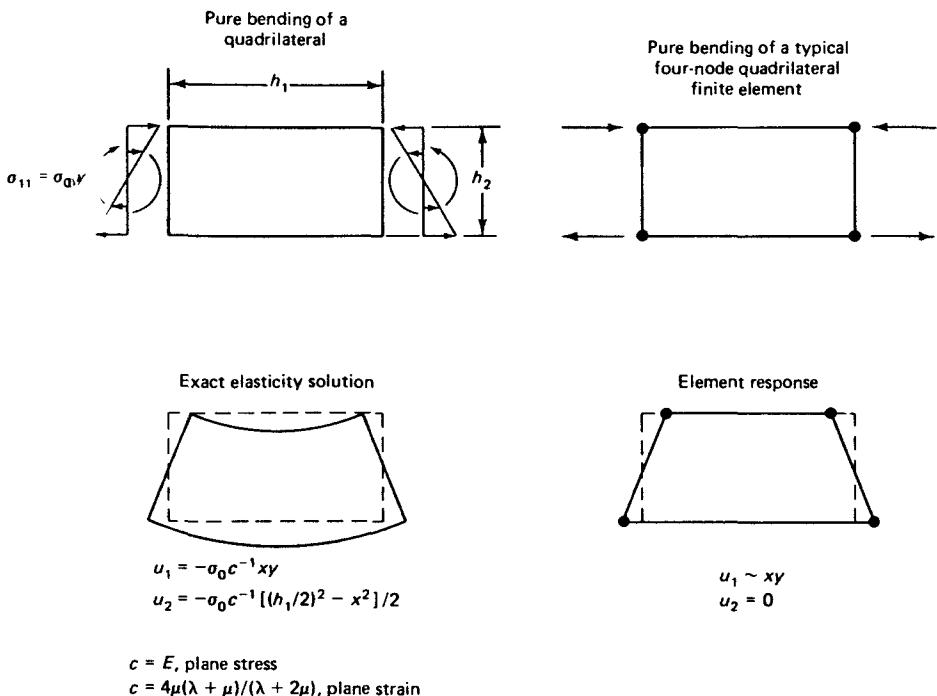


Figure 4.7.1

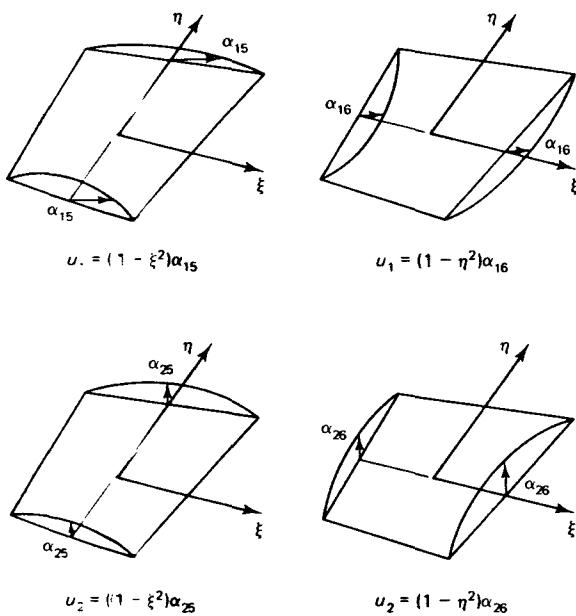


Figure 4.7.2 Incompatible displacement modes.

definition of element body and surface forces. To see why this is the case, consider a *constant* body force, \boldsymbol{f} . It is reasonable to insist that the sum of the element forces produced by \boldsymbol{f} equals $\text{vol}(\Omega^\epsilon) \boldsymbol{f}$, where $\text{vol}(\Omega^\epsilon) = \int_{\Omega^\epsilon} d\Omega$. This is easily seen to be the case for any set of shape functions satisfying $\sum_{a=1}^n N_a = 1$. For example, in the case of the standard bilinear quadrilateral, we have

$$\begin{aligned}\sum_{a=1}^4 f_a^\epsilon &= \sum_{a=1}^4 \int_{\Omega^\epsilon} N_a \boldsymbol{f} d\Omega \\ &= \int_{\Omega^\epsilon} \underbrace{\sum_{a=1}^4 N_a}_{1} d\Omega \boldsymbol{f} \\ &= \int_{\Omega^\epsilon} d\Omega \boldsymbol{f} \\ &= \text{vol}(\Omega^\epsilon) \boldsymbol{f}\end{aligned}\tag{4.7.5}$$

If we insist on including forces associated with the additional modes, the sum total is too great, as may be seen by calculating as above:

$$\sum_{a=1}^6 \int_{\Omega^\epsilon} N_a \boldsymbol{f} d\Omega = \left[\text{vol}(\Omega^\epsilon) + \underbrace{\sum_{a=5}^6 \int_{\Omega^\epsilon} N_a d\Omega}_{>0} \right] \boldsymbol{f}\tag{4.7.6}$$

A similar conclusion may be drawn for surface forces.

Because the generalized displacements associated with the incompatible modes are unique to each element, they may be eliminated on the element level. (This process is analogous to the elimination of pressure degrees of freedom in the case of discontinuous pressure fields; see Sec. 4.3, Procedure 3.) This leads to a “condensed” element stiffness of normal size for a four-node quadrilateral, i.e., 8×8 . The process of deriving the stiffness is given as follows:

We begin with the standard definition of \mathbf{k} , namely,

$$\mathbf{k} = \int_{\Omega^\epsilon} \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega\tag{4.7.7}$$

where

$$\mathbf{B} = [\mathbf{B}_1 \quad \mathbf{B}_2 \quad \mathbf{B}_3 \quad \mathbf{B}_4 \quad | \quad \mathbf{B}_5 \quad \mathbf{B}_6]\tag{4.7.8}$$

The \mathbf{B}_a 's are defined in the usual way. The appearance of \mathbf{B}_5 and \mathbf{B}_6 is due to the presence of the incompatible modes. Each \mathbf{B}_a has dimension 3×2 ; consequently, \mathbf{B} has dimension 3×12 and \mathbf{k} has dimension 12×12 . It can be written in the following partitioned form:

$$\mathbf{k} = \begin{bmatrix} \mathbf{k}_{dd} & | & \mathbf{k}_{d\alpha} \\ \hline \mathbf{k}_{\alpha d} & | & \mathbf{k}_{\alpha\alpha} \end{bmatrix}\tag{4.7.9}$$

Note that \mathbf{k}_{dd} is the usual 8×8 stiffness of the four-node quadrilateral, $\mathbf{k}_{d\alpha} = \mathbf{k}_{ad}^T$ has dimension 8×4 , and $\mathbf{k}_{\alpha\alpha}$ has dimension 4×4 . By virtue of the fact that there are no element forces corresponding to the incompatible modes, we can write

$$\mathbf{k}_{ad}\mathbf{d}^\epsilon + \mathbf{k}_{\alpha\alpha}\boldsymbol{\alpha}^\epsilon = \mathbf{0} \quad \text{where } \boldsymbol{\alpha}^\epsilon = \begin{Bmatrix} \alpha_5^\epsilon \\ \alpha_6^\epsilon \end{Bmatrix} \quad (4.7.10)$$

and solve for $\boldsymbol{\alpha}^\epsilon$ in terms of the nodal displacements¹¹:

$$\boldsymbol{\alpha}^\epsilon = -\mathbf{k}_{\alpha\alpha}^{-1}\mathbf{k}_{ad}\mathbf{d}^\epsilon \quad (4.7.11)$$

Equation (4.7.11) may then be used to eliminate the $\boldsymbol{\alpha}^\epsilon$'s from the equations. This process leads to the following 8×8 element stiffness:

$$\boxed{\widetilde{\mathbf{k}} = \mathbf{k}_{dd} - \mathbf{k}_{d\alpha}\mathbf{k}_{\alpha\alpha}^{-1}\mathbf{k}_{ad}} \quad (4.7.12)$$

Equation (4.7.12) is the stiffness of the incompatible modes element. The process used to obtain (4.7.12) is frequently referred to as *static condensation* in the structural mechanics literature. At this point, element stiffness assembly may proceed in the usual way.

Remark

The preceding developments define the original incompatible modes element of Wilson et al. A theoretical investigation was given by Lesaint [69]. Numerical studies indicated improved behavior for the incompatible modes element in bending applications. However, a flaw was soon discovered: When the element took the shape of an arbitrary quadrilateral (not a rectangle or a parallelogram), erratic behavior was noted. In this case the element was found to fail the patch test. Taylor et al. [70] further modified the incompatible modes element so that this deficiency was corrected. This is described next.

Version 2, Taylor et al. Taylor argued as follows: If the nodal displacements are set in accord with a given linear polynomial in the spatial coordinates, then the incompatible modes should not be activated (i.e., $\boldsymbol{\alpha}^\epsilon \equiv \mathbf{0}$). This is seen to be an alternative statement of the completeness condition, discussed in Sec. 3.1. Recall that completeness means that the displacement field throughout the element exactly coincides with the given linear polynomial. It may be concluded from (4.7.10) that $\boldsymbol{\alpha}^\epsilon = \mathbf{0}$ if $\mathbf{k}_{d\alpha}^\epsilon\mathbf{d}^\epsilon = \mathbf{0}$. By (4.7.7) and (4.7.8), (4.7.10) can be written in the form

$$\int_{\Omega^\epsilon} \mathbf{B}_a^T \boldsymbol{\sigma}^{(4)} d\Omega = - \sum_{b=5}^6 \left(\int_{\Omega^\epsilon} \mathbf{B}_a^T \mathbf{D} \mathbf{B}_b d\Omega \right) \alpha_b^\epsilon, \quad a = 5, 6 \quad (4.7.13)$$

¹¹ It may be argued that $\mathbf{k}_{\alpha\alpha}$ is nonsingular.

where

$$\begin{aligned}\boldsymbol{\sigma}^{(4)}(\mathbf{x}) &= \mathbf{D}\boldsymbol{\epsilon}^{(4)}(\mathbf{x}) \\ &= \mathbf{D}\left(\sum_{b=1}^4 \mathbf{B}_b(\mathbf{x})\mathbf{d}_b^e\right)\end{aligned}\quad (4.7.14)$$

is the stress attributable only to the nodal displacements.¹² Because the nodal values of displacement are set in accord with a linear polynomial and the *bilinear shape functions are complete* (see Chapter 3), the strains, $\boldsymbol{\epsilon}^{(4)}$, will be constant, and thus, by (4.7.14), so will $\boldsymbol{\sigma}^{(4)}$. The following conditions are thus equivalent:

$$\mathbf{0} = \int_{\Omega^e} \mathbf{B}_a^T d\Omega \quad \boldsymbol{\sigma}^{(4)} \quad \text{and} \quad \mathbf{0} = k_{ad}\mathbf{d}^e \quad (4.7.15)$$

where $a = 5, 6$ and $\boldsymbol{\sigma}^{(4)}$ is arbitrary. Consequently, it is required that

$$\mathbf{0} = \int_{\Omega^e} \mathbf{B}_a^T d\Omega, \quad a = 5, 6, \quad (4.7.16)$$

for all configurations of the element. Employing the results of Sec. 3.9, we obtain the following:

$$\int_{\Omega^e} \mathbf{B}_5 d\Omega = \int_{\Omega^e} \begin{bmatrix} N_{5,x} & 0 \\ 0 & N_{5,y} \\ N_{5,y} & N_{5,x} \end{bmatrix} d\Omega \quad (4.7.17)$$

$$= 2 \int_{-1}^{+1} \int_{-1}^{+1} \begin{bmatrix} -\xi y_{,\eta} & 0 \\ 0 & \xi x_{,\eta} \\ \xi x_{,\eta} & -\xi y_{,\eta} \end{bmatrix} d\xi d\eta$$

$$\int_{\Omega^e} \mathbf{B}_6 d\Omega = \int_{\Omega^e} \begin{bmatrix} N_{6,x} & 0 \\ 0 & N_{6,y} \\ N_{6,y} & N_{6,x} \end{bmatrix} d\Omega \quad (4.7.18)$$

$$= 2 \int_{-1}^{+1} \int_{-1}^{+1} \begin{bmatrix} \eta y_{,\xi} & 0 \\ 0 & -\eta x_{,\xi} \\ -\eta x_{,\xi} & \eta y_{,\xi} \end{bmatrix} d\xi d\eta$$

From (4.7.17) and (4.7.18) we see that \mathbf{B}_5 and \mathbf{B}_6 will integrate to zero if the derivatives of x and y with respect to ξ and η are constants. It may be verified that if the element is a rectangle or parallelogram, then the derivatives are constants. How-

¹²In general, the stress for the incompatible modes element should be computed in the usual way, i.e.,

$$\boldsymbol{\sigma}(\mathbf{x}) = \mathbf{D}(\mathbf{x})\boldsymbol{\epsilon}(\mathbf{x}) = \mathbf{D}(\mathbf{x}) \left[\sum_{a=1}^4 \mathbf{B}_a(\mathbf{x})\mathbf{d}_a^e + \sum_{a=5}^6 \mathbf{B}_a(\mathbf{x})\boldsymbol{\alpha}_a^e \right]$$

Expression (4.7.14) is just an intermediate quantity introduced for purposes of derivation. However, if $\boldsymbol{\sigma}$ is computed at $\mathbf{x}(\xi, \eta) = \mathbf{x}(0, 0)$, then \mathbf{B}_5 and \mathbf{B}_6 may be omitted because they vanish identically at this location (see (4.7.17) and (4.7.18)) and, consequently, $\boldsymbol{\sigma}(\mathbf{x}(0, 0)) = \boldsymbol{\sigma}^{(4)}(\mathbf{x}(0, 0))$.

ever, in the general quadrilateral configuration, linear terms appear, and this is the reason why the element fails in these cases. Taylor proposed modifying the element by replacing the derivatives in (4.7.17) and (4.7.18) by their values at $\xi = \eta = 0$. The definitions for Taylor's B_5 and B_6 are then:

$$B_5 = \frac{2}{j} \begin{bmatrix} -\xi y_{,\eta}(0, 0) & 0 \\ 0 & \xi x_{,\eta}(0, 0) \\ \xi x_{,\eta}(0, 0) & -\xi y_{,\eta}(0, 0) \end{bmatrix} \quad (4.7.19)$$

$$B_6 = \frac{2}{j} \begin{bmatrix} \eta y_{,\xi}(0, 0) & 0 \\ 0 & -\eta x_{,\xi}(0, 0) \\ -\eta x_{,\xi}(0, 0) & \eta y_{,\xi}(0, 0) \end{bmatrix} \quad (4.7.20)$$

With the above modifications, the element passes the patch test for arbitrary configurations. Calculations in support of this assertion are presented in [70]. Clearly, version 2 is suitable for general analysis, whereas version 1 should be used only in rectangular or parallelogrammic configurations.

Example

The bending behavior of the incompatible modes element is assessed on the elasticity solution previously considered (see Figs. 4.4.1 and 4.4.2 and Table 4.4.1). Results are presented in Table 4.7.1. The superiority of the incompatible modes element in the present situation is self-evident.

TABLE 4.7.1 Comparison of Bilinear Quadrilateral Elements on Beam-bending Problem. Tip Displacement Normalized by Exact Solution

| | <i>U</i> 2 | <i>S</i> 1 | <i>U</i> 2 with incompatible modes |
|---------------|------------|------------|------------------------------------|
| $\nu = 0.3$ | 0.904 | 0.912 | 0.995 |
| $\nu = 0.499$ | 0.334 | 0.937 | 0.992 |

Key:

*U*2 2 × 2 uniform integration (exact in the present cases)

*S*1 selective integration (2 × 2 on μ -term, one-point on λ -term)

The behavior of the elements as Poisson's ratio approaches $\frac{1}{2}$ is shown in Fig. 4.7.3.¹³ As may be seen, the standard four-node element and linear triangle deteriorate rapidly, as is to be expected. The selective integration element and incompatible modes element maintain their accuracy even for very large ratios of λ/μ . Deterioration commences at about $\lambda/\mu = 10^9$ due to the fact that rounding errors in equation solving

¹³ Recall that $\lambda/\mu = 2\nu/(1 - 2\nu)$.

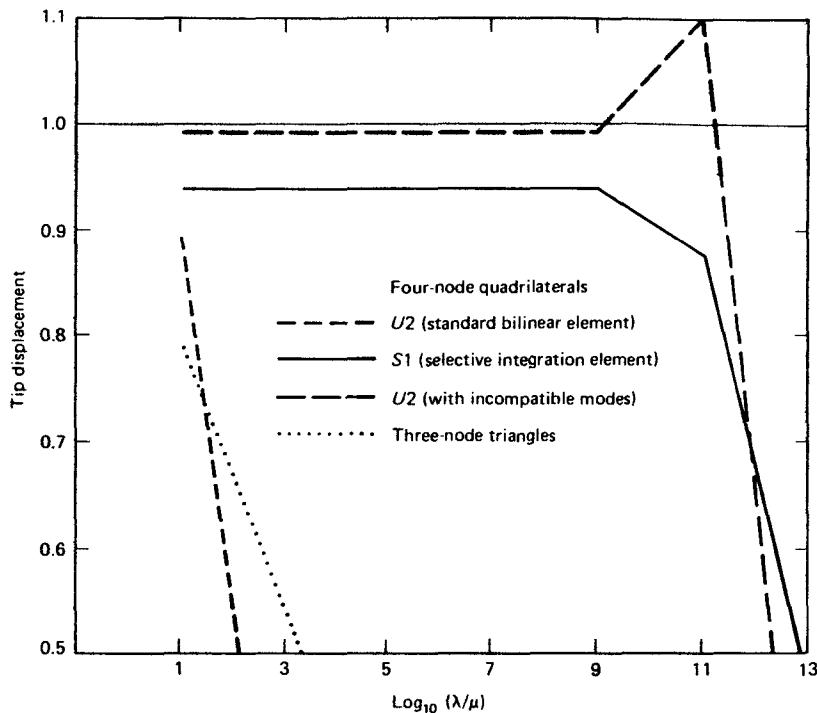


Figure 4.7.3

overwhelm the λ contributions, which are nine orders of magnitude smaller than the μ contributions. (A 64-bit floating-point word was used in the present calculations. This corresponds to approximately 16 digits.) The point at which deterioration begins is beyond the region of practical importance.¹⁴

To explain the good behavior of the incompatible modes element in nearly incompressible applications, we may attempt to calculate its constraint ratio. In this case the number of equations is six due to the presence of the four generalized displacements associated with the incompatible modes. Because the divergence of the displacement field is a linear polynomial, there are three incompressibility constraints. Consequently, the constraint ratio is $\frac{6}{3} = 2$, the optimal value.

Remarks

1. The incompatible modes element behaves well in bending, even with only one element through the thickness.
2. Due to the additional shape functions present and static condensation, the element is somewhat expensive to form. An alternative, but more economical element, which attains virtually identical results, is considered in the next section.

¹⁴ The floating-point word length determines when deterioration commences in situations like this. If a shorter word is used, only smaller ratios of λ/μ can be effectively handled.

3. The following steps are required to generalize the implementation of the standard four-node element to the incompatible modes element:

- i. Add shape functions N_5 and N_6 and their derivatives to the shape function subroutine. The derivatives of x and y with respect to ξ and η need to be evaluated at $\xi = \eta = 0$ in these calculations as per the Taylor modification in version 2.
- ii. The B matrix needs to be enlarged to account for the incompatible modes. With particular reference to implementations 2 and 3 of Sec. 3.10, increase the row/column index of the element nodal do loops (i.e., NEN) from four to six to accommodate the incompatible modes.
- iii. Statically condense the stiffness.

4. The generalization to three dimensions is straightforward.

5. The Wilson-Taylor element is used in several large linear structural analysis codes. Its use in nonlinear analysis proves somewhat awkward and thus it does not appear to be widely used in the nonlinear regime.

Exercise 1 (Linear Nonconforming Triangle). Consider a straight-edged triangle with three nodes located at the midsides (see Fig. 4.7.4). Assume the following linear shape functions:

$$N_1(r, s, t) = 1 - 2r \quad (4.7.21)$$

$$N_2(r, s, t) = 1 - 2s \quad (4.7.22)$$

$$N_3(r, s, t) = 1 - 2t \quad (4.7.23)$$

where r, s, t are triangular coordinates (see Appendix 3.I). Note that the shape functions are nonconforming. (Sketch them!)

- a. Show that $\int_{\Omega^e} N_a N_b d\Omega = 0$ if $a \neq b$. (This orthogonality property is very useful in dynamics.)
- b. Along with (4.7.21) through (4.7.23), assume a constant pressure field. Show that the incompressibility constraint ratio is 3. (Thus the element does not lock. It has been used successfully in incompressible problems.)
- c. Derive shape functions for the analogous tetrahedron with nodes on the centers of the faces.

These elements have a number of other interesting properties. See Thomasset [71] for additional details. Unfortunately, the practical usefulness of elements of this type in elasticity is limited by the fact that they can give rise to spurious mechanisms as illustrated in Fig. 4.7.5. Element II is only attached to element I at node A. Therefore element II can rigidly rotate about node A.

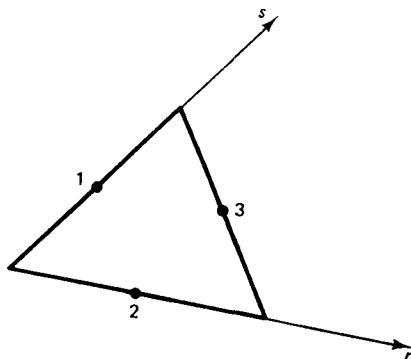


Figure 4.7.4

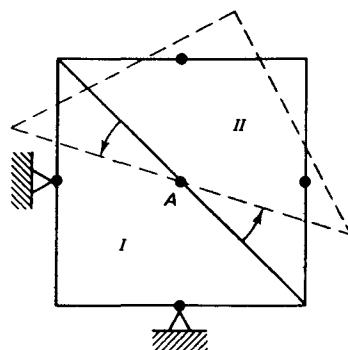


Figure 4.7.5

4.8 HOURGLASS STIFFNESS

Use of one-point quadrature to formulate the four-node bilinear element stiffness is quite economical (it requires approximately one-fourth the calculational effort of the standard 2×2 rule), but it results in rank deficiency as we observed in Sec. 4.6. The two modes that are rendered singular are the so-called hourglass patterns illustrated in Fig. 4.6.2. Kosloff and Frazier [72] have presented a very simple way to reintroduce the stiffness of the hourglass modes, which has two important attributes: (1) The superior bending behavior of the (modified) incompatible modes element is obtained;¹⁵ (2) the element formulation costs are only a fraction greater than the one-point quadrature element and therefore are much less expensive than the incompatible modes and standard bilinear elements. The stiffness for the Kosloff-Frazier element consists of the one-point quadrature stiffness matrix plus an hourglass stiffness matrix which is constructed from the exact pure bending modes (see Fig. 4.7.1). The derivation proceeds as follows:

First, consider the exact stress and displacement field illustrated in Fig. 4.7.1. The nodal displacements may be written as

$$d_{1a} = \frac{-\sigma_0}{4c} \text{vol}(\Omega^e) \phi_a \quad (4.8.1)$$

$$d_{2a} = 0 \quad (4.8.2)$$

$$\phi_a = (-1)^a \quad (\text{hourglass mode}) \quad (4.8.3)$$

where $a = 1, 2, 3, 4$, and $\text{vol}(\Omega^e) = h_1 h_2$. Recall that $c = E$ in plane stress, and $c = 4\mu(\lambda + \mu)/(\lambda + 2\mu)$ in plane strain. By taking the dot product of (4.8.1) with

¹⁵In the isotropic case, in which the material properties are also constant, the Kosloff-Frazier element is identical to the incompatible modes element.

ϕ_a we can obtain the following explicit expression for σ_0 :

$$\sigma_0 = -\frac{c}{\text{vol}(\Omega^\epsilon)} \left(\sum_{a=1}^4 \phi_a d_{1a} \right) \quad (4.8.4)$$

The quantity $\sum_{a=1}^4 \phi_a d_{1a}$ in (4.8.4) is a measure of the amplitude of the hourglass mode.

Next we wish to calculate the internal force vector at node a due to the given stress field:

$$\begin{aligned} f_a^{\text{int}} &= \int_{\Omega^\epsilon} \mathbf{B}_a^T \boldsymbol{\sigma} \, d\Omega = \int_{\Omega^\epsilon} \begin{bmatrix} N_{a,x} & 0 & N_{a,y} \\ 0 & N_{a,y} & N_{a,x} \end{bmatrix} \begin{Bmatrix} \sigma_0 y \\ 0 \\ 0 \end{Bmatrix} \, d\Omega \\ &= \sigma_0 \int_{\Omega^\epsilon} \begin{Bmatrix} yN_{a,x} \\ 0 \end{Bmatrix} \, d\Omega \quad (4.8.5) \\ &= \begin{Bmatrix} \frac{c \text{vol}(\Omega^\epsilon)}{48} (\xi_{,x})^2 \phi_a \sum_{b=1}^4 \phi_b d_{1b} \\ 0 \end{Bmatrix} \end{aligned}$$

Exercise 1. Fill in the details to obtain (4.8.5).

An analogous calculation for the bending stress defined by

$$\sigma_{11} = \sigma_{12} = 0, \quad \sigma_{22} = \sigma_0 x \quad (4.8.6)$$

leads to the following expression for internal force:

$$f_a^{\text{int}} = \int_{\Omega^\epsilon} \mathbf{B}_a^T \boldsymbol{\sigma} \, d\Omega = \begin{Bmatrix} 0 \\ \frac{c \text{vol}(\Omega^\epsilon)}{48} (\eta_{,y})^2 \phi_a \sum_{b=1}^4 \phi_b d_{2b} \end{Bmatrix} \quad (4.8.7)$$

The next step is to combine (4.8.5) and (4.8.7), and generalize to the case in which the rectangular element is not aligned with the global x - and y -axes. After some algebraic manipulations the following expressions are obtained:

$$f_a^{\text{int}} = \{f_{ia}^{\text{int}}\} \quad (4.8.8)$$

$$f_{ia}^{\text{int}} = s_{ij} \phi_a \sum_{b=1}^4 \phi_b d_{jb} \quad (\text{sum on } j = 1, 2) \quad (4.8.9)$$

$$s_{ij} = \frac{c \text{vol}(\Omega^\epsilon)}{48} (\xi_{,x_i} \xi_{,x_j} + \eta_{,x_i} \eta_{,x_j}) \quad (4.8.10)$$

where $x_1 = x$ and $x_2 = y$. Clearly, (4.8.9) reduces to the previous formulas under the appropriate restrictions. In general, (4.8.10) is evaluated at $\xi = \eta = 0$.

Exercise 2. Derive (4.8.9) and (4.8.10).

From (4.8.9) we may immediately deduce the formula for the 8×8 hourglass stiffness:

$$k^{\text{hourglass}} = [k_{ab}^{\text{hourglass}}], \quad 1 \leq a, b \leq 4 \quad (4.8.11)$$

$$k_{ab}^{\text{hourglass}} = \phi_a \phi_b s \quad (4.8.12)$$

where

$$s = [s_{ij}], \quad 1 \leq i, j \leq 2 \quad (4.8.13)$$

The final step in the derivation is to extend to the case of an arbitrary quadrilateral geometry. In order that the patch test be passed, the definition of the hourglass mode, (4.8.3), needs to be generalized. The requirement set forth by Kosloff is that the generalized hourglass mode be *orthogonal* to an arbitrary linear polynomial.¹⁶ This may be expressed in the following way in terms of a “trial” hourglass mode $\tilde{\phi}_a$:

$$\sum_{a=1}^4 \tilde{\phi}_a = 0 \quad (4.8.14)$$

$$\sum_{a=1}^4 \tilde{\phi}_a x_{1a} = 0 \quad (4.8.15)$$

$$\sum_{a=1}^4 \tilde{\phi}_a x_{2a} = 0 \quad (4.8.16)$$

Because the system (4.8.14) through (4.8.16) is underdetermined, we need to add an additional condition. For this purpose let us specify $\tilde{\phi}_4 = 1$. Then (4.8.14) through (4.8.16) may be solved for $\tilde{\phi}_1$, $\tilde{\phi}_2$, and $\tilde{\phi}_3$. In addition to the previously given orthogonality condition, ϕ_a needs to satisfy

$$\sum_{a=1}^4 \phi_a^2 = 4 \quad (4.8.17)$$

which was used in the derivation of (4.8.4). Thus the desired ϕ_a may be obtained by normalizing $\tilde{\phi}_a$:

$$\phi_a = 2 \tilde{\phi}_a / \left(\sum_{a=1}^4 \tilde{\phi}_a^2 \right)^{1/2} \quad (4.8.18)$$

Equation (4.8.18) is then used in (4.8.9) and (4.8.12) to define the hourglass mode.

Remarks

1. The generalization to three dimensions is described in Kosloff and Frazier [72]. Several examples which illustrate the effectiveness of the hourglass stiffness elements are presented in [72].

¹⁶This guarantees that the generalized hourglass mode will not be aroused by rigid body modes and constant strain states, i.e., the patch test will be satisfied.

2. Because of the inherent economy of the hourglass stiffness approach, considerable attention has been drawn to it. Among the recent developments given by Flanagan and Belytschko [73] and Belytschko et al. [74] are techniques for developing ϕ_a for two- and three-dimensional problems without solving any equations; this fundamental idea is presented in the following exercise.

Exercise 3. (T. Belytschko)

a. Let

$$b_{1a} = \frac{\partial N_a}{\partial x_1}$$

$$b_{2a} = \frac{\partial N_a}{\partial x_2}$$

Show that

$$\sum_{a=1}^4 b_{ia} x_{ja} = \delta_{ij}$$

Hint: Start with $x_{i,j} = \delta_{ij}$ and use the isoparametric expansion for x_i .

b. Let

$$s_a = 1, \quad h_a = (-1)^a$$

Consider the following expression for $\tilde{\phi}_a$:

$$\tilde{\phi}_a = a_1 b_{1a} + a_2 b_{2a} + a_3 s_a + h_a$$

where the a_i 's are arbitrary constants. Using equations (4.8.14) through (4.8.16) and part (a), show that

$$\tilde{\phi}_a = h_a - \sum_{b=1}^4 [(h_b x_{1b}) b_{1a} + (h_b x_{2b}) b_{2a}]$$

Note that this provides a direct solution for the "trial" hourglass mode; see [74] for application to three-dimensional problems.

Notes: In the papers of Belytschko and his colleagues, the terminology **hourglass mode** is reserved for $h_a = (-1)^a$ regardless of the shape of the element, and $\tilde{\phi}_a$ is referred to as the **hourglass stabilization (or suppression) operator**. The notation γ_a is used in place of $\tilde{\phi}_a$ by Belytschko.

ADDITIONAL EXERCISES AND PROJECTS

Exercise 1. Consider the two-dimensional plane strain beam, Example 1 of Sec. 4.4. The exact solution of this problem is easily derived and is given as follows:

$$\sigma_{11} = -\frac{P \tilde{x} y}{I}$$

$$\sigma_{22} = 0$$

$$\sigma_{12} = \frac{P}{2I}(c^2 - y^2)$$

$$\frac{6E^1 I}{P} u_1(x, y) = -y\{3(L^2 - \tilde{x}^2) + (2 + \nu^1)(y^2 - c^2)\}$$

$$\frac{6E^1 I}{P} u_2(x, y) = (\tilde{x}^3 - L^3) - \{(4 + 5\nu^1)c^2 + 3L^2\}(\tilde{x} - L) + 3\nu^1 \tilde{x} y^2$$

where $\tilde{x} = L - x$ and

| | E^1 | ν^1 |
|--------------|-----------------------|-----------------------|
| Plane stress | E | ν |
| Plane strain | $\frac{E}{1 - \nu^2}$ | $\frac{\nu}{1 - \nu}$ |

Obtain approximate solutions of the above elasticity problem by using the program DLEARN.

- Employ the following data in your calculations:

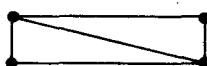
$$P = -1, \quad L = 16, \quad c = 2, \quad E = 1, \quad \nu = 0.3 \text{ and } 0.499$$

(The latter value of Poisson's ratio corresponds to the "nearly incompressible case.")

- The mesh to be used is depicted in Fig. 4.4.2. You will need to compute nodal forces corresponding to the traction boundary conditions along $x = 0$ and $x = L$.
- Assume plane strain conditions.
- Use four-node quadrilaterals and employ each of the following quadrature rules:
 - 2×2
 - \bar{B} with 2×2
 - one-point

- Use three-node triangles in the

- bisection pattern:



- cross-diagonal pattern:



(There are 10 cases in all.)

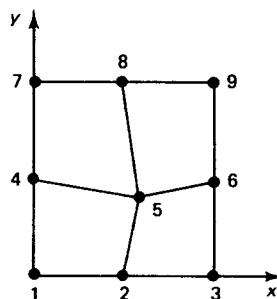
- In each case, compare the computed vertical tip displacement (i.e., $u_2(16, 0)$), the horizontal displacement of the root section (i.e., $u_1(0, y)$, $y \in [-c, c]$) and the bending and shear stress (i.e., σ_{11} and σ_{12} , respectively) for $x = 7, y \in [-c, c]$, with the exact solution.
- Write a brief report summarizing your findings and include an evaluation of the performance of each element.

Instructions for Using DLEARN

Consult the DLEARN User's Manual in Chapter 11 regarding set up. The data file for this problem, case (a), is presented in Table 11.5.2. Make sure you completely understand it. In particular, verify the consistent nodal forces in the data file by performing the calculations by hand. Your first run should be a data check run (i.e., IEXEC.EQ.0). Carefully check the output to ensure that you have input your data correctly. Keep using the data check mode until you are satisfied that your input data is correct. Then set IEXEC.EQ.1, which will cause the program to execute.

Run case (a) first and check the vertical tip displacement normalized by the exact value against the results presented in Table 4.4.1. If the comparison is satisfactory, proceed to the other cases. Otherwise, back to the drawing board—you've done something wrong!

Exercise 2. Consider the patch of four-node quadrilateral elements shown.



The nodal coordinates are:

| a | x_a | y_a |
|-----|-------|-------|
| 1 | 0 | 0 |
| 2 | 1 | 0 |
| 3 | 2 | 0 |
| 4 | 0 | 1 |
| 5 | 1.1 | 0.8 |
| 6 | 2 | 1 |
| 7 | 0 | 2 |
| 8 | 1 | 2 |
| 9 | 2 | 2 |

Use DLEARN to execute the following displacement boundary-value problems ($1 \leq a \leq 4, 6 \leq a \leq 9$):

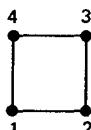
| Test number | d_{1a} | d_{2a} |
|-------------|----------|----------|
| 1 | 1 | 0 |
| 2 | 0 | 1 |
| 3 | x_a | 0 |
| 4 | 0 | x_a |
| 5 | y_a | 0 |
| 6 | 0 | y_a |

Assume the plane strain option and use the following data: $E = 1.0$, $\nu = 0.3$. Test the following options:

- a. 2×2 quadrature
- b. \bar{B} with 2×2 quadrature
- c. one-point quadrature

Based upon the results you obtain, determine whether or not each element passes the patch test. (Consider results "exact" if they are correct to one-half or more of the number of significant digits employed. Due to rounding errors, we cannot expect truly exact results on digital computers.)

Exercise 3. The rank check option in DLEARN can be used to determine the number of zero eigenvalues of the global stiffness matrix K . That is, if IRANK.EQ.2, the diagonal elements of the diagonal matrix D in the factorization $K = U^T D U$, where U is an upper triangular matrix, are printed. The number of zeros in D equals the number of zero eigenvalues of K , which corresponds to the number of zero-energy modes of the structure in question. Normally, there are, at most, three zero-energy modes—the rigid body modes. Consider the single element shown:



| a | x_a | y_a |
|-----|-------|-------|
| 1 | 0 | 0 |
| 2 | 1 | 0 |
| 3 | 1 | 1 |
| 4 | 0 | 1 |

Assume the plane strain option and use the following data: $E = 1.0$ and $\nu = 0.3$. Note there are no kinematic boundary conditions. Perform rank check analyses for the following options:

- 2×2 quadrature
- \bar{B} with 2×2 quadrature
- One-point quadrature

Comment on your results.

Exercise 4. It has been proposed to use C^0 triangular elements with complete quintic (i.e., fifth-order) polynomials for problems of two-dimensional elasticity. Determine the order of accuracy (i.e., a and b) in the following:

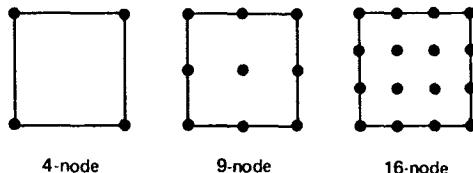
$$\begin{aligned}\|e\|_0 &\leq ch^a \|u\|_{k+1} \\ \|e\|_1 &\leq ch^b \|u\|_{k+1}\end{aligned}$$

Justify your answers.

Exercise 5. A series of convergence rate studies on linear elastic boundary-value problems was performed. All results indicate that $e = u^h - u$ behaves as follows:

$$\|e\|_1 = O(h^2)$$

which of the following elements was used in the studies?



Justify your answer.

Exercise 6. Consider the following problem:

$$K \begin{bmatrix} 1 & \epsilon \\ \epsilon & 1 \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}$$

subject to the constraint

$$d_1 + d_2 = 0$$

Assume $K = 1$ and $\epsilon = 0$.

- Formulate the matrix problem by way of the Lagrange-multiplier method. Determine d_1 , d_2 , and the Lagrange multiplier.
- Formulate the matrix problem by way of the penalty function approximation. Determine the approximations of d_1 , d_2 , and the Lagrange multiplier as functions of the penalty parameter.

Exercise 7. Consider the linear system $Kd = F$. It is desired to modify the system so that the following constraint is maintained:

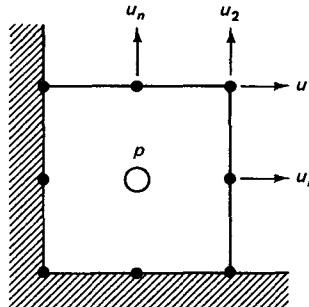
$$\alpha d_P + \beta d_Q = \gamma$$

where α , β , and γ are given constants, and $1 \leq P < Q \leq n_{eq}$. Develop modified equation systems by

- The Lagrange multiplier method
- The penalty method

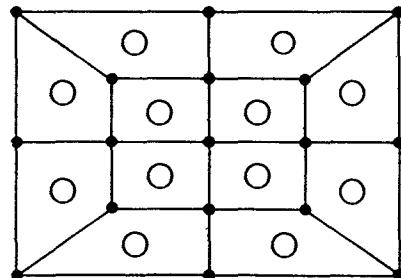
Exercise 8. Determine the constraint ratios for the following elements.

- The “enriched” bilinear displacements–constant pressure quadrilateral shown, in which normal-displacement degrees of freedom are added to each edge:



- The “enriched” trilinear displacements–constant pressure brick in which normal-displacement degrees of freedom have been added to each face. This is the three-dimensional analog of the element in part (a).

Exercise 9. Calculate the constraint ratio for the accompanying macroelement. (This macroelement was developed by Patrick Le Tallec and was shown to converge.)



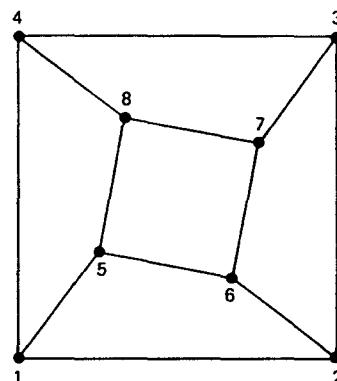
Exercise 10. To determine if a new finite element for two-dimensional heat conduction is convergent, a patch test is performed. The results are as follows:

| a | x_a | y_a | Problem 1 | Problem 2 | Problem 3 | d_a |
|-----|-------|-------|-----------|-----------|-----------|-----------------------------|
| 1 | 1.0 | 0.0 | 0.0 | 1.0 | 1 | Interior node \rightarrow |
| 2 | 2.0 | 0.0 | 0.0 | 2.0 | 1 | |
| 3 | 3.0 | 0.0 | 0.0 | 3.0 | 1 | |
| 4 | 1.0 | 1.0 | 1.0 | 1.0 | 1 | |
| 5 | 2.1 | 1.2 | 1.2 | 2.0 | 1 | |
| 6 | 3.0 | 1.0 | 1.0 | 3.0 | 1 | |
| 7 | 1.0 | 3.0 | 3.0 | 1.0 | 1 | |
| 8 | 2.0 | 3.0 | 3.0 | 2.0 | 1 | |
| 9 | 3.0 | 3.0 | 3.0 | 3.0 | 1 | |

Nodes 1 through 4 and 6 through 9 are boundary nodes.

Does the element pass the patch test? Justify your answer.

Exercise 11. Consider the following patch of two-dimensional elasticity elements. Define six prescribed-displacement, boundary-value problems by setting the boundary nodes (i.e., 1 to 4) to values in accord with linear monomials. Determine the solution at nodes 5 to 8 that must be attained in order that the patch test is passed. The coordinates of the nodes are:

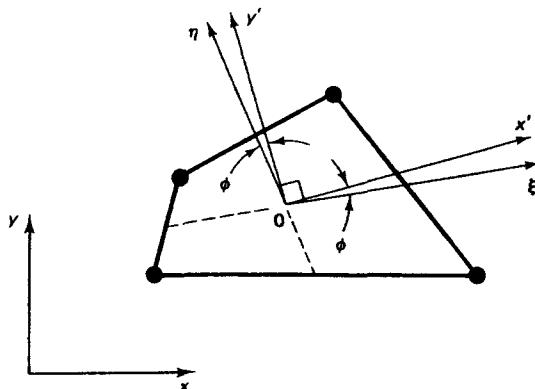


| a | x_a | y_a |
|-----|-------|-------|
| 1 | 0 | 0 |
| 2 | 3 | 0 |
| 3 | 3 | 3 |
| 4 | 0 | 3 |
| 5 | 0.75 | 1 |
| 6 | 2 | 0.75 |
| 7 | 2.25 | 2 |
| 8 | 1 | 2.25 |

Projects

Various efforts have been made to design four-node quadrilateral elements with improved bending behavior. The Wilson, Taylor, and Kosloff-Frazier elements, discussed in the previous two sections, are prime examples. One of the first attempts employed selective integration [75]. The element was successful in the rectangular configuration if the rectangle was aligned with the global coordinate system but behaved poorly otherwise. Kavanagh and Key [76] traced the erratic behavior to a lack of "invariance" of the formulation. This defect can be easily corrected and is described next. The corrected element is now used in several structural analysis computer programs. A local rectangular Cartesian coordinate system (i.e., x' , y') may be uniquely defined by constructing it so that its axes make equal angles with the natural ξ , η -system. Coordinates are rotated into the x' , y' -system and the element stiffness and force vector are calculated with respect to x' , y' . Before assembly, the stiffness and force are rotated into global coordinates. The improved bending behavior is facilitated by the use of a one-point quadrature on the shear term in the x' , y' -system, namely,

$$\int_{\Omega'} 4\mu w_{(1', 2')} u_{(1', 2')} d\Omega$$



where the 1', 2'-subscripts indicate differentiation with respect to x' , y' -coordinates, respectively. The remaining stiffness terms may be integrated with the normal 2×2 rule. These terms may be written as

$$\int_{\Omega'} \left(\lambda w_{(i',j')} u_{(i',j')} + 2\mu (w_{(1',1)} u_{(1',1')} + w_{(2',2)} u_{(2',2')}) \right) d\Omega$$

The reason for using the x' , y' -system should be apparent: The shear term is not invariant with respect to change of reference frame. Thus use of a local preferred system, independent of the global x , y -system, is necessitated.

Project 1. Develop a finite element subroutine in DLEARN for the element just described.

Check out your element by performing a rank check (correct rank of 5 should be attained), patch tests, and several sample problems. For example, perform a comparison of the element's behavior with other DLEARN elements on the beam bending problem considered previously. Make sure that you have correctly programmed the transformation between the x , y - and x' , y' -systems. Print out the coordinates of the element in both systems for an irregular element configuration and check by hand. Perform a pure-bending single-element test (see Fig. 4.7.1) with a rectangular element aligned with the x , y -system and with one skew to it. The same results should be attained if transformed to a common frame. These tests will check the invariance of the formulation.

Project 2. A criticism that may be lodged at the previous element is its obvious inability to handle incompressible phenomena. This is due to the 2×2 Gauss quadrature treatment of the λ -term. A modified version designed to alleviate the incompressible locking problem would involve one-point quadrature on

$$\int_{\Omega'} (\lambda w_{(i,j)} u_{(i,j)} + 4\mu w_{(1',2)} u_{(1',2')}) d\Omega$$

and 2×2 quadrature on the remainder, namely,

$$\int_{\Omega'} 2\mu (w_{(1',1)} u_{(1',1')} + w_{(2',2)} u_{(2',2')}) d\Omega$$

Program this element and perform the check-out tests described under Project 1. In addition, assess the behavior in nearly incompressible situations by performing calculations in which the ratio λ/μ is varied. Use the plane strain two-dimensional beam problem as an example and compare your results with those presented in Fig. 4.7.3.

Project 3. The previous two projects involve elements that are applicable only to the isotropic case. For anisotropic applications, we may use the \bar{B} -formulation described in Sec. 4.5. Set up the \bar{B} matrices corresponding to the selective integration schemes of Projects 1 and 2. In place of one-point quadrature treatment, use the element mean value in the definition of \bar{B} . This will cause a difference only in the axisymmetric configuration. (The difference is beneficial; see discussion in Sec. 4.5.1.) Program the various \bar{B} -elements and evaluate as described under Projects 1 and 2.

Appendix 4.

Mathematical Preliminaries

In posing the variational form of the boundary-value problem, we found it useful to define collections of functions that satisfied certain conditions. This leads us to the study of *linear spaces*. We begin with a description of the important properties of these spaces essential for finite element error analysis.

4.1.1 BASIC PROPERTIES OF LINEAR SPACES

We do not wish to discuss such dry and uninteresting material as the “axioms of a linear space.” Basically, a *real* (as in real numbers) *linear space* is a collection of objects for which the operations of addition and scalar multiplication are defined and behave as follows: If x and y are members of the linear space and α and β are scalars, then $\alpha x + \beta y$ is also a member of the linear space. Because of this property we sometimes say a linear space is *closed* under addition and scalar multiplication.

Example 1 (\mathbb{R}^n Is a Linear Space)

\mathbb{R}^n consists of n -tuples (i.e., *points*) $x = (x_1, x_2, \dots, x_n)$, where each $x_i \in \mathbb{R}$, $1 \leq i \leq n$. Addition means component by component addition:

$$\begin{aligned}x + y &= (x_1, x_2, \dots, x_n) + (y_1, y_2, \dots, y_n) \\&= (x_1 + y_1, x_2 + y_2, \dots, x_n + y_n)\end{aligned}$$

Scalar multiplication is defined in the usual way: Let $\alpha \in \mathbb{R}$; then

$$\alpha x = (\alpha x_1, \alpha x_2, \dots, \alpha x_n)$$

To show \mathbb{R}^n is a linear space we must show $\alpha x + \beta y$ is in \mathbb{R}^n , where $x, y \in \mathbb{R}^n$, and $\alpha, \beta \in \mathbb{R}$.

$$\begin{aligned}\alpha x + \beta y &= (\alpha x_1, \alpha x_2, \dots, \alpha x_n) + (\beta y_1, \beta y_2, \dots, \beta y_n) \\ &= (\alpha x_1 + \beta y_1, \alpha x_2 + \beta y_2, \dots, \alpha x_n + \beta y_n)\end{aligned}$$

The last result is an n -tuple of real numbers. Hence

$$\alpha x + \beta y \in \mathbb{R}^n$$

Linear spaces often come with important additional structure; namely, *inner products* and *norms*.

Definition. An *inner product* $\langle \cdot, \cdot \rangle$ on a real linear space A is a map $\langle \cdot, \cdot \rangle: A \times A \rightarrow \mathbb{R}$ (i.e., $\langle \cdot, \cdot \rangle$) assigns to an ordered pair $x, y \in A$ a real number denoted $\langle x, y \rangle$ with the following properties.

Let $x, y, z \in A$ and $\alpha \in \mathbb{R}$; then

- i. $\langle x, y \rangle = \langle y, x \rangle$ (symmetry)
 - ii. $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle$
 - iii. $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$
 - iv. $\langle x, x \rangle \geq 0$, and $\langle x, x \rangle = 0$ if and only if $x = 0$. (positive-definiteness)
-

Example 2

Without getting technical about the spaces involved, it can be easily shown that, under suitable hypotheses, all the $a(\cdot, \cdot)$, (\cdot, \cdot) , and $(\cdot, \cdot)_r$ encountered in previous chapters satisfy (i)–(iv) and therefore are inner products. Recall that the positive definiteness of $a(\cdot, \cdot)$ follows from positivity requirements on the constitutive tensor of the field theory in question and the homogeneous, essential (i.e., “kinematic”) boundary conditions of the collection of functions under consideration. The latter condition is seen to be necessary. For example, suppose $v, w: [0, 1] \rightarrow \mathbb{R}$ and $a(w, v) = \int_0^1 w_{,x} v_{,x} dx$. Then $a(\cdot, \cdot)$ is positive-definite on the collection $H_0^1 = \{w \mid w \in H^1, w(1) = 0\}$ but not on the collection H^1 . The function $u(x) = \text{constant} \neq 0$ is in H^1 and $a(u, u) = 0$.

Definition. Let $\{A, \langle \cdot, \cdot \rangle\}$ be an inner product space (i.e., a linear space A with an inner product $\langle \cdot, \cdot \rangle$ defined on A). Then $x, y \in A$ are said to be *orthogonal* or *perpendicular* (with respect to $\langle \cdot, \cdot \rangle$) if $\langle x, y \rangle = 0$.

Schwarz inequality. A key property of inner products is the *Schwarz inequality*, which states that for all x and y

$$\langle x, y \rangle^2 \leq \langle x, x \rangle \langle y, y \rangle$$

The proof of this fact follows directly from the definition of an inner product, and is given as follows.

Proof. Let $\alpha \in \mathbb{R}$. Then by definition

$$\begin{aligned} 0 &\leq \langle x + \alpha y, x + \alpha y \rangle \\ &= \langle x, x \rangle + 2\alpha \langle x, y \rangle + \alpha^2 \langle y, y \rangle \end{aligned}$$

The last expression is a nonnegative quadratic in α . Hence, its discriminant is non-positive, i.e.,

$$\langle x, y \rangle^2 - \langle x, x \rangle \langle y, y \rangle \leq 0 \quad \blacksquare$$

Definition. A **norm** $\|\cdot\|$ on a linear space A is a map $\|\cdot\|: A \rightarrow \mathbb{R}$, with the following properties.

Let $x, y \in A$ and $\alpha \in \mathbb{R}$; then

- i. $\|x\| \geq 0$, and $\|x\| = 0$ if and only if $x = 0$ } (positive-definiteness)
- ii. $\|\alpha x\| = |\alpha| \|x\|$
- iii. $\|x + y\| \leq \|x\| + \|y\|$ (triangle inequality)

Note that an inner product space $\{A, \langle \cdot, \cdot \rangle\}$ possesses a **natural norm** defined by

$$\|x\| = \langle x, x \rangle^{1/2}$$

Exercise 1. Verify that the definition of a norm is satisfied by the above expression. *Hint:* Use the Schwarz inequality to prove the triangle inequality holds.

Example 3

\mathbb{R}^n equipped with the dot product, defined by $\langle x, y \rangle = x \cdot y$, is an inner product space. The natural norm $\|x\| = (x \cdot x)^{1/2}$ is the length of the vector with tail at the origin and tip at x .

Example 4

$L_2(0, 1)$, equipped with the inner product $\langle u, v \rangle = \int_0^1 uv \, dx$, is an inner product space. The natural norm, denoted by $\|\cdot\|_0$, is defined by $\|u\|_0 = (\int_0^1 u^2 \, dx)^{1/2}$.

Example 5

$H^1(0, 1)$, equipped with the inner product $\langle u, v \rangle = \int_0^1 (uv + u_{,x}v_{,x}) \, dx$, is an inner product space. The natural norm, denoted by $\|\cdot\|_1$, is defined by

$$\|u\|_1 = \left(\int_0^1 (u^2 + u_{,x}^2) \, dx \right)^{1/2}$$

Remarks

1. Each of the spaces in the preceding examples satisfies a technical condition called **completeness**. Roughly speaking, completeness means that the limit of a convergent sequence of elements in the space is also in the space. An inner product

space which is complete, in its natural norm, is called a *Hilbert space*. Hilbert spaces have a rich mathematical theory and are particularly important to the study of linear partial differential equations. Examples 3 through 5 are Hilbert spaces. For the purposes we have in mind, we shall not need to delve further into this subject.

2. In terms of a natural norm, the Schwarz inequality takes the form

$$|\langle x, y \rangle| \leq \|x\| \|y\|$$

This is the form frequently used in applications.

Definition. A *seminorm* $|\cdot|$ on a linear space A is a map $|\cdot| : A \rightarrow \mathbb{R}$ with the following properties.

Let $x, y \in A$ and $\alpha \in \mathbb{R}$; then

- i. $|x| \geq 0$ (positive-semidefiniteness)
- ii. $|\alpha x| = |\alpha| |x|$
- iii. $|x + y| \leq |x| + |y|$ (triangle inequality)

Remark

The only difference between a norm and seminorm is that a norm is positive-definite, whereas a seminorm is positive-semidefinite.

Example 6

$|u| = (\int_0^1 u_{,x} u_{,x} dx)^{1/2}$ defines a seminorm on $H^1([0, 1])$ but not a norm.

4.1.2 SOBOLEV NORMS

Consider a domain $\Omega \subset \mathbb{R}^n$, $n \geq 1$, and let $u, v : \Omega \rightarrow \mathbb{R}$.

The $L_2(\Omega)$ *inner product and norm* are defined by

$$(u, v) = \int_{\Omega} uv \, d\Omega$$

and

$$\|u\|_0 = (u, u)^{1/2}$$

respectively.

The $H^1(\Omega)$ *inner product and norm* are defined by

$$(u, v)_1 = \int_{\Omega} (uv + u_{,i}v_{,i}) \, d\Omega \quad (\text{sum, } 1 \leq i \leq n)$$

and

$$\|u\|_1 = (u, u)_1^{1/2}$$

respectively.

Analogously, we may define the $H^s(\Omega)$ *inner product and norm* by

$$(u, v)_s = \int_{\Omega} (uv + u_{,i}v_{,i} + u_{,ij}v_{,ij} + \cdots + u_{,\underbrace{\substack{j \dots k}}_{s \text{ indices}}}v_{,\underbrace{\substack{j \dots k}}_{s \text{ indices}}}) d\Omega$$

(sum, $1 \leq i, j, \dots, k \leq n$)

and

$$\|u\|_s = (u, u)^{1/2}$$

respectively.

The case in which $u, v : \Omega \rightarrow \mathbb{R}^m$ (i.e., are vector-valued) is essentially the same.

Let $H^s(\Omega, \mathbb{R}^m)$ be the space of \mathbb{R}^m -valued functions $u = (u_1, u_2, \dots, u_m)$, for which each $u_i \in H^s(\Omega)$, $1 \leq i \leq m$.

The $H^s(\Omega, \mathbb{R}^m)$ *inner product and norm* are defined by

$$(u, v)_s = \int_{\Omega} (u_i v_i + u_{i,j} v_{i,j} + u_{i,jk} v_{i,jk} + \cdots + u_{i,\underbrace{\substack{j \dots l}}_{s \text{ indices}}} v_{i,\underbrace{\substack{j \dots l}}_{s \text{ indices}}}) d\Omega$$

(sum, $1 \leq i \leq m, 1 \leq j, k, \dots, l \leq n$)

and

$$\|u\|_s = (u, u)^{1/2}$$

respectively.

We will also find it useful to define the $H^r(\Omega, \mathbb{R}^m)$ *seminorm*:

$$|u|_r = \left(\int_{\Omega} u_{i,\underbrace{\substack{j \dots l}}_{r \text{ indices}}} u_{i,\underbrace{\substack{j \dots l}}_{r \text{ indices}}} d\Omega \right)^{1/2} \quad (\text{sum, } 1 \leq i \leq m, 1 \leq j, k, \dots, l \leq n)$$

Remarks

1. It simplifies the writing considerably if we agree to drop the arguments in $H^s(\Omega, \mathbb{R}^m)$, the particular Ω and \mathbb{R}^m being understood by the context.
2. Note that $H^0 = L_2$.

Definition. The *sth Sobolev space*, denoted by H^s , is the collection of functions $u : \Omega \rightarrow \mathbb{R}^m$ with finite H^s norm.

Remarks

1. Sobolev spaces are, in particular, Hilbert spaces.
2. The inclusion property

$$H' \subset H^s, \quad r \geq s$$

follows directly from the definitions.

Sobolev Imbedding Theorems

The most-celebrated theorems concerning Sobolev spaces go under the name of **Sobolev imbedding theorems**. A particularly useful result of this kind tells us when functions in H^s are smooth in the classical sense.

Definition. Let C_b^k , $k \geq 0$, be the space of functions $u : \Omega \rightarrow \mathbb{R}^m$ which are

- i. Bounded
- ii. Continuous and have continuous and bounded derivatives of order j , $1 \leq j \leq k$.

Theorem. If Ω is an open set in \mathbb{R}^n (not a hypersurface!) and $s > n/2 + k$,

$$H^s \subset C_b^k \quad \blacksquare$$

Example 7

Suppose $\Omega =]0, 1[\subset \mathbb{R}$. Then functions in H^1 are continuous and bounded (i.e., are C_b^0 functions). Functions in H^2 are C_b^1 functions, etc.

Example 8

Suppose Ω is an open set in \mathbb{R}^2 . Then $H^2 \subset C_b^0$.

Remarks

1. The imbedding result is “sharp” in the sense that if $s \leq n/2 + k$, then there exists a function in H^s which is *not* in C_b^k .
2. Note that the range space \mathbb{R}^m plays *no* role in the imbedding theorem.

Exercise 2. Let $\Omega = \{x, y \in \mathbb{R}^2 \mid r = (x^2 + y^2)^{1/2} \leq \frac{1}{2}\}$. Show that $\log(\log r^{-1})$ is in $H^1(\Omega)$ but not in $C_b^0(\Omega)$.

4.1.3 APPROXIMATION PROPERTIES OF FINITE ELEMENT SPACES IN SOBOLEV NORMS

The fundamental error estimates for finite element approximations to solutions of linear elliptic boundary-value problems are stated in terms of Sobolev norms of the error. The intent of this section is to answer the following question of approximation.

Given a Sobolev space H' and a finite element space \mathcal{S}^h , how well can we approximate a given $u \in H'$ if we are allowed to pick any member of \mathcal{S}^h ? The answer to this question is given in the work of Ciarlet and Raviart [77, 78] (and others; see references therein). The book by Oden and Reddy [79] has a nice account of these “approximability” theorems. Rather than get into the details of this work, which can be rather technical, we shall just state the end results in the form of a set of hypotheses about the finite element space under consideration. We shall try then to communicate

the practical meaning of the hypotheses and point out which finite element spaces that we are apt to employ actually satisfy these hypotheses.

Consider a collection of finite element spaces $\{\mathcal{S}^h | h \in]0, D]\}$, $D = \text{diameter } \Omega\}$. Each \mathcal{S}^h is finite dimensional. If $h_1 < h_2$, we are to think of \mathcal{S}^{h_1} as “refinement” of \mathcal{S}^{h_2} , with the dimension of \mathcal{S}^{h_1} , being greater than that of \mathcal{S}^{h_2} .

Remark

\mathcal{S}^h need not be defined for every $h \in]0, D]$. However, for each $\epsilon \in]0, D]$ \mathcal{S}^h must be defined for an $h < \epsilon$. That is, we want to be able to take arbitrarily small values of h .

Definition

A collection of finite element spaces will be called k, m -regular (or simply *regular*) if for each fixed h

- $\mathcal{S}^h \subset H^m$.
- For every $u \in H^r$, $r \geq 0$, and for all s such that $0 \leq s \leq \min\{r, m\}$, there exist a $U^h \in \mathcal{S}^h$ and constant c , independent of u and h , such that

$$\|u - U^h\|_s \leq ch^\alpha \|u\|_r$$

where $\alpha = \min(k + 1 - s, r - s)$, the rate of convergence.

Remarks

1. The essence of (ii) is that the more refined the mesh, the better U^h is able to approximate u . It is also clear that higher derivatives are approximated less accurately than lower derivatives.

2. To establish that a collection of finite element spaces is k, m -regular requires a “uniformity” condition on the mesh refinements as $h \rightarrow 0$. For example, consider $\Omega \subset \mathbb{R}^n$ and $\bar{\Omega} = \cup_{e=1}^{n_{el}} \bar{\Omega}^e$.

Let

$$h^e = \text{diameter } \Omega^e$$

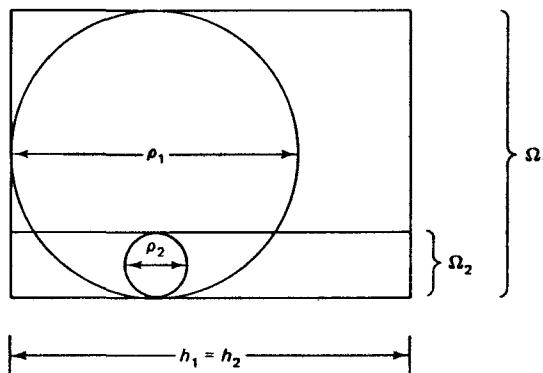
$$\rho^e = \text{diameter of largest sphere contained in } \Omega^e$$

$$h = \max_{1 \leq e \leq n_{el}} (h^e)$$

$$\rho = \min_{1 \leq e \leq n_{el}} (\rho^e)$$

$$\sigma = \frac{h}{\rho} \quad (\text{aspect ratio})$$

A sufficient uniformity condition to obtain all the above results is that $\sigma \leq \sigma_0$ for all \mathcal{S}^h . When this is the case, the collection of spaces is called *quasi-uniform*. As may be seen, quasi-uniformity prevents the aspect ratio from “blowing up” as the mesh is refined. For example, this precludes the mesh refinement from being one-dimensional, as illustrated in the accompanying figure.



Without a condition such as quasi-uniformity, the best result typically obtainable takes the form (assuming $u \in H^{k+1}$)

$$\|u - U^h\|_s \leq C \frac{h^{k+1}}{\rho^s} \|u\|_{k+1}$$

The deleterious effect of “slivers” is apparent from the above relation. Quasi-uniformity allows us to replace the ratio h^{k+1}/ρ^s by h^{k+1-s} (with appropriate adjustment to the constant) and hence the definition of k, m -regular collections.

In the following examples, we give without proof some examples of k, m -regular collections of finite element spaces. (For further reading, see the papers of Ciarlet and Raviart [77, 78], and the book by Oden and Reddy [79].)

Example 9

Let \mathcal{S}^h denote the piecewise linear finite element space on $]0, 1[$ consisting of N elements. The collection $\{\mathcal{S}^h | h = 1/N, N \text{ a positive integer}\}$ is $1, 1$ -regular. That is, for each $h = 1/N$

- i. $\mathcal{S}^h \subset H^1$.
- ii. For every $u \in H^2$ there exists a $U^h \in \mathcal{S}^h$ such that

$$\|u - U^h\|_1 \leq ch|u|_2$$

$$\|u - U^h\|_0 \leq ch^2|u|_2$$

where c is a constant independent of u and h . (If the given u is in H^1 , but not in H^2 , then instead of the above, we have only that $\|u - U^h\|_0 \leq ch|u|_1$.)

Example 10

The collection of piecewise quadratic finite element spaces on $]0, 1[$ defined in identical fashion to Example 9 is $2, 1$ regular. Assuming $u \in H^3$, condition (ii) for this case becomes

$$\|u - U^h\|_1 \leq ch^2 \|u\|_3$$

$$\|u - U^h\|_0 \leq ch^3 \|u\|_3$$

If each "middle node" is located exactly at the midpoint of the element, then the norms on the right-hand side of the above may be replaced by seminorms.

Example 11

The collection of Hermite finite element spaces on $]0, 1[$, defined as in Example 9, is 3, 2-regular. Assuming $u \in H^4$, condition (ii) becomes

$$\|u - U^h\|_2 \leq ch^2 |u|_4$$

$$\|u - U^h\|_1 \leq ch^3 |u|_4$$

$$\|u - U^h\|_0 \leq ch^4 |u|_4$$

Example 12 (Isoparametric Elements)

- a. Consider a collection of finite element spaces consisting of isoparametric elements on a domain $\Omega \subset \mathbb{R}^n$, $n \geq 1$. We assume there is no geometric error in modeling the boundary of the domain, i.e., $\bar{\Omega} = \cup_{e=1}^{n_e} \bar{\Omega}^e$. Such a collection is *at least* 1, 1-regular. In particular

$$\|u - U^h\|_1 \leq ch |u|_2$$

$$\|u - U^h\|_0 \leq ch^2 |u|_2$$

for every $u \in H^2$.

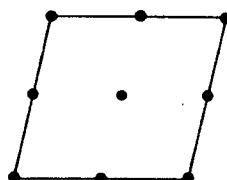
- b. Assume the same conditions as in part (a). In addition, assume each isoparametric element mesh is such that its isoparametric transformation $x = x(\xi)$ is a linear polynomial in ξ and that the element shape functions are capable of exactly representing a complete k th-degree polynomial in ξ . Then the collection is k , 1-regular. That is,

$$\|u - U^h\|_1 \leq ch^k |u|_{k+1}$$

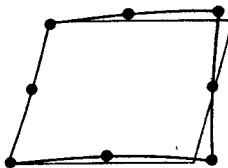
$$\|u - U^h\|_0 \leq ch^{k+1} |u|_{k+1}$$

for every $u \in H^{k+1}$.

For example, a collection of spaces consisting of the nine-node quadrilaterals in rectangular, or parallelogram (see the accompanying figure), configurations is 2, 1-regular.



- c. Assume the same conditions as in part (a). Consider the case in which the isoparametric functions are not necessarily linear polynomials in ξ but for which the deviation from linearity is "small."



For example, the accompanying curved quadrilateral element illustrates what we mean by a small deviation. As in the previous case, if the element shape functions are capable of exactly representing a complete k th degree polynomial in ξ then the family of spaces in question is k , 1-regular, but here

$$\|u - U^h\|_1 \leq ch^k \|u\|_{k+1}$$

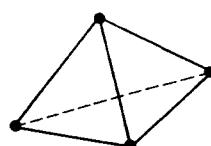
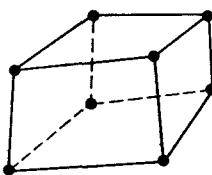
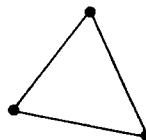
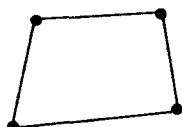
$$\|u - U^h\|_0 \leq ch^{k+1} \|u\|_{k+1}$$

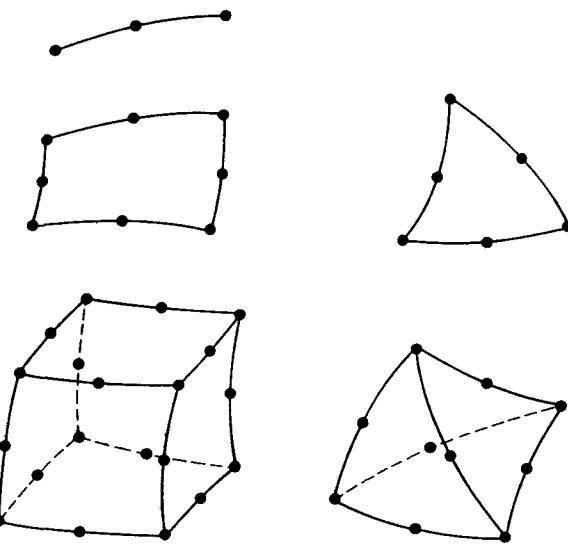
That is, in the curved case the seminorms on the right-hand side of the estimates are replaced by norms. Thus we have the optimal interpolation estimates for curved elements, if they are not too distorted. However, as the magnitude of distortion is increased, the approximation capability of the elements deteriorates (see [79], p. 283). What happens is the “constant” c becomes progressively larger as the element is more and more distorted.

Remark

As is clear from the previous discussion, the main condition to ascertain in order to determine the approximation capability of a finite element space is the highest-degree complete polynomial that the element is capable of exactly representing. Values of k for some of the standard C^0 -elements are given in the accompanying figures. As is clear, there are a multitude of k , 1-regular families of finite element spaces. Constructing k , m -regular families, $m \geq 2$, is possible but not as straightforward.

$k = 1$



$k = 2$ 

4.1.4 HYPOTHESES ON $a(\cdot, \cdot)$

All the problem classes considered thus far can be written in the form

$$a(w, u) = (w, f) + (w, h)_\Gamma$$

This equation is to hold for all $w \in \mathcal{V} \subset H_0^m$, for some $m \geq 1$, where the zero subscript on H_0^m indicates that certain homogeneous essential boundary conditions are built into the definition of H_0^m .

Definition. Two norms, $\|\cdot\|^{(1)}$ and $\|\cdot\|^{(2)}$, on a linear space A , are said to be *equivalent* if there exist constants $c_1, c_2 > 0$, such that for all $x \in A$,

$$c_1\|x\|^{(1)} \leq \|x\|^{(2)} \leq c_2\|x\|^{(1)}$$

Remarks

1. Our main hypothesis is that we assume $\|\cdot\|_m$ and $a(\cdot, \cdot)^{1/2}$ define equivalent norms on H_0^m , i.e., for all $w \in H_0^m$

$$c_1\|w\|_m \leq a(w, w)^{1/2} \leq c_2\|w\|_m$$

2. $a(\cdot, \cdot)$ is sometimes called the *energy inner product*, and the induced natural norm, as defined, is called the *energy norm*.

The following examples are stated without proof.

Example 13

Consider $a(w, v) = \int_0^1 w_{,x}v_{,x} dx$. Then $\|\cdot\|_1$ and $a(\cdot, \cdot)^{1/2}$ define equivalent norms on

$H_0^1(]0, 1[)$. (Recall that the subscript zero is taken to mean that members of this space are zero at $x = 1$.)

Example 14

Consider the generalized heat conduction problem. Let

$$a(w, v) = \int_{\Omega} w_{,i} \kappa_{ij} v_{,j} d\Omega$$

where $\Omega \subset \mathbb{R}^n$, $1 \leq i, j \leq n$, and repeated indices are to be summed. The conductivities $\kappa_{ij} = \kappa_{ji}$ are assumed positive definite in the sense that

$$\kappa_{ij}(x) y_i y_j \geq \delta y_i y_i$$

for a positive constant δ and all $y_i \in \mathbb{R}$, $1 \leq i \leq n$, and all $x \in \Omega$. In addition, we require

$$\kappa_{ij}(x) \leq M, \quad 1 \leq i, j \leq n$$

where M is some positive constant. Under these circumstances, $\|\cdot\|_1$ and $a(\cdot, \cdot)^{1/2}$ define equivalent norms on $H_0^1(\Omega)$.

Example 15

Consider classical elasticity theory. Let

$$a(w, v) = \int_{\Omega} w_{(i,j)} c_{ijkl} v_{(k,l)} dx$$

where $\Omega \subset \mathbb{R}^n$, $1 \leq i, j, k, l \leq n$, and repeated indices are to be summed. The hypotheses on the elastic coefficients read as follows: For all $x \in \Omega$ and $\psi_{ij} = \psi_{ji}$, there exist positive constants δ and M such that

$$c_{ijkl}(x) = c_{jikl}(x) = c_{ijlk}(x) = c_{klij}(x)$$

$$c_{ijkl}(x) \psi_{ij} \psi_{kl} \geq \delta \psi_{ij} \psi_{ij}$$

$$c_{ijkl}(x) \leq M$$

Under these circumstances $\|\cdot\|$ and $a(\cdot, \cdot)^{1/2}$ are equivalent norms on $H_0^1(\Omega, \mathbb{R}^n)$.

Example 16

Consider the case of Bernoulli-Euler beam theory. Assume the domain is simply $]0, 1[$. Define

$$a(w, v) = \int_0^1 w_{,xx} v_{,xx} dx$$

Then $\|\cdot\|_2$ and $a(\cdot, \cdot)^{1/2}$ define equivalent norms on $H_0^2(]0, 1[)$. The subscript zero may denote any physically reasonable homogeneous, essential boundary conditions, e.g.,

$$w(0) = w(1) = 0 \quad (\text{simply supported})$$

$$w(0) = w(1) = w_{,x}(0) = w_{,x}(1) = 0 \quad (\text{built in})$$

$$\left. \begin{array}{l} w(0) = w_{,x}(0) = 0 \\ \text{or} \quad w(1) = w_{,x}(1) = 0 \end{array} \right\} \quad (\text{cantilever})$$

etc.

We have asserted that all classes of boundary-value problems studied thus far, under suitable hypotheses on constitutive coefficients and boundary conditions, define energy norms equivalent to an appropriate Sobolev norm. To prove some of the above results is *not* so easy! The interested reader may profitably consult Mikhlin [80] and Fichera [81].

Appendix 4.II

Advanced Topics in the Theory of Mixed and Penalty Methods: Pressure Modes and Error Estimates

David S. Malkus

4.II.1 PRESSURE MODES, SPURIOUS AND OTHERWISE

There seems to be a perplexing variety of finite elements that are candidates for employment in problems with the incompressibility constraint. As has been shown earlier, each element seems to have drawbacks of some kind, and none seems to be ideally suited to such problems. Consider the case of the bilinear displacement–constant pressure isoparametric element: It has an “optimal” constraint ratio; it is a simple element, well suited to nonlinear problems because of relatively low (re-)assembly cost. However, until recently, no error estimates could be derived for this element in the incompressible case, even for the simplest meshes. Fortin [82] discovered the spurious pressure modes associated with this element and observed that state-of-the-art methods for obtaining error estimates could not be applied to the element. Even more disturbing was the development of simple, physically reasonable test problems in which the element gave obviously meaningless results [83]. Even though the bilinear displacement–constant pressure element was widely used in practice and seemed to be effective in many important physical applications, the seeds of doubt were sown.

Many investigators tended to shy away from using any element with spurious modes and opted for elements that were “safe” but had obvious drawbacks. A common feature of the “safe” elements is that many seem to have a limiting constraint ratio r , larger than two, whereas many of the elements with $r = 2$ exhibit spurious modes—at least on some types of meshes (usually very simple ones), with some types of boundary conditions. Many of the safe elements have continuous pressure fields as well, which appears to rule out employment in penalty methods. We will shortly investigate the reasons for safe elements being classified as such, but it might be useful to recall some safe elements:

1. Quadratic displacement triangle–constant pressure (Fig. 4.3.4(j))
2. Quadratic displacement triangle–continuous linear pressure (Fig. 4.3.6(d))
3. Biquadratic quadrilateral–constant pressure
4. Biquadratic quadrilateral–linear discontinuous pressure (Fig. 4.3.4(h))
5. Biquadratic quadrilateral–continuous bilinear pressure (Fig. 4.3.6(c))

This list is not exhaustive, but it is representative. Numbers 2, 4, and 5 have optimal error estimates but have drawbacks that have either been mentioned or will be explored. The reason that the standard error estimates for incompressible elements [84] steer the practitioner toward the underconstrained and inconvenient elements is that the standard estimates demand too much from a Lagrange multiplier (or related penalty pressure). The role of the Lagrange multiplier has been seen to be a twofold role of enforcer of the constraint and of pressure solution. The choice made in all five of the safe elements is to choose elements in which the role of enforcer has to some extent been sacrificed to avoid pressure modes. Before we can make a convincing argument along these lines, however, we must say something more definitive about the observations of Sani et al. [83]. At first sight, their results seem to say, simply, that reasonable problems can be concocted for which bilinear displacement–constant pressure elements will not work at all. To see that this is not really the case, we need to investigate the much maligned “pressure mode” in more detail. We will show conclusively that pressure modes are natural and even useful phenomena, more like sharks or buzzards than the finite element demons they have been made out to be. Our investigation will be largely confined to two-dimensional elements, because these are ones most easily visualized, and these are the ones for which current theory is most completely worked out. Much of what we say either carries over to three dimensions or can be easily generalized to such cases. The reader should proceed to three-dimensional problems with caution, however, as there *are* indeed important questions unresolved about, for example, integration accuracy in isoparametric three-dimensional reduced and selective formulations, and the possible generalization of macro-elements which work in two dimensions to three dimensions. In this investigation, we will try to get as much insight as possible from an “algebraic” approach to the problem at hand rather than a “functional analytic” approach. Inevitably, some functional analysis will creep in toward the end, but we hope not more than the reader has already been exposed to in this text.

A **pressure mode** is defined to be any pressure trial function, $q^h \not\equiv 0$, which satisfies

$$\int_{\Omega} q^h \operatorname{div} u^h d\Omega = 0 \quad \text{for all } u^h \in \mathcal{S}^h \quad (4.II.1)$$

Although this might seem to be a very restrictive condition on q^h , the existence of such nonzero q^h can be crucially dependent on boundary conditions: Let $q^h = \text{constant}$ in Ω , and apply the divergence theorem to (4.II.1):

$$\int_{\Gamma} \mathbf{u}^h \cdot \mathbf{n} d\Gamma = 0 \quad \text{for all } \mathbf{u}^h \in \mathcal{S}^h \quad (4.II.2)$$

Thus $q^h = \text{constant}$ in Ω satisfies (4.II.1) whenever the essential boundary conditions imposed on \mathcal{S}^h imply $\mathbf{u}^h \cdot \mathbf{n} = 0$ on Γ , that is, whenever volume is *globally* preserved by the boundary conditions. It should be noted that (4.II.1) and (4.II.2) are the same as (4.2.10). Whenever $\mathbf{u}^h \cdot \mathbf{n}$ is specified on all Γ , preserving volume leads to a pressure mode; failing to preserve volume leads to no solution. Many elements used in practice also allow other nonzero solutions to (4.II.1). Any such pressure mode, other than the constant, has been dubbed a *spurious mode*.

If we consider (4.3.21) with $M = \mathbf{0}$, we can see that a pressure mode satisfies

$$\begin{bmatrix} \bar{K} & G \\ G^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{0} \\ q \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \end{Bmatrix} \quad (4.II.3)$$

where q is the vector of nodal values of the pressure mode. To see why this is the case, consider the matrix form of (4.II.1):

$$0 = (q^h, \text{div } \mathbf{u}^h) = \mathbf{q}^T \mathbf{G}^T \mathbf{d} \quad \text{for all } \mathbf{d}$$

where \mathbf{d} is the vector of nodal values of a $\mathbf{u}^h \in \mathcal{S}^h$, and the product (\cdot, \cdot) denotes the L_2 inner product of (4.II.1) and will be used wherever possible hereafter. Thus $\mathbf{G}\mathbf{q} = \mathbf{0}$ as required. Note that by (4.II.3) a pressure mode is a pressure distribution that can arise as a nonzero response to zero applied loads. The constant pressure mode in the case when $\mathbf{u}^h \cdot \mathbf{n} = 0$ on Γ has a physical interpretation. Spurious modes do not, hence their name.

4.II.2 EXISTENCE AND UNIQUENESS OF SOLUTIONS IN THE PRESENCE OF MODES

As we indicated earlier, (4.II.3) shows that the matrix equation (4.3.21) is singular when the element has pressure modes. Before we try to obtain a solution, it behoves us to consider whether there *are* any solutions. Ominously, the answer is: *not always*. Much of the controversy about spurious modes has resulted from this fact, which can easily be overlooked—particularly when the related penalty method is employed in place of (4.3.21). There is a subtlety about (4.3.21) as it relates to what follows. Numerical or exact quadrature may have been used to evaluate the matrices involved. In particular, the former may make $Q = -M$ (slightly) different from the L_2 -inner product. In all cases, we assume that Q is positive-definite. We also use two additional notions of orthogonality: that implied by the usual vector dot product, and Q -orthogonality, when quadrature is consistently applied to compute M and the vectors involved in the dot product. We will be careful to say “ Q -orthogonal” when that is what is intended. The key algebraic fact is contained in the following lemma.

Lemma. Using the notation of (4.3.21), given any vector of nodal values, H , there is a vector of displacement nodal values, d , such that

$$H = G^T d$$

if and only if for any \mathbf{q} such that

$$\mathbf{G}\mathbf{q} = \mathbf{0} \quad (4.I)$$

(i.e., \mathbf{q} is a pressure mode),

$$\mathbf{H}^T \mathbf{q} = 0 \quad (4.I)$$

Proof. We seek a solution to

$$\mathbf{G}^T \mathbf{d} = \mathbf{H} \quad (4.I)$$

That is, we wish to know when \mathbf{H} is in the range of \mathbf{G}^T . But the range of \mathbf{G}^T is precisely the orthogonal complement of the solutions of (4.II.4), as we will argue: If \mathbf{H} is in the range of \mathbf{G}^T , then $\mathbf{H} = \mathbf{G}^T \mathbf{d}$ for some \mathbf{d} ; thus $\mathbf{q}^T \mathbf{H} = \mathbf{q}^T \mathbf{G}^T \mathbf{d} = 0$ for all solutions of (4.II.4). Conversely, let $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k$ denote the columns of \mathbf{G}^T and suppose (4.II.5) is satisfied for all solutions to (4.II.4). The statement that $\mathbf{q}^T \mathbf{G}^T = \mathbf{0}^T$ is equivalent to the statement that \mathbf{q} is in the orthogonal complement of $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k$, which clearly spans the range of \mathbf{G}^T (though the \mathbf{v}_i 's need not be independent). Thus (4.II.5) says that \mathbf{H} is in the orthogonal complement of the orthogonal complement of the range of \mathbf{G}^T , or the range of \mathbf{G}^T itself. This concludes the proof.

The lemma provides the following existence condition.

Theorem 1. The incompressible case of (4.3.21), i.e., the $\mathbf{M} = \mathbf{0}$ case, has one solution if and only if (4.II.5) is satisfied for all solutions to (4.II.4).

Proof. The second equation of (4.3.21) is just (4.II.6), which, in view of the lemma, is satisfied only if (4.II.5) is satisfied for all solutions to (4.II.4). Therefore we know that there is a \mathbf{d}_1 with $\mathbf{G}^T \mathbf{d}_1 = \mathbf{H}$. We will attempt to find a solution of the first equation of (4.3.21), in the form $\mathbf{d}_1 + \mathbf{d}_0$, where $\mathbf{G}^T \mathbf{d}_0 = \mathbf{0}$. This can be done by solving

$$\bar{\mathbf{K}}(\mathbf{d}_1 + \mathbf{d}_0) + \mathbf{G}\mathbf{p} = \bar{\mathbf{F}}$$

using the “discrete Poisson equation,” (4.3.26), which we rederive here so we can examine the intermediate steps:

$$\begin{aligned} \bar{\mathbf{K}}\mathbf{d}_0 + \mathbf{G}\mathbf{p} &= \bar{\mathbf{F}} - \bar{\mathbf{K}}\mathbf{d}_1 \\ \mathbf{d}_0 + \bar{\mathbf{K}}^{-1}\mathbf{G}\mathbf{p} &= \bar{\mathbf{K}}^{-1}\bar{\mathbf{F}} - \mathbf{d}_1 \end{aligned} \quad (4.II)$$

Obviously, more than one \mathbf{p} can satisfy (4.II.7), since \mathbf{d}_0 is as yet undetermined (e.g., $\mathbf{p} = \mathbf{0}$ will work). But we also have the requirement that $\mathbf{G}^T \mathbf{d}_0 = \mathbf{0}$; if we can find \mathbf{p} such that this is satisfied, then (4.II.7) defines an incompressible \mathbf{d}_0 such that $\mathbf{d}_1 + \mathbf{d}_0$ satisfies the requirements of the theorem. This implies

$$\mathbf{G}^T \bar{\mathbf{K}}^{-1} \mathbf{G}\mathbf{p} = \mathbf{G}^T \bar{\mathbf{K}}^{-1} \bar{\mathbf{F}} - \mathbf{H} \quad (4.II)$$

The argument of the lemma shows that this has a solution \mathbf{p} if and only if the right-hand side is orthogonal to the kernel of $\mathbf{G}^T \bar{\mathbf{K}}^{-1} \mathbf{G}$. (The form or dimension of \mathbf{G}^T was in no way taken into account in the proof of the lemma. Here $\mathbf{G}^T \bar{\mathbf{K}}^{-1} \mathbf{G}$ is symmetric, so it plays the role of both \mathbf{G} and \mathbf{G}^T in the lemma.) The kernel of $\mathbf{G}^T \bar{\mathbf{K}}^{-1} \mathbf{G}$ is precisely

the subspace of solutions to (4.II.4). (This can be seen by taking $\mathbf{L}\mathbf{L}^T = \bar{\mathbf{K}}$ and writing $\mathbf{G}^T \bar{\mathbf{K}}^{-1} \mathbf{G} = (\mathbf{L}^{-1} \mathbf{G})^T \mathbf{L}^{-1} \mathbf{G}$.) By the definition of \mathbf{q} and assumption of the theorem, for any solution, \mathbf{q} , to (4.II.4),

$$\mathbf{q}^T \mathbf{G}^T \bar{\mathbf{K}}^{-1} \bar{\mathbf{F}} - \mathbf{q}^T \mathbf{H} = 0$$

Thus there are solutions to (4.II.8) of the form $\mathbf{p} + \mathbf{q}$, where \mathbf{q} is any solution to (4.II.4) and \mathbf{p} is orthogonal to all such \mathbf{q} . Given any such $\mathbf{p} + \mathbf{q}$, \mathbf{d}_0 is well determined by (4.II.7) (and depends only on \mathbf{p}). This completes the proof. ■

Corollary 1.1. When the solutions of Theorem 1 exist, $\{\mathbf{d}, \mathbf{p}\}$ is always unique if \mathbf{p} is taken to be \mathbf{Q} -orthogonal to all solutions to (4.II.4). When the only solution to (4.II.4) is $\mathbf{q} \equiv \mathbf{0}$, $\{\mathbf{d}, \mathbf{p}\}$ is the only solution; otherwise any $\{\mathbf{d}, \mathbf{p} + \mathbf{q}\}$ is a solution, where \mathbf{q} solves (4.II.4).

Proof. First we show that \mathbf{d} is unique (if it exists) in all cases: Suppose that there are two solutions, $\{\mathbf{d}_a, \mathbf{p}_a\}$ and $\{\mathbf{d}_b, \mathbf{p}_b\}$. Subtraction of equations shows

$$\begin{aligned}\bar{\mathbf{K}}(\mathbf{d}_a - \mathbf{d}_b) + \mathbf{G}(\mathbf{p}_a - \mathbf{p}_b) &= \mathbf{0} \\ \mathbf{G}^T(\mathbf{d}_a - \mathbf{d}_b) &= \mathbf{0}\end{aligned}$$

Thus

$$\mathbf{d}_a - \mathbf{d}_b + \bar{\mathbf{K}}^{-1} \mathbf{G}(\mathbf{p}_a - \mathbf{p}_b) = \mathbf{0} \quad (4.II.9)$$

and, going again to a discrete Poisson equation:

$$\mathbf{G}^T \bar{\mathbf{K}}^{-1} \mathbf{G}(\mathbf{p}_a - \mathbf{p}_b) = \mathbf{0} \quad (4.II.10)$$

We deduce that $\mathbf{p}_a - \mathbf{p}_b$ is in fact a solution to (4.II.4) (by taking the Cholesky decomposition of $\mathbf{G}^T \bar{\mathbf{K}}^{-1} \mathbf{G}$, as before). This has two important ramifications. First, it shows that $\mathbf{G}(\mathbf{p}_a - \mathbf{p}_b) = \mathbf{0}$; thus from (4.II.9), $\mathbf{d}_a = \mathbf{d}_b$, which proves the desired uniqueness.

To discuss the second ramification of (4.II.10), we use the uniqueness result we have just proved to write any solution of Theorem 1 as $\{\mathbf{d}, \bar{\mathbf{p}}\}$ for the fixed, unique \mathbf{d} and some pressure solution. The Hilbert projection theorem (with respect to \mathbf{Q} -orthogonality) enables us to write $\bar{\mathbf{p}}$ as

$$\bar{\mathbf{p}} = \mathbf{p}_0 + \mathbf{q}_0$$

where

$$\begin{aligned}\mathbf{p}_0^T \mathbf{Q} \mathbf{q}_0 &= 0 \\ \mathbf{G} \mathbf{q}_0 &= \mathbf{0}\end{aligned}$$

by taking \mathbf{q}_0 to be the projection of $\bar{\mathbf{p}}$ onto the null-space of \mathbf{G} . Observe that

$$\bar{\mathbf{K}}\mathbf{d} + \mathbf{G}(\mathbf{p}_0 + \mathbf{q}_0) = \mathbf{F}$$

$$\mathbf{G}^T \mathbf{d} = \mathbf{H}$$

but also

$$\bar{\mathbf{K}}\mathbf{d} + \mathbf{G}\mathbf{p}_0 = \bar{\mathbf{F}}$$

$$\mathbf{G}^T \mathbf{d} = \mathbf{H}$$

since $\mathbf{G}\mathbf{q}_0 = \mathbf{0}$. Thus $\{\mathbf{d}, \mathbf{p}_0\}$ is also a solution. Let $\{\mathbf{d}, \bar{\mathbf{p}}\}$ be some other solution
Similar arguments show

$$\bar{\bar{\mathbf{p}}} = \mathbf{p}_1 + \mathbf{q}_1$$

$$\mathbf{p}_1^T \mathbf{Q} \mathbf{q}_1 = 0$$

$$\mathbf{G}\mathbf{q}_1 = \mathbf{0}$$

and the results of (4.II.10) shows that

$$\mathbf{G}(\bar{\bar{\mathbf{p}}} - \bar{\mathbf{p}}) = \mathbf{0}$$

but

$$\mathbf{G}(\bar{\bar{\mathbf{p}}} - \bar{\mathbf{p}}) = \mathbf{G}(\mathbf{p}_1 - \mathbf{p}_0) = \mathbf{0}$$

A simple linear algebra argument (left to the reader) shows that if both \mathbf{p}_1 and \mathbf{p}_0 are \mathbf{Q} -orthogonal to the null space of \mathbf{G} , then $\mathbf{G}(\mathbf{p}_1 - \mathbf{p}_0) = \mathbf{0}$ only if $\mathbf{p}_1 = \mathbf{p}_0$. Thus the representative pressure solution, \mathbf{Q} -orthogonal to the solutions of (4.II.4), call it \mathbf{p} , is unique as required, and all other solutions are of the form $\mathbf{p} + \mathbf{q}$ with $\mathbf{G}\mathbf{q} = \mathbf{0}$. Thus the corollary is proved in its entirety. ■

We summarize developments to this point:

1. For solutions to (4.3.21) to exist, the right-hand side \mathbf{H} -vector must be orthogonal to the “pressure modes” in the sense of (4.II.5).
2. Orthogonality to the constant pressure mode, when it exists, is assured by global volume preservation of the boundary conditions.
3. When the solvability condition of (4.II.5) is satisfied, the “ \mathbf{d} -part” of the solution to (4.3.21) is unique, and there is a unique pressure solution \mathbf{Q} -orthogonal to the pressure modes, though any particular pressure solution may consist of this \mathbf{p} plus an arbitrary pressure mode.

4.II.3 TWO SIDES OF PRESSURE MODES

We should remind ourselves at this point just what the \mathbf{H} -vector is and why it will automatically satisfy the solvability condition with respect to the constant pressure mode but may have trouble with other pressure modes: Following Table 4.3.1, (4.II.5) requires

$$(q^h, \operatorname{div} \mathbf{g}^h) = 0$$

(4.II.11)

for all modes q^h with nodal-values q satisfying (4.II.4). Suppose $q^h = \text{constant}$ in Ω is a solution to (4.II.4); then (4.II.11) becomes (4.II.2). The solvability condition with respect to the constant mode implies that the (small-strain) volume change is zero for an incompressible elastic body. This must be arranged for by the analyst and (4.II.5) says that (4.II.2) must be satisfied exactly. We must take into account the exact volume change induced by q^h . This may be tricky if the boundary is complicated and may require a priori evaluation of boundary integrals of q^h , because the interpolated volume change implied by q^h may be slightly different from the volume change in the exact boundary conditions we are interpolating. But the point is that preserving volume exactly makes physical sense and is something most analysts would try to do—even if it did pose some routine technical difficulties.

When q^h is a spurious pressure mode, there is no obvious physical interpretation of (4.II.11). Spurious pressure modes place additional, apparently unphysical, constraints on q^h , which we would be unlikely to satisfy as a matter of course. The problem in dealing with such constraints is twofold: First the pressure modes for the element, mesh, and boundary conditions under consideration must be characterized. Second, the essential boundary conditions may have to be modified in such a way that (4.II.11) is satisfied, *exactly*. This seems to pose formidable obstacles for the analyst, and at this point we might wonder whether opting for safe elements without spurious modes might indeed be the wisest course. The fact is that we have given all the bad news about spurious modes first, and while we cannot sweep the technical difficulties of using elements which have them under the rug, there now is good news: First, elements which have spurious modes seem to have some distinct advantages not available with safe elements; second, employment of elements with spurious modes in a penalty formulation seems to alleviate many of the potential difficulties; and third, new analysis and error estimates have shown the way systematically to obtain good results for some elements with spurious modes. Before we pursue these ideas further, it seems appropriate to look at a now-classical example of the interaction between pressure modes and boundary conditions. The problem considered is that of driven cavity flow described in Sec. 4.4.1 and illustrated in Fig. 4.4.6. The reader more familiar with elasticity is reminded of the formal equivalence of Stokes flow and linear elasticity discussed in Sec. 4.2.

Example 1 (Driven Cavity Problem—A Stokes-flow Problem with Inhomogeneous Boundary Conditions)

Let N_1, N_2, \dots, N_M be the global shape functions associated with nodes 1, 2, ..., M of Fig. 4.II.1. Consider three ways of specifying the unit velocity of the driven lid, obtained from three choices for q^h (see Table 4.3.1):

Case 1: Leaky lid

$$q^h = \begin{Bmatrix} \sum_{i=1}^M N_i \\ 0 \end{Bmatrix} = \begin{Bmatrix} q_1^h \\ q_2^h \end{Bmatrix}$$

Note that

$$\varphi_1^h(x_1, x_2) = \begin{cases} \frac{x_2}{h} & x_2 \geq 0 \\ 0 & x_2 < 0 \end{cases}$$

Thus $\operatorname{div} \varphi^h \equiv 0$, and (4.II.11) is automatically satisfied for any possible modes. This was the φ^h used in Sec. 4.4.1.

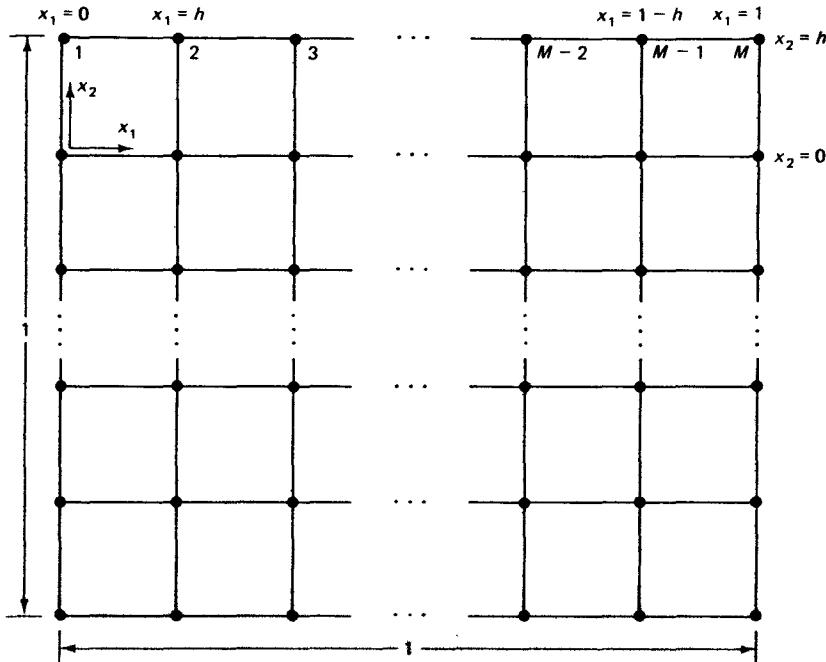


Figure 4.II.1 Discretization of the driven cavity (see Fig. 4.4.6) by bilinear displacement-constant pressure elements with $M - 1$ elements per side. Elements are of side length h .

Case 2: Ramp over one element

$$\varphi^h = \begin{cases} N_2 + \sum_{i=3}^{M-2} N_i + N_{M-1} \\ 0 \end{cases}$$

In this case, we assume that $M \geq 4$ and that the summation term is zero if $M = 4$. Observe that instead of allowing flux through the domain as in case 1, the velocity component, u_1 , is zero at nodes 1 and M and linearly rises to $u_1 = 1$ in the first element and linearly falls from $u_1 = 1$ to zero in the $(M - 1)$ st element. In this case

$$\varphi_1^h = \begin{cases} \frac{x_1 x_2}{h^2} & 0 \leq x_1 \leq h, x_2 \geq 0 \\ x_2/h & h < x_1 < 1 - h, x_2 \geq 0 \\ \frac{(1 - x_1)x_2}{h^2} & 1 - h \leq x_1 \leq 1, x_2 \geq 0 \\ 0 & x_2 < 0 \end{cases}$$

and thus

$$\operatorname{div} \varphi^h = \begin{cases} \frac{x_2}{h^2} & 0 \leq x_1 \leq h, x_2 \geq 0 \\ 0 & h < x_1 < 1 - h, x_2 \geq 0 \\ \frac{-x_2}{h^2} & 1 - h \leq x_1 \leq 1, x_2 \geq 0 \\ 0 & x_2 < 0 \end{cases}$$

Note that

$$\int_{\Omega} \operatorname{div} \varphi^h d\Omega = 0$$

But the bilinear velocity-constant pressure element has an additional mode, the **checkerboard mode** (see [86], for example):

$$c_B^h = (-1)^{i+j-M+1} \quad i, j = 1, 2, \dots, M-1$$

in the element which is the i th element in the x_1 -direction and the j th in the x_2 -direction. In particular, for the top row of elements, the checkerboard mode may be expressed as

$$c_B^h = (-1)^i \quad i = 1, 2, \dots, M-1$$

Thus a simple calculation shows that

$$\int_{\Omega} c_B^h \operatorname{div} \varphi^h d\Omega = -\frac{h}{2}[1 + (-1)^{M-1}]$$

So for an even number of elements in the x_1 -direction (i.e., M odd), (4.II.11) is *not* satisfied. Evidently, *no* solution to the discrete problem exists in this case.

Case 3: Ramp over two elements

In this case we take $\varphi_2^h \equiv 0$ and

$$\varphi_1^h = \begin{cases} \frac{x_1 x_2}{2h^2} & 0 \leq x_1 \leq 2h, x_2 \geq 0 \\ x_2/h & 2h < x_1 < 1 - 2h, x_2 \geq 0 \\ \frac{(1 - x_1)x_2}{2h^2} & 1 - 2h \leq x_1 \leq 1, x_2 \geq 0 \\ 0 & x_2 < 0 \end{cases}$$

We assume here that $M \geq 5$, so that at the very least we have nodes at $x_1 = 0, h, 2h, 1 - h$, and 1. Equivalently, we could write \mathbf{q}^h in terms of the N_i 's as before, using nodal values of 0, 0.5, and 1 at the first three nodes on the lid, to "ramp" u_i linearly from zero to one over the first two elements; the same nodal values in reverse order are used at the last three nodes to bring u_i down from 1 to 0. We find here that

$$\operatorname{div} \mathbf{q}^h = \begin{cases} \frac{x_2}{2h^2} & 0 \leq x_1 \leq 2h, x_2 \geq 0 \\ 0 & 2h < x_1 < 1 - 2h, x_2 \geq 0 \\ \frac{-x_2}{2h^2} & 1 - 2h \leq x_1 \leq 1, x_2 \geq 0 \\ 0 & x_2 < 0 \end{cases}$$

Consequently

$$\int_{\Omega} \operatorname{div} \mathbf{q}^h d\Omega = 0$$

and

$$\begin{aligned} \int_{\Omega} c_B^h \operatorname{div} \mathbf{q}^h d\Omega &= \int_0^h \int_0^{2h} c_B^h \operatorname{div} \mathbf{q}^h dx_1 dx_2 + \int_0^h \int_{1-2h}^1 c_B^h \operatorname{div} \mathbf{q}^h dx_1 dx_2 \\ &= - \int_0^h \left[\int_0^h \frac{x_2}{2h^2} dx_1 - \int_h^{2h} \frac{x_2}{2h^2} dx_1 \right] dx_2 \\ &\quad - (-1)^M \int_0^h \left[\int_{1-2h}^{1-h} \frac{x_2}{2h^2} dx_1 - \int_{1-h}^1 \frac{x_2}{2h^2} dx_1 \right] dx_2 = 0 \end{aligned}$$

We get zero because each bracketed term is zero independently. Unlike case 2, the value of $(-1)^M$ is not crucial, and (4.II.11) is always satisfied for any $M \geq 5$.

We summarize the important observations of Example 1:

1. In all three cases, the boundary condition is the same in the limit $h \rightarrow 0$.
2. In the leaky-lid case, $\mathbf{u}^h \cdot \mathbf{n}$ is nonzero in the elements at the top corners of Ω , but (4.II.2) is satisfied because $\mathbf{u}^h \cdot \mathbf{n}$ is of equal magnitude and opposite in sign at corresponding points of the first and last elements of the top row, i.e., *mass is globally conserved*. In other cases, mass is globally conserved because there is no flow through the domain boundary. These cases have obvious analogies to preservation of volume in elasticity.

There is strong evidence to suggest that essential boundary conditions can always be modified to satisfy (4.II.11) without loss of accuracy, because it can be shown—at least for interior approximation—that the checkerboard modes cannot contribute to approximation accuracy [85]. We suspect that this result can be generalized to approximation of boundary conditions.

What we have seen so far is only part of the story about spurious modes. If the reader wonders why we dwell at such length on spurious modes and why we will continue to do so for much of this appendix, we should say at this point that not only is learning to understand and to live with spurious modes a practical matter—it allows the analyst a much wider choice of finite elements for problems with the incompressibility constraint—but understanding spurious modes is also the key to understanding the whole subject of constraints in the finite element method. We now are in a position to appreciate this fact. What we have observed about spurious modes up to now suggests that they are invariably tied to boundary conditions, and are some sort of unphysical artifact of elements which would best be avoided. The only spurious modes we have actually *seen* so far *are* indeed associated with boundary conditions. However, there is another kind of spurious mode which is illustrated by the following example.

Example 2 (The Crossed-Triangle Element and Redundant Constraints)

This example concerns the linear crossed-triangle macroelement discussed in Remark 3 of Sec. 4.4, which is illustrated in Fig. 4.4.4(a). Given $\mathbf{u}^h \in \mathcal{S}^h$, consider a typical macroelement and let e_1, e_2, e_3, e_4 denote the four subtriangles, ordered in counter-clockwise fashion. It is important to note that the typical macroelement under consideration can be an arbitrary quadrilateral, but the central node *must* fall on the intersection of the diagonals of the quadrilateral. Mercier proved that

$$\operatorname{div} \mathbf{u}^h|_{e_1} - \operatorname{div} \mathbf{u}^h|_{e_2} + \operatorname{div} \mathbf{u}^h|_{e_3} - \operatorname{div} \mathbf{u}^h|_{e_4} = 0$$

Mercier then argued that for \mathcal{V}^h based on piecewise constants on the triangles, only three of the four incompressibility constraints are independent, since setting $\operatorname{div} \mathbf{u}^h = 0$ on three triangles forces it to be zero on the fourth. As explained in Sec. 4.4, this has the effect of modifying the apparent constraint ratio of $r = 1$ to a more acceptable $r = \frac{4}{3}$.

This argument, which shows that the macroelement is not overconstrained, can be turned around to show that the macroelement also has many modes. Consider again our typical macroelement, labeled M , and define

$$c_M^h \equiv \begin{cases} (-1)^i / a^{e_i} & \text{on triangle } e_i \\ 0 & \text{outside of macroelement } M \end{cases}$$

where a^{e_i} is the area of triangle e_i . Now

$$\int_{\Omega} c_M^h \operatorname{div} \mathbf{u}^h d\Omega = \int_{\Omega^M} c_M^h \operatorname{div} \mathbf{u}^h d\Omega = \sum_{i=1}^4 a^{e_i} (c_M^h \operatorname{div} \mathbf{u}^h)_{e_i}$$

since c_M^h and $\operatorname{div} \mathbf{u}^h$ are constant on triangles. But the definition of c_M^h and Mercier's equation immediately lead to the conclusion that

$$\int_{\Omega} c_M^h \operatorname{div} \mathbf{u}^h d\Omega = 0$$

for any $\mathbf{u}^h \in \mathcal{S}^h$, and thus c_M^h is a pressure mode by (4.II.1). Such a c_M^h was constructed without the imposition of any boundary conditions, and it could have been constructed

for a mesh consisting of a single macroelement. Such a mode, then, must be a mode of the macroelement's "element" matrix. Note that there is one such mode for each macroelement.

To summarize the salient points of Example 2:

1. There are evidently two kinds of spurious modes: There are global ones introduced by the imposition of certain types of boundary conditions when certain elements are used. These are the kind discovered by Fortin and they do not appear to be very useful. Then there is the possibility of *local spurious modes*, which are spurious modes of the macroelement matrix without any imposed boundary conditions; these are responsible for the favorable constraint ratio of crossed triangles.
2. The crossed triangle element has the incompressibility constraint enforced in the strongest possible way by the Lagrange multiplier—*pointwise exactly*.
3. A price that must be paid, it seems, for having an element that satisfies the incompressibility constraint exactly is the presence of a global checkerboard mode. This occurs for the crossed triangle on meshes which would have such a mode for bilinear rectangles. The mode is identical to the mode associated with the bilinear-constant pressure element. (The proof of this may be found in [86].)

We will refer to spurious modes of the kind observed by Fortin, induced by boundary conditions as "(spurious) modes of the first kind," or "global (spurious) modes." Those of the type that we just encountered in Example 2, which are due to redundant constraints on the element level, we refer to as "(spurious) modes of the second kind," or "local (spurious) modes." The discovery of the redundancy of constraints is due to Nagtegaal et al. [87], and Mercier [88, 89]. Mercier showed that crossing triangles also leads to a dramatic improvement in the constraint ratio for quadratic triangles as well as linear ones (more about this later). These authors did not phrase their discovery in what now can easily be seen to be a nearly paradoxical statement: Some elements seemed to work *only* because they had spurious modes, but standard error analysis predicted their failure precisely *because* they had spurious modes. Something had to give, and it turns out that it was standard error analysis which gave way. Still, at this point, it may seem a dubious proposition to employ elements with local spurious modes: To increase the constraint ratio in any meaningful way with local modes seems to require a very substantial number of such modes. To arrange satisfaction of (4.II.11) for all such modes at first sight seems a tedious task at best and hopeless at worst. Then, even if we could resolve that problem, according to (4.II.3), each independent pressure mode leads to an additional rank deficiency of the system of equations to be solved. This could pose formidable computational problems. Evidently, however, it was intended by some grand design that we should be able to use elements with spurious modes, and each of these problems can be dealt with easily. In what follows we use the word "element," but with Example 2 in mind, the element could just as well be a macroelement.

Theorem 2. Consider the incompressible form of (4.3.21), with a pressure trial space, \mathcal{P}^h , based on the same nodal pattern as the displacement space, \mathcal{S}^h , but with *discontinuous* pressures (no sharing of nodal values between elements at all). If the element gradient matrix, \mathbf{g}^ϵ , has a spurious mode, \mathbf{q}^ϵ , i.e., if

$$\mathbf{g}^\epsilon \mathbf{q}^\epsilon = \mathbf{0} \quad (4.\text{II}.12)$$

there is a corresponding spurious mode $\mathbf{q}^h \in \mathcal{P}^h$ for each such element in the mesh such that

$$\mathbf{G}\mathbf{q} = \mathbf{0}$$

However, such a spurious mode satisfies (4.II.11) for any \mathbf{g}^h , i.e., a spurious mode arising from a local, or element, spurious mode is always consistent with any inhomogeneous boundary conditions imposed via “ \mathbf{q} -node” specification.

Proof: To construct a pressure mode from an element mode, just extend the element mode outside the element with a zero value on all the rest of Ω ; this can be done because the pressures admit a discontinuity at the element boundary. One such mode is thus constructed for each element in the mesh, and such modes are \mathbf{Q} -orthogonal since they have disjoint supports. Thus we have constructed the desired number of independent modes.

To see the second part of the theorem, consider the constraint matrix, $\bar{\mathbf{G}}^T$, associated with the space $\mathcal{S}^h \times \mathcal{P}^h$, i.e., without any essential boundary conditions. We may deduce that $\bar{\mathbf{G}}\mathbf{q} = \mathbf{0}$ for any local \mathbf{q} constructed as in the first part of this proof. This follows since

$$\bar{\mathbf{G}}\mathbf{q} = \mathbf{0} \Leftrightarrow \mathbf{g}^\epsilon \mathbf{q}^\epsilon = \mathbf{0} \quad (4.\text{II}.13)$$

for the \mathbf{g}^ϵ and \mathbf{q}^ϵ from which \mathbf{q} was constructed, which may in turn be attributed to the fact that there is no nodal-value sharing. But then, $\mathbf{q}^h \in \mathcal{S}^h$, and thus (4.II.13) implies (4.II.11) when \mathbf{q}^h in (4.II.11) is taken to be the mode whose nodal values appear in (4.II.13). This concludes the proof of the theorem. ■

Theorem 2 is very important in practice: It means that *no modification of essential boundary conditions is required to accommodate the spurious modes of the second kind*, which must be numerous to be useful. These spurious modes—indeed any spurious modes—always have a dual displacement field which is weakly incompressible. It is not possible to identify the spurious mode with a specific weakly incompressible field, but rather we infer that there is such a field because each spurious mode increases the effective constraint ratio referred to in Example 2. To see this, we need to sharpen the definition of constraint ratio given in Sec. 4.3. The new definition explicitly counts the redundancy of constraints and incorporates this number in the computation of n_c of the earlier definition. We note also that the new definition is based on the constraint matrix, not the mesh; therefore it is not tied to the special test mesh of Fig. 4.3.3. We do lose the nice property of that mesh which assured that for discontinuous pressures, r could be found by considering only one element; it will differ slightly because of the possibility of more or fewer “ \mathbf{q} -nodes”, and the

possibility of spurious modes of the first kind introduced by some boundary conditions. It will agree with the previous r in the limit of mesh refinement if the spurious modes are properly taken into account in computing n_c .

Definition. (Redefinition of the constraint ratio.) The *constraint ratio*, \bar{r} , is defined as the dimension of \mathcal{S}^h divided by the number of effective constraints implied by \mathbf{G}^T , which is the number of independent rows (equations) of \mathbf{G}^T or the number of independent columns of \mathbf{G} .

Theorem 3. Let “dim” denote “dimension of” (as a vector space), and “ker” denote “kernel (null space) of”. The *constraint ratio* (as redefined above) is given by

$$\bar{r} = \frac{\dim \mathcal{S}^h}{n_z}$$

where

$$n_z \equiv \dim \mathcal{D}^h - \dim \ker \mathbf{G} \quad (4.II.14)$$

Proof: To establish the result requires only that n_z be the number of independent rows of \mathbf{G}^T . The law of nullity of linear algebra states that the rank of \mathbf{G} is equal to the dimension of the domain space, $\dim \mathcal{D}^h$, minus the nullity of \mathbf{G} , $\dim \ker \mathbf{G}$. But then the number of independent equations in \mathbf{G}^T is equal to its rank, which is also the rank of \mathbf{G} , by an equally basic result of linear algebra. This concludes the proof. ■

So actually the theorem is a linear-algebraic triviality, but by writing the triviality in the way we did in (4.II.14) we learn something nontrivial: *Each spurious mode reduces the number of constraints by one—in general, not just for the crossed triangle*. We next deal with the second objection to spurious modes voiced earlier; the rank deficiency of the matrix equations, which is evidently increased by one for each mode.

4.II.4 PRESSURE MODES IN THE PENALTY FORMULATION

While there may be ways to solve the resultant equations in Lagrange-multiplier form, despite the singularity, we believe that the most useful elements possessing spurious modes of the second kind are ideally suited for penalty methods. These are the crossed triangular elements of arbitrary degree which are useful because, as we observed in Example 2 with crossed linear triangles, exact incompressibility can be obtained by the Lagrange-multiplier solution. For these elements the exact incompressibility of the Lagrange-multiplier solution translates into an $O(\frac{h}{\lambda})$ compressibility error of the penalty solution. This is a much better compressibility error than can be expected from most elements, which will usually have $O(\frac{h}{\lambda}) + O(h^m)$ compressibility error, for some m dependent on both \mathcal{S}^h and \mathcal{D}^h . This means that in most cases the h -dependent term will dominate, and a higher degree of satisfaction of the constraint can only be

obtained by mesh refinement, which is unlikely to lead to a dominance of the penalty term, even for the finest meshes. We will amplify on this theme as we go along, but first we will pursue a question which may seem puzzling at the moment: What happens to pressure modes in a penalty formulation?

We have seen in an earlier section that the penalty formulation produces positive-definite matrices of standard finite element form; the only obstacle to a solution procedure is penalty-induced ill-conditioning, which can be minimized by judicious choice of the penalty. This applies to reduced and selective integration techniques and \bar{B} methods. It should be noted that in plane problems using Cartesian coordinates, the crossed triangle family of elements can be integrated *exactly*, as was implied in Example 2. (All integrands consist of polynomials as long as the global triangle domain is an affine mapping of the parent domain.) The spurious modes of the second kind in the limiting Lagrange-multiplier method give a favorable constraint ratio for the penalty method. One can easily deduce that the integration *must* be exact for the discontinuous pressure trial space based on the integration points to contain the divergences of the displacements. Actually, in such cases, the reduced and selective, \bar{B} , and exactly integrated slightly compressible formulations are identical. This brings us squarely up against the question of axisymmetry: Current wisdom seems to suggest that the \bar{B} approach is the best approach here; use the pressure shape functions which give the best constraint ratio versus pressure accuracy (for the crossed triangle, for example, this would be the pressures based on the nodes of the quadrature formula which would have been exact in Cartesian coordinates). Use a quadrature formula to evaluate the integrals of (4.5.1) and (4.5.2) which is “accurate enough” near the axis of symmetry (i.e., the singularity) to give confidence in the accuracy of the evaluation of the integrals. Use as many points—at least for the elements near the symmetry line—as experience leads you to believe you need. You may have to experiment to balance cost of using many points with the quality of solutions you desire. One thing you need not worry about is the constraint ratio, since the \bar{B} formulation has freed the choice of quadrature formula from assumed pressure interpolation. If reduced integration is used, or in the case of the crossed triangles, the formula which would have been exact in Cartesian coordinates is used, the equivalence theorem of Sec. 4.4 between penalty method and *numerically integrated* mixed method still holds, but the quadrature formula might lead to excessive integration error in the region of the singularity at the axis of symmetry in the numerically integrated mixed method. Thus, although the penalty method is equivalent to a mixed method, it might not be a particularly good mixed method. It was this difficulty that the original mean-dilatational methods were intended to address.

The astute reader may have already guessed why we are up against the problem of axisymmetry: Whether we use accurately integrated \bar{B} or try to get away with reduced or selective integration, the advantages of the crossed triangle elements seem to disappear in axisymmetry. Because the space “ $\text{div } \mathcal{S}^h$ ” of divergences of members of \mathcal{S}^h contains u^h/r terms, $\text{div } \mathcal{S}^h$ is no longer contained in \mathcal{D}^h and thus the exact incompressibility of the Lagrange-multiplier solution is lost. It does appear that this problem can be circumvented by using trial functions that are not polynomials and are specially suited to the coordinate system. This would also lead to quadrature formulas

that would be exact for the crossed triangles in axisymmetry. What we say subsequently will be true for axisymmetry, but the advantage of using the elements with spurious modes of the second kind is yet to be realized in axisymmetry.

The next theorem, about modes in the penalty formulation, gives a partial answer to the question of what happens to modes in penalty methods: They are invisible, at least as far as the penalty pressure is concerned. Part (b) of the following theorem requires some techniques beyond the scope of the present discussion. A proof can be constructed based on arguments like those used in [89] to prove a similar result for the continuous rather than discrete problem. The reader who has digested the material in Appendix 4.I and this appendix might be able to see how this is done, and the bravest of readers are invited to try a proof as an exercise.

Theorem 4. Consider either a reduced or selective penalty method or a \bar{B} method and the corresponding mixed method of (4.3.21), with all matrices involved in the mixed and penalty formulations evaluated with the appropriate numerical quadrature to make an exact equivalence between the *appropriately integrated* mixed method and penalty method (exact integration allowed), in the sense of Malkus-Hughes [36]. Use the same quadrature on the relevant matrices of the limiting Lagrange-multiplier method. Assume the algebraic consistency condition, (4.II.5), is satisfied with respect to all modes of the Lagrange-multiplier method, for the penalty method as well as the limiting Lagrange-multiplier method. Let $\epsilon = \mu/\lambda$ and p_ϵ^h be the penalty pressure solution corresponding to that value of ϵ . Let p^h be the Lagrange multiplier, or $\epsilon = 0$, solution of Corollary 1.1, which is Q -orthogonal to all pressure modes. Then

- p_ϵ^h is Q -orthogonal to all the pressure modes of the corresponding Lagrange-multiplier method.
- $\|p_\epsilon^h - p^h\|_0 < C\epsilon$, for a constant, C , independent of ϵ .

Proof of Part (a). In any of the penalty methods under consideration—reduced or selective or \bar{B} —the pressure is computed by a *weak form* of (4.4.14).

$$\langle q^h, p_\epsilon^h \rangle = -\lambda \langle q^h, \operatorname{div} u^h \rangle \quad \text{for all } q^h \in \mathcal{P}^h \quad (4.\text{II}.15)$$

where $\langle \cdot, \cdot \rangle$ denotes the *appropriately integrated L_2 -inner product*, which implies satisfaction of Eq. (4.4.14) at the integration points (i.e., \mathcal{P}^h -nodes) in the reduced and selective cases. In general, (4.II.15) is equivalent to

$$p_\epsilon = \mathbf{M}^{-1} \mathbf{G}^T \mathbf{d} \quad (4.\text{II}.16)$$

where the use of numerical integration on the matrices corresponds to the use of the appropriate $\langle \cdot, \cdot \rangle$; p_ϵ is the vector of nodal values of p_ϵ^h and \mathbf{d} the nodal values of the penalty displacement solution. Taking the Q -inner product of a pressure mode with nodal values, \mathbf{q} , involves premultiplying (4.II.16) by $\mathbf{q}^T \mathbf{Q}^T$, from which the desired result follows immediately. This concludes the proof of (a). ■

There are a few important points related to the theorem that should be emphasized:

1. In almost all cases, the quadrature applied to \mathbf{M} is accurate enough to imply that the penalty p_ϵ^h is L_2 -orthogonal to a constant pressure mode, should one exist. This is particularly true when the quadrature is exact on \mathbf{M} , which is true in many practical cases.
2. While the algebraic consistency condition is assumed to hold and the penalty pressures are free of modes, these facts alone will be seen to be not enough to guarantee the convergence of the penalty pressures to the exact solution. We will see that a postprocessing of the penalty pressures is often wise [85, 90].
3. It is possible to show that the same $O(\epsilon)$ convergence rate holds for penalty displacement solutions to corresponding Lagrange-multiplier solutions. The readers who try to prove this and part (b) of the theorem should be warned that if the element fails to satisfy the “LBB, or Babuška-Brezzi condition,” (discussed further in the next section) the constant C may appear to be unfavorably dependent on h . However, this does not appear to be the case in practice.

Finally, there is a neat heuristic way to describe what happens to pressure modes in a penalty formulation. Recall that the coefficient matrix of the linear system to be solved in such cases is

$$\bar{\mathbf{K}} = \mathbf{G}\mathbf{M}^{-1}\mathbf{G}^\top$$

So the pressure modes are “trapped inside” the penalty matrix and visible on the outside only as extra weakly incompressible modes. According to Theorem 4, the pressure modes remain invisible even when the penalty pressure is recovered from the displacements. The only effect of the modes is observed if we fail to modify the essential boundary conditions to make the \mathbf{H} -vector orthogonal to the modes. Then the effects can produce dramatically nonsensical results [83], which caused much confusion and consternation among researchers until all this was sorted out (to the extent it is now).

4.II.5 THE BIG PICTURE

In what follows, we want to put together what we have observed about pressure modes in an overall heuristic picture of the incompressibility constraint in the finite element method. We want our picture to be painted with broad strokes, overlooking but not denying the kind of detailed structure we have observed to this point. We can make things easy for ourselves by thinking of exactly integrated Lagrange-multiplier methods in Cartesian coordinates; this is the underlying prototypical method from whose success or failure the success or failure of all subtle variants and closely related methods will follow.

We argue that there are two types of finite elements, *both* of which are useful in the constrained finite element method, *both* of which can lead to optimal convergence rates in specific cases: “locking elements” and “nonlocking elements.” To make this distinction, we need to define two subspaces of the trial solution space. The

first is composed of solutions satisfying the *weak constraint*,

$$\mathbf{W}^h = \{u^h \in \mathcal{S}^h \mid \operatorname{div} u^h \perp \mathcal{P}^h\}$$

where “ \perp ” refers to L_2 -orthogonality. Since we are assuming exact integration, we can see that \mathbf{W}^h is the subspace of all weakly incompressible trial solutions with respect to the constraint as enforced by the Lagrange-multiplier method. The next space is one to which the reader may not have given much thought: the space of all trial solutions that are exactly pointwise-incompressible,

$$\mathbf{V}^h = \{u^h \in \mathcal{S}^h \mid \operatorname{div} u^h = 0\}$$

Clearly $\mathbf{V}^h \subset \mathbf{W}^h$. Note that the definition of \mathbf{V}^h is intrinsic to the displacement space alone and makes no reference to a pressure trial space.

Definition. Consider the test mesh of Fig. 4.3.3; an element is a *locking element* if there is a constant, C_1 , independent of n_{es} such that

$$\dim \mathbf{V}^h \leq C_1 n_{es}$$

An element is a *nonlocking element* if there exists a constant, C_2 , independent of n_{es} such that

$$\dim \mathbf{V}^h \geq C_2 n_{es}^2$$

We note that the determination of whether an element is of the locking or nonlocking type is independent of any choice of pressure trial space. The prototypical examples of the two kinds of element are as follows:

Locking elements

1. Linear triangles in nonredundant arrangements (such as Fig. 4.3.4(a))
2. All Lagrange elements
3. All serendipity elements.

Nonlocking elements

1. All triangles of degree greater than or equal to two
2. Linear crossed triangles

We are now ready to shop for a space of Lagrange multipliers and pressures, knowing that if we try to enforce the incompressibility constraint too vigorously with locking elements, the results can reflect what the name implies. With the nonlocking elements we evidently can be more vigorous in enforcement of the constraint; how vigorous we can or should be will be tempered by what follows. The first caution we have is contained in the next theorem, which covers only the special cases in which there is a containment relation one way or another between $\operatorname{div} \mathcal{S}^h$ and \mathcal{P}^h :

Theorem 5. For an exactly integrated Lagrange-multiplier method,

- a. If $\mathcal{P}^h \subset \operatorname{div} \mathcal{S}^h$, then there are no modes and incompressibility is imposed only approximately by the weak constraint when the containment is proper.
- b. If $\mathcal{P}^h = \operatorname{div} \mathcal{S}^h$, then there are no modes and incompressibility is imposed exactly.
- c. If $\mathcal{P}^h \supset \operatorname{div} \mathcal{S}^h$, then there are modes if the containment is proper and incompressibility is imposed exactly.

Proof: The result follows easily from reconsidering the equation

$$(q^h, \operatorname{div} u^h) = 0 \quad (4.\text{II}.17)$$

In case (a), if (4.II.17) is to hold for fixed q^h and all $u^h \in \mathcal{S}^h$, there is a u^h with $\operatorname{div} u^h = q^h$ by assumption; thus any q^h satisfying (4.II.17) is zero. If \mathcal{P}^h is not all of $\operatorname{div} \mathcal{S}^h$ then there is a nonzero member of $\operatorname{div} \mathcal{S}^h$ orthogonal to all of \mathcal{P}^h . Such a field is not exactly incompressible but is weakly incompressible. This takes care of (a). In case (b), we easily deduce that there are no modes for the same reason there are none in case (a). This time there is nothing in $\operatorname{div} \mathcal{S}^h$ orthogonal to all of \mathcal{P}^h except the zero field. Thus all weakly incompressible fields are exactly incompressible fields. This takes care of case (b). In case (c), weakly incompressible implies exactly incompressible for the same reason as in case (b). When the containment is proper, there is something nonzero in \mathcal{P}^h orthogonal to all of $\operatorname{div} \mathcal{S}^h$, because the latter is a subspace of the former, but any such pressure is a mode by definition. This takes care of case (c) and proves the theorem. ■

The proof of the following corollary is left to the reader:

Corollary 5.1. When the global constant is a pressure mode, Theorem 5 holds if \mathcal{P}^h is replaced by the subspace of \mathcal{P}^h consisting of members of \mathcal{P}^h orthogonal to constants, and the word “mode” is replaced by the words “mode other than the global constant.”

What the theorem and corollary say is that in the special case of containment holding one way or another as described, modes result from having more pressures than needed to enforce exact incompressibility. We are at the heart of the matter; now is the time to bear down and hold fast to the unraveling threads.

One of the first questions which the reader is likely to ask is: Are there any elements for which the containment relations hold? We have to hedge the answer somewhat by saying that it is *easy* to find elements for which case (c) of Theorem 5 and Corollary 5.1 hold, but it is very hard to verify cases (a) and (b). The reason we have to hedge is a *crucial* point: It is easy to identify the possible polynomial terms in $\operatorname{div} \mathcal{S}^h$, and in most cases, there *seem* to be standard finite element spaces that could be taken for \mathcal{P}^h and contain *precisely* those terms. For example, $\operatorname{div} \mathcal{S}^h$ for quadratic triangles contains pure linear terms, for quadratic serendipity elements $\operatorname{div} \mathcal{S}^h$ contains pure quadratic terms, for biquadratic elements $\operatorname{div} \mathcal{S}^h$ contains just those terms associated with quadratic serendipity elements, and so on. But $\operatorname{div} \mathcal{S}^h$ always contains fewer independent members than the total number of possible polynomial terms. The reason is that interelement continuity enforced in \mathcal{S}^h constrains the polynomials in the

divergences to be related between elements in some complicated and hard-to-characterize way, so that fewer than all possible combinations of piecewise discontinuous polynomial terms available are actually allowed. If one attempts to strike the balance of case (b)—which seems to be the golden mean for nonlocking elements—by assuring that the correct polynomial terms are independently present in \mathcal{P}^h on each element, the results will almost surely be modes. Since it is an extremely difficult task to assure that \mathcal{D}^h is entirely composed of members which are divergences of continuous displacements, it is no more easy to arrange case (a) either. The only practical course, if exact incompressibility is desired, is to match all possible polynomial terms independently in \mathcal{D}^h and engender modes. Recent work of L. R. Scott and M. Vogelius appears to show that with higher-degree triangles and a particular choice of boundary condition, there are—almost miraculously—examples of the occurrence of case (b) [91].

Theorem 5 and its corollary are of practical importance for nonlocking elements and triangles, in particular. It forces us to subtly reinterpret spurious modes of the second kind: The picture we got in the wake of Theorem 3 was a picture in which \mathcal{D}^h was given as if from on high. The members of \mathcal{D}^h were assumed to be independent constraints until proven otherwise—if there were too many constraints, we could buy back an incompressible field only at the price of a mode. Now the point of view adopted starts by looking at V^h and tries to get a \mathcal{D}^h that is accurate enough and as close as possible to $\text{div } \mathcal{S}^h$; this can be done easily only by throwing in all possible polynomial terms, and in so doing we usually get too many pressures, at least as far as enforcing incompressibility without modes is concerned. These excess pressures just get dumped into the mode pool; as long as we have assured at least case (b) of Theorem 5, we could continue (uselessly) to pour on the pressures, and they would continue to run off into the mode pool. Those readers who are familiar with Fraeijs de Veubeke's limitation principle [42] will now recognize our new interpretation of modes of the second kind. We are saying that $\text{div } \mathcal{S}^h$ usually falls inconveniently “in between” possible choices for \mathcal{D}^h , and modes result from crossing the limitation line in choosing a convenient and accurate \mathcal{D}^h .

Many of the locking elements can be successfully employed by backing off on the incompressibility constraint to obtain some kind of analogy to case (a) of Theorem 5. The most successful of these elements appear to be the Lagrange elements. Taking these examples, we see that an element of the “bi- k degree” is really an element of degree k in terms of attainable accuracy, but it contains terms of degree $2k$. Apparently optimal accuracy can be obtained by choosing pressure trial spaces that contain complete polynomials of degree $k - 1$. The best example of this is the biquadratic displacement–pure linear pressure element of Fig. 4.3.4(h): an optimal, safe element, and probably the best of all safe elements. The key to the success of Lagrange elements is that optimal pressure accuracy can be obtained without including all the polynomial terms from the divergences of the “partial polynomials” of degree higher than $k - 1$. Inclusion of such terms would, of course, lock the element. Note that the same cannot be said of triangles: A triangle of degree k has only terms of degree $k - 1$ in $\text{div } \mathcal{S}^h$, and it seems that they all need to be included to get optimal

accuracy. Backing off on the constraints for triangles seems doomed to suboptimality—unless continuous pressure fields are used. Nodal-value sharing can reduce the constraint ratio to avoid modes while retaining full accuracy of the pressures. Continuous pressure fields are not very suitable for pushing to the ultimately low level of compressibility, because there is no recourse to penalty methods using globally connected pressure fields if modes are incurred. And as has been observed for equal-order interpolations (Fig. 4.3.6(a) and (b)) modes can result without enforcing a high degree of incompressibility. Continuous pressure fields can be employed successfully with quadrilaterals (see Fig. 4.3.4(c)), but they seem most appropriately used to retain optimal accuracy with triangles and avoid spurious modes. (Of course, the price of sharpness of enforcement of the constraint is extra unknowns and lack of positive-definiteness.)

Back to biquadratics with discontinuous pressure fields: Biquadratic displacement–linear pressure elements back off on the constraint as far as possible, and there are no modes of either the first or second kind [92]. The biquadratic displacement–bilinear pressure element (Fig. 4.3.4(g)) moves closer to the edge and can give a spurious mode of the first kind [92]. The advantages of the first of these two elements is that it is safe and optimal; the advantage of the second is that it enforces the constraint more strongly and can be used in a reduced selective formulation. The biquadratic displacement–linear pressure element must be used in $\bar{\mathcal{B}}$ -form to avoid the integration error possible using the three-point formula. This is no real drawback and can be done very efficiently, but it does mean that the analyst cannot implement the element just by changing the integration formula in a pre-existing code. We do not know if the biquadratic displacement–linear pressure element precisely satisfies case (a) of Theorem 5 or its corollary; that is difficult to check. The spirit of the element is very much the same as case (a): By taking a subspace of the polynomial terms in the divergences, the effect is like taking a \mathcal{D}^h contained in $\text{div } \mathcal{S}^h$, even if precise containment is not achieved. For the biquadratic displacement–bilinear pressure element, in many circumstances where there is a constant pressure mode there is a partner spurious mode [92]; when this happens it means that containment cannot hold either way between $\text{div } \mathcal{S}^h$ and \mathcal{D}^h . (The reader should try to prove this as an exercise.) The biquadratic displacement–bilinear pressure element and the bilinear displacement–constant pressure elements are sitting on the edge of over-constraining. In fact the spurious mode of the first kind, which often accompanies the global constant mode, can be seen to be symptomatic of this. If no boundary conditions are enforced, $\bar{\mathcal{G}}$ has no spurious mode. When, say homogeneous boundary conditions are enforced all around Ω , \mathcal{S}^h —and thus $\text{div } \mathcal{S}^h$ —are correspondingly reduced in dimension, but \mathcal{D}^h is not reduced in size. For the two elements under consideration, exactly two modes result: the checkerboard and the constant, which are precisely the modes that should have been removed from \mathcal{D}^h in a corresponding reduction in size. In general, modes of the first kind result because enforcing some essential boundary conditions engenders one or more incompressibility constraints, rendering redundant one or more of those imposed by \mathcal{D}^h . Modes of the second kind can arise because added degrees of freedom in excess of those required to force exact incompressibility are, of necessity, redundant. Modes are the inevitable partner of redundant constraints.

4.II.6 ERROR ESTIMATES AND PRESSURE SMOOTHING

We would like to be able to say that now that we have come to grips with modes, the way to error estimates is clear. The way to error estimates for safe elements is clear and has been for some time, but to say the same for some of the other elements discussed so far would amount to journalistic excess. The fact is that recent theoretical advances [85, 86, 90, 91, 93–95] have led to the establishment of error estimates for elements such as the bilinear displacement–constant pressure element, which at one time appeared to be a lost cause—in theory, at least [93, 95]. These results apply in the most generality to the bilinear displacement–constant pressure element, but require quite restrictive assumptions in other cases. Also these results have so far been established for homogeneous boundary conditions on all of Γ , which of course, are automatically consistent with all modes in the sense of Theorem 1. It appears to be an easy matter to extend these results to consistent inhomogeneous boundary conditions, but we are not aware that anyone has done so to date. Still, these results tend to inspire the authors to believe that more comprehensive estimates are possible. Even though the theoretical advances are limited in scope, when they are supported by a reinterpretation of computational experience in the light of what is now known about pressure modes, a coherent picture seems to emerge, which suggests that many more elements than just the safe ones can lead to convergent schemes. Now that we understand the role of the algebraic consistency condition imposed by modes, we know that Example 1 and other related puzzles uncovered in [83] do *not* mean that the elements are useless but mean that some essential boundary conditions need to be modified. In fact, spurious modes of the first kind are observed for some elements only on geometrically simple meshes, and these are precisely the sorts of meshes on which error estimates can be obtained by the new techniques. A detailed exposition of the new analysis is beyond the scope of the present discussion. That is probably just as well; it is often the case in mathematics that new discoveries are presented rather awkwardly at first and neat exposition results only after repeated reworking and refining. Here we would like to give a brief summary of an approach for obtaining estimates in the spirit of the new analysis, because there are some important practical consequences. The most important is that convergence of the pressures in the L_2 -norm cannot be guaranteed—even though it is often observed in practice. Because the pressures may not converge, we will propose that “smoothing” or “filtering” of the pressures be undertaken as a routine procedure when using all but the safe elements.

To make things simple and to stick to conditions for which estimates have been worked out rigorously, let us consider a simple elasticity problem—the problem of Sec. 4.2 with $\Gamma = \Gamma_g$ and $\mathbf{q} \equiv \mathbf{0}$. Some of the results require Ω to be a rectangle, but the general prescription does not put any restriction on the domain. Restrictions arise from trying to apply the prescription to specific elements, so we will mention them only when the need arises. The analysis we summarize here was developed by Malkus and Olsen [85, 86]. Much of it was inspired by a desire to synthesize the ideas found in [88, 89, 93, 94] in a way that relates those works to the more heuristic criteria for successful constrained finite elements that we have studied here. We wanted an error analysis that somehow recovered the notions of “overconstraining” and “underconstraining” and related those notions to actual convergence rates. A fuller

explanation of this synthesis may be found, as first presented, in [85] and, as later refined, in [86]. We will deal with the estimates for the Lagrange-multiplier method of Sec. 4.3, which leads to (4.3.21) with $M = \mathbf{0}$ in the discrete form. We assume that the integrations are exact for the matrices involved. Again, to apply our general prescription to specific elements, severe restrictions on the mesh are often required but will be mentioned only when necessary. In most, if not all, situations of practical import, the deviatoric part of the strain energy, $\bar{a}(\mathbf{u}, \mathbf{u})$ of (4.3.14) is positive-definite and bounded; that is, it satisfies (4.1.3). Thus the energy is equivalent to the $\|\cdot\|_1$ -norm and will be used in its place from here on. Whenever we use the symbol $\|\mathbf{u}\|_{\mathcal{E}}$, it will be understood to mean $\bar{a}(\mathbf{u}, \mathbf{u})^{1/2}$ in what follows.

The central idea in relating our more heuristic concept of constraint balance to the more-rigorous notion of convergence rate estimate is projection onto the subspace of weakly divergenceless trial functions, \mathbf{W}^h , given by

$$Z_h : H^1 \rightarrow \mathbf{W}^h \quad (4.II.18)$$

which assigns to each $\mathbf{u} \in H^1$ its (deviatoric) energy best approximation from \mathbf{W}^h . (H^s denotes the Sobolev spaces defined in Appendix 4.I; they consist of vector-valued functions when associated with displacements and scalar-valued functions when associated with pressures.) In the present case we mean projection in the sense of the Hilbert projection theorem—not projection onto \mathcal{S}^h as was discussed in Sec. 4.1, but projection onto the special subspace of \mathcal{S}^h whose members satisfy the weak constraint imposed by the Lagrange-multiplier method. We will argue that if we could estimate the accuracy, in the energy norm, of \mathbf{W}^h as an approximating space of functions, then we could estimate the accuracy of the finite element displacement solution. The functions that \mathbf{W}^h needs to approximate are possible exact solutions; as such, they are exactly incompressible and smoother than arbitrary functions in H^1 . A key point is that, under such circumstances, the estimates of the displacements can be uncoupled from the convergence rate of the pressure solution, which cannot be guaranteed. In [85], it is proved that if \mathbf{u} is the exact displacement solution, \mathbf{u}^h is the finite element solution, and p is the exact pressure solution (L_2 -orthogonal to constants), then

$$\bar{a}(Z_h \mathbf{u} - \mathbf{u}^h, Z_h \mathbf{u} - \mathbf{u}^h)^{1/2} \leq C \|q^h - p\|_0 \quad \text{for all } q^h \in \mathcal{P}^h \quad (4.II.19)$$

for a constant, C , which is equal to zero if $\mathcal{P}^h \supset \operatorname{div} \mathcal{S}^h$. Some points here bear strong emphasis:

1. The finite element pressure solution appears *nowhere* in (4.II.19) nor is its accuracy required to derive the result.
2. The arbitrariness of the q^h in the right-hand side of (4.II.19) means that it could be taken to be the nodal interpolate to p (should that exist) or the L_2 -best approximation to p , still retaining the inequality. Thus the right-hand side can be made as small as the pressures are accurate.

3. Equation (4.II.19) says that the finite element displacement solution is as close to being the energy best approximation from \mathbf{W}^h to the exact displacement solution as is constrained by the *accuracy* of the pressure trial space—except in case (b) or (c) of Corollary 5.1, when the finite element solution *is* that best approximation.
4. Equation (4.II.19) should be compared and contrasted to the results of Sec. 4.1, where it was found that the unconstrained finite element solution was the best energy approximation from all of \mathcal{S}^h to the exact solution: here we have an “almost best approximation” (best when $C = 0$) from a constrained subspace, \mathbf{W}^h .

By (4.II.19) and the triangle inequality,

$$\|u - u^h\|_E \leq \|u - Z_h u\|_E + C\|q^h - p\|_0 \quad \text{for all } q^h \in \mathcal{P}^h \quad (4.II.20)$$

So we have a displacement estimate if we could just estimate $\|u - Z_h u\|_E$. This is precisely what the constraint ratio was designed to give us an idea of, in its own humble way. The constraint ratio is a measure of whether \mathbf{W}^h has a big enough dimension to be an accurate constrained trial space. The ratio compares the dimension of \mathbf{W}^h to the dimension of a trial space assumed to be accurate— \mathcal{S}^h —and asks the question: Is the dimension of \mathbf{W}^h sufficiently large to be at least half the dimension of \mathcal{S}^h ? For heuristic reasons, it is assumed that the ratio one-half is optimal for two-dimensional problems because the exact incompressibility constraint seems to “use up” half the degrees of freedom of the exact solution, in that it makes one vector component linearly dependent on the other. The three-dimensional generalization is obvious. A delicate balance is implied by (4.II.19) and (4.II.20), as it was with our earlier more heuristic discussion in Sec. 4.3. The displacement solution can be only as accurate as the pressure solution allows, owing to a pressure accuracy term in the displacement estimate. *Pressure accuracy is what does the constraining.* But we cannot take arbitrarily accurate pressures for reasons we have already observed—too many pressures may overconstrain the problem for locking elements. Too many pressures cause many modes, which cannot sharpen the constraint after a certain point. Most importantly, the best that can be done is to force $V^h \equiv \mathbf{W}^h$, which can only work for a nonlocking element; this does not improve the accuracy of the displacement solution beyond the corresponding estimate for $\|u - Z_h u\|_E$, no matter how many pressures we pour on. We will not be able to discuss it in detail here, but error estimates for the smoothed pressures contain a displacement *solution* accuracy term, so pouring on the pressures cannot produce a more-accurate pressure solution, no matter how accurate the pressure trial space has the potential to be. By the way, this last remark points out the beauty of this kind of analysis: It gives an estimate for the displacements first, then the pressures can be considered, with a displacement estimate already in hand. All of this presumes that we can indeed estimate $\|u - Z_h u\|_E$.

Before we go on to discuss ways of estimating the accuracy of \mathbf{W}^h as an approximating space, we pause to sharpen our notion of over- and underconstrained elements. Our base accuracy level will be the accuracy in energy of \mathcal{S}^h as an approximating space for approximation of all—not just incompressible—fields, \mathbf{u} such that $\|\mathbf{u}\|_{k+1}$ is finite, with k as in Sec. 4.1. This level of accuracy is thus $O(h^k)$ and is what will be referred to as “optimal”.

Definition. (Constrained Optimality) A finite element for incompressible media—displacement and pressure taken into consideration—will be called ***underconstrained*** if the pressure trial space is less than $O(h^k)$ accurate in approximating pressures for which $\|p\|_k$ is finite. The element will be called ***overconstrained*** if $\|\mathbf{u} - Z_h \mathbf{u}\|_E$ is less than $O(h^k)$ accurate, no matter how smooth \mathbf{u} is in terms of the Sobolev norm. An element will be called ***optimally constrained*** if it is neither over- nor underconstrained.

Equations (4.II.19) and (4.II.20) imply that optimally constrained elements have optimal error estimates, at least as far as convergence rate is concerned. Some specific error estimates require a higher than optimal degree of smoothness of the exact displacement solution to get an optimal convergence rate. The most recent estimates for the bilinear displacement–constant pressure element suggests that this extra smoothness requirement is an artifact of proof technique, as opposed to a genuine restriction. Compare the refined results of [95], which require no such extra smoothness, to the earlier proofs for that element which did require it [93]. However, the extra smoothness is still formally required to get estimates for other elements; this is the reason for the terminology “no matter how smooth” in the definition of overconstrained. We want to allow elements that require extra smoothness to show that $\|\mathbf{u} - Z_h \mathbf{u}\|_E$ is optimal to be counted as optimally constrained.

Examples of optimally constrained elements are bilinear displacement–constant pressure elements and linear crossed triangles. For the former, the techniques of [93] are refined in [86] and applied to rectangular elements “rectangulating” a domain, which is essentially composed of unions of rectangles, in a grid-type called “rectangularly regular.” For the second element, it is shown that for meshes of rectangular elements, whenever the bilinear displacement–constant pressure element is optimal, so is the crossed triangle element. Since the bilinear displacement–constant pressure element estimate of [93] requires the exact displacement solution to be in H^3 , so does the crossed triangle estimate of [86]. The newer estimates of [95] for the bilinear element require that the exact solution be only in H^2 , and this translates directly into a similar estimate for the crossed triangles on rectangularly regular meshes, and most probably on more complicated ones. The delicate matter in showing this optimality is that neither of the two elements satisfy the LBB-condition [85, 86, 90]. This was proposed in its most cogent form by Brezzi [84], and Fortin [82] was the first to recognize that the bilinear rectangle did not satisfy it. The condition in most practical situations requires that there be no pressure modes—other than the global constant, when appropriate—and that the operator represented by $GQ^{-1}G^T$ be a uniformly invertible operator when restricted to the energy orthocomplement of the incompressible trial functions. Such a restriction always

makes the operator represented by $\mathbf{G}\mathbf{Q}^{-1}\mathbf{G}^T$ algebraically invertible, but the notion of uniformity means that the energy norm of its restricted inverse is bounded as the mesh is refined. This can be determined from a relatively simple eigenvalue problem [85, 96], which can actually be solved numerically to find out whether an element satisfies the LBB condition on a given sequence of meshes. One of the observations that was at first most puzzling to those trying to untangle this whole business was that it is possible to have boundary conditions for which, say, the bilinear displacement–constant pressure element would have no modes, yet the uniformity requirement would not be met. On other sequences of meshes with the same boundary conditions, numerical evidence could be interpreted to say only that there were no modes *and* the uniformity condition was met. A convergence theory that could hold with or without the LBB condition seemed to be required. If an element satisfies the LBB condition, then it is automatically not overconstrained, because an immediate consequence of the LBB condition is that $\|\mathbf{u} - \mathbf{Z}_h\mathbf{u}\|_E$ is optimal, with no requirement of extra smoothness on \mathbf{u} [82, 84–86]. All the elements that we have enumerated as “safe” satisfy the LBB condition (or a subtle variant of it, which can take its place for continuous pressure fields). Indeed all the elements flagged as satisfying the LBB condition in Figs. 4.3.4 and 4.3.6 satisfy the condition or its variant and have no spurious modes. So indeed, elements that satisfy the LBB condition can be, and often are, underconstrained. We have already discussed that matter thoroughly.

We close this brief discussion of displacement estimates for the simple elasticity problem by observing that nonlocking elements can be either overconstrained, underconstrained, or optimal, depending on the choice of \mathcal{D}^h . One class of displacement elements can serve to illustrate all the possibilities: quadratic triangles. With piecewise constant pressures, they satisfy the LBB condition and are underconstrained (Fig. 4.3.4(j)); with pure linear, discontinuous pressures, they do not satisfy LBB, are exactly incompressible, and in arrangements without many redundant constraints (like Fig. 4.3.4(i)) are overconstrained and have six modes (five spurious) in the simple elasticity problem on a square domain with square elements [85]. Crossing the quadratic triangles, as with linear crossed triangles, gives an element that is optimally constrained but has one mode of the second kind for each macroelement and an unknown number of the first kind [88]. Finally, as observed earlier, using a continuous linear pressure field with quadratic triangles (Fig. 4.3.6(d)) yields an optimal element with no spurious modes and convergent pressures.

At last, we close this section with the topic that is least clear to us at the time of this writing—pressure smoothing. There is no big picture we know of that gives a heuristic guide to what should work and what will not. In many problems with exact displacement solutions in $H^{k+1}(\Omega)$, optimal convergence to the “raw” pressure representative, p , of Corollary 1.1 is observed, in spite of the fact that the uniformity requirement of the LBB condition is not met. It is not easy to see how this can be, since the best the estimates of [85, 86, 90, 93–95] can say is that the blowup of the norm of the inverse of the relevant operator is multiplying the error term for the finite element pressure solution. We do have examples of problems in which the exact displacement solution is less smooth than $H^{k+1}(\Omega)$, so that a loss of convergence rate due to a singularity in the exact solution would be predicted by standard analysis, and

where an apparent nonconvergence of the pressures is observed [85]. We also know that the only way a pressure error estimate can be obtained for elements that do not satisfy the LBB condition is by using some kind of pressure “filtering” or “smoothing.” This amounts to approximating the raw pressure computed by solving the Lagrange-multiplier or penalty problem by projecting it into an auxiliary pressure trial space, call it \mathcal{X}^h . The idea is to choose \mathcal{X}^h so that the L_2 -projection operator smooths out the spurious oscillations in the raw pressures. These oscillations may be components of spurious modes, and in many smoothing schemes, \mathcal{X}^h is chosen so that spurious modes in \mathcal{P}^h are projected onto the zero member of \mathcal{X}^h . This would have the effect of wiping out the q -component of Corollary 1.1. But theoretically and in practice, at least for the problems with rougher exact solutions, there is more to project out than modes. This can be fairly certainly deduced, because oscillation of penalty pressures can be observed in these rougher problems [85, 86], and penalty pressures cannot be infected with a mode component, according to Theorem 4. Presumably, the oscillations of penalty pressures can only be due to the failure of the uniformity portion of the LBB condition. If the LBB condition is satisfied, the raw pressures converge, and there is no need for any smoothing. We know that it is possible in theory to choose \mathcal{X}^h so that it fulfills all the remarkable requirements outlined here, because for some elements and meshes proofs to that effect have been devised [85, 86, 93–95]. The proofs we know of work only with even numbers of elements per side, and \mathcal{X}^h must be composed of 2×2 composite quadrilaterals. This is hardly a desirable choice of pressure element from the computational point of view, since it mixes global and element level computations. There is an abstract condition, a kind of generalized LBB condition [85], which can be applied to \mathcal{X}^h and can predict when a smoothing scheme should work, but it is so hard to prove that it has been successfully applied only in very restrictive circumstances [85].

We and others have found, through numerical experimentation, that many smoothing schemes will work fairly well, in spite of our inability to prove anything about them. The one we have come to favor is the one proposed in [97] and described in Sec. 4.4.1. It was proposed long before we had any idea of all the subtleties that have recently emerged. This is the choice of \mathcal{X}^h as a continuous pressure field based on the displacement element nodes. In the case of linear crossed triangles, the continuous pressure field \mathcal{X}^h is based on the macroelement corner nodes [85, 90]. This scheme has the advantage of referring pressures to displacement nodes, so that both quantities are referred to the same spatial locations. The pressure approximation is continuous, so that some of the good aspects of incompressible elements with continuous pressure fields are recovered by the smoothing scheme. It should be pointed out that since the smoothed pressure can, at best, be expected to converge to the exact pressure in L_2 , there is no requirement for convergence at given points. These schemes leave remnants of the oscillations at completely unshared nodes (domain corners), and the pressures in the strip of elements nearest to boundaries are less accurate than could be hoped [85, 90, 97]. There are ways to correct the boundary glitches, which are described in Sec. 4.4.1.

So we have come to the end of this not-so-brief discussion of the subtleties involved in the choice of finite elements for problems with the incompressibility

constraint. We have presented an overview of the subject, but we cannot hope to be definitive at this point because this area is one in which new ideas are rapidly developing, and old ideas are being explained in ever simpler and more unified ways. The reader with a strong background and interest in functional analysis might prepare himself or herself to keep abreast of these developments by further reading. We particularly recommend study of [82, 84–86, 91, 93, 95], which provide a deeper analysis of the problems discussed here.

REFERENCES

Section 4.1

1. M. M. Vainberg, *Variational Methods for the Study of Nonlinear Operators*. San Francisco: Holden-Day, 1964.
2. K. Washizu, *Variational Methods in Elasticity and Plasticity*. Oxford: Pergamon Press, 1968.
3. J. T. Oden and J. N. Reddy, *Variational Methods in Theoretical Mechanics*. Heidelberg: Springer-Verlag, 1976.
4. G. Duvaut and J. L. Lions, *Les Inéquations en Mécanique et en Physique*. Paris: Dunod, 1972.
5. G. Strang and G. Fix, *An Analysis of the Finite Element Method*. Englewood Cliffs, N. J.: Prentice-Hall, 1973.
6. P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*. Amsterdam: North-Holland, 1978.
7. J. T. Oden and G. F. Carey, *Finite Elements: Mathematical Aspects*, Vol. IV. Englewood Cliffs, N. J.: Prentice-Hall, 1983.

Section 4.2

8. I. S. Sokolnikoff, *Mathematical Theory of Elasticity* (2nd ed.). New York: McGraw-Hill, 1956.

Section 4.3

9. L. R. Herrmann, "Elasticity Equations for Nearly Incompressible Materials by a Variational Theorem," *AIAA J.*, 3 (1965), 1896–1900.
10. R. L. Taylor, K. S. Pister, and L. R. Herrmann, "On a Variational Theorem for Incompressible and Nearly Incompressible Orthotropic Elasticity," *International Journal of Solids and Structures*, 4 (1968), 875–883.
11. S. W. Key, "A Variational Principle for Incompressible and Nearly Incompressible Anisotropic Elasticity," *International Journal of Solids and Structures*, 5 (1969), 951–964.
12. R. L. Sani, P. M. Gresho, R. L. Lee, D. F. Griffiths, and M. Engleman, "The Cause and Cure (?) of the Spurious Pressures Generated by Certain FEM Solutions of the Navier-Stokes Equations, Parts I and II," *International Journal for Numerical Methods in Fluids*, 1 (1981), 17–43 and 171–204.

13. O. C. Zienkiewicz, *The Finite Element Method*. London: McGraw-Hill, 1977.
14. J. C. Nagtegaal, D. M. Parks, and J. R. Rice, "On Numerically Accurate Finite Element Solutions in the Fully Plastic Range," *Computer Methods in Applied Mechanics and Engineering*, 4 (1974), 153–178.
15. J. H. Argyris, P. C. Dunne, T. Angelopoulos, and B. Bichat, "Large Natural Strains and Some Special Difficulties Due to Nonlinearity and Incompressibility in Finite Elements," *Computer Methods in Applied Mechanics and Engineering*, 4 (1974), 219–278.
16. D. S. Malkus and T. J. R. Hughes, "Mixed Finite Element Methods—Reduced and Selective Integration Techniques: A Unification of Concepts," *Computer Methods in Applied Mechanics and Engineering*, 15, no. 1 (1978), 68–81.
17. T. J. R. Hughes and H. Allik, "Finite Elements for Compressible and Incompressible Continua," *Proceedings of the Symposium on Civil Engineering*, Vanderbilt University, Nashville, Tenn. (1969), 27–62.
18. E. G. Thompson, "Average and Complete Incompressibility in the Finite Element Method," *International Journal for Numerical Methods in Engineering*, 9 (1975), 925–932.
19. M. Crouzeix and P. A. Raviart, "Conforming and Nonconforming Finite Element Methods for Solving the Stationary Stokes Equation," *Revue Française d'Automatique Informatique et Recherche Opérationnelle*, R-3 (1973), 33–76.
20. V. Ruas, "A Class of Asymmetric Simplicial Finite Element Methods for Solving Finite Incompressible Elasticity Problems," *Computer Methods in Applied Mechanics and Engineering*, 27 (1981), 319–343.
21. M. Bercovier and O. Pironneau, "Error Estimates for Finite Element Solutions of the Stokes Problem in Primitive Variables," *Numerische Mathematik*, 33 (1979), 211–224.
22. F. Thomasset, *Implementation of Finite Element Methods for Navier-Stokes Equations*. New York: Springer-Verlag, 1981.
23. M. Fortin, "Old and New Finite Elements for Incompressible Flows," *International Journal for Numerical Methods in Fluids*, 1 (1981) 347–364.
24. D. F. Griffiths, "Finite Elements for Incompressible Flows," *Mathematical Methods in Applied Science*, 1 (1979), 16–31.
25. D. F. Griffiths, "The Construction of Approximately Divergence-free Finite Elements," *Mathematics of Finite Elements and Applications III*, ed. J. Whiteman. London: Academic Press, 1979, 239–245.
26. D. F. Griffiths, "An Approximately Divergence-free 9-node Velocity Element (with Variations) for Incompressible Flows," *International Journal for Numerical Methods in Fluids*, 1 (1981), 232–346.
27. D. F. Griffiths, "The Effect of Pressure Approximations on Finite Element Calculations of Incompressible Flows," in *Numerical Methods for Fluid Dynamics*, eds. K. W. Morton and M. J. Baines. London: Academic Press, 1982.
28. J. T. Oden and O. Jacquotte, "A Stable Second-Order Accurate Finite Element Scheme for the Analysis of Two-Dimensional Incompressible Viscous Flows," in *Proceedings of the Fourth International Symposium on Finite Element Methods in Flow Problems*, ed. T. Kawai, Chuo University, Tokyo, July 26–29, 1982.
29. J. Boland and R. Nicolaides, "Stability of Finite Elements Under Divergence Constraints," *SIAM Journal of Numerical Analysis*, 20, no. 4 (1983), 722–731.
30. J. T. Oden and G. F. Carey, *Finite Elements: Mathematical Aspects*, Vol. IV. Englewood Cliffs, N. J.: Prentice-Hall, 1984.

31. C. Johnson and J. Pitkäranta, "Analysis of Some Mixed Finite Element Methods Related to Reduced Integration," *Mathematics of Computation*, 38 (1982), 375–400.
32. J. Boland and R. Nicolaides, "Stable and Semistable Low Order Finite Elements for Viscous Flow," *SIAM Journal of Numerical Analysis*, 22 (1985), 474–492.
33. J. Pitkäranta and R. Stenberg, "Error Bounds for the Approximation of the Stokes Problem Using Bilinear/Constant Elements on Irregular Quadrilateral Meshes," Report-MAT-A222, Helsinki University of Technology, Institute of Mathematics, Finland, 1984.
34. F. Brezzi and J. Pitkäranta, "On the Stabilization of Finite Element Approximations of the Stokes Equations," Report-MAT-A219, Helsinki University of Technology, Institute of Mathematics, Finland, 1984.
35. T. J. R. Hughes, L. P. Franca, and M. Balestra, "Circumventing the Babuška-Brezzi Condition: A Stable Petrov-Galerkin Formulation of the Stokes Problem Accommodating Equal-Order Interpolation," *Computer Methods in Applied Mechanics and Engineering*, 59(1986), 85–99.

Section 4.4

36. D. S. Malkus and T. J. R. Hughes, "Mixed Finite Element Methods—Reduced and Selective Integration Techniques: A Unification of Concepts," *Computer Methods in Applied Mechanics and Engineering*, 15, no. 1 (1978), 63–81.
37. T. J. R. Hughes, "Equivalence of Finite Elements for Nearly Incompressible Elasticity," *Journal of Applied Mechanics*, 44 (1977), 181–183.
38. I. Fried, "Finite Element Analysis of Incompressible Material by Residual Energy Balancing," *International Journal of Solids and Structures*, 10 (1974), 993–1002.
39. T. J. R. Hughes and H. Allik, "Finite Elements for Compressible and Incompressible Continua," *Proceedings of the Symposium on Civil Engineering*. Vanderbilt University, Nashville, Tenn. (1969), 27–62.
40. J. C. Nagtegaal, D. M. Parks, and J. R. Rice, "On Numerically Accurate Finite Element Solutions in the Fully Plastic Range," *Computer Methods in Applied Mechanics and Engineering*, 4 (1974), 153–178.
41. B. Mercier, "A Conforming Finite Element Method for Two-Dimensional Incompressible Elasticity," *International Journal for Numerical Methods in Engineering*, 14, no. 6 (1979), 942–945.
42. B. Fraeijs de Veubeke, "Displacement and Equilibrium Models in the Finite Element Method," in *Stress Analysis*, eds. O. C. Zienkiewicz and G. S. Holister. London: John Wiley, 1965.
43. L. R. Herrmann, "Elasticity Equations for Incompressible and Nearly Incompressible Materials by a Variational Theorem," *AIAA J.*, 3 (1965), 1896–1900.
44. O. C. Zienkiewicz, R. L. Taylor, and J. M. Too, "Reduced Integration Technique in General Analysis of Plates and Shells," *International Journal for Numerical Methods in Engineering*, 3 (1971), 275–290.
45. D. J. Naylor, "Stresses in Nearly Incompressible Materials by Finite Elements with Application to the Calculation of Excess Pore Pressures," *International Journal for Numerical Methods in Engineering*, 8 (1974), 443–460.
46. O. C. Zienkiewicz and P. N. Godbole, "Viscous Incompressible Flow with Special Reference to Non-Newtonian (Plastic) Fluids," in *Finite Element Methods in Fluids*, Vol. 1. London: John Wiley, 1975.

47. W. P. Doherty, E. L. Wilson, and R. L. Taylor, "Stress Analysis of Axisymmetric Solids Utilizing Higher Order Quadrilateral Finite Elements," SESM Report No. 69-3, Department of Civil Engineering, University of California, Berkeley, 1969.
48. J. H. Argyris, P. C. Dunne, T. Angelopoulos, and B. Bichat, "Large Natural Strains and Some Special Difficulties Due to Nonlinearity and Incompressibility in Finite Elements," *Computer Methods in Applied Mechanics and Engineering*, 4 (1974), 219–278.
49. D. S. Malkus, "Finite Element Analysis of Incompressible Solids," Ph.D. Thesis, Boston University, Boston (1975).
50. D. S. Malkus, "A Finite Element Displacement Model Valid for Any Value of the Compressibility," *International Journal of Solids and Structures*, 12 (1976), 731–738.
51. E. Hinton, "Least Squares Analysis Using Finite Elements," M.Sc.Thesis, Civil Engineering Department, University of Wales, Swansea, 1968.
52. R. L. Lee, P. M. Gresho, and R. L. Sani, "Numerical Smoothing Techniques Applied to Some Finite Element Solutions of the Navier-Stokes Equations," Second International Conference on Finite Elements in Water Resources, London, England, July 10–14, 1978. See also: "Smoothing Techniques for Certain Primitive Variable Solutions of the Navier-Stokes Equations," *International Journal for Numerical Methods in Engineering*, 14 (1979), 1785–1804.
53. T. J. R. Hughes, W. K. Liu, and A. Brooks, "Review of Finite Element Analysis of Incompressible Viscous Flows by the Penalty Function Formulation," *Journal of Computational Physics*, 30, no. 1 (1979), 1–60.
54. J. Barlow, "Optimal Stress Locations in Finite Element Models," *International Journal for Numerical Methods in Engineering*, 10 (1976), 243–251.

Section 4.5

55. J. C. Nagtegaal, D. M. Parks, and J. R. Rice, "On Numerically Accurate Finite Element Solutions in the Fully Plastic Range," *Computer Methods in Applied Mechanics and Engineering*, 4 (1974), 153–178.
56. R. L. Taylor, K. S. Pister, and L. R. Herrmann, "On a Variational Theorem for Incompressible and Nearly-Incompressible Orthotropic Elasticity," *International Journal of Solids and Structures*, 4 (1968), 875–883.
57. S. W. Key, "A Variational Principle for Incompressible and Nearly Incompressible Anisotropic Elasticity," *International Journal of Solids and Structures*, 5 (1969), 951–964.
58. T. J. R. Hughes, "Generalization of Selective Integration Procedures to Anisotropic and Nonlinear Media," *International Journal for Numerical Methods in Engineering*, 15 (1980), 1413–1418.
59. T. J. R. Hughes and D. S. Malkus, "A General Penalty/Mixed Equivalence Theorem for Anisotropic, Incompressible Finite Elements," in *Hybrid and Mixed Finite Element Methods*, eds. S. N. Atluri, R. H. Gallagher, and O. C. Zienkiewicz. London: John Wiley, 1983, 487–496.
60. T. J. R. Hughes, "Recent Developments in Computer Methods for Structural Analysis" *Nuclear Engineering and Design*, 57, no. 2 (1980), 427–439.
61. J. C. Simo and T. J. R. Hughes, "On the Variational Foundations of Assumed Strain Methods," *Journal of Applied Mechanics*, 53, no. 1 (1986), 51–54.

62. T. J. R. Hughes, J. C. Simo, T. Belytschko, and H. Stolarski, "Foundations of Assumed Strain Methods in Finite Element Analysis," *Computer Methods in Applied Mechanics and Engineering*, to appear.

Section 4.6

63. G. P. Bazeley, Y. K. Cheung, B. M. Irons, and O.C. Zienkiewicz, "Triangular Elements in Plate Bending—Conforming and Nonconforming Solutions," *Proceedings of the First Conference on Matrix Methods in Structural Mechanics*, Wright-Patterson AFB, Ohio, 1965.
64. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*. Englewood Cliffs, N. J.: Prentice-Hall, 1973.
65. F. Stummel, "The Limitations of the Patch Test," *International Journal for Numerical Methods in Engineering*, 15 (1980), 177–188.
66. B. Irons and M. Loikkanen, "An Engineer's Defense of the Patch Test," *International Journal for Numerical Methods in Engineering*, 19, no. 9 (1983), 1391–1401.
67. R. L. Taylor, J. C. Simo, O. C. Zienkiewicz, and A. C. Chan, "The Patch Test: A Condition for Assessing Finite Element Convergence," *International Journal for Numerical Methods in Engineering*, 22, no. 1 (1986), 39–62.

Section 4.7

68. E. L. Wilson, R. L. Taylor, W. P. Doherty, and J. Ghaboussi, "Incompatible Displacement Models," in *Numerical and Computer Models in Structural Mechanics*, eds. S. J. Fenves, N. Perrone, A. R. Robinson, and W. C. Schnobrich. New York: Academic Press, 1973, 43–57.
69. P. Lesaint, "On the Convergence of Wilson's Non-conforming Element for Solving the Elastic Problem," *Computer Methods in Applied Mechanics and Engineering*, 7 (1976), 1–16.
70. R. L. Taylor, P. J. Beresford, and E. L. Wilson, "A Nonconforming Element for Stress Analysis," *International Journal for Numerical Methods in Engineering*, 10, no. 6 (1976), 1211–1219.
71. F. Thomasset, *Implementation of Finite Element Methods for the Navier-Stokes Equations*. New York: Springer-Verlag, 1981.

Section 4.8

72. D. Kosloff and G. A. Frazier, "Treatment of Hourglass Patterns in Low Order Finite Element Codes," *Numerical and Analytical Methods in Geomechanics* 2 (1978), 57–72.
73. D. P. Flanagan and T. Belytschko, "A Uniform Strain Hexahedron and Quadrilateral with Orthogonal Hourglass Control," *International Journal for Numerical Methods in Engineering*, 17 (1981), 679–706.
74. T. Belytschko, J. S-J. Ong, W. K. Liu, and J. M. Kennedy, "Hourglass Control in Linear and Nonlinear Problems," *Computer Methods in Applied Mechanics and Engineering*, 43 (1984), 251–276.

Section 4.9

75. W. P. Doherty, E. L. Wilson, and R. L. Taylor, "Stress Analysis of Axisymmetric Solids Utilizing Higher Order Quadrilateral Finite Elements," SESM Report 69-3, Department of Civil Engineering, University of California, Berkeley, 1969.
76. K. Kavanagh and S. W. Key, "A Note on Selective and Reduced Integration Techniques in the Finite Element Method," *International Journal for Numerical Methods in Engineering*, 4, no. 1 (1972) 148–150.

Appendix 4.I

77. P. G. Ciarlet and P. A. Raviart, "General Lagrange and Hermite Interpolation in \mathbb{R}^n with Application to the Finite Element Method," *Archive for Rational Mechanics and Analysis*, 46 (1972), 177–199.
78. P. G. Ciarlet and P. A. Raviart, "Interpolation Theory over Curved Elements," *Computer Methods in Applied Mechanics and Engineering*, 1, (1972), 217–249.
79. J. T. Oden and J. N. Reddy, *The Mathematical Theory of Finite Elements*. New York: Wiley-Interscience, 1976.
80. S. G. Mikhlin, *Variational Methods in Mathematical Physics*. Oxford; Pergamon Press, 1964.
81. G. Fichera, "Existence Theorems in Elasticity," in *Mechanics of Solids II*, Vol. VIa/2, *Encyclopedia of Physics*. Berlin: Springer-Verlag, 1972.

Appendix 4.II

82. M. Fortin, "An Analysis of the Convergence of Mixed Finite Element Methods," *Revue Française d'Automatique Informatique et Recherche Opérationnelle*, 11 (1977), 341–354.
83. R. L. Sani, P. M. Gresho, R. L. Lee, D. F. Griffiths, and M. Engelman, "The Cause and Cure (?) of the Spurious Pressures Generated by Certain FEM Solutions to the Navier-Stokes Equations, Parts I and II," *International Journal for Numerical Methods in Fluids*, 1 (1981), 17–43 and 171–204.
84. F. Brezzi, "On the Existence, Uniqueness and Approximation of Saddle Point Problems Arising From Lagrangian Multipliers," *Revue Française d'Automatique Informatique et Recherche Opérationnelle*, 8 (1974), 129–151.
85. D. S. Malkus and E. T. Olsen, "Obtaining Error Estimates for Optimally Constrained Finite Elements," *Computer Methods in Applied Mechanics and Engineering*, 42 (1984), 331–353.
86. E. T. Olsen, "Stable Finite Elements for Non-Newtonian Flows; First-Order Elements which Fail the LBB Condition," Ph.D. Thesis, Illinois Institute of Technology, Chicago, August, 1983.
87. J. C. Nagtegaal, D. M. Parks, and J. R. Rice, "On Numerically Accurate Finite Element Solutions in the Fully Plastic Range," *Computer Methods in Applied Mechanics and Engineering*, 4 (1974), 153–178.
88. B. Mercier, "A Conforming Finite Element for Two-Dimensional Incompressible Elasticity," *International Journal for Numerical Methods in Engineering*, 14, no. 6 (1979), 942–945.
89. B. Mercier, *Topics in the Finite Element Solution of Elliptic Problems*, Tata Institute of Fundamental Research Lecture Series. Berlin: Springer-Verlag, 1979.

90. B. Bernstein, D. S. Malkus, and E. T. Olsen, "A Finite Element for Incompressible Plane Flows of Fluids with Memory," *International Journal for Numerical Methods in Fluids*, 5 (1985), 43–70.
91. L. R. Scott and M. Vogelius, "Norm Estimates for a Maximal Right Inverse of the Divergence Operator in Spaces of Piecewise Polynomials," Technical Note BN-1013, Institute for Physical Science and Technology, University of Maryland, November, 1983.
92. M. Engelman, R. L. Sani, P. M. Gresho, and M. Bercovier, "Consistent vs. Reduced Integration Penalty Methods for Incompressible Media Using Several Old and New Elements," *International Journal for Numerical Methods in Fluids*, 2 (1981), 25–42.
93. C. Johnson and J. Pitkäranta, "Analysis of Some Mixed Methods Related to Reduced Integration," *Mathematics of Computation*, 38 (1982), 375–400.
94. J. T. Oden, N. Kikuchi, and Y. J. Song, "Penalty Finite Element Methods for Stokesian Flows," *Computational Methods in Applied Mechanics and Engineering*, 31 (1982), 297–329.
95. J. Pitkäranta and R. Stenberg, "Error Bounds for the Approximation of the Stokes Problem Using Bilinear/Constant Elements on Irregular Quadrilateral Meshes," Report MAT-A222, Helsinki University of Technology, Institute of Mathematics, Finland, 1984.
96. D. S. Malkus, "Eigenproblems Associated With the Discrete LBB Condition for Incompressible Finite Elements," *International Journal of Engineering Science*, 19 (1981), 1299–1310.
97. T. J. R. Hughes, W. K. Liu, and A. Brooks, "Review of Finite Element Analysis of Incompressible Viscous Flows by the Penalty Function Formulation," *Journal of Computational Physics*, 30, no. 1 (1979), 1–60.

5

The C^0 -Approach to Plates and Beams

5.1 INTRODUCTION

The classical Poisson-Kirchhoff theory of plates requires C^1 -continuity, just as does the classical Bernoulli-Euler beam theory (see Sec. 1.16). Continuous (i.e., C^0) finite element interpolations are easily constructed. The same cannot be said for multi-dimensional C^1 -interpolations. It has taken considerable ingenuity to develop compatible C^1 -interpolation schemes for two-dimensional plate elements based on classical theory, and the resulting schemes have always been extremely complicated in one way or another.

More and more, there is a turning away from Poisson-Kirchhoff type elements to elements based upon theories which accommodate transverse shear strains (Reissner and Mindlin theories) and require only C^0 -continuity. This approach opens the way to a greater variety of interpolatory schemes but is not without its own inherent difficulties. Recently, displacement-type elements have been derived based upon Reissner-Mindlin theory, which seem to be superior to plate elements derived heretofore. This chapter discusses the basic techniques and considerations involved and summarizes recent developments in this area.

Following this, a similar approach is discussed in the context of beams' and frames in which transverse shearing strains are accounted for. This also proves to be extremely simple and effective.

5.2 REISSNER-MINDLIN PLATE THEORY

5.2.1 Main Assumptions

All quantities are referred to a fixed system of rectangular, Cartesian coordinates. A

general point in this system is denoted by (x_1, x_2, x_3) or (x, y, z) , whichever is more convenient. Throughout, Latin and Greek indices take on the values 1, 2, 3 and 1, respectively.

The main assumptions of the plate theory are

1. The domain Ω is of the following special form:

$$\Omega = \{(x, y, z) \in \mathbb{R}^3 \mid z \in \left[-\frac{t}{2}, \frac{t}{2} \right], (x, y) \in A \subset \mathbb{R}^2\}$$

where t is the plate thickness and A is its area. The boundary of A is denoted by

2. $\sigma_{33} = 0$.
3. $u_\alpha(x, y, z) = -z\theta_\alpha(x, y)$.
4. $u_3(x, y, z) = w(x, y)$.

Remarks

1. In Assumption 1, we may take the plate thickness t to be a function of x and y , if desired.

2. Assumption 2 is the plane stress hypothesis. It contradicts Assumption 4 but ultimately causes no problem. The justification of the present theory is its usefulness in practical structural engineering applications. No plate theory is completely consistent with the three-dimensional theory and, at the same time, both simple and useful. Assumption 2 is to be substituted into the constitutive equation; ϵ_{33} is to be solved for and subsequently eliminated.

3. Assumption 3 implies that plane sections remain plane. θ_α is interpreted as the rotation of a fiber initially normal to the plate midsurface (i.e., $z = 0$).

4. By Assumption 4, the transverse displacement, w , does not vary through the thickness.

The sign convention is illustrated in Fig. 5.2.1. “Right-hand-rule” rotations $\hat{\theta}_\alpha$ are defined by $\theta_\alpha = -e_{\alpha\beta}\hat{\theta}_\beta$, where $e_{\alpha\beta}$ is the alternator tensor, viz.,

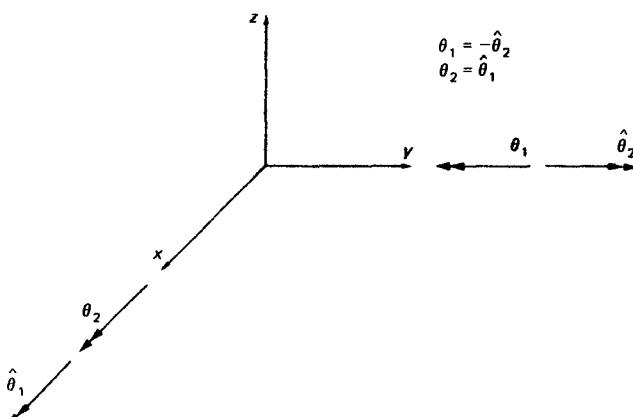


Figure 5.2.1 Sign conventions for rotations. $\hat{\theta}_1, \hat{\theta}_2$ are right-hand-rule rotations; θ_1, θ_2 are rotations that simplify the development of the plate theory.

$$[e_{\alpha\beta}] = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad (5.2.1)$$

We prefer to develop the theory in terms of θ_α rather than $\hat{\theta}_\alpha$ because the algebra is greatly simplified, due to the absence of alternator tensors. In typical structural analysis computer programs, the right-hand-rule convention is usually, but not always, adopted. Consequently, it is the responsibility of the analyst to determine which convention is being employed. It is common when analysts use a new program that errors are made because of lack of careful attention to this point. *Whenever a plate or shell analysis is being undertaken the analyst should check the rotation–bending moment sign convention of the computer program being used before embarking upon that analysis.*

Plate kinematics are summarized in Fig. 5.2.2.

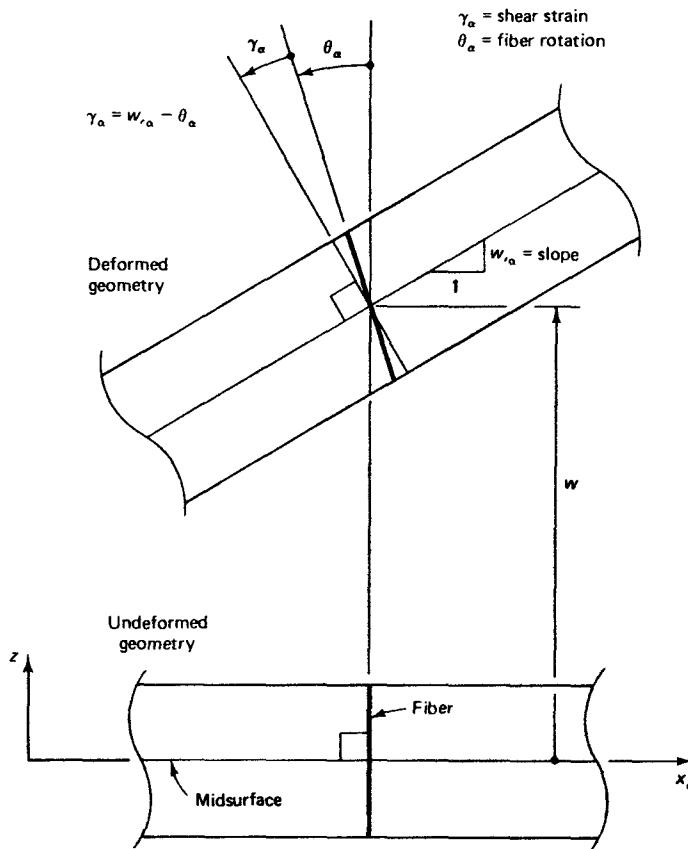


Figure 5.2.2 Plate kinematics. Transverse shear strains need not vanish in the present theory.

5.2.2 Constitutive Equation

The reduced form of the constitutive equation used in the plate theory is determined by substituting Assumption 2 into the three-dimensional constitutive equation and eliminating ϵ_{33} . For simplicity, we shall consider the isotropic case in which

$$\sigma_{ij} = \lambda \delta_{ij} \epsilon_{kk} + 2\mu \epsilon_{ij} \quad (5.2.2)$$

where λ and μ are the Lamé coefficients and δ_{ij} is the Kronecker delta. Assumption 2 implies

$$\epsilon_{33} = \frac{-\lambda}{\lambda + 2\mu} \epsilon_{aa} \quad (5.2.3)$$

$$\boxed{\sigma_{\alpha\beta} = \bar{\lambda} \delta_{\alpha\beta} \epsilon_{\gamma\gamma} + 2\mu \epsilon_{\alpha\beta}} \quad (5.2.4)$$

$$\boxed{\sigma_{\alpha 3} = 2\mu \epsilon_{\alpha 3}} \quad (5.2.5)$$

where

$$\bar{\lambda} = \frac{2\lambda\mu}{\lambda + 2\mu} \quad (5.2.6)$$

$\bar{\lambda}$ and μ may be eliminated in favor of E and ν (Young's modulus and Poisson's ratio, respectively):

$$\bar{\lambda} = \frac{\nu E}{1 - \nu^2} \quad (5.2.7)$$

$$\mu = \frac{E}{2(1 + \nu)} \quad (5.2.8)$$

5.2.3 Strain-displacement Equations

Assumptions 3 and 4 lead to the following form of the strain-displacement equations:

$$\epsilon_{\alpha\beta} = u_{(\alpha, \beta)} = -z\theta_{(\alpha, \beta)} \quad (5.2.9)$$

$$\epsilon_{\alpha 3} = u_{(\alpha, 3)} = \frac{-\theta_\alpha + w_{,\alpha}}{2} \quad (5.2.10)$$

Note that the normal-fiber rotation (i.e., θ_α) and slope (i.e., $w_{,\alpha}$) are not necessarily the same and thus transverse shear strains are accommodated. This is to be contrasted with *classical Poisson-Kirchhoff* (i.e., “thin plate”) theory in which $\theta_\alpha = w_{,\alpha}$ and, consequently, $\epsilon_{\alpha 3} = 0$. In the thin plate limit, we usually expect very small transverse shear strains.

5.2.4 Summary of Plate Theory Notations

| | |
|--|--|
| w | (transverse displacement) |
| θ_α | (rotation vector) |
| $\kappa_{\alpha\beta} = \theta_{(\alpha,\beta)}$ | (curvature tensor) |
| $\gamma_\alpha = -\theta_\alpha + w_\alpha$ | (shear strain vector) |
| $m_{\alpha\beta} = \int_{-t/2}^{t/2} \sigma_{\alpha\beta} z \, dz$ | (moment tensor) |
| $q_\alpha = \int_{-t/2}^{t/2} \sigma_{\alpha 3} \, dz$ | (shear force vector) |
| W | (prescribed boundary displacement) |
| Θ_α | (prescribed boundary rotations) |
| $F = \int_{-t/2}^{t/2} f_3 \, dz + \langle h_3 \rangle^1$ | (total applied transverse force per unit area) |
| $C_\alpha = \int_{-t/2}^{t/2} f_\alpha z \, dz + \langle h_\alpha z \rangle$ | (total applied couple per unit area) |
| $M_\alpha = \int_{-t/2}^{t/2} h_\alpha z \, dz$ | (prescribed boundary moments) |
| $Q = \int_{-t/2}^{t/2} h_3 \, dz$ | (prescribed boundary shear force) |

5.2.5 Variational Equation

The variational equation of the plate theory is derived from the variational equation of the three-dimensional theory by making use of the preceding relations. The main steps are as follows.

i. Let s_q and s_h be subregions of s which satisfy $\overline{s_q \cup s_h} = s$ and $s_q \cap s_h = \emptyset$. The integrals appearing in the three-dimensional variational equations are replaced by the following iterated integrals:

$$\int_{\Omega} \dots d\Omega = \int_A \int_{-t/2}^{t/2} \dots dz \, dA \quad (5.2.11)$$

$$\int_{\Gamma_h} \dots d\Gamma = \int_A \langle \dots \rangle \, dA + \int_{s_h} \int_{-t/2}^{t/2} \dots dz \, ds \quad (5.2.12)$$

¹ The operator $\langle \dots \rangle$ is defined as follows: Let f be an arbitrary function of x , y , and z . Then $\langle f(x, y, z) \rangle = f(x, y, -t/2) + f(x, y, t/2)$.

The kinematic relations are also employed, yielding

$$\begin{aligned}
 0 &= \int_A \int_{-t/2}^{t/2} [\bar{u}_{(\alpha, \beta)} \sigma_{\alpha\beta} + 2\bar{u}_{(\alpha, 3)} \sigma_{\alpha 3}] dz dA \\
 &\quad - \int_A \int_{-t/2}^{t/2} (\bar{u}_\alpha \ell_\alpha + \bar{u}_3 \ell_3) dz dA \\
 &\quad - \int_A (\langle \bar{u}_\alpha h_\alpha \rangle + \langle \bar{u}_3 h_3 \rangle) dA \\
 &\quad - \int_{s_h} \int_{-t/2}^{t/2} (\bar{u}_\alpha h_\alpha + \bar{u}_3 h_3) dz ds \\
 &= \int_A \int_{-t/2}^{t/2} (-\bar{\kappa}_{\alpha\beta} \sigma_{\alpha\beta} z + \bar{\gamma}_\alpha \sigma_{\alpha 3}) dz dA \\
 &\quad - \int_A \int_{-t/2}^{t/2} (-\bar{\theta}_\alpha \ell_\alpha z + \bar{w} \ell_3) dz dA \\
 &\quad - \int_A (-\bar{\theta}_\alpha \langle h_\alpha z \rangle + \bar{w} \langle h_3 \rangle) dA \\
 &\quad - \int_{s_h} \int_{-t/2}^{t/2} (-\bar{\theta}_\alpha h_\alpha z + \bar{w} h_3) dz ds \tag{5.2.13}
 \end{aligned}$$

where

$$\bar{\kappa}_{\alpha\beta} = \bar{\theta}_{(\alpha, \beta)} \tag{5.2.14}$$

$$\bar{\gamma}_\alpha = -\bar{\theta}_\alpha + \bar{w}_{,\alpha} \tag{5.2.15}$$

Note. In the preceding relations we have used quantities with superposed bars to denote weighting functions in order to avoid notational conflicts and a proliferation of new notations.

ii. The definitions of force resultants are used, yielding

$$\begin{aligned}
 0 &= \int_A (-\bar{\kappa}_{\alpha\beta} m_{\alpha\beta} + \bar{\gamma}_\alpha q_\alpha) dA \\
 &\quad - \int_A (-\bar{\theta}_\alpha C_\alpha + \bar{w} F) dA \\
 &\quad - \int_{s_h} (-\bar{\theta}_\alpha M_\alpha + \bar{w} Q) ds \tag{5.2.16}
 \end{aligned}$$

iii. Integration by parts indicates, under the usual hypotheses, the differential equations and boundary conditions that are satisfied:

$$\begin{aligned}
 0 &= \int_A \bar{\theta}_\alpha \underbrace{(m_{\alpha\beta,\beta} - q_\alpha + C_\alpha)}_{\text{moment equilibrium}} dA \\
 &- \int_A \bar{w} \underbrace{(q_{\alpha,\alpha} + F)}_{\substack{\text{transverse} \\ \text{equilibrium}}} dA \\
 &+ \int_{s_h} \left\{ \bar{\theta}_\alpha \underbrace{(-m_{\alpha n} + M_\alpha)}_{\substack{\text{moment} \\ \text{boundary} \\ \text{conditions}}} + \bar{w} \underbrace{(q_n - Q)}_{\substack{\text{shear} \\ \text{boundary} \\ \text{condition}}} \right\} ds \quad (5.2.17)
 \end{aligned}$$

where

$$m_{\alpha n} = m_{\alpha\beta} n_\beta \quad (5.2.18)$$

$$q_n = q_\alpha n_\alpha \quad (5.2.19)$$

iv. Explicit forms of the constitutive equations in terms of the plate-theory variables are computed as follows:

$$\begin{aligned}
 m_{\alpha\beta} &= \int_{-t/2}^{t/2} \sigma_{\alpha\beta} z dz \\
 &= \int_{-t/2}^{t/2} (\bar{\lambda} \delta_{\alpha\beta} \epsilon_{\gamma\gamma} + 2\mu \epsilon_{\alpha\beta}) z dz \\
 &= -\frac{t^3}{12} [\bar{\lambda} \delta_{\alpha\beta} \theta_{\gamma,\gamma} + 2\mu \theta_{(\alpha,\beta)}] \\
 &= -c_{\alpha\beta\gamma\delta} K_{\gamma\delta} \quad (5.2.20)
 \end{aligned}$$

where

$$c_{\alpha\beta\gamma\delta} = \frac{t^3}{12} [\mu (\delta_{\alpha\gamma} \delta_{\beta\delta} + \delta_{\alpha\delta} \delta_{\beta\gamma}) + \bar{\lambda} \delta_{\alpha\beta} \delta_{\gamma\delta}] \quad (5.2.21)$$

$$\begin{aligned}
 q_\alpha &= \int_{-t/2}^{t/2} \sigma_{\alpha 3} dz \\
 &= \int_{-t/2}^{t/2} 2\mu \epsilon_{\alpha 3} dz \\
 &= t\mu (-\theta_\alpha + w_{,\alpha}) \\
 &= c_{\alpha\beta} \gamma_\beta \quad (5.2.22)
 \end{aligned}$$

$$c_{\alpha\beta} = t\mu \delta_{\alpha\beta} \quad (5.2.23)$$

Remarks

- Symmetry of the stiffness matrix will follow from the symmetries

$$c_{\alpha\beta\gamma\delta} = c_{\gamma\delta\alpha\beta} \quad (5.2.24)$$

$$c_{\alpha\beta} = c_{\beta\alpha} \quad (5.2.25)$$

The additional symmetries

$$c_{\alpha\beta\gamma\delta} = c_{\beta\alpha\gamma\delta} = c_{\alpha\beta\delta\gamma} \quad (5.2.26)$$

also hold.

2. To achieve results consistent with classical bending theory it is necessary to introduce a shear correction factor, κ , in the shear force–shear strain constitutive equation. This can be done by replacing $c_{\alpha\beta}$ by $\kappa c_{\alpha\beta}$. Throughout it is assumed that $\kappa = \frac{5}{6}$.

3. More-general material behavior (e.g., orthotropy) can be considered by appropriately redefining the elastic coefficients $c_{\alpha\beta\gamma\delta}$ and $c_{\alpha\beta}$.

5.2.6 Strong Form

The formal statement of the strong form of the plate theory boundary-value problem is as follows.

Given F , C_α , M_α , Q , W , and Θ_α , find w and θ_α such that

$$m_{\alpha\beta,\beta} - q_\alpha + C_\alpha = 0 \quad (5.2.27)$$

$$q_{\alpha,\alpha} + F = 0 \quad (5.2.28)$$

$$m_{\alpha\beta} = -c_{\alpha\beta\gamma\delta}\kappa_{\gamma\delta} \quad (5.2.29)$$

$$q_\alpha = c_{\alpha\beta}\gamma_\beta \quad (5.2.30)$$

$$\kappa_{\alpha\beta} = \theta_{(\alpha,\beta)} \quad (5.2.31)$$

$$\gamma_\alpha = -\theta_\alpha + w_{,\alpha} \quad (5.2.32)$$

$$\theta_\alpha = \Theta_\alpha \quad (5.2.33)$$

$$w = W \quad (5.2.34)$$

$$m_{\alpha n} = m_{\alpha\beta}n_\beta = M_\alpha \quad (5.2.35)$$

$$q_n = q_\alpha n_\alpha = Q \quad (5.2.36)$$

Sign conventions for stress resultants are depicted in Fig. 5.2.3.

5.2.7 Weak Form

The statement of the variational, or weak, form of the boundary-value problem is as follows.

Given F , C_α , M_α , Q , W , and Θ_α , find $\{\theta_1, \theta_2, w\} \in \mathcal{S}$ such that, for all $\{\bar{\theta}_1, \bar{\theta}_2, \bar{w}\} \in \mathcal{V}$

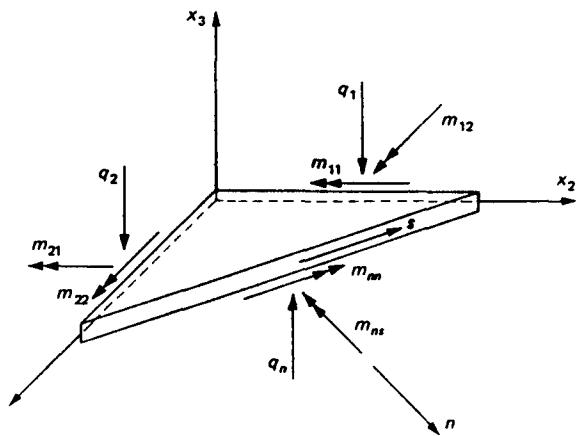
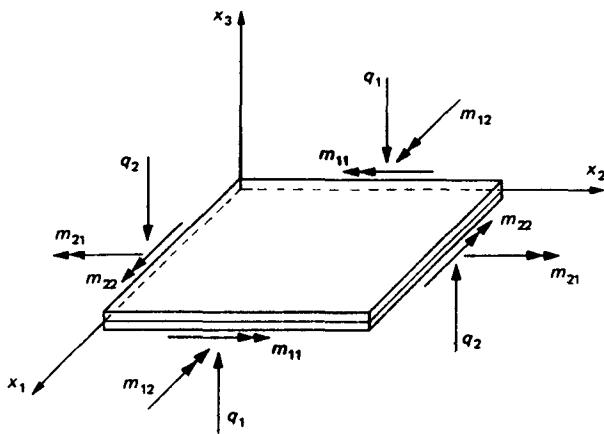


Figure 5.2.3 Sign convention for stress resultants.

$$\begin{aligned}
 0 = & \int_A [\bar{\theta}_{(\alpha, \beta)} C_{\alpha\beta} \gamma_\delta \theta_{(\gamma, \delta)} + \bar{\gamma}_\alpha C_{\alpha\beta} \gamma_\beta] dA \\
 & + \int_A (\bar{\theta}_\alpha C_\alpha - \bar{w} F) dA \\
 & + \int_{S_k} (\bar{\theta}_\alpha M_\alpha - \bar{w} Q) ds
 \end{aligned} \tag{5.2.37}$$

We assume that if

$$u = \begin{Bmatrix} w \\ \theta_1 \\ \theta_2 \end{Bmatrix} \in \mathcal{S} \quad (\text{the trial solution space})$$

Sec. 5.2 Reissner–Mindlin Plate Theory

then

$$\begin{Bmatrix} w \\ \theta_1 \\ \theta_2 \end{Bmatrix} = \begin{Bmatrix} W \\ \Theta_1 \\ \Theta_2 \end{Bmatrix} \quad \text{on } s_q$$

and that if

$$u = \begin{Bmatrix} \bar{w} \\ \bar{\theta}_1 \\ \bar{\theta}_2 \end{Bmatrix} \in \mathcal{V} \quad (\text{the weighting function space})$$

then

$$\begin{Bmatrix} \bar{w} \\ \bar{\theta}_1 \\ \bar{\theta}_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix} \quad \text{on } s_q$$

Exercise 1. Show that (5.2.27), (5.2.28), (5.2.35), and (5.2.36) are implied by (5.2.37).

Exercise 2. Put (5.2.37) into abstract notation: $a(\bar{u}, u) = (\bar{u}, f) + (\bar{u}, k)_\Gamma$. Define f and k and show that $a(\cdot, \cdot)$; (\cdot, \cdot) ; and $(\cdot, \cdot)_\Gamma$ are symmetric, bilinear forms.

5.2.8 Matrix Formulation

The matrix formulation of the variational equation is given as follows.

$$0 = \int_A (\bar{\kappa}^T D^b \kappa + \bar{\gamma}^T D^s \gamma) dA$$

$$+ \int_A (\bar{\theta}^T C - \bar{w} F) dA$$

$$+ \int_{s_k} (\bar{\theta}^T M - \bar{w} Q) ds \quad (5.2.38)$$

where

$$\theta = \begin{Bmatrix} \theta_1 \\ \theta_2 \end{Bmatrix} \quad \bar{\theta} = \begin{Bmatrix} \bar{\theta}_1 \\ \bar{\theta}_2 \end{Bmatrix} \quad (5.2.39)$$

$$\gamma = \begin{Bmatrix} \gamma_1 \\ \gamma_2 \end{Bmatrix} \quad \bar{\gamma} = \begin{Bmatrix} \bar{\gamma}_1 \\ \bar{\gamma}_2 \end{Bmatrix} \quad (5.2.40)$$

$$\kappa = \begin{Bmatrix} \kappa_{11} \\ \kappa_{22} \\ 2\kappa_{12} \end{Bmatrix} \quad \bar{\kappa} = \begin{Bmatrix} \bar{\kappa}_{11} \\ \bar{\kappa}_{22} \\ 2\bar{\kappa}_{12} \end{Bmatrix} \quad (5.2.41)$$

$$\mathbf{D}^b = \begin{bmatrix} D_{11}^b & D_{12}^b & D_{13}^b \\ & D_{22}^b & D_{23}^b \\ \text{symmetric} & & D_{33}^b \end{bmatrix} \quad (5.2.42)$$

$$D_{IJ}^b = c_{\alpha\beta\gamma\delta}, \quad (5.2.43)$$

| I/J | α/γ | β/δ |
|-----|-----------------|----------------|
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 3 | 1 | 2 |

$$\mathbf{D}^s = \begin{bmatrix} D_{11}^s & D_{12}^s \\ \text{symm.} & D_{22}^s \end{bmatrix} \quad (5.2.44)$$

$$D_{\alpha\beta}^s = c_{\alpha\beta} \quad (5.2.45)$$

5.2.9 Finite Element Stiffness Matrix and Load Vector

The finite element stiffness matrix and load vector may be obtained directly from the matrix form of the variational equation. The finite element approximations of w , \bar{w} , θ_α , and $\bar{\theta}_\alpha$ are denoted by w^h , \bar{w}^h , θ_α^h , and $\bar{\theta}_\alpha^h$, respectively. In a typical element, possessing n_{en} nodes,

$$w^h = \sum_{a=1}^{n_{en}} N_a w_a^h \quad (5.2.46)$$

$$\bar{w}^h = \sum_{a=1}^{n_{en}} N_a \bar{w}_a^h \quad (5.2.47)$$

$$\theta_\alpha^h = \sum_{a=1}^{n_{en}} N_a \theta_{\alpha a}^h \quad (5.2.48)$$

$$\bar{\theta}_\alpha^h = \sum_{a=1}^{n_{en}} N_a \bar{\theta}_{\alpha a}^h \quad (5.2.49)$$

where N_a is the shape function associated with node a , and w_a^h , \bar{w}_a^h , $\theta_{\alpha a}^h$, and $\bar{\theta}_{\alpha a}^h$ are the a th nodal values of w^h , \bar{w}^h , θ_α^h , and $\bar{\theta}_\alpha^h$, respectively.

Remark

It is not necessary to assume θ_α^h and w^h are defined in terms of the same shape functions and nodal patterns. However, in the applications we have in mind, this will be the case.

Define

$$\mathbf{d}^e = \{d_p^e\} \quad (5.2.50)$$

$$\bar{d}^e = \{\bar{d}_p^e\} \quad (5.2.51)$$

$$d_p^e = \begin{cases} w_a^h & p = 3a - 2 \\ \theta_{1a}^h & p = 3a - 1 \\ \theta_{2a}^h & p = 3a \end{cases} \quad (5.2.52)$$

$$\bar{d}_p^e = \begin{cases} \bar{w}_a^h & p = 3a - 2 \\ \bar{\theta}_{1a}^h & p = 3a - 1 \\ \bar{\theta}_{2a}^h & p = 3a \end{cases} \quad (5.2.53)$$

$$\kappa = B^b d^e \quad \bar{\kappa} = B^b \bar{d}^e \quad (5.2.54)$$

$$\gamma = B^s d^e \quad \bar{\gamma} = B^s \bar{d}^e \quad (5.2.55)$$

$$B^b = [B_1^b, B_2^b, \dots, B_{n_{en}}^b] \quad (5.2.56)$$

$$B^s = [B_1^s, B_2^s, \dots, B_{n_{en}}^s] \quad (5.2.57)$$

$$B_a^b = \begin{bmatrix} 0 & N_{a,x} & 0 \\ 0 & 0 & N_{a,y} \\ 0 & N_{a,y} & N_{a,x} \end{bmatrix} \quad (5.2.58)$$

$$B_a^s = \begin{bmatrix} N_{a,x} & -N_a & 0 \\ N_{a,y} & 0 & -N_a \end{bmatrix} \quad (5.2.59)$$

With these definitions, the following expressions for the element stiffness and load may be obtained:

$$k^e = k_b^e + k_s^e \quad (5.2.60)$$

$$k_b^e = \int_{A^e} B^{b^T} D^b B^b dA \quad (\text{bending stiffness}) \quad (5.2.61)$$

$$k_s^e = \int_{A^e} B^{s^T} D^s B^s dA \quad (\text{shear stiffness}) \quad (5.2.62)$$

$$f^e = \{f_p^e\} \quad (5.2.63)$$

$$f_p^e = \begin{cases} \int_{A^e} N_a F dA + \int_{s^e \cap s_h} N_a Q ds & p = 3a - 2 \\ - \int_{A^e} N_a C_1 dA - \int_{s^e \cap s_h} N_a M_1 ds & p = 3a - 1 \\ - \int_{A^e} N_a C_2 dA - \int_{s^e \cap s_h} N_a M_2 ds & p = 3a \end{cases} \quad (5.2.64)$$

A^e and s^e are the area and the boundary, respectively, of the e th element. The adjustment to f_p^e for prescribed displacements is given by

$$f_p^e \leftarrow f_p^e - \sum_{q=1}^{n_{ee}} k_{pq}^e q_q, \quad n_{ee} = 3n_{en} \quad (5.2.65)$$

where

$$q_p = \begin{cases} W(x_a, y_a) & p = 3a - 2 \\ \Theta_1(x_a, y_a) & p = 3a - 1 \\ \Theta_2(x_a, y_a) & p = 3a \end{cases} \quad (5.2.66)$$

The element stresses may be obtained from the following relations:

$$\begin{Bmatrix} m_{xx} \\ m_{yy} \\ m_{xy} \end{Bmatrix} = -D^b B^b d^e \quad (\text{bending moments}) \quad (5.2.67)$$

$$\begin{Bmatrix} q_x \\ q_y \end{Bmatrix} = D^e B^e d^e \quad (\text{shear resultants}) \quad (5.2.68)$$

Exercise 3. (Arrays with Respect to Right-hand-rule Rotations.) Show that if right-hand-rule rotations are being employed, i.e., if $\theta_{aa}^h \leftarrow \hat{\theta}_{aa}^h$ in (5.2.48), and, likewise, if $\bar{\theta}_{aa}^h \leftarrow \hat{\bar{\theta}}_{aa}^h$ in (5.2.49), then in place of (5.2.58), (5.2.59), and (5.2.64), respectively, we need to use

$$B_a^b = \begin{bmatrix} 0 & 0 & -N_{a,x} \\ 0 & N_{a,y} & 0 \\ 0 & N_{a,x} & -N_{a,y} \end{bmatrix}$$

$$B_a^e = \begin{bmatrix} N_{a,x} & 0 & N_a \\ N_{a,y} & -N_a & 0 \end{bmatrix}$$

$$f_p^e = \begin{cases} \text{same as in (5.2.64)}, & p = 3a - 2 \\ -\int_{A^e} N_a C_2 dA - \int_{s^e \cap s_h} N_a M_2 ds, & p = 3a - 1 \\ \int_{A^e} N_a C_1 dA + \int_{s^e \cap s_h} N_a M_1 ds, & p = 3a \end{cases}$$

5.3 PLATE-BENDING ELEMENTS

5.3.1 Some Convergence Criteria

It is important to realize that convergence criteria for elements derived from the present theory are quite different than those for elements derived from thin plate theory. *Necessary* conditions in the present case are:

1. All three rigid body modes must be exactly representable
2. The following five constant strain states must be exactly representable:

$$\left. \begin{array}{l} \theta_{1,1} \\ \theta_{2,2} \\ \frac{1}{2}(\theta_{1,2} + \theta_{2,1}) \\ -\theta_1 + w_{,1} \\ -\theta_2 + w_{,2} \end{array} \right\} \quad \begin{array}{l} (\text{curvatures}) \\ \\ \\ (\text{transverse shear strains}) \end{array}$$

These conditions are satisfied for standard isoparametric elements and for the non standard isoparametric elements described in the following sections.

5.3.2 Shear Constraints and Locking

An important consideration in the development of plate-bending elements, based upon the present theory, is the number of shear strain constraints engendered in the thin plate limit (i.e., as $t \rightarrow 0$). To see this, we consider a heuristic example.

Example 1

Assume a four-node isoparametric quadrilateral element and, for simplicity, assume the element is of rectangular plan and the sides are aligned with the global x - and y -axes. In this case, the element expansions may be written as

$$w^h = \beta_0 + \beta_1x + \beta_2y + \beta_3xy \quad (5.3.1)$$

$$\theta_a^h = \gamma_{a0} + \gamma_{a1}x + \gamma_{a2}y + \gamma_{a3}xy \quad (5.3.2)$$

where β_i and γ_{ai} , $0 \leq i \leq 3$, are constants that depend upon the nodal parameters w_i^h and θ_{aa}^h , $1 \leq a \leq 4$, respectively. The conditions

$$\begin{aligned} 0 &= \gamma_1 \\ &= -\theta_1^h + w_{,1}^h \\ &= (-\gamma_{10} + \beta_1) - \gamma_{11}x + (-\gamma_{12} + \beta_3)y - \gamma_{13}xy \end{aligned} \quad (5.3.3)$$

$$\begin{aligned} 0 &= \gamma_2 \\ &= -\theta_2^h + w_{,2}^h \\ &= (-\gamma_{20} + \beta_2) + (-\gamma_{21} + \beta_3)x - \gamma_{22}y - \gamma_{23}xy \end{aligned} \quad (5.3.4)$$

impose eight constraints per element and are approximately in force as $t \rightarrow 0$ if exact integration of \mathbf{k}_s^h is performed. (Two-by-two Gauss integration is exact in this case.) In a large rectangular mesh, there are approximately three degrees-of-freedom per element and thus the element tends to be overly constrained. In practice, worthless numerical results are obtained [1]. To alleviate the “locking” effect, one might consider using one-point Gauss quadrature for \mathbf{k}_s^h . Clearly, this results in only two constraints per element, and now there are more degrees of freedom than there are constraints. This element, with one-point shear integration and 2×2 bending integration, was proposed and shown to be effective by Hughes et al. [2].

Arguments similar to those in Example 1 have been used to evaluate other possibilities (see Pugh et al. [1, 3] and Malkus and Hughes [4]).

The situation is seen to be similar to that for the incompressible problem discussed in Chapter 4. Again we shall define the *constraint ratio*, r , for the standard mesh of Fig. 4.3.3 by

$$r = \frac{n_{eq}}{n_c}$$

where, in the present case, n_{eq} is the total number of displacement and rotation equations after boundary conditions have been imposed and n_c is the total number of shear strain constraints. Again, the idea is that as the number of equations in the standard mesh approaches infinity, r should approximate the ratio of equilibrium equations to constraints for the governing system of partial differential equations (in the present case, 3 and 2, respectively). Consequently, here the ideal value of r would be $\frac{3}{2}$. Smaller values would indicate the presence of too many shear strain constraints and a potential for locking. A larger value would indicate too few shear strain constraints and suggest that the Kirchhoff limit might be poorly approximated. Note that for the fully integrated four-node element discussed above, $r = \frac{3}{8}$, indicative of locking, whereas if one-point Gaussian quadrature is used for k_s^e , then $r = \frac{3}{2}$, the optimal value.

We wish to emphasize again that the constraint ratio is only a quick device for estimating an element's propensity to lock. (See the discussion in Sec. 4.3.7.) In fact, the constraint ratio is not as successful for plates as for incompressible continuum elements. (There are excellent plate elements with constraint ratios less than $\frac{3}{2}$. See, for example, Sec. 5.3.7.) A superior, yet still simple, methodology for assessing the tendency of plates to lock is based upon the *Kirchhoff mode concept* [17, 18]. Much of the recent work on plate element design explicitly or implicitly employs this concept. The interested reader should consult [17] for a complete description.

5.3.3 Boundary Conditions

It is important to realize that boundary conditions in the present theory are not always the same as those for the classical thin plate theory. The differences occur in the specification of the "simply supported" case. In the present theory, there are two ways of going about this, depending on the actual physical constraint. Rather than being an additional complication, this freedom turns out to be a considerable benefit, for it enables the solution of problems in which thin plate finite elements have heretofore failed (see Rossow [5] and Scott [6]).

Consider a smooth portion of the plate boundary and a local s, n -coordinate system to it (s denotes the tangential direction and n the outward normal direction; see Fig. 5.3.1). The most common boundary conditions encountered in practice are given as follows.

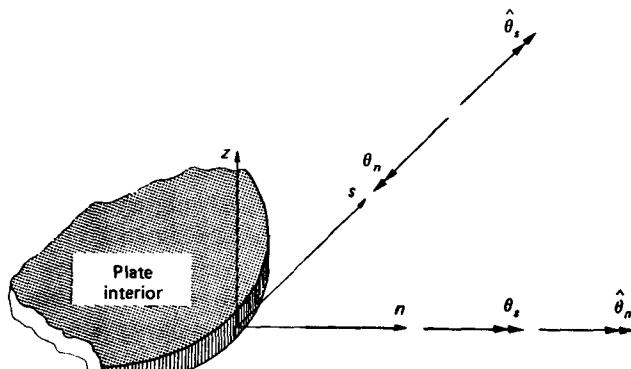


Figure 5.3.1 Local tangential-normal coordinate system at plate boundary.

Clamped

$$w = 0$$

$$\theta_s = 0$$

$$\theta_n = 0$$

Free

$$Q = 0$$

$$M_s = 0$$

$$M_n = 0$$

Simply supported

$$SS_1 \quad w = 0$$

$$M_s = 0$$

$$M_n = 0$$

$$SS_2 \quad w = 0$$

$$\theta_s = 0$$

$$M_n = 0$$

Symmetric

$$Q = 0$$

$$M_s = 0$$

$$\theta_n = 0$$

Skew symmetric²

$$w = 0$$

$$\theta_s = 0$$

$$M_n = 0$$

In thin-plate theory, SS_2 is the appropriate simply supported boundary condition, since $w = 0$ along the boundary necessitates $\partial w / \partial s = 0$, and the absence of shear strains requires $\theta_s = \partial w / \partial s$. When curved-boundary, simply supported plates are approximated by straight-edged, thin plate elements, SS_2 leads to difficulties. For in this case, specifying $\partial w / \partial s = 0$ at interelement boundaries, for which s is not collinear (see Fig. 5.3.2), implies $\partial w / \partial n = 0$ as well. Thus, as the mesh is refined, the clamped boundary condition is achieved. In other words, the correct solution to the *wrong* problem is attained! Strategies for circumventing this “paradox” tend to be inconvenient from an implementational standpoint (for further details, see Scott [6]).

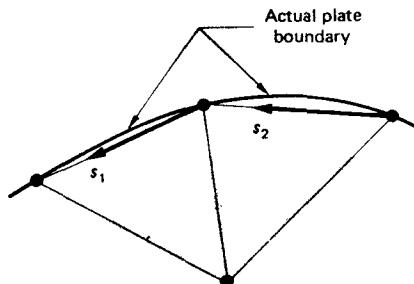


Figure 5.3.2 Approximation to curved plate boundary by straight-edged finite elements.

Another “paradox”, discussed by Rossow [5], concerns the solution of a uniformly loaded, simply supported, rhombic thin plate. The analytic solution is singular in the vicinity of the obtuse vertices, with moments of *opposite* sign. However, Sander [7] has reported solutions in which thin-plate elements yield moments of the *same* sign. This phenomenon may also be traced to an overconstraining of thin plate elements caused by SS_2 . (Some numerical results for the rhombic plate are presented in Sec. 5.3.8.)

A pleasant feature of the present theory is that the alternative, simply supported boundary condition, SS_1 , completely obviates these “paradoxes.” In cases in which there is no danger of overconstraining, such as simply supported rectangular plates, SS_2 may be employed to further eliminate degrees of freedom, for greater economy of solution. Computations in support of these assertions are presented in the following sections.

Rhee [8] has discussed the polygonal approximation to simply supported, curved-edged, ***thin plates***. Rhee has demonstrated convergence of thin plate elements for this case by setting boundary-node values of w , but not $\partial w / \partial x_\alpha$, equal to zero. This simple technique appears to be the most computationally attractive strategy for overcoming the convergence difficulties described above. We note, however, that this

² Observe that SS_2 and the skew-symmetric case are the same.

approach does not achieve $w = 0$ along the edge of the plate, except in the fine-mesh limit.

5.3.4 Reduced and Selective Integration Lagrange Plate Elements

To avoid shear “locking” in thin plates, some form of reduced integration is suggested. The two most obvious possibilities are *uniform reduced integration* and *selective reduced integration*.

In uniform reduced integration, both the bending and shear terms are integrated with the same rule, which is of lower order than the “normal” one. Apparently, the first example of a uniform reduced integration plate element was the eight-node serendipity element of Zienkiewicz et al. [9], in which 2×2 Gauss quadrature was employed. Although this element has received wide use, it has now been shown to behave poorly in the thin plate limit [1, 3]. Nevertheless, reduced integration still represents a considerable improvement over normal integration.

In selective reduced integration, the bending term is integrated with the normal rule, whereas the shear term is integrated with a lower-order rule.

Integration rules for Lagrange plate-bending elements are indicated in Fig. 5.3.3. These elements have been investigated by Hughes et al. [2, 10] and Pugh et al. [1, 3], among others.

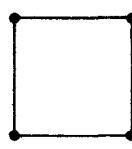
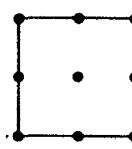
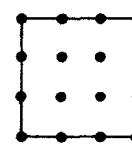
| | | | |
|---|---|---|---|
| |  |  |  |
| w, θ_1, θ_2 shape functions | Bilinear | Biquadratic | Bicubic |
| Uniform reduced integration | 1×1 $U1$ | 2×2 $U2$ | 3×3 $U3$ |
| Selective reduced integration | 1×1 shear 2×2 bending $S1$ | 2×2 shear 3×3 bending $S2$ | 3×3 shear 4×4 bending $S3$ |

Figure 5.3.3 Lagrange plate elements. Three degrees of freedom per node:
 w, θ_1, θ_2 .

Numerical Examples

Several numerical examples are presented, which indicate the accuracy attainable by the reduced integration Lagrange elements. Bending moments are reported at "optimal points" [11] (i.e., the Gauss points of the shear integration rule). A Poisson's ratio of 0.3 and Young's modulus of 10.92×10^5 were used throughout. Generally, the meshes are constructed so that there are four U1, or S1, elements for every U2, or S2, element, resulting in the same number of equations. The plates analyzed herein may be considered thin, and so comparison is made with classical thin plate theory. However, since shear deformations are included in the present theory, convergence to slightly greater displacements is to be expected.

Square plate

Convergence studies were carried out for both simply supported and clamped, thin, square plates, which were subjected to uniform and concentrated loads. Mesh types are depicted in Fig. 5.3.4 and results are presented in Fig. 5.3.5. The geometric data employed were $L = 10$ and $t = 0.1$. Refinements were constructed by bisection. In this problem, the SS₂ simply supported boundary condition produces convergent solutions and results in fewer equations.

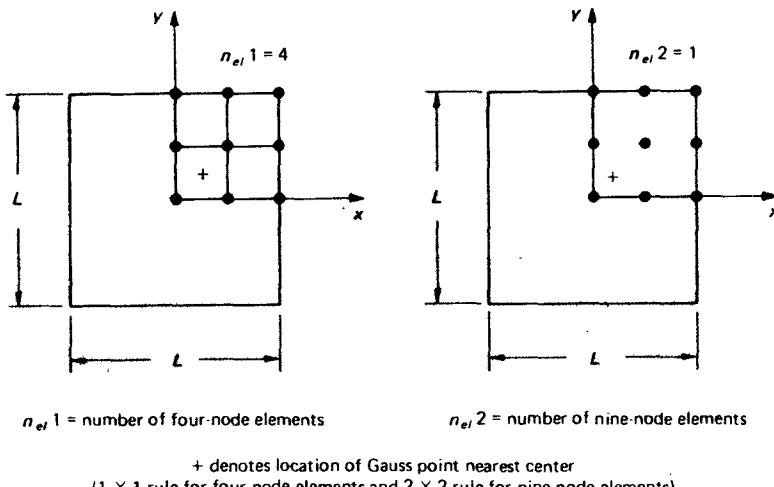


Figure 5.3.4 Square plate. Due to symmetry, only one quadrant is discretized.

Circular plate

A convergence study, similar to the preceding one, was carried out for thin circular plates. Meshes are depicted in Fig. 5.3.6 and results are presented in Fig. 5.3.7. The geometric data employed were $R = 5$ and $t = 0.1$. In this case, it is necessary to use

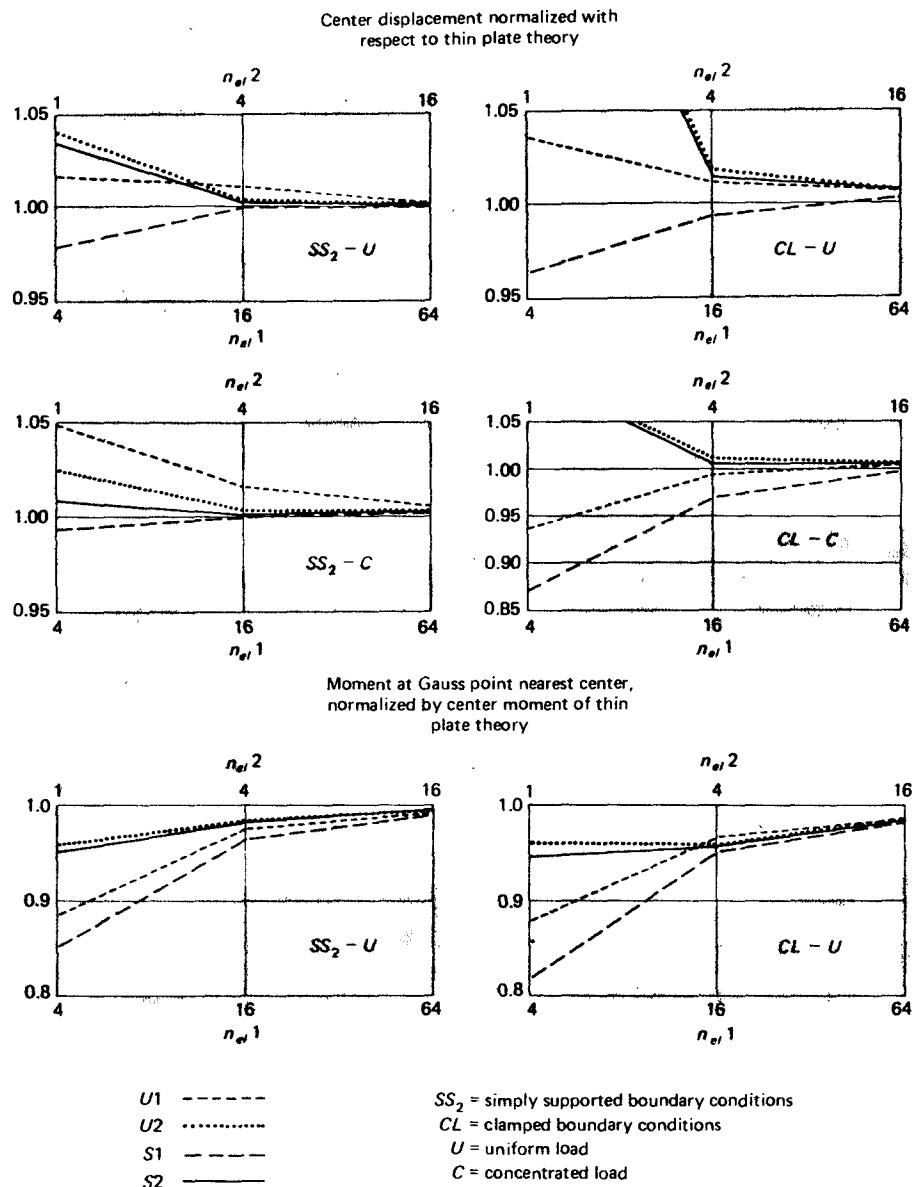


Figure 5.3.5 Convergence study for thin, square plate.

the SS₁ boundary condition to achieve convergence in the simply supported case (see Sec. 5.3.3).

Bending moments for elements U1 and S1 are plotted in Fig. 5.3.8.

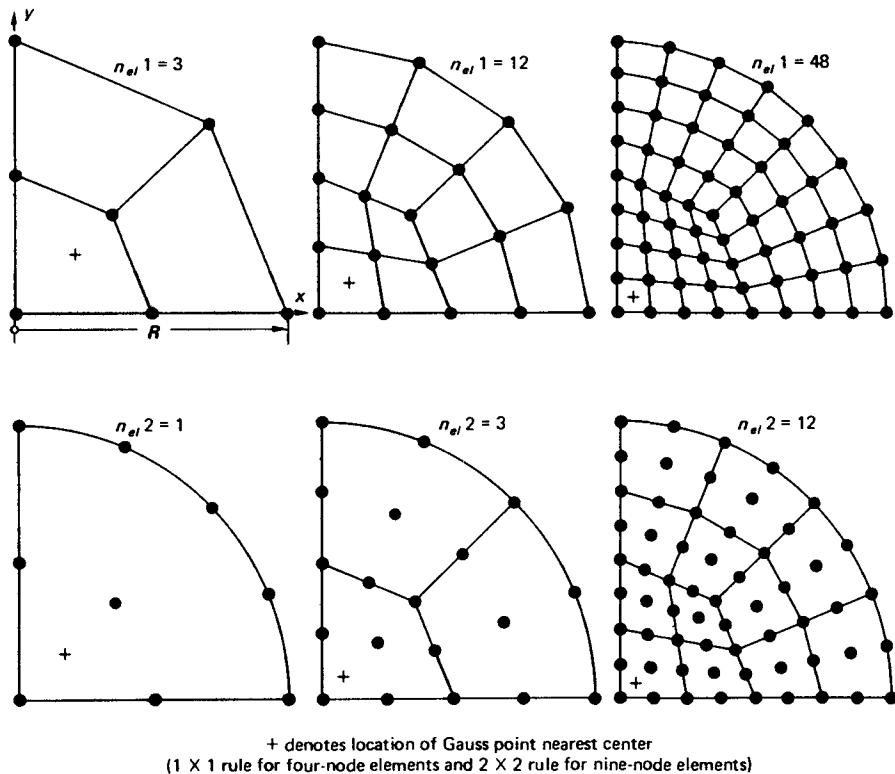


Figure 5.3.6 Circular plate. Due to symmetry, only one quadrant is discretized.

5.3.5 Equivalence with Mixed Methods

The reduced and selective integration Lagrange plate elements have been shown to be identical to elements derived from a mixed formulation, in which the shear forces, q_α , are variables in addition to w and θ_α ; see Malkus and Hughes [4]. The shear variables, with nodes at the Gauss points of the shear integration rules, are discontinuous between elements. Some examples of equivalent elements are given below.

Example 2

Let w and θ_α be approximated by bilinear shape functions and let q_α take on constant values within each element. If one-point Gauss quadrature is used to construct the stiffness, it is equivalent to $U1$. If 2×2 Gauss quadrature is used on the bending term and one-point Gauss quadrature is used on the remaining shear term, then the element is equivalent to $S1$. In each case, the constant value of q_α in the mixed formulation is the same as that computed from w and θ_α at the centroid of $U1$ or $S1$, respectively.

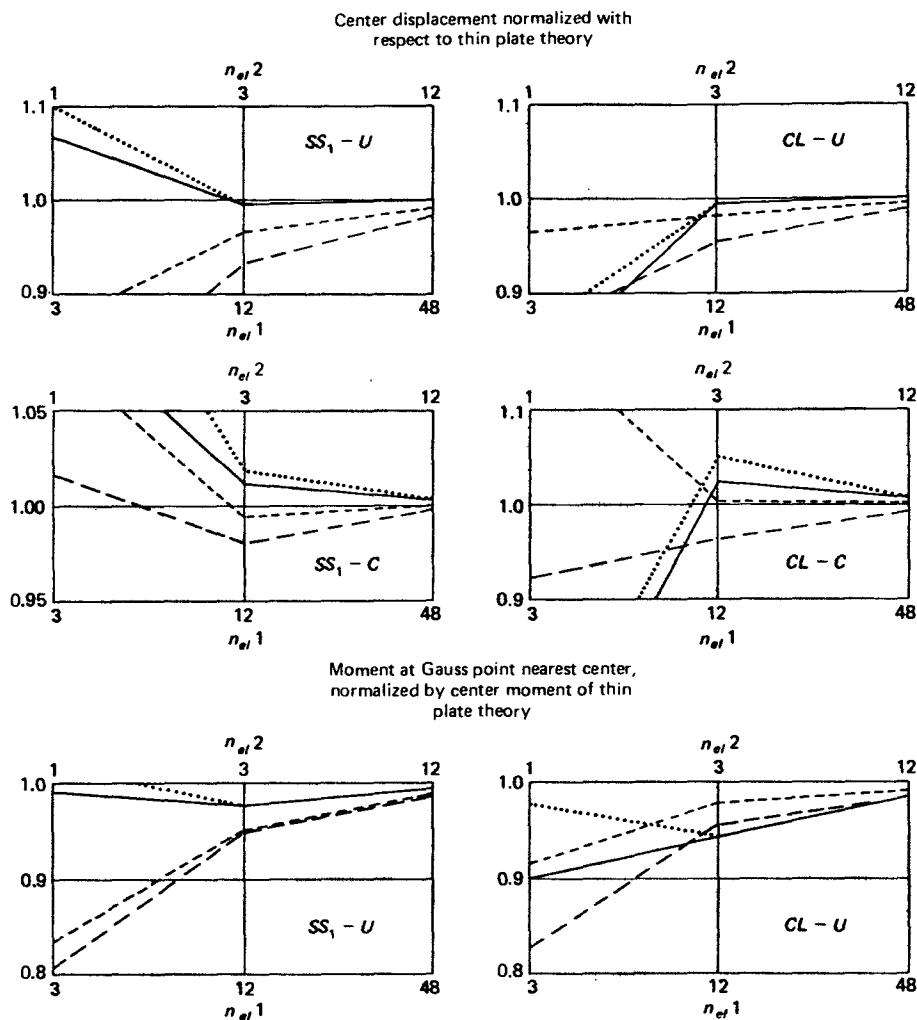


Figure 5.3.7 Convergence study for thin circular plate.

Example 3

Let w and θ_α be approximated by biquadratic shape functions and let q_α be approximated by bilinear shape functions. The nodes for q_α are the 2×2 Gauss points. If the 2×2 Gauss rule is used for all terms, the element is equivalent to $U2$. If the 3×3 rule is used on the bending term and the 2×2 rule is used on the shear terms, the element is equivalent to $S2$. In each case, the bilinear variation of q_α within each element in the mixed formulation may be recaptured in $U2$ and $S2$, respectively, by passing a bilinear function through the 2×2 Gauss point values of q_α computed from w and θ_α .

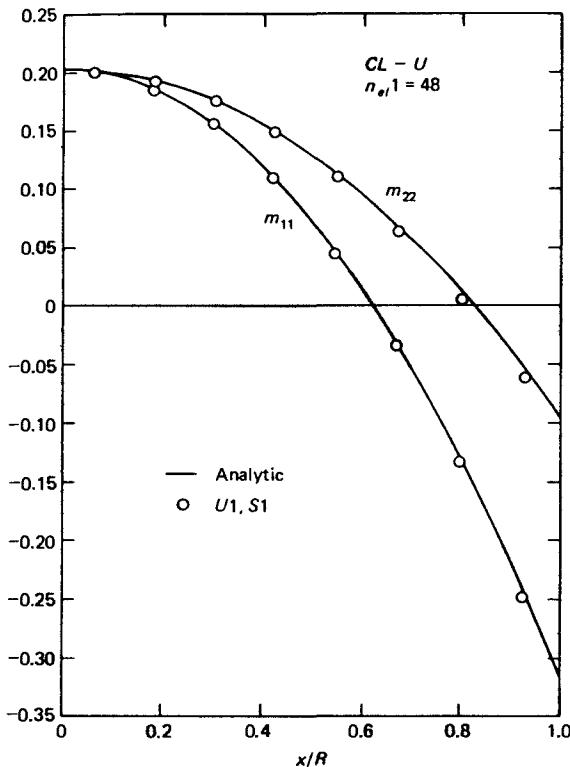


Figure 5.3.8 Circular plate. Bending moments at Gauss points nearest x -axis.

Additional details on these and related matters may be found in Malkus and Hughes [4].

5.3.6 Rank Deficiency

Although reduced integration of Lagrange elements alleviates the shear-locking problem, it has a deleterious side effect: rank deficiency (i.e., there are zero-energy modes in excess of the three rigid-body modes; see Table 5.3.1). This can result in oscillatory errors in certain cases (see Sec. 5.3.7) and, occasionally, even in a singular global stiffness matrix. As can be clearly seen, the number is lower for each selective reduced integration element than for the corresponding uniform reduced integration element, an advantage of the former.

The four zero-energy modes of a square $U1$ element are depicted in Fig. 5.3.9. Element $S1$ possesses only the w hourglass and the in-plane twist modes.

The zero-energy modes for a square $U2$ element are:

$$w = x_1^2 x_2^2 - \frac{1}{3}(x_1^2 + x_2^2), \quad \theta_1 = \theta_2 = 0 \quad (5.3.5)$$

$$w = 0, \quad \theta_1 = 0, \quad \theta_2 = (x_1^2 - \frac{1}{3})(x_2^2 - \frac{1}{3}) \quad (5.3.6)$$

$$w = 0, \quad \theta_1 = (x_1^2 - \frac{1}{3})(x_2^2 - \frac{1}{3}), \quad \theta_2 = 0 \quad (5.3.7)$$

$$w = 0, \quad \theta_1 = x_1(x_2^2 - \frac{1}{3}), \quad \theta_2 = -x_2(x_1^2 - \frac{1}{3}) \quad (5.3.8)$$

Only (5.3.5) is present in $S2$.

TABLE 5.3.1 Zero-energy Modes, in Excess of Rigid-body Modes, for Reduced Integration, Lagrange, Plate Elements

| Element | $U1$ | $U2$ | $U3$ | $S1$ | $S2$ | $S3$ |
|-----------------------------|------|------|------|------|------|------|
| Number of zero-energy modes | 4 | 4 | 4* | 2 | 1 | 1** |

* Pugh et al. [3].

** Wong [28].

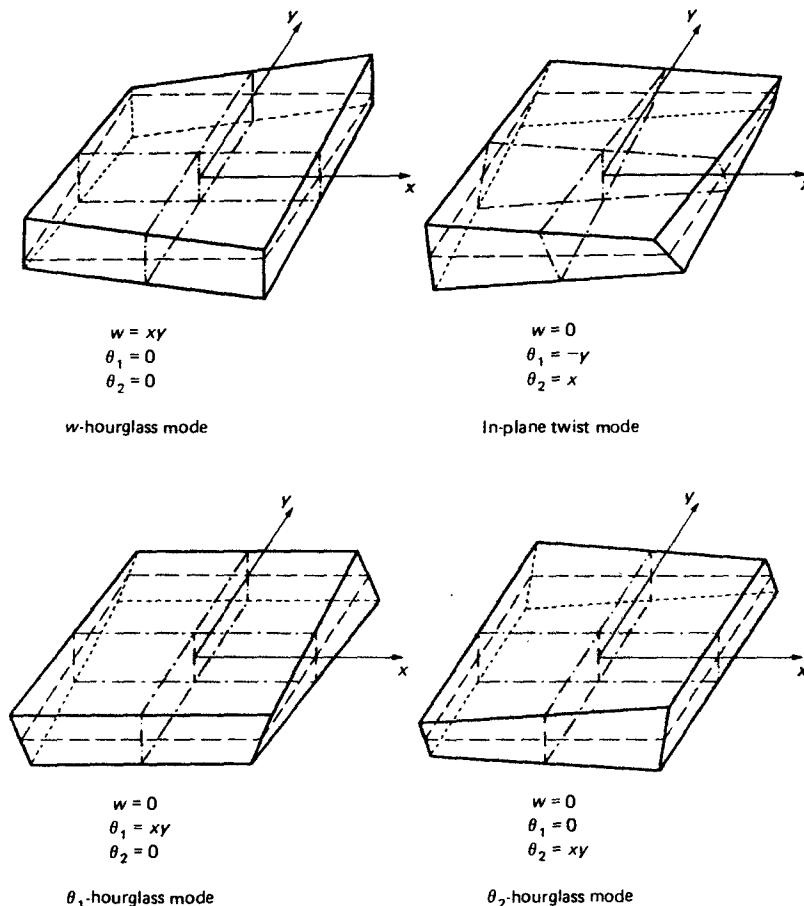


Figure 5.3.9 Zero-energy modes of element $U1$.

In general, boundary conditions render the assembled stiffness matrix positive-definite, and so the zero-energy modes are not globally present. A guideline that we have employed with success for selective integration elements is: If the boundary conditions preclude the rigid mode from forming in one element, it is also precluded in the remainder of the mesh.

It turns out that a sufficient condition for satisfying the guideline for elements S1 and S2 is that two adjacent nodal values of w be specified in at least one element.

A critical test is provided by a point-supported, pergola roof. (We thank P. C. Jennings for suggesting this problem to us.) The roof is assumed to be a square plate, subjected to uniform transverse load, and "fixed" at the center. A mesh of square S1 elements was constructed with the center node clamped (i.e., $w = \theta_1 = \theta_2 = 0$). Rank-check analyses have revealed that the in-plane twist mode disappears in a mesh of two or more S1 elements, so it is not a problem. However, the w hourglass is present unless eliminated via boundary conditions. As is clear, a global pattern, in which each element hourglasses, is possible in the present situation, since only one w degree of freedom is specified. (A rank-check analysis confirmed this to be the case, indicating one zero mode.) An alternative mesh was employed, in which the point support was distributed in a diamond pattern; see Fig. 5.3.10. (In fact, this is more in keeping with the physical situation.) Since $w = 0$ along an edge of each of the four central elements, these elements cannot hourglass. A rank-check analysis revealed a positive-definite stiffness, in confirmation of this. Note that it is crucial to have $w = 0$ at the center; otherwise the w hourglass reappears.

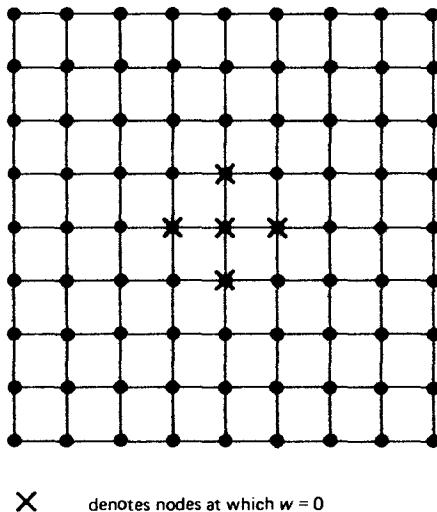


Figure 5.3.10 Pergola roof. Mesh of S1 elements exhibiting no zero-energy modes.

Due to the potential problems created by rank deficiency, a number of research efforts have been undertaken to develop Reissner-Mindlin-type elements which have correct rank and maintain accuracy in both thin and thick plate applications. Several elements of this type are discussed in the following sections.

Before going further, we note that there has been considerable interest in the one-point quadrature element U1 because of its obvious economy and simplicity.

Some efforts have been made to stabilize its spurious rigid body modes by way of the hourglass-stiffness technique. The interested reader may consult [12] and [13] for details.

5.3.7 The Heterosis Element

The first higher-order Mindlin-type element to simultaneously achieve correct rank and high accuracy was the heterosis element of Hughes and Cohen [14]. The heterosis element represents a synthesis of the selectively integrated nine-node Lagrange element and eight-node serendipity element. The heterosis element combines the attributes of its progenitors but avoids the shortcomings. It may be recalled that the selectively integrated nine-node Lagrange element is rank deficient; and the eight-node serendipity element tends to be overconstrained—the shear-locking phenomenon—occasionally resulting in poor convergence characteristics [1, 10].

The heterosis element has consistently performed well on numerical tests, including cases in which the serendipity and Lagrange elements are poor. Pertinent information for the element is presented in Fig. 5.3.11, along with descriptions of the serendipity and Lagrange elements. The name heterosis was chosen because, in genetics, the improvement in characteristics exhibited by hybrids over those of the parents is called heterosis.

Implementation

The use of two interpolatory schemes, as ostensibly necessitated by the heterosis elements, would be a drawback. However, implementation can be made without such considerations [15]. Specifically, we start with routines that generate the stiffness and internal force for the Lagrange element (denoted by k_{Lag}^e , and f_{Lag}^e , respectively). Then a transformation matrix, \mathbf{H} , is constructed, which restrains the transverse displacements of the internal nodes to interpolate the serendipity shape functions. The heterosis stiffness, k_{het}^e , and internal force, f_{het}^e , are then defined by

$$\mathbf{k}_{\text{het}}^e = \mathbf{H}^T \mathbf{k}_{\text{Lag}}^e \mathbf{H} \quad (5.3.9)$$

$$\mathbf{f}_{\text{het}}^e = \mathbf{H}^T \mathbf{f}_{\text{Lag}}^e \quad (5.3.10)$$

The matrix \mathbf{H} is defined as follows:

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{24} & \mathbf{0}_{25,2} \\ \mathbf{h} & \mathbf{I}_2 \end{bmatrix} \quad (5.3.11)$$

$$\mathbf{h} = [\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_8] \quad (5.3.12)$$

$$\mathbf{h}_a = \begin{bmatrix} N_a^{\text{ser}}(0, 0) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad 1 \leq a \leq 8 \quad (5.3.13)$$

where \mathbf{I}_m denotes the $m \times m$ identity matrix; $\mathbf{0}_{m,n}$ denotes the $m \times n$ zero matrix; and $N_a^{\text{ser}}(0, 0)$ is the serendipity shape function associated with node a evaluated at the origin of isoparametric coordinates (i.e., location of node 9).

| | Serendipity | Heterosis | Lagrange |
|---------------------------------------|-------------|-------------|----------|
| w-shape functions | Serendipity | Serendipity | Lagrange |
| θ_1, θ_2 -shape functions | Serendipity | Lagrange | Lagrange |
| Integration scheme | U2 | S2 | S2 |
| Number of spurious zero-energy modes | 1* | 0 | 1 |
| Constraint ratio | 1.125 | 1.375 | 1.5 |

Key:

● w, θ_1, θ_2 degrees of freedom

○ θ_1, θ_2 degrees of freedom

U2 = 2 X 2 Gauss

S2 = { bending 3 X 3 Gauss
shear 2 X 2 Gauss

*Not communicable in a mesh of two or more elements.

Figure 5.3.11 Quadrilateral plate elements.

The matrix multiplications in (5.3.9) and (5.3.10) are formal, and in actual practice sparse coding may be employed to obtain $k_{\text{het}}^{\text{ser}}$ and $f_{\text{het}}^{\text{ser}}$ efficiently. Stresses for the heterosis element may again be obtained via (5.2.67) and (5.2.68), where the center transverse displacement is defined by

$$w_9 = \sum_{a=1}^8 N_a^{\text{ser}}(0, 0) w_a \quad (5.3.14)$$

Note that the derivation of the heterosis element arrays depends upon the serendipity shape functions only through their values at node 9. These eight numbers are independent of the particular element and may be stored once and for all in an element routine through the use of a FORTRAN DATA statement.

An alternative implementation of the heterosis element, based upon the "hierarchical concept," has been described in [16].

Higher-order versions of the heterosis element have been proposed in [15]; see Fig. 5.3.12.

| | | | |
|--------------------|-------|-------|-------|
| | | | |
| Integration scheme | S2 | S3 | S4 |
| Constraint ratio | 1.375 | 1.278 | 1.219 |

Key: ● w, θ_1, θ_2 degrees of freedom

○ θ_1, θ_2 degrees of freedom

S_n { bending $(n+1) \times (n+1)$ Gauss
shear $n \times n$ Gauss

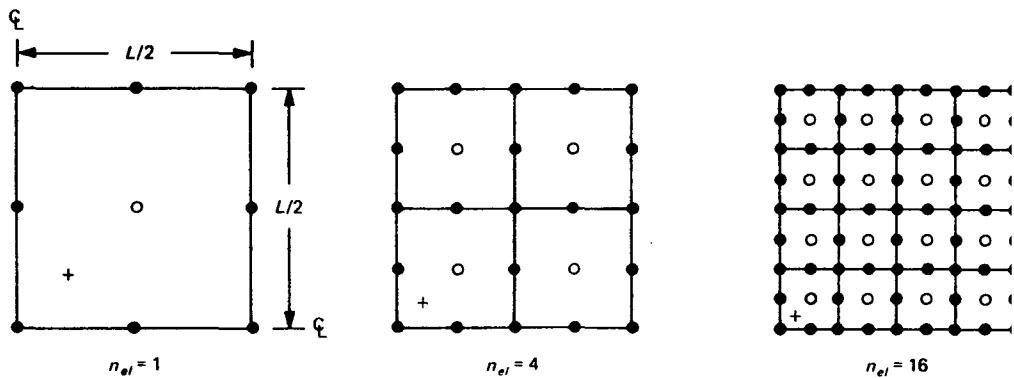
Figure 5.3.12 Heterosis plate elements.

Numerical Examples

A Poisson's ratio of 0.3 and a Young's modulus of 10.92×10^5 were used throughout.

Square plate

Mesh patterns are depicted in Fig. 5.3.13. The geometric data employed were $L = 10$ and t (thickness) = 0.1.



⊕ denotes location of Gauss point (2 × 2 rule) nearest center

Figure 5.3.13 Square plate. Due to symmetry, only one quadrant is discretized.

The results of a convergence study are presented in Fig. 5.3.14. The serendipity element exhibits particularly poor convergence characteristics for the clamped cases. The Lagrange and heterosis elements demonstrate good convergence properties, with the former being somewhat more accurate.

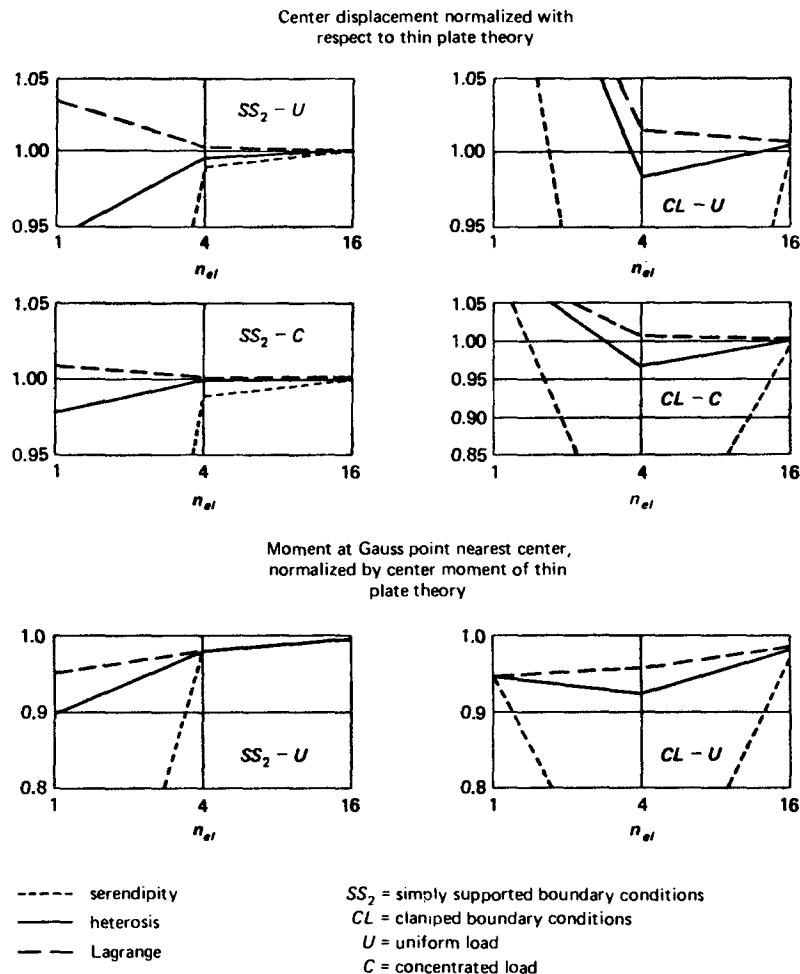


Figure 5.3.14 Convergence study for thin square plate.

The ability of the elements to handle extremely thin plates is illustrated in Fig. 5.3.15. Both the Lagrange and heterosis elements exhibit “stable” convergence characteristics for large length-to-thickness ratios. Numerical deterioration commences at $L/t = 10^6$ because the shear stiffness is $O(L^2/t^2)$ times the bending stiffness, resulting in the bending effects being overwhelmed by rounding errors. The “plateaus” exhibited by the Lagrange and heterosis elements, up to $L/t = 10^6$, may be extended *ad infinitum* by the technique proposed in [2]. On the other hand, the serendipity element diverges rapidly for relatively small values of L/t (i.e., $O(10^2)$).

Rectangular plate

Convergence studies for rectangular plates are presented in [15]. Sample results for the meshes shown in Fig. 5.3.16 are presented in Fig. 5.3.17.

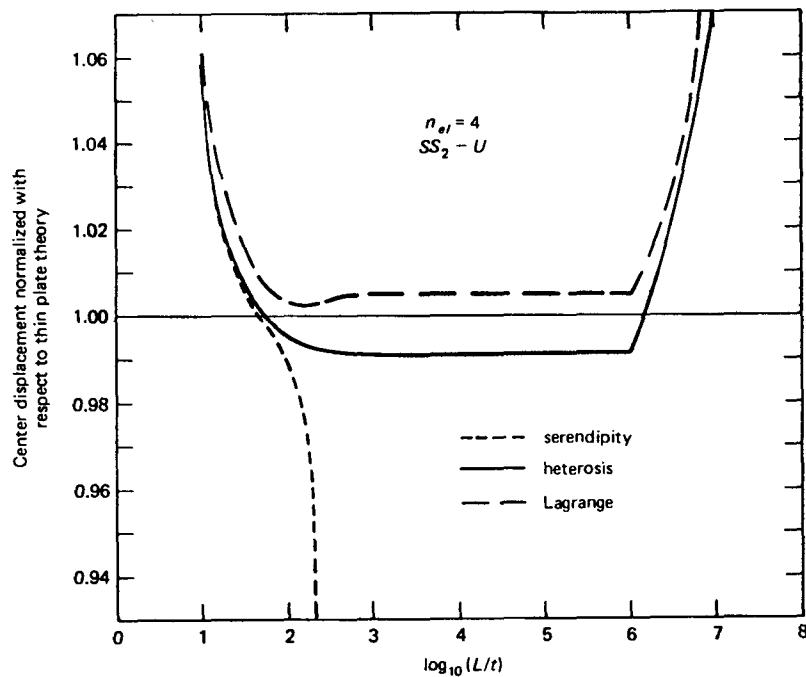


Figure 5.3.15 Aspect ratio study for square plate.

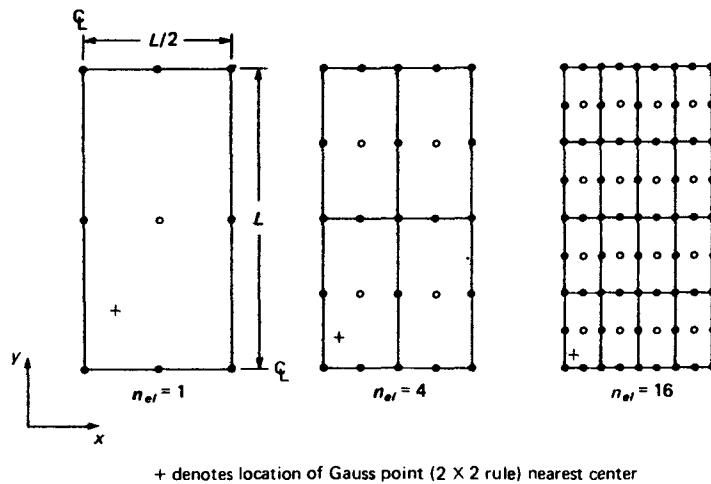


Figure 5.3.16 Rectangular plate (aspect ratio = 2). Due to symmetry, only one quadrant is discretized.

Circular plate

The meshes for this problem are depicted in Fig. 5.3.18. A value of $R = 5$ is used

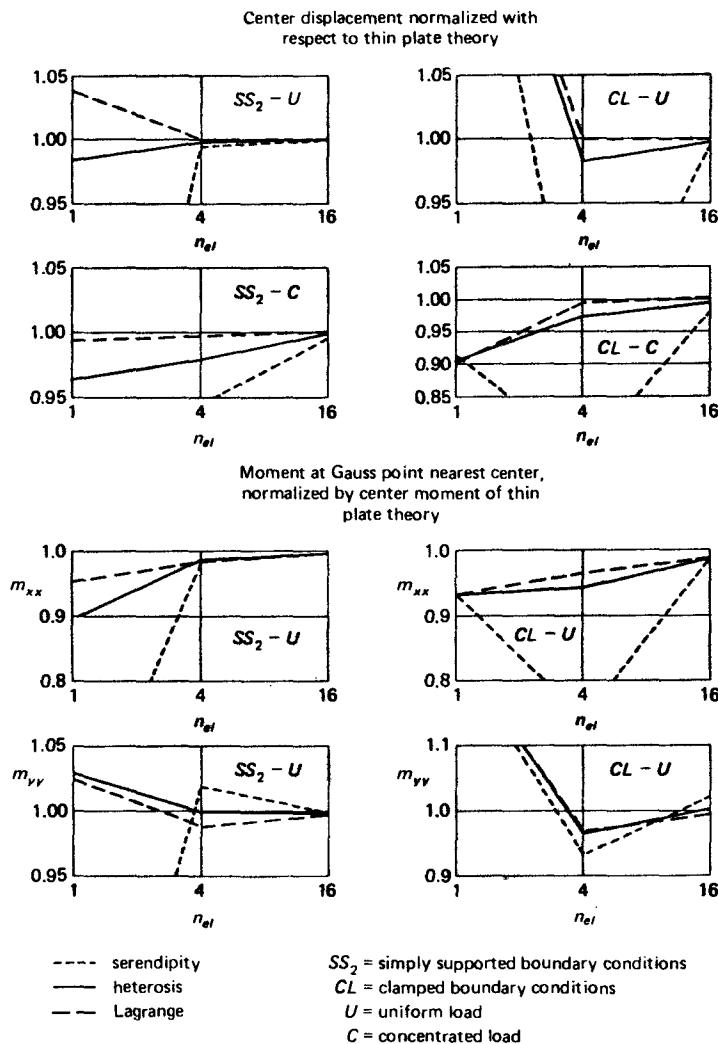


Figure 5.3.17 Convergence study for thin rectangular plate (aspect ratio = 2).

for all cases. Thin plate (i.e., $t = 0.1$) convergence results are presented in Fig. 5.3.19. The heterosis element is consistently superior to the serendipity element.

As we have seen so far, the presence of the spurious zero-energy mode generally does not adversely affect the accuracy of the Lagrange element. The reason for this is that when two adjacent nodal values of transverse displacement are set to zero, the mode is no longer present (i.e., the global stiffness matrix is not rank deficient). However, when the element thickness-to-length ratio increases beyond a certain value, the mode is only weakly coupled to the boundary and may become manifest in certain singular situations. This is illustrated in Fig. 5.3.20, where significant oscillations in transverse displacement may be noted. (In Fig. 5.3.20, r denotes the radial

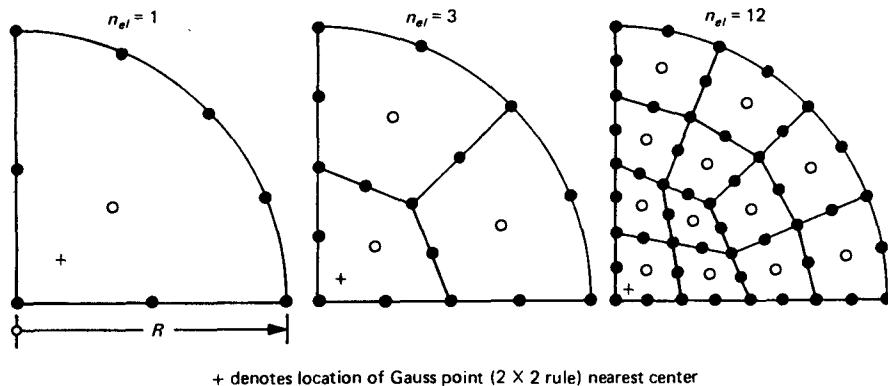
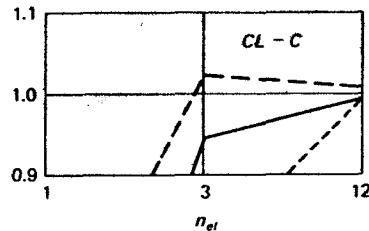
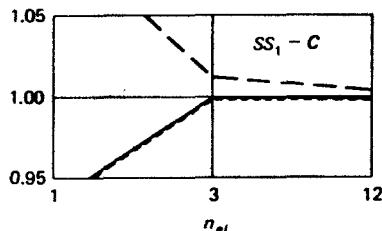
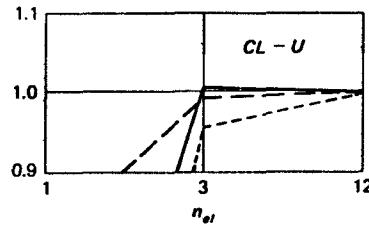
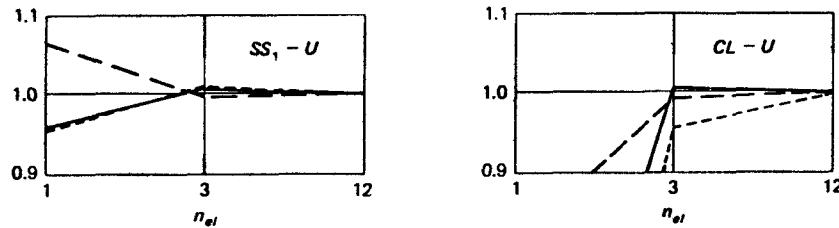
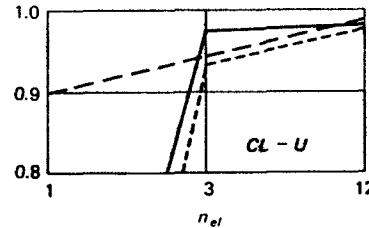
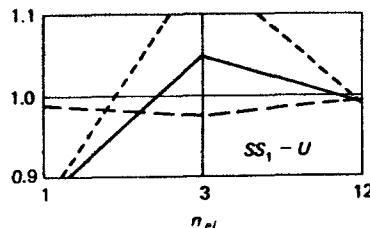


Figure 5.3.18 Circular plate. Due to symmetry, only one quadrant is discretized.

Center displacement normalized with
respect to thin plate theory



Moment at Gauss point nearest center,
normalized by center moment of thin
plate theory



----- serendipity
——— heterosis
- - - - - Lagrange

SS_1 = simply supported boundary conditions
 CL = clamped boundary conditions
 U = uniform load
 C = concentrated load

Figure 5.3.19 Convergence study for thin circular plate.

coordinate, D is the flexural rigidity, and P is the amplitude of the uniform load.) The heterosis element, having correct rank, does not exhibit this deficiency. (For this problem, the serendipity results are almost identical to the heterosis results. At points at which the plot symbols tended to overlap, we included only the heterosis symbol.)

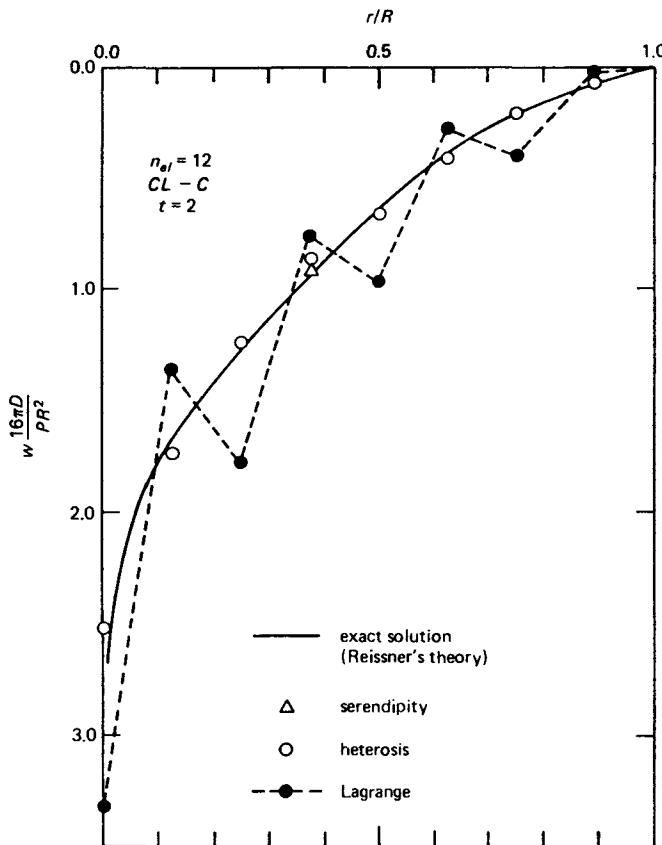


Figure 5.3.20 Comparison of transverse displacements for thick circular plate.

5.3.8 T1: A Correct-rank, Four-node Bilinear Element

A special procedure for interpolating transverse shear strains is used in element T1 (Hughes-Tezduyar [17]) to achieve correct rank. The technique falls within the framework of the \bar{B} -approach, described in the preceding chapter, as will be shown shortly.

Pertinent geometric and kinematic data are shown in Fig. 5.3.21. Note that the direction vectors have unit length (e.g., $\|e_{11}\| = 1$, etc.). Let w_a and θ_a denote the transverse displacement and rotation vector, respectively, associated with node a . Throughout, a subscript b will equal $a + 1$ modulo 4. That is

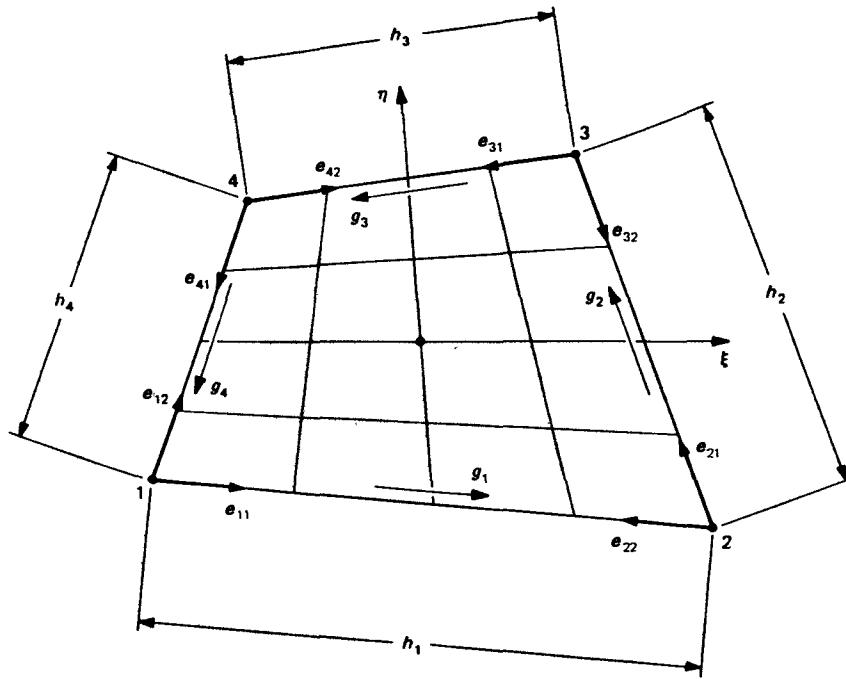


Figure 5.3.21 Geometric and kinematic data for the four-node, bilinear, quadrilateral element, $T1$.

| a | b | |
|-----|-----|--|
| 1 | 2 | |
| 2 | 3 | |
| 3 | 4 | |
| 4 | 1 | |

(5.3.15)

The definition of the element shear strains is facilitated by the following steps:

1. For each element side, define a shear strain component, located at the midpoint in a direction parallel to the side, viz.,

$$g_a = \frac{w_b - w_a}{h_a} - e_{ai} \cdot \frac{\theta_b + \theta_a}{2} \quad (5.3.16)$$

The g_a 's are schematically illustrated in Fig. 5.3.21.

2. For each node, define a shear strain vector (see Fig. 5.3.22 for a geometric interpretation of this process).

$$\gamma_b = \gamma_{b1} e_{b1} + \gamma_{b2} e_{b2} \quad (5.3.17)$$

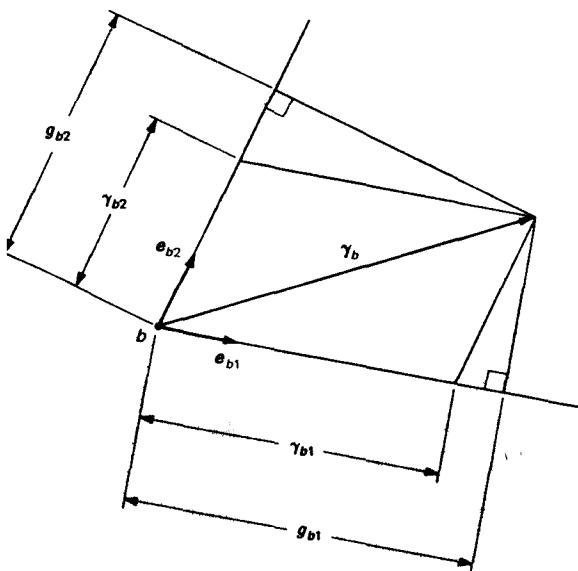


Figure 5.3.22 Definition of nodal transverse shear strain vector.

$$\gamma_{b2} = (1 - \alpha_b^2)^{-1}(g_{b2} - g_{b1}\alpha_b) \quad (5.3.18)$$

$$\gamma_{b1} = (1 - \alpha_b^2)^{-1}(g_{b1} - g_{b2}\alpha_b) \quad (5.3.19)$$

$$\alpha_b = e_{b1} \cdot e_{b2} \quad (5.3.20)$$

$$g_{b1} = g_b \quad (5.3.21)$$

$$g_{b2} = -g_a \quad (5.3.22)$$

3. Interpolate the nodal values by way of the bilinear shape functions (N_a 's):

$$\boldsymbol{\gamma} = \sum_{a=1}^4 N_a \boldsymbol{\gamma}_a \quad (5.3.23)$$

Remarks

1. If the nodal transverse displacements and rotations are specified to consistently interpolate a constant transverse shear strain field—for example, $\bar{\gamma}$ —then the preceding steps will result in $\boldsymbol{\gamma} = \bar{\boldsymbol{\gamma}}$. That is, *constant transverse shear deformation modes are exactly representable in the general quadrilateral geometry*.

2. In the rectangular configuration, the strains take on the following form (we assume the origin of coordinates coincides with the element center):

$$\gamma_1(x_1, x_2) = w_{,1}(0, 0) - \theta_1(0, 0) + x_2[w_{,12} - \theta_{1,2}(0, 0)] \quad (5.3.24)$$

$$\gamma_2(x_1, x_2) = w_{,2}(0, 0) - \theta_2(0, 0) + x_1[w_{,21} - \theta_{2,1}(0, 0)] \quad (5.3.25)$$

where $w_{,12} = w_{,21} = \text{constant}$. In this case the linear variations of γ_1 with x_2 and γ_2 with x_1 may be clearly seen. Note that there are four scalar transverse shear strain

modes. (This may be concluded in general from Steps 1 to 3, which amount to an interpolation of the four scalar parameters g_1, g_2, g_3 , and g_4 .) These modes include the two constant transverse shear modes and the hourglass and in-plane twist modes, thus enabling the element to achieve correct rank. In the rectangular configuration, the transverse shear strain variation is equivalent to the selective integration scheme of MacNeal's QUAD4 [18]. The generalizations to quadrilateral configurations differ somewhat.

3. The constraint ratio (as defined in Sec. 5.3.2) for the present element is $\frac{3}{2}$, the optimal value. This figure is arrived at by observing that the tangential component of transverse shear strain is *continuous* across element boundaries, effectively halving the number of ostensible constraints. (Spilker and Munir [19–21] have proposed an alternative constraint-counting measure for plates called a *rotational constraint index*.)

Implementation

All aspects of the element formulation are identical to the standard four-node, bilinear quadrilateral except that in place of the usual transverse shear strain–nodal displacement matrix, \mathbf{B}^s (see (5.2.55), (5.2.57), (5.2.59), (5.2.62), and (5.2.68)), we need to use $\bar{\mathbf{B}}^s$, which is defined as follows (recall the relation between subscripts a and b ; see (5.3.15)):

$$\bar{\mathbf{B}}^s = [\bar{\mathbf{B}}_1^s \bar{\mathbf{B}}_2^s \bar{\mathbf{B}}_3^s \bar{\mathbf{B}}_4^s] \quad (5.3.26)$$

$$\bar{\mathbf{B}}_b^s = [\bar{\mathbf{B}}_{b1}^s \bar{\mathbf{B}}_{b2}^s \bar{\mathbf{B}}_{b3}^s], \quad 1 \leq b \leq 4 \quad (5.3.27)$$

$$\bar{\mathbf{B}}_{b1}^s = h_a^{-1} \mathbf{G}_a - h_b^{-1} \mathbf{G}_b \quad (5.3.28)$$

$$\bar{\mathbf{B}}_{b2}^s = \frac{e_{b2}^1 \mathbf{G}_a - e_{b1}^1 \mathbf{G}_b}{2} \quad (5.3.29)$$

$$\bar{\mathbf{B}}_{b3}^s = \frac{e_{b2}^2 \mathbf{G}_a - e_{b1}^2 \mathbf{G}_b}{2} \quad (5.3.30)$$

$$\mathbf{G}_a = (1 - \alpha_a^2)^{-1} N_a (\mathbf{e}_{a1} - \alpha_a \mathbf{e}_{a2}) - (1 - \alpha_b^2)^{-1} N_b (\mathbf{e}_{b2} - \alpha_b \mathbf{e}_{b1}) \quad (5.3.31)$$

$$\mathbf{e}_{b1} = \begin{cases} e_{b1}^1 \\ e_{b1}^2 \end{cases} \quad (5.3.32)$$

⋮

Two-by-two Gaussian quadrature is used to integrate all element contributions.

Numerical Examples

T1 is compared with the four-node reduced integration elements S1 and U1, described previously and, in one case, the “twisted ribbon,” comparison is made with an element proposed by Robinson [22], dubbed LORA, and MacNeal's QUAD4 [18].

Thin Square and Rectangular Plates

In the cases studied, the differences between S1 and T1 are indiscernible on the scale of the plots. Square-plate results for T1 are the same as the S1 results of Fig. 5.3.5. The meshes for rectangular plates are shown in Figs. 5.3.23 and 5.3.24, and results are presented in Figs. 5.3.25 and 5.3.26. These problems test the response of the elements to changes in planar aspect ratio. As can be seen, there is some deterioration of accuracy with planar aspect ratio—a common but not well-understood phenomenon for virtually all finite elements.

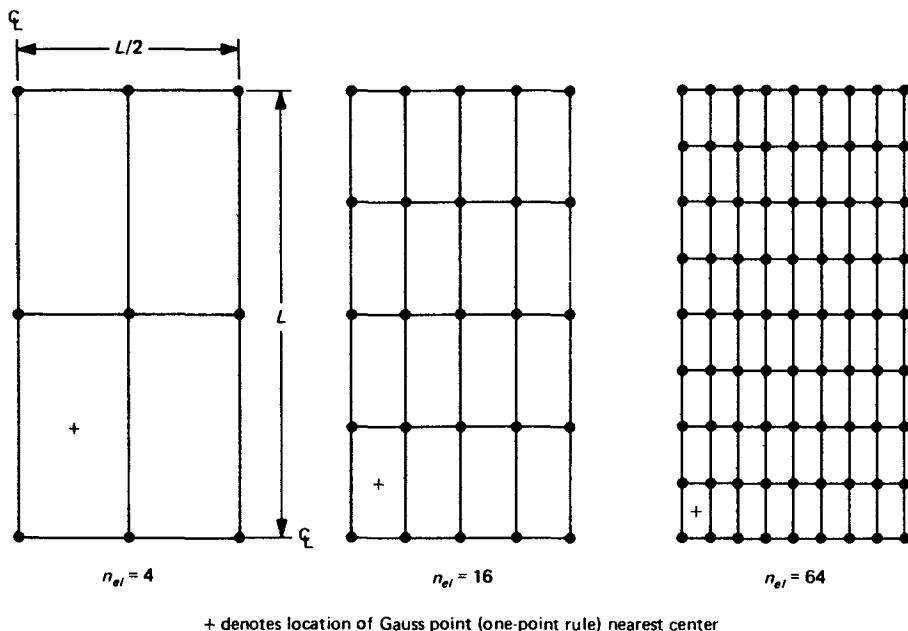


Figure 5.3.23 Rectangular plate meshes (aspect ratio = 2). Due to symmetry, only one quadrant is discretized.

Thin Circular Plate

These problems test the behavior of the elements in nonrectangular configurations. The meshes are the same as shown in Fig. 5.3.6 and convergence results are presented in Fig. 5.3.27. In this case, T1 is generally the best performer, although all elements perform well.

Thin Rhombic Plate

The configuration and mesh are shown in Fig. 5.3.28. The length parameter $a = 100$. The plate is uniformly loaded and simply supported boundary conditions (SS_1) are employed. This problem is a difficult one, since there is a singularity at the obtuse vertex. The analytical solution reveals that the x_1 and x_2 bending moments have

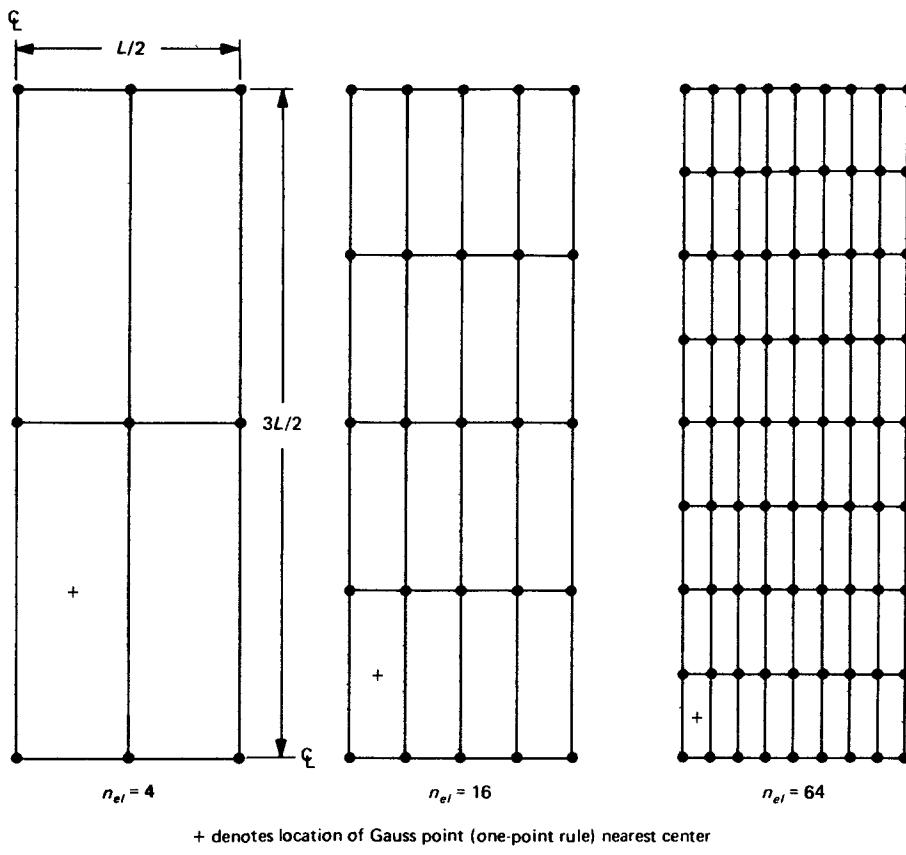


Figure 5.3.24 Rectangular plate meshes (aspect ratio = 3). Due to symmetry, only one quadrant is discretized.

opposite signs in the vicinity of the obtuse vertex. Many thin plate elements yield pathological results for this problem in that moments with the *same* sign are obtained (see Sec. 5.3.3 for a discussion and references). Moment results are presented in Fig. 5.3.29. The general trend for each element is correct. However, the elements have a tendency to oscillate somewhat, as may be seen. The worst oscillations are produced by *U1*. Considering that the mesh is not biased to favor the singularity and that the problem is a numerically difficult one, the accuracy of the results obtained for *S1* and *T1* is considered to be fairly good.

Thick Circular Plate

This problem was considered previously for the heterosis, Lagrange, and serendipity elements (see Fig. 5.3.20). In the present case, the 48-element mesh shown in Fig. 5.3.6 was employed. The singularity gives rise to almost identical oscillatory patterns for the rank-deficient elements *S1* and *U1*, as may be seen in Fig. 5.3.30. On the other hand, element *T1* produces very accurate results.

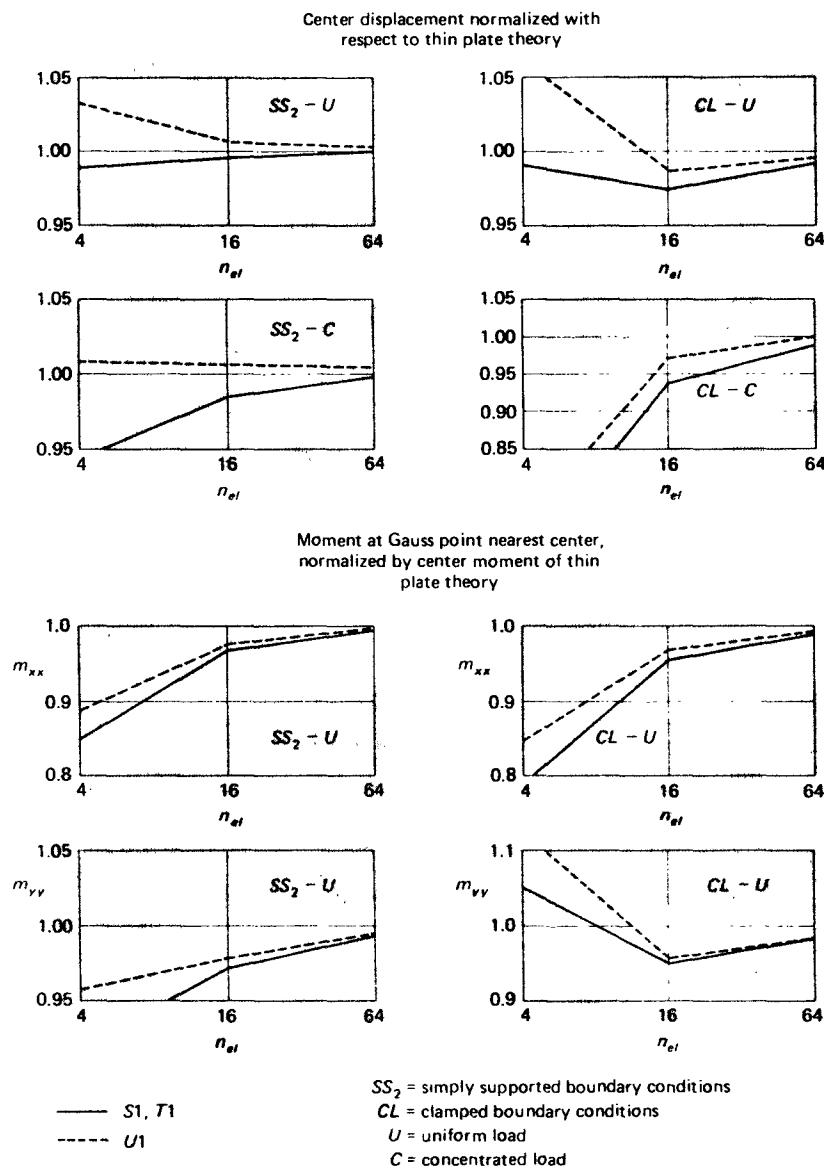


Figure 5.3.25 Convergence study for thin rectangular plate (aspect ratio = 2).

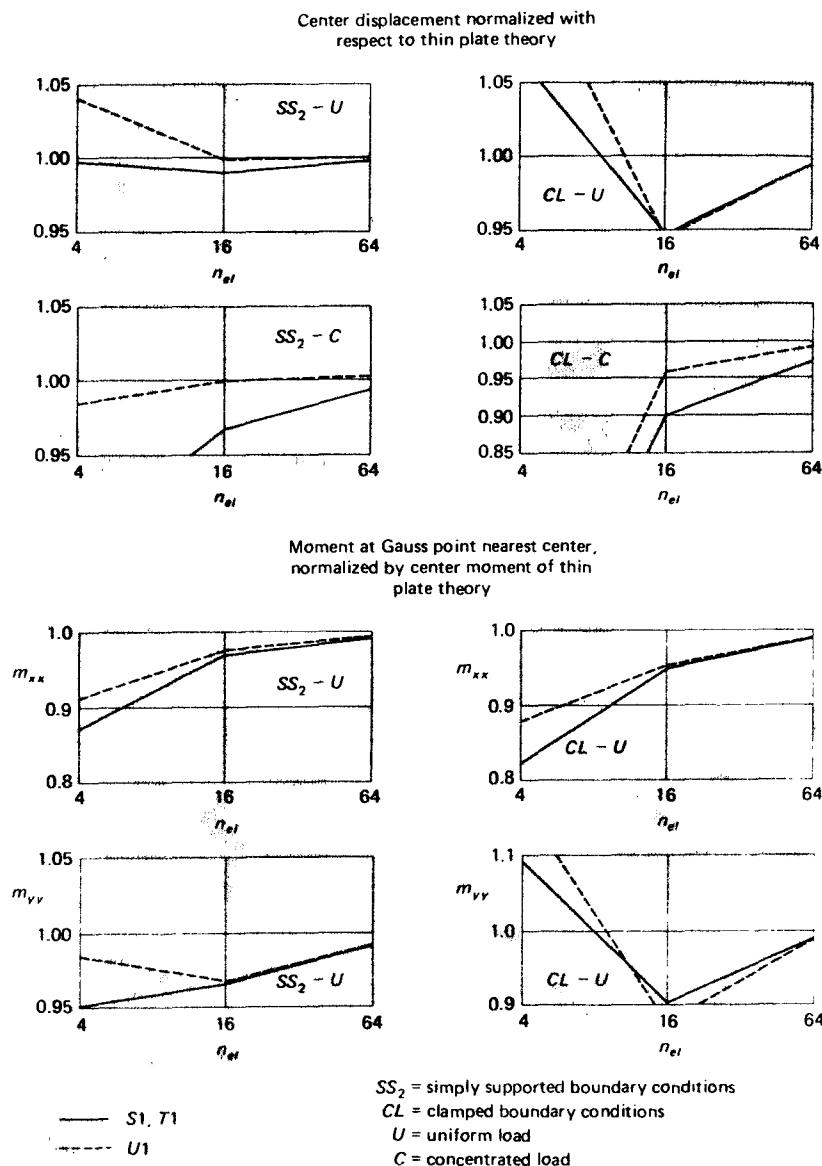


Figure 5.3.26 Convergence study for thin rectangular plate (aspect ratio = 3).

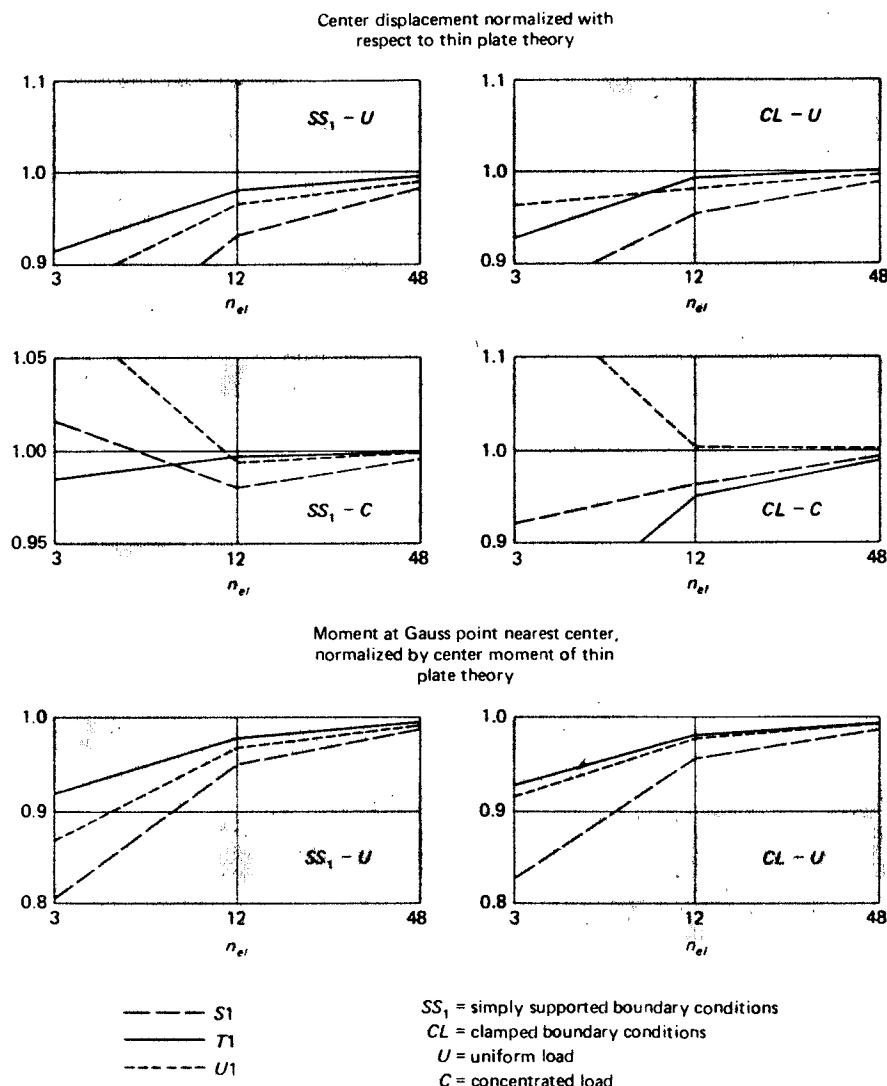


Figure 5.3.27 Convergence study for thin circular plate.

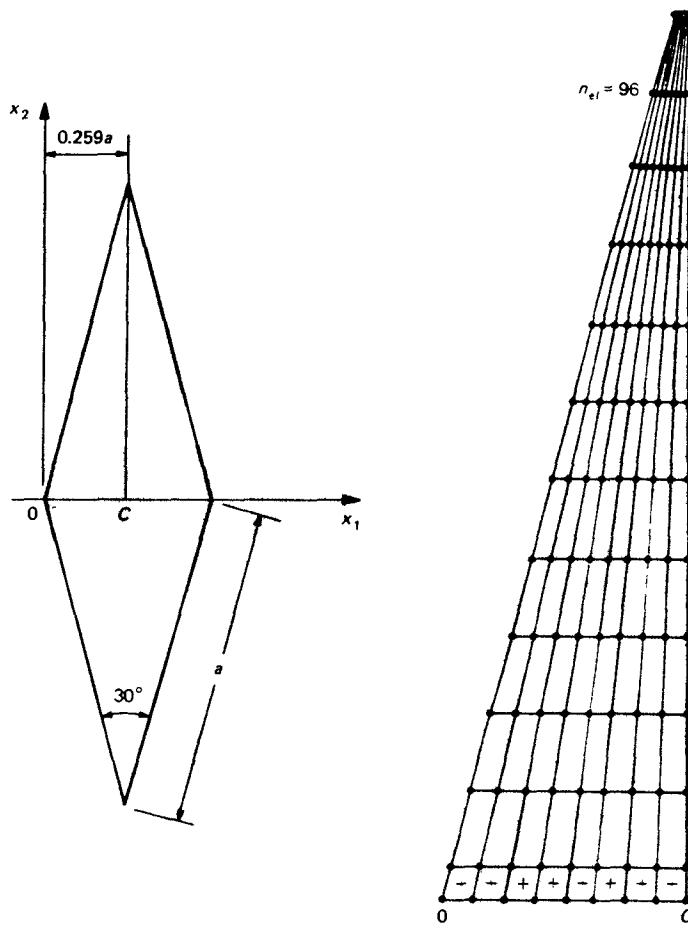


Figure 5.3.28 Rhombic plate mesh. Due to symmetry, only one quadrant is discretized.

Twisted Ribbon

Configurations, data and results for this problem are shown in Fig. 5.3.31. In each analysis, only one element is employed. Robinson [22] has proposed this as a critical single element test for plate-bending elements. Comparisons are made with data presented in [22] for Robinson's element, LORA, and MacNeal's QUAD4 [18].

For cases A and B (fully fixed boundary), comparison is made with respect to a benchmark analysis, reported upon in [22], involving 16 high-precision elements. As may be seen, the results for *T*1 are superior to the results for both LORA and QUAD4. Furthermore, no deterioration with increasing aspect ratio is detected. For this case, elements *S*1 and *U*1 exhibit pathological behavior due to rank deficiency (not shown).

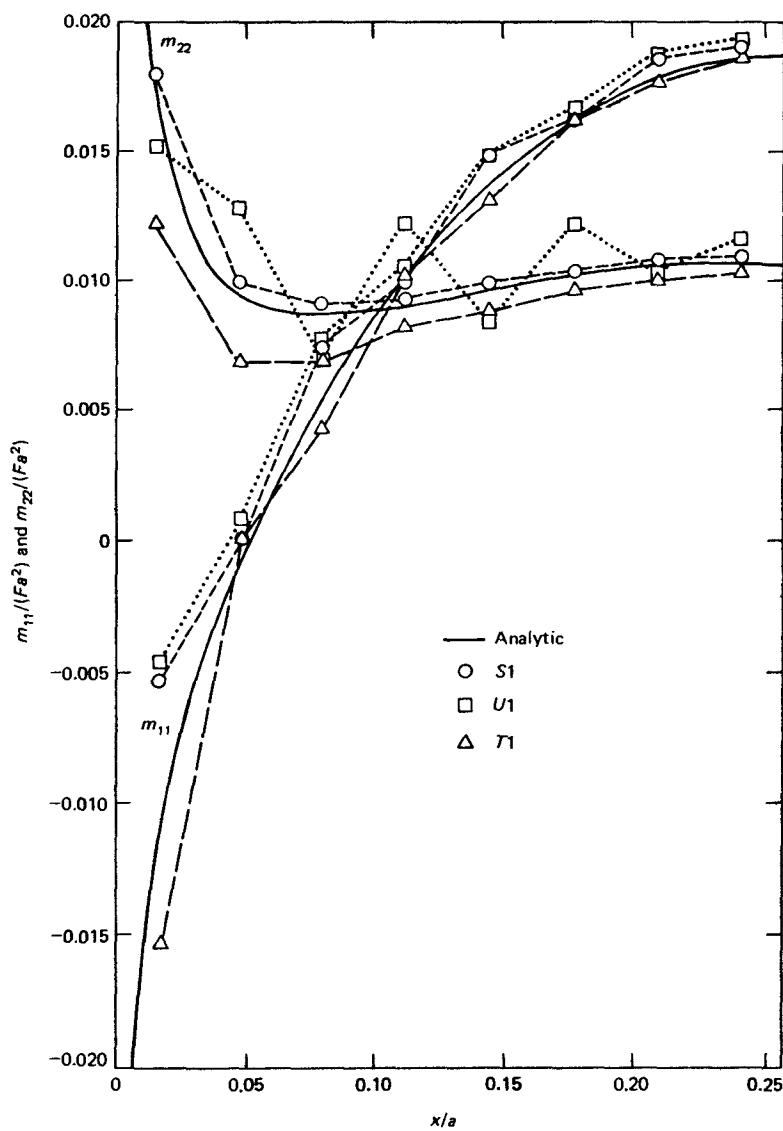


Figure 5.3.29 Bending moments for rhombic plate.

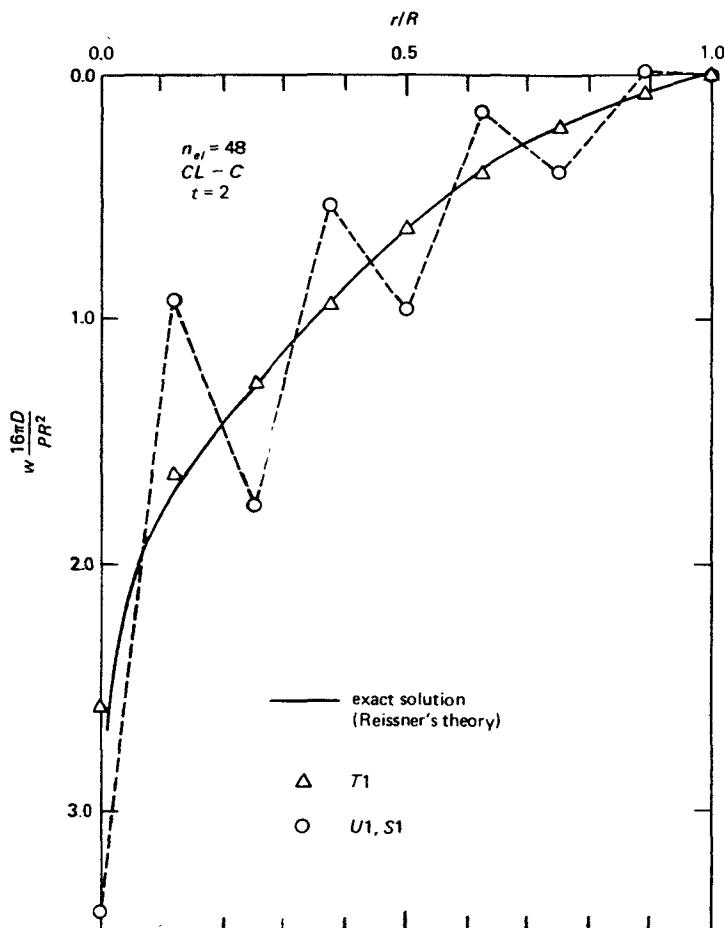


Figure 5.3.30 Displacement results for thick circular plate.

If only half the domain is modeled and antisymmetrical boundary conditions are enforced (cases C and D), the exact solution is one of pure twist. For these cases, S_1 and T_1 yield exact solutions, whereas U_1 still behaves pathologically (not shown).

Remarks

1. A rationale for the development of T_1 , based upon the *Kirchhoff mode concept*, is presented in [17]. The basic idea emanates from [18].
2. MacNeal [23] has more recently presented a new version of QUAD4 in which the definition of the transverse shear strain field has essential features in common with T_1 .

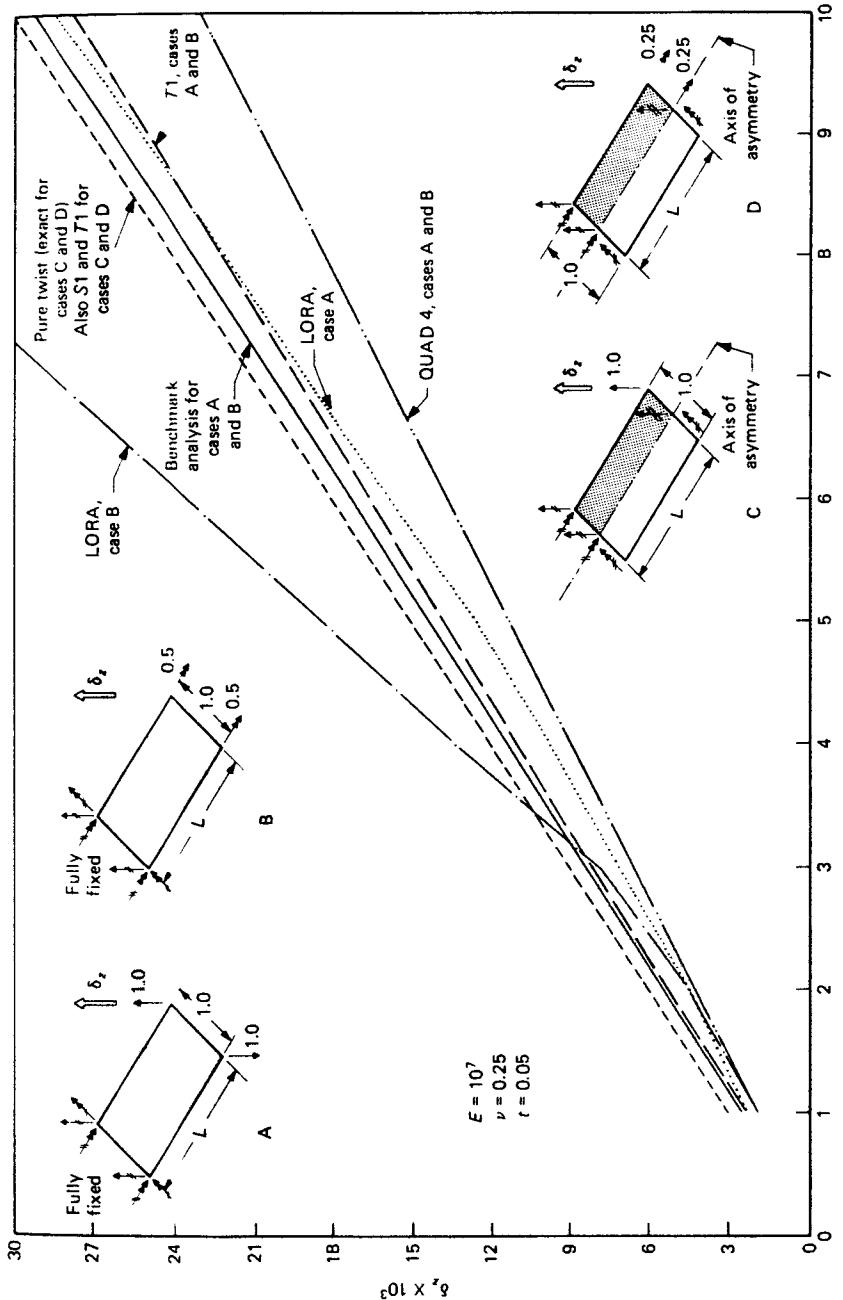


Figure 5.3.31 Displacement results for the twisted ribbon. One element used in all cases.

5.3.9 The Linear Triangle

A three-node triangle employing linear shape functions for w and θ_a exhibits severe shear locking under normal circumstances. The technique used in the previous section to define the transverse shear strain field for $T1$ may also be used in this case to alleviate locking while maintaining correct rank [24]. The development follows (5.3.15) through (5.3.23) closely. The only changes involve the nodal indices, which this time are defined by the relation

| a | b |
|-----|-----|
| 1 | 2 |
| 2 | 3 |
| 3 | 1 |

(5.3.33)

which replaces (5.3.15), and the summation in (5.3.23) is taken over $a = 1, 2, 3$ wherein the N_a 's represent linear (instead of bilinear) shape functions. Figure 5.3.32 depicts element geometric and kinematic quantities. Implementation follows along the same lines as for $T1$ (see (5.3.26) through (5.3.32)). The only changes involve replacing (5.3.26) by

$$\bar{\mathbf{B}}^s = [\bar{\mathbf{B}}_1^s \bar{\mathbf{B}}_2^s \bar{\mathbf{B}}_3^s] \quad (5.3.34)$$

and limiting $b \leq 3$ in (5.3.27).

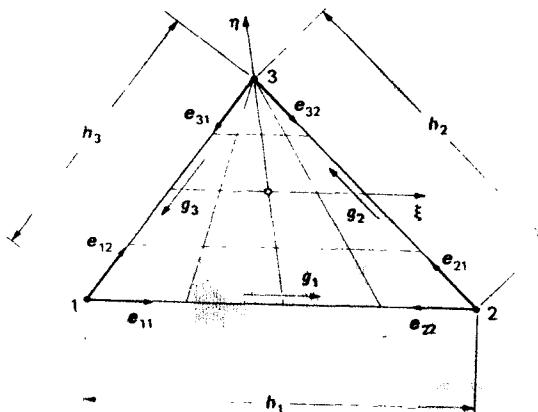


Figure 5.3.32 Geometric and kinematic data for the three-node linear triangular element.

Numerical Examples

Two different quadrature treatments of the linear triangle have been used: one-point centroidal quadrature and 2×2 Gauss quadrature in the ξ, η -system (see Fig. 5.3.32), which is exact in the present circumstances. (Note that centroidal quadrature is not the same as one-point Gauss quadrature.) A single triangle possesses correct rank (i.e., 6) when integrated exactly, but possesses one spurious mechanism when

underintegrated by one-point centroidal quadrature. However, in any assemblage of two or more elements, the spurious mechanism disappears, and thus it does not appear to be of serious consequence in practical computing.

Calculations were performed for simply supported, thin, square and circular plates subjected to uniform loads. The edge length of the square plate and radius of the circular plate were taken to be 10.0 and 5.0, respectively. In each case, due to symmetry, only one quadrant of the plate was discretized. Meshes are depicted in Figs. 5.3.33 and 5.3.34. Note that the cross-diagonal meshes involve approximately twice the number of unknowns as the other mesh types. The SS_1 boundary condition was used in each case. The quadrature treatment of the consistent load was the same as used for the stiffness.

Numerical results for the cases studied are presented in Tables 5.3.2 and 5.3.3. Moment results were obtained at the centroids of the triangular elements nearest the plate center. When the vertices of two triangles coincided with the plate center, moments were averaged over the two elements. It is immediately apparent from Table 5.3.2 that exact quadrature behaves very poorly for mesh types A and B.

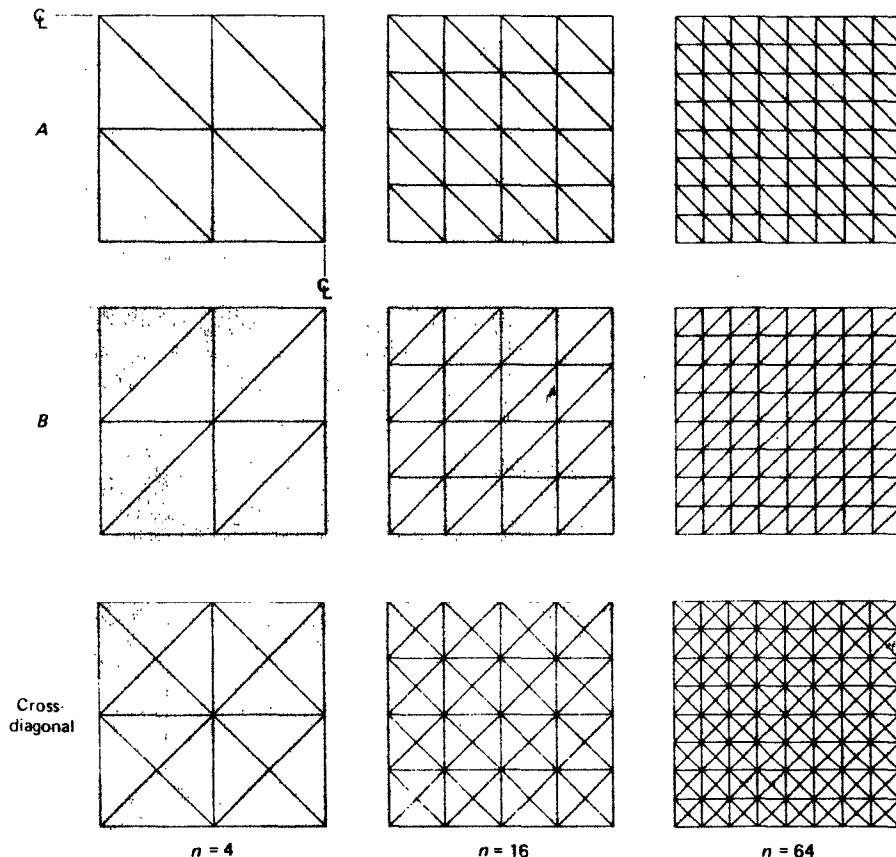


Figure 5.3.33 Square plate meshes. Due to symmetry, only one quadrant is discretized.

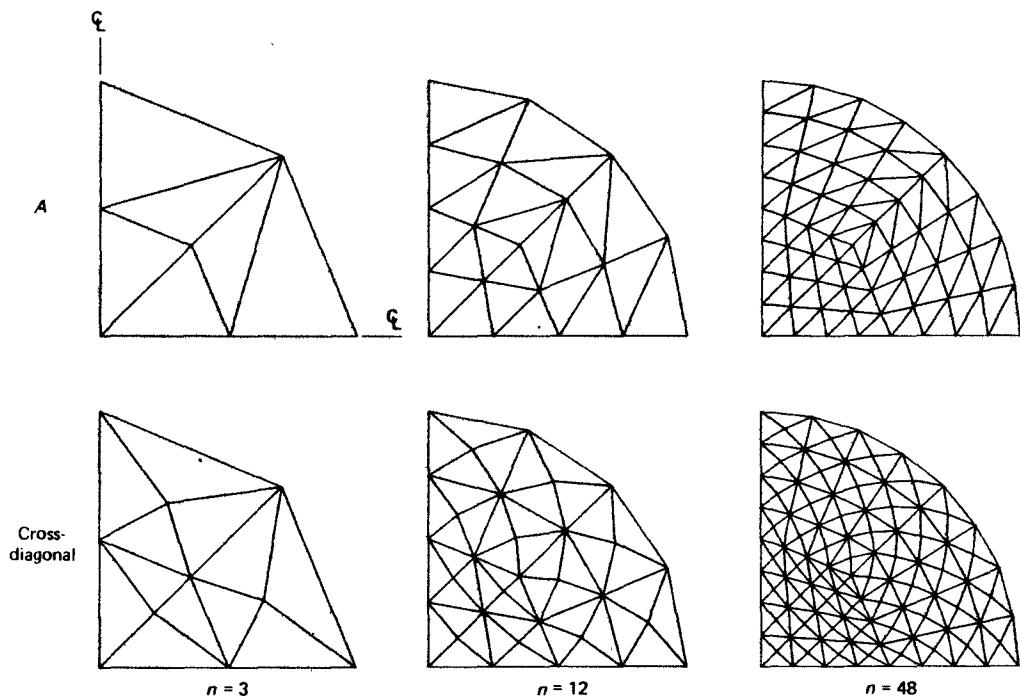


Figure 5.3.34 Circular plate meshes. Due to symmetry, only one quadrant is discretized.

TABLE 5.3.2 Center Displacement and Moment for Simply Supported, Thin, Square Plate Subjected to Uniform Load

a. Displacement

| <i>n</i> | One-point centroidal quadrature | | | Exact quadrature | | |
|----------|---------------------------------|----------|----------------|------------------|----------|----------------|
| | <i>A</i> | <i>B</i> | cross-diagonal | <i>A</i> | <i>B</i> | cross-diagonal |
| 4 | 0.681 | 0.883 | 0.912 | 0.681 | 0.002 | 0.912 |
| 16 | 0.784 | 0.989 | 0.978 | 0.773 | 0.036 | 0.978 |
| 64 | 0.947 | 0.999 | 0.994 | 0.827 | 0.363 | 0.994 |

b. Moments

| <i>n</i> | One-point centroidal quadrature | | | Exact quadrature | | |
|----------|---------------------------------|----------|----------------|------------------|----------|----------------|
| | <i>A</i> | <i>B</i> | cross-diagonal | <i>A</i> | <i>B</i> | cross-diagonal |
| 4 | 0.555 | 0.904 | 0.919 | 0.555 | 0.002 | 0.919 |
| 16 | 0.564 | 1.127 | 0.979 | 0.539 | 0.036 | 0.979 |
| 64 | 0.835 | 1.098 | 0.996 | 0.580 | 0.386 | 0.996 |

TABLE 5.3.3 Center Displacement and Moment for Simply Supported, Thin, Circular Plate Subjected to Uniform Load

| n | A | | Cross-diagonal | |
|----|--------------|--------|----------------|--------|
| | Displacement | Moment | Displacement | Moment |
| 3 | 0.703 | 0.576 | 0.927 | 0.885 |
| 12 | 0.912 | 0.878 | 0.981 | 0.976 |
| 48 | 0.948 | 0.975 | 0.976 | 0.986 |

Results presented are for one-point centroidal quadrature.

On the other hand, for the cross-diagonal meshes, exact quadrature yields quite satisfactory results. One-point centroidal quadrature represents some improvement for mesh types A and B, but still yields moments that are not satisfactory. Note that one-point quadrature for the cross-diagonal pattern yields identical results to those obtained by exact quadrature.

It is quite clear from the results of Table 5.3.2 that only the cross-diagonal meshes yield consistently satisfactory results. Furthermore, there is no advantage to exact quadrature. The economic superiority of one-point quadrature makes it the obvious choice for practical use.

The results for the circular plate, presented in Table 5.3.3, are satisfactory for both mesh patterns studied.

Results for cross-diagonal meshes are compared with quadrilateral T1 elements in Fig. 5.3.35. The moments for T1 are computed at the Gauss point (2×2 rule) nearest the center of the plate. As may be seen, the displacement results in this case favor T1, whereas the moment results for the linear triangle are superior. However, the sampling points for moments is closer to the center for the triangles than for T1, and this favors the triangle results, especially on coarse meshes.

Remarks

1. MacNeal [25] has developed a linear triangular bending element, TRIA3, for the MSC (MacNeal-Schwendler Corporation) version of the NASTRAN program. This element is very similar to the linear triangle presented herein, except a “residual bending flexibility” is introduced to further improve behavior. The concept of residual bending flexibility, introduced by MacNeal in [18], will be illustrated in Section 5.5 in the context of the linear beam element.

2. A similar idea involving a “c-factor” modification [26], has been employed by Garnet et al. [27] to alleviate the tendency to lock for the standard three-node linear triangle (This element does not employ special interpolation of the transverse shear strain.)

Discussion

Numerical results for the exactly integrated linear triangle described in this section are somewhat erratic for certain mesh configurations. However, the cross-diagonal pattern

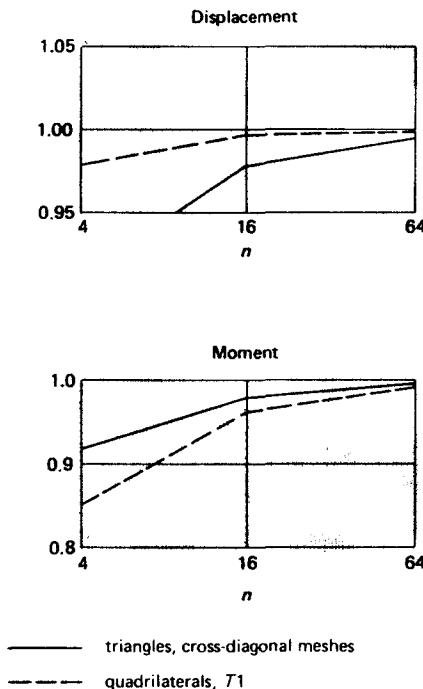


Figure 5.3.35 Comparison of center displacement and moment for triangular and quadrilateral elements; simply supported, thin, square plate subjected to uniform load.

was found to be successful for both exact and reduced integration (i.e., one-point centroidal quadrature). Although one-point quadrature results in rank deficiency for a single element, this is not a practical detriment, since the rank deficiency disappears in an assemblage of two or more elements. Due to the greater economy of one-point quadrature, it appears the obvious choice for practical use.

Comparison of results with those for the bilinear quadrilateral T_1 suggests that linear triangles in the cross-diagonal pattern may be competitive. In the comparison, cross-diagonal triangulated meshes required approximately twice the number of unknowns as for bilinear quadrilaterals. On the other hand, the center-node degrees of freedom are easily eliminated via elementwise static condensation, and the one-point evaluation over each triangle is amenable to very efficient element programming.

Techniques described in Remarks 1 and 2 have been introduced to improve the performance of linear triangular bending elements. A firm theoretical foundation is, however, not yet available, but it is to be hoped that this will be forthcoming so that a coherent exposition of these ideas may be presented in the near future.

5.3.10 The Discrete Kirchhoff Approach

In the discrete Kirchhoff approach, thin plate elements are developed by insisting that the Kirchhoff-theory constraint of zero transverse shear strains be satisfied at a discrete number of points within the element. To illustrate the approach, we will consider the development of the 12-degree-of-freedom, four-node quadrilateral DKQ (discrete Kirchhoff quadrilateral [29]):

i. The domain of the element is a straight-edged quadrilateral. The rotations are initially defined by

$$\boldsymbol{\theta} = \sum_{a=1}^8 N_a \boldsymbol{\theta}_a \quad (5.3.35)$$

where the N_a 's are the eight-node serendipity shape functions (see Sec. 3.7). Thus we begin with 16 rotational degrees of freedom. The midside degrees of freedom are eventually eliminated.

ii. Let \mathbf{n} and \mathbf{s} denote normal and tangential unit vectors along a side, with orientation as defined in Fig. 5.3.1. The normal component of $\boldsymbol{\theta}$ is required to vary linearly along each side, i.e.,

$$\mathbf{n} \cdot \boldsymbol{\theta}_c = \frac{\mathbf{n} \cdot (\boldsymbol{\theta}_a + \boldsymbol{\theta}_b)}{2} \quad (5.3.36)$$

where $c = a + 4$, $a = 1, 2, 3, 4$, and $b = 2, 3, 4, 1$, respectively. This amounts to 4 constraints.

iii. The transverse displacement is defined *only* along the element boundary. It is assumed to vary cubically along each edge, according to Hermite interpolation (see Sec. 1.16). Thus nodal values of w and $w_{,a}$ are required to define the edge displacement. This introduces 12 more degrees of freedom, for a total of 28.

iv. Kirchhoff constraints are imposed as follows:

1. Corner nodes:

$$w_{,a} = \boldsymbol{\theta}_a \quad (\text{i.e., } \gamma_a = 0) \quad (5.3.37)$$

2. Midside nodes³

$$w_{,s} = s_a \boldsymbol{\theta}_a \quad (\text{i.e., } s_a \gamma_a = 0) \quad (5.3.38)$$

where $w_{,s} = s_a w_{,a}$ (sum). Equation (5.3.37) represents 8 constraints and (5.3.38) gives 4 more. These 12 Kirchhoff constraints, and the 4 normal rotation conditions under (ii), enable reduction to 12 degrees of freedom, namely, the corner node displacements and rotations. Furthermore, explicit formulas may then be obtained for the rotations:

$$\boldsymbol{\theta}_a = \sum_{p=1}^{12} N_{ap} d_p^e \quad (5.3.39)$$

where

$$d_p^e = \begin{cases} w_a & p = 3a - 2 \\ \theta_{1a} & p = 3a - 1 \\ \theta_{2a} & p = 3a \end{cases} \quad (5.3.40)$$

³ A handy formula for $w_{,s}$ at the midside nodes is given by:

$$w_{,s_c} = \frac{3}{2h_a} (w_b - w_a) + \frac{1}{4} (w_{,b} - w_{,a})$$

where $w_{,s_c} = w_{,s}(x_c)$; the nodal indexing is as defined under (ii); and h_a is the edge length (see Fig. 5.3.21).

(Expressions for the shape functions, N_{ap} , are somewhat lengthy. The interested reader is referred to the original sources for further details.)

v. In the development of the element stiffness, the transverse shear terms are simply ignored. Thus the bending stiffness is completely defined by the derivatives of the rotation interpolations derived under (iv). (See (5.2.56) through (5.2.62)).

Remarks

1. By virtue of the fact that the tangential shear strain component vanishes at three distinct points along each edge, it vanishes identically along the element boundary. (This follows from the observation that it varies quadratically without imposition of the Kirchhoff constraints.)

2. The exact integration of the element stiffness (in the rectangular configuration) requires 3×3 Gaussian quadrature. However, it is observed that the 2×2 rule maintains correct rank and so, due to its greater economy, it is recommended in practice [29].

3. No internal interpolation for transverse displacement is defined. This creates ambiguities in the correct definition of consistent transverse applied forces. (The same is true for the definition of element inertial properties in dynamics.) In [29], two ad hoc possibilities are studied: bilinear variation, and an incomplete quartic scheme. Both seem to perform adequately in practice.

4. It is asserted that convergence to the thin plate solution is to be expected because the transverse shear stiffness is neglected and because the Kirchhoff hypothesis is satisfied (tangentially) along the boundary.

5. A triangular element, "DKT," employing virtually identical concepts, was the historical predecessor of DKQ (see [30–32]). References [31, 32] should be consulted for additional discussion about discrete Kirchhoff elements and pertinent references to the literature.

In addition to discrete Kirchhoff constraints, elements have been derived that also employ integral Kirchhoff constraints (e.g., the area, or boundary, mean shear strain may be required to vanish). Most prominent among these are perhaps Irons' SEMILOOF [33] and Lyons' ISOFLEX [34] elements. SEMILOOF is rather complicated; however, one of Lyons' elements is a four-node 12-degree-of-freedom quadrilateral. Crisfield [35] has developed a modified version of Lyons' four-node element in which the Kirchhoff constraints are given in explicit algebraic form, thus avoiding cumbersome element-level calculations.

A somewhat related approach has been employed by Tessler and Hughes in the development of a four-node quadrilateral [36] and three-node triangle [37]. For the quadrilateral, bilinear rotations are employed, and, initially, an eight-node serendipity interpolation is used for transverse displacement. The midside transverse-displacement degrees of freedom are removed by constraining the tangential component of transverse shear strain to be constant along each edge. The triangle is developed along similar lines, starting with linear rotations and quadratic transverse displacements. By virtue of the fact that transverse-shear deformation modes are retained, these elements differ from discrete-Kirchhoff elements in that they are applicable to moderately thick

as well as thin plates. The good behavior of the elements requires a “c-factor,” or “residual bending flexibility,” type of modification. See [36, 37] for further details.

5.3.11 Discussion of Some Quadrilateral Bending Elements

*T*1 and the latest version of QUAD4 [23] have much in common. The treatment of transverse shear is equivalent in the rectangular configuration and differs only slightly in the general quadrilateral configuration. Three additional devices are employed in an effort to further improve the behavior of QUAD4:

i. The shearing properties of the plate are modified to account for a “residual bending flexibility” correction. The concept of residual bending flexibility is an intriguing one. In simple cases, such as beams, a clear exposition of the basic idea is possible and its beneficial effect is obvious (see Sec. 5.5 and [18]). However, for arbitrary geometries and in envisioning extension to nonlinear cases, it becomes less clear how to proceed systematically. This modification tends to make QUAD4 more flexible than *T*1, which is a step in the right direction because *T*1 tends to err on the stiff side.

ii. The component of twisting curvature, κ_{12} , at the 2×2 Gauss points is replaced by $2\kappa_{12}(\tilde{\xi}_l) - \kappa_{12}(0)$, where $\tilde{\xi}_l$, $l = 1, 2, 3, 4$, represent the 2×2 Gauss point coordinates. (κ_{12} is defined with respect to a locally defined Cartesian coordinate system, thus making sense invariantly.)

iii. A “tuning parameter” is introduced into the material properties to improve planar aspect ratio behavior.

The interested reader may consult [18, 23] for rationale and additional details. In some instances, these techniques lead to improved behavior compared with *T*1 (see Fig. 5.3.36), whereas, occasionally, the opposite is true (see Fig. 5.3.31). Whether or not the improvements engendered justify the additional complications is a somewhat subjective matter.

Recently, Bathe and Dvorkin [38] proposed a four-node plate element. It is immediately apparent that this element is identical to *T*1 for rectangular configurations. Results presented in Bathe and Dvorkin [39] indicated superiority of their element over *T*1 in rhombic configurations, but these results proved to be in error. It has been subsequently established that the Bathe-Dvorkin element is also identical to *T*1 in rhombic configurations.

In comparing DKQ with *T*1 and QUAD4, one must first note that DKQ is a thin plate element and thus does not apply to cases in which shear deformations must be accounted for. In most test cases the convergence of DKQ seems somewhat slower than *T*1 and QUAD4. However, in some cases it is superior. DKQ seems to be particularly insensitive to distortion, so overall the accuracy level appears commensurate. The shape functions are somewhat more complicated than the bilinear ones used for *T*1 and QUAD4. Nevertheless, 2×2 quadrature suffices, so the cost of element calculations is probably not substantially different.

There is considerable interest in the development of methodology for controlling the spurious zero-energy modes of the one-point quadrature element (i.e., *U*1). Efficient methods have been proposed by Belytschko and Tsay [40] and Park and

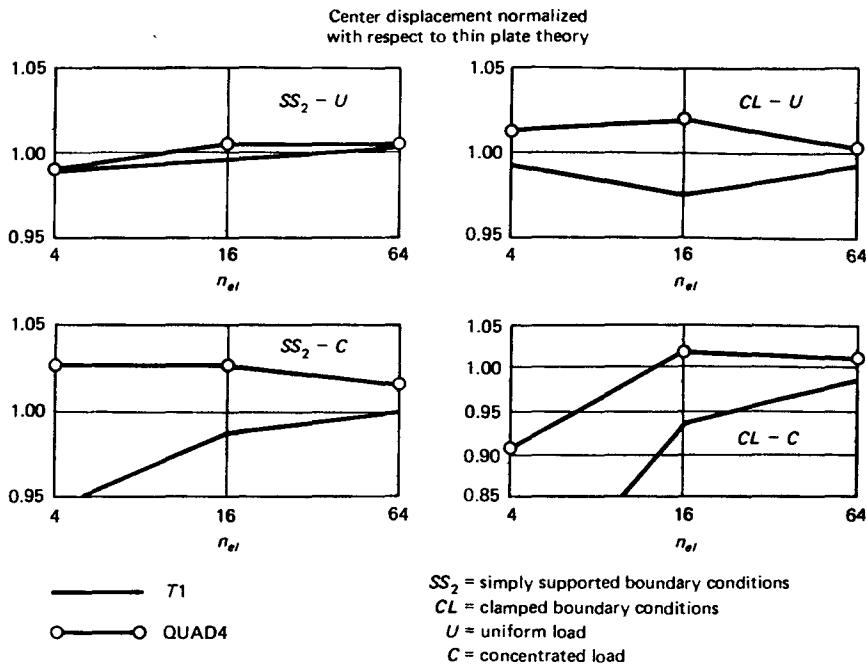


Figure 5.3.36 Comparison of $T1$ and $QUAD4$ for thin rectangular plate (aspect ratio = 2).

Flaggs [41]. Several new improved, nine-node elements have also been proposed recently; see Belytschko, Ong, and Liu [42], Crisfield [43], and Huang and Hinton [44].

Plate element design is still an active area of research and we anticipate many additional developments in the future.

5.4 BEAMS AND FRAMES

In this section we present a simple theory for finite element approximations of beam and frame structures. Throughout, Latin indices take on the values 1, 2, 3, and Greek indices take on the values 1, 2.

5.4.1 Main Assumptions

1. Domain. We assume from the start that the domain is divided into segments, or elements, interconnected at nodal points. An example of such a structure is depicted in Fig. 5.4.1.

$$\Omega = \bigcup_{e=1}^{n_e} \Omega^e$$

$$\Omega^e = \{(x_1^e, x_2^e, x_3^e) \in \mathbb{R}^3 \mid x_3^e \in [0, h^e], (x_1^e, x_2^e) \in A^e \subset \mathbb{R}^2\}$$

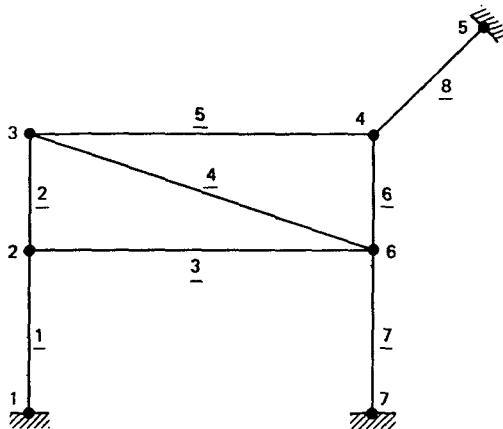


Figure 5.4.1 A structural model consisting of beam elements; $n_{el} = 8$, $n_{np} = 7$.

h^e = length of the e th beam element

A^e = cross-section area

We assume the (x_1^e, x_2^e, x_3^e) -axes are *locally* defined with respect to the beam segment and are principal axes, i.e.,

$$0 = \int_A x_1^e dA = \int_A x_2^e dA = \int_A x_1^e x_2^e dA$$

Note. To save some writing, we shall frequently omit the superscript e in the sequel.

2. $\sigma_{\alpha\beta} = 0$

3. $u_1(x_1, x_2, x_3) = w_1(x_3) - x_2 \theta_3(x_3)$

$u_2(x_1, x_2, x_3) = w_2(x_3) + x_1 \theta_3(x_3)$

$u_3(x_1, x_2, x_3) = w_3(x_3) - x_1 \theta_2(x_3) + x_2 \theta_1(x_3)$

Remarks

1. In Assumption 1, the beam is taken to be prismatic. However, no essential difficulty is encountered if we take A^e to be a function of x_3^e .

2. Assumption 2 is used in the constitutive equation to eliminate $\epsilon_{\alpha\beta}$.

3. The kinematic conditions of Assumption 3 do not include warping (i.e., plane sections remain plane).

The sign convention on θ_i follows the right-hand rule and is depicted in Fig. 5.4.2.

For convenience in the sequel, we recall the equations of classical linear elasticity theory:

$$\sigma_{ij,j} + \epsilon_i = 0 \quad \left. \begin{array}{l} \\ \end{array} \right\} \quad (5.4.1)$$

$$\sigma_{ij} = c_{ijkl} \epsilon_{kl} \quad \left. \begin{array}{l} \\ \end{array} \right\} \quad \text{in } \Omega \quad (5.4.2)$$

$$\epsilon_{ij} = u_{(i,j)} \quad \left. \begin{array}{l} \\ \end{array} \right\} \quad (5.4.3)$$

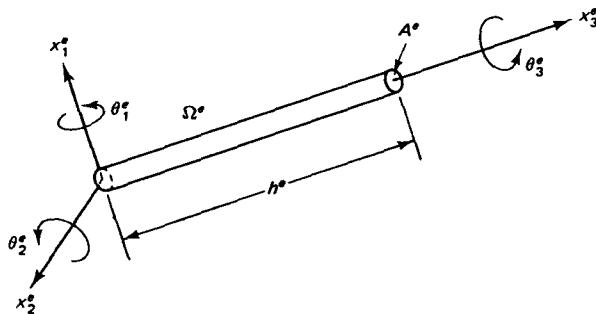


Figure 5.4.2

$$u_i = q_i \quad \text{on } \Gamma_q \quad (5.4.4)$$

$$\sigma_{ij} n_j = h_i \quad \text{on } \Gamma_k \quad (5.4.5)$$

5.4.2 Constitutive Equation

For simplicity, we assume the homogeneous isotropic case:

$$\sigma_{ij} = \lambda \epsilon_{kk} \delta_{ij} + 2\mu \epsilon_{ij} \quad (5.4.6)$$

For this case, Assumption 2 becomes

$$0 = \sigma_{\alpha\beta} = \lambda \epsilon_{kk} \delta_{\alpha\beta} + 2\mu \epsilon_{\alpha\beta} \quad (5.4.7)$$

Contracting this relation leads to

$$\epsilon_{\alpha\alpha} = \frac{-\lambda}{\lambda + \mu} \epsilon_{33} \quad (5.4.8)$$

Substituting this expression into the preceding one enables us to solve for $\epsilon_{\alpha\beta}$, viz.,

$$\epsilon_{\alpha\beta} = \frac{-\lambda \epsilon_{33}}{2(\lambda + \mu)} \delta_{\alpha\beta} \quad (5.4.9)$$

The expression for $\epsilon_{\alpha\alpha}$ can be used to obtain the classical expression for σ_{33} ,

$$\begin{aligned} \sigma_{33} &= \lambda \epsilon_{kk} + 2\mu \epsilon_{33} \\ &= \lambda \epsilon_{\alpha\alpha} + (\lambda + 2\mu) \epsilon_{33} \end{aligned}$$

$$\sigma_{33} = E \epsilon_{33} \quad (5.4.10)$$

In addition, we will need expressions for $\sigma_{\alpha 3}$, which by definition are

$$\sigma_{\alpha 3} = 2\mu \epsilon_{\alpha 3} \quad (5.4.11)$$

5.4.3 Strain-displacement Equations

Employing the kinematic conditions stated in Assumption 3, we obtain the following results:

$$\epsilon_{\alpha\beta} = u_{(\alpha, \beta)} = 0 \quad (5.4.12)$$

$$\epsilon_{13} = \frac{w'_1 - x_2 \theta'_3 - \theta_2}{2} \quad (5.4.13)$$

$$\epsilon_{23} = \frac{w'_2 + x_1 \theta'_3 + \theta_1}{2} \quad (5.4.14)$$

$$\epsilon_{33} = w'_3 - x_1 \theta'_2 + x_2 \theta'_1 \quad (5.4.15)$$

where the primes denote differentiation with respect to x_3 .

Remark

As is often the case in beam and plate theories, the stress and kinematic assumptions lead to “microscopic” inconsistencies. For example, from Assumption 2 we have deduced $\epsilon_{\alpha\beta} = -\lambda_{33} \delta_{\alpha\beta}/[2(\lambda + \mu)]$, whereas from Assumption 3 we have $\epsilon_{\alpha\beta} = 0$. For purposes of calculating $\epsilon_{\alpha\beta}$, the former expression is preferred.

These inconsistencies ultimately cause no harm, and, as we have remarked previously, the ultimate justification of a “macroscopic” theory such as this one is its usefulness in practical structural engineering applications.

5.4.4 Definitions of Quantities Appearing in the Theory

Let \mathcal{F} be the set of node numbers at which forces and moments are applied, and let \mathcal{D} be the set of node numbers at which displacements and rotations are prescribed. (For simplicity, we shall assume $\mathcal{F} \cap \mathcal{D} = \emptyset$. In practice, as is always the case, we may specify forces and moments and displacements and rotations to mutually exclusive degrees of freedom at a node.) Let \mathbf{Q}_A and \mathbf{M}_A denote the 3×1 vectors of applied point loads and moments, respectively, at node $A \in \mathcal{F}$. (We assume the components of \mathbf{M}_A are defined in terms of the right-hand rule.) Let \mathbf{W}_A and $\boldsymbol{\Theta}_A$ denote the vectors of prescribed displacements and rotations, respectively, at node $A \in \mathcal{D}$.

| Quantity | Description |
|----------------------------------|--------------|
| w_i | Displacement |
| θ_i | Rotation |
| $\kappa_\alpha = \theta'_\alpha$ | Curvature |

| Quantity | Description |
|--|---|
| $\gamma_1 = w'_1 - \theta_2$ $\gamma_2 = w'_2 + \theta_1$ | Shear strains |
| $\epsilon = w'_3$ | Axial strain |
| $\psi = \theta'_3$ | Twist |
| $m_1 = \int_{A^e} \sigma_{33} x_2 dA$ $m_2 = \int_{A^e} \sigma_{33} x_1 dA$ | Bending moments |
| $m_3 = \int_{A^e} (\sigma_{23} x_1 - \sigma_{13} x_2) dA$ | Twisting moment |
| $q_a = \int_{A^e} \sigma_{a3} dA$ | Shear force |
| $q_3 = \int_{A^e} \sigma_{33} dA$ | Axial force |
| $W_i = \{W_{iA}\}, A \in \mathcal{D}$ | Prescribed nodal displacements |
| $\Theta_i = \{\Theta_{iA}\}, A \in \mathcal{D}$ | Prescribed nodal rotations |
| $Q_i = \{Q_{iA}\}, A \in \mathcal{F}$ | Prescribed nodal forces |
| $M_i = \{M_{iA}\}, A \in \mathcal{F}$ | Prescribed nodal moments |
| $F_i = \{F_f\}, 1 \leq f \leq n_{el}$ | Element applied forces per unit length |
| $F_f = \int_{A^e} f_f dA$ | |
| $C_i = \{C_{if}\}, 1 \leq f \leq n_{cl}$ | Element applied couples per unit length |
| $C_1 = \int_{A^e} f_3 x_2 dA$ $C_2 = \int_{A^e} f_3 x_1 dA$ | |
| $C_3 = \int_{A^e} (f_2 x_1 - f_1 x_2) dA$ | |
| $I_1' = \int_{A^e} x_2^2 dA$ $I_2' = \int_{A^e} x_1^2 dA$ | Element cross-section properties |
| $J' = I_1' + I_2'$ | |

Local-Global Transformations

Let $\hat{x}_1, \hat{x}_2, \hat{x}_3$ denote the coordinates of the global system. Similarly, a “hat” above a quantity will indicate components with respect to the global coordinate system. For example,

$$\hat{w} = \begin{Bmatrix} \hat{w}_1 \\ \hat{w}_2 \\ \hat{w}_3 \end{Bmatrix} \quad \text{and} \quad \hat{\theta} = \begin{Bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \\ \hat{\theta}_3 \end{Bmatrix} \quad (5.4.16)$$

are the global representations of w and θ , respectively. Let t^e denote the transformation matrix which rotates vector quantities in the e th local system into the global system. For example,

$$\hat{w}^e = t^e w^e \quad \hat{\theta}^e = t^e \theta^e \quad (5.4.17)$$

t^e is given by

$$t^e = [t_{ij}^e] \quad 1 \leq i, j \leq 3 \quad (5.4.18)$$

$$t_{ij}^e = e_i \cdot e_j^e \quad (5.4.19)$$

where e_i is a global unit basis vector and e_j^e is a local unit basis vector. Since t^e is orthogonal (i.e., $t^{e^{-1}} = t^{e^T}$), it follows that

$$w^e = t^{e^T} \hat{w}^e \quad \theta^e = t^{e^T} \hat{\theta}^e \quad (5.4.20)$$

and so on.

5.4.5 Variational Equation

We begin with the variational equation for the three-dimensional theory and incorporate the preceding relations:

$$0 = \int_{\Omega} \bar{u}_{(i,j)} \sigma_{ij} d\Omega - \int_{\Omega} \bar{u}_i f_i d\Omega - \int_{\Gamma_h} \bar{u}_i h_i d\Gamma \quad (5.4.21)$$

i. Iterate integrals and employ kinematic relations. Let $dA = dx_1 dx_2$; then

$$\int_{\Omega} \dots d\Omega = \sum_{e=1}^{n_{el}} \int_{\Omega^e} \dots d\Omega = \sum_{e=1}^{n_{el}} \int_0^{h^e} \int_{A^e} \dots dA dx_3 \quad (5.4.22)$$

Without loss of generality, we shall assume $\Gamma_h = \emptyset$.

$$\begin{aligned} 0 &= \sum_{e=1}^{n_{el}} \left\{ \int_0^{h^e} \int_{A^e} (2\bar{u}_{(\alpha, 3)} \sigma_{\alpha 3} + \bar{u}_{3, 3} \sigma_{33}) dA dx_3 \right. \\ &\quad \left. - \int_0^{h^e} \int_{A^e} \bar{u}_i f_i dA dx_3 \right\} \\ &= \sum_{e=1}^{n_{el}} \left\{ \int_0^{h^e} \int_{A^e} [(\bar{w}'_1 - x_2 \bar{\theta}'_3 - \bar{\theta}_2) \sigma_{13} + (\bar{w}'_2 + x_1 \bar{\theta}'_3 + \bar{\theta}_1) \sigma_{23} \right. \\ &\quad \left. + (\bar{w}'_3 - x_1 \bar{\theta}'_2 + x_2 \bar{\theta}'_1) \sigma_{33}] dA dx_3 - \int_0^{h^e} \int_{A^e} [(\bar{w}_1 - x_2 \bar{\theta}_3) f_1 \right. \\ &\quad \left. + (\bar{w}_2 + x_1 \bar{\theta}_3) f_2 + (\bar{w}_3 - x_1 \bar{\theta}_2 + x_2 \bar{\theta}_1) f_3] dA dx_3 \right\} \end{aligned} \quad (5.4.23)$$

ii. Use definitions of force resultants.

$$0 = \sum_{e=1}^{n_{el}} \left\{ \int_0^{h^e} [(\bar{w}'_1 - \bar{\theta}_2)q_1 + (\bar{w}'_2 + \bar{\theta}_1)q_2 + \bar{w}'_3q_3 + \bar{\theta}'_1m_1 - \bar{\theta}'_2m_2 + \bar{\theta}'_3m_3] dx_3 \right. \\ \left. - \int_0^{h^e} [\bar{w}_1F_1 + \bar{w}_2F_2 + \bar{w}_3F_3 + \bar{\theta}_1C_1 - \bar{\theta}_2C_2 + \bar{\theta}_3C_3] dx_3 \right\}$$

(5.4.24)

iii. Integration by parts indicates the differential equations which are satisfied, viz.,

$$0 = \sum_{e=1}^{n_{el}} \left\{ \int_0^{h^e} - [\underbrace{\bar{w}_1(q'_1 + F_1)}_{\text{transverse equilibrium}} \right. \\ + \underbrace{\bar{w}_2(q'_2 + F_2)}_{\text{transverse equilibrium}} \\ + \underbrace{\bar{w}_3(q'_3 + F_3)}_{\text{axial equilibrium}} \\ + \underbrace{\bar{\theta}_1(m'_1 - q_2 + C_1)}_{\text{moment equilibrium}} \\ - \underbrace{\bar{\theta}_2(m'_2 - q_1 + C_2)}_{\text{moment equilibrium}} \\ + \underbrace{\bar{\theta}_3(m'_3 + C_3)}_{\text{torsional equilibrium}}] dx_3 \Big\} \\ + \text{contributions to nodal force and moment equilibrium conditions}$$

(5.4.25)

Remark

Each element contributes end forces and moments to the nodes. Since the "barred" kinematic quantities are assumed continuous at the nodes, the sum of the element contributions must vanish, implying the six nodal equilibrium conditions. We shall add nodal applied forces and moments to the theory later on.

iv. Obtain explicit forms of the constitutive equations in terms of beam theory variables.

$$\begin{aligned}
 q_1 &= \int_A \sigma_{13} dA \\
 &= \mu \int_A (w'_1 - x_2 \theta'_3 - \theta_2) dA \\
 &= \mu A \gamma_1
 \end{aligned} \tag{5.4.26}$$

$$\begin{aligned}
 q_2 &= \int_A \sigma_{23} dA \\
 &= \mu \int_A (w'_2 + x_1 \theta'_3 + \theta_1) dA \\
 &= \mu A \gamma_2
 \end{aligned} \tag{5.4.27}$$

$$\begin{aligned}
 q_3 &= \int_A \sigma_{33} dA \\
 &= \int_A (w'_3 - x_1 \theta'_2 + x_2 \theta'_1) dA \\
 &= E A \epsilon
 \end{aligned} \tag{5.4.28}$$

$$\begin{aligned}
 m_1 &= \int_A \sigma_{33} x_2 dA \\
 &= E \int_A (w'_3 - x_1 \theta'_2 + x_2 \theta'_1) x_2 dA \\
 &= EI_1 \kappa_1
 \end{aligned} \tag{5.4.29}$$

$$\begin{aligned}
 m_2 &= \int_A \sigma_{33} x_1 dA \\
 &= E \int_A (w'_3 - x_1 \theta'_2 + x_2 \theta'_1) x_1 dA \\
 &= -EI_2 \kappa_2
 \end{aligned} \tag{5.4.30}$$

$$\begin{aligned}
 m_3 &= \int_A (\sigma_{23} x_1 - \sigma_{13} x_2) dA \\
 &= \mu \int_A [(w'_2 + x_1 \theta'_3 + \theta_1) x_1 - (w'_1 - x_2 \theta'_3 - \theta_2) x_2] dA \\
 &= \mu J \psi
 \end{aligned} \tag{5.4.31}$$

Remark

“Shear correction factors” may be introduced by suitably modifying the appro-

priate terms. That is, replace

$$q_\alpha = \mu A \gamma_\alpha \quad (5.4.32)$$

by

$$q_\alpha = \mu A_\alpha^s \gamma_\alpha \quad (\text{no sum}) \quad (5.4.33)$$

where A_α^s is the "effective shear area" for the α direction.

5.4.6 Strong Form

The domain of the beam theory boundary-value problem consists of n_{el} line segments in \mathbb{R}^3 , interconnected at nodes. Each line segment corresponds to the $[0, h^\epsilon]$ portion of the x_3^ϵ axis in the local, element coordinate system. By a slight abuse of notation, we shall write the element domain $[0, h^\epsilon]$. For each node A , let \mathcal{E}_A be the set of element numbers of elements attached to node A .

Let

$$\mathbf{q}_A^\epsilon = \begin{cases} \mathbf{q}^\epsilon(0) & (\text{node } A \text{ corresponds to } x_3^\epsilon = 0) \\ -\mathbf{q}^\epsilon(h^\epsilon) & (\text{node } A \text{ corresponds to } x_3^\epsilon = h^\epsilon) \end{cases} \quad (5.4.34)$$

$$\mathbf{q}^\epsilon = \begin{Bmatrix} q_1^\epsilon \\ q_2^\epsilon \\ q_3^\epsilon \end{Bmatrix} \quad (5.4.35)$$

$$\mathbf{m}_A^\epsilon = \begin{cases} \mathbf{m}^\epsilon(0) & (\text{node } A \text{ corresponds to } x_3^\epsilon = 0) \\ -\mathbf{m}^\epsilon(h^\epsilon) & (\text{node } A \text{ corresponds to } x_3^\epsilon = h^\epsilon) \end{cases} \quad (5.4.36)$$

$$\mathbf{m}^\epsilon = \begin{Bmatrix} m_1^\epsilon \\ -m_2^\epsilon \\ m_3^\epsilon \end{Bmatrix} \quad (5.4.37)$$

Given $F_i: \bigcup_{e=1}^{n_{el}} [0, h^\epsilon] \rightarrow \mathbb{R}$ and $C_i: \bigcup_{e=1}^{n_{el}} [0, h^\epsilon] \rightarrow \mathbb{R}$, Q_A and M_A , $A \in \mathcal{F}$, and W_A and Θ_A , $A \in \mathcal{D}$, find $w_i: \bigcup_{e=1}^{n_{el}} [0, h^\epsilon] \rightarrow \mathbb{R}$ and $\theta_i: \bigcup_{e=1}^{n_{el}} [0, h^\epsilon] \rightarrow \mathbb{R}$ such that

$$q'_1 + F_1 = 0 \quad (5.4.38)$$

$$q'_2 + F_2 = 0 \quad (5.4.39)$$

$$q'_3 + F_3 = 0 \quad (5.4.40)$$

$$m'_1 - q_2 + C_1 = 0 \quad (5.4.41)$$

$$m'_2 - q_1 + C_2 = 0 \quad (5.4.42)$$

$$m'_3 + C_3 = 0 \quad (5.4.43)$$

$$q_1 = \mu A_1^s \gamma_1 \quad (5.4.44)$$

$$q_2 = \mu A_2^s \gamma_2 \quad \left. \begin{array}{l} \text{in }]0, h^\epsilon[, 1 \leq e \leq n_{el} \end{array} \right. \quad (5.4.45)$$

$$q_3 = EA \epsilon \quad (5.4.46)$$

$$m_1 = EI_1 \kappa_1 \quad (5.4.47)$$

$$m_2 = -EI_2\kappa_2 \quad | \quad (5.4.48)$$

$$m_3 = \mu J\psi \quad | \quad (5.4.49)$$

$$\gamma_1 = w'_1 - \theta_2 \quad | \quad (5.4.50)$$

$$\gamma_2 = w'_2 + \theta_1 \quad | \quad (5.4.51)$$

$$\epsilon = w'_3 \quad | \quad (5.4.52)$$

$$\kappa_1 = \theta'_1 \quad | \quad (5.4.53)$$

$$\kappa_2 = \theta'_2 \quad | \quad (5.4.54)$$

$$\psi = \theta'_3 \quad | \quad (5.4.55)$$

$$t^\epsilon w_A^\epsilon = W_A \quad | \quad A \in \mathcal{D}, e \in \mathcal{E}_A \quad (5.4.56)$$

$$t^\epsilon \theta_A^\epsilon = \Theta_A \quad | \quad A \in \mathcal{D}, e \in \mathcal{E}_A \quad (5.4.57)$$

$$\sum_{e \in \mathcal{E}_A} t^\epsilon q_A^\epsilon + Q_A = 0 \quad | \quad A \in \mathcal{F} \quad (5.4.58)$$

$$\sum_{e \in \mathcal{E}_A} t^\epsilon m^\epsilon + M_A = 0 \quad | \quad A \in \mathcal{F} \quad (5.4.59)$$

5.4.7 Weak Form

The spaces we need are as follows:

$$\begin{aligned} \mathcal{S} &= \mathcal{S}\left(\bigcup_{e=1}^{n_{el}} [0, h^\epsilon]\right) \\ &= \{(w, \theta) \mid w_A = W_A, \theta_A = \Theta_A, A \in \mathcal{D}\} \end{aligned} \quad (5.4.60)$$

$$\begin{aligned} \mathcal{O} &= \mathcal{O}\left(\bigcup_{e=1}^{n_{el}} [0, h^\epsilon]\right) \\ &= \{(\bar{w}, \bar{\theta}) \mid w_A = 0, \theta_A = 0, A \in \mathcal{D}\} \end{aligned} \quad (5.4.61)$$

Given F , C , Q_A , M_A , W_A , and Θ_A , as above, find $(w, \theta) \in \mathcal{S}$ such that for all $(\bar{w}, \bar{\theta}) \in \mathcal{O}$

$$\begin{aligned} 0 &= \sum_{e=1}^{n_{el}} \left\{ \int_0^{h^\epsilon} (\bar{\gamma}_1 \mu A_1^\epsilon \gamma_1 + \bar{\gamma}_2 \mu A_2^\epsilon \gamma_2 + \bar{\kappa}_1 EI_1 \kappa_1 + \bar{\kappa}_2 EI_2 \kappa_2 + \bar{\epsilon} EA \epsilon \right. \\ &\quad \left. + \bar{\psi} \mu J \psi \right) dx_3^\epsilon \\ &\quad - \int_0^{h^\epsilon} (\bar{w}_1 F_1 + \bar{w}_2 F_2 + \bar{w}_3 F_3 + \bar{\theta}_1 C_1 - \bar{\theta}_2 C_2 + \bar{\theta}_3 C_3) dx_3^\epsilon \Big\} \\ &\quad - \sum_{A \in \mathcal{F}} (\bar{w}_A^T Q_A + \bar{\theta}_A^T M_A) \end{aligned} \quad (5.4.62)$$

This can be written in the usual way:

$$a(\bar{u}, u) = (\bar{u}, f) + (\bar{u}, h)_\Gamma \quad (5.4.6)$$

where

$$u = \begin{Bmatrix} w_1 \\ w_2 \\ w_3 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{Bmatrix} \quad \bar{u} = \begin{Bmatrix} \bar{w}_1 \\ \bar{w}_2 \\ \bar{w}_3 \\ \bar{\theta}_1 \\ \bar{\theta}_2 \\ \bar{\theta}_3 \end{Bmatrix} \quad f = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ C_1 \\ -C_2 \\ C_3 \end{Bmatrix} \quad h = \begin{Bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ M_1 \\ M_2 \\ M_3 \end{Bmatrix} \quad (5.4.6)$$

$$a(\bar{u}, u) = \sum_{e=1}^{n_{el}} \left\{ \int_0^{h^e} (\bar{\gamma}_1 \mu A_1^s \gamma_1 + \bar{\gamma}_2 \mu A_2^s \gamma_2 + \bar{\kappa}_1 EI_1 \kappa_1 + \bar{\kappa}_2 EI_2 \kappa_2 + \bar{\epsilon} EA \epsilon + \bar{\psi} EJ \psi) dx_3^e \right\} \quad (5.4.6e)$$

$$(\bar{u}, f) = \sum_{e=1}^{n_{el}} \int_0^{h^e} (\bar{w}_1 F_1 + \bar{w}_2 F_2 + \bar{w}_3 F_3 + \bar{\theta}_1 C_1 - \bar{\theta}_2 C_2 + \bar{\theta}_3 C_3) dx_3^e \quad (5.4.6f)$$

$$(\bar{u}, h)_\Gamma = \sum_{A \in \mathcal{F}} (\bar{w}_A^T Q_A + \bar{\theta}_A^T M_A) \quad (5.4.6g)$$

As usual, $a(\cdot, \cdot)$, (\cdot, \cdot) , and $(\cdot, \cdot)_\Gamma$ are symmetric bilinear forms.

5.4.8 Matrix Formulation of the Variational Equation

$$0 = \sum_{e=1}^{n_{el}} \left\{ \int_0^{h^e} [\underbrace{\bar{\gamma}^T D^s \gamma}_{\text{shear}} + \underbrace{\bar{\kappa}^T D^b \kappa}_{\text{bending}} + \underbrace{\bar{\epsilon} (EA) \epsilon}_{\text{axial}} + \underbrace{\bar{\psi} (\mu J) \psi}_{\text{torsional}}] dx_3^e \right\} - \int_0^{h^e} [\bar{w}^T F + \bar{\theta}^T C] dx_3^e - \sum_{A \in \mathcal{F}} (\bar{w}_A^T Q_A + \bar{\theta}_A^T M_A) \quad (5.4.68)$$

where

$$\Theta = \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{Bmatrix} \quad \bar{\Theta} = \begin{Bmatrix} \bar{\theta}_1 \\ \bar{\theta}_2 \\ \bar{\theta}_3 \end{Bmatrix} \quad (5.4.69)$$

$$\boldsymbol{w} = \begin{Bmatrix} w_1 \\ w_2 \\ w_3 \end{Bmatrix} \quad \bar{\boldsymbol{w}} = \begin{Bmatrix} \bar{w}_1 \\ \bar{w}_2 \\ \bar{w}_3 \end{Bmatrix} \quad (5.4.70)$$

$$\boldsymbol{\gamma} = \begin{Bmatrix} \gamma_1 \\ \gamma_2 \end{Bmatrix} \quad \bar{\boldsymbol{\gamma}} = \begin{Bmatrix} \bar{\gamma}_1 \\ \bar{\gamma}_2 \end{Bmatrix} \quad (5.4.71)$$

$$\boldsymbol{\kappa} = \begin{Bmatrix} \kappa_1 \\ \kappa_2 \end{Bmatrix} \quad \bar{\boldsymbol{\kappa}} = \begin{Bmatrix} \bar{\kappa}_1 \\ \bar{\kappa}_2 \end{Bmatrix} \quad (5.4.72)$$

$$\boldsymbol{D}^s = \begin{bmatrix} \mu A_1^s & 0 \\ 0 & \mu A_2^s \end{bmatrix} \quad (5.4.73)$$

$$\boldsymbol{D}^b = \begin{bmatrix} EI_1 & 0 \\ 0 & EI_2 \end{bmatrix} \quad (5.4.74)$$

5.4.9 Finite Element Stiffness Matrix and Load Vector

We shall assume that

$$w_i^h = \sum_{a=1}^{n_{en}} N_a w_{ia}^h \quad (5.4.75)$$

$$\theta_i^h = \sum_{a=1}^{n_{en}} N_a \theta_{ia}^h \quad (5.4.76)$$

Remark

It is not necessary to assume the same shape functions for transverse and extensional displacements or for bending and torsional rotations. Arguments can be made, in fact, that there are some conceptual and practical advantages to employing different interpolations in the present context (see Tessler [45] and Hughes and Tezduyar [46]). However, for the elements we wish to emphasize in Sec. 5.5, this generality is unnecessary.

Define

$$\boldsymbol{d}^\epsilon = \{d_p^\epsilon\} \quad (5.4.77)$$

$$\bar{\boldsymbol{d}}^\epsilon = \{\bar{d}_p^\epsilon\} \quad (5.4.78)$$

$$d_p^\epsilon = \begin{cases} w_{ia}^h & p = 6a - 6 + i \\ \theta_{ia}^h & p = 6a - 3 + i \end{cases} \quad (5.4.79)$$

$$\bar{d}_p^\epsilon = \begin{cases} \bar{w}_{ia}^h & p = 6a - 6 + i \\ \bar{\theta}_{ia}^h & p = 6a - 3 + i \end{cases} \quad (5.4.80)$$

Thus there are six degrees of freedom per node and we can write

$$\boldsymbol{\kappa} = \mathbf{B}^b \boldsymbol{d}^e \quad \bar{\boldsymbol{\kappa}} = \mathbf{B}^b \bar{\boldsymbol{d}}^e \quad (5.4.81)$$

$$\boldsymbol{\gamma} = \mathbf{B}^s \boldsymbol{d}^e \quad \bar{\boldsymbol{\gamma}} = \mathbf{B}^s \bar{\boldsymbol{d}}^e \quad (5.4.82)$$

$$\boldsymbol{\epsilon} = \mathbf{B}^a \boldsymbol{d}^e \quad \bar{\boldsymbol{\epsilon}} = \mathbf{B}^a \bar{\boldsymbol{d}}^e \quad (5.4.83)$$

$$\boldsymbol{\psi} = \mathbf{B}' \boldsymbol{d}^e \quad \bar{\boldsymbol{\psi}} = \mathbf{B}' \bar{\boldsymbol{d}}^e \quad (5.4.84)$$

$$\mathbf{B}^b = [B_1^b, \dots, B_{n_{en}}^b] \quad (5.4.85)$$

$$\mathbf{B}^s = [B_1^s, \dots, B_{n_{en}}^s] \quad (5.4.86)$$

$$\mathbf{B}^a = [B_1^a, \dots, B_{n_{en}}^a] \quad (5.4.87)$$

$$\mathbf{B}' = [B_1', \dots, B_{n_{en}}'] \quad (5.4.88)$$

$$\mathbf{B}_c^b = \begin{bmatrix} 0 & 0 & 0 & N_c' & 0 & 0 \\ 0 & 0 & 0 & 0 & N_c' & 0 \end{bmatrix} \quad (5.4.89)$$

$$\mathbf{B}_c^s = \begin{bmatrix} N_c' & 0 & 0 & 0 & -N_c & 0 \\ 0 & N_c' & 0 & N_c & 0 & 0 \end{bmatrix} \quad (5.4.90)$$

$$\mathbf{B}_c^a = [0 \ 0 \ N_c' \ 0 \ 0 \ 0] \quad (5.4.91)$$

$$\mathbf{B}_c' = [0 \ 0 \ 0 \ 0 \ 0 \ N_c'] \quad (5.4.92)$$

where $1 \leq c \leq n_{en}$.

With these definitions, we obtain the following expressions for the stiffness and load (with respect to the local coordinate system):

$$\boldsymbol{k}^e = \boldsymbol{k}_b^e + \boldsymbol{k}_s^e + \boldsymbol{k}_a^e + \boldsymbol{k}_t^e \quad (5.4.93)$$

$$\boldsymbol{k}_b^e = \int_0^{h^e} \mathbf{B}^{b^T} \mathbf{D}^b \mathbf{B}^b dx_3^e \quad (\text{bending stiffness}) \quad (5.4.94)$$

$$\boldsymbol{k}_s^e = \int_0^{h^e} \mathbf{B}^{s^T} \mathbf{D}^s \mathbf{B}^s dx_3^e \quad (\text{shear stiffness}) \quad (5.4.95)$$

$$\boldsymbol{k}_a^e = \int_0^{h^e} \mathbf{B}^{a^T} (EA) \mathbf{B}^a dx_3^e \quad (\text{axial stiffness}) \quad (5.4.96)$$

$$\boldsymbol{k}_t^e = \int_0^{h^e} \mathbf{B}^{t^T} (\mu J) \mathbf{B}' dx_3^e \quad (\text{torsional stiffness}) \quad (5.4.97)$$

$$\boldsymbol{f}^e = \{f_p^e\} \quad (5.4.98)$$

$$f_p^e = \begin{cases} \int_0^{h^e} N_a F_i dx_3^e & p = 6a - 6 + i \\ (-1)^{i+1} \int_0^{h^e} N_a C_i dx_3^e & p = 6a - 3 + i \end{cases} \quad (5.4.99)$$

5.4.10 Representation of Stiffness and Load in Global Coordinates

Before assembly it is necessary to transform k^e and f^e into global coordinates. Assume all “internal” degrees of freedom (if any) have been statically eliminated and only the 12 degrees of freedom corresponding to the two end nodes remain (i.e., k^e and f^e have dimension 12×12 and 12×1 , respectively, with the usual ordering). Let

$$\underset{12 \times 12}{T^e} = \begin{bmatrix} t^e & 0 & 0 & 0 \\ 0 & t^e & 0 & 0 \\ 0 & 0 & t^e & 0 \\ 0 & 0 & 0 & t^e \end{bmatrix} \quad (5.4.100)$$

Then

$$\hat{k}^e = T^e k^e T^{eT} \quad (5.4.101)$$

and

$$\hat{f}^e = T^e f^e \quad (5.4.102)$$

are the globally oriented counterparts of k^e and f^e , respectively. These formulas can be derived as follows. First, note that $\hat{d}^e = T^e d^e$, $\hat{c}^e = T^e c^e$ and T^e is orthogonal. Compute:

$$\begin{aligned} \hat{c}^{eT} \hat{k}^e \hat{d}^e &= c^{eT} k^e d^e \\ &= \hat{c}^{eT} (T^e k^e T^{eT}) \hat{d}^e \end{aligned} \quad (5.4.103)$$

$$\begin{aligned} \hat{c}^{eT} \hat{f}^e &= c^{eT} f^e \\ &= \hat{c}^{eT} (T^e f^e) \end{aligned} \quad (5.4.104)$$

The above are to hold for all \hat{c}^e and \hat{d}^e . Hence, the results follow.)

Remark

It may be noted that “trusses” are special cases of the preceding theory. For a truss, only axial extension effects are accounted for (i.e., bending, transverse shear, and torsion are neglected). Thus the rotational degrees of freedom may be ignored.

I.5 REDUCED INTEGRATION BEAM ELEMENTS

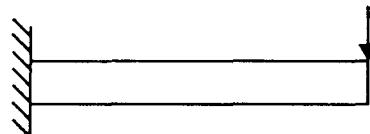
Given the assumption made in Sec. 5.4 that the transverse displacements and transverse rotations are interpolated with the *same* shape functions, there is a tendency to “lock” if the shear term is not handled appropriately. An effective strategy is to employ uniform reduced integration in which the quadrature rule is taken to be one order lower than the “normal” one. The situation is summarized Fig. 5.5.1. Correct rank is attained in all cases. Note that the reduced rule exactly integrates all terms of the stiffness except the transverse shear contributions.

The basic one-point quadrature, two-noded element, in which linear inter-

| | | | |
|-----------------|-----------|-----------|-------------|
| | | | |
| Shape functions | Linear | Quadratic | Cubic |
| Quadrature rule | One-point | Two-point | Three-point |

Figure 5.5.1 Uniform reduced integration beam elements.

pulation functions are used for displacements and rotations, was first proposed in [47]. Although it is apparent that such low-order interpolations are incapable of exactly representing typical bending behavior, the improvement engendered by reduced integration is dramatic, as may be seen from the example presented in Fig. 5.5.2. The



Normalized Tip Displacement

| Number of elements | One-point | Two-point |
|--------------------|-----------|------------------------|
| Thick beam | | |
| 1 | 0.762 | 0.416×10^{-1} |
| 2 | 0.940 | 0.445 |
| 4 | 0.985 | 0.762 |
| 8 | 0.996 | 0.927 |
| 16 | 0.999 | 0.981 |
| Thin beam | | |
| 1 | 0.750 | 0.200×10^{-4} |
| 2 | 0.938 | 0.800×10^{-4} |
| 4 | 0.984 | 0.320×10^{-4} |
| 8 | 0.996 | 0.128×10^{-3} |
| 16 | 0.999 | 0.512×10^{-3} |

Figure 5.5.2 Comparison of reduced integration (one-point) and full integration (two-point) for linear beam element.

reduced-integration element may be shown to be equivalent to an exactly integrated one in which quadratic shape functions are used for transverse displacement and linear shape functions are used for axial displacement and rotations. The middle-node displacement needs to be statically condensed to achieve identical stiffnesses. This observation was apparently first made by H. Allik [48].

The three-node, quadratic reduced-integration element has been in use for some time. It is interesting to note that, upon static condensation of the internal node, the corresponding beam element yields the "exact" stiffness matrix of structural theory, which is based upon cubic transverse displacement interpolation (e.g., see [49]). (This observation is also due to H. Allik [48].) The salubrious effect of reduced quadrature is again apparent.

Residual Bending Flexibility

MacNeal [50] has used the concept of "residual bending flexibility" to further improve the two-node, one-point quadrature beam element. He has shown that if μA_α is replaced by μA_α^s , where

$$\overline{\mu A_\alpha^s} \stackrel{\text{def}}{=} \left(\frac{1}{\mu A_\alpha} + \frac{h^2}{12EI_\alpha} \right)^{-1}$$

then the exact stiffness of structural analysis theory is also obtained. The term $h^2/(12EI_\alpha)$ is called the **residual bending flexibility**. Thus by combining the reduced integration and residual bending flexibility concepts, linear functions can be made to achieve the accuracy of cubics in bending. This simple and efficient element seems ideally suited for most practical applications.

Exercise 1. Consider the case of two-dimensional beam bending. Neglect axial and torsional effects as well as out-of-plane bending and shear. Assume further that linear shape functions are employed; and the nodal degrees of freedom are ordered as follows: w_1 , θ_1 , w_2 , θ_2 (see Fig. 5.5.3).

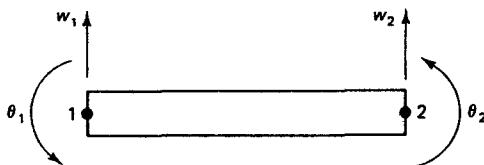


Figure 5.5.3

- i. Derive the following expression for bending stiffness:

$$k_b = \frac{EI}{h} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

ii. Use one-point Gaussian quadrature to derive the following shear stiffness:

$$k_s^{(1)} = \frac{\mu A^s}{h} \begin{bmatrix} 1 & \frac{h}{2} & -1 & \frac{h}{2} \\ \frac{h}{2} & \frac{h^2}{4} & -\frac{h}{2} & \frac{h^2}{4} \\ -1 & -\frac{h}{2} & 1 & -\frac{h}{2} \\ \frac{h}{2} & \frac{h^2}{4} & -\frac{h}{2} & \frac{h^2}{4} \end{bmatrix}$$

iii. Use two-point Gaussian quadrature to obtain the following shear stiffness:

$$k_s^{(2)} = \frac{\mu A^s}{h} \begin{bmatrix} 1 & \frac{h}{2} & -1 & \frac{h}{2} \\ \frac{h}{2} & \frac{h^2}{3} & -\frac{h}{2} & \frac{h^2}{6} \\ -1 & -\frac{h}{2} & 1 & -\frac{h}{2} \\ \frac{h}{2} & \frac{h^2}{6} & -\frac{h}{2} & \frac{h^2}{3} \end{bmatrix}$$

iv. Show that the rank of $k_s^{(1)}$ is one and the rank of $k_s^{(2)}$ is two.

v. Use the residual bending flexibility technique to modify the one-point shear stiffness of part (ii). Combine this result with the bending stiffness derived in part (i) to form the total stiffness. (This is the "exact" stiffness to which we alluded previously.) Take the limit $\mu \rightarrow \infty$ ("infinite shear stiffness") and show that the resulting stiffness is identical to the one obtained from Bernoulli-Euler theory with Hermite cubics (see Sec. 1.16).

Exercise 2. Use the results of Problem 1 to calculate the three-dimensional 12×12 beam stiffness in the local coordinate system accounting for axial and torsional effects as well as bending and transverse shear about two planes.

REFERENCES

Section 5.3

1. E. D. L. Pugh, "The Static and Dynamic Analysis of Mindlin Plates by Isoparametric Finite Elements," M.Sc. Thesis, C/M/125/76, Department of Civil Engineering, University College of Swansea, U.K., 1976.
2. T. J. R. Hughes, R. L. Taylor, and W. Kanoknukulchai, "A Simple and Efficient Element for Plate Bending," *International Journal for Numerical Methods in Engineering*, 11, no. 10 (1977), 1529-1543.
3. E. D. L. Pugh, E. Hinton, and O. C. Zienkiewicz, "A Study of Quadrilateral Plate Bending Elements with 'Reduced' Integration," *International Journal for Numerical Methods in Engineering*, 12, no. 7 (1978), 1059-1079.

4. D. S. Malkus and T. J. R. Hughes, "Mixed Finite Element Methods—Reduced and Selective Integration Techniques: A Unification of Concepts," *Computer Methods in Applied Mechanics and Engineering*, 15, no. 1 (1978), 63–81.
5. M. Rossow, "Efficient C⁰ Finite Element Solution of Simply Supported Plates of Polygonal Shape," *Journal of Applied Mechanics*, 44 (1977), 347–349.
6. L. R. Scott, "A Survey of Displacement Methods for the Plate Bending Problem," U.S.–Germany Symposium on Formulations and Computational Algorithms in Finite Element Analysis, Massachusetts Institute of Technology, Cambridge, Massachusetts, August 9–13, 1976.
7. G. Sander, "Application of the Dual Analysis Principle," in *High Speed Computing of Elastic Structures, Proceedings of IUTAM Symposium*, Liege, Belgium (1971), 167–207.
8. H. C. Rhee, Ph.D. Thesis, Georgia Institute of Technology, Atlanta (1976).
9. O. C. Zienkiewicz, R. L. Taylor, and J. M. Too, "Reduced Integration Technique in General Analysis of Plates and Shells," *International Journal for Numerical Methods in Engineering*, 3 (1971), 275–290.
10. T. J. R. Hughes, M. Cohen, and M. Haroun, "Reduced and Selective Integration Techniques in the Finite Element Analysis of Plates," *Nuclear Engineering and Design*, 46 (1978), 203–222.
11. J. Barlow, "Optimal Stress Locations in Finite Element Models," *International Journal for Numerical Methods in Engineering*, 10 (1976), 243–251.
12. T. Belytschko, C. S. Tsay, and W. K. Liu, "A Stabilization Matrix for the Bilinear Mindlin Plate Element," *Computer Methods in Applied Mechanics and Engineering*, 29 (1981), 313–327.
13. R. L. Taylor, "Finite Elements for General Shell Analysis," *Preprints of the 5th International Seminar on Computational Aspects of the Finite Element Method*, West Berlin, Germany, August 20–21, 1979.
14. T. J. R. Hughes and M. Cohen, "The 'Heterosis' Family of Plate Finite Elements," *Proceedings of the ASCE Electronic Computations Conference*, St. Louis, Missouri, August 6–8, 1979.
15. T. J. R. Hughes and M. Cohen, "The 'Heterosis' Finite Element for Plate Bending," *Computers and Structures*, 9 (1978), 445–450.
16. D. R. J. Owen and E. Hinton, *Finite Elements in Plasticity: Theory and Practice*, Swansea, U.K.: Pineridge Press, 1980.
17. T. J. R. Hughes and T. E. Tezduyar, "Finite Elements Based Upon Mindlin Plate Theory With Particular Reference to the Four-node Bilinear Isoparametric Element," *Journal of Applied Mechanics*, (September 1981), 587–596.
18. R. H. MacNeal, "A Simple Quadrilateral Shell Element," *Computers and Structures*, 8 (1978), 175–183.
19. R. L. Spilker and N. I. Munir, "The Hybrid-Stress Model for Thin Plates," *International Journal for Numerical Methods in Engineering*, 15, no. 8 (1980), 1239–1260.
20. R. L. Spilker and N. I. Munir, "A Serendipity Cubic-Displacement Hybrid-Stress Element for Thin and Moderately Thick Plates," *International Journal for Numerical Methods in Engineering*, 15, no. 8 (1980), 1261–1278.
21. R. L. Spilker and N. I. Munir, "A Hybrid-Stress Quadratic Serendipity Displacement Mindlin Plate Bending Element," *Computers and Structures*, 12 (1980), 11–21.

22. J. Robinson, "LORA—An Accurate Four Node Stress Plate Bending Element," *International Journal for Numerical Methods in Engineering*, 14, no. 2 (1979), 296–306.
23. R. H. MacNeal, "Derivation of Element Stiffness Matrices by Assumed Strain Distributions," *Nuclear Engineering and Design*, 70 (1982), 3–12.
24. T. J. R. Hughes and R. L. Taylor, "The Linear Triangular Bending Element," in *The Mathematics of Finite Elements and Applications IV, MAFELAP 1981* London: Academic Press, 1982, pp. 127–142.
25. R. H. MacNeal, "The TRIA3 Plate Element," Memo RHM-37, MacNeal Schwendler Corporation, October 1976.
26. I. Fried and S. K. Yang, "Triangular, Nine-degree-of-freedom, C⁰ Plate Bending Element of Quadratic Accuracy," *Quarterly Journal of Applied Mathematics*, 31 (1973), 303–312.
27. H. Garnet, J. Crouzet-Pascal, and A. B. Pifko, "Aspects of a Simple Triangular Plate Bending Finite Element," *Computers and Structures*, 12 (1980), 783–789.
28. T. K. Wong, "Nonlinear Analysis of Reinforced Concrete Slab Systems Using Cubic Mindlin Plate Elements," M.Sc. Thesis, Department of Civil Engineering, University College of Swansea, U.K., January 1981.
29. J. L. Batoz and M. Ben Tahar, "Formulation et Evaluation d'un Nouvel Élément Quadrilatéral à 12 D.L. pour la Flexion des Plaques Mincees," Département de Génie Mécanique, Université de Technologie, Compiègne, France.
30. J. A. Stricklin, W. Haisler, P. Tisdale, and R. Gunderson, "A Rapidly Converging Triangular Plate Element," *AIAA J.*, 7, no. 1 (1969), 180–181.
31. J. L. Batoz, K. J. Bathe, and L. W. Ho, "A Study of Three-node Triangular Plate Bending Elements," *International Journal for Numerical Methods in Engineering*, 15 (1980), 1771–1812.
32. J. L. Batoz, "An Explicit Formulation for an Efficient Triangular Plate-Bending Element," *International Journal for Numerical Methods in Engineering*, 18 (1982), 1077–1089.
33. B. M. Irons, "The Semiloof Shell Element," *Finite Elements for Thin Shells and Curved Members*, (eds. D. G. Ashwell and R. H. Gallagher) London: John Wiley, 1976, Chapter 11, pp. 197–222.
34. L. P. R. Lyons, "A General Finite Element System with Special Reference to the Analysis of Cellular Structures," Ph.D. Thesis, Imperial College, London, 1977.
35. M. A. Crisfield, "A Four-noded Plate Bending Element Using Shear Constraints; A Modified Version of Lyons' Element," *Computer Methods in Applied Mechanics and Engineering*, 38 (1983), 93–120.
36. A. Tessler and T. J. R. Hughes, "An Improved Treatment of Transverse Shear in the Mindlin-type Four-node Quadrilateral Element," *Computer Methods in Applied Mechanics and Engineering*, 39 (1983), 311–335.
37. A. Tessler and T. J. R. Hughes, "Three-node Mindlin Plate Element with Improved Transverse Shear," *Computer Methods in Applied Mechanics and Engineering*, 50 (1985), 71–101.
38. K. J. Bathe and E. N. Dvorkin, "A Four-node Plate Bending Element Based on Mindlin/Reissner Plate Theory and a Mixed Interpolation," *International Journal for Numerical Methods in Engineering*, 21 (1985), 367–383.

39. K. J. Bathe and E. N. Dvorkin, "A Formulation of General Shell Elements —The Use of Mixed Interpolation of Tensorial Components," *Proceedings of the Conference on Numerical Methods in Engineering: Theory and Applications*, University College of Swansea, U.K., January 1985.
40. T. Belytschko and C. S. Tsay, "A Stabilization Procedure for the Quadrilateral Plate Bending Element with One-point Quadrature," *International Journal for Numerical Methods in Engineering*, 19 (1983), 405–420.
41. K. C. Park and D. L. Flagg, "A Symbolic Fourier Synthesis of a One-point Integrated Quadrilateral Plate Element," *Computer Methods in Applied Mechanics and Engineering*, 48, no. 2 (1985), 203–236.
42. T. Belytschko, J. S.-J. Ong, and W. K. Liu, "A Consistent Control of Spurious Singular Modes in the 9-node Lagrange Element for the Laplace and Mindlin Plate Equations," *Computer Methods in Applied Mechanics and Engineering*, 44 (1984), 269–295.
43. M. A. Crisfield, "A Quadratic Mindlin Element Using Shear Constraints," *Computers and Structures*, 18 (1984), 833–852.
44. H. C. Huang and E. Hinton, "A Nine-node Lagrangian Plate Element with Enhanced Shear Interpolation," *Engineering Computations*, 1 (1984), 369–379.

Section 5.4

45. A. Tessler and S. B. Dong, "On a Hierarchy of Conforming Timoshenko Beam Elements," *Computers and Structures*, 14 (1981), 335–344.
46. T. J. R. Hughes and T. E. Tezduyar, "Finite Elements Based Upon Mindlin Plate Theory With Particular Reference to the Four-node Bilinear Isoparametric Element," *Journal of Applied Mechanics*, (September 1981), 587–596.

Section 5.5

47. T. J. R. Hughes, R. L. Taylor, and W. Kanoknukulchai, "A Simple and Efficient Element for Plate Bending," *International Journal for Numerical Methods in Engineering*, 11, no. 10 (1977), 1529–1543.
48. H. Allik, Private communications, 1976.
49. R. D. Cook, *Concepts and Applications of Finite Element Analysis*, New York: John Wiley, 1974.
50. R. H. MacNeal, "A Simple Quadrilateral Shell Element," *Computers and Structures*, 8, (1978), 175–183.

6

The C^0 -Approach to Curved Structural Elements

6.1 INTRODUCTION

In this chapter we present a general formulation for curved structural elements. Throughout, transverse shear deformations are accounted for. This enables the use of C^0 interpolations as in the plate and beam theories of Chapter 5. Despite the fact that for over 20 years, intense interest has been focused on the development of shell finite elements, there is still some dissatisfaction with available methodology. Currently, a number of research efforts are directed at deepening the understanding of and improving shell finite element capabilities. Many different approaches have been developed in finite element shell analysis, and an enormous literature now exists. No attempt will be made to review the literature in this brief chapter. (A literature review in finite element shell analysis would entail in itself a major work!) The approach presented herein is felt to be quite general and the one currently gaining favor. The extension of the present ideas to nonlinear analysis is relatively straightforward (see, e.g., [1–3] and references therein). This is viewed as an important attribute of this type of approach because practical shell analysis often involves consideration of nonlinear effects such as buckling.

By now the reader should be familiar with the scheme of developing finite element formulations advocated herein—namely, a classical statement of the boundary-value problem is posed first. Then a corresponding weak formulation is developed and subsequently discretized by introducing finite element interpolation functions. The “shell theory” presented in this chapter is simply three-dimensional elasticity with certain kinematic and mechanical assumptions built in. The situation is essentially the same as in Chapter 5, where Reissner-Mindlin plate theory was devel-

oped from three-dimensional theory.¹ To avoid a cumbersome presentation of the basic theory, we will take advantage of the reader's familiarity with the material in Chapters 1–5 and move ahead to the development of the finite element formulation.

In Sec. 6.2 we present the three-dimensional theory. The reduction to practically important two-dimensional cases is treated in Sec. 6.3. The two-dimensional formulation applies to shells of revolution, tubes, rings, and curved beams.

DOUBLY CURVED SHELLS IN THREE DIMENSIONS

6.2.1 Geometry

The geometry of a typical quadrilateral shell element is defined by the following relations:

$$\mathbf{x}(\xi, \eta, \zeta) = \bar{\mathbf{x}}(\xi, \eta) + \mathbf{X}(\xi, \eta, \zeta) \quad (6.2.1)$$

$$\bar{\mathbf{x}}(\xi, \eta) = \sum_{a=1}^{n_{en}} N_a(\xi, \eta) \bar{\mathbf{x}}_a \quad (6.2.2)$$

$$\mathbf{X}(\xi, \eta, \zeta) = \sum_{a=1}^{n_{en}} N_a(\xi, \eta) \mathbf{X}_a(\zeta) \quad (6.2.3)$$

$$\mathbf{X}_a(\zeta) = z_a(\zeta) \hat{\mathbf{X}}_a \quad (\text{no sum}) \quad (6.2.4)$$

$$z_a(\zeta) = N_+(\zeta) z_a^+ + N_-(\zeta) z_a^- \quad (6.2.5)$$

$$N_+(\zeta) = \frac{1}{2}(1 + \zeta), \quad N_-(\zeta) = \frac{1}{2}(1 - \zeta) \quad (6.2.6)$$

In (6.2.1) through (6.2.6), \mathbf{x} denotes the position vector of a generic point of the shell; $\bar{\mathbf{x}}$ is the position vector of a point in the reference surface; \mathbf{X} is a position vector based at a point in the reference surface which defines the "fiber direction" through the point; $\bar{\mathbf{x}}_a$ is the position vector of nodal point a ; N_a denotes a two-dimensional shape function associated with node a ; n_{en} is the number of element nodes; $\hat{\mathbf{X}}_a$ is a unit vector emanating from node a in the fiber direction; and z_a is a "thickness function," associated with node a , which is defined by the location of the reference surface.

Equations (6.2.1) through (6.2.6) represent a smooth mapping of the biunit cube into the physical shell domain; see Fig. 6.2.1. For ζ fixed, the surface defined by (6.2.1) is called a *lamina*; for ξ, η fixed, the line defined by (6.2.1) is called the *fiber*. The fibers are generally *not* perpendicular to the laminae. Sometimes the fiber is referred to as the "pseudonormal".

For a particular choice of two-dimensional shape functions, (6.2.1) through (6.2.6) are precisely defined upon specification of $\bar{\mathbf{x}}_a$, $\hat{\mathbf{X}}_a$, z_a^+ , and z_a^- ($a = 1, 2, \dots, n_{en}$). It is convenient in practice to take as input the coordinates of the top and bottom surfaces of the shell along each nodal fiber (\mathbf{x}_a^+ and \mathbf{x}_a^- , respectively) and a parameter $\zeta \in [-1, +1]$, which defines the location of the reference surface. For example, if $\zeta = -1, 0, +1$ (respectively), then the reference surface is taken to be the bottom,

¹ In the present context the approach is sometimes described as the "degenerated shell element" procedure [4].

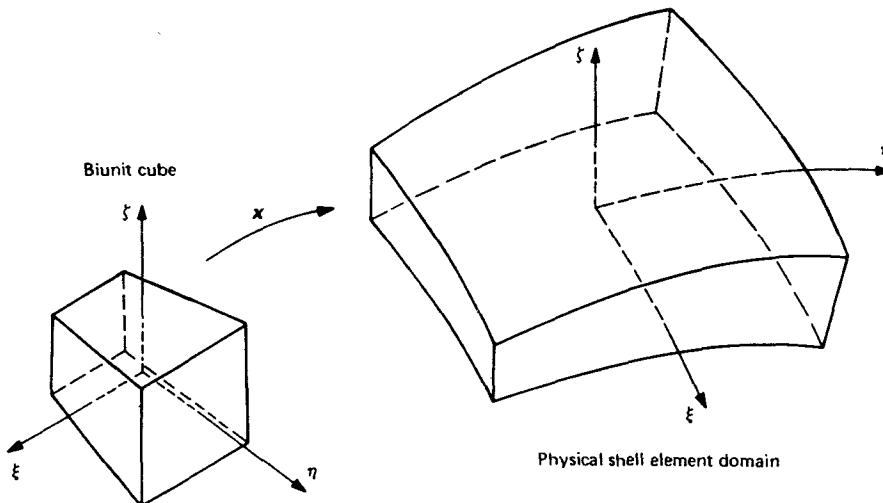


Figure 6.2.1

middle, top (respectively) of the shell. This option enables compatible modeling of shell-continuum interfaces. From these data we may calculate

$$\bar{x}_a = \frac{1}{2}(1 - \bar{\zeta})x_a^- + \frac{1}{2}(1 + \bar{\zeta})x_a^+ \quad (6.2.7)$$

$$\hat{X}_a = \frac{x_a^+ - x_a^-}{\|x_a^+ - x_a^-\|} \quad (6.2.8)$$

$$z_a^+ = \frac{1}{2}(1 - \bar{\zeta})\|x_a^+ - x_a^-\| \quad (6.2.9)$$

$$z_a^- = -\frac{1}{2}(1 + \bar{\zeta})\|x_a^+ - x_a^-\| \quad (6.2.10)$$

where $\|\cdot\|$ denotes the Euclidean norm (i.e., $\|x\| = (x_1^2 + x_2^2 + x_3^2)^{1/2}$). An illustration of these ideas is presented in Fig. 6.2.2.

The top- and bottom-surface coordinates are uniquely defined at element interfaces. Consequently, there are no gaps or overlaps along element boundaries.

6.2.2 Lamina Coordinate Systems

At each integration point in the element, a Cartesian reference frame is erected so that two axes are tangent to the lamina through the point. The frame is defined by its orthonormal basis vectors e'_1 , e'_2 , e'_3 in which e'_3 is perpendicular to the lamina (see Fig. 6.2.3). The basis vectors are calculated as follows.

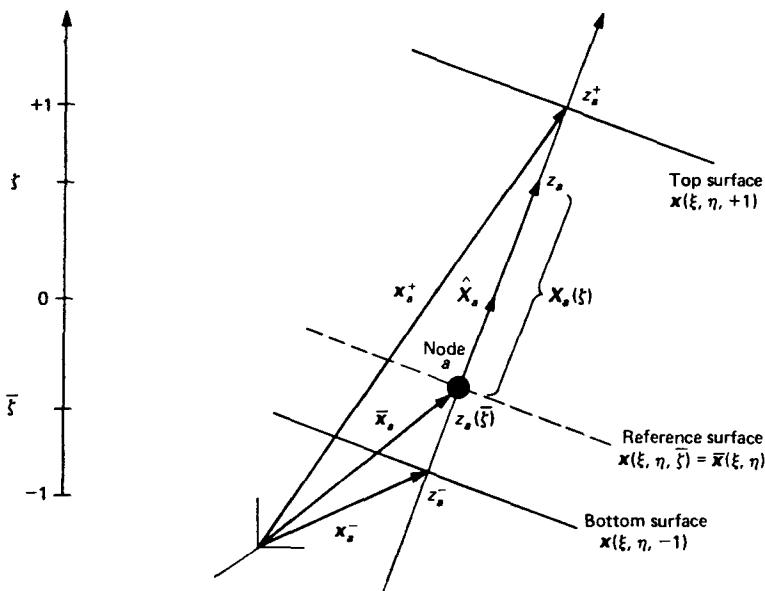


Figure 6.2.2

Construct unit tangent vectors to the ξ - and η -coordinate directions:

$$\mathbf{e}_\xi = \frac{\mathbf{x}_{,\xi}}{\|\mathbf{x}_{,\xi}\|} \quad (6.2.11)$$

$$\mathbf{e}_\eta = \frac{\mathbf{x}_{,\eta}}{\|\mathbf{x}_{,\eta}\|} \quad (6.2.12)$$

(6.2.11) and (6.2.12) suffice to define \mathbf{e}'_3 :

$$\mathbf{e}'_3 = \frac{\mathbf{e}_\xi \times \mathbf{e}_\eta}{\|\mathbf{e}_\xi \times \mathbf{e}_\eta\|} \quad (6.2.13)$$

The vectors tangent to the lamina are selected so that the angle between \mathbf{e}'_1 and \mathbf{e}_ξ is

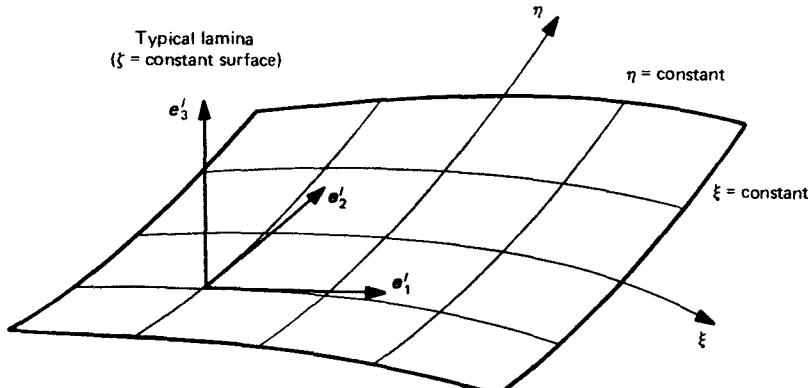


Figure 6.2.3 Typical lamina coordinate system.

the same as the angle between e_η and e_2' and so that the e_1', e_2' -basis is as “close” as possible to the e_ξ, e_η -basis. Thus

$$e_1' = \frac{\sqrt{2}}{2}(e_\alpha - e_\beta) \quad (6.2.14)$$

$$e_2' = \frac{\sqrt{2}}{2}(e_\alpha + e_\beta) \quad (6.2.15)$$

where

$$e_\alpha = \frac{\frac{1}{2}(e_\xi + e_\eta)}{\left\| \frac{1}{2}(e_\xi + e_\eta) \right\|} \quad (6.2.16)$$

$$e_\beta = \frac{e_3' \times e_\alpha}{\| e_3' \times e_\alpha \|} \quad (6.2.17)$$

Note that e_3' is not generally tangent to the fiber direction; see Fig. 6.2.4. The e_3' direction is used for purposes of invoking the plane stress hypothesis.

In the sequel, it will be necessary to transform quantities from the global coordinate system to the lamina system. This is facilitated by the following matrix:

$$q = [q_{ij}] = [e_1' \ e_2' \ e_3']^T: \text{global} \rightarrow \text{lamina} \quad (6.2.18)$$

in which the superscript T denotes *transpose*. Clearly, q is orthogonal.

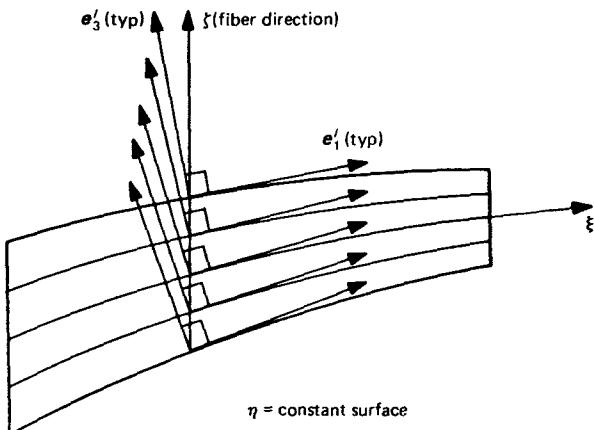


Figure 6.2.4 Lamina coordinate systems along a fiber.

6.2.3 Fiber Coordinate Systems

At each node a unique local Cartesian coordinate system is constructed, which is used as a reference frame for rotations. The only requirement that the frame must satisfy is that one direction coincide with the fiber direction. This in itself is not sufficient to define the frame. In many cases, the physical situation may be such that an “obvious” choice presents itself. For example, in the case of a cylindrical shell, it is natural to choose the cylindrical basis vectors: e_θ, e_z, e_r . When there is no obvious choice, an

algorithm may be employed to select the basis. For example, let \hat{X} denote the unit basis vector in the fiber direction. (We omit the nodal subscript a throughout this discussion for notational clarity.) Let e_1, e_2, e_3 denote the global Cartesian basis, i.e.,

$$e_1 = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix} \quad e_2 = \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix} \quad e_3 = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} \quad (6.2.19)$$

The global Cartesian components of \hat{X} are denoted by $\hat{X}_i, i = 1, 2, 3$.

Algorithm

1. Let $a_i = |\hat{X}_i|, i = 1, 2, 3$.
2. $j = 1$.
3. If $a_1 > a_3$, then $a_3 = a_1$, and $j = 2$.
4. If $a_2 > a_3, j = 3$.
5. $e'_3 = \hat{X}$.
6. $e'_2 = (\hat{X} \times e_j) / \|\hat{X} \times e_j\|$.
7. $e'_1 = e'_2 \times \hat{X}$.

The orthonormal fiber basis obtained (i.e., e'_1, e'_2, e'_3) satisfies the condition that if \hat{X} is “close” to e_3 , then e'_1, e'_2, e'_3 will be “close” to e_1, e_2, e_3 , respectively (see Fig. 6.2.5).

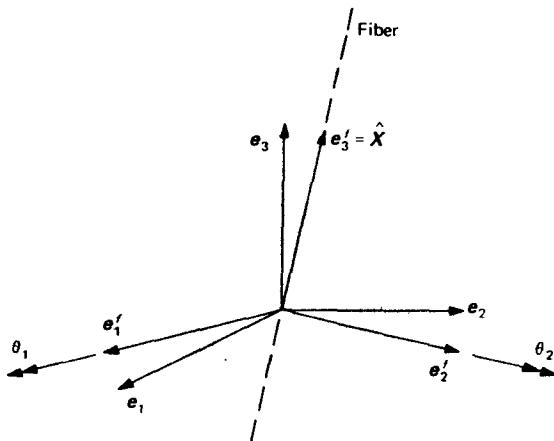


Figure 6.2.5 Nodal fiber basis and right-hand-rule sign convention for rotations.

6.2.4 Kinematics

The displacement of the shell is assumed to take the following form:

$$\mathbf{u}(\xi, \eta, \zeta) = \bar{\mathbf{u}}(\xi, \eta) + \mathbf{U}(\xi, \eta, \zeta) \quad (6.2.20)$$

$$\bar{\mathbf{u}}(\xi, \eta) = \sum_{a=1}^{n_a} N_a(\xi, \eta) \bar{\mathbf{u}}_a \quad (6.2.21)$$

$$U(\xi, \eta, \zeta) = \sum_{a=1}^{n_{en}} N_a(\xi, \eta) U_a(\zeta) \quad (6.2.22)$$

$$U_a(\zeta) = z_a(\zeta) \hat{U}_a \quad (\text{no sum}) \quad (6.2.23)$$

where u is the displacement of a generic point; \bar{u} is the displacement of a point on the reference surface; and U is the “fiber displacement”. The vector \hat{U}_a is constructed such that the fiber may rotate, but not stretch, viz.²,

$$\hat{U}_a = \theta_{a2} e'_{a1} - \theta_{a1} e'_{a2} \quad (6.2.24)$$

The quantities θ_{a1} and θ_{a2} represent the rotations of the fiber about the basis vectors e'_{a1} and e'_{a2} , respectively. The right-hand-rule sign convention is illustrated in Fig. 6.2.5. The “fiber inextensibility hypothesis,” manifested by (6.2.24), is analogous to plate Assumption 4, Sec. 5.2.1.

6.2.5 Reduced Constitutive Equation

We begin with the constitutive equation of three-dimensional elasticity *written with respect to the lamina coordinate system at the point under consideration*:

$$\sigma^l = D^l \epsilon^l \quad (6.2.25)$$

The superscript l is used to emphasize that the components are in the lamina system. The ordering of components is given as follows:

$$\sigma^l = \{\sigma_l^i\} = \left\{ \begin{array}{c} \sigma_{11}^l \\ \sigma_{22}^l \\ \sigma_{12}^l \\ \sigma_{23}^l \\ \sigma_{31}^l \\ \sigma_{33}^l \end{array} \right\} \quad (6.2.26)$$

$$\epsilon^l = \{\epsilon_l^i\} = \left\{ \begin{array}{c} \frac{\partial u_1^l}{\partial x_1^l} \\ \frac{\partial u_2^l}{\partial x_1^l} \\ \frac{\partial u_2^l}{\partial x_2^l} \\ \frac{\partial u_1^l}{\partial x_2^l} + \frac{\partial u_2^l}{\partial x_1^l} \\ \frac{\partial u_2^l}{\partial x_3^l} + \frac{\partial u_3^l}{\partial x_2^l} \\ \frac{\partial u_3^l}{\partial x_1^l} + \frac{\partial u_1^l}{\partial x_3^l} \\ \frac{\partial u_3^l}{\partial x_3^l} \end{array} \right\} \quad (6.2.27)$$

² Equation (6.2.24) is the linearized version of fiber inextensibility.

$$D_{IJ}^l = c_{ijklm}^l \quad (6.2.28)$$

| I/J | i/k | j/m |
|-------|-------|-------|
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 3 | 1 | 2 |
| 4 | 2 | 3 |
| 5 | 3 | 1 |
| 6 | 3 | 3 |

(6.2.29)

The constitutive equation needs to be modified in order to enforce the *zero normal-stress condition*³ in the 3-direction of the lamina system. This amounts to solving for the 33-component of strain in terms of the remaining components:

$$\sigma_6^l = 0 \Rightarrow \epsilon_6^l = - \frac{\left(\sum_{I=1}^5 D_{6I}^l \epsilon_I^l \right)}{D_{66}^l} \quad (6.2.30)$$

Equivalently, this may be expressed in tensor form:

$$\sigma_{33}^l = 0 \Rightarrow \epsilon_{33}^l = - \frac{\left(\sum_{ij \neq 33} c_{33ij}^l \epsilon_{ij}^l \right)}{c_{3333}^l} \quad (6.2.31)$$

Substituting (6.2.30) in (6.2.25) results in

$$\begin{aligned} \sigma_I^l &= \sum_{J=1}^5 D_{IJ}^l \epsilon_J^l + D_{I6}^l \epsilon_6^l \\ &= \sum_{J=1}^5 \left(\frac{D_{IJ}^l - D_{I6}^l D_{6J}^l}{D_{66}^l} \right) \epsilon_J^l \end{aligned} \quad (6.2.32)$$

and thus

$$\tilde{\sigma}^l = \tilde{D}^l \tilde{\epsilon}^l \quad (6.2.33)$$

where

³This is the analog of plate Assumption 2, Sec. 5.2.1.

$$\tilde{\sigma}^I = \begin{Bmatrix} \sigma_1^I \\ \sigma_2^I \\ \sigma_3^I \\ \sigma_4^I \\ \sigma_5^I \end{Bmatrix} \quad (6.2.34)$$

$$\tilde{\epsilon}^I = \begin{Bmatrix} \epsilon_1^I \\ \epsilon_2^I \\ \epsilon_3^I \\ \epsilon_4^I \\ \epsilon_5^I \end{Bmatrix} \quad (6.2.35)$$

$$\tilde{D}_{IJ}^I = [\tilde{D}_{IJ}^I], \quad 1 \leq I, J \leq 5 \quad (6.2.36)$$

$$\tilde{D}_{IJ}^I = D_{IJ}^I - \frac{D_{I6}^I D_{6J}^I}{D_{66}^I} \quad (6.2.37)$$

(6.2.33) is called the *reduced constitutive equation*. The reason for placing the 33-components in the last entries of σ and ϵ should now be apparent.

Exercise 1. Show that for the isotropic case

$$\tilde{D}^I = \frac{E}{(1 - \nu^2)} \begin{bmatrix} 1 & \nu & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ & & \frac{1-\nu}{2} & 0 & 0 \\ & & \text{symmetric} & \frac{1-\nu}{2} & 0 \\ & & & & \frac{1-\nu}{2} \end{bmatrix} \quad (6.2.38)$$

where E is Young's modulus and ν is Poisson's ratio.

Shear Correction Factors

To attain results consistent with classical bending theory, *shear correction factors* need to be introduced. In (6.2.38), this amounts to multiplying the transverse shearing moduli by $\kappa = \frac{5}{6}$, viz.,

$$\widetilde{D}^I = \frac{E}{(1 - \nu^2)} \begin{bmatrix} 1 & \nu & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 \\ & & \frac{1-\nu}{2} & 0 & 0 \\ \text{symmetric} & & & \frac{\kappa(1-\nu)}{2} & 0 \\ & & & & \frac{\kappa(1-\nu)}{2} \end{bmatrix} \quad (6.2.39)$$

6.2.6 Strain-displacement Matrix

In application to shells, special treatment needs to be given to transverse shear and membrane terms to prevent "mesh-locking" phenomena [5–13]. A particularly effective treatment may be performed by employing the reduced-selective integration concept. In the present formulation, we make use of the \bar{B} -method introduced in [14, 15] and presented in Chapter 4, which enables implementation of selective integration type procedures by a simple modification of the strain-displacement matrix. This technique has advantages in anisotropic situations because it engenders only a minor change to the standard element-array implementation.

The definition of the strain-displacement matrix adopted is given as follows:

$$\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_{n_{en}}] \quad (6.2.40)$$

The strain-displacement transformation may be written as

$$\underbrace{\widetilde{\boldsymbol{\epsilon}}}_{5 \times 1} = \sum_{a=1}^{n_{en}} \underbrace{\mathbf{B}_a}_{5 \times 5} \underbrace{\begin{Bmatrix} \bar{\mathbf{u}}_a \\ \theta_{a1} \\ \theta_{a2} \end{Bmatrix}}_{5 \times 1} \quad (6.2.41)$$

where each \mathbf{B}_a has the form:

$$\mathbf{B}_a = [b_I^u \ b_I^\theta] = \begin{bmatrix} b_1^u & b_1^\theta \\ b_2^u & b_2^\theta \\ b_3^u & b_3^\theta \\ \hline \cdots & \cdots \\ b_4^u & b_4^\theta \\ b_5^u & b_5^\theta \end{bmatrix} \quad (6.2.42)$$

The b_I^u 's and b_I^θ 's are row vectors:

$$b_I^u = \langle b_{Im}^u \rangle = \langle b_{I1}^u \ b_{I2}^u \ b_{I3}^u \rangle \quad (6.2.43)$$

$$b_I^\theta = \langle b_{I\alpha}^\theta \rangle = \langle b_{I1}^\theta \ b_{I2}^\theta \rangle \quad (6.2.44)$$

Explicit formulas may be obtained by calculating the displacement gradients:

$$\begin{aligned}\frac{\partial u_i^f}{\partial x_j^f} &= \sum_{m=1}^3 q_{im} \frac{\partial u_m}{\partial x_j^f} \\ &= \sum_{m=1}^3 q_{im} \sum_{a=1}^{n_a} \left(\frac{\partial N_a}{\partial x_j^f} \bar{u}_{am} + \frac{\partial (N_a z_a)}{\partial x_j^f} (\theta_{a2} e_{am1}^f - \theta_{a1} e_{am2}^f) \right)\end{aligned}\quad (6.2.45)$$

The notation

$$e_{aa}^f = \{e_{am\alpha}^f\} = \begin{Bmatrix} e_{a1\alpha}^f \\ e_{a2\alpha}^f \\ e_{a3\alpha}^f \end{Bmatrix} \quad (6.2.46)$$

has been used in (6.2.45). From (6.2.45), we may read the following definitions:

$$b_{1m}^u = q_{1m} \frac{\partial N_a}{\partial x_1^f} \quad (6.2.47)$$

$$b_{2m}^u = q_{2m} \frac{\partial N_a}{\partial x_2^f} \quad (6.2.48)$$

$$b_{3m}^u = q_{1m} \frac{\partial N_a}{\partial x_2^f} + q_{2m} \frac{\partial N_a}{\partial x_1^f} \quad (6.2.49)$$

$$b_{4m}^u = q_{2m} \frac{\partial N_a}{\partial x_3^f} + q_{3m} \frac{\partial N_a}{\partial x_2^f} \quad (6.2.50)$$

$$b_{5m}^u = q_{3m} \frac{\partial N_a}{\partial x_1^f} + q_{1m} \frac{\partial N_a}{\partial x_3^f} \quad (6.2.51)$$

$$b_{1\alpha}^\theta = \omega_{a1\alpha} \frac{\partial (N_a z_a)}{\partial x_1^f} \quad (6.2.52)$$

$$b_{2\alpha}^\theta = \omega_{a2\alpha} \frac{\partial (N_a z_a)}{\partial x_2^f} \quad (6.2.53)$$

$$b_{3\alpha}^\theta = \omega_{a1\alpha} \frac{\partial (N_a z_a)}{\partial x_2^f} + \omega_{a2\alpha} \frac{\partial (N_a z_a)}{\partial x_1^f} \quad (6.2.54)$$

$$b_{4\alpha}^\theta = \omega_{a2\alpha} \frac{\partial (N_a z_a)}{\partial x_3^f} + \omega_{a3\alpha} \frac{\partial (N_a z_a)}{\partial x_2^f} \quad (6.2.55)$$

$$b_{5\alpha}^\theta = \omega_{a3\alpha} \frac{\partial (N_a z_a)}{\partial x_1^f} + \omega_{a1\alpha} \frac{\partial (N_a z_a)}{\partial x_3^f} \quad (6.2.56)$$

where

$$\omega_{a11} = - \sum_{m=1}^3 q_{im} e_{am2}^f \quad (6.2.57)$$

$$\omega_{a12} = \sum_{m=1}^3 q_{im} e_{am1}^f \quad (6.2.58)$$

The differentiations indicated in (6.2.47) through (6.2.56) are calculated by transforming corresponding global quantities:

$$\frac{\partial}{\partial x_j^l} = \sum_{m=1}^3 q_{jm} \frac{\partial}{\partial x_m} \quad (6.2.59)$$

Let b denote any term in the matrix B_a . Suppose we wish to treat b with a reduced lamina quadrature rule. In this case b will be replaced by its reduced quadrature counterpart \bar{b} , which is defined as follows: Assume a lamina quadrature rule is specified to integrate the element stiffness. Think of this rule as the "normal" one for the element. Another quadrature rule is introduced which may be thought of as the "reduced" rule. For this rule, \bar{n}_{int} and $\bar{\xi}_l$, $\bar{\eta}_l$ denote the number of points and locations, respectively. A special set of shape functions, \bar{N}_l 's, is defined with nodes at the quadrature points (i.e., $N_k(\bar{\xi}_l, \bar{\eta}_l) = \delta_{kl}$, $1 \leq k, l \leq \bar{n}_{int}$).

For example, if the element under consideration was a quadrilateral and the reduced rule was the 2×2 Gauss-Legendre rule, then the \bar{N}_l 's would be bilinear functions interpolating the 2×2 Gauss points.

The general form of \bar{b} is given by

$$\bar{b}(\xi, \eta) = \sum_{l=1}^{\bar{n}_{int}} \bar{N}_l(\xi, \eta) b(\bar{\xi}_l, \bar{\eta}_l) \quad (6.2.60)$$

As an example, consider the four-node bilinear quadrilateral. The normal rule is the 2×2 Gauss-Legendre rule. Take as the reduced rule the one-point Gauss-Legendre rule, so that $\bar{n}_{int} = 1$, $\bar{N}_1 = 1$, and $\bar{\xi}_1 = \bar{\eta}_1 = 0$ (i.e., the element center). Then (6.2.60) reduces to

$$\bar{b}(\xi, \eta) = b(0, 0) \quad (6.2.61)$$

That is, the value at the center of the element is used to compute \bar{b} .

In shell elements, selective reduced integration has been used on membrane and transverse-shear terms to avoid locking. The terms in the matrix B_a that are affected are indicated next.

Reduced integration of transverse shear

$$B_a = \begin{bmatrix} b_1^u & b_1^g \\ b_2^u & b_2^g \\ b_3^u & b_3^g \\ \hline \bar{b}_4^u & \bar{b}_4^g \\ \bar{b}_5^u & \bar{b}_5^g \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \hline \times & \times & \\ \times & \times & \end{bmatrix} \quad (\times \text{ signifies location of affected terms}) \quad (6.2.62)$$

Uniform reduced integration of membrane effects

$$B_a = \begin{bmatrix} \bar{b}_1^u & b_1^g \\ \bar{b}_2^u & b_2^g \\ \bar{b}_3^u & b_3^g \\ \hline b_4^u & b_4^g \\ b_5^u & b_5^g \end{bmatrix} = \begin{bmatrix} \times & \cdot & \\ \times & \cdot & \\ \times & \cdot & \\ \hline \cdot & \cdot & \\ \cdot & \cdot & \end{bmatrix} \quad (6.2.63)$$

Selective reduced integration of membrane effects

In this case, only the membrane shear-strain term is underintegrated:

$$\mathbf{B}_a = \begin{bmatrix} b_1^u & b_1^\theta \\ b_2^u & b_2^\theta \\ \bar{b}_3^u & b_3^\theta \\ b_4^u & b_4^\theta \\ b_5^u & b_5^\theta \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \quad (6.2.64)$$

Note that there is no “invariance” problem in this case, since the entries are defined with respect to lamina coordinates, which in turn are defined intrinsically by the element geometry.

Cases (6.2.62) and (6.2.63), or (6.2.62) and (6.2.64), may be combined. For example,

Reduced integration of transverse shear and membrane effects:

$$\mathbf{B}_a = \begin{bmatrix} \bar{b}_1^u & b_1^\theta \\ \bar{b}_2^u & b_2^\theta \\ \bar{b}_3^u & b_3^\theta \\ \bar{b}_4^u & \bar{b}_4^\theta \\ \bar{b}_5^u & \bar{b}_5^\theta \end{bmatrix} = \begin{bmatrix} \times & \cdot \\ \times & \cdot \\ \times & \cdot \\ \times & \times \\ \times & \times \end{bmatrix} \quad (6.2.65)$$

Remark

In certain situations, different reduced rules may be employed on different terms. For example, suppose that two different reduced rules are used, one for membrane effects (“ \bar{b} -treatment”) and one for transverse shear (“ $\bar{\bar{b}}$ -treatment”). Then

$$\mathbf{B}_a = \begin{bmatrix} \bar{b}_1^u & b_1^\theta \\ \bar{b}_2^u & b_2^\theta \\ \bar{b}_3^u & b_3^\theta \\ \bar{\bar{b}}_4^u & \bar{b}_4^\theta \\ \bar{b}_5^u & \bar{b}_5^\theta \end{bmatrix} = \begin{bmatrix} \times & \cdot \\ \times & \cdot \\ \times & \cdot \\ \times & \times \\ \times & \times \end{bmatrix} \quad (6.2.66)$$

Quite elaborate schemes, involving several different rules, have been developed to produce effective shell elements (e.g., see MacNeal’s description of the QUAD4 shell element used in NASTRAN [16]).

Exercise 2. The displacement interpolation is not form-identical to the geometric mapping.

Thus the elements may not be classified as isoparametric. However, it may be shown that the displacement interpolation entails all rigid body modes and constant strain states. Present an argument which corroborates this assertion.

6.2.7 Stiffness Matrix

The element stiffness is defined as follows:

$$\underbrace{\mathbf{k}}_{5n_{en} \times 5n_{en}} = [\mathbf{k}_{ab}] \quad (6.2.67)$$

$$\underbrace{\mathbf{k}_{ab}}_{5 \times 5} = \int_{\square} \int_{-1}^{+1} \mathbf{B}_a^T \tilde{\mathbf{D}}' \mathbf{B}_b j \, d\xi \, d\eta \quad (6.2.68)$$

fiber integral

where

$$\int_{\square} \dots \, d\square = \int_{-1}^{+1} \int_{-1}^{+1} \dots \, d\xi \, d\eta \quad (\text{lamina integral}) \quad (6.2.69)$$

$$j = \det \begin{bmatrix} x_{1,\xi} & x_{1,\eta} & x_{1,\zeta} \\ x_{2,\xi} & x_{2,\eta} & x_{2,\zeta} \\ x_{3,\xi} & x_{3,\eta} & x_{3,\zeta} \end{bmatrix} \quad (6.2.70)$$

6.2.8 External Force Vector

We allow for body, surface, and edge force vectors.

Body force

The element body force vector is given by

$$\underbrace{\mathbf{f}^{\text{body}}}_{5n_{en} \times 1} = \{f_a^{\text{body}}\} \quad (6.2.71)$$

$$\underbrace{f_a^{\text{body}}}_{5 \times 1} = \int_{\square} \int_{-1}^{+1} \mathbf{N}_a^T \boldsymbol{\ell} j \, d\xi \, d\eta \quad (6.2.72)$$

where

$$\mathbf{N}_a = \begin{bmatrix} N_a & 0 & 0 & -N_a z_a e_{a12}^f & N_a z_a e_{a11}^f \\ 0 & N_a & 0 & -N_a z_a e_{a22}^f & N_a z_a e_{a21}^f \\ 0 & 0 & N_a & -N_a z_a e_{a32}^f & N_a z_a e_{a31}^f \end{bmatrix} \quad (6.2.73)$$

The definition of \mathbf{N}_a follows directly from the kinematic assumptions.

Surface force

The element surface force vector is defined by

$$f_a^{\text{surf}} = \{f_a^{\text{surf}}\} \quad (6.2.74)$$

$$f_a^{\text{surf}} = \int_{\square} N_a^T \mathbf{h} j_s d\square, \quad \zeta = \begin{cases} +1 & \text{top} \\ -1 & \text{bottom} \end{cases} \quad (6.2.75)$$

where

$$j_s = \|x_{,\xi} \times x_{,\eta}\| \quad (\text{surface Jacobian}) \quad (6.2.76)$$

and \mathbf{h} is the surface force vector (per unit surface area).

It is convenient to allow for pressure and shear surface force vectors as separate cases.

Pressure

In this case

$$\mathbf{h} = -\zeta p \mathbf{n}, \quad (\zeta = +1 \text{ or } -1) \quad (6.2.77)$$

$$\mathbf{n} = \frac{\mathbf{e}_\xi \times \mathbf{e}_\eta}{\|\mathbf{e}_\xi \times \mathbf{e}_\eta\|} \quad (6.2.78)$$

where p is the pressure and \mathbf{n} is the unit normal vector to the surface.

Shear

We assume that the shear is specified in the ξ and/or η directions on the surface in question. In this case the surface force vector is given by

$$\mathbf{h} = h_\xi \mathbf{e}_\xi + h_\eta \mathbf{e}_\eta \quad (6.2.79)$$

where h_ξ and h_η are the shears in the ξ and η directions, respectively.

Edge force

Suppose we wish to apply a distributed loading along an $\eta = +1$ or -1 edge. Let \mathbf{h} denote the distributed surface force. The nodal forces are

$$f_a^{\text{edge}} = \int_{-1}^{+1} \int_{-1}^{+1} (N_a^T \mathbf{h} j_e) \Big|_{\eta=+1 \text{ or } -1} d\xi d\eta \quad (6.2.80)$$

where

$$j_e = \|x_{,\xi} \times x_{,\zeta}\| \quad (\text{edge surface Jacobian}) \quad (6.2.81)$$

The case of loading along an $\xi = +1$ or -1 edge is handled by interchanging ξ and

η in (6.2.80) and (6.2.81). Note that when the reference surface is not taken to be the midsurface, nodal moments are produced even when h is constant (i.e., in general, $f_{a4} \neq 0, f_{a5} \neq 0$).

If edge forces or moments are specified per unit edge length, then nodal forces are computed as follows: Consider an $\eta = +1$ or -1 edge. Let $f_i^{\text{line}} = f_i^{\text{line}}(\xi)$ denote the edge force and let $m_i^{\text{line}} = m_i^{\text{line}}(\xi)$ denote the edge moment. The nodal forces are then given by

$$f_a^{\text{edge}} = \int_{-1}^{+1} N_a \left| \begin{array}{c} \eta = +1 \text{ or } -1 \\ \hline \end{array} \right\} \begin{Bmatrix} f_1^{\text{line}} \\ f_2^{\text{line}} \\ f_3^{\text{line}} \\ \hline m_1^{\text{line}} \\ m_2^{\text{line}} \end{Bmatrix} \|x_{,\xi}\| d\xi \quad (6.2.82)$$

Note that m_1^{line} and m_2^{line} must have the same sense as θ_1 and θ_2 . The result is made applicable to an $\xi = +1$ or -1 edge if ξ and η are interchanged in (6.2.82).

The element external force vector is defined by

$$\mathbf{f}^{\text{ext}} = \mathbf{f}^{\text{body}} + \mathbf{f}^{\text{surf}} + \mathbf{f}^{\text{edge}} \quad (6.2.83)$$

6.2.9 Fiber Numerical Integration

In the general case, fiber integrals need be evaluated by a numerical integration technique. Several ways of going about this present themselves, each having advantages in certain circumstances.

If the integrand is a smooth function of ζ (e.g., when the shell consists of one homogeneous layer), then Gaussian quadrature is most efficient. If the reference surface is taken to be the midsurface, then the one-point Gauss rule (i.e., midpoint rule) senses membrane and transverse-shear effects, whereas at least two points are required to manifest the bending behavior. In the one-layer case, an efficient alternative is to perform the fiber integration analytically as in [12]. See also [17] for recent developments.

If the shell is built up from a series of layers of different materials such that the material properties and stresses are discontinuous functions of ζ , then Gaussian rules may be effectively used over each layer. If there are a large number of approximately equal-sized layers, then the midpoint rule on each layer should suffice. If, on the other hand, there are a small number of layers or if the layers vary considerably in thickness, then different Gaussian rules should be assigned to individual layers. This may be

facilitated by allowing the user to input the location and weights of the fiber quadrature rule. Thus any special set of circumstances may be accommodated.

6.2.10 Stress Resultants

Bending moments, membrane forces and transverse shear resultants may be computed at any lamina point, say, $\xi = (\xi, \eta)$.

Moments

The moments may be calculated from the following expressions:

$$m_{\alpha\beta}(\xi) = \int_{-1}^{+1} \sigma'_{\alpha\beta}(\xi, \zeta) z(\xi, \zeta) d\zeta z_{,\zeta}(\xi), \quad 1 \leq \alpha, \beta \leq 2 \quad (6.2.84)$$

$$z(\xi, \zeta) = N_+(\zeta) z^+(\xi) + N_-(\zeta) z^-(\xi) \quad (6.2.85)$$

$$z^\pm(\xi) = \sum_{a=1}^{n_m} N_a(\xi) z_a^\pm \quad (6.2.86)$$

$$z_{,\zeta}(\xi) = \frac{z^+(\xi) - z^-(\xi)}{2} \quad (6.2.87)$$

Membrane forces

The membrane forces may be computed as follows:

$$n_{\alpha\beta}(\xi) = \int_{-1}^{+1} \sigma'_{\alpha\beta}(\xi, \zeta) d\zeta z_{,\zeta}(\xi), \quad 1 \leq \alpha, \beta \leq 2 \quad (6.2.88)$$

Transverse shears

The shears may be computed from the following formula:

$$q_\alpha(\xi) = \int_{-1}^{+1} \sigma'_{\alpha 3}(\xi, \zeta) d\zeta z_{,\zeta}(\xi), \quad 1 \leq \alpha \leq 2 \quad (6.2.89)$$

The integrations of (6.2.84), (6.2.88), and (6.2.89) are performed using the quadrature points of the fiber rule.

The sign conventions for the stress resultants are illustrated in Fig. 6.2.6.

6.2.11 Shell Elements

The development of shell elements is still an active area of research. We shall describe only some of the simpler elements currently used.

Lagrange Elements

The lamina shape functions and quadrature rules for the Lagrange elements are shown in Fig. 6.2.7. In each case the appropriate reduced rule is one order lower than the

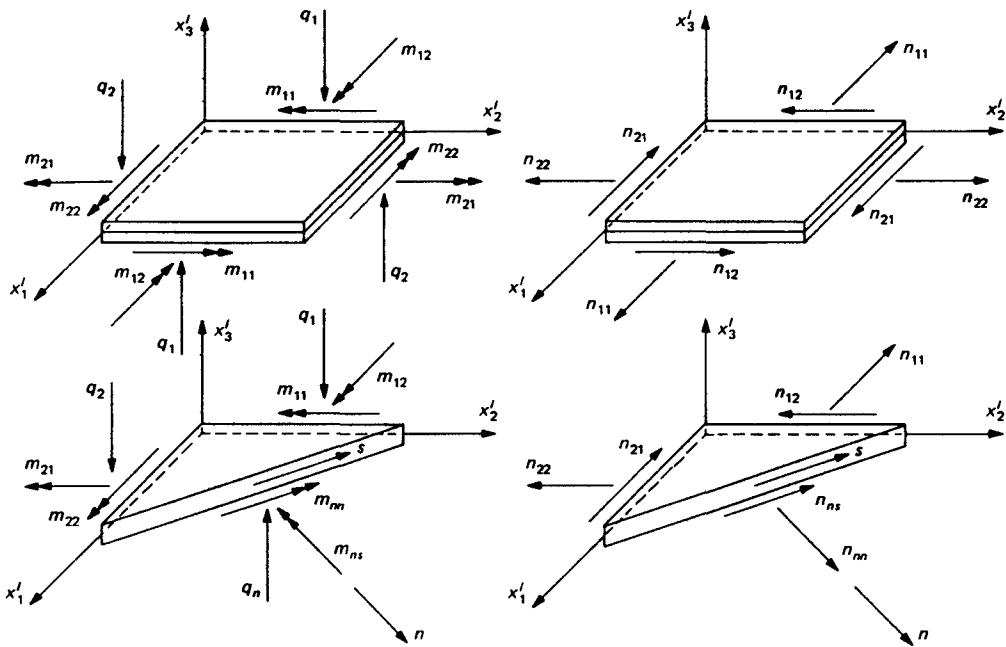


Figure 6.2.6 Sign conventions for stress resultants.

| | | | |
|------------------------|--------------|--------------|--------------|
| Lamina shape functions | Bilinear | Biquadratic | Bicubic |
| Normal Gaussian rule | 2×2 | 3×3 | 4×4 |
| Reduced Gaussian rule | 1×1 | 2×2 | 3×3 |

Figure 6.2.7 Lagrange shell elements.

normal rule. If the normal rule and reduced rule are combined, as described in Sec. 6.2.6, the element is called a *selective integration element*. If either the normal or reduced rule is used exclusively, then the element is called a *uniform integration element*. Uniform normal integration tends to cause elements to “lock” in thin shell applications. This phenomenon is especially pronounced for low-order elements but lessens as the order of interpolation is increased (e.g., the normally integrated bicubic element is felt by some to be a fairly good performer). On the other hand, selective and uniform reduced integration elements behave well in thin shell applications but may occasionally engender rank deficiency (i.e., spurious “mechanisms”). The problem is less acute for the selective integration elements than for the uniform reduced integration elements. In some situations, the mechanisms are precluded from global forming by boundary conditions, but nevertheless they represent a potentially dangerous deficiency.

Research has been undertaken to efficiently remove mechanisms. One successful procedure is described next.

Heterosis Element

The heterosis concept was originally developed in [5] to eliminate the spurious zero energy mode in the nine-node, selectively integrated Lagrange plate element (see also Chapter 5). The resulting element possesses correct rank and behaves well in the thick plate limit.

Implementation of the heterosis shell element begins by constructing the array for the selectively integrated Lagrange element (e.g., say \mathbf{k}_{Lag} , \mathbf{f}_{Lag} , etc.). Then projection matrix, \mathbf{H} , is constructed from the serendipity shape functions associated with the element-boundary nodes. The role \mathbf{H} plays is to eliminate all translational degrees of freedom from the internal (i.e., nonboundary) nodes by restraining them to interpolate the serendipity shape functions (see Sec. 5.3.7 for analogous details for the plate). The heterosis arrays are then defined by

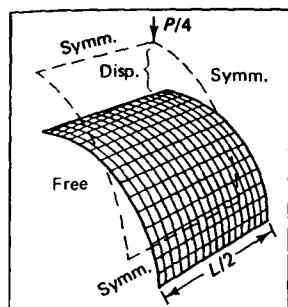
$$\mathbf{k}_{\text{het}} = \mathbf{H}^T \mathbf{k}_{\text{Lag}} \mathbf{H} \quad (6.2.90)$$

$$\mathbf{f}_{\text{het}} = \mathbf{H}^T \mathbf{f}_{\text{Lag}} \quad (6.2.91)$$

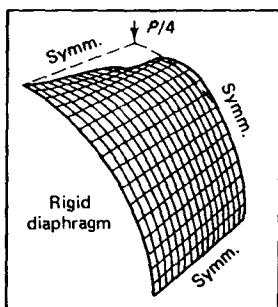
At this point, if desired, the internal rotational degrees of freedom may be removed by static condensation. Both transverse shear and membrane effects need to be treated with reduced quadrature to avoid locking. The heterosis shell element has been around for a while but still remains one of the better elements for general shell analysis [17].

Example. (Pinched Cylinder [17])

A thin circular cylinder of length L , radius R , and thickness t was subjected to equal and opposite point loads P as shown in Fig. 6.2.8. Due to symmetry, only one octant of the cylinder needed to be modeled. Two end conditions were considered: free ends and ends restrained by a rigid diaphragm, Figs. 6.2.8(a) and 6.2.8(b), respectively. Uniform and selective integration elements were employed in a convergence study. For the selective integration elements, bending terms were treated with the “normal rule” and membrane and shear terms were treated with the “reduced rule”; see (6.2.65). Results for the following elements are presented:

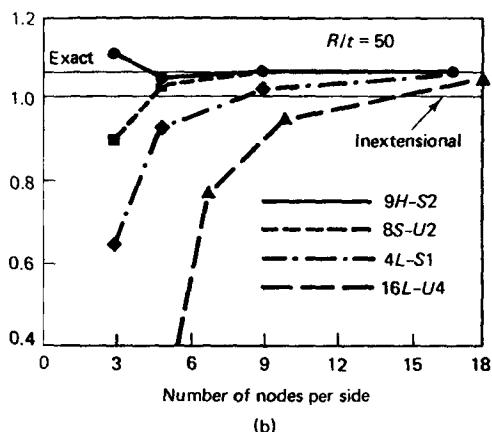


(a)

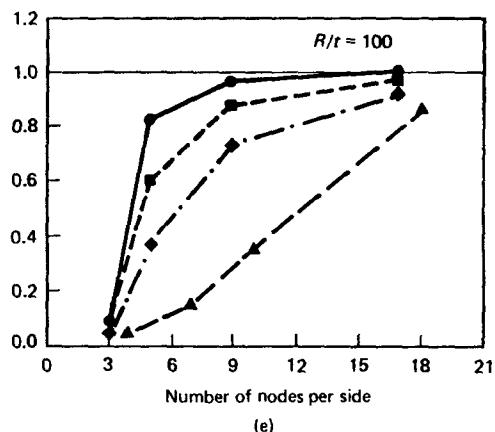


(d)

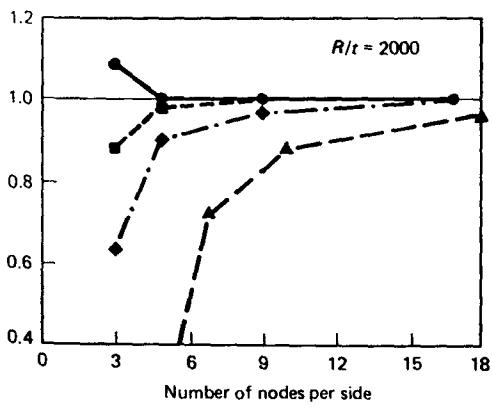
Displacement under the load normalized
with respect to thin shell theory



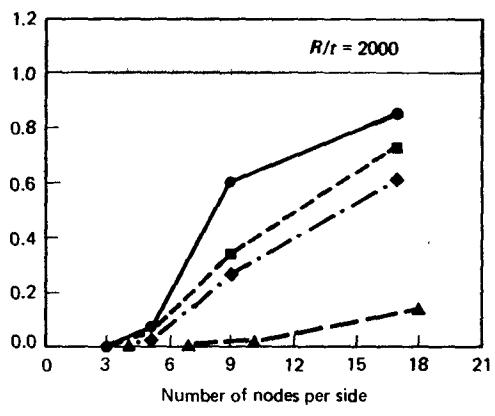
(b)



(e)



(c)



(f)

Figure 6.2.8 Convergence study for pinched cylinder.

- 9H-S2** Nine-node heterosis element with selective integration.
8S-U2 Eight-node serendipity element with uniform integration.
4L-S1 Four-node Lagrange element with selective integration.
16L-U4 Sixteen-node Lagrange element with uniform integration.

The integration rules are designated by:

$$\begin{aligned} Un & \quad n \times n \text{ Gaussian quadrature.} \\ Sn & \quad \begin{cases} (n+1) \times (n+1) \text{ Gaussian quadrature normal rule.} \\ n \times n \text{ Gaussian quadrature reduced rule.} \end{cases} \end{aligned}$$

It may be seen from the results that convergence for the open-ended case is more rapid than for the rigid-diaphragm case. In the latter case, when $R/t = 2000$, convergence is particularly slow; see Fig. 6.2.8(f). The reason for this is the exact solution involves highly localized deformations in the vicinity of the load, which are poorly resolved by the uniform meshes employed. Shells often exhibit fine-scale behavior, and the need for locally refined meshes is apparent.

In all cases presented, the heterosis element converges fastest while the fully integrated Lagrange element (16L-U4) converges slowest. The latter element possesses correct rank. Due to the underintegration of membrane terms, the heterosis element possesses one mechanism (see Fig. 4.6.3). However, this mechanism is non-communicable in a mesh of two or more elements so, for all practical purposes, this element may be viewed as having correct rank. The four-node Lagrange element (4L-S1) possesses four mechanisms: two membrane hourglass modes (see Fig. 4.6.2); and the in-plane twist mode and transverse-displacement hourglass mode (see Fig. 5.3.9). The three hourglass modes are communicable and, under certain circumstances, can be aroused globally. Consequently, extreme caution is advised in the use of this element. For further evaluations of these and other elements see [17]. Likewise, the serendipity element (8S-U2) possesses two noncommunicable mechanisms.

6.2.12 Some References to the Recent Literature

The thesis of Stanley [17] compares many existing shell elements and includes evaluation of some recently developed elements, such as the nine-node element of Park and Stanley [19] and the four-node elements of Dvorkin and Bathe [20] and Park, Stanley, and Flaggs [21]. See also Stanley, Park, and Hughes [22]. Belytschko, Liu, and their associates have developed several new improved elements in recent years. See, for example, Belytschko et al. [23, 24]. Most of the new elements use some form of strain projection or interpolation and thus may be viewed as “ \bar{B} -elements” (see Sec. 4.5.2). Another successful element within this category is the nine-node element of Huang and Hinton [25]. Hallquist and his colleagues at the Lawrence Livermore National Laboratory have developed an unprecedented capability for fully nonlinear transient analysis based upon one-point quadrature elements with hourglass control (e.g., see Hallquist et al. [26]).

There is great research interest in the development of shell finite element analysis procedures. The state of the art is summarized in Hughes and Hinton [27].

6.2.13 Simplifications: Shells as an Assembly of Flat Elements

A simple but crude approximation to curved shells may be constructed by way of flat elements. For example, consider the shell geometry discretized into flat triangles. A local Cartesian coordinate system is constructed in the plane of each triangle. A plate bending element stiffness and “membrane” element stiffness (i.e., plane-stress two-dimensional element stiffness) are generated and combined in this coordinate system. Note that there is no coupling in the local system because the plate and membrane have different degrees of freedom. The stiffness matrix and force vector are then transformed from local coordinates to global coordinates before assembly. The procedure is the same as for the straight beam-frame element described in Chapter 5. Further details follow:

Consider an n_{en} -noded flat element. Assume an element Cartesian coordinate system defined by a basis $\{e_1^e, e_2^e, e_3^e\}$ in which e_3^e is normal to the element and e_1^e , and e_2^e are in the plane of the element. The global-local transformation is defined by

$$t^e = [t_{ij}^e] \quad 1 \leq i, j \leq 3 \quad (6.2.92)$$

$$t_{ij}^e = e_i \cdot e_j^e \quad (6.2.93)$$

where e_1, e_2, e_3 is the global Cartesian basis. Let

$$k_m^e = 2n_{en} \times 2n_{en} \text{ membrane stiffness}$$

$$k_b^e = 3n_{en} \times 3n_{en} \text{ bending stiffness}$$

These are placed in an element matrix of dimension $6n_{en} \times 6n_{en}$:

$$k^e = \begin{bmatrix} k_m^e & 0 & 0 \\ 0 & k_b^e & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (6.2.94)$$

If adjacent elements are in the same plane, rank deficiency can clearly occur by virtue of the zero stiffness associated with rotations about the normal (see (6.2.94)). Ill-conditioning may also occur if the planes defined by adjacent elements almost coincide. For these reasons, a “fictitious” $n_{en} \times n_{en}$ stiffness, k_f^e , is often added to stabilize the assemblage:

$$k^e = \begin{bmatrix} k_m^e & 0 & 0 \\ 0 & k_b^e & 0 \\ 0 & 0 & k_f^e \end{bmatrix} \quad (6.2.95)$$

The fictitious stiffness resists rotations about the normal to the plane of the element. These are sometimes referred to as the *drilling degrees of freedom*. See [28] for a study of this approach. The rows and columns of k^e are now reordered so that the three displacement degrees of freedom at each node precede the three rotational degrees of freedom. The situation is very much like that for the straight beam element in three-dimensional space presented in Chapter 5. The transformation from local to global is performed as follows:

$$\underbrace{\hat{k}^e}_{6n_{en} \times 6n_{en}} = T^e \tilde{k}^e T^{eT} \quad (6.2.96)$$

where \tilde{k}^e is the reordered version of k^e and

$$\underbrace{T^e}_{6n_{en} \times 6n_{en}} = \begin{bmatrix} t^e & 0 & \cdots & 0 \\ 0 & t^e & & \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & t^e \end{bmatrix} \quad (6.2.97)$$

The (reordered) element force vector is similarly transformed:

$$\underbrace{\hat{f}^e}_{6n_{en} \times 1} = T^e \tilde{f}^e \quad (6.2.98)$$

Assembly may now be performed in the usual fashion.

An intriguing alternative to the use of a fictitious stiffness is to employ a membrane element that incorporates drilling degrees of freedom. Until recently, a successful formulation of this type had never been developed. In an excellent paper, Bergan and Felippa [29] have developed an accurate membrane element with drilling degrees of freedom. Taylor and Simo [30] have also developed a triangular element of this kind and linked it with the DKT bending element to create a C^0 -compatible shell element.

6.3 SHELLS OF REVOLUTION; RINGS AND TUBES IN TWO DIMENSIONS

6.3.1 Geometric and Kinematic Descriptions

The geometric and kinematic assumptions for the two-dimensional cases follow directly from the three-dimensional case treated in Sec. 6.2 by omitting the ξ -dependence and ignoring the third (out-of-plane) component of all vectors. In this case, the geometry of the shell is represented by a two-dimensional quadrilateral section as shown in Fig. 6.3.1. As in Sec. 6.2, the ζ -coordinate defines the fiber direction, and the η -coordinate lines coincide with shell laminae. This may be expressed by

$$\begin{aligned} x(\eta, \zeta) &= \bar{x}(\eta) + X(\eta, \zeta) \\ &= \sum_{a=1}^{n_{en}} N_a(\eta) \bar{x}_a + \sum_{a=1}^{n_{en}} N_a(\eta) z_a(\zeta) \hat{X}_a \\ &= \sum_{a=1}^{n_{en}} N_a(\eta) (\bar{x}_a + z_a(\zeta) \hat{X}_a) \end{aligned} \quad (6.3.1)$$

where $N_a(\eta)$ represents a one-dimensional lamina shape function and \bar{x}_a , $z_a(\zeta)$, and \hat{X}_a are defined by (6.2.5) through (6.2.10).

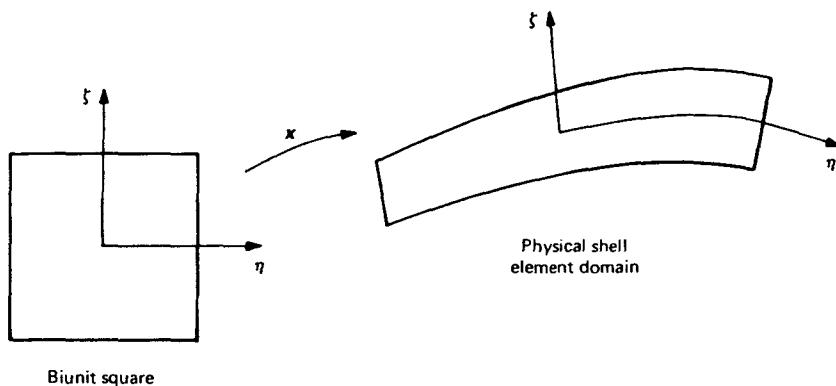


Figure 6.3.1

Lamina coordinate systems

Lamina coordinate systems are defined by (see Fig. 6.3.2)

$$\mathbf{e}_1^l = \begin{Bmatrix} e_{11}^l \\ e_{12}^l \end{Bmatrix} = \frac{\mathbf{x}_{,\eta}}{\|\mathbf{x}_{,\eta}\|} \quad (6.3.2)$$

$$\mathbf{e}_2^l = e_{11}^l \mathbf{e}_2 - e_{12}^l \mathbf{e}_1 \quad (6.3.3)$$

where

$$\mathbf{e}_1 = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}, \quad \mathbf{e}_2 = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \quad (6.3.4)$$

The global-lamina transformation is defined by

$$\mathbf{q} = [q_{ij}] = [\mathbf{e}_1^l \mathbf{e}_2^l]^T : \text{global} \rightarrow \text{lamina} \quad (6.3.5)$$

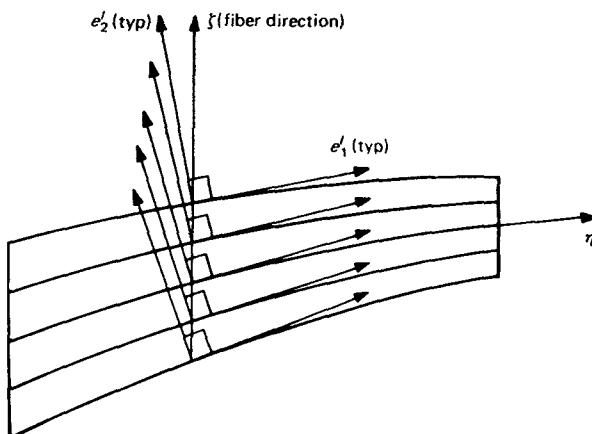


Figure 6.3.2 Lamina coordinate systems along a fiber.

Fiber coordinate systems

The fiber coordinate basis is defined as follows:

$$\mathbf{e}_2^f = \hat{\mathbf{X}} \quad (6.3.6)$$

$$\mathbf{e}_1^f = \hat{\mathbf{X}}_2 \mathbf{e}_1 - \hat{\mathbf{X}}_1 \mathbf{e}_2 \quad (6.3.7)$$

The kinematics are defined by

$$\mathbf{u}(\eta, \zeta) = \sum_{a=1}^{n_{en}} N_a(\eta)(\bar{\mathbf{u}}_a - z_a(\zeta)\theta_a \mathbf{e}_{a1}^f) \quad (6.3.8)$$

See Fig. 6.3.3 for an interpretation of the nodal quantities.

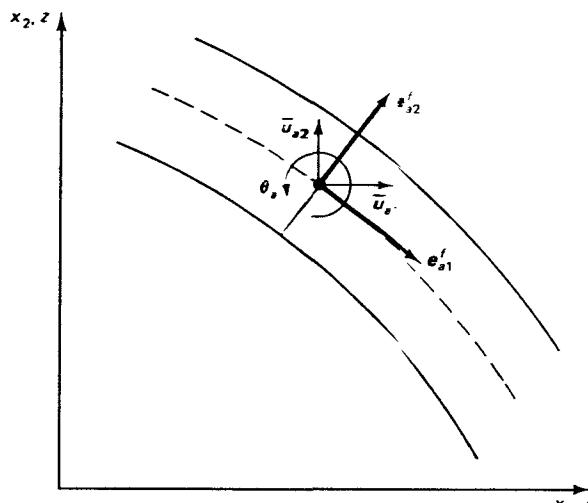


Figure 6.3.3 Nodal kinematic quantities for the two-dimensional cases.

6.3.2 Reduced Constitutive Equations

We begin with the three-dimensional constitutive equation written in lamina coordinates

$$\boldsymbol{\sigma}^l = \mathbf{D}^l \boldsymbol{\epsilon}^l \quad (6.3.9)$$

The ordering is indicated by the following

$$\boldsymbol{\sigma}^l = \{\sigma_i^l\} = \left\{ \begin{array}{l} \sigma_{11}^l \\ \sigma_{12}^l \\ \sigma_{33}^l \\ \sigma_{22}^l \\ \sigma_{23}^l \\ \sigma_{31}^l \end{array} \right\} \quad (6.3.10)$$

In what follows, we will assume that

$$\sigma'_{22} = 0 \quad (\text{zero normal stress in lamina coordinates}) \quad (6.3.11)$$

$$\sigma'_{23} = \sigma'_{31} = 0 \quad (\text{zero out-of-plane shear stresses}) \quad (6.3.12)$$

As in the three-dimensional case, the corresponding strain components can be eliminated,

$$\begin{Bmatrix} \epsilon'_1 \\ \epsilon'_5 \\ \epsilon'_6 \end{Bmatrix} = - \begin{bmatrix} D'_{44} & D'_{45} & D'_{46} \\ D'_{54} & D'_{55} & D'_{56} \\ D'_{64} & D'_{65} & D'_{66} \end{bmatrix}^{-1} \begin{bmatrix} D'_{41} & D'_{42} & D'_{43} \\ D'_{51} & D'_{52} & D'_{53} \\ D'_{61} & D'_{62} & D'_{63} \end{bmatrix} \begin{Bmatrix} \epsilon'_1 \\ \epsilon'_2 \\ \epsilon'_3 \end{Bmatrix} \quad (6.3.13)$$

resulting in a statically condensed matrix of elastic coefficients:

$$\tilde{D}' = \begin{bmatrix} D'_{11} & D'_{12} & D'_{13} \\ D'_{21} & D'_{22} & D'_{23} \\ D'_{31} & D'_{32} & D'_{33} \end{bmatrix} - \begin{bmatrix} D'_{14} & D'_{15} & D'_{16} \\ D'_{24} & D'_{25} & D'_{26} \\ D'_{34} & D'_{35} & D'_{36} \end{bmatrix} \begin{bmatrix} D'_{44} & D'_{45} & D'_{46} \\ D'_{54} & D'_{55} & D'_{56} \\ D'_{64} & D'_{65} & D'_{66} \end{bmatrix}^{-1} \begin{bmatrix} D'_{41} & D'_{42} & D'_{43} \\ D'_{51} & D'_{52} & D'_{53} \\ D'_{61} & D'_{62} & D'_{63} \end{bmatrix} \quad (6.3.14)$$

The reduced constitutive equation is

$$\tilde{\sigma}' = \tilde{D}' \tilde{\epsilon}' \quad (6.3.15)$$

where

$$\tilde{\sigma}' = \begin{Bmatrix} \sigma'_1 \\ \sigma'_2 \\ \sigma'_3 \end{Bmatrix} = \begin{Bmatrix} \sigma'_{11} \\ \sigma'_{12} \\ \sigma'_{33} \end{Bmatrix}, \text{ etc.} \quad (6.3.16)$$

Exercise 1. Consider the isotropic case. Show that

$$\tilde{D}' = \frac{E}{(1-\nu^2)} \begin{bmatrix} 1 & 0 & \nu \\ 0 & \frac{1-\nu}{2} & 0 \\ \nu & 0 & 1 \end{bmatrix} \quad (6.3.17)$$

As always, a shear correction factor, $\kappa = \frac{5}{6}$, needs to be introduced to match classical bending results:

$$\tilde{D}' = \frac{E}{(1-\nu^2)} \begin{bmatrix} 1 & 0 & \nu \\ 0 & \frac{\kappa(1-\nu)}{2} & 0 \\ \nu & 0 & 1 \end{bmatrix} \quad (6.3.18)$$

The above result may be used in the *axisymmetric* and *plane strain* cases. In the case of *plane stress*, further simplifications ensue. In this case the out-of-plane normal stress, σ'_{33} , is also assumed zero. Proceeding along now familiar lines:

$$\epsilon'_3 = - \left(\sum_{j=1}^2 \frac{\tilde{D}'_{3j}\epsilon'_j}{\tilde{D}'_{33}} \right) \quad (6.3.19)$$

$$\tilde{D}'_{1j} \leftarrow \tilde{D}'_{1j} - \frac{\tilde{D}'_{13}\tilde{D}'_{3j}}{\tilde{D}'_{33}} \quad (6.3.20)$$

Exercise 2. Show that in the isotropic case, the plane stress assumption causes (6.3.18) to be replaced with:

$$\boxed{\tilde{D}' = \begin{bmatrix} E & 0 & 0 \\ 0 & \frac{\kappa E}{2(1+\nu)} & 0 \\ 0 & 0 & 0 \end{bmatrix}} \quad (6.3.21)$$

Remark

The various two-dimensional cases have the following physical applications (see Fig. 6.3.4):

Axisymmetric: shells of revolution subjected to axisymmetric loading.

Plane strain: long tubes loaded in the plane perpendicular to the generating axis with the load not dependent upon the axial coordinate.

Plane stress: rings and curved beams with rectangular cross sections loaded in plane.

6.3.3 Strain-displacement Matrix

The strains are defined in terms of the displacement by

$$\tilde{\epsilon}' = \{ \tilde{\epsilon}' \} = \left\{ \begin{array}{c} \frac{\partial u'_1}{\partial x'_1} \\ \frac{\partial u'_1}{\partial x'_2} + \frac{\partial u'_2}{\partial x'_1} \\ \vdots \\ \frac{u_1}{x_1} \end{array} \right\} \quad (\text{axisymmetric}) \quad (6.3.22)$$

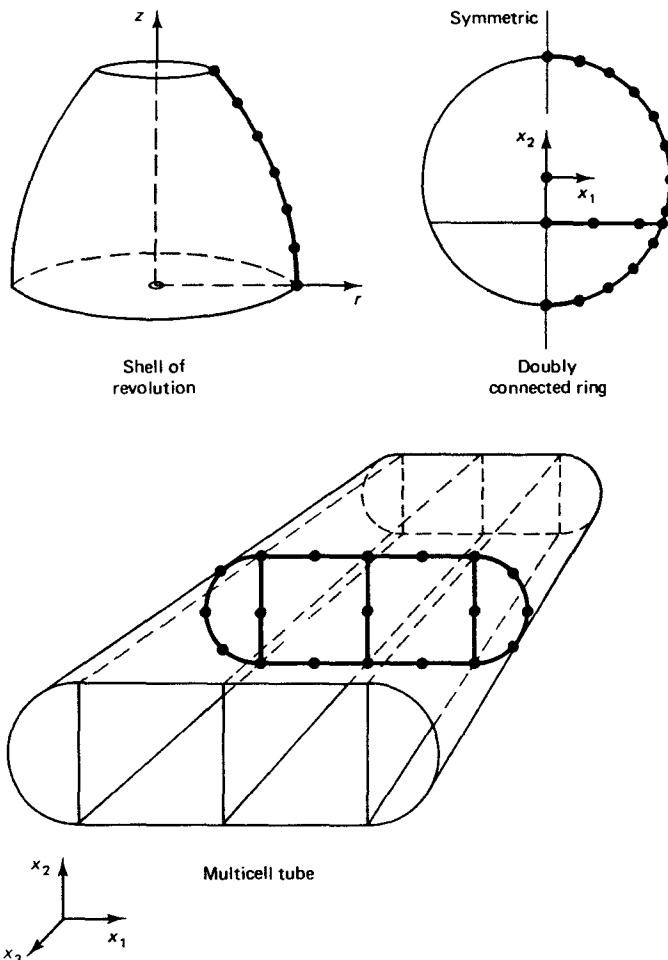


Figure 6.3.4

$$\tilde{\boldsymbol{\epsilon}}^l = \{ \tilde{\epsilon}^l \} = \left\{ \begin{array}{c} \frac{\partial u_1^l}{\partial x_1^l} \\ \frac{\partial u_1^l}{\partial x_2^l} + \frac{\partial u_2^l}{\partial x_1^l} \\ \vdots \\ 0 \end{array} \right\} \quad (\text{plane strain and plane stress}) \quad (6.3.23)$$

where $x_1 = r$ and $x_2 = z$ in axisymmetric analysis. It is our aim to derive an explicit representation of the matrix \mathbf{B}_a , where

$$\tilde{\boldsymbol{\epsilon}}^l = \sum_{a=1}^{n_{en}} \mathbf{B}_a \begin{Bmatrix} \bar{u}_{a1} \\ \bar{u}_{a2} \\ \theta_a \end{Bmatrix} \quad (6.3.24)$$

$$\mathbf{B}^a = \begin{bmatrix} b_{11}^u & b_{12}^u & b_1^\theta \\ b_{21}^u & b_{22}^u & b_2^\theta \\ b_{31}^u & 0 & b_3^\theta \end{bmatrix} \quad (\text{axisymmetric}) \quad (6.3.25)$$

$$= \begin{bmatrix} b_{11}^u & b_{12}^u & b_1^\theta \\ b_{21}^u & b_{22}^u & b_2^\theta \\ 0 & 0 & 0 \end{bmatrix} \quad (\text{plane strain and plane stress}) \quad (6.3.26)$$

To derive formulas for the b 's we need first to form the displacement gradients:

$$\begin{aligned} \frac{\partial u_i^f}{\partial x_j^f} &= \sum_{m=1}^2 q_{im} \frac{\partial u_m}{\partial x_j^f} \\ &= \sum_{m=1}^2 q_{im} \sum_{a=1}^{n_{en}} \left(\frac{\partial N_a}{\partial x_j^f} \bar{u}_{am} - \frac{\partial (N_a z_a)}{\partial x_j^f} \right) \theta_a e_{am1}^f \end{aligned} \quad (6.3.27)$$

In (6.3.27) we have used the notation:

$$e_{al}^f = \{e_{am1}^f\} = \left\{ \begin{array}{l} e_{a11}^f \\ e_{a21}^f \end{array} \right\} \quad (6.3.28)$$

From (6.3.8) and (6.3.27) we may read the desired expressions:

$$b_{1m}^u = q_{1m} \frac{\partial N_a}{\partial x_1^f} \quad (6.3.29)$$

$$b_{2m}^u = q_{1m} \frac{\partial N_a}{\partial x_2^f} + q_{2m} \frac{\partial N_a}{\partial x_1^f} \quad (6.3.30)$$

$$b_{31}^u = \frac{N_a}{x_1} \quad (6.3.31)$$

$$b_1^\theta = - \left(\sum_{m=1}^2 q_{1m} e_{am1}^f \right) \frac{\partial (N_a z_a)}{\partial x_1^f} \quad (6.3.32)$$

$$b_2^\theta = - \left(\sum_{m=1}^2 q_{1m} e_{am1}^f \right) \frac{\partial (N_a z_a)}{\partial x_2^f} - \left(\sum_{m=1}^2 q_{2m} e_{am1}^f \right) \frac{\partial (N_a z_a)}{\partial x_1^f} \quad (6.3.33)$$

$$b_3^\theta = - \frac{N_a z_a e_{a11}^f}{x_1} \quad (6.3.34)$$

In (6.3.31) and (6.3.34), x_1 is given by (6.3.1). The differentiations indicated in (6.3.29), (6.3.30), (6.3.32), and (6.3.33) are performed in global coordinates and then transformed:

$$\frac{\partial}{\partial x_j^f} = \sum_{n=1}^2 q_{jn} \frac{\partial}{\partial x_n} \quad (6.3.35)$$

6.3.4 Stiffness Matrix

The element stiffness matrix is defined by

$$\underbrace{\mathbf{k}}_{3n_{en} \times 3n_{en}} = [\mathbf{k}_{ab}] \quad (6.3.36)$$

$$\underbrace{\mathbf{k}_{ab}}_{3 \times 3} = \int_{-1}^{+1} \underbrace{\int_{-1}^{+1} \mathbf{B}_a^T \tilde{\mathbf{D}} \mathbf{B}_b j d\zeta d\eta}_{\text{fiber integral}} \quad (6.3.37)$$

lamina integral

where

$$j = \begin{cases} \bar{j} & (\text{plane strain and plane stress}) \\ x_1 \bar{j} & (\text{axisymmetric}) \end{cases} \quad (6.3.38)$$

$$\bar{j} = \det \begin{bmatrix} x_{1,\eta} & x_{1,\zeta} \\ x_{2,\eta} & x_{2,\zeta} \end{bmatrix} \quad (6.3.39)$$

6.3.5 External Force Vector

Body force

The element body force vector is given by

$$\underbrace{\mathbf{f}^{\text{body}}}_{3n_{en} \times 1} = \{\mathbf{f}_a^{\text{body}}\} \quad (6.3.40)$$

$$\underbrace{\mathbf{f}_a^{\text{body}}}_{3 \times 1} = \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{N}^T \mathbf{f} j d\zeta d\eta \quad (6.3.41)$$

where j is defined by (6.3.38) and (6.3.39), and

$$\mathbf{N}_a = \begin{bmatrix} N_a & 0 & -N_a z_a e_{a11}^f \\ 0 & N_a & -N_a z_a e_{a21}^f \end{bmatrix} \quad (6.3.42)$$

Surface force

The element surface force vector is given by

$$\mathbf{f}_a^{\text{surf}} = \{\mathbf{f}_a^{\text{surf}}\} \quad (6.3.43)$$

$$\mathbf{f}_a^{\text{surf}} = \int_{-1}^{+1} \mathbf{N}_a^T \mathbf{h} j_s d\eta, \quad \zeta = \begin{cases} +1 & \text{top} \\ -1 & \text{bottom} \end{cases} \quad (6.3.44)$$

where

$$\mathbf{j}_s = \begin{cases} \bar{\mathbf{j}}_s & \text{(plane stress and plane strain)} \\ x_1 \bar{\mathbf{j}}_s & \text{(axisymmetric)} \end{cases} \quad (6.3.45)$$

$$\bar{\mathbf{j}}_s = \| \mathbf{x}_{,\eta} \| \quad (6.3.46)$$

and \mathbf{h} is the surface force vector (per unit surface area), which includes both normal pressure and tangential shear as special cases, viz.,

Pressure

$$\mathbf{h} = -\zeta p e_2^l, \quad (\zeta = +1 \text{ or } -1) \quad (6.3.47)$$

In (6.3.47), p is the pressure.

Shear

$$\mathbf{h} = h_\eta e_1^l \quad (6.3.48)$$

In (6.3.48), h_η is the tangential shear in the η direction.

6.3.6 Stress Resultants

The stress resultants are given as follows.

Moment

$$m_{11}(\eta) = \int_{-1}^{+1} \sigma_{11}^l(\eta, \zeta) z(\eta, \zeta) d\zeta z_{,\zeta}(\eta) \quad (6.3.49)$$

$$z(\eta, \zeta) = N_+(\zeta) z^+(\eta) + N_-(\zeta) z^-(\eta) \quad (6.3.50)$$

$$z^\pm(\eta) = \sum_{a=1}^{n_m} N_a(\eta) z_a^\pm \quad (6.3.51)$$

$$z_{,\zeta}(\eta) = \frac{z^+(\eta) - z^-(\eta)}{2} \quad (6.3.52)$$

Membrane forces

$$n_{ii}(\eta) = \int_{-1}^{+1} \sigma_{ii}^l(\eta, \zeta) d\zeta z_{,\zeta}(\eta), \quad ii = 11 \text{ or } 33 \quad (\text{no sum}) \quad (6.3.53)$$

Shear

$$q(\eta) = \int_{-1}^{+1} \sigma_{12}^l(\eta, \zeta) d\zeta z_{,\zeta}(\eta) \quad (6.3.54)$$

The fiber integrations may be calculated by a variety of quadrature rules. A discussion of various possibilities is presented in Sec. 6.2.9.

The sign conventions for the stress resultants are illustrated in Fig. 6.3.5.

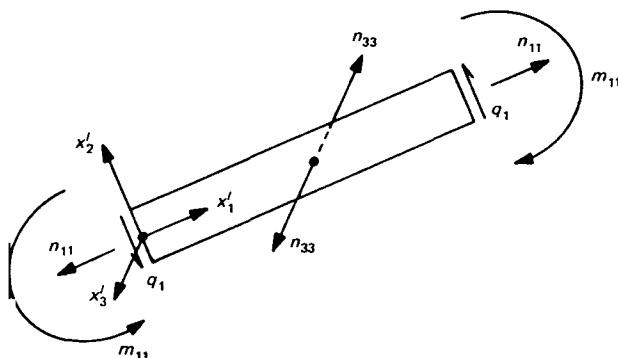


Figure 6.3.5 Sign conventions for stress resultants.

6.3.7 Boundary Conditions

The boundary conditions are essentially the same as for the three-dimensional case. That is, the kinematic conditions are the displacements and rotation of the node, and the mechanical conditions are the edge forces and moment. These latter quantities are specified per unit out-of-plane thickness in the plane strain and plane stress cases, and per unit radian in the axisymmetric case.

6.3.8 Shell Elements

The shell elements advocated herein employ one-dimensional Lagrangian interpolation in the η direction and ***uniform reduced quadrature*** (see Fig. 6.3.6). It is essential to use reduced integration to avoid shear and membrane-bending “locking” phenomena. *Rank deficiency problems*, which afflict many three-dimensional shell

| | | | |
|-----------------|-----------|-----------|-------------|
| | | | |
| Shape functions | Linear | Quadratic | Cubic |
| Quadrature rule | One-point | Two-point | Three-point |

Figure 6.3.6 Uniform reduced integration two-dimensional shell elements.

elements derived along similar lines, are not present at all in the two-dimensional uniform reduced integration Lagrange elements.

The two-node element was first suggested in [31] and developed subsequently for linear applications in [32] and [33]. In [33] a number of problems were solved, and the accuracy and economy of the element were demonstrated. The three-node element seems to have been employed by several investigators and its origin is uncertain.

A possible improvement in the two-node element in axisymmetric applications might be made by using a "mean value" of the matrix B_a (with respect to integration) rather than the value at $\eta = 0$. This is in keeping with the approach originally presented for incompressibility by Nagtegaal et al. [34] and discussed more recently in [35] (see also Chapter 4).

REFERENCES

Section 6.1

1. T. J. R. Hughes and W. K. Liu, "Nonlinear Finite Element Analysis of Shells: Part I. Three-dimensional Shells," *Computer Methods in Applied Mechanics and Engineering*, 26 (1981), 331–362.
2. T. J. R. Hughes and W. K. Liu, "Nonlinear Finite Element Analysis of Shells: Part II. Two-dimensional Shells," *Computer Methods in Applied Mechanics and Engineering*, 27 (1981), 167–181.
3. T. J. R. Hughes and E. Carnoy, "Nonlinear Finite Element Shell Formulation Accounting for Large Membrane Strains," *Computer Methods in Applied Mechanics and Engineering*, 39 (1983), 69–82.
4. S. Ahmad, B. M. Irons, and O. C. Zienkiewicz, "Analysis of Thick and Thin Shell Structures by Curved Finite Elements," *International Journal for Numerical Methods in Engineering*, 2 (1970), 419–451.

Section 6.2

5. T. J. R. Hughes and M. Cohen, "The 'Heterosis' Finite Element for Plate Bending," *Computers and Structures*, 9 (1978), 445–450.
6. T. J. R. Hughes and M. Cohen, "The 'Heterosis' Family of Plate Finite Elements," *Proceedings of the ASCE Electronic Computations Conference*, St. Louis, Missouri, August 6–8, 1979.
7. T. J. R. Hughes, M. Cohen, and M. Haroun, "Reduced and Selective Integration Techniques in the Finite Element Analysis of Plates," *Nuclear Engineering and Design*, 46, no. 1 (1978), 203–222.
8. T. J. R. Hughes, R. L. Taylor, and W. Kanoknukulchai, "A Simple and Efficient Element for Plate Bending," *International Journal for Numerical Methods in Engineering*, 11, no. 10 (1977), 1529–1543.
9. D. S. Malkus and T. J. R. Hughes, "Mixed Finite Element Methods—Reduced and Selective Integration Techniques: A Unification of Concepts," *Computer Methods in Applied Mechanics and Engineering*, 15, no. 1 (1978), 63–81.
10. E. Onate, E. Hinton, and N. Glover, *Techniques for Improving the Performance of Ahmad*

- Shell Elements*, C/R/313/78, Department of Civil Engineering, University of Wales, Swansea, U.K., 1978.
11. E. D. L. Pugh, E. Hinton, and O. C. Zienkiewicz, "A Study of Quadrilateral Plate Bending Elements with 'Reduced' Integration," *International Journal for Numerical Methods in Engineering*, 12, no. 7 (1978), 1059–1079.
 12. O. C. Zienkiewicz, R. L. Taylor, and J. M. Too, "Reduced Integration Technique in General Analysis of Plates and Shells," *International Journal for Numerical Methods in Engineering*, 3 (1971), 275–290.
 13. H. Stolarski and T. Belytschko, "Shear and Membrane Locking in Curved C⁰ Elements," *Computer Methods in Applied Mechanics and Engineering*, 41 (1983), 279–296.
 14. T. J. R. Hughes, "Recent Developments in Computer Methods for Structural Analysis," *Nuclear Engineering and Design*, 57, no. 2 (1980), 427–439.
 15. T. J. R. Hughes, "Generalization of Selective Integration Procedures to Anisotropic and Nonlinear Media," *International Journal for Numerical Methods in Engineering*, 15, no. 9 (1980), 1413–1418.
 16. R. H. MacNeal, "A Simple Quadrilateral Shell Element," *Computers and Structures*, 8 (1978), 175–183.
 17. G. M. Stanley, "Continuum-based Shell Elements," Ph.D. Thesis, Division of Applied Mechanics, Stanford University, August, 1985.
 18. T. J. R. Hughes, W. K. Liu, and I. Levit, "Nonlinear Dynamic Finite Element Analysis of Shells," in *Nonlinear Finite Element Analysis in Structural Mechanics*, eds. W. Wunderlich et al., Berlin: Springer-Verlag, 1981, 151–168.
 19. K. C. Park and G. M. Stanley, "A Curved C⁰ Shell Element based on Assumed Natural-Coordinate Strains", *Journal of Applied Mechanics*, 53 (1986), 278–290.
 20. E. N. Dvorkin and K. J. Bathe, "A Continuum Mechanics based Four-node Shell Element for General Nonlinear Analysis," *Engineering Computations*, 1 (1984), 77–88.
 21. K. C. Park, G. M. Stanley, and D. L. Flaggs, "A Uniformly Reduced Four-noded C⁰ Shell Element with Consistent Rank Corrections," *Computers and Structures*, 20 (1985), 129–139.
 22. G. M. Stanley, K. C. Park, and T. J. R. Hughes, "Continuum-Based Resultant Shell Elements," in *Finite Element Methods for Plate and Shell Structures 1: Element Technology*, eds. T. J. R. Hughes and E. Hinton. Swansea, U.K.: Pineridge Press, 1986.
 23. T. Belytschko, W. K. Liu, J. S-J. Ong, and D. Lam, "Implementation and Application of a 9-node Lagrange Shell Element with Spurious Mode Control," *Computers and Structures*, 20 (1985), 121–128.
 24. T. Belytschko, H. Stolarski, W. K. Liu, N. Carpenter, and J. S-J. Ong, "Stress Projection for Membrane and Shear Locking in Shell Finite Elements," *Computer Methods in Applied Mechanics and Engineering*, 51 (1985), 221–258.
 25. E. C. Huang and E. Hinton, "Elastic-Plastic and Geometrically Nonlinear Analysis of Plates and Shells Using a New Nine-Node Element," *Finite Elements for Nonlinear Problems*, eds. P. Bergan et al., Berlin: Springer-Verlag, 1986, 283–297.
 26. J. O. Hallquist, D. J. Benson, and G. L. Goudreau, "Implementation of a Modified Hughes-Liu Shell into a Fully Vectorized Explicit Finite Element Code," *Finite Elements for Nonlinear Problems*, eds. P. Bergan et al., Berlin: Springer-Verlag, 1986, 465–479.
 27. T. J. R. Hughes and E. Hinton, eds., *Finite Element Methods for Plate and Shell Structures 1: Element Technology and Finite Element Methods for Plate and Shell Structures 2: Formulations and Algorithms*. Swansea, U.K.: Pineridge Press, 1986.

28. W. Kanoknukulchai, "A Simple and Efficient Finite Element for General Shell Analysis," *International Journal for Numerical Methods in Engineering*, 14 (1979), 179–200.
29. P. G. Bergan and C. A. Felippa, "A Triangular Membrane Element with Rotational Degrees of Freedom," *Computer Methods in Applied Mechanics and Engineering*, 50 (1985), 25–69.
30. R. L. Taylor and J. C. Simo, "Bending and Membrane Elements for Analysis of Thick and Thin Shells," pp. 587–591, *Proceedings of the NUMETA '85 Conference* (eds. J. Midleton and G. N. Pande) Swansea, 7–11 January 1985. Published by A. A. Balkema, Rotterdam.

Section 6.3

31. T. J. R. Hughes, R. L. Taylor, and W. Kanoknukulchai, "A Simple and Efficient Element for Plate Bending," *International Journal for Numerical Methods in Engineering*, 11, no. 10 (1977), 1529–1543.
32. G. L. Goudreau, *A Computer Module for One-Step Dynamic Response of an Axisymmetric Plane Linear Elastic Thin Shell*, Lawrence Livermore Laboratory Report No. UCID-17730, February 1978.
33. O. C. Zienkiewicz, J. Bauer, K. Morgan, and E. Oñate, "A Simple Element for Axisymmetric Shells with Shear Deformation," *International Journal for Numerical Methods in Engineering*, 11, no. 10 (1977), 1545–1558.
34. J. C. Nagtegaal, D. M. Parks, and J. R. Rice, "On Numerically Accurate Finite Element Solutions in the Fully Plastic Range," *Computer Methods in Applied Mechanics and Engineering*, 4 (1974) 153–178.
35. J. C. Nagtegaal and J. E. de Jong, "Some Computational Aspects of Elastic-Plastic Large Strain Analysis," *International Journal for Numerical Methods in Engineering*, 17 (1981) 15–41.

7

Formulation of Parabolic, Hyperbolic, and Elliptic-Eigenvalue Problems

In order to understand this chapter, a detailed understanding of the material and notational conventions of Chapter 2 is required.

7.1. PARABOLIC CASE: HEAT EQUATION

The heat equation is the prototypical “parabolic” partial differential equation of mathematical physics (e.g., see Stakgold [1]). The formulation given here generalizes the formulation of Chapter 2, which was restricted to steady heat conduction, to time-dependent heat conduction processes. We view all the data to be functions of time, denoted by t , as well as space. For example, the volumetric heat source f is written as

$$f: \Omega \times]0, T[\rightarrow \mathbb{R} \quad (7.1.1)$$

which means f is a function of $x \in \Omega$ and $t \in]0, T[$, the open time interval of length $T > 0$. Likewise, we write $\varrho = \varrho(x, t)$. Similarly, the boundary data is also time-dependent:

$$\varrho: \Gamma_\varrho \times]0, T[\rightarrow \mathbb{R} \quad (7.1.2)$$

$$h: \Gamma_h \times]0, T[\rightarrow \mathbb{R} \quad (7.1.3)$$

Note that Γ_ϱ and Γ_h are assumed *not* to vary with time. To render the initial/boundary-value problem well posed, an *initial condition* on temperature must also be specified. We denote the initial temperature by

$$u_0 : \Omega \rightarrow \mathbb{R} \quad (7.1.4)$$

Two other material properties come into play: The *density*, ρ , and *capacity*, c ; both are assumed positive functions of $x \in \Omega$.

The *strong form* of the initial/boundary-value problem may now be stated:

$$(S) \left\{ \begin{array}{l} \text{Given } f, g, h, \text{ and } u_0, \text{ as in (7.1.1) through (7.1.4), find } u : \bar{\Omega} \times [0, T] \rightarrow \mathbb{R} \\ \text{such that} \\ \rho c u_{,t} + q_{i,i} = f \quad \text{on } \Omega \times]0, T[\quad (\text{heat equation}) \quad (7.1.5) \\ u = g \quad \text{on } \Gamma_g \times]0, T[\quad (7.1.6) \\ -q_i n_i = h \quad \text{on } \Gamma_h \times]0, T[\quad (7.1.7) \\ u(x, 0) = u_0(x) \quad x \in \Omega \quad (7.1.8) \end{array} \right.$$

Recall from Chapter 2 that

$$q_i = -\kappa_{ij} u_{,j} \quad (7.1.9)$$

where $\kappa_{ij} = \kappa_{ij}(x)$ denotes the conductivity tensor. The notation $u_{,t}$ denotes the time derivative of u (i.e., $u_{,t} = \partial u / \partial t$).

Let \mathcal{V} denote the usual space of weighting functions satisfying zero-temperature boundary conditions. Note that functions in \mathcal{V} do *not* depend on time in any way. Let $\mathcal{S} = \mathcal{S}_t$ denote the space of trial solutions. Note that \mathcal{S} varies as a function of t due to the temperature boundary condition, (7.1.2),

$$\mathcal{S} = \mathcal{S}_t = \left\{ u(\cdot, t) \mid u(x, t) = g(x, t), x \in \Gamma_g, u(\cdot, t) \in H^1(\Omega) \right\} \quad (7.1.10)$$

The *weak formulation* consists of the following:

$$(W) \left\{ \begin{array}{l} \text{Given } f, g, h, \text{ and } u_0, \text{ find } u(t) \in \mathcal{S}_t, t \in [0, T] \text{ such that for all } w \in \mathcal{V}, \\ (w, \rho c \dot{u}) + a(w, u) = (w, f) + (w, h)_\Gamma \quad (7.1.11) \\ (w, \rho c u(0)) = (w, \rho c u_0) \quad (7.1.12) \end{array} \right.$$

Remark

In the weak formulation, x is suppressed as an argument of u . Thus $u(0)$ represents the function u of x at time 0 [i.e., $u(0) = u(\cdot, 0)$]. The superposed dot is used to denote time differentiation. These notations, among other things, enable us to remove commas from the weak formulation which might be confused with those normally appearing in the bilinear forms. Equation (7.1.11) is the weak formulation of the heat equation (7.1.5) and heat-conduction boundary condition (7.1.7), and (7.1.12) is the weak form of the initial condition (7.1.8).

Exercise 1. Verify the formal equivalence of (S) and (W) by proceeding along the lines of Sec. 2.3. The procedure is virtually identical to Sec. 2.3 if (7.1.11) is written as

$$a(w, u) = (w, \tilde{\ell}) + (w, h)_\Gamma$$

where

$$\tilde{\ell} = \ell - \rho c \dot{u}$$

To develop the analogous Galerkin formulation, we proceed as in Sec. 2.4. The approximate weighting function space, \mathcal{V}^h , is identical to before. Functions $u^h \in \mathcal{S}^h$ are built up in the usual way:

$$u^h = v^h + q^h \quad (7.1.13)$$

where $v^h(t) \in \mathcal{V}^h$ (i.e., for each fixed t , $v^h(\cdot, t)$ is a function in \mathcal{V}^h) and q^h enables satisfaction of the temperature boundary condition. Note that $q^h(t) \in \mathcal{S}_t^h$. All the functions in (7.1.13) depend upon both space and time, i.e.,

$$u^h(x, t) = v^h(x, t) + q^h(x, t) \quad (7.1.14)$$

The *Galerkin formulation* is stated as follows:

$$(G) \left\{ \begin{array}{l} \text{Given } \ell, q, h, \text{ and } u_0, \text{ find } u^h = v^h + q^h, u^h(t) \in \mathcal{S}_t^h, \text{ such that for all } \\ w^h \in \mathcal{V}^h, \\ (w^h, \rho c \dot{v}^h) + a(w^h, v^h) = (w^h, \ell) + (w^h, h)_\Gamma - (w^h, \rho c \dot{q}^h) - a(w^h, q^h) \\ (w^h, \rho c v^h(0)) = (w^h, \rho c u_0) - (w^h, \rho c q^h(0)) \end{array} \right. \quad (7.1.15)$$

$$(7.1.16)$$

Note that all given quantities appear on the right-hand sides of (7.1.15) and (7.1.16). The Galerkin equation given by (7.1.15) is called a *semidiscrete* equation because time is left continuous.

The matrix equations require representations of v^h and q^h in terms of basis functions. These are given by

$$v^h(x, t) = \sum_{A \in n-n_q} N_A(x) d_A(t) \quad (7.1.17)$$

$$q^h(x, t) = \sum_{A \in n_q} N_A(x) q_A(t) \quad (7.1.18)$$

Note that the entire time-dependence is carried by the nodal values (i.e., the d_A 's and q_A 's). The shape functions are identical to those used previously (see Chapters 2 and 3). In particular, they are *not* time-dependent.

Remark

Upon encountering representations such as (7.1.17) and (7.1.18) for the first time, we may wonder if the approximation is valid only when the exact solution is "separable" in x and t (e.g., see [1] for a discussion of the separation-of-variables

technique). This is not the case. Even nonseparable exact solutions may be approximated arbitrarily closely by representations like (7.1.17) and (7.1.18).

To develop the matrix equations, we substitute (7.1.17) and (7.1.18) into (7.1.15) and (7.1.16) and proceed along the same lines as in Secs. 2.4 and 2.5. The end result is the following *matrix problem*:

Given $F :]0, T[\rightarrow \mathbb{R}^{n_{eq}}$, find $d : [0, T] \rightarrow \mathbb{R}^{n_{eq}}$ such that

$$Md + Kd = F \quad t \in]0, T[\quad (7.1.19)$$

$$d(0) = d_0 \quad (7.1.20)$$

where

$$M = \sum_{\epsilon=1}^{n_{el}} \mathbf{A}_\epsilon (m^\epsilon) \quad (7.1.21)$$

$$m = [m_{ab}^\epsilon] \quad (7.1.22)$$

$$m_{ab}^\epsilon = \int_{\Omega^\epsilon} N_a \rho c N_b \, d\Omega \quad (7.1.23)$$

$$K = \sum_{\epsilon=1}^{n_{el}} \mathbf{A}_\epsilon (k^\epsilon) \quad (7.1.24)$$

$$k^\epsilon = [k_{ab}^\epsilon] \quad (7.1.25)$$

$$k_{ab}^\epsilon = \int_{\Omega^\epsilon} \mathbf{B}_a^T D \mathbf{B}_b \, d\Omega \quad (7.1.26)$$

$$F(t) = F_{\text{nodal}}(t) + \sum_{\epsilon=1}^{n_{el}} \mathbf{A}_\epsilon (f^\epsilon(t)) \quad (7.1.27)$$

$$f^\epsilon = \{f_a^\epsilon\} \quad (7.1.28)$$

$$f_a^\epsilon = \int_{\Omega^\epsilon} N_a \dot{\ell} \, d\Omega + \int_{\Gamma_k^\epsilon} N_a \dot{h} \, d\Gamma - \sum_{b=1}^{n_{eq}} (k_{ab}^\epsilon q_b^\epsilon + m_{ab}^\epsilon \dot{q}_b^\epsilon) \quad (7.1.29)$$

$$d_0 = M^{-1} \sum_{\epsilon=1}^{n_{el}} \mathbf{A}_\epsilon (\hat{d}^\epsilon) \quad (7.1.30)$$

$$\hat{d}^\epsilon = \{\hat{d}_a^\epsilon\} \quad (7.1.31)$$

$$\hat{d}_a^\epsilon = \int_{\Omega^\epsilon} N_a \rho c u_0 \, d\Omega - \sum_{b=1}^{n_{eq}} m_{ab}^\epsilon q_b^\epsilon(0) \quad (7.1.32)$$

Remarks

1. If we compare with the steady formulation of Chapter 2, we see that the only new matrix which appears is M , the *capacity matrix*. That M is symmetric and positive-definite follows directly from its definition.

2. Equation (7.1.19) is a coupled system of ordinary differential equations. Thus to solve the initial-value problem, i.e., (7.1.19) and (7.1.20), we need to introduce algorithms for solving systems of ordinary differential equations. This subject is taken up in subsequent chapters.

3. Observe in (7.1.29) that the element capacity matrices come into play in accounting for the effect of specified boundary temperatures.

4. It is common in practice to simplify the specification of initial conditions. That is, rather than employ (7.1.30) through (7.1.32), which emanate from (7.1.16), we directly specify d_0 such that the given nodal values are interpolated [i.e., $d_{0A} = u_0(x_A)$, $A \in \eta - \eta_g$].

5. The element capacity matrix m^e turns out to be virtually identical to the element "mass" matrix, which will be introduced in the next section. Consequently, we shall postpone more detailed consideration of the structure of m^e until later.

6. Recall that $g_b(t) = 0$ if degree of freedom b of element e is *not* specified. The same rule applies to \dot{g}_b .

Exercise 2. The details of arriving at (7.1.19) through (7.1.32) are straightforward given familiarity with the analogous developments in Chapter 2. If the results are not "obvious," the reader should fill in all details in step-by-step fashion.

Example

The one-dimensional heat equation is, of course, just a special case of the general formulation above. A strong form of the initial/boundary-value problem can be set which is a generalization of the one-dimensional model problem considered in Chapter 1. The equations are

$$(S) \left\{ \begin{array}{ll} \rho c u_{,t} - (\kappa u_{,x})_{,x} = f & \text{on }]0, L[\times]0, T[\\ u(L, t) = g(t) & t \in]0, T[\\ -\kappa u_{,x}(0, t) = h(t) & t \in]0, T[\\ u(x, 0) = u_0(x) & x \in]0, L[\end{array} \right. \quad \begin{array}{l} (7.1.33) \\ (7.1.34) \\ (7.1.35) \\ (7.1.36) \end{array}$$

The element arrays are

$$m_{ab}^e = \int_{\Omega^e} N_a \rho c N_b d\Omega \quad (7.1.37)$$

$$k_{ab}^e = \int_{\Omega^e} N_{a,x} \kappa N_{b,x} d\Omega \quad (7.1.38)$$

$$f_a^e = \int_{\Omega^e} N_a f d\Omega + N_a(0) \delta_{e1} h - \sum_{b=1}^{n_e} (k_{ab}^e g_b^e + m_{ab}^e \dot{g}_b^e) \quad (7.1.39)$$

7.2 HYPERBOLIC CASE: ELASTODYNAMICS AND STRUCTURAL DYNAMICS

The developments of this section generalize those of Secs. 2.7 through 2.10. The nature of the generalization is similar in format to that of the previous section and the reader should first become familiar with Sec. 7.1 before considering this section even if he or she is uninterested in heat conduction.

The initial conditions this time involve specification of both displacements and velocities. Thus

$$u_{0i} : \Omega \rightarrow \mathbb{R} \quad (7.2.1)$$

and

$$\dot{u}_{0i} : \Omega \rightarrow \mathbb{R} \quad (7.2.2)$$

are given functions for each i , $1 \leq i \leq n_{sd}$. The superposed-dot notation in (7.2.2) is symbolic rather than operational.

The remaining prescribed data are

$$\ell_i : \Omega \times]0, T[\rightarrow \mathbb{R} \quad (7.2.3)$$

$$g_i : \Gamma_{q_i} \times]0, T[\rightarrow \mathbb{R} \quad (7.2.4)$$

$$h_i : \Gamma_{h_i} \times]0, T[\rightarrow \mathbb{R} \quad (7.2.5)$$

The **density**, $\rho : \Omega \rightarrow \mathbb{R}$, assumed to be positive, needs also to be specified in the present case.

The **strong form** of the initial/boundary-value problem is

$$(S) \left\{ \begin{array}{ll} \text{Given } \ell_i, g_i, h_i, u_{0i} \text{ and } \dot{u}_{0i}, \text{ as in (7.2.1) through (7.2.5), find } u_i : \overline{\Omega} \times [0, T] \rightarrow \mathbb{R} \text{ such that} \\ \rho u_{i,tt} = \sigma_{ij,j} + \ell_i & \text{on } \Omega \times]0, T[\quad (\text{equation of motion}) \\ u_i = g_i & \text{on } \Gamma_{q_i} \times]0, T[\\ \sigma_{ij}n_j = h_i & \text{on } \Gamma_{h_i} \times]0, T[\\ u_i(x, 0) = u_{0i}(x) & x \in \Omega \\ u_{i,t}(x, 0) = \dot{u}_{0i}(x) & x \in \Omega \end{array} \right. \quad \begin{array}{l} (7.2.6) \\ (7.2.7) \\ (7.2.8) \\ (7.2.9) \\ (7.2.10) \end{array}$$

Recall that $\sigma_{ij} = c_{ijkl} u_{(k,l)}$ and $c_{ijkl} = c_{ijkl}(x)$. Note that the second time derivative (i.e., acceleration) appears in (7.2.6). This is the reason that two initial conditions are required.

The corresponding **weak formulation** is:¹

$$(W) \left\{ \begin{array}{l} \text{Given } \ell, q, h, u_0, \text{ and } \dot{u}_0, \text{ find } u(t) \in \mathcal{S}_t, t \in [0, T], \text{ such that for all } w \in \mathcal{V}, \\ \boxed{(w, \rho \ddot{u}) + a(w, u) = (w, \ell) + (w, h)_\Gamma} \quad (7.2.11) \\ (w, \rho u(0)) = (w, \rho u_0) \quad (7.2.12) \\ (w, \rho \dot{u}(0)) = (w, \dot{u}_0) \quad (7.2.13) \end{array} \right.$$

Exercise 1. Verify the formal equivalence of (S) and (W). The arguments of Chapter 2 may be used virtually unaltered if (7.2.11) is written as

$$a(w, u) = (w, \tilde{\ell}) + (w, h)_\Gamma$$

where

$$\tilde{\ell} = \ell - \rho \ddot{u}$$

The *semidiscrete Galerkin formulation of elastodynamics* is:

$$(G) \left\{ \begin{array}{l} \text{Given } \ell, q, h, u_0, \text{ and } \dot{u}_0, \text{ find } u^h = v^h + q^h, u^h(t) \in \mathcal{S}_t^h, \text{ such that for all } w^h \in \mathcal{V}^h, \\ \boxed{(w^h, \rho \ddot{v}^h) + a(w^h, v^h) = (w^h, \ell) + (w^h, h)_\Gamma - (w^h, \rho \ddot{q}^h) - a(w^h, q^h)} \quad (7.2.14) \\ (w^h, \rho v^h(0)) = (w^h, \rho u_0) - (w^h, \rho q^h(0)) \quad (7.2.15) \\ (w^h, \rho \dot{v}^h(0)) = (w^h, \rho \dot{u}_0) - (w^h, \rho \dot{q}^h(0)) \quad (7.2.16) \end{array} \right.$$

The representations of v^h and q^h are given by

$$v_i^h(x, t) = \sum_{A \in \eta - \eta_{q_i}} N_A(x) d_{iA}(t) \quad (7.2.17)$$

$$q_i^h(x, t) = \sum_{A \in \eta_{q_i}} N_A(x) q_{iA}(t) \quad (7.2.18)$$

These and the usual arguments lead to the *matrix problem*:

Given $F:]0, T[\rightarrow \mathbb{R}^{n_q}$, find $d:]0, T[\rightarrow \mathbb{R}^{n_d}$ such that

$$\boxed{M \ddot{d} + Kd = F \quad t \in]0, T[} \quad (7.2.19)$$

$$\boxed{d(0) = d_0} \quad (7.2.20)$$

$$\dot{\mathbf{d}}(0) = \dot{\mathbf{d}}_0$$

(7.2.21)

where

$$\mathbf{M} = \sum_{\epsilon=1}^{n_{el}} (\mathbf{m}^\epsilon)$$

$$\mathbf{m}^\epsilon = [m_{pq}^\epsilon]$$

(7.2.23)

$$m_{pq}^\epsilon = \delta_{ij} \int_{\Omega^\epsilon} N_a \rho N_b d\Omega$$

(7.2.24)

(Recall $p = n_{ed}(a - 1) + i$ and $q = n_{ed}(b - 1) + j$)

$$\mathbf{K} = \sum_{\epsilon=1}^{n_{el}} (\mathbf{k}^\epsilon)$$

(7.2.25)

$$\mathbf{k}^\epsilon = [k_{pq}^\epsilon]$$

(7.2.26)

$$k_{pq}^\epsilon = \mathbf{e}_i^T \int_{\Omega^\epsilon} \mathbf{B}_a^T \mathbf{D} \mathbf{B}_b d\Omega \mathbf{e}_j$$

(7.2.27)

$$(M) \quad \mathbf{F}(t) = \mathbf{F}_{\text{nodal}}(t) + \sum_{\epsilon=1}^{n_{el}} (\mathbf{f}^\epsilon(t))$$

$$\mathbf{f}^\epsilon = \{f_p^\epsilon\}$$

$$f_p^\epsilon = \int_{\Omega} N_a f_i d\Omega + \int_{\Gamma_{h_i}} N_a h_i d\Gamma - \sum_{q=1}^{n_{eq}} (k_{pq}^\epsilon q_q^\epsilon + m_{pq}^\epsilon \ddot{q}_q^\epsilon) \quad (\text{no sum on } i)$$

(7.2.28)

(7.2.29)

(7.2.30)

$$\mathbf{d}_0 = \mathbf{M}^{-1} \sum_{q=1}^{n_{el}} (\hat{\mathbf{d}}^\epsilon)$$

(7.2.31)

$$\hat{\mathbf{d}}^\epsilon = \{\hat{d}_p^\epsilon\}$$

(7.2.32)

$$\hat{d}_p^\epsilon = \int_{\Omega^\epsilon} N_a \rho u_{0i} d\Omega - \sum_{q=1}^{n_{eq}} m_{pq}^\epsilon q_q^\epsilon(0)$$

(7.2.33)

$$\dot{\mathbf{d}}_0 = \mathbf{M}^{-1} \sum_{\epsilon=1}^{n_{el}} (\hat{\mathbf{d}}^\epsilon)$$

(7.2.34)

$$\hat{\mathbf{d}}^\epsilon = \{\hat{d}_p^\epsilon\}$$

(7.2.35)

$$\hat{d}_p^\epsilon = \int_{\Omega^\epsilon} N_a \rho \dot{u}_{0i} d\Omega - \sum_{q=1}^{n_{eq}} m_{pq}^\epsilon \dot{q}_q^\epsilon(0)$$

(7.2.36)

Remarks

1. The main addition to what we have encountered previously in the elastostatics formulation of Chapter 2 is the **mass matrix**, \mathbf{M} . The reader should verify that \mathbf{M} is symmetric and positive-definite. Except for the Kronecker delta and different material parameter inside the integrand, the mass and capacity matrix [(7.1.21) through (7.1.23)] are identical. To appreciate the origin of the Kronecker delta, we will sketch part of the calculation that leads to the matrix formulation. If we restrict attention to the i th element, then

$$\begin{aligned} (\mathbf{w}^h, \mathbf{p}\ddot{\mathbf{u}}^h)^e &= \int_{\Omega^e} w_i^h \rho \ddot{u}_i^h d\Omega \\ &= \delta_{ij} \int_{\Omega^e} w_i^h \rho \ddot{u}_j^h d\Omega \\ &= \sum_{A,B} c_{iA} \delta_{ij} \int_{\Omega^e} N_A \rho N_B d\Omega \ddot{d}_{jB} \quad (\text{summation of } i \text{ and } j \text{ implied}) \end{aligned} \quad (7.2.37)$$

2. Equation (7.2.19) is a coupled system of second-order ordinary differential equations. Algorithms for solving equations of this type are described in Chapter 9.

3. Note that the element mass is involved in the adjustment of forces due to nonzero boundary accelerations (see (7.2.30)).

4. As mentioned in the previous section, we usually simplify the specification of initial conditions in practice. Nodal interpolates in this case take the form

$$d_{0P} = u_{0i}(x_A) \quad (7.2.38)$$

$$\dot{d}_{0P} = \dot{u}_{0i}(x_A) \quad (7.2.39)$$

where $P = \text{LM}(i, A)$. Recall LM is the array described in Chapter 2.

Exercise 2. Fill in the omitted details leading to the matrix formulation of elastodynamics.

Viscous Damping

In structural dynamics we often work with systems of the form

$$\boxed{\mathbf{M}\ddot{\mathbf{d}} + \mathbf{C}\dot{\mathbf{d}} + \mathbf{K}\mathbf{d} = \mathbf{F}} \quad (7.2.40)$$

where \mathbf{C} is the **viscous damping matrix**. A particularly convenient form of \mathbf{C} is the **Rayleigh damping matrix**

$$\boxed{\mathbf{C} = a\mathbf{M} + b\mathbf{K}} \quad (7.2.41)$$

parameters. The two constituents of Rayleigh damping are seen to

be mass and stiffness proportional. We would like to enlarge our theoretical framework to include Rayleigh damping. The necessary modifications are as follows: Replace the equation of motion, (7.2.6), by

$$\rho u_{,tt} + a \rho u_{,t} = \sigma_{ij,j} + f_i \quad (7.2.42)$$

where the generalized Hooke's law is modified to account for the stiffness proportional effect, namely,

$$\sigma_{ij} = c_{ijkl}(u_{(k,l)} + b \dot{u}_{(k,l)}) \quad (7.2.43)$$

In addition to the appearance of the $C\dot{d}$ -term in (7.2.40), the effect of the viscous damping matrix is also felt in modifying the forces due to prescribed displacement boundary conditions. Specifically,

$$f_p^\epsilon = \text{right-hand side of (7.2.30)} - \sum_{q=1}^{n_{el}} c_{pq}^\epsilon \dot{q}_q^\epsilon \quad (7.2.44)$$

where

$$c^\epsilon = am^\epsilon + bk^\epsilon \quad (7.2.45)$$

Everything else remains the same. The parameters a and b may be selected to produce desired damping characteristics.

Example

In one dimension, the above formulation leads to a *wave equation*. This also may be viewed as a generalization of the one-dimensional model problem of Chapter 1. Various interpretations are possible. For example, the axial motion of an elastic rod of length L is governed by the equation

$$\rho u_{,tt} = \sigma_{,x} + f \quad \text{on }]0, L[\times]0, T[\quad (7.2.46)$$

where

$$\sigma = E u_{,x} \quad (7.2.47)$$

and $E = E(x)$ is Young's modulus. Boundary and initial conditions may be specified in analogous fashion to Chapter 1, namely,

$$u(L, t) = q(t) \quad t \in]0, T[\quad (7.2.48)$$

$$-Eu_{,x}(0, t) = h(t) \quad t \in]0, T[\quad (7.2.49)$$

$$u(x, 0) = u_0(x) \quad x \in]0, L[\quad (7.2.50)$$

$$\dot{u}(x, 0) = \dot{u}_0(x) \quad x \in]0, L[\quad (7.2.51)$$

The resulting element arrays are virtually identical to the ones encountered in the one-dimensional heat equation example at the end of Sec. 7.1, viz.,

$$m_{ab}^\epsilon = \int_{\Omega'} N_a \rho N_b \, d\Omega \quad (7.2.52)$$

$$k_{ab}^e = \int_{\Omega^e} N_{a,x} E N_{b,x} d\Omega \quad (7.2.53)$$

$$f_a^e = \int_{\Omega^e} N_a f d\Omega - N_a(0) \delta_e/h - \sum_{b=1}^{n_e} (k_{ab}^e q_b + m_{ab}^e \ddot{q}_b) \quad (7.2.54)$$

Note that in the present case, element equation and node numbers coincide (i.e., $p = a$ and $q = b$). Compare (7.1.37) through (7.1.39) with (7.2.52) through (7.2.54).

Exercise 3. In previous chapters, the element stiffness matrix, (7.2.53), was evaluated (without E) for typical C^0 shape functions. Assuming ρ is constant, evaluate the element mass matrix, m^e , for the following shape functions:

- i. Piecewise linears.

$$\left[\text{Answer: } m^e = \frac{\rho h}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}. \right]$$

- ii. Piecewise quadratics.

$$\left[\text{Answer: } m^e = \frac{\rho h}{30} \begin{bmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{bmatrix}. \right]$$

Exercise 4. The initial/boundary-value problem for the deflection of an *elastic membrane on a Winkler foundation* is stated as follows:

Given $f : \Omega \times]0, T[\rightarrow \mathbb{R}$, $g : \Gamma_g \times]0, T[\rightarrow \mathbb{R}$, $h : \Gamma_h \times]0, T[\rightarrow \mathbb{R}$, $u_0 : \Omega \rightarrow \mathbb{R}$, and $\dot{u}_0 : \Omega \rightarrow \mathbb{R}$, find $u : \overline{\Omega} \times [0, T] \rightarrow \mathbb{R}$ such that

$$u_{,tt} - \alpha u_{,n} + f = u_{,tt} \quad \text{on } \Omega \times]0, T[$$

$$u = g \quad x \in \Gamma_g, t \in]0, T[$$

$$u_{,n} = h \quad x \in \Gamma_h, t \in]0, T[$$

$$u = u_0 \quad x \in \Omega, t = 0$$

$$u_{,t} = \dot{u}_0 \quad x \in \Omega, t = 0$$

where $u_{,n} = \partial u / \partial n$ is the derivative in the direction of the outward unit normal vector and $\alpha > 0$.

Set up the following finite element paraphernalia:

- A semidiscrete weak form of the problem in which the h -boundary condition is "natural."
- The corresponding Galerkin form of the problem.
- The matrix ordinary-differential-equation problem.

Solution

- a. Find $u \in \mathcal{S}$, such that for all $w \in \mathcal{V}$

$$(w, \ddot{u}) + a(w, u) = (w, f) + (w, h)_\Gamma$$

$$(w, u(0) - u_0) = 0$$

$$(w, \dot{u}(0) - \dot{u}_0) = 0$$

where

$$(w, \ddot{u}) = \int_{\Omega} w \ddot{u} \, d\Omega$$

$$a(w, u) = \int_{\Omega} (w_{,i} u_{,i} + \alpha w u) \, d\Omega$$

$$(w, f) = \int_{\Omega} w f \, d\Omega$$

$$(w, h)_{\Gamma} = \int_{\Gamma_h} w h \, d\Gamma$$

⋮

- b. Find $u^h = v^h + q^h \in \mathcal{S}^h$, $v^h \in \mathcal{V}^h$ such that for all $w^h \in \mathcal{V}^h$ (as usual)

$$(w^h, \ddot{u}^h) + a(w^h, u^h) = (w^h, f) + (w, h)_{\Gamma}$$

$$(w^h, \dot{v}^h) + a(w^h, v^h) = (w^h, f) + (w^h, h)_{\Gamma} - (w^h, \dot{q}^h) - a(w^h, q^h)$$

(Likewise for initial conditions.)

- c. $M\ddot{d} + Kd = F$; $d(0) = d_0$; $\dot{d}(0) = \dot{d}_0$

where

$$m_{ab}^{\epsilon} = \int_{\Omega^{\epsilon}} N_a N_b \, d\Omega$$

$$k_{ab}^{\epsilon} = \int_{\Omega^{\epsilon}} (N_{a,i} N_{b,i} + \alpha N_a N_b) \, d\Omega$$

$$f_a^{\epsilon} = \int_{\Omega^{\epsilon}} N_a f \, d\Omega + \int_{\Gamma_h^{\epsilon}} N_a h \, d\Gamma - \sum_{b=1}^{n_{en}} (m_{ab}^{\epsilon} \ddot{q}_b^{\epsilon} + k_{ab}^{\epsilon} q_b^{\epsilon})$$

(Usual assembly algorithm plus initial conditions.)

7.3 EIGENVALUE PROBLEMS: FREQUENCY ANALYSIS AND BUCKLING

To each of the classes of time-dependent problems considered in Sec. 7.1 and 7.2 there are corresponding eigenvalue problems. Eigenvalue problems play particularly important roles in the analysis of elastic solids and structures. The natural frequencies and mode shapes of free vibration are determined by eigenvalue problems as are buckling loads and modes. Some examples illustrating potential applications are as follows.

Free Vibration of an Elastic Rod

The frequencies and mode shapes are governed by the following equation:

$$\lambda \rho u + (E u_{,x})_{,x} = 0 \quad x \in]0, L[\quad (7.3.1)$$

Various homogeneous boundary conditions may be employed. For example, corresponding to our "standard" one-dimensional case we have

$$u(L) = 0 \quad (\text{fixed}) \quad (7.3.2)$$

$$-E u_{,x}(0) = 0 \quad (\text{free}) \quad (7.3.3)$$

The nontrivial solutions of (7.3.1) through (7.3.3) are countably infinite. That is, for each $k = 1, 2, \dots, \infty$, there is an eigenvalue λ_k and corresponding eigenfunction $u_{(k)} :]0, L[\rightarrow \mathbb{R}$ which satisfy (7.3.1) through (7.3.3). It is a simple exercise to show that each λ_k is positive, $0 < \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \dots$, and that the eigenfunctions are orthogonal in the sense that

$$\int_0^L u_{(k)} \rho u_{(l)} dx = \delta_{kl} \quad (7.3.4)$$

The eigenvalue $\lambda_k = \omega_k^2$, where ω_k is the k th natural frequency and $u_{(k)}$ is the corresponding normal mode shape. The eigenfunctions are determined only up to a multiplicative constant. The arbitrariness may be removed by the normalization

$$\int_0^L \rho u^2 dx = 1 \quad \text{or} \quad \max_{x \in [0, L]} |u(x)| = 1.$$

A *weak formulation* of this problem is given by the following:

$$(W) \left\{ \begin{array}{l} \text{Find all eigenpairs } \{\lambda, u\}, u \in \mathcal{S}(\mathcal{V}), \text{ such that for all } w \in \mathcal{V} \\ \boxed{a(w, u) - \lambda(w, \rho u) = 0} \\ \text{where} \\ a(w, u) = \int_0^L w_{,x} E u_{,x} dx \\ (w, \rho u) = \int_0^L w \rho u dx \end{array} \right. \quad (7.3.5)$$

$$(7.3.6)$$

$$(7.3.7)$$

The essential boundary condition, $u(L) = 0$, needs to be built into the definition of \mathcal{V} .

The equivalence of strong and weak forms follows from by now standard arguments. Note

$$a(w, u) - \lambda(w, \rho u) = - \int_0^L w (\lambda \rho u + (E u_{,x})_{,x}) dx - (w E u_{,x}) \Big|_{x=0} \quad (7.3.8)$$

The remaining details are left as an exercise for the reader.

The corresponding **Galerkin formulation** is as follows:

Find all $\lambda^h \in \mathbb{R}$ and $u^h \in \mathcal{S}^h (\equiv \mathcal{V}^h)$ such that for all $w^h \in \mathcal{V}^h$

$$a(w^h, u^h) - \lambda^h(w^h, pu^h) = 0 \quad (7.3.9)$$

Substitution of shape-function expansions for w^h and u^h gives rise to the **matrix eigenvalue problem**.

Find eigenvalues λ_k^h and eigenvectors ψ_k , $k = 1, 2, \dots, n_{eq}$, that satisfy

$$(K - \lambda_k^h M)\psi_k = 0 \quad (\text{no sum on } k) \quad (7.3.10)$$

The eigenvectors ψ_k contain the nodal values of $u_{(k)}^h$. That is, the A th component of ψ_k is $u_{(k)}^h(x_A)$. In terms of the matrix problem, the orthogonality condition may be expressed as [see (7.3.4)]

$$\psi_k^T M \psi_l = \delta_{kl} \quad (7.3.11)$$

The element matrices that contribute to M and K are defined by (7.2.52) and (7.2.53), respectively.

Buckling of a Thin Beam ("Elastic Instability")

Consider a thin (i.e., Bernoulli-Euler) beam subjected to simply supported boundary conditions (see Fig. 7.3.1). We assume the beam is subjected to a uniform axial compression, λ , and wish to calculate the critical value of λ (i.e., "buckling load"). The equations are

$$\lambda u_{,xx} + (EIu_{,xx})_{,xx} = 0 \quad (7.3.12)$$

$$u(0) = u(L) = 0 \quad (7.3.13)$$

$$u_{,xx}(0) = u_{,xx}(L) = 0 \quad (7.3.14)$$

There are an infinite number of eigenpairs that satisfy (7.3.12) through (7.3.14). The smallest eigenvalue, $\lambda_1 > 0$, is the critical buckling load and the corresponding eigenfunction, $u_{(1)}$, is the buckling mode. An exact analysis reveals that $\lambda_k = k^2\pi^2 EI/L^2$ and $u_{(k)} = \sin k\pi x/L$.

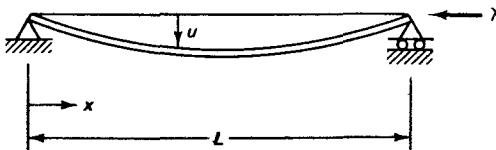


Figure 7.3.1 Simply supported beam under axial compression.

A **weak formulation** of the problem is:

Find $u \in \mathcal{S} (\equiv \mathcal{V})$ such that for all $w \in \mathcal{V}$

$$a(w, u) - \lambda b(w, u) = 0$$

(7.3.15)

(W) where

$$a(w, u) = \int_0^L w_{,xx} EI u_{,xx} dx \quad (7.3.16)$$

$$b(w, u) = \int_0^L w_{,x} u_{,x} dx \quad (7.3.17)$$

An appropriate space of functions consists of H^2 -functions (see Sec. 1.16) with (7.3.13) as essential boundary conditions. Integration by parts and $w(0) = w(L) = 0$ reveals

$$a(w, u) - \lambda b(w, u) = \int_0^L w(\lambda u_{,xx} + (EIu_{,xx})_{,xx}) dx + (w_{,x} EI u_{,xx}) \Big|_0^L \quad (7.3.18)$$

The equivalence of strong and weak forms may be argued in the usual way with the aid of this expression.

The corresponding **Galerkin formulation** is:

Find $u^h \in \mathcal{S}^h (\equiv \mathcal{V}^h)$ such that for all $w^h \in \mathcal{V}^h$,

$$a(w^h, u^h) - \lambda^h b(w^h, u^h) = 0$$

(7.3.19)

The C^1 -continuous Hermite cubics (see Sec. 1.16) serve to define \mathcal{V}^h .

The matrix problem has the same form as (7.3.10). The element contributions are defined by

$$k_{pq}^e = \int_{\Omega^e} N_{p,xx} EI N_{q,xx} dx$$

(7.3.20)

$$m_{pq}^e = \int_{\Omega^e} N_{p,x} N_{q,x} dx$$

(7.3.21)

Strictly speaking, (7.3.21) is not a mass matrix. It is usually referred to as an *initial-stress*, or *geometric, stiffness* in this context.

Exercise 1. Evaluate m^e [i.e., (7.3.21)] using Hermite cubics. (Observe that three-point Gauss quadrature enables exact integration.) Recall that the exact expression for k^e was obtained in Sec. 1.16.

Free Vibration of a Thin Beam

The vibration of a thin beam is governed by

$$-\lambda \rho A u + (EIu_{,xx})_{,xx} = 0 \quad (7.3.22)$$

where $A = A(x)$ is the cross-sectional area. Comparison with (7.3.12) reveals that in place of $\lambda u_{,xx}$, the term $-\lambda \rho A u$ appears. In the simply supported case, the weak formulation is

$$a(w, u) - \lambda(w, \rho A u) = 0 \quad (7.3.23)$$

where $a(w, u)$ is again defined by (7.3.16) and

$$(w, \rho A u) = \int_0^L w \rho A u \, dx \quad (7.3.24)$$

The matrix problem takes the form (7.3.10) and the element arrays are defined by (7.3.20) and

$$m_{pq}^e = \int_{\Omega^e} N_p \rho A N_q \, dx \quad (7.3.25)$$

Exercise 2. Evaluate (7.3.25) using Hermite cubics.

7.3.1 Standard Error Estimates

For the types of problems considered in Chapters 1 through 3, standard error estimates are available for eigenvalues and eigenfunctions. These estimates are the counterparts of those given in Sec. 4.1. The more-sophisticated methods presented after Sec. 4.1 and in Chapters 5 and 6 lie outside the realm of these estimates and therefore must be considered on a separate basis. The present estimates indicate the optimality of the Galerkin finite element method for the standard classes of elliptic eigenvalue problems. The main results are [2]:

$$\lambda_l \leq \lambda_l^h \leq \lambda_l + ch^{2(k+1-m)} \lambda_l^{(k+1)/m} \quad (7.3.26)$$

$$\|u_{(l)}^h - u_{(l)}\|_m \leq ch^{(k+1-m)} \lambda_l^{(k+1)/2m} \quad (7.3.27)$$

where c is a constant independent of h and λ_l .

Remarks

1. Note that the first inequality in (7.3.26) says that the l th approximate eigenvalue is bounded from below by the l th exact eigenvalue. This “upper-bound” property is *not* preserved once the Galerkin rules are violated (e.g., when reduced integration or incompatible modes are employed).

2. The rate of convergence (i.e., power of h) of eigenvalues is *twice* that of eigenfunctions in the H^m -norm [compare (7.3.26) with (7.3.27)].

3. The appearance of powers of the eigenvalue on the right-hand sides of (7.3.26) and (7.3.27) suggests that the quality of approximation *deteriorates* for higher modes (recall $0 < \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \dots$). This will be shown to be the case in specific examples.

4. In frequency analysis we are interested in the error in $\omega_l^h = (\lambda_l^h)^{1/2}$. It turns out that the order of error in the frequency is the same as that in the eigenvalue. This can be seen as follows: Let

$$\epsilon = \frac{\lambda_l^h}{\lambda_l} - 1 \quad (7.3.28)$$

Then,

$$\frac{\omega_l^h}{\omega_l} = \left(\frac{\lambda_l^h}{\lambda_l} \right)^{1/2} = (1 + \epsilon)^{1/2} = 1 + \frac{\epsilon}{2} + O(\epsilon^2) \quad (7.3.29)$$

and so

$$\frac{\omega_l^h}{\omega_l} - 1 = O(\epsilon) = O(h^{2(k+1-m)}) \quad (7.3.30)$$

Thus we can expect the same rate of convergence for frequencies and eigenvalues.

5. An L_2 -estimate for eigenfunctions is also available [2]:

$$\| \mathbf{u}_{(l)}^h - \mathbf{u}_{(l)} \|_0 \leq ch^\sigma \lambda_l^{(k+1)/(2m)} \quad (7.3.31)$$

where

$$\sigma = \min\{k + 1, 2(k + 1 - m)\} \quad (7.3.32)$$

6. The accuracy of the eigenvalues and eigenfunctions are measures of the quality of both M and K , in other words, the entire spatial discretization.

Example 1

Consider an elastic boundary value problem ($m = 1$) and assume linear elements are employed ($k = 1$). The error estimates take the form:

$$\frac{\omega_l^h}{\omega_l} - 1 = O(h^2)$$

$$\| \mathbf{u}_{(l)}^h - \mathbf{u}_{(l)} \|_1 = O(h)$$

For quadratic-level elements ($k = 2$) we have

$$\frac{\omega^h}{\omega_l} - 1 = O(h^4)$$

$$\|u_{(l)}^h - u_{(l)}\|_1 = O(h^2)$$

Example 2

Consider the Hermite cubic beam element ($m = 2$, $k = 3$):

$$\frac{\omega^h}{\omega_l} - 1 = O(h^4)$$

$$\|u_{(l)}^h - u_{(l)}\|_2 = O(h^2)$$

Remark

If numerical quadrature is employed to integrate the element mass and/or stiffness, the same conditions apply as described in Sec. 4.1. That is, a rule accurate enough to exactly integrate all monomials through degree $\bar{k} + k - 2m$ is sufficient to maintain full rate of convergence. (Recall \bar{k} is the order of the highest-order monomial appearing in the element shape functions.) A sufficient condition for convergence is that the rule exactly integrate monomials through degree $\bar{k} - m$.

Exercise 3. Use the “minimax” characterization of eigenvalues, i.e.,

$$\lambda_l = \min_{E_l} \{ \max_{u \in E_l} \mathcal{R}(u) \} \quad (7.3.33)$$

$$\lambda_l^h = \min_{E_l^h} \{ \max_{u^h \in E_l^h} \mathcal{R}(u^h) \} \quad (7.3.34)$$

where \mathcal{R} denotes the *Rayleigh quotient*, defined by

$$\mathcal{R}(u) = \frac{a(u, u)}{(u, \rho u)} \quad (7.3.35)$$

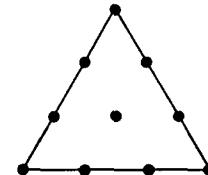
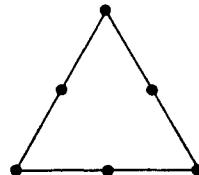
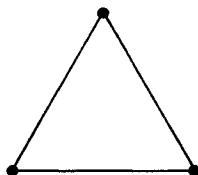
and E_l and E_l^h are l -dimensional subspaces of H^m and \mathcal{S}^h , respectively, to establish $\lambda_l \leq \lambda_l^h$, $l = 1, 2, \dots, n_{eq}$. (The bilinear form in the denominator of (7.3.35) may be $b(u, u)$, $(u, \rho cu)$, etc., as in previously discussed cases.) This result is obtained in [2].

Exercise 4. An isoparametric finite element is used in a series of convergence rate studies on linear elastic eigenvalue problems. All tests indicate that the fundamental frequency behaves as follows:

$$\log\left(\frac{\omega^h}{\omega} - 1\right) \sim 2 \log h$$

where ω is the exact frequency, ω^h is the finite element frequency, and h is the mesh parameter.

- a. Which of the following elements was used in the study?



Justify your answer.

- b. What is the rate of convergence of eigenvectors in the L_2 and H^1 norms?

Exercise 5. Consider the application of bilinear quadrilateral elements to frequency calculations in elasticity. The mass matrix is to be constructed using a Gaussian quadrature formula. The choice is between the one-point, 2×2 , and 3×3 rules. Pick the rule involving the minimum number of points that guarantees that the full rate of convergence of consistent mass is maintained. Explain how you arrived at your answer.

Solution for Exercise 5 The rule must exactly integrate polynomials of degree $\bar{k} + k - 2m = 2 + 1 - 2 = 1$ (i.e., constants and linear terms must be integrated exactly). The *one-point Gauss rule* achieves this accuracy.

7.3.2 Alternative Definitions of the Mass Matrix; Lumped and Higher-order Mass

The recipe given by the Galerkin formulation for the mass matrix is sometimes referred to as variationally “consistent.” Prior to establishment of the variational foundations of the finite element method, mass matrices were concocted in ad hoc fashion. Archer [3] is generally acknowledged as having first pointed out the correctness of *consistent mass*. The consistent-mass matrix leads to the optimal error estimates described in the preceding section. However, as is often the case in finite element analysis, there is strong motivation for breaking the basic rules. This is particularly true in the development of mass matrices. In practice, diagonal or “lumped” mass matrices are often employed due to their general economy and because they lead to some especially attractive time-integration schemes (so-called explicit methods; see Chapters 8 and 9). There are several ways of going about the construction of lumped-mass matrices. One way is to employ nodal quadrature rules.

Mass Lumping by Nodal Quadrature

An integration formula with integration points at the nodes takes the form (see Chapter 3):

$$\int_{\square} g(\xi) d\square \cong \sum_{a=1}^{n_{\text{eq}}} g(\xi_a) W_a \quad (7.3.36)$$

The effect of such a rule on the mass matrix is to diagonalize it, viz.,

$$\begin{aligned}
 m_{pq}^e &= \delta_{ij} \int_{\Omega^e} N_a \rho N_b d\Omega \\
 &= \delta_{ij} \int_{\square} N_a \rho N_b j d\square \\
 &\cong \delta_{ij} \sum_{c=1}^{n_{en}} \underbrace{N_a(\xi_c)}_{\delta_{ac}} \rho(\xi_c) \underbrace{N_b(\xi_c)}_{\delta_{bc}} j(\xi_c) W_c \\
 &= \begin{cases} \delta_{ij} \rho(\xi_a) j(\xi_a) W_a & a = b \\ 0 & a \neq b \end{cases} \quad (\text{no sum on } a)
 \end{aligned} \tag{7.3.37}$$

A theory of numerical quadrature presented in [2] and described earlier defines under what conditions the nodal quadrature mass matrix is convergent and retains full order of accuracy. In the one-dimensional case, full rate of convergence is maintained if the rule is capable of exactly integrating polynomials of degree $2(k - m)$; the minimum condition for convergence is a rule that can exactly integrate polynomials of degree $k - m$. (It should be pointed out that these are *sufficient* conditions. They may not be *necessary*.)

Example 1

Consider piecewise linear shape functions in one dimension and assume $\rho = \text{constant}$. The trapezoidal rule, namely,

$$\xi_1 = -1, \quad \xi_2 = 1, \quad W_1 = W_2 = 1$$

is sufficiently accurate to maintain full rate of convergence (i.e., $2(k - m) = 2(1 - 1) = 0$; thus only constants need to be exactly integrated). It may be verified that in this case

$$m_{\text{lumped}}^e = \frac{\rho h^e}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7.3.38}$$

Example 2

Consider piecewise quadratics. Simpson's rule,

$$\xi_1 = -1, \quad \xi_2 = 0, \quad \xi_3 = 1, \quad W_1 = W_3 = \frac{1}{3}, \quad W_2 = \frac{4}{3}$$

is capable of exactly integrating cubic polynomials and so attains full rate of convergence. (The requirement is $2(k - m) = 2(2 - 1) = 2$; i.e., quadratic polynomials.) The lumped-mass matrix is (verify!):

$$m_{\text{lumped}}^e = \frac{\rho h^e}{6} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.3.39}$$

Numerical confirmation of the convergence rate of the fundamental frequency is presented in Fig. 7.3.2. Frequency spectra are presented in Fig. 7.3.3. The upper-bound property of consistent mass (i.e., $\omega^h/\omega \geq 1$) is clearly in evidence in Fig. 7.3.3.

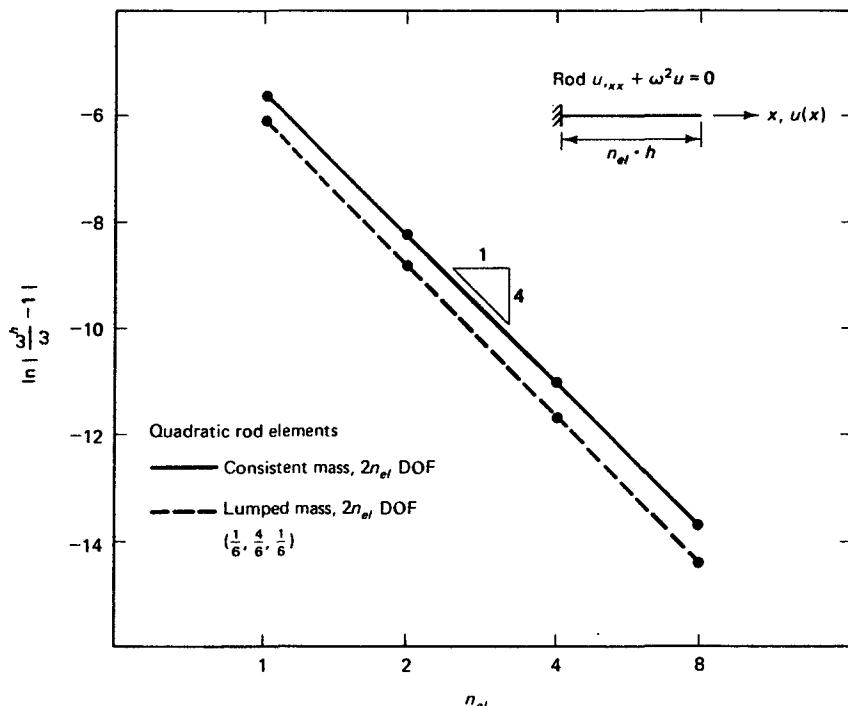


Figure 7.3.2 Convergence of the fundamental frequency for quadratic rod elements (Hughes et al. [4]).

Example 3

For the standard four-node cubic element we need a rule capable of exactly integrating quartic polynomials (i.e., $2(k - m) = 2(3 - 1) = 4$). Unfortunately, for the usual setup in ξ -space, this is impossible. The restriction of interior nodal placement at $\xi = \pm \frac{1}{3}$ limits the order of convergence to cubic:

| g | $\int_{-1}^{+1} g d\xi$ | $\sum_{a=1}^4 g(\xi_a) W_a$ | |
|---------|-------------------------|--|--|
| 1 | 2 | $= W_1 + W_2 + W_3 + W_4$ | $\Rightarrow \begin{cases} W_1 = \frac{1}{4} \\ W_2 = \frac{3}{4} \\ W_3 = W_2 \\ W_4 = W_1 \end{cases}$ |
| ξ | 0 | $= -W_1 - \frac{1}{3}W_2 + \frac{1}{3}W_3 + W_4$ | |
| ξ^2 | $\frac{2}{3}$ | $= W_1 + \frac{1}{9}W_2 + \frac{1}{9}W_3 + W_4$ | |
| ξ^3 | 0 | $= -W_1 - \frac{1}{27}W_2 + \frac{1}{27}W_3 + W_4$ | |
| ξ^4 | $\frac{2}{5}$ | $\neq W_1 + \frac{1}{81}W_2 + \frac{1}{81}W_3 + W_4$ | |

Example 4

Let us reconsider the four-node cubic element. We need one additional parameter in order to create a nodal rule that exactly integrates quartic polynomials. If we relax the condition that the interior nodes are located at $\xi = \pm \frac{1}{3}$, a sufficiently accurate rule can

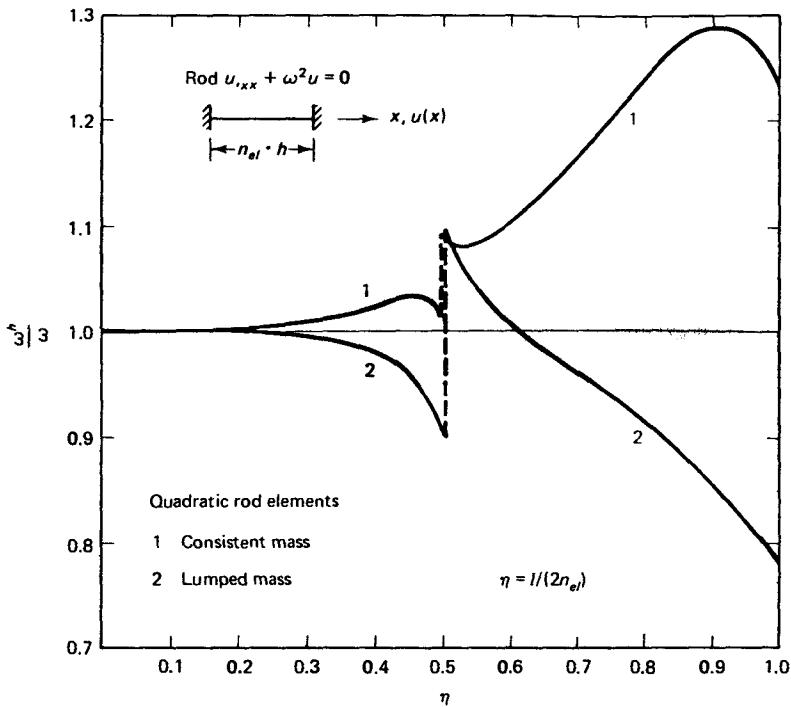


Figure 7.3.3 Frequency spectra for quadratic rod elements (Hughes et al. [4]). η is the mode number.

be developed. Consider the ξ -space setup shown in Fig. 7.3.4. The object is to pick c along with the weights so that an arbitrary quartic polynomial is exactly integrated. The idea is similar to that which led to the Gauss rules. Assume $W_4 = W_1$ and $W_3 = W_2$. Then

| g | $\int_{-1}^{+1} g d\xi$ | $W_1(g(-1) + g(+1)) + W_2(g(-c) + g(+c))$ |
|---------|-------------------------|---|
| 1 | 2 | $= 2(W_1 + W_2)$ |
| ξ | 0 | $= 0$ |
| ξ^2 | $\frac{2}{3}$ | $= 2(W_1 + c^2 W_2)$ |
| ξ^3 | 0 | $= 0$ |
| ξ^4 | $\frac{2}{3}$ | $= 2(W_1 + c^4 W_2)$ |
| ξ^5 | 0 | $= 0$ |

There are three equations for the three unknowns. The first equation is used to eliminate

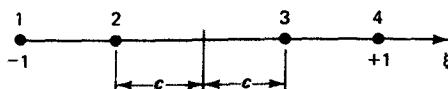


Figure 7.3.4

W_1 from the second and third. Dividing the third equation by the second yields $c^2 = \frac{1}{5}$. Thus $W_1 = \frac{1}{6}$ and $W_2 = \frac{5}{6}$. The mass then is

$$\mathbf{m}_{\text{lumped}}^e = \frac{\rho h^e}{12} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.3.40)$$

This element mass was originally developed by Fried and Malkus [5]. Their numerical tests confirmed the theoretically predicted convergence rate (i.e., $\lambda^h - \lambda = O(h^6)$).

Note that the relocation of the interior nodes necessitates the derivation of new shape functions.

Remark

The rule derived above is an example of a *Lobatto quadrature rule*. The Lobatto rules are similar to the Gauss rules, except in Lobatto rules the end points of the interval are always included. The locations of the interior points are determined to maximize the accuracy of the rule. The trapezoidal rule and Simpson's rule are the first two Lobatto rules. Lobatto rules for $n_{en} = 3$ through $n_{en} = 10$ are listed in Table 7.3.1.

TABLE 7.3.1 Lobatto quadrature rules

| | $\pm \xi_a$ | W_a | | $\pm \xi_a$ | W_a |
|---|-------------|-------------|----|---------------|---------------|
| 3 | 1.00000 000 | 0.33333 333 | 7 | 1.00000 000 | 0.04761 904 |
| | 0.00000 000 | 1.33333 333 | | 0.83022 390 | 0.27682 604 |
| | | | | 0.46884 879 | 0.43174 538 |
| | | | | 0.00000 000 | 0.48761 904 |
| 4 | 1.00000 000 | 0.16666 667 | 8 | 1.00000 000 | 0.03571 428 |
| | 0.44721 360 | 0.83333 333 | | 0.87174 015 | 0.21070 422 |
| | | | | 0.59170 018 | 0.34112 270 |
| | | | | 0.20929 922 | 0.41245 880 |
| 5 | 1.00000 000 | 0.10000 000 | 9 | 1.00000 00000 | 0.02777 77778 |
| | 0.65465 367 | 0.54444 444 | | 0.89975 79954 | 0.16549 53616 |
| | 0.00000 000 | 0.71111 111 | | 0.67718 62795 | 0.27453 87126 |
| | | | | 0.36311 74638 | 0.34642 85110 |
| 6 | 1.00000 000 | 0.06666 667 | 10 | 1.00000 00000 | 0.02222 22222 |
| | 0.76505 532 | 0.37847 495 | | 0.91953 39082 | 0.13330 59908 |
| | 0.28523 152 | 0.55485 838 | | 0.73877 38651 | 0.22488 93420 |
| | | | | 0.47792 49498 | 0.29204 26836 |
| | | | | 0.16527 89577 | 0.32753 97612 |

Exercise 6. Derive the cubic shape functions of the four-node *Lobatto element*.

Example 5

Consider the Hermite cubic beam element. Because the nodes are located at the end points, the trapezoidal rule will produce a lumped mass:

$$\mathbf{m}_{\text{lumped}}^e = \frac{\rho h^4}{2} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (7.3.41)$$

Note that half the mass is lumped at each translational degree of freedom and that there are zero masses at the rotational (i.e., slope) degrees of freedom. In order for the full rate of convergence of consistent mass to be guaranteed, the rule should be able to integrate exactly polynomials of degree $2(k - m) = 2(3 - 2) = 2$. The trapezoidal rule is capable of integrating only linear polynomials. Nevertheless, computational results suggest full accuracy is achieved, i.e., (see Fig. 7.3.5),

$$\frac{\omega_i^h}{\omega_i} - 1 = O(h^4) \quad (7.3.42)$$

(This does not contradict the numerical integration theory, which provides sufficient, rather than necessary, conditions.)

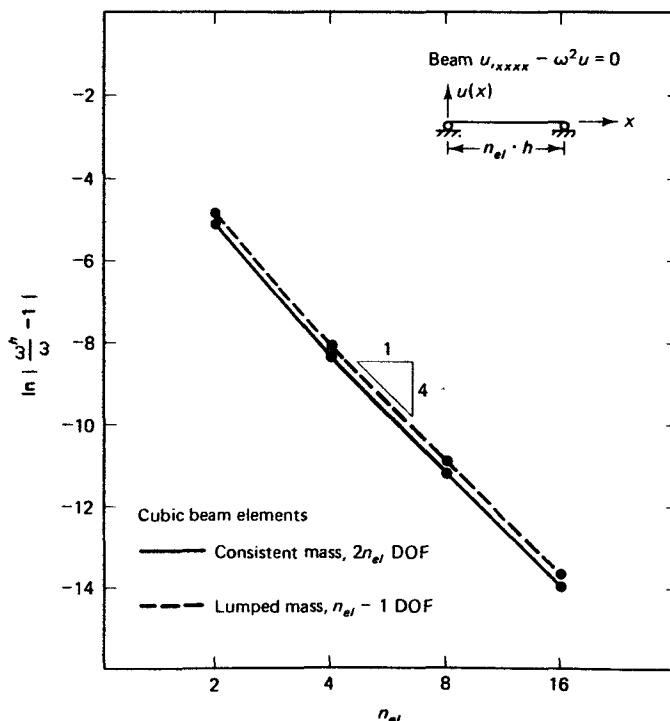


Figure 7.3.5 Convergence of the fundamental frequency for a simply supported beam (Hughes et al. [4]).

Example 6

Optimal quadrature rules have been developed for triangles by Fried and Malkus [5]. The rules are summarized in Fig. 7.3.6.

The theoretical eigenvalue convergence rate for each of the three triangles has been verified numerically in [5]. Note that for the quadratic triangle there are zero masses at the vertices. For the cubic triangle the edge nodes need to be relocated as shown. The vertex masses in this case are negative. Zero and negative masses pose significant impediments in practical applications. (See Remark 1, following the examples.)

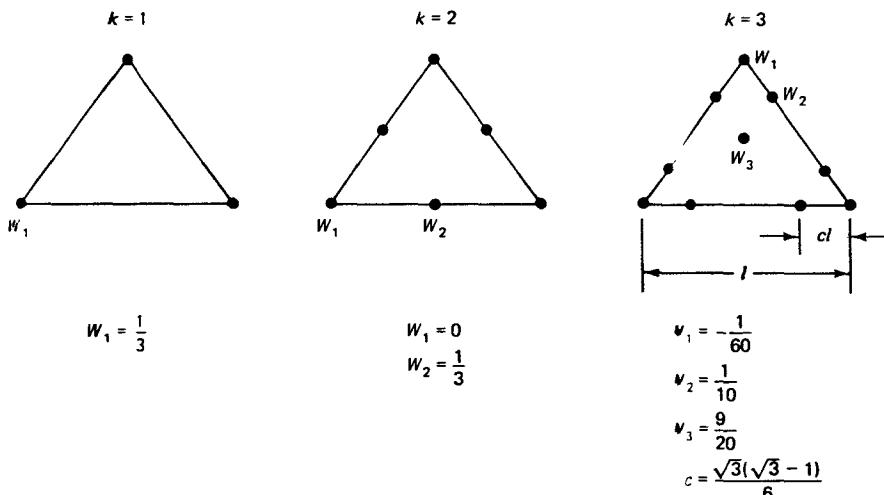
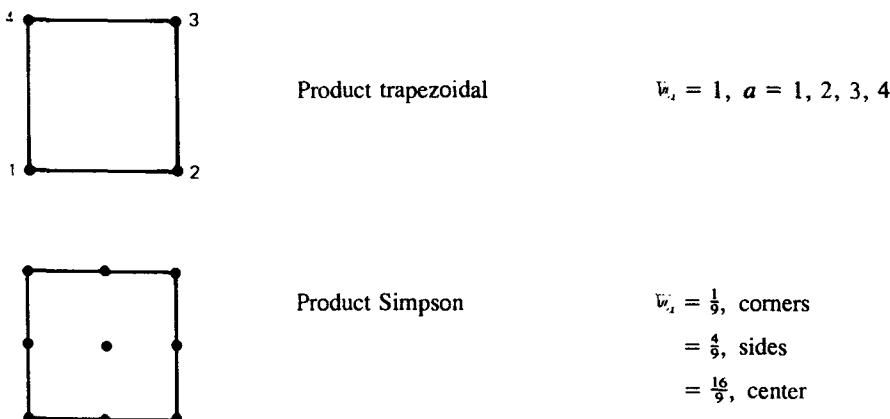
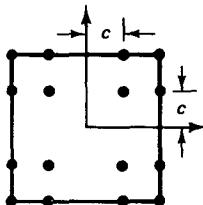


Figure 7.3.6 Optimally accurate nodal quadrature rules for triangles.

Example 7

For nodal patterns that are products of one-dimensional Lobatto patterns, products of Lobatto rules may be employed. Some examples are illustrated in Fig. 7.3.7 below and on the top of p. 443.



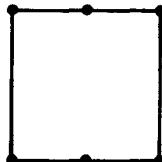


Product Lobatto

$$W_a = \frac{1}{36}, \text{ corners}$$

$$= \frac{5}{36}, \text{ sides}$$

$$= \frac{25}{36}, \text{ interior}$$



$$\text{Trapezoidal } \uparrow \times \text{ Simpson } \rightarrow \quad W_a = \frac{1}{3}, \text{ corners}$$

$$= \frac{4}{3}, \text{ sides}$$

Figure 7.3.7 Optimal nodal quadrature rules for product Lobatto patterns.

Example 8

For nodal patterns that are not products of one-dimensional patterns, optimal nodal quadrature rules must be worked out on a case-by-case basis. As an example, consider the eight-node serendipity element (see Fig. 7.3.8). Let the weights assigned to corner and side nodes be denoted W_1 and W_2 , respectively. A rule can be developed that is accurate for all cubic polynomials. This turns out to be of high enough precision to guarantee full rate of convergence (i.e., $\bar{k} + k - 2m = 3 + 2 - 2 = 3$). However, the corner masses are negative and thus this mass matrix is unsuitable for practical use. The calculations leading to the rule are summarized below:

| g | $\int_{\square} g \, d\square$ | $W_1 \sum_{a=1}^4 g_a + W_2 \sum_{a=5}^8 g_a$ |
|-------------|--------------------------------|---|
| 1 | 4 | $= 4(W_1 + W_2)$ |
| ξ | 0 | $= 0$ |
| η | 0 | $= 0$ |
| $\xi\eta$ | 0 | $= 0$ |
| ξ^2 | $\frac{4}{3}$ | $= 4W_1 + 2W_2$ |
| η^2 | $\frac{4}{3}$ | $= 4W_1 + 2W_2$ |
| ξ^3 | 0 | $= 0$ |
| $\xi^2\eta$ | 0 | $= 0$ |
| $\xi\eta^2$ | 0 | $= 0$ |
| η^3 | 0 | $= 0$ |

$$\therefore W_1 = -\frac{1}{3}, W_2 = \frac{4}{3}$$

Remarks

1. We have asserted that negative masses are unacceptable in practice. The reason for this can be seen by considering single degree-of-freedom model equations of parabolic and hyperbolic type:

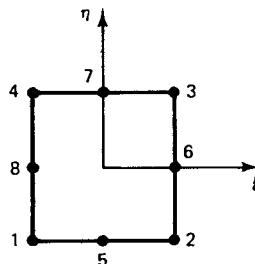


Figure 7.3.8 Eight-node serendipity element.

$$\begin{cases} \ddot{d} + \lambda d = 0, & d = \exp(-\lambda t)d_0 \quad (\text{exponential decay}) \\ -\ddot{d} + \lambda d = 0, & d = \exp(\lambda t)d_0 \quad (\text{unbounded}) \\ \ddot{d} + \omega^2 d = 0, & d = A \sin \omega t + B \cos \omega t, \quad (\text{oscillatory and bounded}) \\ -\ddot{d} + \omega^2 d = 0, & d = A \sinh \omega t + B \cosh \omega t \quad (\text{unbounded}) \end{cases}$$

In each case negative mass produces nonphysical behavior. Modes in a multidegree of freedom system will behave in this fashion, vitiating the true solution, when negative masses are present.

2. If nodal quadrature is used in axisymmetric analysis, zero masses result along the axis of symmetry for all elements. This can be seen from the following:

$$m_{pq}^e = 2\tau \delta_{ij} \int_{\Omega^e} \rho N_a N_b r dr dz \quad (7.3.43)$$

$$\cong \begin{cases} 2\pi \delta_{ij} \rho(\xi_a) r(\xi_a) j(\xi_a) W_a, & a = b \\ 0 & a \neq b \end{cases} \quad (7.3.44)$$

Along the axis of symmetry, $r = 0$, and consequently so are the masses of nodes located there.

As we have seen, due to a tendency to generate zero and negative masses, nodal quadrature is not always an effective tool for producing useful lumped-mass matrices. Thus, other methods have been considered.

Row-sum technique

In the row-sum technique the lumped mass is defined by

$$m'_{ii} = \begin{cases} \delta_{ij} \int_{\Omega^e} \rho N_a d\Omega & a = b \\ 0 & a \neq b \end{cases} \quad (7.3.45)$$

The name derives from the fact that

$$\sum_{b=1}^{n_{en}} \int_{\Omega^e} N_a \rho N_b d\Omega = \int_{\Omega^e} N_a \rho \underbrace{\left(\sum_{b=1}^{n_{en}} N_b \right)}_1 d\Omega = \int_{\Omega^e} N_a \rho d\Omega \quad (7.3.46)$$

That is, the elements in each row are summed and lumped on the diagonal.

The row-sum technique eliminates the problem of zero masses at nodes along the z -axis in problems of axisymmetry. However, like nodal quadrature, it sometimes produces negative masses. This is the case for corner nodes of the eight-node serendipity element.

"Special Lumping Technique"

The special lumping technique was developed by Hinton et al. [6]. The name is not too revealing, but the method has an important attribute: *It always produces positive lumped masses*. The idea is to set the entries of the lumped-mass matrix proportional to the diagonal entries of the consistent mass. (By virtue of positive-definiteness, the diagonal entries of consistent mass are necessarily positive.) The constant of proportionality is selected to conserve the total element mass. The formula is

$$m_{pq}^e = \begin{cases} \alpha \delta_{ij} \int_{\Omega^e} \rho N_a^2 d\Omega & a = b \\ 0 & a \neq b \end{cases} \quad (7.3.47)$$

where

$$\alpha = \frac{\int_{\Omega^e} \rho d\Omega}{\left(\sum_{a=1}^{n_m} \int_{\Omega^e} \rho N_a^2 d\Omega \right)} \quad (7.3.48)$$

total element mass sum of diagonal entries of consistent mass

The special lumping procedure has been shown numerically to work well on structural and solid mechanics problems [6]. Optimal rates of convergence are typically achieved. Presently, it is the only lumping method that can be recommended for arbitrary elements. Unfortunately, no mathematical theory in support of it has been forthcoming.

Exercise 7. For simple elements, the lumping procedures described above tend to produce similar if not identical masses. Verify this assertion by calculating element masses for the linear triangle and bilinear quadrilateral. Assume the quadrilateral is in the rectangular configuration and in all cases take $\rho = \text{constant}$. Use nodal quadrature, row-sum, and the special lumping techniques.

Exercise 8. Calculate diagonal mass matrices for the one-dimensional three-node element by the row-sum and special lumping techniques: Compare the results with the diagonal mass matrix obtained by Simpson's rule. Comment.

Remark

Although lumped masses have often been used successfully in solid and struc-

tural mechanics and heat conduction, some disappointing results have been obtained in flux mechanics (see Gresho et al. [7]).

Higher-Order Mass

The accuracy of consistent mass can often be achieved by a much simpler lumped mass. A question naturally arises: Is consistent mass the best nondiagonal element mass? The answer appears to be: not always. There are examples of element mass matrices that exhibit superior accuracy to consistent mass. However, no general theory of obtaining higher-order accurate mass matrices exists yet.

Example 1

The first and simplest example of a higher-order accurate mass matrix was presented by Goudreau [8]. The matrix is to be viewed as an alternative for the two-node rod element. Recall the consistent and lumped mass matrices for this element are (respectively)

$$\mathbf{m}_{\text{consistent}}^e = \frac{\rho h^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad (7.3.49)$$

$$\mathbf{m}_{\text{lumped}}^e = \frac{\rho h^e}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (7.3.50)$$

Both of these mass matrices result in second-order accurate eigenvalues. The higher-order mass matrix is simply the average of (7.3.49) and (7.3.50):

$$\mathbf{m}_{\text{higher-order}}^e = \frac{\rho h^e}{12} \begin{bmatrix} 5 & 1 \\ 1 & 5 \end{bmatrix} \quad (7.3.51)$$

For a uniform mesh, it can be analytically shown that (7.3.51) leads to fourth-order accurate frequencies [8]. Numerical results supporting this assertion are presented for a fixed-fixed rod in Fig. 7.3.9. The same order of convergence is exhibited for a non-uniform mesh in Fig. 7.3.10 and for fixed-free boundary conditions in Fig. 7.3.11. Frequency spectra are presented in Fig. 9.1.4.)

For purposes of generalization to other elements it may be observed that (7.3.51) may be produced by employing the quadrature rule $\xi_1 = -\xi_2 = \sqrt{\frac{2}{3}}$, $W_1 = W_2 = 1$, on the element mass integral [10]. Clearly this rule can be iterated for multidimensional applications.

Another way of deriving (7.3.51) is to take a linear combination of the consistent mass and stiffness:

$$\begin{aligned} \mathbf{m}^e &= \mathbf{m}_{\text{consistent}}^e + \left(\frac{1}{6} - r\right) \frac{\rho(h^e)^2}{E} \mathbf{k}^e \\ &= \frac{\rho h^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} + \rho h^e \left(\frac{1}{6} - r\right) \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \end{aligned} \quad (7.3.52)$$

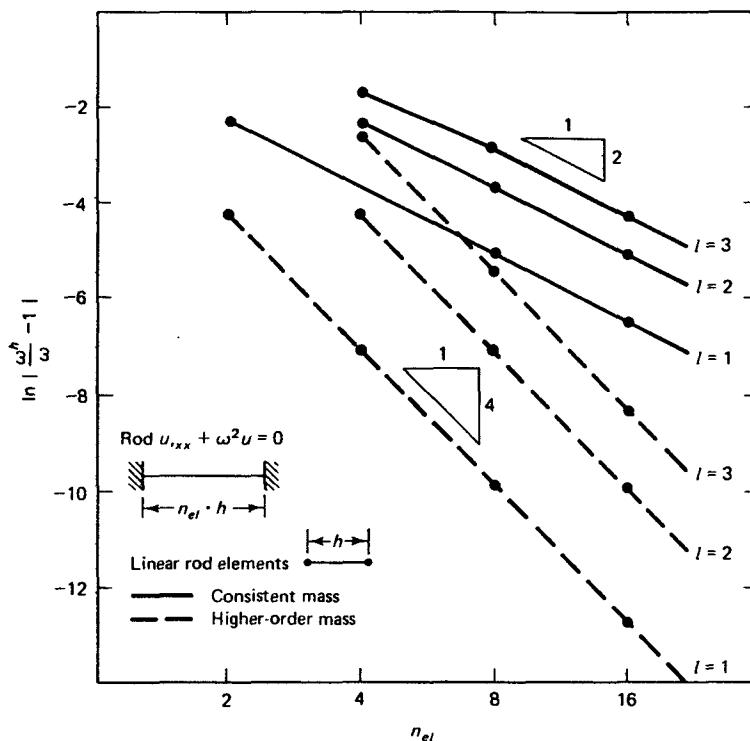


Figure 7.3.9 Convergence of the first three frequencies for linear rod elements [9].

Clearly, for $r = \frac{1}{12}$ we arrive at (7.3.51). The consistent and lumped mass matrices correspond to $r = \frac{1}{6}$ and $r = 0$, respectively.

Example 2

A higher-order mass matrix for the Hermite cubic beam element may be derived by taking a linear combination of the consistent mass and stiffness:

$$\mathbf{m}_{\text{higher-order}}^e = \mathbf{m}_{\text{consistent}}^e + \alpha \frac{\rho A (h^e)^4}{EI} \mathbf{k}^e \quad (7.3.53)$$

Taylor and Iding [11] first experimented with matrices of this form, but did not obtain an optimal value of α . The value $\alpha = \frac{1}{720}$ was derived in [9] and shown to lead to sixth-order accurate frequencies for a uniform mesh. Numerical confirmation is presented in Fig. 7.3.12 for a simply supported beam. Frequency spectra are presented in Fig. 7.3.13.

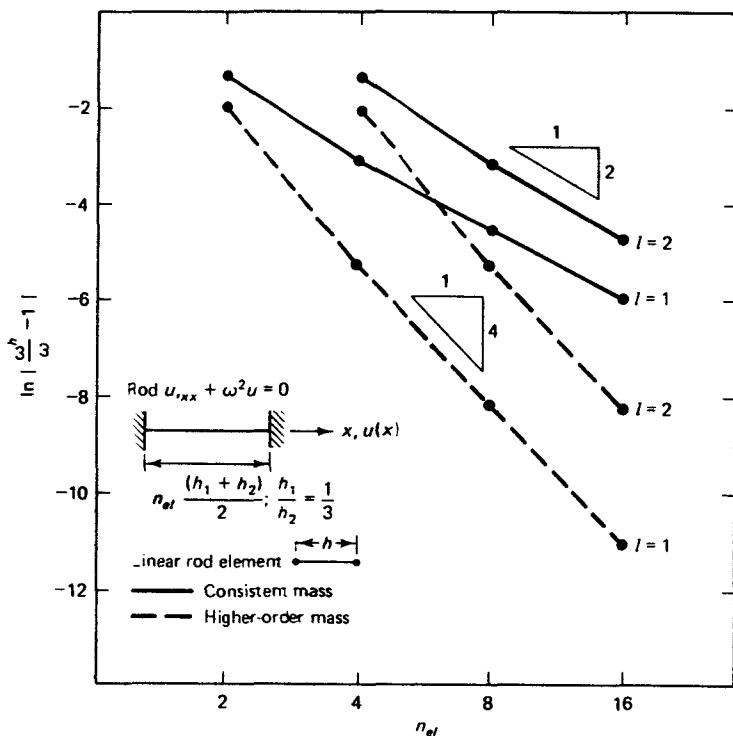


Figure 7.3.10 Convergence of the first two frequencies for linear rod elements with nonuniform mesh [9].

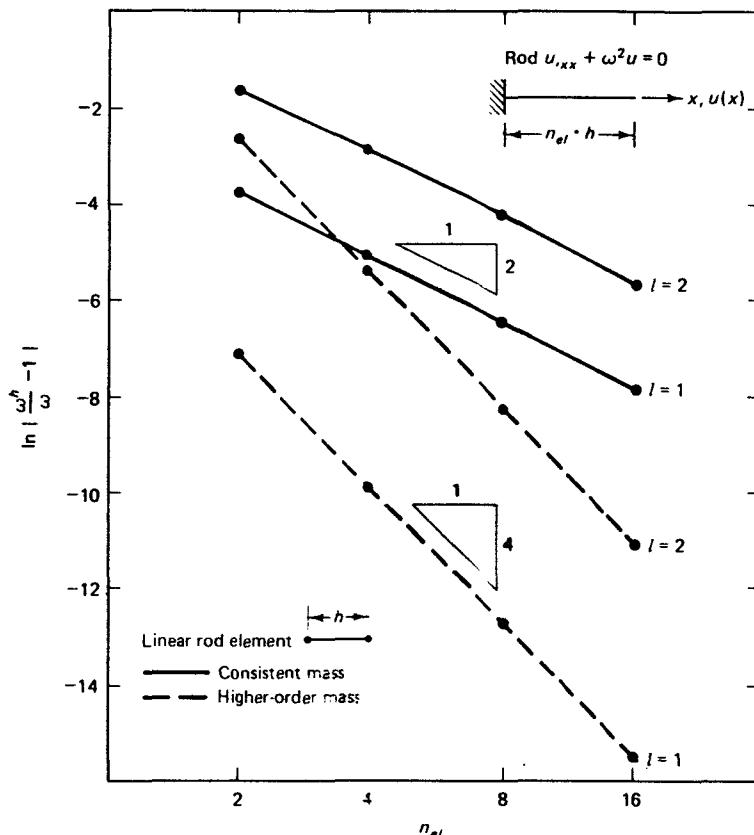


Figure 7.3.11 Convergence of the first two frequencies for linear rod elements with one end fixed and one end free [9].

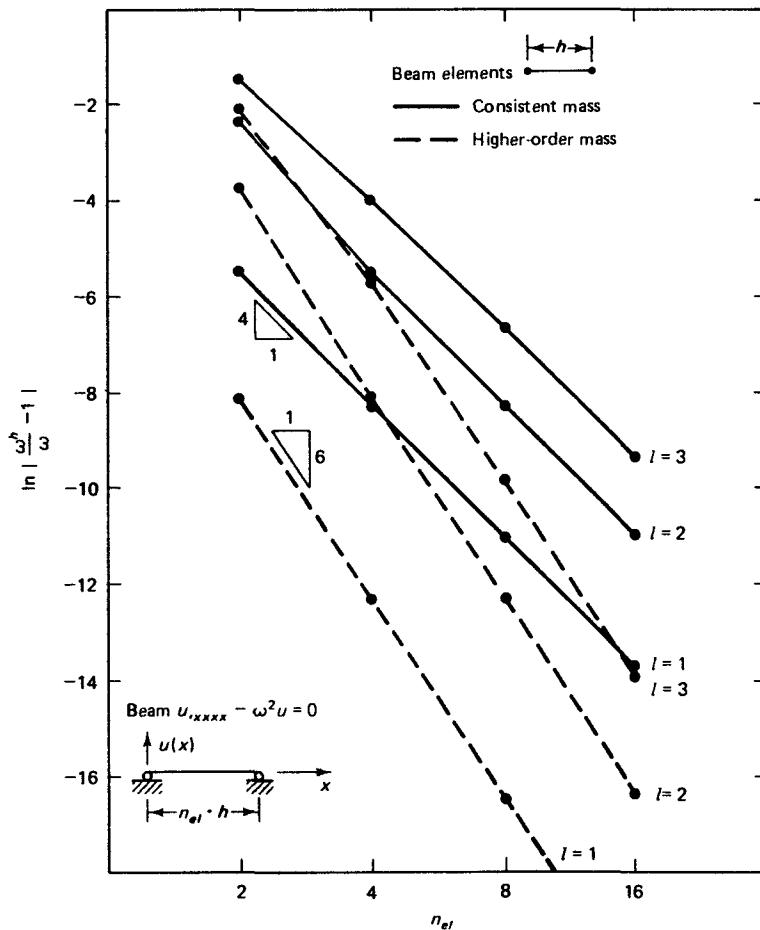


Figure 7.3.12 Convergence of the first three frequencies of a simply supported beam [9].

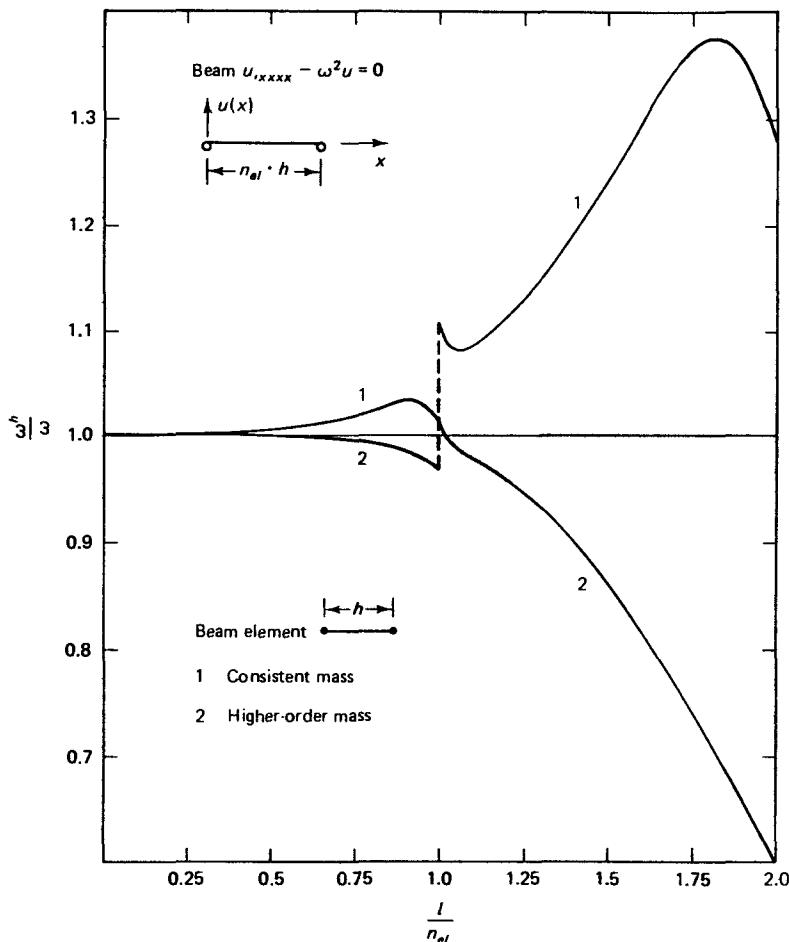


Figure 7.3.13 Frequency spectra for the cubic beam element [9].

7.3.3 Estimation of Eigenvalues

In time-dependent calculations it is often necessary to have a conservative estimate of the maximum eigenvalue (i.e., $\lambda_{n_{eq}}^h$). (We will return to this point in subsequent chapters.) It can be shown that [12,13]

$$\lambda_{n_{eq}}^h \leq \max_e (\lambda_{\max}^e) \quad (7.3.54)$$

where λ_{\max}^e is the maximum eigenvalue of element e . The eigenvalues of element e satisfy²

$$(k^e - \lambda^e m^e) \Psi^e = 0 \quad (7.3.55)$$

In the remainder of this subsection we will present an argument in support of (7.3.54).

Lemma

$$\lambda_{n_{eq}}^h = \max_{\substack{\Phi \in \mathbb{R}^{n_{eq}} \\ \Phi \neq 0}} \mathcal{R}(\Phi) \quad (7.3.56)$$

where $\mathcal{R}(\Phi) = \Phi^T K \Phi / \Phi^T M \Phi$ is the Rayleigh quotient, and the maximum occurs for $\Phi = c \Psi_{n_{eq}}$, where c is an arbitrary nonzero constant.

Proof. The eigenvectors $\Psi_1, \Psi_2, \dots, \Psi_{n_{eq}}$ form a basis for $\mathbb{R}^{n_{eq}}$. Thus we can expand any vector, say Φ , as

$$\Phi = \sum_{l=1}^{n_{eq}} c_l \Psi_l \quad (7.3.57)$$

and therefore

$$\Phi^T M \Phi = \sum_{l=1}^{n_{eq}} |c_l|^2 \quad (7.3.58)$$

$$\Phi^T K \Phi = \sum_{l=1}^{n_{eq}} \lambda_l^h |c_l|^2 \quad (7.3.59)$$

For $\Phi \neq 0$,

$$\mathcal{R}(\Phi) = \frac{\sum_{l=1}^{n_{eq}} \lambda_l^h |c_l|^2}{\sum_{l=1}^{n_{eq}} |c_l|^2} \leq \lambda_{n_{eq}}^h \quad (7.3.60)$$

by virtue of $\lambda_1^h \leq \lambda_2^h \leq \dots \leq \lambda_{n_{eq}}^h$. Because $\mathcal{R}(c \Psi_{n_{eq}}) = \lambda_{n_{eq}}^h$, the proof of the lemma is complete. ■

²Solution of the element eigenvalue problem is actually rarely done. Whenever possible an exact formula is developed for λ_{\max}^e . Otherwise a conservative estimate is developed. This subject will be discussed further in Chapter 9.

Let \mathbf{K} and \mathbf{M} be global matrices formed by assembling the collections of matrices $\{\mathbf{k}^e\}_{1}^{n_e}$ and $\{\mathbf{m}^e\}_{1}^{n_e}$ according to some connectivity array (i.e., an “LM-array”; see Chapter 2). Consider the symmetric, block-diagonal matrices

$$\tilde{\mathbf{K}} = \begin{bmatrix} \mathbf{k}^1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{k}^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{k}^{n_e} \end{bmatrix} \quad (7.3.61)$$

$$\tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{m}^1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{m}^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{m}^{n_e} \end{bmatrix} \quad (7.3.62)$$

The dimension of $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{M}}$, say \tilde{n}_{eq} , clearly satisfies $\tilde{n}_{eq} \geq n_{eq}$.

The eigenvalue problems for the \mathbf{K} , \mathbf{M} , and $\tilde{\mathbf{K}}$, $\tilde{\mathbf{M}}$ systems are (respectively)

$$(\mathbf{K} - \lambda^h \mathbf{M}) \Psi = \mathbf{0} \quad (7.3.63)$$

and

$$(\tilde{\mathbf{K}} - \tilde{\lambda}^h \tilde{\mathbf{M}}) \tilde{\Psi} = \mathbf{0} \quad (7.3.64)$$

This lemma may also be applied to the eigenvalue problem (7.3.64). Thus

$$\tilde{\lambda}_{\tilde{n}_{eq}}^h = \max_{\substack{\tilde{\Phi} \in \mathbb{R}^{\tilde{n}_{eq}} \\ \tilde{\Phi} \neq \mathbf{0}}} \frac{\tilde{\Phi}^T \tilde{\mathbf{K}} \tilde{\Phi}}{\tilde{\Phi}^T \tilde{\mathbf{M}} \tilde{\Phi}} \quad (7.3.65)$$

Let $\mathbb{R}_c^{\tilde{n}_{eq}}$ denote the set of vectors $\tilde{\Phi} \in \mathbb{R}^{\tilde{n}_{eq}}$ such that components of $\tilde{\Phi}$ that correspond to the same degree of freedom in $\mathbb{R}^{n_{eq}}$ under the identification of the connectivity array used in assembling \mathbf{K} and \mathbf{M} are to be equal. With this definition, it is clear that

$$\lambda_{n_{eq}}^h = \max_{\substack{\tilde{\Phi} \in \mathbb{R}_c^{\tilde{n}_{eq}} \\ \tilde{\Phi} \neq \mathbf{0}}} \frac{\tilde{\Phi}^T \tilde{\mathbf{K}} \tilde{\Phi}}{\tilde{\Phi}^T \tilde{\mathbf{M}} \tilde{\Phi}} \leq \tilde{\lambda}_{\tilde{n}_{eq}}^h \quad (7.3.66)$$

Remarks

1. The equality in (7.3.66) may be obvious to some readers. If it is not, the following simple example should be helpful:

Let

$$\begin{aligned}
 \mathbf{k}^e &= \begin{bmatrix} k_{11}^e & k_{12}^e \\ k_{21}^e & k_{22}^e \end{bmatrix} \\
 \mathbf{m}^e &= \begin{bmatrix} m_{11}^e & m_{12}^e \\ m_{21}^e & m_{22}^e \end{bmatrix} \\
 e &= 1, 2 \\
 \mathbf{K} &= \begin{bmatrix} k_{11}^1 & k_{12}^1 & 0 \\ k_{21}^1 & (k_{22}^1 + k_{11}^1) & k_{12}^2 \\ 0 & k_{21}^2 & k_{22}^2 \end{bmatrix} \\
 \mathbf{M} &= \begin{bmatrix} m_{11}^1 & m_{12}^1 & 0 \\ m_{21}^1 & (m_{22}^1 + m_{11}^1) & m_{12}^2 \\ 0 & m_{21}^2 & m_{22}^2 \end{bmatrix} \\
 \underbrace{\widetilde{\mathbf{K}}}_{4 \times 4} &= \begin{bmatrix} \mathbf{k}^1 & \mathbf{0} \\ \mathbf{0} & \mathbf{k}^2 \end{bmatrix} \\
 \underbrace{\widetilde{\mathbf{M}}}_{4 \times 4} &= \begin{bmatrix} \mathbf{m}^1 & \mathbf{0} \\ \mathbf{0} & \mathbf{m}^2 \end{bmatrix} \\
 \boldsymbol{\Phi} &= \left\{ \begin{array}{c} \phi_1 \\ \phi_2 \\ \phi_3 \end{array} \right\} \\
 \widetilde{\boldsymbol{\Phi}} &= \left\{ \begin{array}{c} \phi_1 \\ \phi_2 \\ \phi_2 \\ \phi_3 \end{array} \right\} \in \mathbb{R}_c^4
 \end{aligned}$$

The equality in (7.3.66) is established for the example under consideration by the following calculations:

$$\begin{aligned}
 \widetilde{\boldsymbol{\Phi}}^T \widetilde{\mathbf{K}} \widetilde{\boldsymbol{\Phi}} &= \begin{Bmatrix} \phi_1 \\ \phi_2 \end{Bmatrix}^T \mathbf{k}^1 \left\{ \begin{array}{c} \phi_1 \\ \phi_2 \end{array} \right\} + \begin{Bmatrix} \phi_2 \\ \phi_3 \end{Bmatrix}^T \mathbf{k}^2 \left\{ \begin{array}{c} \phi_2 \\ \phi_3 \end{array} \right\} \\
 &= \boldsymbol{\Phi}^T \mathbf{K} \boldsymbol{\Phi} \\
 \widetilde{\boldsymbol{\Phi}}^T \widetilde{\mathbf{M}} \widetilde{\boldsymbol{\Phi}} &= \begin{Bmatrix} \phi_1 \\ \phi_2 \end{Bmatrix}^T \mathbf{m}^1 \left\{ \begin{array}{c} \phi_1 \\ \phi_2 \end{array} \right\} + \begin{Bmatrix} \phi_2 \\ \phi_3 \end{Bmatrix}^T \mathbf{m}^2 \left\{ \begin{array}{c} \phi_2 \\ \phi_3 \end{array} \right\} \\
 &= \boldsymbol{\Phi}^T \mathbf{M} \boldsymbol{\Phi}
 \end{aligned}$$

2. The inequality in (7.3.66) follows directly from (7.3.65).

In terms of the element matrices, the eigenvalue problem (7.3.64) can be written as

$$\begin{bmatrix} k^1 - \tilde{\lambda}^h m^1 & 0 & \dots & 0 \\ 0 & k^2 - \tilde{\lambda}^h m^2 & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & k^{n_{el}} - \tilde{\lambda}^h m^{n_{el}} \end{bmatrix} \begin{Bmatrix} \tilde{\Psi}^1 \\ \tilde{\Psi}^2 \\ \vdots \\ \tilde{\Psi}^{n_{el}} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{Bmatrix} \quad (7.3.67)$$

From this expression it can be seen that the eigenvalues of (7.3.64) are the same as the totality of eigenvalues of the uncoupled element problems

$$(k^e - \lambda^e m^e) \Psi^e = 0, \quad e = 1, \dots, n_{el} \quad (7.3.68)$$

Therefore

$$\tilde{\lambda}_{n_{eq}}^h = \max_e (\lambda_{\max}^e) \quad (7.3.69)$$

where

$$\lambda_{\max}^e = \max_{\substack{\Psi^e \in \mathbb{R}^{n_{ee}} \\ \Psi^e \neq 0}} \frac{\Psi^{eT} k^e \Psi^e}{\Psi^{eT} m^e \Psi^e}$$

Combining (7.3.66) with (7.3.69), we have that

$$\lambda_{n_{eq}}^h \leq \max_e (\lambda_{\max}^e)$$

which completes the proof. ■

Appendix 7.1

Error Estimates for Semidiscrete Galerkin Approximations

In [14] the following error estimates are obtained (c denotes a constant).

Parabolic case

$$\|e(t)\|_0 \leq ch^\mu \left[\|u(t)\|_{k+1} + \exp(-\lambda_1^h t) \|u(0)\|_{k+1} + \int_0^t \exp(\lambda_1^h(\tau - t)) \|\dot{u}(\tau)\|_{k+1} d\tau \right] \quad (7.1.1)$$

where $e = u^h - u$ and $\mu = \min\{k+1, 2(k+1-m)\}$

For each fixed t , the term in (7.1.1) in square brackets will be bounded: Thus this result establishes the convergence of $u^h(t)$ to $u(t)$ in the L_2 -norm as $h \rightarrow 0$.

Hyperbolic case

Let $E(u, \dot{u}) = \frac{1}{2}[(\dot{u}, \rho \ddot{u}) + a(u, u)]$ denote the **total energy**. The square root of E defines a norm on $\mathcal{V} \times L_2$ equivalent to the $H^m \times L_2$ norm. The main result is

$$E(e(t), \dot{e}(t))^{1/2} \leq c \left\{ h^\nu \left[\|u(0)\|_{k+1} + \|u(t)\|_{k+1} \right] \right\}$$

$$+ h^\mu \left[\| \dot{u}(0) \|_{k+1} + \| \dot{u}(t) \|_{k+1} + \int_0^t \| \ddot{u}(\tau) \|_{k+1} d\tau \right] \quad (7.1)$$

where $\nu = k + 1 - m$ and $\mu = \min \{k + 1, 2(k + 1 - m)\}$. For each fixed t , terms in square brackets are bounded. Thus $E(e(t), \dot{e}(t))^{1/2} \rightarrow 0$ as $h \rightarrow 0$, which turn implies the convergence of $u^h(t)$ to $u(t)$ in H^m and $\dot{u}^h(t)$ to $\dot{u}(t)$ in L_2 . Because $\nu \leq \mu$, the rate of convergence is ν . However, note that the integral in (7.1.2) is $O(h^{\mu-1})$. Therefore, this estimate is only good for times no smaller than $O(h^{-m})^3$. Otherwise, the error is $O(h^\mu)$.

If you are interested in learning how to obtain results of this kind, consult [14].

Exercise 1. In the parabolic case, assume f , g , and h are zero. Obtain the following growth/decay estimates:

$$\begin{aligned}\| u(t) \|_0 &\leq \exp(-\lambda_1 t) \| u(0) \|_0 \\ \| u^h(t) \|_0 &\leq \exp(-\lambda_1^h t) \| u^h(0) \|_0\end{aligned}$$

Exercise 2. In the hyperbolic case, assume f , g , and h are zero. Establish the conservation of total energy:

$$\begin{aligned}E(u(t), \dot{u}(t)) &= E(u(0), \dot{u}(0)) \\ E(u^h(t), \dot{u}^h(t)) &= E(u^h(0), \dot{u}^h(0))\end{aligned}$$

REFERENCES

Section 7.1

1. I. Stakgold, *Boundary-Value Problems of Mathematical Physics*, Vol. II. New York: Macmillan, 1968.

Section 7.3

2. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*. Englewood Cliffs, N.J.: Prentice-Hall, 1973.
3. J. S. Archer, "Consistent Mass Matrix for Distributed Systems," *Proceedings of ASC* 89, ST4 (1963) 161–178.
4. T. J. R. Hughes, H. M. Hilber, and R. L. Taylor, "A Reduction Scheme for Problems of Structural Dynamics," *International Journal of Solids and Structures*, 12 (1976) 749–767.

³This statement needs to be suitably nondimensionalized to be made precise. For example, consider the axial motion of an elastic rod (see (7.2.46) and (7.2.47)). In this case the **bar wave velocity** $c = \sqrt{E/\rho}$, assumed constant. Note h/c has dimensions of time. Thus the rate of convergence ν is good up to times $\sim (h/c)^{-m}$.

5. I. Fried and D. S. Malkus, "Finite Element Mass Matrix Lumping by Numerical Integration Without Convergence Rate Loss," *International Journal of Solids and Structures*, 11 (1976) 461–466.
 6. E. Hinton, T. Rock, and O. C. Zienkiewicz, "A Note on Mass Lumping and Related Processes in the Finite Element Method," *Earthquake Engineering and Structural Dynamics*, 4 (1976), 245–249.
 7. P. Gresho, R. Lee, and R. Sani, "Advection-dominated Flows, with Emphasis on the Consequences of Mass Lumping," in *Finite Elements in Fluids*, Vol. 3, London: John Wiley, 1978, 335–350.
 8. G. L. Goudreau, "Evaluation of Discrete Methods for the Linear Dynamic Response of Elastic and Viscoelastic Solids," UC SESM Report 69-15, University of California. Berkeley, June 1970.
 9. H. M. Hilber, T. J. R. Hughes, and R. L. Taylor, "Mass Matrices for Improved Finite Element Eigenvalue Approximations," unpublished report, December 1975.
 10. T. J. R. Hughes and T. E. Tezduyar, "Stability and Accuracy Analysis of Some Fully Discrete Algorithms for the One-Dimensional Second-Order Wave Equation," *Computers and Structures*, 19 (1984), 665–668.
 11. R. L. Taylor and R. Iding, "Application of Extended Variational Principles to Finite Element Analysis," in *Variational Methods in Engineering*, Vol. I, Southampton University Press, 1973, 2.54–2.67.
 12. B. M. Irons, "Applications of a Theorem on Eigenvalues to Finite Element Problems." (CR/132/70), University of Wales, Department of Civil Engineering, Swansea, U.K.. 1970.
 13. T. J. R. Hughes, K. S. Pister, and R. L. Taylor, "Implicit-Explicit Finite Elements in Nonlinear Transient Analysis." *Computer Methods in Applied Mechanics and Engineering*, 17/18 (1979), 159–182.
- Appendix 7.1**
14. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*. Englewood Cliffs. N.J.: Prentice-Hall, 1973.

8

Algorithms for Parabolic Problems

8.1 ONE-STEP ALGORITHMS FOR THE SEMIDISCRETE HEAT EQUATION: GENERALIZED TRAPEZOIDAL METHOD

Recall that the semidiscrete heat equation can be written as

$$M\dot{d} + Kd = F \quad (8.1.1)$$

where M is the capacity matrix, K is the conductivity matrix, F is the heat supply vector, d is the temperature vector, and \dot{d} is the time (t) derivative of d . The matrices M and K are assumed symmetric, M is positive-definite, and K is positive-semidefinite (usually K is positive-definite too). The heat supply is a prescribed function of t . We write $F = F(t)$ for $t \in [0, T]$. Equation (8.1.1) is to be thought of as a coupled system of n_{eq} ordinary differential equations.

The initial-value problem consists of finding a function $d = d(t)$ satisfying (8.1.1), and the initial condition

$$d(0) = d_0 \quad (8.1.2)$$

where d_0 is given.

Perhaps the most well known and commonly used algorithms for solving (8.1.1) are members of the *generalized trapezoidal family of methods*, which consists of the following equations:

$$Mv_{n+1} + Kd_{n+1} = F_{n+1} \quad (8.1.3)$$

$$d_{n+1} = d_n + \Delta t v_{n+\alpha} \quad (8.1.4)$$

$$v_{n+\alpha} = (1 - \alpha)v_n + \alpha v_{n+1} \quad (8.1.5)$$

where d_n and v_n are the approximations to $d(t_n)$ and $\dot{d}(t_n)$, respectively; $F_{n+1} = F(t_{n+1})$; Δt is the time step, assumed constant for the time being; and α is a parameter, taken to be in the interval $[0, 1]$.

Some well-known members of the generalized trapezoidal family are identified below.

| α | Method |
|---------------|---|
| 0 | Forward differences; forward Euler |
| $\frac{1}{2}$ | Trapezoidal rule; midpoint rule; Crank-Nicolson |
| 1 | Backward differences; backward Euler |

So that the reader has a basic appreciation of the computations entailed by the algorithm, we will present next a brief overview of implementational considerations.

Implementation 1: v-form

The computational problem is to determine d_{n+1} and v_{n+1} given d_n and v_n . The procedure begins at $n = 0$ with d_0 known. The initial value, v_0 , may be determined from the time-discrete heat equation evaluated at $t = 0$:

$$Mv_0 = F_0 - Kd_0 \quad (8.1.6)$$

There are several ways of implementing the recursion relationship that takes us from step n to step $n + 1$. Let the *predictor value* of d_{n+1} be defined by

$$\tilde{d}_{n+1} = d_n + (1 - \alpha)\Delta t v_n \quad (8.1.7)$$

Note that (8.1.4) and (8.1.5) can be combined and expressed in terms of (8.1.7):

$$d_{n+1} = \tilde{d}_{n+1} + \alpha\Delta t v_{n+1} \quad (8.1.8)$$

Substituting this expression into (8.1.3) results in an equation that may be solved for v_{n+1} :

$$(M + \alpha\Delta t K)v_{n+1} = F_{n+1} - K\tilde{d}_{n+1} \quad (8.1.9)$$

Observe that the terms on the right-hand side of (8.1.9) are known. Once v_{n+1} is determined, (8.1.8) serves to define d_{n+1} , and so on.

Remarks

1. In the case of $\alpha = 0$, the method is said to be *explicit*. The attribute of explicit methods may be seen from (8.1.9) if M is assumed “lumped” (i.e., diagonal). In this case the solution may be advanced without the necessity of equation solving.

2. If $\alpha \neq 0$, the method is said to be *implicit*. In these cases a system of equations, with coefficient matrix $(M + \alpha\Delta t K)$, needs to be solved at each step to advance the solution. As long as Δt is constant, only one factorization is required.

3. The right-hand-side vector is formed one element at a time. Recall that the definition of F is given in this way (see (7.1.27) through (7.1.29)). The product

$$K \tilde{d}_{n+1} = \sum_{e=1}^{n_{el}} (k^e \tilde{d}_{n+1}^e) \quad (8.1.10)$$

where \tilde{d}_{n+1}^e contains zeros in degrees of freedom that correspond to prescribed boundary conditions. The effect of the boundary-condition terms is already accounted for in the definition of F [see (7.1.29)]. This aspect of the implementation warrants further discussion. We will postpone the details until we consider a general implementation of a class of hyperbolic and hyperbolic-parabolic algorithms in Chapter 9. The general case can be specialized to the present circumstances.

4. It is useful to note that the implementation manifested by (8.1.9) can easily be generalized to so-called implicit-explicit methods, where part of K is treated implicitly and part explicitly. The *only* required change is to replace K on the *left-hand side* of (8.1.9) with the part to be treated implicitly, say K' . This could be the assembly of a subset of elements, which leads to *implicit-explicit element mesh partitions* (see Hughes et al. [1–3]). It may be observed that the implementation is trivial. We shall put off a more detailed discussion until Chapter 9.

Implementation 2: *d*-form

Another possibility is to eliminate v_{n+1} from (8.1.3) via (8.1.8). Thus in place of (8.1.9) we have

$$\frac{1}{\alpha\Delta t} (M + \alpha\Delta t K) d_{n+1} = F_{n+1} + \frac{1}{\alpha\Delta t} M \tilde{d}_{n+1} \quad (8.1.11)$$

In this implementation, (8.1.11) is used to determine d_{n+1} and then v_{n+1} may be determined from (8.1.8), i.e.,

$$v_{n+1} = \frac{d_{n+1} - \tilde{d}_{n+1}}{\alpha\Delta t} \quad (8.1.12)$$

The advantage of this implementation occurs when M is diagonal. In this case the calculation of the right-hand side of (8.1.11) may be performed much more economically than the right-hand side of (8.1.9). The equation-solving burden is of course the same for (8.1.9) and (8.1.11).

Remark

Note that if $\alpha\Delta t \rightarrow \infty$ in (8.1.11), the equilibrium, or steady-state solution (i.e., one for which $\dot{\mathbf{d}} = \mathbf{0}$) is approached, viz.,

$$\mathbf{K}\mathbf{d}_{n+1} \rightarrow \mathbf{F}_{n+1} \quad (8.1.13)$$

This fact can be exploited in a transient analysis computer program if the equilibrium solution is desired. Just select a value of $\alpha\Delta t$ large enough so that (8.1.13) holds to desired precision.

Exercise 1. Derive an implementation in which the v_n 's are unnecessary. This results in a saving of computer storage. [Answer: $(\mathbf{M} + \alpha\Delta t \mathbf{K})\mathbf{d}_{n+1} = (\mathbf{M} - (1 - \alpha)\Delta t \mathbf{K})\mathbf{d}_n + \Delta t(\alpha\mathbf{F}_{n+1} + (1 - \alpha)\mathbf{F}_n)$.]

ANALYSIS OF THE GENERALIZED TRAPEZOIDAL METHOD

The primary requirement of the algorithms given in the last section is that they converge. We shall call an algorithm *convergent* if for t_n fixed and $\Delta t = t_n/n$, $\mathbf{d}_n \rightarrow \mathbf{d}(t_n)$ as $\Delta t \rightarrow 0$. To establish the convergence of an algorithm, two additional notions must be considered: *stability* and *consistency*. We shall show later on that once stability and consistency are verified, convergence is automatic. In addition, we shall be concerned with the *accuracy* of an algorithm, i.e., the rate of convergence as $\Delta t \rightarrow 0$, and allied topics such as the behavior of the (spurious) higher modes of the semidiscrete system. There are several techniques that can be employed to study the characteristics of an algorithm. In the present context the most revealing approach appears to be the "modal approach" (sometimes called spectral, or Fourier, analysis) in which the problem is decomposed into n_{eq} uncoupled scalar equations. It can be rigorously established that the behavior of the entire coupled system reduces to consideration of the individual modal equations that comprise it. Our first step in analyzing the family of algorithms introduced in Sec. 8.1 will be to perform the reduction to single-degree-of-freedom (SDOF) form.

8.2.1 Modal Reduction to SDOF Form

The essential property used in reducing to SDOF form is the orthogonality of the eigenvectors of the associated eigenvalue problem. Recall that

$$(\mathbf{K} - \lambda_l^h \mathbf{M})\boldsymbol{\psi}_l = \mathbf{0}, \quad l \in \{1, 2, \dots, n_{eq}\} \quad (8.2.1)$$

where

$$0 \leq \lambda_1^h \leq \lambda_2^h \leq \dots \leq \lambda_{n_{eq}}^h \quad (8.2.2)$$

and

$$\boldsymbol{\psi}_l^T \mathbf{M} \boldsymbol{\psi}_m = \delta_{lm} \quad (\text{orthonormality}) \quad (8.2.3)$$

where δ_{lm} is the Kronecker delta. Furthermore, the eigenvectors $\{\psi_l\}_{l=1}^{n_{eq}}$ constitute a *basis* for $\mathbb{R}^{n_{eq}}$, meaning that any element in $\mathbb{R}^{n_{eq}}$ can be written as a linear combination of the ψ_l 's. From the orthonormality property, it immediately follows that

$$\psi_l^T K \psi_m = \lambda_l^h \delta_{lm} \quad (\text{no sum}) \quad (8.2.4)$$

These properties will be used to decompose *both* the semidiscrete heat equation and the generalized trapezoidal algorithm.

Semidiscrete heat equation. Let

$$\mathbf{d}(t) = \sum_{m=1}^{n_{eq}} d_{(m)}(t) \psi_m \quad (8.2.5)$$

from which it follows that

$$\dot{\mathbf{d}}(t) = \sum_{m=1}^{n_{eq}} \dot{d}_{(m)}(t) \psi_m \quad (8.2.6)$$

The scalar-valued functions $d_{(m)}(t)$ (*Fourier coefficients*) are obtained by premultiplying (8.2.5) by $\psi_l^T M$ and invoking the orthonormality property. The result is

$$d_{(l)}(t) = \psi_l^T M \mathbf{d}(t) \quad (8.2.7)$$

(We have included the subscript in parentheses to avoid notational confusion.) The coefficients in (8.2.6) are obtained by differentiating (8.2.7):

$$\dot{d}_{(l)}(t) = \psi_l^T M \dot{\mathbf{d}}(t) \quad (8.2.8)$$

Employing (8.2.5) and (8.2.6) in (8.1.1) and premultiplying by ψ_l^T yields

$$\sum_{m=1}^{n_{eq}} (\dot{d}_{(m)} \psi_l^T M \psi_m + d_{(m)} \psi_l^T K \psi_m) = \psi_l^T F \quad (8.2.9)$$

from which it follows that

$$\dot{d}_{(l)} + \lambda_l^h d_{(l)} = F_{(l)} \quad (8.2.10)$$

the l th-modal equation, where $F_{(l)}(t) = \psi_l^T F(t)$.

The initial condition for (8.2.10) is obtained by premultiplying (8.1.2) by $\psi_l^T M$:

$$\begin{aligned} d_{(l)}(0) &= \psi_l^T M \mathbf{d}(0) \\ &= \psi_l^T M d_0 \\ &\stackrel{\text{def}}{=} d_{0(l)} \end{aligned} \quad (8.2.11)$$

Solution of (8.2.10) for each mode allows us to construct the solution \mathbf{d} of the original problem via (8.2.5).

To simplify the subsequent writing further we shall omit the l th-modal subscript in (8.2.10) and (8.2.11). Thus our typical modal initial-value problem consists of the following equations:

$$\dot{\mathbf{d}} + \lambda^h \mathbf{d} = F, \quad t \in [0, T] \quad \left. \right\} \quad (\text{SDOF model problem}) \quad (8.2.12)$$

$$d(0) = d_0 \quad (8.2.13)$$

Generalized trapezoidal algorithm. The procedure for decomposing the generalized trapezoidal algorithm is similar to that for the semidiscrete equation. The main results are collected here:

$$\mathbf{d}_n = \sum_{m=1}^{n_{eq}} d_{n(m)} \Psi_m \quad (8.2.14)$$

$$\mathbf{d}_{n+1} = \sum_{m=1}^{n_{eq}} d_{n+1(m)} \Psi_m \quad (8.2.15)$$

$$d_{n(l)} = \Psi_l^T M \mathbf{d}_n \quad (8.2.16)$$

$$d_{n+1(l)} = \Psi_l^T M \mathbf{d}_{n+1} \quad (8.2.17)$$

$$\begin{aligned} & \sum_{m=1}^{n_{eq}} [d_{n+1(m)} \Psi_l^T (M + \alpha \Delta t K) \Psi_m \\ & - d_{n(m)} \Psi_l^T (M - (1 - \alpha) \Delta t K) \Psi_m] = \Delta t \Psi_l^T F_{n+\alpha} \quad (8.2.18) \\ & F_{n+\alpha} = (1 - \alpha) F_n + \alpha F_{n+1} \quad (8.2.19) \\ & (1 + \alpha \Delta t \lambda_l^h) d_{n+1(l)} = (1 - (1 - \alpha) \Delta t \lambda_l^h) d_{n(l)} + \Delta t F_{n+\alpha(l)} \quad (8.2.20) \end{aligned}$$

The initial value, $d_{0(l)}$, is defined by (8.2.11). As before, we will omit the l th-modal subscript in (8.2.20). Thus we may write

$$\left. \begin{aligned} (1 + \alpha \Delta t \lambda^h) d_{n+1} &= (1 - (1 - \alpha) \Delta t \lambda^h) d_n + \Delta t F_{n+\alpha} \\ d_0 \text{ given} & \end{aligned} \right\} \quad \begin{array}{l} (\textit{temporally discretized}) \\ \textit{SDOF model problem} \end{array} \quad (8.2.21)$$

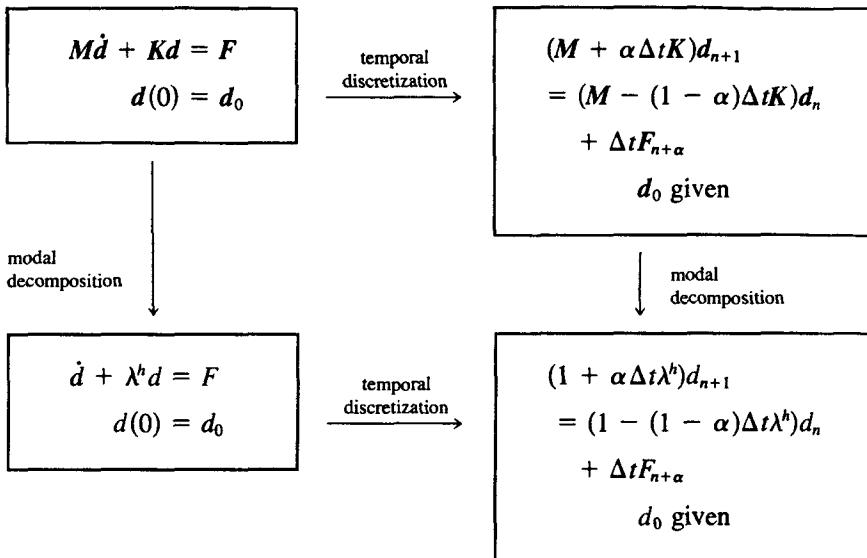
Remarks

1. The convergence of \mathbf{d}_n to $\mathbf{d}(t_n)$ can be established by showing the Fourier coefficients converge [i.e., $d_{n(l)}$ converges to $d_{(l)}(t_n)$ for each $l \in \{1, 2, \dots, n_{eq}\}$]. The proof of this goes as follows. Let $\mathbf{e}(t_n) = \mathbf{d}_n - \mathbf{d}(t_n)$ denote the error in \mathbf{d}_n , and let $e_{(l)}(t_n) = d_{n(l)} - d_{(l)}(t_n)$ denote the l th-Fourier component of $\mathbf{e}(t_n)$. Then

$$\begin{aligned} \mathbf{e}(t_n)^T M \mathbf{e}(t_n) &= \sum_{l, m=1}^{n_{eq}} (e_{(l)}(t_n) \Psi_l)^T M (e_{(m)}(t_n) \Psi_m) \\ &= \sum_{l, m=2}^{n_{eq}} e_{(l)}(t_n) e_{(m)}(t_n) \Psi_l^T M \Psi_m \\ &= \sum_{l, m=1}^{n_{eq}} e_{(l)}(t_n) e_{(m)}(t_n) \delta_{lm} \quad (\text{orthonormality}) \\ &= \sum_{l=1}^{n_{eq}} (e_{(l)}(t_n))^2 \quad (8.2.22) \end{aligned}$$

and so $\mathbf{e}(t_n)^T \mathbf{M} \mathbf{e}(t_n) \rightarrow 0$ if and only if $e_{(l)}(t_n) \rightarrow 0$ for each $l \in \{1, 2, \dots, n_{eq}\}$. Because \mathbf{M} is assumed positive-definite, $\mathbf{e}(t_n)^T \mathbf{M} \mathbf{e}(t_n) \rightarrow 0$ if and only if $\mathbf{e}(t_n) \rightarrow 0$. Consequently, we need consider only the SDOF problems in subsequent discussion.

2. Note that directly applying the generalized trapezoidal method to (8.2.12) also results in (8.2.20). This fact is depicted in the following commutative diagram:



8.2.2 Stability

To motivate the appropriate notion of stability for the case under consideration, we shall investigate the behavior of the homogeneous model equation

$$\dot{d} + \lambda^h d = 0 \quad (8.2.23)$$

This is a first-order ordinary differential equation, which can be easily solved. The solution at time t_{n+1} for initial value $d(t_n)$, $t_{n+1} > t_n$, is

$$d(t_{n+1}) = \exp(-\lambda^h(t_{n+1} - t_n))d(t_n) \quad (8.2.24)$$

from which it follows that

$$\left. \begin{aligned} |d(t_{n+1})| &< |d(t_n)|, & \lambda^h > 0 \\ d(t_{n+1}) &= d(t_n), & \lambda^h = 0 \end{aligned} \right\} \quad (8.2.25)$$

The conditions in (8.2.25) are what we wish to mimic in the temporally discrete case.

The homogeneous temporally discrete model equation is

$$(1 + \alpha \Delta t \lambda^h) d_{n+1} = (1 - (1 - \alpha) \Delta t \lambda^h) d_n \quad (8.2.26)$$

Noting that $(1 + \alpha \Delta t \lambda^h) > 0$ for all allowable values of the parameters, we can write (8.2.26) as

$$d_{n+1} = A d_n \quad (8.2.27)$$

where $A = (1 - (1 - \alpha) \Delta t \lambda^h) / (1 + \alpha \Delta t \lambda^h)$ is the *amplification factor*.

Our stability requirements will be that

$$\left. \begin{array}{l} |d_{n+1}| < |d_n|, \quad \lambda^h > 0 \\ d_{n+1} = d_n, \quad \lambda^h = 0 \end{array} \right\} \quad (8.2.28)$$

From the definition of A , the second of (8.2.28) is automatic. The first condition is equivalent to insisting that

$$|A| < 1 \quad (8.2.29)$$

for $\lambda^h > 0$. To determine the restrictions imposed upon α , Δt , and λ^h , it is convenient to rewrite (8.2.29) as $-1 < A < 1$, viz.,

$$\boxed{-1 < \frac{(1 - (1 - \alpha) \Delta t \lambda^h)}{(1 + \alpha \Delta t \lambda^h)} < 1} \quad (8.2.30)$$

The right-hand inequality is satisfied for all allowable values of the parameters, and the left-hand inequality is satisfied whenever $\alpha \geq \frac{1}{2}$. However, if $\alpha < \frac{1}{2}$, the left-hand inequality requires $\lambda^h \Delta t < 2/(1 - 2\alpha)$. For a given λ^h this imposes an upper bound on the size of the allowable time step. The greater the λ^h , the smaller the time step required.

Remark

An algorithm for which stability imposes a time step restriction is called *conditionally stable*. An algorithm for which there is no time step restriction imposed by stability is called *unconditionally stable*.

The significance of the stability concept introduced may be seen from the following example:

Example

Note that the solution of (8.2.27) may be written

$$d_n = A^n d_0 \quad (8.2.31)$$

If d_n is to behave like the solution of (8.2.23) (i.e., decay), A^n must converge (i.e., as $n \rightarrow \infty$, $A^n \rightarrow 0$). If $|A| < 1$, clearly this will be the case. On the other hand, if

$|A| > 1$, growth will occur. Even if the value of $|A|$ is only slightly larger than 1, disastrous growth can occur. To see what can happen, consider the numerical data in Table 8.2.1. The necessity of keeping $|A| < 1$ should be clearly apparent, or else virtually unbounded errors will enter a computation.

TABLE 8.2.1 A^n for Various Values of A and n

| $A \backslash n$ | 100 | 1000 |
|------------------|-----------------------|------------------------|
| .99 | .37 | 4.32×10^{-5} |
| 1.01 | 2.70 | 2.09×10^4 |
| .9 | 2.66×10^{-5} | 1.75×10^{-46} |
| 1.1 | 1.39×10^4 | 2.47×10^{41} |

Remarks

1. In the conditionally stable case, the stability condition $\Delta t < 2/[(1 - 2\alpha)\lambda^h]$ must hold for all modes (i.e., all $\lambda_l^h, l \in \{1, 2, \dots, n_{eq}\}$) in the system. The greatest λ_l^h , namely, $\lambda_{n_{eq}}^h$, imposes the most stringent restriction upon the time step (i.e., $\Delta t < 2/[(1 - 2\alpha)\lambda_{n_{eq}}^h]$). In fact, for the heat conduction problem, it can be shown that $\lambda_{n_{eq}}^h = O(h^{-2})$, where h is the mesh parameter, and thus the critical time step must satisfy $\Delta t < \text{constant} \cdot h^2$. In a large system of equations (i.e., $n_{eq} \gg 1$), this condition is a severe constraint; thus unconditionally stable algorithms are generally preferred.

2. In Fig. 8.2.1 the behavior of A as a function of $\lambda^h \Delta t$ is depicted for several values of α . The value of $\lambda^h \Delta t$ for which $A = 0$ is called the *oscillation limit* because for greater values, the sign of A^n changes from step to step. For the unconditionally stable algorithms (i.e., ones for which $\alpha \geq \frac{1}{2}$), the asymptotic value of the amplification factor satisfies $|A_\infty| \leq 1$. Thus for all fixed $\lambda^h \Delta t$, $|A| < 1$ and all spurious high modal components decay. However, for $\alpha = \frac{1}{2}$ (or very near $\frac{1}{2}$) and $\lambda^h \Delta t \gg 1$, $A \approx -1$, and thus high modal components will behave like $(-1)^n$. This "sawtooth" pattern in time manifests itself frequently in computations. In reporting data for a calculation in which this is the case, these spurious higher modes may be filtered out by reporting the step-to-step averages, $(d_{n+1} + d_n)/2$, because $(-1)^{n+1} + (-1)^n = 0$.

Summary: Stability for the generalized trapezoidal methods

$$\text{Amplification factor: } A = \frac{1 - (1 - \alpha)\Delta t \lambda^h}{1 + \alpha \Delta t \lambda^h}$$

$$\text{Stability requirement: } |A| < 1 \text{ for } \lambda^h = \lambda_{n_{eq}}^h \quad (= \text{maximum eigenvalue})$$

$$\text{Unconditional stability: } \alpha \geq \frac{1}{2}$$

$$\text{Conditional stability: } \alpha < \frac{1}{2}, \quad \Delta t < \frac{2}{(1 - 2\alpha)\lambda_{n_{eq}}^h}$$

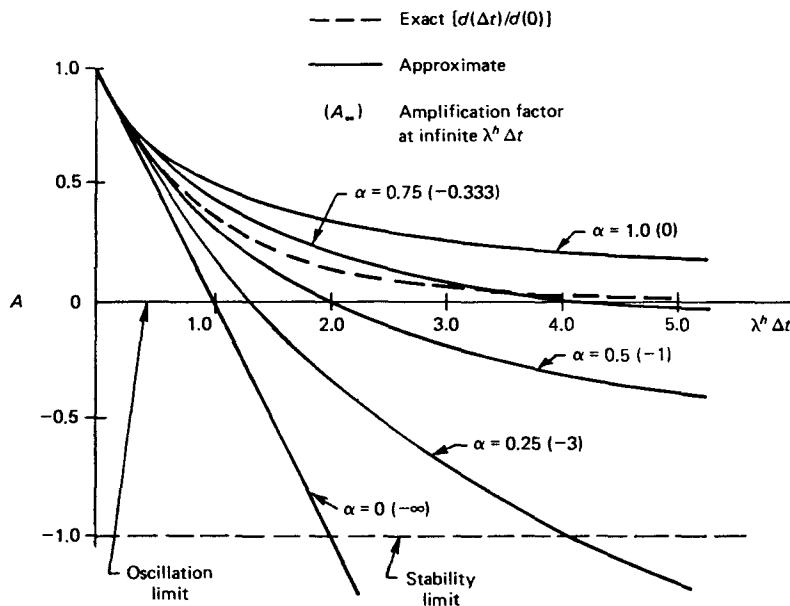


Figure 8.2.1 Amplification factor for typical one-step methods.

8.2.3 Convergence

The temporally discrete model problem may be written in the form

$$d_{n+1} - Ad_n - L_n = 0 \quad (8.2.32)$$

where the *load* $L_n = \Delta t F_{n+\alpha}/(1 + \alpha \Delta t \lambda^h)$. If we replace d_n and d_{n+1} in the left-hand side of (8.2.32) by the corresponding exact values, we obtain an expression of the form

$$d(t_{n+1}) - Ad(t_n) - L_n = \Delta t \cdot \tau(t_n) \quad (8.2.33)$$

where $\tau(t_n)$ is called the *local truncation error*. If $|\tau(t)| \leq c \Delta t^k$, for all $t \in [0, T]$, where c is a constant independent of Δt , and $k > 0$, the algorithm defined by (8.2.32) is called *consistent*; k is called the *order of accuracy* or *rate of convergence*.

Proposition. The generalized trapezoidal methods are consistent, and furthermore $k = 1$ for all $\alpha \in [0, 1]$, except $\alpha = \frac{1}{2}$, in which case $k = 2$.

Proof. Expand $d(t_{n+1})$ and $d(t_n)$ about $t_{n+\alpha}$ in finite Taylor expansions and use the model equation to eliminate t -derivatives of $d(t_{n+\alpha})$:

$$d(t_{n+1}) = d(t_{n+\alpha}) + (1 - \alpha)\Delta t \dot{d}(t_{n+\alpha}) + \frac{(1 - \alpha)\Delta t^2}{2} \ddot{d}(t_{n+\alpha}) + \frac{(1 - \alpha)\Delta t^3}{3!} \dddot{d}(t_{n+\alpha}) + O(\Delta t^4)$$

$$d(t_n) = d(t_{n+\alpha}) + (-\alpha\Delta t) \dot{d}(t_{n+\alpha}) + \frac{(-\alpha\Delta t)^2}{2} \ddot{d}(t_{n+\alpha}) + \frac{(-\alpha\Delta t)^3}{3!} \dddot{d}(t_{n+\alpha}) + O(\Delta t^4)$$

$$\begin{aligned} & \Delta t(1 + \alpha\Delta t\lambda^h) \tau(t_n) \\ &= (1 + \alpha\Delta t\lambda^h) d(t_{n+1}) - (1 - (1 - \alpha)\Delta t\lambda^h) d(t_n) - \Delta t F_{n+\alpha} \\ &= \{(1 + \alpha\Delta t\lambda^h) - (1 - (1 - \alpha)\Delta t\lambda^h)\} d(t_{n+\alpha}) \\ &+ \{(1 + \alpha\Delta t\lambda^h)(1 - \alpha)\Delta t - (1 - (1 - \alpha)\Delta t\lambda^h)(-\alpha\Delta t)\} \dot{d}(t_{n+\alpha}) \\ &+ \left\{ (1 + \alpha\Delta t\lambda^h) \frac{(1 - \alpha)\Delta t^2}{2} - (1 - (1 - \alpha)\Delta t\lambda^h) \frac{(-\alpha\Delta t)^2}{2} \right\} \ddot{d}(t_{n+\alpha}) \\ &+ \left\{ (1 + \alpha\Delta t\lambda^h) \frac{(1 - \alpha)\Delta t^3}{3!} - (1 - (1 - \alpha)\Delta t\lambda^h) \frac{(-\alpha\Delta t)^3}{3!} \right\} \dddot{d}(t_{n+\alpha}) \\ &- \Delta t \left([\alpha + (1 - \alpha)] F(t_{n+\alpha}) + [\alpha(1 - \alpha)\Delta t + (1 - \alpha)(-\alpha\Delta t)] \dot{F}(t_{n+\alpha}) \right. \\ &+ \left. \left\{ \alpha \frac{(1 - \alpha)\Delta t^2}{2} + (1 - \alpha) \frac{(-\alpha\Delta t)^2}{2} \right\} \ddot{F}(t_{n+\alpha}) \right. \\ &+ \left. \left\{ \alpha \frac{(1 - \alpha)\Delta t}{3!} + (1 - \alpha) \frac{(-\alpha\Delta t)^3}{3!} \right\} \dddot{F}(t_{n+\alpha}) \right) + O(\Delta t^4) \end{aligned}$$

Using $\dot{d} + \lambda^h d = F$ and time derivatives of same, it can be shown that

$$\tau = (1 - 2\alpha)O(\Delta t^1) + O(\Delta t^2)$$

(Fill in the remaining details as an exercise.) ■

Remark

Thus the trapezoidal rule ($\alpha = \frac{1}{2}$) is the only member of the family of methods that is second-order accurate.

Theorem. Consider equations (8.2.32) and (8.2.33). Let t_n be fixed (n , and consequently Δt , are allowed to vary), and assume the following conditions hold:

- i. $|A| \leq 1$ (stability)
- ii. $|\tau(t)| \leq c\Delta t^k, \quad t \in [0, T], \quad k > 0$ (consistency)

Then $e(t_n) \rightarrow 0$ as $\Delta t \rightarrow 0$.

Proof. Subtract (8.2.33) from (8.2.32):

$$e(t_{n+1}) = Ae(t_n) - \Delta t \cdot \tau(t_n) \quad \text{error equation} \quad (8.2.34)$$

Replace $e(t_n)$ on the right-hand side by the error equation for the previous step, i.e.,

$$e(t_n) = Ae(t_{n-1}) - \Delta t \cdot \tau(t_{n-1})$$

to obtain

$$e(t_{n+1}) = A^2 e(t_{n-1}) - \Delta t A \tau(t_{n-1}) - \Delta t \cdot \tau(t_n)$$

Now repeat this procedure to eliminate $e(t_{n-1})$ from the right-hand side, viz.,

$$e(t_{n+1}) = A^3 e(t_{n-2}) - \Delta t A^2 \tau(t_{n-2}) - \Delta t A \tau(t_{n-1}) - \Delta t \cdot \tau(t_n)$$

and so on. The final result is

$$e(t_{n+1}) = A^{n+1} e(0) - \Delta t \sum_{i=0}^n A^i \tau(t_{n-i})$$

The first term on the right vanishes since $e(0) = d_0 - d(0) = 0$. Taking absolute values evaluated at t_n instead of t_{n+1} and using some elementary facts:

$$\begin{aligned} |e(t_n)| &= \Delta t \left| \sum_{i=0}^{n-1} A^i \tau(t_{n-1-i}) \right| \\ &\leq \Delta t \sum_{i=0}^{n-1} |A|^i |\tau(t_{n-1-i})| \\ &\leq \Delta t \sum_{i=0}^{n-1} |\tau(t_{n-1-i})| \quad (\text{stability}) \\ &\leq t_n \max |\tau(t)| \quad t \in [0, T] \\ &\leq t_n c \Delta t^k \quad (\text{consistency}) \end{aligned}$$

Therefore $e(t_n) \rightarrow 0$ as $\Delta t \rightarrow 0$, and furthermore the rate of convergence is k (i.e., $e(t_n) = O(\Delta t^k)$). ■

Remarks

1. This result establishes the convergence of the generalized trapezoidal methods. The maximal rate of convergence is 2 and is attained by the trapezoidal rule.
2. This theorem is a particular example of perhaps the most celebrated theorem in numerical analysis, the Lax equivalence theorem, which may be stated as “consistency plus stability is necessary and sufficient for convergence.”

Exercise 1. Suppose that in (8.2.32) $F_{n+\alpha}$ is replaced by an approximation, $F_{n+\alpha}^{\Delta t}$, which

satisfies $|F_{n+\alpha} - F_{n+\alpha}^{\Delta t}| \leq \text{constant} \cdot \Delta t^q$. What condition must q satisfy to retain k th-order accuracy?

8.2.4 An Alternative Approach to Stability: The Energy Method

An alternative approach to stability is the energy method [4]. In the present circumstances the results are identical to those obtained by the modal approach. However, the energy method is often applicable to situations in which the modal approach is unsuccessful. We shall illustrate the energy method on the generalized trapezoidal family of algorithms.

It simplifies the analysis if the following discrete operators are introduced:

$$[\mathbf{d}_n] = \mathbf{d}_{n+1} - \mathbf{d}_n \quad (\text{undivided forward-difference operator}) \quad (8.2.35)$$

$$\langle \mathbf{d}_n \rangle = \frac{\mathbf{d}_{n+1} + \mathbf{d}_n}{2} \quad (\text{mean-value operator}) \quad (8.2.36)$$

Note that

$$\langle \mathbf{d}_n \rangle^T \mathbf{M} [\mathbf{d}_n] = \frac{1}{2} [\mathbf{d}_n^T \mathbf{M} \mathbf{d}_n] \quad (8.2.37)$$

Equation (8.1.4) may be written as

$$\begin{aligned} [\mathbf{d}_n] &= \Delta t \mathbf{v}_{n+\alpha} \\ &= \Delta t (\langle \mathbf{v}_n \rangle + (\alpha - \frac{1}{2}) [\mathbf{v}_n]) \end{aligned} \quad (8.2.38)$$

By combining the temporally discrete heat equations at steps n and $n + 1$, we arrive at

$$\mathbf{M} \mathbf{v}_{n+\alpha} + \mathbf{K} \mathbf{d}_{n+\alpha} = \mathbf{F}_{n+\alpha} \quad (8.2.39)$$

This may also be written as

$$\frac{1}{\Delta t} \mathbf{M} [\mathbf{d}_n] + \mathbf{K} \mathbf{d}_{n+\alpha} = \mathbf{F}_{n+\alpha} \quad (8.2.40)$$

Premultiplying (8.2.40) by $\langle \mathbf{d}_n \rangle^T$ leads to

$$\begin{aligned} \frac{1}{2\Delta t} [\mathbf{d}_n^T \mathbf{M} \mathbf{d}_n] &= -\langle \mathbf{d}_n \rangle^T \mathbf{K} \mathbf{d}_{n+\alpha} + \langle \mathbf{d}_n \rangle^T \mathbf{F}_{n+\alpha} \\ &= -\langle \mathbf{d}_n \rangle^T \mathbf{K} \left(\langle \mathbf{d}_n \rangle + \left(\alpha - \frac{1}{2} \right) [\mathbf{d}_n] \right) + \langle \mathbf{d}_n \rangle^T \mathbf{F}_{n+\alpha} \\ &= -\langle \mathbf{d}_n \rangle^T \mathbf{K} \langle \mathbf{d}_n \rangle - \frac{(\alpha - \frac{1}{2})}{2} [\mathbf{d}_n^T \mathbf{K} \mathbf{d}_n] + \langle \mathbf{d}_n \rangle^T \mathbf{F}_{n+\alpha} \end{aligned} \quad (8.2.41)$$

Rearranging (8.2.41) results in

$$\boxed{d_{n+1}^T A d_{n+1} = d_n^T A d_n - 2\Delta t \langle d_n \rangle^T K \langle d_n \rangle + 2\Delta t \langle d_n \rangle^T F_{n+\alpha}} \quad (8.2.42)$$

where

$$A = M + \left(\alpha - \frac{1}{2} \right) \Delta t K \quad (8.2.43)$$

By positive-semidefiniteness,

$$d_{n+1}^T A d_{n+1} \leq d_n^T A d_n + 2\Delta t \langle d_n \rangle^T F_{n+\alpha} \quad (8.2.44)$$

For purposes of stability, we may consider the *homogeneous case*:

$$d_{n+1}^T A d_{n+1} \leq d_n^T A d_n \quad (8.2.45)$$

From which we may conclude

$$d_n^T A d_n \leq d_0^T A d_0, \quad \text{for all } n \quad (8.2.46)$$

As long as A is positive-definite d_n will not amplify. We take this as the definition of *energy stability*. Let c be an arbitrary vector. We may expand c in terms of the eigenvectors:

$$c = \sum_{l=1}^{n_{eq}} c_{(l)} \psi_l \quad (8.2.47)$$

Therefore,

$$c^T A c = \sum_{l=1}^{n_{eq}} \left(1 + \left(\alpha - \frac{1}{2} \right) \Delta t \lambda_l^h \right) c_{(l)}^2 \quad (8.2.48)$$

and so

$$1 + \left(\alpha - \frac{1}{2} \right) \Delta t \lambda_l^h > 0, \quad \text{for all } l = 1, 2, \dots, n_{eq} \quad (8.2.49)$$

is the stability condition. This condition is satisfied if $\alpha \geq \frac{1}{2}$, or if

$$\lambda_{n_{eq}}^h \Delta t < \frac{2}{1 - 2\alpha} \quad (8.2.50)$$

These are the same stability conditions arrived at previously.

Equation (8.2.42) is a discrete analog of the growth/decay identity satisfied by the exact solution $d(t)$ of the semidiscrete heat equation. This can be seen as follows. Multiply (8.1.1) by d^T and integrate over the time interval $[t_n, t_{n+1}]$:

$$\begin{aligned} 0 &= \int_{t_n}^{t_{n+1}} d^T (M \dot{d} + Kd - F) dt \\ &= \int_{t_n}^{t_{n+1}} \left(\frac{1}{2} \frac{d}{dt} (d^T M d) + d^T K d - d^T F \right) dt \end{aligned} \quad (8.2.51)$$

from which it follows that (cf. (8.2.42)):

$$\mathbf{d}(t_{n+1})^T \mathbf{M} \mathbf{d}(t_{n+1}) = \mathbf{d}(t_n)^T \mathbf{M} \mathbf{d}(t_n) - 2 \int_{t_n}^{t_{n+1}} (\mathbf{d}^T \mathbf{K} \mathbf{d} - \mathbf{d}^T \mathbf{F}) dt$$

(8.2.52)

8.2.5 Additional Exercises

Exercise 2. Consider the following one-parameter (α) family of *predictor-corrector algorithms*:

$$\begin{aligned} \mathbf{M} \mathbf{v}_{n+1} + \mathbf{K} \tilde{\mathbf{d}}_{n+1} &= \mathbf{F}_{n+1} \\ \tilde{\mathbf{d}}_{n+1} &= \mathbf{d}_n + (1 - \alpha) \Delta t \mathbf{v}_n && \text{(predictor)} \\ \mathbf{d}_{n+1} &= \tilde{\mathbf{d}}_{n+1} + \alpha \Delta t \mathbf{v}_{n+1} && \text{(corrector)} \end{aligned}$$

Assume $\alpha \in [0, 1]$.

- i. Determine an expression for the amplification factor A and plot A versus $\lambda^h \Delta t$.
- ii. Determine under what circumstances the algorithms are stable.
- iii. Obtain an expression for the local truncation error.
- iv. Determine the rate of convergence.
- v. If \mathbf{M} is diagonal, is the algorithm implicit or explicit?

Answers: The modal equations are:

$$\mathbf{v}_{n+1} + \lambda^h \tilde{\mathbf{d}}_{n+1} = \mathbf{F}_{n+1} \quad (a)$$

$$\tilde{\mathbf{d}}_{n+1} = \mathbf{d}_n + (1 - \alpha) \Delta t \mathbf{v}_n \quad (b)$$

$$\mathbf{d}_{n+1} = \tilde{\mathbf{d}}_{n+1} + \alpha \Delta t \mathbf{v}_{n+1} \quad (c)$$

$$(a) \text{ and } (c) \text{ imply } (1 - \alpha \Delta t \lambda^h) \mathbf{v}_{n+1} + \lambda^h \mathbf{d}_{n+1} = \mathbf{F}_{n+1} \quad (d)$$

$$\text{Likewise at } n: (1 - \alpha \Delta t \lambda^h) \mathbf{v}_n + \lambda^h \mathbf{d}_n = \mathbf{F}_n \quad (e)$$

$$(b) \text{ and } (c) \text{ imply } \mathbf{d}_{n+1} = \mathbf{d}_n + (1 - \alpha) \Delta t \mathbf{v}_n + \alpha \Delta t \mathbf{v}_{n+1} \quad (f)$$

Multiply (f) by $(1 - \alpha \Delta t \lambda^h)$:

$$\begin{aligned} (1 - \alpha \Delta t \lambda^h) \mathbf{d}_{n+1} &= (1 - \alpha \Delta t \lambda^h) \mathbf{d}_n + \Delta t (1 - \alpha) \underbrace{(1 - \alpha \Delta t \lambda^h) \mathbf{v}_n}_{(e)} + \alpha \Delta t \underbrace{(1 - \alpha \Delta t \lambda^h) \mathbf{v}_{n+1}}_{(d)} \\ (1 - \alpha \Delta t \lambda^h) \mathbf{d}_{n+1} &= (1 - \alpha \Delta t \lambda^h) \mathbf{d}_n + \Delta t (1 - \alpha) [\mathbf{F}_n - \lambda^h \mathbf{d}_n] + \alpha \Delta t [\mathbf{F}_{n+1} - \lambda^h \mathbf{d}_{n+1}] \\ \mathbf{d}_{n+1} &= (1 - \Delta t \lambda^h) \mathbf{d}_n + \Delta t \underbrace{[(1 - \alpha) \mathbf{F}_n + \alpha \mathbf{F}_{n+1}]}_{\mathbf{F}_{n+\alpha}} \end{aligned}$$

- i. $A = 1 - \Delta t \lambda'$
- ii. $|A| < 1$ implies $\Delta t \lambda' < 2$
- iii. $\Delta t \tau(t_n) = d(t_{n-1}) - (1 - \Delta t \lambda') d(t_n) - \Delta t F_{n+\alpha}$
 $= \Delta t \underbrace{(\dot{d}(t_n) + \lambda' d(t_n) - F(t_n))}_0 + \Delta t^2 \left(\frac{\ddot{d}(t_n)}{2} - \alpha \dot{F}(t_n) \right) + O(\Delta t^3)$
- Therefore $|\tau(n)| \leq c \Delta t$
- iv. $k = 1$
- v. Explicit

Exercise 3. Consider the following *fractional-step* algorithm for the homogeneous model heat equation:

$$\frac{d_{n+1/2} - d_n}{(\Delta t/2)} + \lambda^h d_n = 0 \quad (\text{a})$$

$$\frac{d_{n-1} - d_{n+1/2}}{(\Delta t/2)} + \lambda^h d_{n+1} = 0 \quad (\text{b})$$

- i. Obtain an expression for the amplification factor.
- ii. Determine the conditions under which the algorithm is stable.
- iii. What is the rate of convergence of this algorithm with respect to solutions of $\dot{d} + \lambda^h d = 0$?

Answers:

- i. Solve (a) for $d_{n+1/2}$ and eliminate from (b) to arrive at

$$d_{n-1} = Ad_n; \quad A = \frac{1 - \Delta t \lambda'^2/2}{1 + \Delta t \lambda'^2/2}$$

Observe that A is identical to the one obtained for $\alpha = \frac{1}{2}$ (i.e., trapezoidal rule) in the generalized trapezoidal family.

- ii. Unconditionally stable
- iii. $k = 2$

Exercise 4. Consider the homogeneous semidiscrete heat equation,

$$M\dot{d} + Kd = 0$$

and the following algorithm:

$$d_{n+1} = d_n + \Delta t v_n + \frac{\Delta t^2}{2} a_n$$

$$Mv_{n+1} - Kd_{n+1} = 0$$

$$Ma_{n+1} - Kv_{n+1} = 0$$

- i. Determine the amplification factor for this algorithm and plot it versus $\lambda^h \Delta t$. Hint:

$$\Psi_m^T M (M^{-1} K)^2 \Psi_m = \lambda_m^2.$$

- ii. Determine the stability condition.
- iii. Assuming M is diagonal, is this algorithm implicit or explicit? Justify your answer
- iv. Determine an expression for the local truncation error. Hint: Expand about t_n .
- v. Based on the result of part (iv), determine the order of accuracy of this algorithm.

Answers:

- i. $A = 1 - \lambda^h + (\Delta t \lambda^h)^2 / 2$
- ii. $\Delta t < 2 / \lambda_{\text{eq}}^h$
- iii. Explicit
- iv. $\tau(t_n) = \Delta t^2 \ddot{d}(\bar{t}) / 6$, where $\bar{t} \in [t_n, t_{n+1}]$
- v. $k = 2$

Exercise 5. Modal decomposition of the algorithm

$$d_{n+1} = d_n + \frac{\Delta t}{2} (v_n + v_{n+1}) + \frac{\Delta t^2}{12} (a_n - a_{n+1})$$

$$Mv_{n+1} = -Kd_{n+1} + F_{n+1}$$

$$Ma_{n+1} = -Kv_{n+1} + \dot{F}_{n+1}$$

leads to the difference equation

$$\begin{aligned} \left(1 + \frac{\Delta t \lambda^h}{2} + \frac{(\Delta t \lambda^h)^2}{12}\right) d_{n+1} &= \left(1 - \frac{\Delta t \lambda^h}{2} + \frac{(\Delta t \lambda^h)^2}{12}\right) d_n + \frac{\Delta t}{2} (F_n + F_{n+1}) \\ &\quad - \frac{\Delta t^2 \lambda^h}{12} (F_n - F_{n+1}) + \frac{\Delta t^2}{12} (\dot{F}_n - \dot{F}_{n+1}) \end{aligned} \quad (a)$$

- i. Plot the amplification factor A versus $\Delta t \lambda^h$. Comment on the stability of the algorithm. [Answer: Unconditionally stable.]
- ii. Determine the order of accuracy. [Answer: $k = 4$. This one is a lot of work!]

Remark

The matrix version of (a), which would be implemented on the computer to obtain d_{n+1} , involves the product $KM^{-1}K$ in the coefficient matrix. This product has a profile structure involving approximately twice the number of terms as the original K if M is diagonal and is full if M is consistent. Thus the algorithm involves considerably more storage and computations than does the generalized trapezoidal method, for example. It is also inconvenient to obtain the time derivatives of F required at each step.

Exercise 6. The SDOF model problem under consideration consists of

$$\dot{d} + (\lambda^h + \tilde{\lambda}^h)d = F$$

$$d(0) = d_0$$

Assume

$$\lambda^h > 0, \quad \tilde{\lambda}^h > 0$$

Consider the following one-parameter family of algorithms:

$$\begin{aligned} v_{n+1} + \lambda^h d_{n+1} + \tilde{\lambda}^h \tilde{d}_{n+1} &= F_{n+1} \\ \tilde{d}_{n+1} &= d_n + (1 - \alpha)\Delta t v_n \\ d_{n+1} &= \tilde{d}_{n+1} + \alpha \Delta t v_{n+1} \end{aligned}$$

Assume $\alpha \in [0, 1]$.

- i. Determine an expression for the amplification factor.
- ii. Determine under what circumstances the algorithm is stable.
- iii. Obtain an expression for the local truncation error.
- iv. Determine the rate of convergence.

Answers:

- i. $A = [1 - \alpha \Delta t \tilde{\lambda}^h - (1 - \alpha) \Delta t (\lambda^h + \tilde{\lambda}^h)] / (1 + \alpha \Delta t \lambda^h)$
- ii. $\Delta t [\tilde{\lambda}^h + \lambda^h(1 - 2\alpha)] < 2$
- iii. $\tau(t_n) = \Delta t \left[\left(\alpha - \frac{1}{2} \right) \ddot{d}(t_{n+\alpha}) + \alpha \tilde{\lambda}^h \dot{d}(t_{n+\alpha}) \right]$
- iv. $k = 1$

Exercise 7. Consider the following one-parameter (α) family of iterative, predictor-corrector algorithms:

$$\begin{aligned} Mv_{n+1}^{i+1} + Kd_{n+1}^{i+1} &= F_{n+1} \\ d_{n+1}^0 &= d_n + (1 - \alpha)\Delta t v_n \\ d_{n+1}^{i+1} &= d_{n+1}^0 + \alpha \Delta t v_{n+1}^{i+1} \end{aligned}$$

Assume $\alpha \in [0, 1]$ and $i = 0, 1, \dots, I$, where $I + 1$ is the total number of iterations. Define

$$d_{n+1} = d_{n+1}^{I+1}$$

$$v_{n+1} = v_{n+1}^{I+1}$$

(With $I = 0$, this algorithm is the same as the one considered in Exercise 2.) Take the case $I = 1$.

- i. Determine an expression for the amplification factor.
- ii. Determine under what circumstances the algorithms are stable.
- iii. Obtain an expression for the local truncation error.
- iv. Determine the rate of convergence as a function of α .

Solution:

$$Mv_{n+1}^1 + Kd_{n+1}^0 = F_{n+1} \quad (\text{a})$$

$$Mv_{n+1}^2 + Kd_{n+1}^1 = F_{n+1} \quad (\text{b})$$

$$d_{n+1}^0 = d_n + (1 - \alpha)\Delta t v_n \quad (\text{c})$$

$$d_{n+1}^1 = d_{n+1}^0 + \alpha\Delta t v_{n+1}^1 \quad (\text{d})$$

$$d_{n+1}^2 = d_{n+1}^0 + \alpha\Delta t v_{n+1}^2 \quad (\text{e})$$

$$(a) \Rightarrow v_{n+1}^1 = M^{-1}(F_{n+1} - Kd_{n+1}^0) \quad (\text{f})$$

$$(d) \text{ and } (f) \Rightarrow d_{n+1}^1 = d_{n+1}^0 + \alpha\Delta t M^{-1}(F_{n+1} - Kd_{n+1}^0) \quad (\text{g})$$

$$(b) \text{ and } (g) \Rightarrow Mv_{n+1}^2 + Kd_{n+1}^0 + \alpha\Delta t KM^{-1}F_{n+1} - \alpha\Delta t KM^{-1}Kd_{n+1}^0 = F_{n+1}$$

That is,

$$Mv_{n+1}^2 + K(I - \alpha\Delta t M^{-1}K)d_{n+1}^0 = (I - \alpha\Delta t KM^{-1})F_{n+1}$$

Let

$$B = I - \alpha\Delta t M^{-1}K$$

$$\tilde{B} = I - \alpha\Delta t KM^{-1}.$$

Therefore,

$$Mv_{n+1}^2 + KBd_{n+1}^0 = \tilde{B}F_{n+1} \quad (\text{h})$$

$$(e) \text{ and } (h) \Rightarrow Mv_{n+1}^2 + KB(d_{n+1}^0 - \alpha\Delta t v_{n+1}^2) = \tilde{B}F_{n+1}$$

Therefore,

$$(M - \alpha\Delta t KB)v_{n+1}^2 + KBd_{n+1}^2 = \tilde{B}F_{n+1}$$

$$v_{n+1}^2 = v_{n+1}, \quad d_{n+1}^2 = d_{n+1}$$

Thus

$$\tilde{M}v_{n+1} + \tilde{K}d_{n+1} = \tilde{F}_{n+1}$$

$$d_{n+1} = d_n + (1 - \alpha)\Delta t v_n + \alpha\Delta t v_{n+1}$$

where

$$\tilde{M} = M - \alpha\Delta t KB, \quad \tilde{K} = KB, \quad \text{and} \quad \tilde{F}_{n+1} = \tilde{B}F_{n+1}$$

Hence, the modal equation is

$$(1 - \alpha\Delta t \lambda^h z)v_{n+1} + \lambda^h z d_{n+1} = zF_{n+1}$$

where

$$d_{n+1} = d_n + (1 - \alpha)\Delta t v_n + \alpha\Delta t v_{n+1}$$

and

$$z = 1 - \alpha\Delta t \lambda^h$$

Therefore

$$(1 - \alpha \Delta t \lambda^h z) \cdot \frac{1}{\alpha \Delta t} (d_{n+1} - d_n - (1 - \alpha) \Delta t v_n) + \lambda^h z d_{n+1} = z F_{n+1}$$

$$d_{n+1} - (1 - \alpha \Delta t \lambda^h z) d_n - (1 - \alpha \Delta t \lambda^h z)(1 - \alpha) \Delta t v_n = \alpha \Delta t z F_{n+1}$$

but

$$(1 - \alpha \Delta t \lambda^h z) v_n + \lambda^h z d_n = z F_n$$

Thus

$$d_{n+1} - (1 - \alpha \Delta t \lambda^h z) d_n + \Delta t (1 - \alpha) (\lambda^h z d_n - z F_n) = \alpha \Delta t z F_{n+1}$$

Therefore,

$$d_{n+1} - (1 - \Delta t \lambda^h z) d_n = \Delta t z F_{n+\alpha}$$

i. *Amplification factor:* $A = 1 - \Delta t \lambda^h (1 - \alpha \Delta t \lambda^h) = 1 - \Delta t \lambda^h + \alpha (\Delta t \lambda^h)^2$

ii. *Stability conditions:* $|A| < 1$, i.e., $-1 < A < 1$.

$$A < 1 \Rightarrow -\Delta t \lambda^h + \alpha (\Delta t \lambda^h)^2 < 0$$

Because $\lambda^h > 0$ and $\Delta t > 0$, it follows that $\alpha \Delta t \lambda^h < 1$ and $\Delta t < 1/\alpha \lambda^h$.

$$A > -1 \Rightarrow \alpha (\Delta t \lambda^h)^2 - \Delta t \lambda^h + 2 > 0$$

i.e.,

$$(\Delta t \lambda^h)^2 - \frac{\Delta t \lambda^h}{\alpha} + \frac{2}{\alpha} > 0 \quad (\alpha > 0)$$

$$f(\Delta t \lambda^h) = \left[\Delta t \lambda^h - \frac{1}{2\alpha} (1 - \sqrt{1 - 8\alpha}) \right] \left[\Delta t \lambda^h - \frac{1}{2\alpha} (1 + \sqrt{1 - 8\alpha}) \right] > 0$$

$$\text{If } \alpha \leq \frac{1}{8}, \Delta t \lambda^h > \frac{1}{2\alpha} (1 + \sqrt{1 - 8\alpha}) \text{ or } \Delta t \lambda^h < \frac{1}{2\alpha} (1 - \sqrt{1 - 8\alpha}).$$

If $\alpha > \frac{1}{8}$, $f(\Delta t \lambda^h) > 0$ for all $\Delta t \lambda^h$.

Stability conditions

$$\text{If } \alpha \leq \frac{1}{8}, \Delta t < \frac{1}{2\alpha \lambda^h} (1 - \sqrt{1 - 8\alpha}).$$

$$\text{If } \alpha > \frac{1}{8}, \Delta t < \frac{1}{\alpha \lambda^h}.$$

iii. Let $d(t_n) = d$, $\dot{d}(t_n) = \dot{d}$, \dots

$$\Delta t \tau(t_n) = d(t_{n+1}) - (1 - \Delta t \lambda^h + \alpha (\Delta t \lambda^h)^2) d(t_n)$$

$$- \alpha \Delta t (1 - \alpha \Delta t \lambda^h) F_{n+1} - \Delta t (1 - \alpha) (1 - \alpha \Delta t \lambda^h) F_n$$

$$= \cancel{d} + \Delta t \dot{d} + \frac{\Delta t^2}{2} \ddot{d} + O(\Delta t^3) - \cancel{d} + \Delta t \lambda^h d - \alpha (\Delta t \lambda^h)^2 d$$

$$\begin{aligned}
 & - (1 - \alpha \Delta t \lambda^h) \Delta t \{ \alpha \dot{F}_n + \alpha \Delta t \dot{F}_n + \alpha \frac{\Delta t^2}{2} \ddot{F}_n + (1 - \cancel{\alpha}) F_n \} \\
 & = \Delta t (\ddot{d} + \lambda^h d - F_n) + \frac{\Delta t^2}{2} (\ddot{d} - 2\alpha(\lambda^h)^2 d - 2\alpha \dot{F}_n + 2\alpha \lambda^h F_n) \\
 & \quad + O(\Delta t^3) \\
 \text{But } \dot{d} + \lambda^h d & = F_n, \text{ so} \\
 \Delta t \tau(t_n) & = \frac{\Delta t^2}{2} (\ddot{d} - 2\alpha \dot{F}_n + 2\alpha \lambda^h (F_n - \lambda^h d)) + O(\Delta t^3) \\
 & = \frac{\Delta t^2}{2} (\ddot{d} - 2\alpha (\dot{F}_n - \lambda^h \dot{d})) + O(\Delta t^3) \\
 & = \frac{\Delta t^2}{2} (1 - 2\alpha) \ddot{d} + O(\Delta t^3)
 \end{aligned}$$

iv. If $\alpha \neq \frac{1}{2}$, $|\tau(t_n)| \leq c \Delta t$

$k = 1$, first order

If $\alpha = \frac{1}{2}$, $|\tau(t_n)| \leq c \Delta t^2$

$k = 2$, second order

8.3 ELEMENTARY FINITE DIFFERENCE EQUATIONS FOR THE ONE-DIMENSIONAL HEAT EQUATION; THE VON NEUMANN METHOD OF STABILITY ANALYSIS

In order to appreciate some of the strengths and weaknesses of recently proposed finite element algorithms, a basic understanding of finite difference equations and their methods of analysis is required. All of the results obtained in this section are in the form of examples and exercises.

The von Neumann Method

A clear description of this technique, and others used for stability analysis of difference equations, is presented in Mitchell and Griffiths [5]. To illustrate its use, we shall consider only a simple example. (The reader interested in further details is urged to consult [5] and references therein.)

Example

Consider the one-dimensional heat equation

$$u_{,t} = k u_{,xx} \quad (8.3.1)$$

where $k = \kappa/\rho c$, and the algorithm (*FTCS: forward in time, centered in space*)

$$u_{n+1}^h(m) = (1 - 2r)u_n^h(m) + r(u_n^h(m+1) + u_n^h(m-1)) \quad (8.3.2)$$

where $u_n^h(m) = u_n^h(x_m)$, and so on, and $r = k\Delta t/h^2$, where h is the node spacing. Equation (8.3.2) is explicit, and so we would like to know for which values of r the algorithm is stable. The error induced by small perturbations in initial data, rounding, and so on, is denoted by $\delta_n(m)$ and satisfies the algorithmic equation, namely,

$$\delta_{n+1}(m) = (1 - 2r)\delta_n(m) + r(\delta_n(m+1) + \delta_n(m-1)) \quad (8.3.3)$$

Furthermore, $\delta_n(m)$ is assumed to take on the following form:

$$\delta_n(m) = \zeta^n \exp(im\xi), \quad i = \sqrt{-1} \quad (8.3.4)$$

Substitution of (8.3.4) into (8.3.3) leads to

$$\zeta = 1 - 4r \sin^2\left(\frac{\xi}{2}\right) \quad (8.3.5)$$

The condition of stability is $|\zeta| \leq 1$ for all ξ , which requires

$$r \leq \frac{1}{2 \sin^2(\xi/2)} \quad (8.3.6)$$

for all ξ , and so $r \leq \frac{1}{2}$ is the stability result (see Remark 1 under Table 8.2.1).

Exercise 1. Consider the following algorithms for the one-dimensional heat equation:

$$u_{n+1}^h(m) = u_n^h(m) + r(u_{n+1}^h(m+1) - 2u_{n+1}^h(m) + u_{n+1}^h(m-1)) \\ (\text{BTCS: backward in time, centered in space}) \quad (8.3.7)$$

$$u_{n+1}^h(m) = u_n^h(m) + \frac{r}{2} \left(u_{n+1}^h(m+1) - 2u_{n+1}^h(m) + u_{n+1}^h(m-1) \right. \\ \left. + u_n^h(m+1) - 2u_n^h(m) + u_n^h(m-1) \right) \quad (\text{Crank-Nicolson}) \quad (8.3.8)$$

$$u_{n+1}^h(m) = u_{n-1}^h(m) + 2r(u_n^h(m+1) - 2u_n^h(m) + u_n^h(m-1)) \quad (\text{leap frog}) \\ (8.3.9)$$

Determine the stability condition for each algorithm based upon the von Neumann approach. [Answer: BTCS and Crank-Nicolson are unconditionally stable; leap frog is unconditionally unstable.]

Local Truncation Error

The local truncation error measures how well the exact solution satisfies the algorithmic equation. For example, consider the FTCS scheme (8.3.2). In this case,

$$u(x_m, t_{n+1}) = (1 - 2r)u(x_m, t_n) + r(u(x_{m+1}, t_n) + u(x_{m-1}, t_n)) + \Delta t \cdot \tau \quad (8.3.10)$$

defines the local truncation error τ . To determine the form of τ , Taylor expansions of u about (x_m, t_n) may be employed. It can be shown that τ in (8.3.10) has the form $O(\Delta t^k, h^l)$ and $k = 1, l = 2$ (verify!). If the algorithm in question is stable, then the exponents k and l govern the rate of convergence of $u_n^h(m)$ to $u(x_m, t_n)$. As long as k and l are greater than zero, the algorithm is called consistent.

Exercise 2. Determine k and l for the algorithms considered in the previous exercise.
 [Answers: BTCS, $k = 1$, $l = 2$; Crank-Nicolson and leap frog, $k = l = 2$.]

Exercise 3. The *DuFort-Frankel scheme* [6] is defined by

$$u_{n+1}^h(m) = u_{n-1}^h(m) + 2r(u_n^h(m+1) - u_{n-1}^h(m) - u_{n+1}^h(m) + u_n^h(m-1)) \quad (8.3.11)$$

Show that this scheme is unconditionally stable, but the local truncation error has the form

$$\tau = O\left(\Delta t^2, h^2, \frac{\Delta t^2}{h^2}\right) \quad (8.3.12)$$

Thus convergence will only be achieved if $\Delta t \rightarrow 0$ faster than $h \rightarrow 0$. Such a scheme is called *conditionally consistent*.

Remark

The DuFort-Frankel scheme is frequently described as an unconditionally stable, explicit method. Rearranging (8.3.11) exhibits the fact that the solution may be obtained explicitly:

$$(1 + 2r)u_{n+1}^h(m) = (1 - 2r)u_{n-1}^h(m) + 2r(u_n^h(m+1) + u_n^h(m-1)) \quad (8.3.13)$$

Conditional consistency seems the price one has to pay to achieve an explicit, unconditionally stable method. The time-step restriction imposed is often much smaller than that needed for accuracy in typical implicit methods such as Crank-Nicolson.

Exercise 4. *Saul'yev's method* [7] is given by

$$u_{n+1}^h(m) = u_n^h(m) + r(u_n^h(m+1) - u_n^h(m) - u_{n+1}^h(m) + u_{n+1}^h(m-1)) \quad (8.3.14)$$

Assume that u^h is specified at node x_0 (i.e., $m = 0$). Further, assume that the equations are solved in the order $m = 1, 2, 3, \dots$, and so on. In this case (8.3.14) can be rearranged so that the solution may be obtained explicitly:

$$(1 + r)u_{n+1}^h(m) = u_n^h(m) + r(u_n^h(m+1) - u_n^h(m) + \underbrace{u_{n+1}^h(m-1)}_{\text{known}}) \quad (8.3.15)$$

Show that Saul'yev's method is unconditionally stable but conditionally consistent.

Exercise 5. Assume piecewise linear finite element functions in one dimension. Assuming $\ell = 0$ and equal node spacing, show that the semidiscrete equations at an internal node take the form

$$\frac{\rho ch}{6} \left(d_{m-1}(t) + 4d_m(t) + d_{m+1}(t) \right) - \frac{\kappa}{h} \left(d_{m-1}(t) - 2d_m(t) + d_{m+1}(t) \right) = 0 \quad (8.3.16)$$

where here the subscripts refer to the node number.

Exercise 6. Apply the generalized trapezoidal family of ordinary differential equation algorithms to (8.3.16). Use the von Neumann method to determine the stability conditions. Obtain local truncation error expressions. Contrast this family of methods with the other difference schemes presented in this section. (Note that $d_m(t_n) = u_n^h(x_m)$ for comparison with the methods of this section.)

Exercise 7. Show that if the consistent mass is replaced by (7.3.52), then (8.3.16) becomes

$$\rho ch \left(\nu \dot{d}_{m-1}(t) + (1 - 2\nu) \dot{d}_m(t) + \nu \dot{d}_{m+1}(t) \right) - \frac{\kappa}{h} \left(d_{m-1}(t) - 2d_m(t) + d_{m+1}(t) \right) = 0 \quad (8.3.17)$$

Combine (8.3.17) with the trapezoidal algorithm (i.e., $\alpha = \frac{1}{2}$) in time. Assuming $\nu = \frac{1}{12}$ (i.e., "higher-order mass"; see Sec. 7.3.2), show that the local truncation error is $O(\Delta t^2, h^4)$.

Exercise 8. Generalize (8.3.17) to include a source term:

$$\begin{aligned} \rho ch \left(\nu \dot{d}_{m-1}(t) + (1 - 2\nu) \dot{d}_m(t) + \nu \dot{d}_{m+1}(t) \right) - \frac{\kappa}{h} \left(d_{m-1}(t) - 2d_m(t) + d_{m+1}(t) \right) \\ = h[\nu f_{m-1}(t) + (1 - 2\nu) f_m(t) + \nu f_{m+1}(t)] \end{aligned} \quad (8.3.18)$$

Show that trapezoidal rule and $\nu = \frac{1}{12}$ result in $\tau = O(\Delta t^2, h^4)$.

Remark

Exercise 8 illustrates that the idea of higher-order mass treatment applies to the source term as well. The right-hand side of (8.3.18), with $\nu = \frac{1}{12}$, can be constructed by assuming a piecewise-linear continuous distribution of f , and by applying the quadrature rule, $\xi_2 = -\xi_1 = \sqrt{\frac{2}{3}}$, $W_1 = W_2 = 1$. This is the rule that generates the higher-order element mass matrix.

Project. Program and compare the methods described in this section. The explicit methods are easy to program. For the implicit methods, you will need to solve a system of equations at each time step. The equation solver in the DLEARN program can be used for this purpose. Define, set up, and solve a simple problem of one-dimensional heat conduction. Vary Δt and h . Compare the performances of the methods on the basis of stability and accuracy.

Remarks

1. Analyses of some other fully discrete schemes for the one-dimensional heat equation may be found in [8].

2. A multidimensional finite element generalization of the Saul'yev scheme has been developed by Trujillo [9]. The capacity matrix is lumped and the conductivity is split into upper and lower triangular parts, K_U and K_L , respectively, such that $K_U = K_L^T$. Solution proceeds in a two-step format in which upper and lower triangular matrices are used in alternating fashion. No factorizations are required and the process has features in common with the Gauss-Seidel iterative procedure for solving linear equations.

8.4. ELEMENT-BY-ELEMENT (EBE) IMPLICIT METHODS

Unconditionally stable, second-order accurate, implicit methods, such as the trapezoidal rule, perform very well in heat conduction analysis. The drawbacks are the storage and equation-solving burden engendered by the coefficient matrix. In recent years attempts have been made to achieve the desirable properties of implicit methods in a simpler computational setting. Methods described in this section employ the element-by-element concept. A product approximation of the element assembly is made so that the inversion of the coefficient matrix is replaced by sequential inversions of element matrices.

Method 1 (Hughes et al. [10]):

Recall that the generalized trapezoidal algorithm can be written in the form

$$(\mathbf{M} + \alpha \Delta t \mathbf{K}) \mathbf{d}_{n+1} = (\mathbf{M} - (1 - \alpha) \Delta t \mathbf{K}) \mathbf{d}_n + \Delta t \mathbf{F}_{n+\alpha} \quad (8.4.1)$$

For future reference, it is convenient to rewrite (8.4.1) as

$$\mathbf{M}^{1/2} \mathbf{d}_{n+1} = \mathbf{A} \mathbf{M}^{1/2} \mathbf{d}_n + \Delta t \mathbf{B} \mathbf{M}^{-1/2} \mathbf{F}_{n+\alpha} \quad (8.4.2)$$

where

$$\mathbf{A} = \mathbf{B}(\mathbf{I} - (1 - \alpha) \Delta t \mathbf{C}) \quad (8.4.3)$$

$$\mathbf{B} = (\mathbf{I} + \alpha \Delta t \mathbf{C})^{-1} \quad (8.4.4)$$

$$\mathbf{C} = \mathbf{M}^{-1/2} \mathbf{K} \mathbf{M}^{-1/2} \quad (8.4.5)$$

and $\mathbf{M}^{1/2}$ is the square root of \mathbf{M} and $\mathbf{M}^{-1/2} = (\mathbf{M}^{1/2})^{-1}$. The square root of \mathbf{M} can be defined for the general case. However, we will restrict attention to the case in which \mathbf{M} is diagonal and $\mathbf{M}^{1/2}$ has the obvious definition. For the method being described, this is necessary in order to achieve computational efficiency. We wish to view \mathbf{A} and \mathbf{B} as functions of α and Δt ,

$$\mathbf{A} = \mathbf{A}(\alpha, \Delta t) \quad (8.4.6)$$

$$\mathbf{B} = \mathbf{B}(\alpha, \Delta t) \quad (8.4.7)$$

One-pass EBE algorithm. Element counterparts of (8.4.3) through (8.4.5) are, respectively,

$$\mathbf{A}^\epsilon = \mathbf{B}^\epsilon(\mathbf{I} - (1 - \alpha) \Delta t \mathbf{C}^\epsilon) \quad (8.4.8)$$

$$\mathbf{B}^\epsilon = (\mathbf{I} + \alpha \Delta t \mathbf{C}^\epsilon)^{-1} \quad (8.4.9)$$

$$\mathbf{C}^\epsilon = \mathbf{M}^{-1/2} \mathbf{K}^\epsilon \mathbf{M}^{-1/2} \quad (8.4.10)$$

Consider the following analog of (8.4.2)

$$\mathbf{M}^{1/2} \mathbf{d}_{n+1} = \left(\prod_{\epsilon=1}^{n_{el}} \mathbf{A}^\epsilon \right) \mathbf{M}^{1/2} \mathbf{d}_n + \Delta t \left(\prod_{\epsilon=1}^{n_{el}} \mathbf{B}^\epsilon \right) \mathbf{M}^{-1/2} \mathbf{F}_{n+\alpha} \quad (8.4.11)$$

where

$$\prod_{e=1}^{n_{el}} A^e = A^1 A^2 \cdots A^{n_{el}} \quad (8.4.12)$$

$$\prod_{e=1}^{n_{el}} B^e = B^1 B^2 \cdots B^{n_{el}} \quad (8.4.13)$$

The matrices (8.4.12) and (8.4.13) may be thought of as approximations to A and B , respectively. Equation (8.4.11) defines an algorithm in which the calculations may be performed on an element-by-element basis. Specifically, K need never be formed, and equation solving involves arrays of element size only. Thus the storage requirements of the algorithm are greatly reduced compared with typical implicit methods. With $\alpha \geq \frac{1}{2}$, unconditional stability and first-order accuracy in Δt are attained by (8.4.11). Note that the ordering of the elements is completely arbitrary and thus no limitation is placed on the topology of the mesh.

Two-pass EBE algorithm. Consider the following algorithm:

$$\begin{aligned} M^{1/2} d_{n+1} &= \left(\prod_{e=1}^{n_{el}} A^e \left(\alpha, \frac{\Delta t}{2} \right) \right) \left(\prod_{e=n_{el}}^1 A^e \left(\alpha, \frac{\Delta t}{2} \right) \right) M^{1/2} d_n \\ &\quad + \Delta t \left(\prod_{e=1}^{n_{el}} B^e \left(\alpha, \frac{\Delta t}{2} \right) \right) \left(\prod_{e=n_{el}}^1 B^e \left(\alpha, \frac{\Delta t}{2} \right) \right) M^{-1/2} F_{n+1}, \end{aligned} \quad (8.4.14)$$

If $\alpha = \frac{1}{2}$, unconditional stability and second-order accuracy in Δt are attained [10].

Remark

The computational properties of this method seem very attractive. Unfortunately, conditional consistency afflicts this procedure just as it does the DuFort-Frankel and Saul'yev methods described in Sec. 8.3. In an effort to retain the present computational architecture but achieve the accuracy behavior of standard implicit methods, the following method has been introduced.

Method 2 (Hughes et al. [11, 12], Winget and Hughes [13])

In this method we employ the element-by-element concept in an iterative equation-solving format. The equations of the generalized trapezoidal algorithm are iteratively solved using *preconditioned conjugate gradients (PCG)* with an element-by-element approximate factorization preconditioner. Thus the properties of the trapezoidal algorithm are inherited. In order to describe the details, let the algorithmic equation, for example,

$$(M - \alpha \Delta t K) v_{n+1} = F_{n+1} - K \tilde{d}_{n+1} \quad (8.4.15)$$

be written as

$$Ax = b \quad (8.4.16)$$

where

$$\mathbf{A} = \mathbf{M} + \alpha \Delta t \mathbf{K} \quad (8.4.17)$$

$$\mathbf{x} = \mathbf{v}_{n+1} \quad (8.4.18)$$

$$\mathbf{b} = \mathbf{F}_{n+1} - \mathbf{K}\tilde{\mathbf{d}}_{n+1} \quad (8.4.19)$$

Note that \mathbf{M} may be nondiagonal in this case. Equation (8.4.16) is solved by the PCG method (see Table 8.4.1).

TABLE 8.4.1 Flowchart of Pre-conditioned Conjugate Gradients

| | |
|---------|---|
| Step 1. | Initialization: |
| | $m = 0, \mathbf{x}_0 = \mathbf{0}$ |
| | $\mathbf{r} = \mathbf{b}$ |
| | $\mathbf{p}_0 = \mathbf{z}_0 = \mathbf{B}^{-1}\mathbf{r}_0$ |
| Step 2. | $\alpha_m = \mathbf{r}_m^T \mathbf{z}_m / \mathbf{p}_m^T \mathbf{A} \mathbf{p}_m$ |
| Step 3. | $\mathbf{x}_{m+1} = \mathbf{x}_m + \alpha_m \mathbf{p}_m$ |
| Step 4. | $\mathbf{r}_{m+1} = \mathbf{r}_m - \alpha_m \mathbf{A} \mathbf{p}_m$ |
| Step 5. | Convergence check: $\ \mathbf{r}_{m+1}\ < \delta \ \mathbf{r}_0\ ?$ |
| | Yes: Return |
| | No: Continue |
| Step 6. | $\mathbf{z}_{m+1} = \mathbf{B}^{-1} \mathbf{r}_{m+1}$ |
| Step 7. | $\beta_m = \mathbf{r}_{m+1}^T \mathbf{z}_{m+1} / \mathbf{r}_m^T \mathbf{z}_m$ |
| Step 8. | $\mathbf{p}_{m+1} = \mathbf{z}_{m+1} + \beta_m \mathbf{p}_m$ |
| Step 9. | $m \leftarrow m + 1, \text{ go to Step 2.}$ |

The preconditioning matrix is denoted by \mathbf{B} in Table 8.4.1. If $\mathbf{B} = \mathbf{I}$ (the identity matrix), then we have the classical conjugate gradients procedure. The more closely \mathbf{B} approximates \mathbf{A} , the faster the convergence will be. An element-by-element procedure will be used to define \mathbf{B} . We have developed a number of these procedures in [12, 13] but will just present here what we view to be the most effective one. To do so it is helpful to introduce the following notations: Let \mathbf{C} be a symmetric, positive-definite matrix. Then we have

$$\mathbf{C} = \mathbf{L}_p(\mathbf{C})\mathbf{D}_p(\mathbf{C})\mathbf{U}_p(\mathbf{C}) \quad (\text{product decomposition}) \quad (8.4.20)$$

$$\mathbf{C} = \mathbf{L}_s(\mathbf{C}) + \mathbf{D}_s(\mathbf{C}) + \mathbf{U}_s(\mathbf{C}) \quad (\text{sum decomposition}) \quad (8.4.21)$$

where the subscripts p and s indicate product and sum, respectively.

Equation (8.4.20) represents the *Crout factorization* of \mathbf{C} . Thus \mathbf{L}_p and \mathbf{U}_p are lower and upper triangular matrices, respectively, with diagonal entries equal to 1, and \mathbf{D}_p is a diagonal matrix.

In equation (8.4.21), \mathbf{L}_s and \mathbf{U}_s are lower and upper triangular arrays, respectively, with diagonal entries equal to 0, and \mathbf{D}_s is diagonal.

By the symmetry of C ,

$$L_p(C) = U_p(C)^T \quad (8.4.22)$$

$$L_s(C) = U_s(C)^T \quad (8.4.23)$$

Corresponding to A^e , we introduce a *scaled, regularized element array* as follows [13]:

$$\widetilde{A}^e = I + D_s(A)^{-1/2}(A^e - D_s(A^e))D_s(A)^{-1/2} \quad (8.4.24)$$

With these notations we can define the *reordered Crout EBE preconditioner*:

$$B = D_s(A)^{1/2} \left(\prod_{e=1}^{n_{el}} L_p(\widetilde{A}^e) \right) \left(\prod_{e=1}^{n_{el}} D_p(\widetilde{A}^e) \right) \left(\prod_{e=n_{el}}^1 U_p(\widetilde{A}^e) \right) D_s(A)^{1/2} \quad (8.4.25)$$

Remarks

1. B is symmetric and positive-definite. Aside from the trivial scaling manifested by the $D_s(A)^{1/2}$ terms in (8.4.25), all computations may be performed on the element level. This involves processing full, small matrices. The major cost constituents are element-level forward reductions–back substitutions and the function evaluation required in Step 2 of Table 8.4.1, i.e., the element-level calculation

$$Ap_m = \sum_{e=1}^{n_{el}} (A^e p_m) \quad (8.4.26)$$

The element factorization cost is not significant because it is amortized over the number of time steps and iterations per step required for convergence.

2. If elements are segregated into noncontiguous subgroups, then calculations are parallelizable. For example, bricklike domains can be decomposed into eight noncontiguous element groups (see Fig. 8.4.1). Because the elements in each subgroup have no common degrees of freedom, they can be processed in parallel. The eight groups, however, need to be processed sequentially. For analogous two-dimensional domains, four element groups need to be employed. In our experiences so far, parallel orderings have converged about as fast as sequential orderings [13]. This means that for bricklike domains the theoretical speed-up factor on a parallel computer is $n_{el}/8$ in three dimensions and $n_{el}/4$ in two dimensions.

3. The potential of this method is greatest in three-dimensional applications where the bandwidth of the coefficient matrix is large. The element-by-element procedure is independent of bandwidth and thus achieves significant operation count advantages [13]. The advantage increases in nonlinear applications where frequent refactorizations are typically necessary [13].

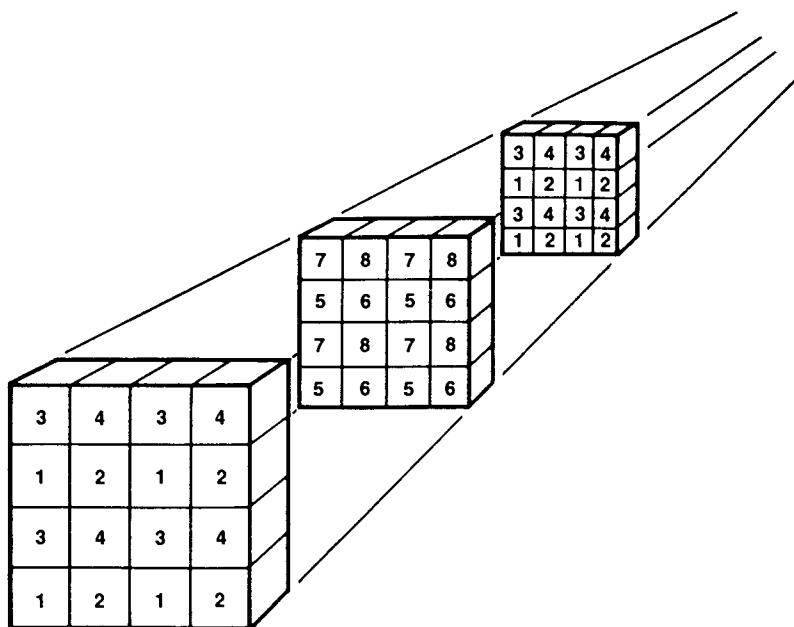


Figure 8.4.1 Decomposition of three-dimensional domain into eight groups of brick elements for parallel processing.

8.5 MODAL ANALYSIS

The technique of modal analysis is often used in place of step-by-step integration procedures such as those described in the previous sections. For the problem under consideration, modal analysis involves the following steps:

Step 1. Solve the eigenvalue problem for the pairs $\{\lambda_l^h, \psi_l\}$, $1 \leq l \leq n_{\text{modes}}$, where $n_{\text{modes}} \leq n_{\text{eq}}$ is the desired number of modes to participate in the subsequent calculations. The determination of n_{modes} is a matter of engineering judgment, which involves the following considerations:

- For each mode l selected, λ_l^h and ψ_l should be good approximations of the exact eigenvalue and eigenvector. This generally requires that $n_{\text{modes}} \ll n_{\text{eq}}$.
- The spatial variation of d and F must be adequately represented by an expansion in the first n_{modes} .

Step 2. Solve the scalar ordinary differential equations

$$\left. \begin{aligned} \dot{d}_{(l)} &= -\lambda_l^h d_{(l)} + F_{(l)} \\ d_{(l)}(0) &= d_{0(l)} \end{aligned} \right\} \quad 1 \leq l \leq n_{\text{modes}} \quad (8.5.1)$$

The solutions may be obtained exactly if each $F_{(l)}$ is a simple function of t . However, it

is generally more convenient to use a step-by-step method. To ensure essentially exact results, a very small time step may be employed. By virtue of the fact that the problems are of a scalar nature, the computational expenditure is generally negligible compared with Step 1.

Step 3. The approximation to $d(t)$ is synthesized from the modes, viz.,

$$d(t) \cong \sum_{l=1}^{n_{\text{modes}}} d_{(l)}(t) \psi_l \quad (8.5.2)$$

The calculations on the right-hand side of (8.5.2) must be performed for each time t at which the solution is required.

Remarks

Step-by-step methods of integration and modal analysis each have attributes that may make them advantageous in particular circumstances. Step-by-step methods are (1) easily coded, (2) felt to be more efficient for short-time calculations, and (3) are generalizable to nonlinear situations. Modal analysis is felt to be more efficient (1) if many analyses of the same configuration are necessary, (2) for long-time calculations, and (3) if only a small number of modes are participating in the solution. The appropriate technique depends heavily on the problem under consideration and thus both approaches should be within the repertoire of the analyst.

REFERENCES

Section 8.1

1. T. J. R. Hughes and W. K. Liu, "Implicit-Explicit Finite Elements in Transient Analysis: Stability Theory," *Journal of Applied Mechanics*, 45 (1978), 371–374.
2. T. J. R. Hughes and W. K. Liu, "Implicit-Explicit Finite Elements in Transient Analysis: Implementation and Numerical Examples," *Journal of Applied Mechanics*, 45 (1978), 375–378.
3. T. J. R. Hughes, K. S. Pister, and R. L. Taylor, "Implicit-Explicit Finite Elements in Nonlinear Transient Analysis," *Computer Methods in Applied Mechanics and Engineering*, 17/18 (1979), 159–182.

Section 8.2

4. R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial-Value Problems* (2nd ed.). New York: Interscience, 1967.

Section 8.3

5. A. R. Mitchell and D. F. Griffiths, *The Finite Difference Method in Partial Differential Equations*. London: John Wiley, 1980.
6. W. F. Ames, *Numerical Methods for Partial Differential Equations*, (2nd ed.). New York: Academic Press, 1977.

7. V. K. Saul'yev, *Integration of Equations of the Parabolic Type by the Method of Nets*. New York: Pergamon Press, 1964.
8. T. J. R. Hughes and T. E. Tezduyar, "Analysis of Some Fully Discrete Algorithms for the One-Dimensional Heat Equation," *International Journal for Numerical Methods in Engineering*, 21 (1985), 163–168.
9. D. M. Trujillo, "An Unconditionally Stable Explicit Algorithm for Finite Element Heat Conduction Analysis," *Nuclear Engineering and Design*, 32 (1975), 110–120.

Section 8.4

10. T. J. R. Hughes, I. Levit, and J. Winget, "Element-by-Element Implicit Algorithms for Heat Conduction," *Journal of Engineering Mechanics*, 109, no. 2 (1983), 576–585.
11. T. J. R. Hughes, I. Levit, and J. Winget, "An Element-by-Element Solution Algorithm for Problems of Structural and Solid Mechanics," *Computer Methods in Applied Mechanics and Engineering*, 36, no. 2 (1983), 241–254.
12. T. J. R. Hughes, J. Winget, I. Levit, and T. E. Tezduyar, "New Alternating Direction Procedures in Finite Element Analysis based upon EBE Approximate Factorizations", in *Recent Developments in Computer Methods for Nonlinear Solid and Structural Mechanics*, eds. S. Atluri and N. Perrone, ASME Applied Mechanics Symposia Series, New York, 1983, 75–109.
13. J. M. Winget and T. J. R. Hughes, "Solution Algorithms for Nonlinear Transient Heat Conduction Analysis Employing Element-by-Element Iterative Strategies," *Computer Methods in Applied Mechanics and Engineering*, 52 (1985), 711–815.

9

Algorithms for Hyperbolic and Parabolic-Hyperbolic Problems

9.1 ONE-STEP ALGORITHMS FOR THE SEMIDISCRETE EQUATION OF MOTION

9.1.1 The Newmark Method

Recall from Chapter 7 that the semidiscrete equation of motion is written as

$$M\ddot{\mathbf{d}} + C\dot{\mathbf{d}} + K\mathbf{d} = \mathbf{F} \quad (9.1.1)$$

where M is the mass matrix, C is the viscous damping matrix, K is the stiffness matrix, \mathbf{F} is the vector of applied forces, and \mathbf{d} , $\dot{\mathbf{d}}$, and $\ddot{\mathbf{d}}$ are the displacement, velocity and acceleration vectors, respectively. We take M , C , and K to be symmetric; M is positive-definite, and C and K are positive-semidefinite.

The initial-value problem for (9.1.1) consists of finding a displacement, $\mathbf{d} = \mathbf{d}(t)$, satisfying (9.1.1) and the given initial data:

$$\mathbf{d}(0) = \mathbf{d}_0 \quad (9.1.2)$$

$$\dot{\mathbf{d}}(0) = \mathbf{v}_0 \quad (9.1.3)$$

Perhaps the most widely used family of direct methods for solving (9.1.1) to (9.1.3) is the *Newmark family* [1], which consists of the following equations:

$$M\mathbf{a}_{n+1} + C\mathbf{v}_{n+1} + K\mathbf{d}_{n+1} = \mathbf{F}_{n+1} \quad (9.1.4)$$

$$\mathbf{d}_{n+1} = \mathbf{d}_n + \Delta t \mathbf{v}_n + \frac{\Delta t^2}{2} [(1 - 2\beta)\mathbf{a}_n + 2\beta\mathbf{a}_{n+1}] \quad (9.1.5)$$

$$v_{n+1} = v_n + \Delta t[(1 - \gamma)a_n + \gamma a_{n+1}] \quad (9.1.6)$$

where d_n , v_n , and a_n are the approximations of $\mathbf{d}(t_n)$, $\dot{\mathbf{d}}(t_n)$, and $\ddot{\mathbf{d}}(t_n)$, respectively. Equation (9.1.4) is simply the equation of motion in terms of the approximate solution, and (9.1.5) and (9.1.6) are finite difference formulas describing the evolution of the approximate solution. The parameters β and γ determine the stability and accuracy characteristics of the algorithm under consideration. Equations (9.1.4) to (9.1.6) may be thought of as three equations for determining the three unknowns d_{n+1} , v_{n+1} , and a_{n+1} , it being assumed that d_n , v_n , and a_n are known from the previous step's calculations. The Newmark family contains as special cases many well-known and widely used methods.

Implementation: a-form

There are several possible implementations. We will sketch one, but we leave further details until Sec. 9.4, which deals with operator and mesh partitions. The results in Sec. 9.4 include the Newmark method as a special case. Define *predictors*:

$$\tilde{d}_{n+1} = d_n + \Delta t v_n + \frac{\Delta t^2}{2}(1 - 2\beta)a_n \quad (9.1.7)$$

$$\tilde{v}_{n+1} = v_n + (1 - \gamma)\Delta t a_n \quad (9.1.8)$$

Equations (9.1.5) and (9.1.6) may then be written as

$$d_{n+1} = \tilde{d}_{n+1} + \beta \Delta t^2 a_{n+1} \quad (9.1.9)$$

$$v_{n+1} = \tilde{v}_{n+1} + \gamma \Delta t a_{n+1} \quad (9.1.10)$$

To start the process, a_0 may be calculated from

$$M a_0 = F - C v_0 - K d_0 \quad (9.1.11)$$

or specified directly. The recursion relation determines a_{n+1} :

$$(M + \gamma \Delta t C + \beta \Delta t^2 K) a_{n+1} = F_{n+1} - C \tilde{v}_{n+1} - K \tilde{d}_{n+1} \quad (9.1.12)$$

Equations (9.1.9) and (9.1.10) may then be used to calculate d_{n+1} and v_{n+1} , respectively.

This form of implementation is convenient for generalization to algorithms that employ "mesh partitions" (see Sec. 9.4) but is not the most efficient implementation.

Exercise 1. Develop an implementation in which a_{n+1} and v_{n+1} are eliminated from (9.1.4) by way of (9.1.9) and (9.1.10). In this implementation the right-hand side of the equation system does not entail stiffness calculations.

9.1.2 Analysis

The convergence analysis of the Newmark family of methods is similar to the convergence analysis given in the previous chapter. The main steps are (1) reduction to an SDOF model problem; (2) definition of a suitable notion of stability, which is shown to hold under certain circumstances; (3) determination of the local truncation error, from which the order of accuracy is obtained, and (4) use of the latter two conditions to prove convergence for the SDOF problem. Some assumption on the form of C is necessary to perform the modal reduction. For example, if Rayleigh damping is assumed, i.e.,

$$C = aM + bK \quad (9.1.13)$$

then the symmetry and positive-definiteness of M and symmetry and positive-semidefiniteness of K enable the decomposition indicated in the commutative diagram on page 494, where ω is the undamped frequency of vibration and $\xi = (a/\omega + b\omega)/2$ is the damping ratio.

The SDOF model problem can be recast in a first-order form analogous to (8.2.32):

$$\boxed{y_{n+1} = Ay_n + L_n} \quad (9.1.14)$$

where A is the *amplification matrix* and

$$y_n = \begin{Bmatrix} d_n \\ v_n \end{Bmatrix} \quad (9.1.15)$$

Difference equations such as (9.1.14) are often referred to as *one-step, multivalue methods*, the number of values being equal to the dimension of the vector y , which in the present case is 2. The complete analysis of the algorithm reduces to consideration of (9.1.14). Further details will be described in this section and in the following sections.

A summary of *stability conditions for the Newmark method* follows [2, 3, 4].

Unconditional

$$2\beta \geq \gamma \geq \frac{1}{2} \quad (9.1.16)$$

Conditional

$$\gamma \geq \frac{1}{2} \quad (9.1.17)$$

$$\beta < \frac{\gamma}{2} \quad (9.1.1)$$

$$\omega^h \Delta t \leq \Omega_{\text{crit}} \quad (9.1.1')$$

where

$$\Omega_{\text{crit}} = \frac{\xi(\gamma - \frac{1}{2}) + [\gamma/2 - \beta + \xi^2(\gamma - \frac{1}{2})^{1/2}]}{(\gamma/2 - \beta)} \quad (\text{critical sampling frequency}) \quad (9.1.2)$$

The stability conditions must be satisfied for each mode in the system. Consequently, the maximum natural frequency, $\omega_{n_{eq}}^h$, is critical and therefore must satisfy (9.1.19). As described in Chapter 7, $\omega_{n_{eq}}^h$ may be bounded by the maximum frequency of the individual elements. Note that if $\gamma = \frac{1}{2}$ viscous damping has no effect on stability. Furthermore, when $\gamma > \frac{1}{2}$ the effect of viscous damping is to increase the critical time step of conditionally stable Newmark methods. *Thus the undamped critical sampling frequency, i.e., $\Omega_{\text{crit}} = (\gamma/2 - \beta)^{-1/2}$, serves as a conservative value when an estimate of the modal damping coefficient is not available.* (Recall that, in the case of Rayleigh damping, ξ is determined by ω^h .) In practice it is often more convenient to express (9.1.19) in terms of the *period of vibration*, $T = 2\pi/\omega$, in which case (9.1.19) becomes $\Delta t/T \leq \Omega_{\text{crit}}/(2\pi)$.

The Newmark family contains as special cases many well-known and widely used methods. Properties of some classical methods are summarized in Table 9.1. The average acceleration method is one of the most widely used methods for structural dynamics applications. Note that the linear acceleration and Fox-Goodwin methods are implicit and *conditionally* stable. They are not felt to be economically competitive for large-scale systems when compared to implicit and *unconditionally* stable techniques such as the average acceleration method. The central difference method

TABLE 9.1.1 Properties of Well-known Members of the Newmark Family of Methods

| Method | Type | β | γ | Stability condition ⁽²⁾ | Order of accuracy ⁽³⁾ |
|--|-------------------------|----------------|---------------|--|----------------------------------|
| Average acceleration (trapezoidal rule) | Implicit | $\frac{1}{4}$ | $\frac{1}{2}$ | Unconditional | 2 |
| Linear acceleration | Implicit | $\frac{1}{6}$ | $\frac{1}{2}$ | $\Omega_{\text{crit}} = 2\sqrt{3} \cong 3.464$ | 2 |
| Fox-Goodwin (royal road) | Implicit | $\frac{1}{12}$ | $\frac{1}{2}$ | $\Omega_{\text{crit}} = \sqrt{6} \cong 2.449$ | 2 |
| Central difference | Explicit ⁽¹⁾ | 0 | $\frac{1}{2}$ | $\Omega_{\text{crit}} = 2$ | 2 |

- Notes:*
1. Strictly speaking, M and C need to be diagonal for the central-difference method to be explicit.
 2. Stability is based upon the undamped case, in which $\xi = 0$.
 3. Second-order accuracy is achieved if and only if $\gamma = \frac{1}{2}$.

$$\begin{aligned}
 M\ddot{\mathbf{d}} + C\dot{\mathbf{d}} + K\mathbf{d} &= F \\
 \mathbf{d}(0) &= \mathbf{d}_0 \\
 \dot{\mathbf{d}}(0) &= \mathbf{v}_0
 \end{aligned}$$

Modal decomposition

$$\begin{aligned}
 \ddot{\mathbf{d}} + 2\xi\omega^h\dot{\mathbf{d}} + (\omega^h)^2\mathbf{d} &= F \\
 \mathbf{d}(0) &= \mathbf{d}_0 \\
 \dot{\mathbf{d}}(0) &= \mathbf{v}_0
 \end{aligned}$$

Temporal
discretization

$$M\ddot{\mathbf{a}}_{n+1} + C\dot{\mathbf{v}}_{n+1} + K\mathbf{d}_{n+1} = F_{n+1}$$

$$\mathbf{d}_{n+1} = \mathbf{d}_n + \Delta t \mathbf{v}_n + \frac{\Delta t^2}{2} \{(1 - 2\beta)\mathbf{a}_n + 2\beta\mathbf{a}_{n+1}\}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t \{(1 - \gamma)\mathbf{a}_n + \gamma\mathbf{a}_{n+1}\}$$

$\mathbf{d}_0, \mathbf{v}_0$ given

Modal
decomposition

$$a_{n+1} + 2\xi\omega^h v_{n+1} + (\omega^h)^2 d_{n+1} = F_{n+1}$$

$$d_{n+1} = d_n + \Delta t v_n + \frac{\Delta t^2}{2} \{(1 - 2\beta)a_n + 2\beta a_{n+1}\}$$

$$v_{n+1} = v_n + \Delta t \{(1 - \gamma)a_n + \gamma a_{n+1}\}$$

d_0, v_0 given

conditionally stable. However, when M and C are diagonal, it is explicit. When the time step restriction is not too severe, as is often the case in elastic wave-propagation problems, the central difference method is generally the most economical direct integration procedure and is thus widely used. The methods enumerated in Table 9.1.1 are numerically compared in Goudreau and Taylor [2].

Exercise 2. The name “central difference method” derives from the fact that (9.1.5) and (9.1.6) may be combined to yield

$$v_n = \frac{d_{n+1} - d_{n-1}}{2\Delta t} \quad (9.1.21)$$

$$a_n = \frac{d_{n+1} - 2d_n + d_{n-1}}{\Delta t^2} \quad (9.1.22)$$

the classical first and second central-difference approximations, respectively. Verify this assertion.

Exercise 3. The average acceleration method may also be derived by applying the trapezoidal rule to the first-order form of the equations of motion. Let

$$\mathbf{y} = \begin{Bmatrix} \mathbf{d} \\ \dot{\mathbf{d}} \end{Bmatrix} \quad (9.1.23)$$

and

$$f(\mathbf{y}, t) = \begin{Bmatrix} \dot{\mathbf{d}} \\ \mathbf{M}^{-1}(\mathbf{F}(t) - \mathbf{Cd} - \mathbf{Kd}) \end{Bmatrix} \quad (9.1.24)$$

Then

$$\dot{\mathbf{y}} = f(\mathbf{y}, t) \quad (9.1.25)$$

defines a first-order system of $2n_{eq}$ ordinary differential equations equivalent to (9.1.1). The trapezoidal-rule approximation of (9.1.25) may be written in terms of the following two equations:

$$z_{n+1} = f(y_{n+1}, t_{n+1}) \quad (9.1.26)$$

$$y_{n+1} = y_n + \frac{\Delta t}{2}(z_n + z_{n+1}) \quad (9.1.27)$$

Here y_n and z_n are the approximations of $y(t_n)$ and $\dot{y}(t_n)$. Show that (9.1.26) and (9.1.27) are equivalent to (9.1.4) through (9.1.6) with $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$.

To analyze the Newmark algorithm, we need to derive explicit formulas for A and L_n in (9.1.14). These may be obtained by eliminating a_n and a_{n+1} from

$$d_{n+1} = d_n + \Delta t v_n + \frac{\Delta t^2}{2}[(1 - 2\beta)a_n + 2\beta a_{n+1}] \quad (9.1.28)$$

$$v_{n+1} = v_n + \Delta t[(1 - \gamma)a_n + \gamma a_{n+1}] \quad (9.1.29)$$

by using

$$a_n + 2\xi\omega^h v_n + (\omega^h)^2 d_n = F_n \quad (9.1.30)$$

$$a_{n+1} + 2\xi\omega^h v_{n+1} + (\omega^h)^2 d_{n+1} = F_{n+1} \quad (9.1.31)$$

This leads to the following expressions:

$$A = A_1^{-1} A_2 \quad (9.1.32)$$

$$L_n = A_1^{-1} \begin{Bmatrix} \frac{\Delta t^2}{2} [(1 - 2\beta)F_n + 2\beta F_{n+1}] \\ \Delta t[(1 - \gamma)F_n + \gamma F_{n+1}] \end{Bmatrix} \quad (9.1.33)$$

$$A_1 = \begin{bmatrix} 1 + \Delta t^2 \beta (\omega^h)^2 & 2\Delta t^2 \beta \xi \omega^h \\ \Delta t \gamma (\omega^h)^2 & 1 + 2\Delta t \gamma \xi \omega^h \end{bmatrix} \quad (9.1.34)$$

$$A_2 = \begin{bmatrix} 1 - \frac{\Delta t^2}{2} (1 - 2\beta)(\omega^h)^2 & \Delta t(1 - \Delta t(1 - 2\beta)\xi \omega^h) \\ -\Delta t(1 - \gamma)(\omega^h)^2 & 1 - 2\Delta t(1 - \gamma)\xi \omega^h \end{bmatrix} \quad (9.1.35)$$

Convergence. The vector of *local truncation errors*, τ is defined by

$$y(t_{n+1}) = Ay(t_n) + L_n + \Delta t \cdot \tau(t_n) \quad (9.1.36)$$

where

$$y(t_n) = \begin{Bmatrix} d(t_n) \\ \dot{d}(t_n) \end{Bmatrix} \quad (9.1.37)$$

Obtaining an explicit expression for τ is tedious, due to the complicated nature of A and L_n . However, we can use an analysis similar to the one described in Chapter 8 to infer the correct exponent of Δt in the entries of τ . It may be argued that $\tau(t) = O(\Delta t^k)$ for all $t \in [0, T]$, where $k = 2$ if $\gamma = \frac{1}{2}$ and $k = 1$ otherwise.

The stability of the Newmark methods is determined by properties of the amplification matrix. The convergence proof follows along the lines of the theorem presented in Sec. 8.2.3. Here, the equation

$$e(t_n) = A^n e(0) - \Delta t \sum_{i=0}^{n-1} A^i \tau(t_{n-1-i}) \quad (9.1.38)$$

determined from (9.1.14) and (9.1.36), is used in place of the scalar analog in Sec.

8.2.3. Again, stability plus consistency implies convergence. The rate of convergence is k . This result holds for an arbitrary one-step, multivalue method. Multistep methods, which will be studied subsequently, may be recast as one-step multivalue methods although the dimension of the vector \mathbf{y} will generally be greater than 2.

Stability. Let $\lambda_i(\mathbf{A})$ denote the eigenvalues of \mathbf{A} . The modulus of $\lambda_i(\mathbf{A})$ is written $|\lambda_i(\mathbf{A})| = (\lambda_i(\mathbf{A})\lambda_i(\overline{\mathbf{A}}))^{1/2}$, where $\lambda_i(\overline{\mathbf{A}})$ is the complex conjugate of $\lambda_i(\mathbf{A})$. The *spectral radius* of \mathbf{A} , $\rho(\mathbf{A})$, is defined by

$$\boxed{\rho(\mathbf{A}) = \max_i |\lambda_i(\mathbf{A})|} \quad (9.1.39)$$

The stability condition is needed to prevent amplification of \mathbf{A}^n as n becomes large. The following conditions will be required:

- i. $\rho(\mathbf{A}) \leq 1$.
- ii. Eigenvalues of \mathbf{A} of multiplicity greater than one are strictly less than one in modulus.

Conditions (i) and (ii) define a *spectrally stable* \mathbf{A} .

That spectral stability bounds the growth of \mathbf{A}^n can be seen as follows. Consider a 2×2 matrix \mathbf{A} such as for the Newmark method. Then \mathbf{A} can be written as either

$$\mathbf{A} = \mathbf{P}\Lambda(\mathbf{A})\mathbf{P}^{-1}, \quad \Lambda(\mathbf{A}) = \begin{bmatrix} \lambda_1(\mathbf{A}) & 0 \\ 0 & \lambda_2(\mathbf{A}) \end{bmatrix} \quad (9.1.40)$$

or

$$\mathbf{A} = \mathbf{Q}\mathbf{J}(\mathbf{A})\mathbf{Q}^{-1}, \quad \mathbf{J}(\mathbf{A}) = \begin{bmatrix} \lambda(\mathbf{A}) & 1 \\ 0 & \lambda(\mathbf{A}) \end{bmatrix} \quad (9.1.41)$$

Equations (9.1.40) and (9.1.41) correspond to the cases in which \mathbf{A} has linearly independent eigenvectors and linearly dependent eigenvectors, respectively. Note that in (9.1.41) the eigenvalue $\lambda(\mathbf{A})$ has multiplicity two. The n th power of \mathbf{A} may be computed with the aid of (9.1.40) or (9.1.41):

$$\mathbf{A}^n = \mathbf{P}\Lambda(\mathbf{A})^n\mathbf{P}^{-1}, \quad \Lambda(\mathbf{A})^n = \begin{bmatrix} \lambda_1(\mathbf{A})^n & 0 \\ 0 & \lambda_2(\mathbf{A})^n \end{bmatrix} \quad (9.1.42)$$

$$\mathbf{A}^n = \mathbf{Q}\mathbf{J}(\mathbf{A})^n\mathbf{Q}^{-1}, \quad \mathbf{J}(\mathbf{A})^n = \begin{bmatrix} \lambda(\mathbf{A})^n & n\lambda(\mathbf{A})^{n-1} \\ 0 & \lambda(\mathbf{A})^n \end{bmatrix} \quad (9.1.43)$$

In the former case \mathbf{A}^n will be bounded as long as $\rho(\mathbf{A}) = \max\{|\lambda_1(\mathbf{A})|, |\lambda_2(\mathbf{A})|\} \leq 1$. In the latter case we need to invoke the stronger condition $|\lambda(\mathbf{A})| < 1$ so that the off-diagonal term does not exhibit growth. Observe that $|\lambda(\mathbf{A})| < 1$ implies $|n\lambda(\mathbf{A})^{n-1}| \rightarrow 0$ as $n \rightarrow \infty$, whereas if $|\lambda(\mathbf{A})|$ was allowed to be equal to one, then

$n|\lambda(\mathbf{A})|^{n-1} = n$. Thus the necessity of condition (ii) in the definition of spectral stability can be appreciated.

Violations of condition (i) produce “explosive” instabilities, which grow like $\rho(\mathbf{A})^n$ (cf. Table 8.2.1). If (i) is in force but (ii) is violated, the growth is much weaker (e.g., $O(n)$ as above). These are referred to as *weak instabilities*. In the general case when the multiplicity of a modulus-one eigenvalue is m , the growth rate is $O(n^{m-1})$.

By virtue of the fact that the stability condition depends only upon the eigenvalues of \mathbf{A} , it can be expressed in terms of the *principal invariants* of \mathbf{A} . For a 2×2 amplification matrix, we have that

$$0 = \det(\mathbf{A} - \lambda(\mathbf{A})\mathbf{I}) = \lambda(\mathbf{A})^2 - 2A_1\lambda(\mathbf{A}) + A_2 \quad (9.1.44)$$

where

$$A_1 = \frac{1}{2}\text{trace } \mathbf{A} = \frac{1}{2}(A_{11} + A_{22}) \quad (9.1.45)$$

$$A_2 = \det \mathbf{A} = A_{11}A_{22} - A_{12}A_{21} \quad (9.1.46)$$

are the principal invariants. The eigenvalues are thus

$$\lambda_{1,2}(\mathbf{A}) = A_1 \pm (A_1^2 - A_2)^{1/2} \quad (9.1.47)$$

Exercise 4. For the Newmark method, use (9.1.32), (9.1.34), and (9.1.35) to show that

$$A_1 = 1 - \frac{[\xi\Omega + \Omega^2(\gamma + \frac{1}{2})/2]}{D} \quad (9.1.48)$$

$$A_2 = 1 - \frac{[2\xi\Omega + \Omega^2(\gamma - \frac{1}{2})]}{D} \quad (9.1.49)$$

where

$$D = 1 + 2\gamma\xi\Omega + \beta\Omega^2 \quad (9.1.50)$$

$$\Omega = \omega^h\Delta t \quad (9.1.51)$$

For a 2×2 amplification matrix, the most convenient characterization of spectral stability has been devised by Hilber [4] (see also [5] for the derivation): The stability region in A_1 , A_2 -space satisfies (see Fig. 9.1.1)

$$\frac{-(A_2 + 1)}{2} \leq A_1 \leq \frac{(A_2 + 1)}{2}, \quad -1 \leq A_2 < 1 \quad (9.1.52)$$

$$-1 < A_1 < 1, \quad A_2 = 1 \quad (9.1.53)$$

Combining (9.1.48) through (9.1.53) leads to the stability conditions quoted previously, namely (9.1.16) through (9.1.20).

High-frequency behavior. Because the higher modes of semidiscrete structural equations are artifacts of the discretization process and not representative of the behavior of the governing partial differential equations, it is generally viewed as desirable and often is considered absolutely necessary to have some form of algorithmic damping present to remove the participation of the high-frequency modal

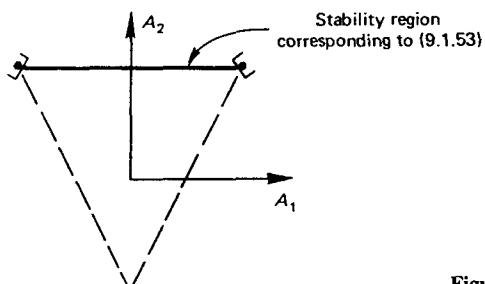
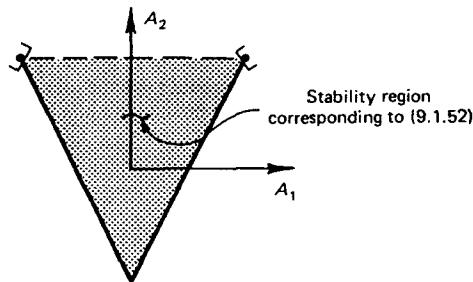


Figure 9.1.1

components. In terms of the Newmark method, $\gamma > \frac{1}{2}$ is necessary to introduce high-frequency dissipation. For a fixed $\gamma > \frac{1}{2}$, one can select β such that high-frequency dissipation is maximized. It turns out that this condition is created by restricting the eigenvalues of the amplification matrix to be complex conjugate. From (9.1.47) it can be seen that

$$A_1^2 < A_2 \quad (9.1.54)$$

is required. The subregion of the spectrally stable region of A_1, A_2 -space is shown in Fig. 9.1.2. The Newmark methods fall within this region if:

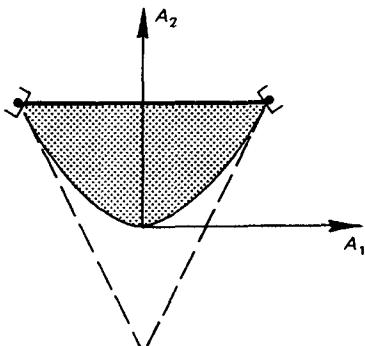


Figure 9.1.2 Stability region for which roots of the amplification matrix are complex conjugates.

Unconditional

$$0 \leq \xi < 1, \quad \gamma \geq \frac{1}{2}, \quad \beta \geq \frac{(\gamma + 1/2)^2}{4} \quad (9.1.55)$$

Conditional

$$0 \leq \xi < 1, \quad \gamma \geq \frac{1}{2}, \quad \Omega < \Omega_{\text{bif}} \quad (9.1.56)$$

where

$$\Omega_{\text{bif}} = \frac{\xi(\gamma - \frac{1}{2})/2 + [(\gamma + \frac{1}{2})^2/4 - \beta + \xi^2(\beta - \gamma/2)]^{1/2}}{(\gamma + \frac{1}{2})^2/4 - \beta} \quad (9.1.57)$$

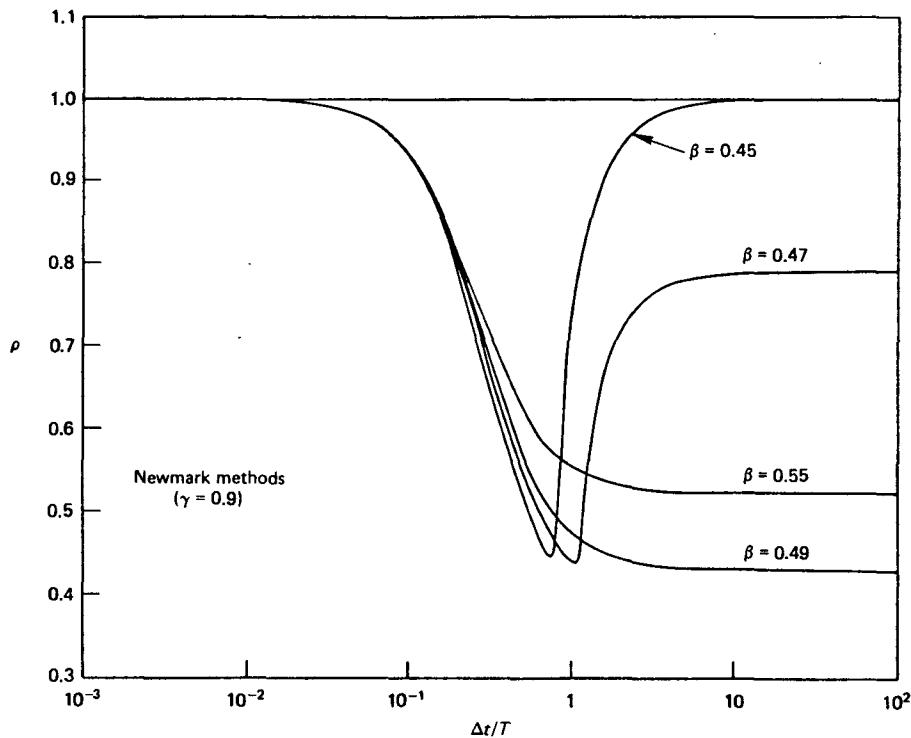
In the undamped case, (9.1.57) becomes simply

$$\Omega_{\text{bif}} = \left[\frac{(\gamma + 1/2)^2}{4} - \beta \right]^{-1/2} \quad (9.1.58)$$

Ω_{bif} is the value of Ω at which complex conjugate eigenvalues bifurcate into real, distinct eigenvalues. Figure 9.1.3 illustrates the significance of this condition. The high modes decay like $\rho(A)^n$. Therefore the minimum value of $\rho_\infty = \lim_{\Delta t/T \rightarrow \infty} \rho(A)$ is the most effective in filtering out the spurious high frequencies. This is achieved by selecting $\beta = (\gamma + \frac{1}{2})^2/4$ (i.e., $0.49 = (0.9 + 0.5)^2/4$). Increasing or decreasing β reduces ρ_∞ . The minimum value of β that retains unconditional stability is $\beta = \gamma/2 = 0.45$. However, as may be seen from Fig. 9.1.3, the high modes are not damped because $\rho_\infty = 1$. For values of β such that $\gamma/2 < \beta < (\gamma + \frac{1}{2})^2/4$, the eigenvalues of A bifurcate and become real beyond some $\Delta t/T$. This occurs at the minimum points of the $\beta = 0.47$ and 0.45 curves in Fig. 9.1.3. For $\beta = 0.55$, no bifurcation occurs. It may be deduced from this discussion that the ideal value of β is indeed $\beta = (\gamma + \frac{1}{2})^2/4$.

Viscous damping. It may seem that an obvious damping mechanism—viscous damping—has been neglected in this discussion. The reason for this is that viscous damping can be shown to damp an intermediate band of frequencies without significant effect in the all-important high modes [5].

In introducing high-frequency dissipation by selecting $\gamma > \frac{1}{2}$, it is, of course, desirable to maintain good accuracy in the low modes. Unfortunately $\gamma \neq \frac{1}{2}$ results in a drop to first-order accuracy, and for this reason there has been considerable research into the development of alternative methods for algorithmically damping the higher modes.

Figure 9.1.3 Spectral radii for Newmark methods for varying β [9].

Exercise 5. Consider the second-order ordinary differential equation

$$\ddot{d} + (\omega^h)^2 d = 0$$

The generalized trapezoidal method (see Sec. 8.1) for this equation consists of the following:

$$\begin{aligned} a_{n+1} + (\omega^h)^2 d_{n+1} &= 0 \\ d_{n+1} &= d_n + \Delta t \{(1 - \alpha)v_n + \alpha v_{n+1}\} \\ v_{n+1} &= v_n + \Delta t \{(1 - \alpha)a_n + \alpha a_{n+1}\} \end{aligned}$$

Set up the 2×2 amplification matrix for this method and determine the stability behavior and the order of accuracy for all $\alpha \in [0, 1]$. In particular, comment on the stability of the forward Euler ($\alpha = 0$) and backward Euler ($\alpha = 1$) methods. [Ans.: $\alpha \geq \frac{1}{2}$, unconditionally stable; $\alpha < \frac{1}{2}$, unconditionally unstable; $\alpha = \frac{1}{2}$, second-order accurate, otherwise first-order accurate.]

Exercise 6. Consider the trapezoidal rule (i.e., $\gamma = \frac{1}{2}$, $\beta = \frac{1}{4}$) and assume $F = 0$. Show that as $\Omega \rightarrow \infty$,

$$d_{n+1} \rightarrow -d_n$$

$$v_{n+1} \rightarrow -v_n$$

$$a_{n+1} \rightarrow -a_n$$

and thus

$$d_n = (-1)^n d_0$$

$$v_n = (-1)^n v_0$$

$$a_n = (-1)^n a_0$$

Remark

The results of the preceding exercise indicate that spurious high-frequency components may be filtered for output purposes by reporting step-to-step averages, e.g., $d_{n+1/2} = (d_{n+1} + d_n)/2$; cf. Sec. 8.2.2.

Exercise 7. Consider unconditionally stable Newmark schemes (i.e., those schemes for which $2\beta \geq \gamma \geq \frac{1}{2}$). Let ρ_∞ denote the asymptotic value of the spectral radius of A as $\Omega \rightarrow \infty$.

- i. Show that if $2\beta = \gamma$, $\rho_\infty = 1$.
 - ii. Let $\beta = (\gamma + \frac{1}{2})^2/4$ and determine an expression for ρ_∞ . Call this value $\bar{\rho}_\infty$. [Ans.: $\bar{\rho}_\infty = |1 - 2/(\gamma + \frac{1}{2})|$.]
 - iii. Show that if $\gamma/2 \leq \beta < (\gamma + \frac{1}{2})^2/4$, $\rho_\infty > \bar{\rho}_\infty$.
 - iv. Show that if $\beta > (\gamma + \frac{1}{2})^2/4$, $\rho_\infty > \bar{\rho}_\infty$.
 - v. Determine values of γ and β such that $\rho_\infty = 0$. [Ans.: From part (ii), $\beta = 1$ and $\gamma = \frac{3}{2}$.]
-

Remark

The preceding exercise analytically establishes the trends shown in Figure 9.1.3 and shows in particular that maximal high-frequency numerical dissipation is provided by picking $\beta = (\gamma + \frac{1}{2})^2/4$ for a given $\gamma > \frac{1}{2}$. In addition, it shows that the algorithm $\beta = 1$ and $\gamma = \frac{3}{2}$ provides the strongest possible high-frequency dissipation. Unfortunately, this algorithm achieves very poor accuracy in the low modes and thus it is not useful for transient analysis. However, the strong damping characteristics may be exploited in quickly bringing a transient solution to equilibrium.

Exercise 8. Consider the following predictor-corrector algorithm:

$$\left. \begin{aligned} \widetilde{d}_{n+1} &= d_n + \Delta t v_n + \frac{\Delta t^2}{2} a_n \\ \widetilde{a}_{n+1} + (\omega^h)^2 \widetilde{d}_{n+1} &= 0 \end{aligned} \right\} \text{predictors}$$

$$\left. \begin{aligned} d_{n+1} &= d_n + \Delta t v_n + \frac{\Delta t^2}{2} \{(1 - 2\beta)a_n + 2\beta \widetilde{a}_{n+1}\} \\ v_{n+1} &= v_n + \Delta t \{(1 - \gamma)a_n + \gamma \widetilde{a}_{n+1}\} \\ a_{n+1} + (\omega^h)^2 d_{n+1} &= 0 \end{aligned} \right\} \text{correctors}$$

Assume $\gamma = \frac{1}{2}$.

- (A) Determine the order of accuracy.
 (B) Determine an expression for the critical time step in terms of β and ω^h .

Solution

$$(A) \quad \begin{array}{l} \tilde{d}_{n+1} = d_n + \Delta t v_n + \frac{\Delta t^2}{2} a_n \\ \downarrow \\ \tilde{a}_{n+1} + (\omega^h)^2 \tilde{d}_{n+1} = 0 \end{array} \quad (a)$$

$$\boxed{\begin{array}{l} d_{n+1} = d_n + \Delta t v_n + \frac{\Delta t^2}{2} \{(1 - 2\beta)a_n + 2\beta \tilde{a}_{n+1}\} \\ \downarrow \\ v_{n+1} = v_n + \frac{\Delta t}{2} \{a_n + \tilde{a}_{n+1}\} \end{array}} \quad (c)$$

$$(d) \quad v_{n+1} = v_n + \frac{\Delta t}{2} \{a_n + \tilde{a}_{n+1}\}$$

$$(e) \quad a_{n+1} + (\omega^h)^2 d_{n+1} = 0$$

The truncation errors contributed by (a) through (e) are

$$\tau_{(a)} = O(\Delta t^2), \quad \tau_{(b)} = 0, \quad \tau_{(c)} = O(\Delta t^2), \quad \tau_{(d)} = O(\Delta t^2), \quad \tau_{(e)} = 0$$

Therefore, on combining the equations as indicated above, it is clear that the algorithm is *second-order accurate*.

$$(B) \quad \begin{Bmatrix} d_{n+1} \\ \Delta t v_{n+1} \end{Bmatrix} = A \begin{Bmatrix} d_n \\ \Delta t v_n \end{Bmatrix}$$

Absorbing Δt into the velocity degrees of freedom has no effect on the stability calculation. Eliminating \tilde{d}_{n+1} , \tilde{a}_{n+1} , and a_n , we get

$$A = \begin{bmatrix} 1 - \frac{\Omega^2}{2} + \beta \frac{\Omega^4}{2} & 1 - \beta \Omega^2 \\ -\Omega^2 + \gamma \frac{\Omega^4}{2} & 1 - \gamma \Omega^2 \end{bmatrix}$$

$$2A_1 = \text{tr } A = 2 - \left(\frac{1}{2} + \gamma\right)\Omega^2 + \frac{\beta}{2}\Omega^4$$

$$A_2 = \det A = 1 + \left(\frac{1}{2} - \gamma\right)\Omega^2 - \frac{\beta}{2}\Omega^4$$

Stability conditions:

$$\text{I. } \begin{cases} \text{(i)} \quad -(A_2 + 1) \leq 2A_1 \leq A_2 + 1 \\ \text{(ii)} \\ \text{(iii)} \quad -1 \leq A_2 < 1 \\ \text{(iv)} \end{cases}$$

or

$$\text{II. } \begin{cases} \text{(i)} \quad -1 < A_1 < 1 \\ \text{(ii)} \\ \text{if } A_2 = 1 \end{cases}$$

$$\text{Ii. } -2 - \left(\frac{1}{4} - \gamma\right)\Omega^2 + \frac{\beta}{2}\Omega^4 \leq 2 - \left(\frac{1}{4} + \gamma\right)\Omega^2 + \frac{\beta}{2}\Omega^4$$

$$\gamma\Omega^2 \leq 2 \quad \left(\Omega^2 \leq 4 \text{ for } \gamma = \frac{1}{2}\right)$$

$$\text{Iii. } \cancel{-} \left(\frac{1}{2} + \cancel{\gamma}\right)\Omega^2 + \frac{\beta}{2}\Omega^4 \leq \cancel{+} \left(\frac{1}{2} - \cancel{\gamma}\right)\Omega^2 - \frac{\beta}{2}\Omega^4$$

$$\beta\Omega^4 \leq \Omega^2$$

$$\beta\Omega^2 \leq 1$$

$$\text{Iiii. } -1 \leq 1 + \left(\frac{1}{2} - \gamma\right)\Omega^2 - \frac{\beta}{2}\Omega^4 \quad \left(\beta\Omega^4 \leq 4 \text{ for } \gamma = \frac{1}{2}\right)$$

$$\text{Iiv. } 1 + \left(\frac{1}{2} - \gamma\right)\Omega^2 - \frac{\beta}{2}\Omega^4 < 1 \quad \left(\beta > 0 \text{ for } \gamma = \frac{1}{2}\right)$$

IIIi. $\left(\text{Note } \gamma = \frac{1}{2} \text{ and } A_2 = 1 \Rightarrow \beta = 0\right)$

$$-1 < \frac{1}{2}(2 - \Omega^2)$$

$$\Omega^2 < 4$$

$$\text{IIIi. } \frac{1}{2}(2 - \Omega^2) < 1 \quad (\text{automatic})$$

Summary ($\gamma = \frac{1}{2}$):

- $\beta \geq 0$
 - If $\beta = 0$, $\Omega^2 < 4$
 - If $\beta > 0$,
- $$\Omega^2 \leq \min \left\{ \left(\frac{4}{\beta} \right)^{1/2}, \frac{1}{\beta}, 4 \right\}$$

Note. If $\beta = \frac{1}{4}$, $\Omega^2 \leq 4$, which is the same as for the central difference method.

9.1.3 Measures of Accuracy: Numerical Dissipation and Dispersion

In the present application it is useful to introduce accuracy measures, other than local truncation error, which measure numerical dissipation and dispersion. To motivate these measures observe that the continuous SDOF model problem can be exactly solved:

$$d(t) = e^{-\xi\omega^h t} (d_0 \cos \omega_d^h t + c \sin \omega_d^h t) \quad (9.1.59)$$

where

$$c = \frac{v_0 + \xi\omega^h d_0}{\omega_d^h} \quad (9.1.60)$$

$$\omega_d^h = (1 - \xi^2)^{1/2} \omega^h \quad (\text{damped natural frequency}) \quad (9.1.61)$$

In arriving at (9.1.59) through (9.1.61), we have assumed the homogeneous and “underdamped” case (i.e., $F = 0$ and $0 \leq \xi < 1$, respectively). Likewise, the discrete solution can be put in the following similar form:

$$d_n = e^{-\bar{\xi}\bar{\omega}^h t_n} (d_0 \cos \bar{\omega}_d^h t_n + \bar{c} \sin \bar{\omega}_d^h t_n) \quad (9.1.62)$$

$$\bar{c} = \frac{d_1 - A_1 d_0}{(A_2 - A_1^2)^{1/2}} = \frac{\frac{1}{2}(A_{11} - A_{22})d_0 + A_{12}v_0}{(A_2 - A_1^2)^{1/2}} \quad (9.1.63)$$

$$\bar{\omega}_d^h = (1 - \xi^2)^{1/2} \bar{\omega}^h \quad (9.1.64)$$

Details of the derivation may be found in Hughes [5]. Comparing (9.1.59) and (9.1.62), we see that $\bar{\xi}$ and $\bar{\omega}^h$ are the algorithmic counterparts of ξ and ω^h , respectively. The measures of numerical dissipation and dispersion chosen are $\bar{\xi}$, the *algorithmic damping ratio*, and $(\bar{T} - T)/T$, the *relative period error*, where $\bar{T} = 2\pi/\bar{\omega}^h$ and $T = 2\pi/\omega^h$. These measures have some advantages over others that have been introduced in the literature (see [5] for elaboration). It is difficult analytically to obtain expressions for $\bar{\xi}$ and $(\bar{T} - T)/T$. Consequently, computer evaluation is usually necessary. Nevertheless, a few analytical results have been obtained for stable Newmark methods, which we collect here:

$$\bar{\xi} = \xi + \frac{1}{2} \left(\gamma - \frac{1}{2} \right) \underbrace{[\Omega + O(\Omega^2)]}_{\geq 0} \quad (9.1.65)$$

$$\frac{\bar{T} - T}{T} = O(\Omega^2) \quad (9.1.66)$$

Results (9.1.65) and (9.1.66) illustrate that first-order errors created by $\gamma \neq \frac{1}{2}$ enter only as dissipation errors and not period errors. Furthermore, $\gamma = \frac{1}{2}$ leads to no numerical dissipation, whereas values of $\gamma > \frac{1}{2}$ create numerical dissipation. Later on we shall graphically compare the Newmark method with other commonly used step-by-step methods on the basis of numerical dissipation and dispersion.

Let $t_n = \bar{T}$. Then

$$\begin{aligned} \exp(-\bar{\xi}\bar{\omega}^h t_n) &= \exp(-2\pi\bar{\xi}) \\ &\cong 1 - 2\pi\bar{\xi} + \dots \end{aligned} \quad (9.1.67)$$

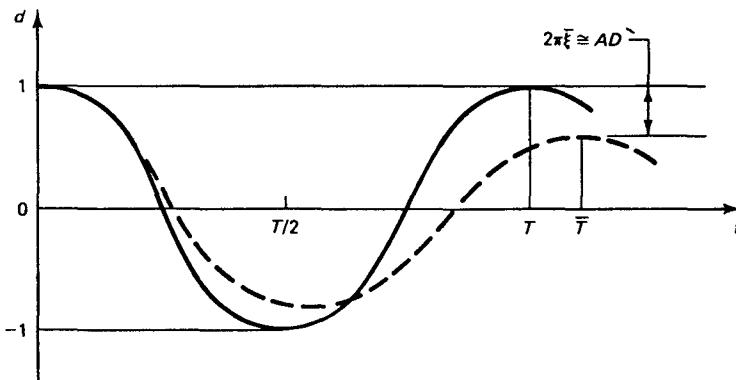
and so, after one period, for small $\bar{\xi}$, the *amplitude decay* is

$$AD \cong 2\pi\bar{\xi} \quad (9.1.68)$$

The accuracy measures are illustrated in the figure on page 506.

9.1.4 Matched Methods

Let us consider an elastodynamics calculation in which a member of the Newmark family of algorithms is to be employed. Assume $\xi = 0$ and $\gamma = \frac{1}{2}$, which implies $\bar{\xi} = 0$ (i.e., no amplitude decay). The periods of individual modes of the finite element model will, however, be distorted by the particular integrator. For example, if the trapezoidal rule is employed ($\beta = \frac{1}{4}$), periods (frequencies, respectively) will be



elongated (decreased, respectively). On the other hand, if the central difference method is employed ($\beta = 0$), periods (frequencies, respectively) will be shortened (increased, respectively).¹

It is important to keep in mind that the periods of the finite element model (i.e., “spatially discrete system”) are already in error when compared with the original, “exact” problem. We recall that if the consistent mass matrix is employed, an upper (lower, respectively) bound property may be established for the finite element frequencies (periods, respectively). Experience has indicated that lumped mass matrices tend to behave in the opposite fashion.

These observations suggest that transient integrators and mass matrices should be “matched” so that the induced period errors may tend to cancel [3]. For example, trapezoidal rule and consistent mass, and central differences and lumped mass, would be appropriate matches in that sense.

The following example, which is taken from [3], makes these ideas precise in the context of a model problem.

Example

Consider a finite element model for the wave equation²

$$u_{,tt} = u_{,xx} \quad (9.1.69)$$

consisting of n_{el} linear elements of equal length $h = 1/n_{el}$. Recall that in this case the element stiffness is given by (we drop the e superscript since all elements are the same):

$$k = \frac{1}{h} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad (9.1.70)$$

A one-parameter (r) family of element mass matrices discussed in Chapter 7 may be written as

¹ See Fig. 9.3.4.

² We may think of (9.1.69) as governing the transient response of a linear elastic rod in which the density, ρ , and Young’s modulus, E , are chosen to attain unit speed of propagation. This simplifies the development somewhat. The general case is described in Remark 1 at the end of this section.

$$\mathbf{m}_{(r)} = h \begin{bmatrix} \frac{1}{2} - r & r \\ r & \frac{1}{2} - r \end{bmatrix} \quad (9.1.71)$$

Recall that particular values of r yield the commonly used mass matrices, e.g.,

- i. *consistent mass* ($r = \frac{1}{6}$)

$$\mathbf{m}_{(1/6)} = \frac{h}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad (9.1.72)$$

- ii. *lumped mass* ($r = 0$)

$$\mathbf{m}_{(0)} = \frac{h}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (9.1.73)$$

- iii. *higher-order mass* ($r = \frac{1}{12}$)

$$\mathbf{m}_{(1/12)} = \frac{h}{12} \begin{bmatrix} 5 & 1 \\ 1 & 5 \end{bmatrix} \quad (9.1.74)$$

In what follows we will have to distinguish carefully between three different frequency parameters, defined as follows:

ω = exact frequency

ω^h = frequency produced by finite element spatial discretization

$\bar{\omega}^h$ = frequency produced by time integration algorithm in conjunction with finite element spatial discretization

Both the finite element spatial discretization and time integration algorithm introduce errors. In practice we see $\bar{\omega}^h$, but we would like to see ω .

The finite element matrix equation corresponding to (9.1.69) is obtained by assembling (9.1.70) and (9.1.71) in the usual way. The result is

$$\mathbf{M}_{(r)} \ddot{\mathbf{d}} + \mathbf{Kd} = 0 \quad (9.1.75)$$

Let $d_A(t)$ denote the displacement at node number A , and let

$$\mathcal{O}d_A = d_{A+1} - 2d_A + d_{A-1} \quad (9.1.76)$$

\mathcal{O} is called the (undivided) second central difference operator. (We assume that the nodes are labeled in ascending order from left to right.) If A represents an interior node, the scalar equation in (9.1.75) associated with node A may be written as

$$h(1 + r\mathcal{O})\ddot{d}_A - \frac{1}{h}\mathcal{O}d_A = 0 \quad (9.1.77)$$

Our aim is to solve (9.1.77) *exactly*. To this end we assume the solution has the form

$$d_A(t) = S_A T(t) \quad (9.1.78)$$

Substituting (9.1.78) in (9.1.77) and adding and subtracting $(\omega^h)^2 T(1 + r\mathcal{O})S_A$ yields an equation convenient for subsequent developments, namely,

$$[\ddot{T} + (\omega^h)^2 T](1 + r\mathcal{O})S_A - \left[(\omega^h)^2 (1 + r\mathcal{O})S_A + \frac{1}{h^2} \mathcal{O}S_A \right] T = 0 \quad (9.1.79)$$

Satisfaction of (9.1.79) is achieved by selecting S_A and T such that

$$[(\omega^h h)^2(1 + r\mathcal{C}) + \mathcal{C}]S_A = 0 \quad (9.1.80)$$

and

$$\ddot{T} + (\omega^h)^2 T = 0 \quad (9.1.81)$$

We assume the general solution of (9.1.80) takes the form

$$S_A = c_1 \sin \frac{A\lambda}{n_{el}} + c_2 \cos \frac{A\lambda}{n_{el}} \quad (9.1.82)$$

Permissible values of λ are determined by imposing homogeneous boundary conditions. The results are

$$\left. \begin{array}{l} \text{Fixed-fixed} \\ \text{Free-free} \end{array} \right\} \quad \lambda = l\pi \quad (9.1.83)$$

$$\left. \begin{array}{l} \text{Free-fixed} \\ \text{Fixed-free} \end{array} \right\} \quad \lambda = \frac{(2l - 1)\pi}{2} \quad (9.1.84)$$

where l is an integer. It is easily shown that the above values of λ coincide with the exact frequencies, ω , obtained by solving the eigenvalue problem associated with (9.1.69).

We shall provide a detailed study of the fixed-fixed case in what follows. The other cases may be carried out analogously and the interested reader may wish to provide the details as an exercise.

A consequence of the boundary conditions in the fixed-fixed case is that $c_2 = 0$ in (9.1.82). Under these circumstances, (9.1.80) becomes

$$(\omega^h h)^2 \sin \frac{A\omega}{n_{el}} + [1 + r(\omega^h h)^2] \left(\sin \frac{(A - 1)\omega}{n_{el}} - 2 \sin \frac{A\omega}{n_{el}} + \sin \frac{(A + 1)\omega}{n_{el}} \right) = 0 \quad (9.1.85)$$

Making use of the trigonometric identity $\sin(a \pm b) = \sin a \cos b \pm \sin b \cos a$ in (9.1.85) results in

$$\left(\frac{\omega^h h}{2} \right)^2 \left[1 + 2r \left(\cos \frac{\omega}{n_{el}} - 1 \right) \right] + \frac{1}{2} \left(\cos \frac{\omega}{n_{el}} - 1 \right) = 0 \quad (9.1.86)$$

The half-angle formula, $\sin^2 a/2 = (1 - \cos a)/2$, can be used to simplify (9.1.86), viz.,

$$\left(\frac{\omega^h h}{2} \right)^2 \left[1 - 4r \sin^2 \frac{\omega}{2n_{el}} \right] - \sin^2 \frac{\omega}{2n_{el}} = 0 \quad (9.1.87)$$

which, upon replacing n_{el} by $1/h$, leads to

$$\frac{\omega^h}{\omega} = \frac{\sin \omega h/2}{(\omega h/2)[1 - 4r \sin^2(\omega h/2)]^{1/2}} \quad (9.1.88)$$

The other boundary conditions result in identical expressions. Equation (9.1.88) is plotted in Fig. 9.1.4 for the three cases cited previously.

We now turn our attention to a frequency expression for the Newmark method, derived in [5], in which we assume $\xi = 0$ and $\gamma = \frac{1}{2}$, viz.,

$$\bar{\omega}^h \Delta t = \arctan \left[\frac{(A_2 - A_1^2)^{1/2}}{A_1} \right] \quad (9.1.89)$$

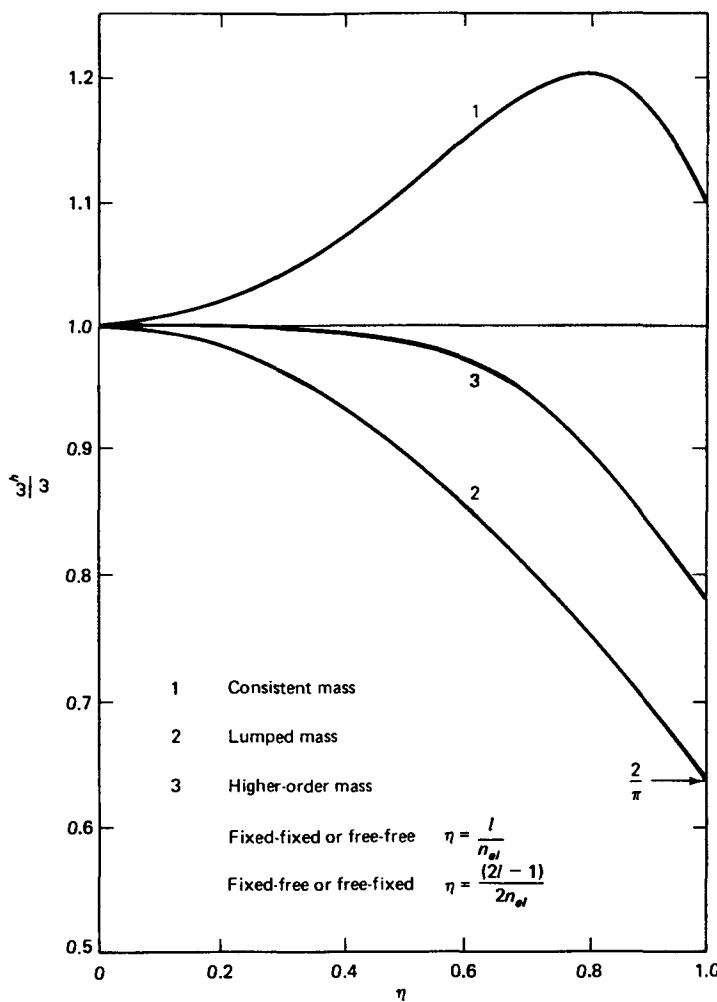


Figure 9.1.4 Error in discrete spectrum of simple bar [2].

where

$$\left. \begin{aligned} A_1 &= 1 - \frac{(\omega^h \Delta t)^2}{[2(1 + \beta(\omega^h \Delta t)^2)]} \\ A_2 &= 1 \end{aligned} \right\} \quad (9.1.90)$$

For reasons which will become obvious, we wish to obtain an expression for $\sin^2(\bar{\omega}^h \Delta t / 2)$. To do this we make use of the trigonometric identities $\cot a = \cot 2a + \csc 2a$ and $\csc a = (\cot^2 a + 1)^{1/2}$. These, (9.1.89), and (9.1.90) enable us to write

$$\cot^2 \frac{\bar{\omega}^h \Delta t}{2} = \frac{1 + A_1}{1 - A_1} = -1 + 4\beta + \left(\frac{2}{\omega^h \Delta t} \right)^2 \quad (9.1.91)$$

Now we use the identity $\sin^2 a = 1/(1 + \cot^2 a)$ to achieve the desired result

$$\sin^2 \frac{\bar{\omega}^h \Delta t}{2} = \frac{1}{4\beta + (2/\omega^h \Delta t)^2} \quad (9.1.92)$$

Equation (9.1.88) is substituted into (9.1.92) to obtain the main result:

$$\sin^2 \frac{\bar{\omega}^h \Delta t}{2} = \frac{\sin^2(\omega h/2)}{4[\beta - r(h/\Delta t)^2] \sin^2(\omega h/2) + (h/\Delta t)^2} \quad (9.1.93)$$

Note that if $h = \Delta t$ and $\beta = r$, then (9.1.93) implies that $\bar{\omega}^h = \omega$, i.e., *the errors introduced by finite element spatial discretization, the particular mass matrix and temporal algorithm all cancel to yield exact results*. The time step $\Delta t = h$ is called the **characteristic time step**. It is equal to the transit time for a “wave” moving at unit speed to traverse one element.

Remarks

1. If equation (9.1.69) is replaced by

$$u_{,tt} = c^2 u_{,xx} \quad (9.1.94)$$

where $c^2 = E/\rho$, then h should be replaced by h/c in (9.1.93). c is called the **characteristic velocity, or speed of sound**; it represents the propagation speed of solutions of (9.1.94). (See [6] for basic results on wave propagation.) Thus if we compute at the characteristic time step, $\Delta t = h/c$, and select $\beta = r$, the numerical results will be exact at the nodes no matter how few elements are employed (“superconvergence”). The combination $\beta = r = 0$ (i.e., central difference method and lumped mass) is the simplest in practice. We note also that for $r = 0$, as $\beta \rightarrow 0^+$, we approach the exact solution. Numerical results along these lines are presented in [7] and Fig. 9.1.5.

2. It is interesting to note that reducing the size of the time step while holding the mesh length fixed can only *worsen* the results. In this case we converge to the exact solution of the spatially discrete, temporally continuous system (i.e., “mass points and springs”) rather than the exact solution of (9.1.94). The characteristic time step also represents the stability limit (i.e., critical time step) of the lumped mass, central difference method, and, therefore, computations cannot be performed at larger time steps. For these reasons, it is generally considered advantageous to compute at a time step as close to critical as possible.

3. So that there is no chance of confusion, we wish to emphasize that in more general settings (e.g., unequal element lengths, variable material properties, multi-dimensional problems, etc.), results obtained by matched methods, such as central differences and lumped mass, will not be exact. However, it is felt that results obtained by matched methods will generally be superior to inappropriate combinations, such as consistent mass and central differences.

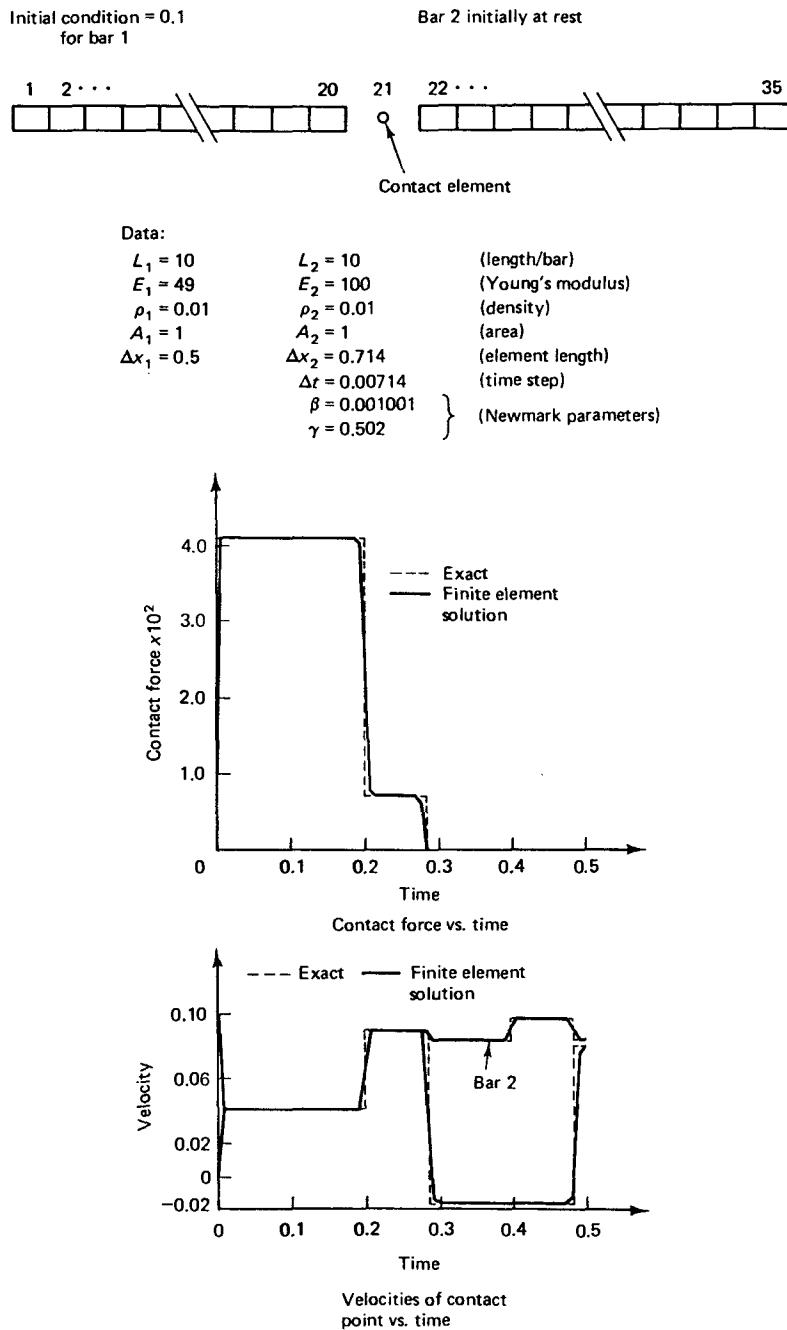


Figure 9.1.5 Impact of two dissimilar bars.

9.1.5 Additional Exercises

Exercise 9. Show that the trapezoidal rule (i.e., $\beta = \frac{1}{4}$, $\gamma = \frac{1}{2}$) exactly conserves total energy when $C = \mathbf{0}$ and $F = \mathbf{0}$.

Solution

In terms of undivided forward-difference and mean-value operators (i.e., $[]$ and $\langle \rangle$, respectively, see Sec. 8.2.4), the Newmark formulas, namely, (9.1.5) and (9.1.6), may be written as

$$[\mathbf{v}_n] = \Delta t \mathbf{a}_{n+\gamma} \quad (9.1.95)$$

$$[\mathbf{d}_n] = \Delta t \langle \mathbf{v}_n \rangle + \Delta t^2 (\beta - \gamma/2) [\mathbf{a}_n] \quad (9.1.96)$$

The *total energy* is defined by

$$E(\mathbf{d}_n, \mathbf{v}_n) = T(\mathbf{v}_n) + U(\mathbf{d}_n) \quad (9.1.97)$$

where

$$T(\mathbf{v}_n) = \frac{1}{2} \mathbf{v}_n^T M \mathbf{v}_n \quad (\text{kinetic energy}) \quad (9.1.98)$$

$$U(\mathbf{d}_n) = \frac{1}{2} \mathbf{d}_n^T K \mathbf{d}_n \quad (\text{strain energy}) \quad (9.1.99)$$

For future use, note that (9.1.4) and the hypotheses result in

$$\mathbf{M}[\mathbf{a}_n] + \mathbf{K}[\mathbf{d}_n] = \mathbf{0} \quad (9.1.100)$$

$$\mathbf{M}\langle \mathbf{a}_n \rangle + \mathbf{K}\langle \mathbf{d}_n \rangle = \mathbf{0} \quad (9.1.101)$$

Collecting results, we obtain the following:

$$\begin{aligned}
 [T(\mathbf{v}_n)] &= [\mathbf{v}_n]^T M \langle \mathbf{v}_n \rangle \quad (\text{by } M = M^T) \\
 &= \mathbf{a}_{n+\gamma}^T M \left([\mathbf{d}_n] - \Delta t^2 \left(\beta - \frac{\gamma}{2} \right) [\mathbf{a}_n] \right) \\
 &= \left(\langle \mathbf{a}_n \rangle + \left(\gamma - \frac{1}{2} \right) [\mathbf{a}_n] \right)^T M [\mathbf{d}_n] - \Delta t^2 \left(\beta - \frac{\gamma}{2} \right) [\mathbf{a}_n] \\
 &= -\langle \mathbf{d}_n \rangle^T K [\mathbf{d}_n] - \left(\gamma - \frac{1}{2} \right) [\mathbf{d}_n]^T K [\mathbf{d}_n] \\
 &\quad - \Delta t^2 \left(\beta - \frac{\gamma}{2} \right) \left([T(\mathbf{a}_n)] + 2 \left(\gamma - \frac{1}{2} \right) T([\mathbf{a}_n]) \right) \\
 &= -[U(\mathbf{d}_n)] - 2 \left(\gamma - \frac{1}{2} \right) U([\mathbf{d}_n]) \\
 &\quad - \Delta t^2 \left(\beta - \frac{\gamma}{2} \right) \left([T(\mathbf{a}_n)] + 2 \left(\gamma - \frac{1}{2} \right) T([\mathbf{a}_n]) \right) \quad (9.1.102)
 \end{aligned}$$

Therefore

$$\begin{aligned} [T(v_n)] + [U(d_n)] + \Delta t^2 \left(\beta - \frac{\gamma}{2} \right) [T(a_n)] \\ = -2 \left(\gamma - \frac{1}{2} \right) U([d_n]) - \Delta t^2 \left(\gamma - \frac{1}{2} \right) (2\beta - \gamma) T([a_n]) \end{aligned} \quad (9.1.103)$$

It follows from (9.1.103) that if $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$,

$$[E(d_n, v_n)] = [T(v_n)] + [U(d_n)] = 0 \quad (9.1.104)$$

That is, *total energy is conserved*.

Remarks

1. From (9.1.103) it follows that if $\gamma = \frac{1}{2}$, $E(d_n, v_n) + \Delta t^2(\beta - \gamma/2)T(a_n)$ is conserved.
2. If $\gamma > \frac{1}{2}$ and $\beta = \gamma/2$, then, from (9.1.103), total energy is nonincreasing. That is $[E(d_n, v_n)] \leq 0$.
3. If $\gamma > \frac{1}{2}$ and $\beta > \gamma/2$, then $E(d_n, v_n) + \Delta t^2(\beta - \gamma/2)T(a_n)$ is nonincreasing.

Exercise 10. (T. K. Caughey). Consider application of the trapezoidal algorithm to $\ddot{d} + (\omega^h)^2 d = 0$. Define $x_n = \langle \langle d_{n-1} \rangle \rangle = (d_{n+1} + 2d_n + d_{n-1})/4$. Show that

$$x_n = \frac{d_n}{1 + (\omega^h \Delta t)^2 / 4}$$

It may be concluded that x_n is a second-order accurate approximation to d_n and acts as a high-frequency filter because $x_n \rightarrow 0$ as $\omega^h \Delta t \rightarrow \infty$.

9.2 SUMMARY OF TIME-STEP ESTIMATES FOR SOME SIMPLE FINITE ELEMENTS

As we remarked previously, sufficient conditions for stability in finite element analysis may be obtained from estimates of the maximum eigenvalues of individual elements. We give some examples.

Example 1 (Two-node linear rod element)

If we assume a *lumped* (i.e., diagonal) mass matrix, then

$$\omega_{\max}^h = \frac{2c}{h} \quad (9.2.1)$$

where h is the element length and $c = \sqrt{E/\rho}$ is the so-called bar-wave velocity, in which E is Young's modulus and ρ is density. The critical time step for the Newmark method with $\beta = 0$, $\gamma = \frac{1}{2}$ (central-difference method) is

$$\Delta t \leq \frac{2}{\omega_{\max}^h} = \frac{h}{c} \quad (9.2.2)$$

which is the time required for a bar wave to traverse one element.

If we assume a *consistent* mass matrix, then

$$\omega_{\max}^h = \frac{2\sqrt{3}c}{h} \quad (9.2.3)$$

resulting in a reduced critical time step, viz.,

$$\Delta t \leq \frac{h}{\sqrt{3}c} \quad (9.2.4)$$

This result is typical: *Consistent-mass matrices tend to yield smaller critical time steps than lumped-mass matrices.*

Example 2 (Three-node quadratic rod element)

For this case, if we assume a lumped mass matrix based on a Simpson's rule weighting [8] (i.e., the ratio of the middle node mass to the end node masses is 4), we get

$$\omega_{\max}^h = \frac{2\sqrt{6}c}{h} \quad (9.2.5)$$

$$\Delta t \leq \frac{h}{\sqrt{6}c} \quad (9.2.6)$$

Comparison of (9.2.6) with (9.2.2) reveals that the allowable time step is about 0.4082 that for linear elements with lumped mass. This is based upon equal element lengths. Perhaps a more equitable comparison is one based upon equal nodal spacing. In this case the ratio doubles to 0.8164, but still the advantage is with linear elements.

Remark

Results for one-dimensional heat conduction may be deduced from the preceding cases by employing

$$\Delta t \leq \frac{2}{\lambda_{\max}^h} \quad (9.2.7)$$

where $\lambda_{\max}^h = (\omega_{\max}^h)^2$ and E/ρ is replaced by $k = \kappa/(\rho c)$, the diffusivity coefficient. For example, corresponding to (9.2.2) we have the classical result

$$\Delta t \leq \frac{h^2}{2k} \quad (9.2.8)$$

Example 3 (The linear beam element)

This element has been described in [9], [10], and Sec. 5.5. Transverse displacement, w , and face rotation, θ , are assumed to vary linearly over the element. One-point Gauss quadrature exactly integrates the bending stiffness and appropriately underintegrates the shear stiffness to avoid “locking” [9]. We assume the trapezoidal rule is used to develop the lumped mass matrix [10]. The matrices for a typical element describing bending in the plane are:

$$k_b = \frac{EI}{h} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \quad (\text{bending stiffness}) \quad (9.2.9)$$

$$k_s = \frac{\mu \hat{A}^s}{h} \begin{bmatrix} 1 & \frac{h}{2} & -1 & \frac{h}{2} \\ \frac{h}{2} & \frac{h^2}{4} & -\frac{h}{2} & \frac{h^2}{4} \\ -1 & \frac{-h}{2} & 1 & \frac{-h}{2} \\ \frac{h}{2} & \frac{h^2}{4} & \frac{-h}{2} & \frac{h^2}{4} \end{bmatrix} \quad (\text{shear stiffness}) \quad (9.2.10)$$

$$m = \rho \hat{A} \frac{h}{2} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{I}{\hat{A}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{I}{\hat{A}} \end{bmatrix} \quad (\text{mass}) \quad (9.2.11)$$

where here \hat{A} is the cross-section area, \hat{A}^s is the shear area, I is the moment of inertia, and μ is the shear modulus. The degree-of-freedom ordering is $w_1, \theta_1, w_2, \theta_2$, where the subscript is the node number.

Solution of the eigenvalue problem results in

$$\omega_{\max}^h = \max \left\{ \frac{2c}{h}, \left(\frac{2c_s}{h} \right) \left[1 + \frac{\hat{A}}{I} \left(\frac{h}{2} \right)^2 \right]^{1/2} \right\} \quad (9.2.12)$$

where c is the bar-wave velocity and $c_s^2 = \mu \hat{A}^s / (\rho \hat{A})$, the beam shear-wave velocity. Thus the critical time step for the central difference operator is given by

$$\Delta t \leq \min \left\{ \frac{h}{c}, \left(\frac{h}{c_s} \right) \left[1 + \frac{\hat{A}}{I} \left(\frac{h}{2} \right)^2 \right]^{-1/2} \right\} \quad (9.2.13)$$

To get a feeling for these quantities, we shall take a typical situation. Assume the cross section is rectangular with depth t and width 1. This results in $\hat{A} = t$ and $I = t^3/12$. We assume the ratio of wave speeds $c/c_s = \sqrt{3}$. This corresponds to a Poisson's ratio of $\frac{1}{4}$ and shear correction factor $\kappa = \hat{A}/\hat{A} = \frac{2}{3}$, so it is a reasonable approximation for most metals. The time step incurred by the bending mode will be critical when

$$\frac{h}{t} \leq \sqrt{\frac{2}{3}} \approx 0.8165 \quad (9.2.14)$$

This would be the case only for a very deep beam or an extremely fine mesh and is thus unlikely in practice. The more typical situation in structural analysis is when $t \ll h$ (i.e., very thin beams or coarse meshing). In this case the critical time step is slightly less than t/c , the time for a bar wave to traverse the thickness. As this is an extremely small time step, the cost of explicit integration becomes prohibitive.

A more favorable condition can be derived by adopting different values for the rotational lumped-mass coefficients. Specifically, take

$$m = \rho \hat{A} \frac{h}{2} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \alpha \end{bmatrix} \quad (9.2.15)$$

and select α to achieve a more desirable critical time step, without upsetting convergence. Beam mass matrices of the above type were introduced by Key and Beisinger [11]. This time, solution of the eigenvalue problem yields

$$\omega_{\max}^h = \max \left\{ \left(\frac{2c}{h} \right) \left(\frac{I}{\alpha \hat{A}} \right)^{1/2}, \left(\frac{2c_s}{h} \right) \left[1 + \frac{1}{\alpha} \left(\frac{h}{2} \right)^2 \right]^{1/2} \right\} \quad (9.2.16)$$

and thus

$$\Delta t \leq \min \left\{ \left(\frac{h}{c} \right) \left(\frac{\alpha \hat{A}}{I} \right)^{1/2}, \left(\frac{h}{c_s} \right) \left[1 + \frac{1}{\alpha} \left(\frac{h}{2} \right)^2 \right]^{-1/2} \right\} \quad (9.2.17)$$

Taking the value $\alpha = h^2/8$ in (9.2.17) and again assuming $\hat{A}/I = 12/t^2$ and $c/c_s = \sqrt{3}$ results in

$$\Delta t \leq \min \left\{ \left(\frac{h}{c} \right) \left(\sqrt{\frac{3}{2}} \cdot \frac{h}{t} \right), \frac{h}{c} \right\} \quad (9.2.18)$$

As long as

$$\frac{h}{t} > \sqrt{\frac{2}{3}} \cong 0.8165 \quad (9.2.19)$$

bar-wave transit time or better is achieved. Condition (9.2.19) will almost certainly be the case in practice. Analogous procedures may be applied to plate and shell elements (see Sec. 9.5 and Hughes, Liu, and Levit [12]). The original paper of Key and Beisinger [11] may be consulted for a treatment of the cubic, Bernoulli-Euler beam element, and analogous shell element considerations.

Example 4 (Quadrilateral and hexahedral elements)

Flanagan and Belytschko [13] have performed a valuable analysis of the one-point quadrature (four-node) quadrilateral and (eight-node) hexahedron, applicable to arbitrary geometric configurations of the elements. They obtain the following estimate of maximum element frequency:

$$\omega_{\max}^h \leq c_d g^{1/2} \quad (9.2.20)$$

where $c_d^2 = (\lambda + 2\mu)/\rho$, dilatational wave velocity; λ and μ are the Lamé parameters; and g is a geometric parameter. For example, in the case of the quadrilateral, g is defined as follows:

$$g = \frac{4}{A^2} \sum_{i=1}^2 \sum_{a=1}^4 B_{ia} B_{ia} \quad (9.2.21)$$

$$[B_{ia}] = \frac{1}{2} \begin{bmatrix} (y_2 - y_4) & (y_3 - y_1) & (y_4 - y_2) & (y_1 - y_3) \\ (x_4 - x_2) & (x_1 - x_3) & (x_2 - x_4) & (x_3 - x_1) \end{bmatrix} \quad (9.2.22)$$

where x_a and y_a are the coordinates of node a and A is the area. (For the definition of g in the case of the hexahedron, see [13].) The estimate, (9.2.20), leads to a *sufficient* condition for stability. For the central difference method (9.2.20) results in

$$\Delta t \leq \frac{2}{c_d g^{1/2}} \quad (9.2.23)$$

As an example of the restriction imposed by (9.2.23), consider a rectangular element with side lengths h_1 and h_2 . In this case (9.2.23) becomes

$$\Delta t \leq \frac{1}{c_d (1/h_1^2 + 1/h_2^2)^{1/2}} \quad (9.2.24)$$

For higher-order elements, there appears that little of a precise nature has been done. Most results are of the form (9.2.23) where the geometric factor is approximated by trial and error. One would hope that, with the aid of automatic symbolic manipulators, improved time-step estimates will become available in the ensuing years for more complex elements, material properties, etc.

Exercises

1. Verify the results obtained in Example 1.
2. Verify the results obtained in Example 2.

Solution of Exercise 2

The stiffness matrix and Simpson's rule lumped mass are:

$$\mathbf{k} = \frac{E}{h} \begin{bmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{bmatrix}$$

$$\mathbf{m} = \frac{\rho h}{6} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The eigenvalues and eigenvectors are:

| Eigenvalue | Eigenvector | |
|--------------|---------------------------|---------------------|
| 0 | { 1 1 1 } ^T | Rigid translation |
| $12 c^2/h^2$ | { -1 0 1 } ^T | Uniform extension |
| $24 c^2/h^2$ | { 1 -1/2 1 } ^T | Quadratic extension |

Discussion

It is instructive to know how the eigenvalues and eigenvectors may be deduced without going through the setup and calculation of the full eigenvalue problem. These ideas prove extremely useful in more complicated two- and three-dimensional situations. For example, consider the linear beam element described by (9.2.9) through (9.2.11). The eigenvalues and eigenvectors are:

| Eigenvalue | Eigenvector | |
|--|---|-------------------|
| 0 | { 1 0 1 0 } ^T | Rigid translation |
| 0 | { -1 2/h 1 2/h } ^T | Rigid rotation |
| $\frac{4c^2}{h^2}$ | { 0 1 0 -1 } ^T | Bending |
| $\left(\frac{4c_s^2}{h^2}\right)\left(1 + \frac{\hat{A}h}{I}\left(\frac{h}{2}\right)^2\right)$ | { 1 $\frac{\hat{A}h}{2I}$ -1 $\frac{\hat{A}h}{2I}$ } ^T | Shear |

They may be obtained as follows: First, note that from physical considerations two of the eigenvectors must correspond to rigid motions. The rigid translation mode is easily guessed, and it can be verified that it is indeed an eigenvector. The rigid

rotation clearly must have the form $\{-1 \ a \ 1 \ a\}^T$, where a is a constant determined by the requirement that the shear force engendered by this mode (i.e., the product of the shear stiffness and this vector) must be zero. This condition entails $a = 2/h$. The bending mode was guessed and it was easily verified to be an eigenvector, viz.,

$$(k_b + k_s - \lambda^h m) \begin{Bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{Bmatrix} = \left(\frac{2EI}{h} - \frac{\rho I h}{2} \lambda^h \right) \begin{Bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \Leftrightarrow \lambda^h = \frac{4c^2}{h^2}$$

The three eigenvectors computed so far are orthogonal with respect to m . Since the fourth eigenvector must be orthogonal to these three (with respect to m), it can be computed by assuming it has the form $\{a_1 \ a_2 \ a_3 \ a_4\}^T$, where a_1, \dots, a_4 are to be identified by the three orthogonality conditions

$$a_1 + a_3 = 0$$

$$-\hat{A}a_1 + \frac{2}{h}Ia_2 + \hat{A}a_3 + \frac{2}{h}Ia_4 = 0$$

$$a_2 - a_4 = 0$$

Taking $a_1 = 1$, these equations yield the mode $\begin{Bmatrix} 1 & \frac{\hat{A}h}{2I} & -1 & \frac{\hat{A}h}{2I} \end{Bmatrix}^T$. The product of this vector with the bending stiffness is zero, and therefore this is a pure shear deformation. The verification that this mode is an eigenvector, and the computation of the eigenvalue is:

$$(k_b + k_s - \lambda^h m) \begin{Bmatrix} 1 \\ \frac{\hat{A}h}{2I} \\ -1 \\ \frac{\hat{A}h}{2I} \end{Bmatrix} = \left(\hat{A}^s G \frac{2}{h} \left[1 + \frac{\hat{A}}{I} \left(\frac{h}{2} \right)^2 \right] - \rho \hat{A} \frac{h}{2} \lambda^h \right) \begin{Bmatrix} 1 \\ \frac{h}{2} \\ -1 \\ \frac{h}{2} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

$$\Leftrightarrow \lambda^h = \left(\frac{4c_s^2}{h^2} \right) \left(1 + \frac{\hat{A}}{I} \left(\frac{h}{2} \right)^2 \right)$$

Exercise 3. Generalize the argument of the preceding discussion to pertain to the lumped-mass matrix given by (9.2.15). Show that the rigid modes are the same, whereas the eigenvalues and eigenvectors of the other modes become:

| Eigenvalue | Eigenvector | |
|--|---|---------|
| $\frac{4c^2}{h^2} \left(\frac{I}{\alpha \hat{A}} \right)$ | $\{0 \quad 1 \quad 0 \quad -1\}^T$ | Bending |
| $\left(\frac{4c_s^2}{h^2} \right) \left[1 + \frac{1}{\alpha} \left(\frac{h}{2} \right)^2 \right]$ | $\left\{ 1 \quad \alpha^{-1} \frac{h}{2} \quad -1 \quad \alpha^{-1} \frac{h}{2} \right\}^T$ | Shear |

Exercise 4. (Key and Beisinger [11]). Consider a thin beam element based upon the Bernoulli-Euler theory in which a Hermite cubic is assumed for the displacement function. The element matrices are given by

$$\mathbf{k} = \frac{2EI}{h^3} \begin{bmatrix} 6 & 3h & -6 & 3h \\ & 2h^2 & -3h & h^2 \\ & & 6 & -3h \\ \text{symmetric} & & & 2h^2 \end{bmatrix}$$

$$\mathbf{m}_1 = \frac{\rho\hat{A}h}{420} \begin{bmatrix} 156 & 22h & 54 & -13h \\ & 4h^2 & 13h & -3h^2 \\ & & 156 & -22h \\ \text{symmetric} & & & 4h^2 \end{bmatrix} \quad (\text{consistent mass})$$

$$\mathbf{m}_2 = \frac{\rho\hat{A}h}{2} \begin{bmatrix} 1 & 0 & 0 & 0 \\ & ah^2 & 0 & 0 \\ & & 1 & 0 \\ \text{symmetric} & & ah^2 & \end{bmatrix} \quad (\text{lumped mass})$$

Determine critical time-step expressions for the consistent and lumped-mass cases. Provide a discussion of the consistent mass time step comparing it with that obtained for lumped mass.

Hint: The answer for the lumped case is

$$\Delta t_{crit} = \frac{h^2}{c} \sqrt{\frac{a\hat{A}}{(12a + 3)I}}$$

Discussion of Exercise 4

Let us again assume a rectangular cross section, as in Example 3. If we also assume $a = \frac{1}{12}$, then

$$\Delta t_{crit} = \left[\frac{h}{2t} \right] \frac{h}{c}$$

Since $\Delta t_{crit} = O(h^2)$, the critical time step will be extremely small for fine meshes. Comparison with the critical time step engendered by the axial beam mode (see Example 1) reveals that bending will be critical if

$$\frac{h}{t} < 2$$

This condition is in force only in extremely fine meshes or when the beam is “deep.”

(And for this case, the present theory is wholly inappropriate!) As long as $h/t \geq 2$, bar-wave transit time or better is achieved. This is the practically important case.

Exercise 5. In a transient frame analysis program, the critical time step must be computed accounting for the following deformation modes:

- Axial mode
- Torsional mode
- Bending modes
- Shear modes (if present)

The smallest time step resulting from the above modes determines the stable step for the element.

Assuming linear elements and lumped mass, determine the critical time step for the torsional mode. (For background information, see Secs. 5.4 and 5.5.) Compare this result with that for the axial mode.

Answer: $\Delta t_{\text{crit}} = h/c_t$, $c_t = \sqrt{\mu/\rho}$. By virtue of $\mu = E/[2(1 + \nu)]$, $\mu < E$; therefore $(\Delta t_{\text{crit}})_{\text{torsional}} > (\Delta t_{\text{crit}})_{\text{axial}}$.

Remark

From the preceding examples, exercises and discussions, it may be concluded that use of special lumped-mass matrices for beam and frame elements will produce critical time steps no worse than that governed by the axial mode (i.e., bar-wave transit time).

Exercise 6. A finite element computer program is being written to solve the wave equation in two-space dimensions. Bilinear quadrilaterals are to be employed. The element stiffness matrices are to be evaluated with one-point Gaussian quadrature, viz.,

$$\mathbf{k}^e = [k_{ab}^e], \quad 1 \leq a, b \leq 4$$

$$k_{ab}^e = 4c^2(jN_{a,i} N_{b,i})|_{(0,0)}$$

(c is a wave speed parameter.) Element lumped-mass matrices are to be evaluated by the product trapezoidal rule, viz.,

$$\mathbf{m}^e = [m_{ab}^e]$$

$$m_{ab}^e = j(\xi_a, \eta_a) \delta_{ab} \quad (\text{no sum})$$

The Newmark method with $\beta = 0$, $\gamma = \frac{1}{2}$ is to be employed for the temporal discretization (i.e., "central differences").

- i. Perform an analysis to determine the critical time step, making use of the following simplifications: Assume that the elements are rectangular and aligned parallel to the global Cartesian coordinate system. Denote by h_1' and h_2' the lengths of the sides in the x_1 and x_2 directions, respectively. Under the present circumstances, the isoparametric paraphernalia simplifies as follows:

$$j(\xi, \eta) = \frac{h_1^\epsilon h_2^\epsilon}{4}$$

$$N_{a,1}(0, 0) = \frac{\xi_a}{2h_1^\epsilon}$$

$$N_{a,2}(0, 0) = \frac{\eta_a}{2h_2^\epsilon}$$

ii. Physically interpret the result you have obtained.

Solution

$$\begin{aligned} i. \quad k_{ab}^\epsilon &= 4c^2(jN_{a,i}N_{b,i})|_{(0,0)} \\ &= 4c^2\left(\frac{h_1^\epsilon h_2^\epsilon}{4}\right)\left(\frac{\xi_a}{2h_1^\epsilon} \frac{\xi_b}{2h_1^\epsilon} + \frac{\eta_a}{2h_2^\epsilon} \frac{\eta_b}{2h_2^\epsilon}\right) \end{aligned}$$

$$\begin{aligned} m_{ab}^\epsilon &= j(\xi_a, \eta_a)\delta_{ab} \quad (\text{no sum}) \\ &= \left(\frac{h_1^\epsilon h_2^\epsilon}{4}\right)\delta_{ab} \end{aligned}$$

Given an eigenvector $\underbrace{\psi}_{4 \times 1} = \{\psi_a\}$, the eigenvalue may be determined from the Rayleigh quotient:

$$\mathcal{R}(\psi) = \frac{\sum_{a,b} k_{ab}^\epsilon \psi_a \psi_b}{\sum_{a,b} m_{ab}^\epsilon \psi_a \psi_b} = \frac{c^2 \sum_{a,b} (\xi_a \xi_b + \eta_a \eta_b) \psi_a \psi_b}{\sum_a \psi_a \psi_a}$$

It is a simple matter to deduce the eigenvectors. There must be a rigid translation and an hourglass mode (one-point quadrature!). These are (respectively)

$$\{1\} = \begin{Bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix}, \quad \{\xi_a \eta_a\} = \begin{Bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{Bmatrix}$$

The eigenvalue corresponding to each is zero. The remaining eigenvectors are orthogonal to these and therefore must have the form

$$\begin{Bmatrix} \alpha \\ \beta \\ -\alpha \\ -\beta \end{Bmatrix}, \quad \text{where } \alpha \text{ and } \beta \text{ are constants}$$

By setting up the eigenproblem, it can be verified that the eigenvectors are:

$$\{\eta_a\} = \begin{Bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{Bmatrix} \quad \{\xi_a\} = \begin{Bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{Bmatrix}$$

Substituting into the Rayleigh quotient yields the eigenvalues (respectively)

$$\lambda^h = \left(\frac{2c}{h_2}\right)^2, \quad \left(\frac{2c}{h_1}\right)^2$$

The critical time step for $\beta = 0$, $\gamma = \frac{1}{2}$ is $2/\omega_{\max}^h$. Therefore

$$\Delta t_{\text{crit}} = \frac{h}{c}, \quad \text{where } h = \min\{h_1, h_2\}$$

- ii. The time step equals the transit time for a wave traveling at velocity c to traverse the shortest distance across the element.

Remark

Further time-step estimates may be found in Belytschko [14].

Exercise 7. Consider the one-dimensional second-order wave equation

$$u_{ttt} = c^2 u_{xx}$$

Perform a von Neumann stability analysis of the central-difference finite-difference method:

$$u_{n+1}^h(m) - 2u_n^h(m) + u_{n-1}^h(m) = r^2(u_n^h(m+1) - 2u_n^h(m) + u_n^h(m-1))$$

where $r = c\Delta t/h$. Obtain an expression for the local truncation error. (See Sec. 8.3 for notational definitions and a description of the von Neumann method.) The central-difference finite-difference method is identical to the use of $\beta = 0$, $\gamma = \frac{1}{2}$ in Newmark's method in conjunction with piecewise linear finite elements and lumped mass.

9.3 LINEAR MULTISTEP (LMS) METHODS

9.3.1 LMS Methods for First-order Equations

Consider a system of first-order, linear, ordinary differential equations

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t) = \mathbf{G}\mathbf{y} + \mathbf{H}(t) \quad (9.3.1)$$

where \mathbf{G} is a given constant matrix and \mathbf{H} is a given vector-valued function of t . A k -step linear multistep method for (9.3.1) is defined by the following expression:

$$\sum_{i=0}^k \{\alpha_i \mathbf{y}_{n+1-i} + \Delta t \beta_i \mathbf{f}(\mathbf{y}_{n+1-i}, t_{n+1-i})\} = 0 \quad (9.3.2)$$

The α_i 's and β_i 's are parameters which define the method. Note that the word "linear" in linear multistep method has nothing to do with the linearity of (9.3.1). Indeed, (9.3.2) is perfectly well defined for a nonlinear function $f(y, t)$. Rather, the linearity pertains to the form of (9.3.2). An excellent reference for LMS methods of this type is Gear [15]. An LMS method is called *explicit* if $\beta_0 = 0$; otherwise it is called *implicit*. It is called a *backward-difference method* if $\beta_i = 0$ for all $i \geq 1$. An example of a one-step LMS method of first-order type is the generalized trapezoidal family discussed in Chapter 8 for which $\alpha_0 = -\alpha_1 = 1$, $\beta_0 = -\alpha$, and $\beta_1 = \alpha - 1$. Recall that $\alpha = 0$ defines an explicit method, and $\alpha = 1$ defines a backward difference method. The greater the number of steps involved in the definition of an LMS method, the greater the "historical data pool" that must be stored in computing—a practical disadvantage.

Exercise 1. Put the semidiscrete heat equation into the first-order form defined by (9.3.1).

Conclude that the eigenvalues of G are nonpositive real numbers. *Thus the spectrum of G resides upon the negative real axis in the complex plane \mathbb{C} (i.e., $\lambda(G) \leq 0$).*

Exercise 2. Put the semidiscrete equation of motion in first-order form. Take $y = \{d, \dot{d}\}^T$.

Conclude that the $2n_{eq}$ eigenvalues of G take on the following forms:

i. $0 \leq \xi < 1$, *underdamped*:

$$\lambda_{1,2}(G) = -\xi\omega^h \pm i\omega^h\sqrt{1 - \xi^2}$$

ii. $\xi = 1$, *critically damped*:

$$\lambda_{1,2}(G) = -\omega^h \quad (\text{double root})$$

iii. $\xi > 1$, *overdamped*:

$$\lambda_{1,2}(G) = -\xi\omega^h \pm \omega^h\sqrt{\xi^2 - 1}$$

When $\xi = 0$, the eigenvalues are complex conjugate, $\pm i\omega^h$, and thus reside on the imaginary axis. *In all cases, the eigenvalues of G are confined to the negative half-plane of \mathbb{C} including the negative real axis (i.e., $\operatorname{Re}(\lambda(G)) \leq 0$).*

Exercise 3. Make a sketch of the complex plane and identify where the eigenvalues, $\lambda(G)$, as determined in Exercises 1 and 2, fall.

Many algorithms of practical interest take the form (9.3.2). The analysis of LMS methods may be performed using modal techniques, as described previously. Stability is generally phrased in terms of the roots of the polynomial associated to (9.3.2), namely,

$$\sum_{i=0}^k (\alpha_i + \Delta t \lambda \beta_i) \zeta^{n+1-i} = 0$$

(9.3.3)

where λ is an eigenvalue of G . (To derive (9.3.3), assume G admits the representation $G = P\Lambda P^{-1}$, where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_N)$ and N is the dimension of G . Show that (9.3.2) can be reduced to an uncoupled system of scalar equations having the form $\sum_{i=0}^k (\alpha_i + \Delta t \lambda \beta_i) z_{n+1-i} = 0$, which has solutions $z_n \sim c \zeta^n$.) An LMS method of the form (9.3.2) is said to be *absolutely stable*, at a fixed $\lambda \Delta t$, if all ζ satisfying (9.3.3) are such that $|\zeta| \leq 1$. The *region of absolute stability* of an LMS method is the set of $\lambda \Delta t \in \mathbb{C}$ at which it is absolutely stable. An LMS method is *A-stable* if solutions of the SDOF analog of (9.3.2) $\rightarrow 0$ as $n \rightarrow \infty$ when $\text{Re}(\lambda) < 0$. Physically speaking, an A-stable algorithm is one which produces solutions which decay to zero whenever the corresponding exact solutions decay to zero. Clearly, if an algorithm is A-stable then the region of absolute stability contains the left half-plane of \mathbb{C} . The condition of A-stability places no limitation on the size of Δt , consequently it is closely related to what we termed “unconditional stability” previously. The only difference is that spectral stability allows eigenvalues of unit modulus which potentially could conserve a solution rather than contract it to zero. It should be apparent that A-stable algorithms are important for many physical problem classes. However, within the class of LMS methods, the subclass of A-stable methods is severely limited. This is the content of a celebrated theorem:

Dahlquist's Theorem [16]

1. An explicit A-stable LMS method does not exist.
2. A third-order accurate A-stable LMS method does not exist.
3. The second-order accurate A-stable LMS method with the smallest error constant³ is the trapezoidal rule.

The upshot of this theorem is that if we seek an A-stable LMS method that possesses some special feature (such as high-frequency numerical dissipation), we necessarily entail some loss of accuracy with respect to the trapezoidal rule. Thus it would appear that the trapezoidal rule is the canonical A-stable LMS method for structural dynamics. The widespread use and popularity of the trapezoidal rule in this context appears to confirm this observation. However, there are other useful A-stable LMS methods for structural dynamics; for example, Park's method [17]. Many structural dynamics algorithms do not fall within the class of first-order LMS methods considered so far. The Newmark family of methods may be mentioned in this regard. Another class of LMS methods for second-order equations will be described subsequently; it contains the Newmark family, among others.

Since A-stable LMS methods are at most second-order accurate, the question arises, “Is there some weakened notion of A-stability which pertains to a physically relevant class of problems and allows for the development of higher-order-accurate methods?” An affirmative answer is provided by the concept of stiff stability proposed

³The error constant is the coefficient of Δt^k in the local truncation error.

by Gear [15]. A method is said to be *stiffly stable* if it is absolutely stable in the region of the $\lambda\Delta t$ -plane defined by $\operatorname{Re}(\lambda\Delta t) < -\delta$, where δ is a positive constant. Gear has developed a family of k -step, k th-order-accurate LMS methods which are stiffly stable for $k = 2, 3, \dots, 6$. Each method is a backward-difference LMS method. The second-order method is, in addition, A -stable; however, it is significantly less accurate than the trapezoidal rule. For $k \geq 3$, the regions of instability include portions of the imaginary axis. Consequently, these stiffly stable algorithms also appear inappropriate for typical structural dynamics applications. Because the region of stability of each of the methods, $k = 2, 3, \dots, 6$, includes the negative real axis, one would anticipate good behavior in application to problems of heat conduction.

Park [17] has developed a second-order accurate, A -stable algorithm, which retains good accuracy in the low frequencies and strong dissipative characteristics in the high frequencies, by combining Gear's two-step and three-step stiffly stable algorithms. The resultant scheme is defined by

Park's Method

$$k = 3, \quad \alpha_0 = -1, \quad \alpha_1 = \frac{15}{10}, \quad \alpha_2 = \frac{-6}{10}, \quad \alpha_3 = \frac{1}{10}, \quad \beta_0 = \frac{6}{10}, \\ \beta_1 = \beta_2 = \beta_3 = 0$$

(9.3.4)

9.3.2 LMS Methods for Second-order Equations

Consider a system of second-order, linear, ordinary differential equations written in the form:

$$\ddot{\mathbf{y}} = f(\mathbf{y}, \dot{\mathbf{y}}, t) = G_0 \mathbf{y} + G_1 \dot{\mathbf{y}} + H(t) \quad (9.3.5)$$

where G_0 and G_1 are given constant matrices. A k -step LMS method for (9.3.5) is given by (Gerardin [18]):

$$\sum_{i=0}^k \{\alpha_i \mathbf{y}_{n+1-i} + \Delta t \beta_i G_1 \mathbf{y}_{n+1-i} + \Delta t^2 \gamma_i [G_0 \mathbf{y}_{n+1-i} + H(t_{n+1-i})]\} = \mathbf{0}$$

(9.3.6)

The method is defined by the values of α_i , β_i and γ_i , $i = 0, 1, \dots, k$. An LMS method of this type is called *explicit* if β_0 and $\gamma_0 = 0$. (If $G_1 = \mathbf{0}$, then clearly only γ_0 need be zero.) The method is a *backward-difference method* if β_i and $\gamma_i = 0$,

$i \geq 1$. (Likewise, if $G_1 = 0$, only the γ_i 's, $i \geq 1$, need be zero.) The equation of motion may be put in the form of (9.3.5) simply by multiplying through by M^{-1} . Observe that with $y = d$, (9.3.6) is a *displacement difference-equation form of the algorithm*. The commonly used algorithms of structural dynamics can all be put into this form, although very few are naturally cast this way.

In the sense of an LMS method for second-order systems, Newmark's method is a two-step method. LMS methods applied to the first-order form of the equation of motion may likewise be recast as second-order-type LMS methods (e.g., Park's method).

The stability properties of second-order-type LMS methods may be investigated in similar fashion to first-order LMS methods. Briefly, modal reduction may be employed which, for the equation of motion, leads to the following polynomial:

$$\sum_{i=0}^k (\alpha_i + 2\xi\omega^h\Delta t \beta_i + (\omega^h\Delta t)^2 \gamma_i) \zeta^{n+1-i} = 0 \quad (9.3.7)$$

An analog of Dahlquist's theorem also holds for LMS methods for second-order equations (see Krieg [19]).

Exercise 4. Derive the following *displacement difference-equation form of Newmark's algorithm*:

$$(M + \gamma\Delta t C + \beta\Delta t^2 K)d_{n+1} + \left[-2M + (1 - 2\gamma)\Delta t C + \left(\frac{1}{2} - 2\beta + \gamma\right)\Delta t^2 K \right]d_n + \left[M - (1 - \gamma)\Delta t C + \left(\frac{1}{2} + \beta - \gamma\right)\Delta t^2 K \right]d_{n-1} + \Delta t^2 \bar{F}_n = 0 \quad (9.3.8)$$

Define \bar{F}_n in terms of F_{n-1} , F_n , and F_{n+1} .

Exercise 5. Modally reduce the homogeneous version of (9.3.8) to the form (9.3.7). Define α_i and β_i , $i = 0, 1, 2$. Show that this results in an identical equation to the one derived from the 2×2 amplification matrix form, (9.1.44). Deduce that the roots of the stability polynomial (9.3.7) are identical to the eigenvalues of the amplification matrix (i.e., $\zeta = \lambda(A)$).

Exercise 6. Consider the scalar difference equation that emanates from (9.3.1), (9.3.2) after modal reduction:

$$\sum_{i=0}^k (\alpha_i + \Delta t \lambda \beta_i) z_{n+1-i} = 0 \quad (9.3.9)$$

Derive a one-step, multivalue method corresponding to (9.3.9), which has the form

$$Z_{n+1} = AZ_n \quad (9.3.10)$$

where $Z_n = \{z_n, z_{n-1}, \dots, z_{n+1-k}\}^T$. That is, explicitly define the $k \times k$ amplification matrix A . Establish that the eigenvalues of A are the same as the roots, ζ , of the stability polynomial, (9.3.3), emanating from (9.3.9).

Exercise 7. Following along the lines of the previous exercise, consider the stability polynomial:

$$\zeta^3 - 2A_1\zeta^2 + A_2\zeta - A_3 = 0 \quad (9.3.11)$$

Derive a 3×3 amplification matrix, as in (9.3.10), which gives rise to (9.3.11). Show that

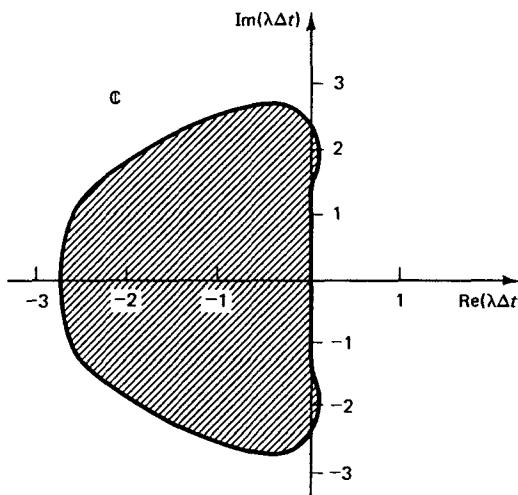
$$A_1 = \frac{1}{2} \text{trace } A \quad (9.3.12)$$

$$A_2 = \text{sum of the principal minors of } A \quad (9.3.13)$$

$$A_3 = \det A \quad (9.3.14)$$

(Equations (9.3.12) through (9.3.14) define the *principal invariants of A*. They arise naturally in calculating the stability polynomial by way of $\det(A - \zeta I) = 0$; cf. (9.3.11).)

Exercise 8. The region of absolute stability of a fourth-order explicit Runge-Kutta LMS method for the model equation $\dot{y} = \lambda y$, where λ is a complex number, is depicted in the accompanying figure. (Assume the eigenvalues of the amplification matrix are distinct.)



Determine the stability conditions for the following cases:

- i. Parabolic case. [Ans.: $-2.7 \leq \lambda^h \Delta t \leq 0$]
- ii. Undamped hyperbolic case. [Ans.: $\omega^h \Delta t \leq 2.5$]

Exercise 9. Consider the trapezoidal-rule algorithm. Show that it is *A*-stable. Determine its region of absolute stability. [Ans.: The left-hand complex plane including the imaginary axis; i.e., it is stable for all $\lambda^h \Delta t$ such that $\text{Re}(\lambda^h \Delta t) \leq 0$.]

Exercise 10. Show that the Euler backward-difference method is *A*-stable. Determine its region of absolute stability.

Exercise 11. Determine the region of absolute stability of the Euler forward-difference algorithm

Exercise 12. Determine local truncation error expressions for LMS methods of type (9.3.2) and (9.3.6).

Answer:

$$\Delta t \tau = \sum_{i=0}^k \{ \alpha_i y(t_{n+1-i}) + \Delta t \beta_i f(y(t_{n+1-i}), t_{n+1-i}) \}$$

$$\Delta t^2 \tau = \sum_{i=0}^k \{ \alpha_i y(t_{n+1-i}) + \Delta t \beta_i G_1 y(t_{n+1-i}) + \Delta t^2 \gamma_i [G_0 y(t_{n+1-i}) + H(t_{n+1-i})] \}$$

9.3.3 Survey of Some Commonly Used Algorithms in Structural Dynamics

This section surveys and compares some algorithms which have been proposed for structural dynamics applications. The algorithms considered are all LMS methods of first-order or second-order type, and the comparison is based upon both stability behavior and accuracy. Accuracy characteristics of LMS methods may be studied in similar fashion to the study of accuracy for Newmark's method. The only complication is that one needs to ignore so-called *spurious roots*, focusing attention on the *principal roots*. For a method possessing an amplification matrix of rank k , assuming linearly independent eigenvectors, the analog of (9.1.62) can be written as

$$d_n = \sum_{i=1}^k c_i \lambda_i^n = e^{-\bar{\xi} \bar{\omega}^h t_n} (\bar{c}_1 \cos \bar{\omega}_d^h t_n + \bar{c}_2 \sin \bar{\omega}_d^h t_n) + \sum_{i=3}^k c_i \lambda_i^n \quad (9.3.15)$$

The first two roots, λ_1 and λ_2 , are the principal roots, and the remaining $k - 2$ are the spurious roots. The principal roots define $\bar{\xi}$ and $\bar{\omega}^h$ through

$$\lambda_{1,2} = e^{(-\bar{\xi} \bar{\omega}^h \pm i \bar{\omega}_d^h) \Delta t} \quad (9.3.16)$$

The comparisons that follow are for the physically undamped case (i.e., $\xi = 0$).

Houbolt's Method

Houbolt's method [20] was one of the earliest employed in structural dynamics computations. It is defined by the following formulas:

$$Ma_{n+1} + Cv_{n+1} + Kd_{n+1} = F_{n+1} \quad (9.3.17)$$

$$a_{n+1} = \frac{2d_{n+1} - 5d_n + 4d_{n-1} - d_{n-2}}{\Delta t^2} \quad (9.3.18)$$

$$v_{n+1} = \frac{11d_{n+1} - 18d_n + 9d_{n-1} - 2d_{n-2}}{6\Delta t} \quad (9.3.19)$$

Combining (9.3.17) through (9.3.19) leads to a three-step LMS method of second-order type which is unconditionally stable and second-order accurate. Furthermore, if $C = \mathbf{0}$, it is a backward-difference, stiffly A -stable method and thus has many features in common with Park's method. The advantage of Park's method is that it is more accurate. However, it entails a significantly greater historical data pool. A disadvantage of both the Houbolt and Park methods is that they require special *starting procedures*.

Exercise 13. Apply Park's method to the first-order form of the equations of motion. Develop the displacement-difference equation form of the algorithm by eliminating v_{n+1-i} , $i = 0, 1, 2, 3$. What is k when the algorithm is viewed as a k -step LMS method of second-order type?

Exercise 14

- i. Derive an amplification matrix for Houbolt's method. That is, define A such that

$$\begin{Bmatrix} d_{n+1} \\ d_n \\ d_{n-1} \end{Bmatrix} = \underbrace{A}_{3 \times 3} \begin{Bmatrix} d_n \\ d_{n-1} \\ d_{n-2} \end{Bmatrix}$$

- ii. Write a computer program that calculates the spectral radius of A . Plot $\rho(A)$ versus $\Delta t/T$.
- iii. Put Houbolt's method into displacement difference equation form and derive an expression for local truncation error. *Hint:* See Exercise 12.
- iv. The *Routh-Hurwitz criterion* gives sufficient conditions for the roots of a stability polynomial, or associated amplification matrix, to be less than or equal to one in modulus. In the present case, the Routh-Hurwitz criterion takes the form:

$$1 - 2A_1 + A_2 - A_3 \geq 0$$

$$3 - 2A_1 - A_2 + 3A_3 \geq 0$$

$$3 + 2A_1 - A_2 - 3A_3 \geq 0$$

$$1 + 2A_1 + A_2 + A_3 \geq 0$$

$$1 - A_2 + A_3(2A_1 - A_3) \geq 0$$

Use these expressions to show that Houbolt's method is unconditionally stable.

Collocation Schemes

Collocation methods generalize and combine aspects of the Newmark method and Wilson- θ method [21]. A systematic analysis of collocation is contained in Hilber and Hughes [22]. The collocation schemes are defined by the following equations:

$$Ma_{n+\theta} + Cv_{n+\theta} + Kd_{n+\theta} = F_{n+\theta}$$

(9.3.20)

$$\mathbf{a}_{n+\theta} = (1 - \theta)\mathbf{a}_n + \theta\mathbf{a}_{n+1} \quad (9.3.21)$$

$$\mathbf{F}_{n+\theta} = (1 - \theta)\mathbf{F}_n + \theta\mathbf{F}_{n+1} \quad (9.3.22)$$

$$\begin{aligned} \mathbf{d}_{n+\theta} = \mathbf{d}_n + \theta\Delta t\mathbf{v}_n + \frac{(\theta\Delta t)^2}{2} &\{(1 - 2\beta)\mathbf{a}_n \\ &+ 2\beta\mathbf{a}_{n+\theta}\} \end{aligned} \quad (9.3.23)$$

$$\mathbf{v}_{n+\theta} = \mathbf{v}_n + \theta\Delta t\{(1 - \gamma)\mathbf{a}_n + \gamma\mathbf{a}_{n+\theta}\} \quad (9.3.24)$$

To (9.3.20) through (9.3.24) are appended the Newmark formulas (i.e., (9.1.5) and (9.1.6)) for purposes of defining \mathbf{d}_{n+1} and \mathbf{v}_{n+1} , respectively. θ is called the *collocation parameter*. The collocation schemes can be put into the form of a three-step LMS method of second-order type. Alternatively, the collocation schemes' model equation can be put in amplification matrix form; see [23]. In this case, the rank of the amplification matrix is 3.

If $\theta = 1$, the scheme reverts to Newmark's method. If $\beta = \frac{1}{6}$ and $\gamma = \frac{1}{2}$, the Wilson- θ methods are obtained. A necessary and sufficient condition for second-order accuracy is that $\gamma = \frac{1}{2}$. Unconditionally stable, second-order accurate schemes are defined by

$$\gamma = \frac{1}{2}, \quad \theta \geq 1, \quad \frac{\theta}{2(\theta + 1)} \geq \beta \geq \frac{2\theta^2 - 1}{4(2\theta^3 - 1)} \quad (9.3.25)$$

The best-behaved collocation schemes were determined in [22]. This amounts to a one-parameter subfamily of methods with $\gamma = \frac{1}{2}$, and $\theta = \theta^*(\beta)$ defined by Table 9.3.1. These methods are referred to as *optimal collocation methods* and are the only ones considered henceforth.

TABLE 9.3.1 Smallest collocation parameter, θ^* , which ensures complex conjugate principle roots as $\Delta t/T \rightarrow \infty$. Corresponding values of algorithmic damping ratio and relative period error for $\Delta t/T = 0.1$ [22]

| β | θ^* | $\bar{\xi}$ | $(\bar{T}-T)/T$ |
|---------------|------------|-----------------------|-----------------|
| $\frac{1}{4}$ | 1 | 0 | 0.032 |
| 0.24 | 1.021712 | 0.60×10^{-4} | 0.032 |
| 0.23 | 1.047364 | 0.27×10^{-3} | 0.033 |
| 0.22 | 1.077933 | 0.70×10^{-3} | 0.034 |
| 0.21 | 1.114764 | 0.14×10^{-2} | 0.036 |
| 0.20 | 1.159772 | 0.27×10^{-2} | 0.039 |
| 0.19 | 1.215798 | 0.46×10^{-2} | 0.043 |
| 0.18 | 1.287301 | 0.77×10^{-2} | 0.050 |
| 0.17 | 1.381914 | 0.13×10^{-1} | 0.060 |
| $\frac{1}{6}$ | 1.420815 | 0.15×10^{-1} | 0.064 |
| 0.16 | 1.514951 | 0.21×10^{-1} | 0.075 |

α -Method (Hilber-Hughes-Taylor Method)

Numerical damping cannot be introduced in the Newmark method without degrading the order of accuracy. To improve upon this situation, Hilber, Hughes, and Taylor [24] introduced the α -method. In the α -method, the finite difference formulas of the Newmark method are retained (i.e., (9.1.5) and (9.1.6)) whereas the time-discrete equation of motion is modified as follows:

$$\mathbf{M}\mathbf{a}_{n+1} + (1 + \alpha)\mathbf{C}\mathbf{v}_{n+1} - \alpha\mathbf{C}\mathbf{v}_n + (1 + \alpha)\mathbf{K}\mathbf{d}_{n+1} - \alpha\mathbf{K}\mathbf{d}_n = \mathbf{F}(t_{n+\alpha}) \quad (9.3.26)$$

where $t_{n+\alpha} = (1 + \alpha)t_{n+1} - \alpha t_n = t_{n+1} + \alpha\Delta t$. Clearly, if $\alpha = 0$ we reduce to the Newmark method. If the parameters are selected such that $\alpha \in [-\frac{1}{3}, 0]$, $\gamma = (1 - 2\alpha)/2$, and $\beta = (1 - \alpha)^2/4$, an unconditionally stable, second-order accurate scheme results [25]. At $\alpha = 0$, we have the trapezoidal rule. Decreasing α increases the amount of numerical dissipation. The α -method can be put in the form of a three-step LMS method for second-order equations, or equivalently, in a rank-3 amplification matrix form; see [23]. The α -method is incorporated in program DLEARN (see Chapter 11).

Comparison of Algorithms

The algorithms compared in Figs. 9.3.1 through 9.3.4 are

1. Houbolt's method
2. Park's method
3. Optimal collocation methods
4. α -methods

These methods have the following features in common: They are implicit, unconditionally stable, second-order accurate, and dissipative. All require approximately the same computational effort. However, Park's method requires a larger historical data pool than the others. The collocation and α -methods permit a parametric control of numerical dissipation, whereas the Houbolt and Park methods do not. (For comparison purposes we also include some results for the Newmark and Wilson- θ methods.)

In Fig. 9.3.1 the spectral radii of the various cases are presented. The spectral radii of the Houbolt and Park methods approach zero as $\Delta t/T \rightarrow \infty$, as is typical of backward-difference schemes. Collocation schemes and α -methods are seen also to possess strong damping in the high-frequency regime. Recall that the effect of $\rho < 1$ is accumulative, e.g., $\rho^n \leq e^{-n(1-\rho)}$, which rapidly approaches zero as n is increased.

In Fig. 9.3.2 algorithmic damping ratios are compared. The Houbolt and collocation methods are seen to affect the low modes (i.e., $\Delta t/T \leq 0.1$) too strongly. The inefficiency of the damping in the collocation scheme versus the α -method can

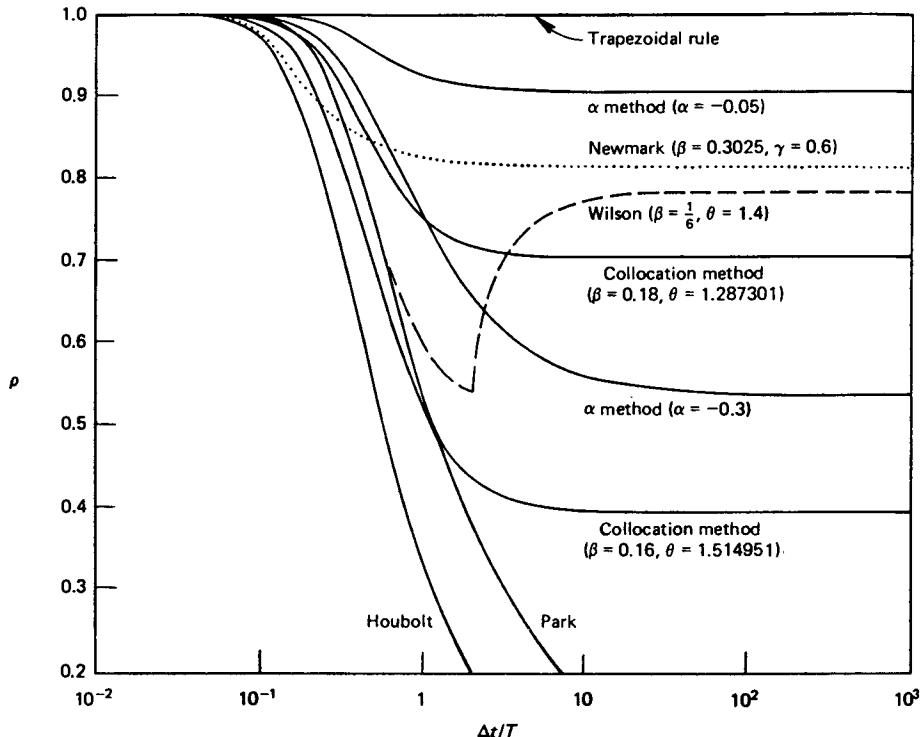


Figure 9.3.1 Spectral radii for α -methods, optimal collocation schemes, and Houbolt, Newmark, Park, and Wilson methods [22].

be seen by comparing the cases $\beta = 0.18$ and $\alpha = -0.3$, respectively. From Fig. 9.3.1, $\alpha = -0.3$ is seen to damp the high modes more strongly than $\beta = 0.18$. On the other hand, from Fig. 9.3.2 the low modes are affected less by $\alpha = -0.3$ than by $\beta = 0.18$.

Relative period errors are compared in Fig. 9.3.3. The collocation schemes and α -methods have smaller errors than the Houbolt and Park methods.

The following observations summarize the comparisons: All methods are capable of sufficiently damping out the high modes. The Houbolt method affects the low modes much too strongly, both from the point of view of damping ratio and of period error. Park's method possesses good low mode damping properties; however, its period error is higher than the collocation schemes and α -methods. The collocation schemes damp the low modes too strongly.

In Fig. 9.3.4, period error results are presented for undamped Newmark methods ($\gamma = \frac{1}{2}$), and comparison is made with results for the Houbolt and Wilson algorithms. Note that for $\beta < \frac{1}{4}$, the methods presented are conditionally stable, which is made evident by the abrupt increases in period error at finite $\Delta t/T$. Notice also that the central difference method ($\beta = 0$) tends to shorten periods, whereas the trapezoidal rule ($\beta = \frac{1}{4}$) increases periods.

As long as the method is stable, $\gamma = \frac{1}{2}$ implies $\bar{\xi} = 0$ for physically undamped

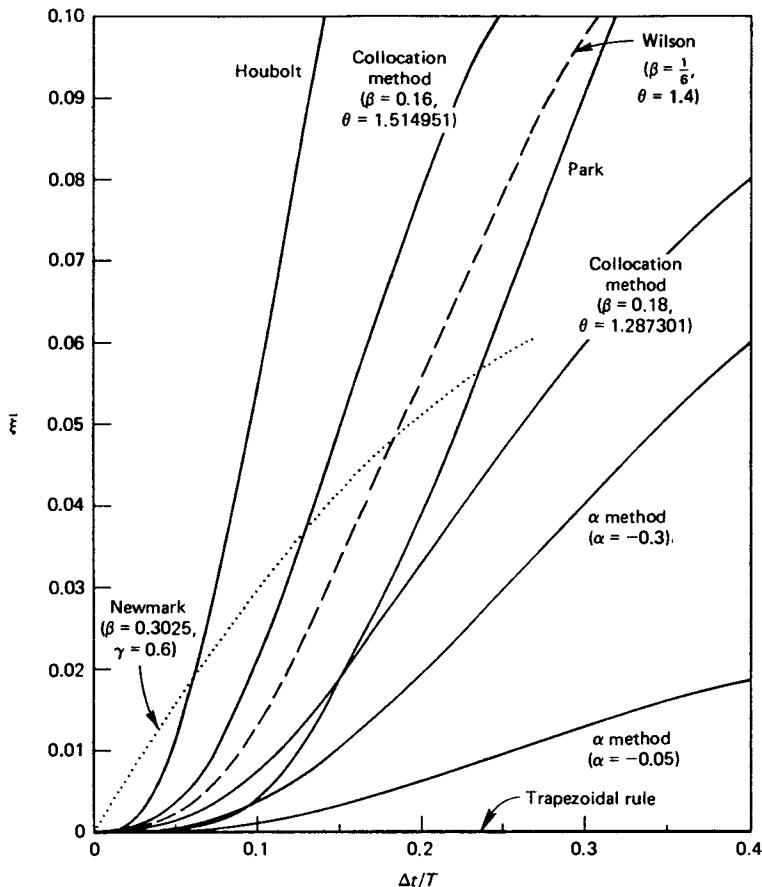


Figure 9.3.2 Algorithmic damping ratios for α -methods, optimal collocation schemes, and Houbolt, Newmark, Park, and Wilson methods [22].

Newmark methods. Recall that $\gamma > \frac{1}{2}$ produces numerical dissipation in Newmark's method but reduces accuracy to first-order. In the physically undamped case, (9.1.65) may be written as

$$\bar{\xi} = \pi \left(\gamma - \frac{1}{2} \right) \frac{\Delta t}{T} + O\left(\left(\frac{\Delta t}{T}\right)^2\right) \quad (9.3.27)$$

Thus the slope of the $\bar{\xi}$ versus $\Delta t/T$ curve is positive at $\Delta t/T = 0$, an indication of first-order accuracy. This may be contrasted with the zero slopes of the second-order methods compared in Fig. 9.3.2. To estimate (9.3.27), assume $\gamma = 0.6$ and $\Delta t/T = 0.1$, and neglect the quadratic term. This results in $\bar{\xi} \approx 0.01\pi$. Connecting the point $(\Delta t/T, \bar{\xi}) = (0.1, 0.01\pi)$ to the origin with a straight line gives a fairly accurate representation of the behavior of $\bar{\xi}$ for the $\gamma = 0.6$ Newmark method, $0 \leq \Delta t/T \leq 0.1$ (cf. Fig. 9.3.2). This excessive low-mode numerical dissipation is the main shortcoming of damped Newmark methods.

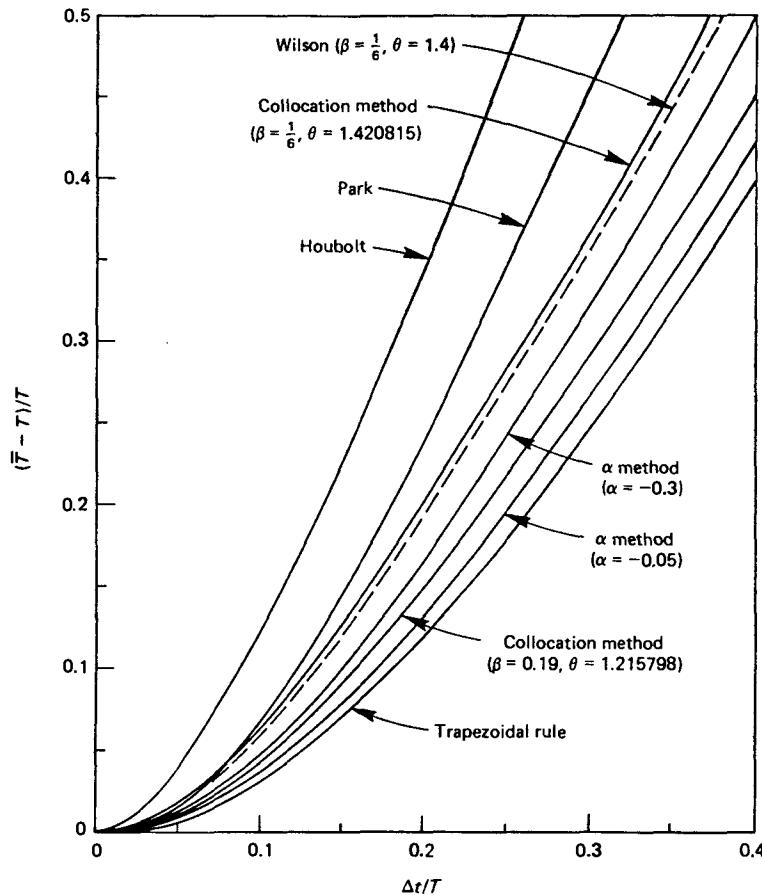


Figure 9.3.3 Relative period errors for α -methods, optimal collocation schemes, and Houbolt, Park, and Wilson methods [22].

Discussion Concerning Time-Stepping Algorithms in Structural Dynamics

Considerable effort has gone into the development of efficient computational methods for the step-by-step integration of the equations of structural dynamics. Although there is no universal consensus, it is generally agreed that for a method to be competitive it should possess the following attributes:

1. Unconditional stability when applied to linear problems.
2. No more than one set of implicit equations to be solved at each step.
3. Second-order accuracy.
4. Controllable algorithmic dissipation in the higher modes.
5. Self-starting.

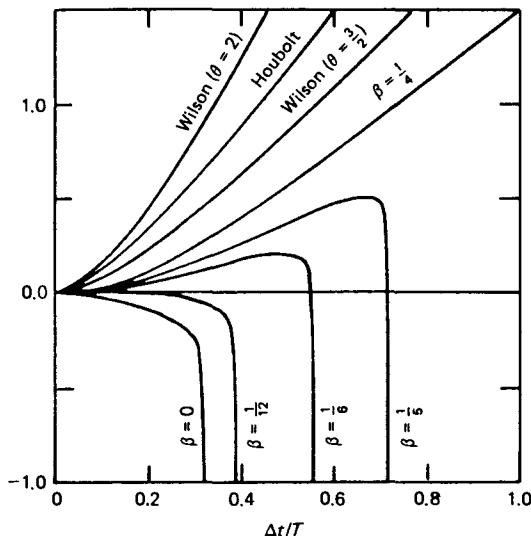


Figure 9.3.4 Period errors for undamped Newmark methods ($\gamma = \frac{1}{2}$) compared with Wilson and Houbolt schemes [26].

Conditionally stable algorithms require that a time step be taken that is less than a constant times the smallest period of the structure. In complicated structural models, containing slender members exhibiting bending effects, this restriction is a stringent one and often entails using time steps that are much smaller than those needed for accuracy, especially when only low-mode response is of interest. For these reasons unconditionally stable algorithms are generally preferred.

Dahlquist's theorem [16] (see also Krieg [19]) asserts that there is no unconditionally stable explicit method amongst the class of linear multistep methods. Thus attribute 1 necessitates the use of implicit methods, and this engenders considerable computational effort since coefficient matrices must be stored and factored. It is important for purposes of efficiency that this work be kept to a minimum. This can be done by insisting that the solution of no more than one implicit system, of the size of the mass and stiffness matrices, be required at each time step. More elaborate schemes have been proposed that require larger implicit systems, or two or more implicit systems of the size of the stiffness and mass to be solved at each step, and improved properties have been obtained (see for example Argyris, Dunne and Angelopoulos [29], Geradin [18], Hilber [23], Krieg and Key [30], and Kurdi [31]). However, these techniques require at least twice the storage and computational effort of the simpler methods and thus have not been widely adopted.

Experience has indicated that in structural dynamics, second-order accurate methods are vastly superior to first-order accurate methods. Another consequence of Dahlquist's theorem is that there is no third-order accurate unconditionally stable linear multistep method. Thus we must be content with second-order accuracy. Furthermore, the second-order method with the smallest error constant is the trapezoidal rule. Hence any effort to obtain some specific property within the context of second-order accurate unconditionally stable methods must result in some degradation of accuracy when compared with the trapezoidal rule. It is important to be aware of this since numerical dissipation is another desirable characteristic and one

not possessed by trapezoidal rule. (Recall, numerical dissipation is used to damp out any spurious participation of the higher modes.)

It is also desirable that an algorithm be self-starting, since our understanding of ones that are not is generally obscured. For example, if an algorithm requires a distinct starting procedure, we must analyze the starting procedure in addition to the algorithm and we must also analyze the interaction of the algorithm with all possible values generated by the starting procedure. A distinct starting procedure may also engender additional coding and computational effort.

Algorithms that are not self-starting involve data from more than two time steps to advance the solution. This requires additional storage, and since no matter how many steps we include we are restricted to at most second-order accuracy, a point of diminishing return is quickly reached. Furthermore, it is possible to achieve all the aforementioned attributes within a “one-step method” (i.e., a method for which the displacements, velocities, and accelerations at one step, along with the given forces, suffice to advance the solution to the next step). *Thus it seems that the second-order accurate, unconditionally stable, one-step method which achieves the optimal balance between effective numerical dissipation and loss of accuracy compared with the trapezoidal rule is the desired algorithm for most problems of structural dynamics.*

Overshoot

In [26] Goudreau and Taylor discovered a peculiar property of the Wilson- θ method. Despite the method’s unconditional stability, numerical experiments revealed a tendency to significantly “overshoot” exact solutions in the first few steps. Argyris et al. [27] and Ghose [28] have also drawn attention to this behavior.

The overshoot phenomenon can occur despite the fact that the method’s amplification matrix is spectrally stable. This suggests that a method’s amplification matrix should be scrutinized more carefully. To see the genesis of the overshoot phenomenon in terms of a spectrally stable, hypothetical amplification matrix, consider

$$\mathbf{A} = \begin{bmatrix} \epsilon & k \\ 0 & \epsilon \end{bmatrix} \quad (9.3.28)$$

where $0 < \epsilon < 1$ and $k \gg 1$. The spectral radius of \mathbf{A} is ϵ . The effect of the spectral radius as $n \rightarrow \infty$ is evident from

$$\mathbf{A}^n = \begin{bmatrix} \epsilon^n & n\epsilon^{n-1}k \\ 0 & \epsilon^n \end{bmatrix} \quad (9.3.29)$$

in which all terms go to zero as $n \rightarrow \infty$. However, due to the presence of k , the term $n\epsilon^{n-1}k$, is very large for n small enough. From this example we see that the long-term, or asymptotic, behavior is governed by the spectral properties of \mathbf{A} . However, the short-term potential for overshoot requires scrutiny of all the entries of \mathbf{A} . To assess an algorithm’s tendency to overshoot, we may consider the SDOF model problem subjected to arbitrary initial conditions d_0 and v_0 and calculate d_1 and v_1 as a function

of Ω . The numerical solution may then be compared with the exact solution. Because we naturally consider only convergent schemes, there can be no overshoot as $\Omega \rightarrow 0$. The other end of the spectrum, $\Omega \rightarrow \infty$, gives an indication of the behavior of the high frequencies in a system in which the time step is large compared with the shortest periods present. This is a typical situation when implicit, unconditionally stable methods are applied to large multidegree-of-freedom systems.

Example

Consider the undamped, homogeneous case (i.e., $\xi = 0$ and $F = 0$) and the limit $\Omega \rightarrow \infty$. Then, the collocation and α -methods behave as follows:

Collocation schemes

$$\left(\frac{\theta}{2(\theta + 1)} \geq \beta \geq \frac{2\theta^2 - 1}{4(2\theta^3 - 1)}, \theta \geq 1 \right)$$

$$d_1 \cong -\frac{1}{2} \left(1 - \frac{1}{\theta} \right) \Omega^2 d_0 + \left(1 - \frac{1}{\theta^2} \right) \Delta t v_0 \quad (9.3.30)$$

$$v_1 \cong \left(\frac{1}{4\beta\theta} - 1 \right) \Omega \omega^h d_0 + \left(1 - \frac{1}{2\beta\theta^2} \right) v_0 \quad (9.3.31)$$

α -methods

$$(\beta = (1 - \alpha)^2/4, \gamma = \frac{1}{2} - \alpha, \alpha \in [-\frac{1}{3}, 0])$$

$$d_1 \cong \left(1 - \frac{1}{2\beta(1 + \alpha)} \right) d_0 \quad (9.3.32)$$

$$v_1 \cong \left(\frac{\gamma}{2\beta} - 1 \right) \Omega \omega^h d_0 + (1 - \gamma/\beta) v_0 \quad (9.3.33)$$

Retaining only significant terms according to order of magnitude and assuming neither $\theta = 1$ nor $\alpha = 0$, we get

Collocation schemes

$$d_1 \cong O(\Omega^2) d_0 + O(\Delta t) v_0 \quad (9.3.34)$$

$$v_1 \cong O(\Omega) \omega^h d_0 + O(1) v_0 \quad (9.3.35)$$

α -methods

$$d_1 \cong O(1) d_0 \quad (9.3.36)$$

$$v_1 \cong O(\Omega) \omega^h d_0 + O(1) v_0 \quad (9.3.37)$$

Both methods exhibit a tendency to overshoot linearly in Ω in the velocity equation due to the initial displacement terms. The α -methods exhibit no overshoot in displacement. On the other hand, d_1 of the collocation schemes overshoots quadratically in Ω due to initial displacement and linearly in Δt due to initial velocity. The superiority of the α -methods over the collocation schemes with regard to overshoot is evident.

For both families the critical condition is caused by initial displacement. In Fig. 9.3.5 numerical results for some of the present schemes are exhibited for initial data $d_0 = 1$ and $v_0 = 0$, and $\Delta t/T = 10$. The trapezoidal rule and representatives of the Newmark and Wilson methods are included for comparison purposes. The trapezoidal rule exhibits no overshoot in either the displacements or velocities. Mild overshooting is exhibited by the

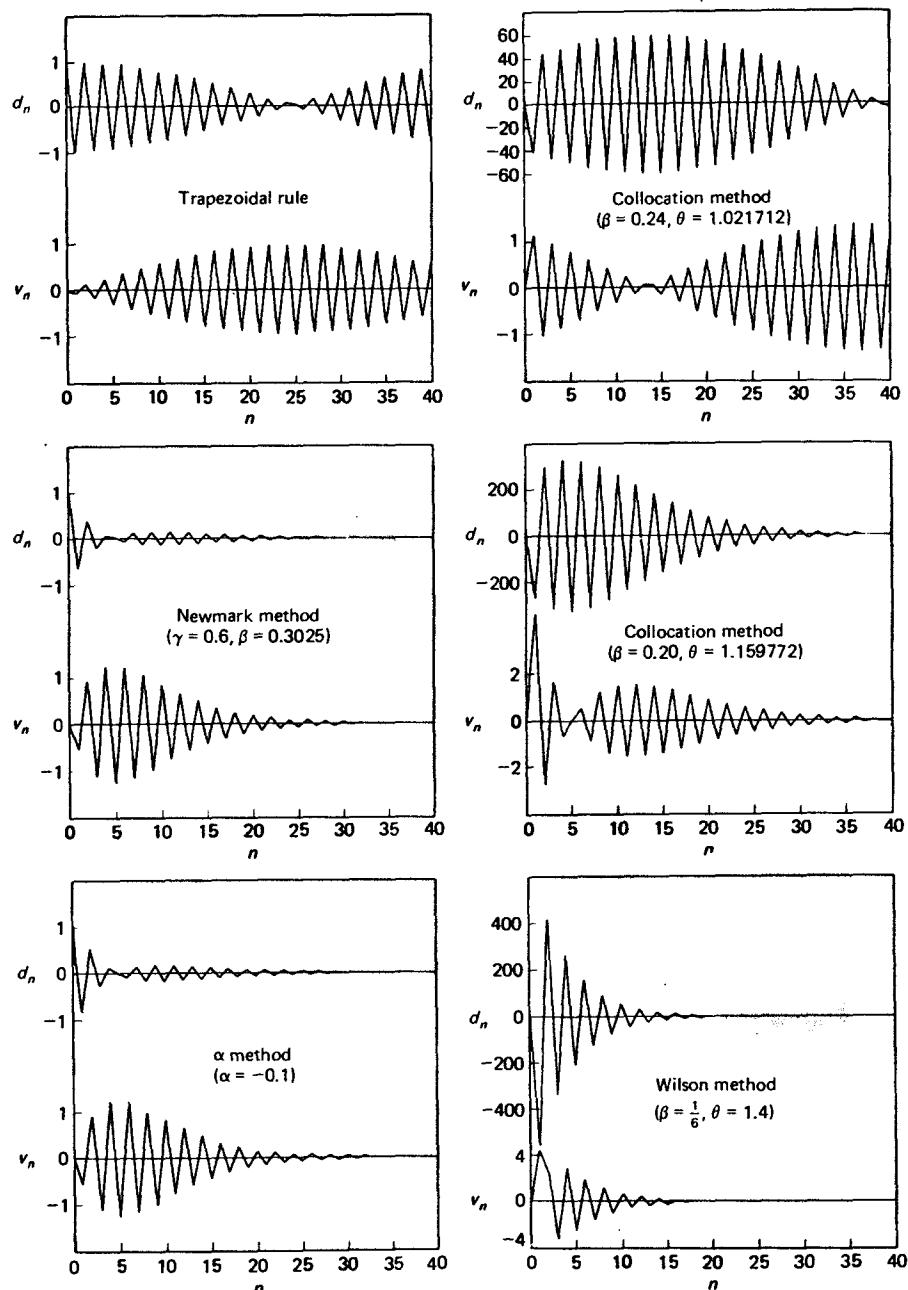


Figure 9.3.5 Comparison of overshoot response for several methods [22].

velocities in the Newmark and α -methods but none in displacements. The pathological displacement overshoot characteristics of two optimal collocation schemes and the Wilson method are manifest, whereas the velocities for these cases overshoot only mildly. Further discussion may be found in Hilber and Hughes [22].

Modal Analysis in Structural Dynamics

In comparing the use of step-by-step integration algorithms with the modal analysis technique, the points articulated in Sec. 8.5 may be reiterated here. In addition, there are other reasons that favor modal analysis for problems of structural dynamics. For example, the frequencies and mode shapes usually need to be computed for design purposes. Their availability effectively eliminates that most costly constituent in modal analysis—solution of the eigenproblem. Modal analysis is also the ideal method of “damping” the higher modes without adversely affecting the lower modes. The retained low modes may be exactly integrated. The contribution of the rest of the modes is eliminated completely.

Exercise 15. (This exercise illustrates how the analytical information determined for algorithms can be used to select stable and accurate time steps prior to performing a full-scale transient analysis.) A transient analysis is being performed of an undamped multidegree-of-freedom structure. The initial-value problem consists of the following:

$$\ddot{\mathbf{M}\mathbf{d}} + \mathbf{Kd} = \mathbf{0}$$

$$\mathbf{d}(0) = \mathbf{d}_0$$

$$\dot{\mathbf{d}}(0) = \mathbf{0}$$

Engineering insight reveals that the response will be primarily in the first six modes. Assume $0 < \omega_1^h \leq \omega_2^h \leq \dots \leq \omega_{eq}^h$, $\omega_6^h = 10\omega_1^h$, and $\omega_{eq}^h = 1000\omega_1^h$. (Recall the period of the l th mode is $T_l = 2\pi/\omega_l^h$. Engineering accuracy dictates that relative period error and amplitude decay (per cycle) be no more than 5 percent for any of the first six modes.)

Determine the largest time step permissible, as a fraction of T_1 , for the following algorithms:

- a. Central difference method ($\beta = 0$, $\gamma = \frac{1}{2}$).
- b. Trapezoidal rule ($\beta = \frac{1}{4}$, $\gamma = \frac{1}{2}$).
- c. Damped Newmark method ($\beta = 0.3025$, $\gamma = 0.6$). (It is safe to assume that dissipation errors will require a smaller time step than period errors.)
- d. α -method ($\alpha = -0.3$).
- e. Wilson- θ method ($\theta = 1.4$).
- f. Houbolt method.
- g. Park method.

Hint: You need to consider the stability condition, and dissipation and dispersion errors of each algorithm. Much of the necessary information is contained in Figs. 9.3.1–9.3.4.

Solution**a. Central difference method**

$$\bar{\xi} = 0 \Rightarrow AD = 0$$

The time step for stability is so small that we do not worry about relative period error.

$$2 = \Omega_{\text{crit}} = \omega_{n_{eq}}^h \Delta t_{\text{crit}} = \frac{2\pi}{T_{n_{eq}}} \Delta t_{\text{crit}} = \frac{2\pi}{T_1} 1000 \Delta t_{\text{crit}}$$

Therefore,

$$\frac{\Delta t}{T_1} \leq \frac{1}{(1000\pi)} \cong \underline{\underline{0.000318}}$$

b. Trapezoidal rule

$$\bar{\xi} = 0 \Rightarrow AD = 0$$

Unconditionally stable.

$$\frac{\bar{T} - T}{T} \leq 0.05 \Rightarrow \frac{\Delta t}{T} \leq 0.125$$

Therefore

$$\frac{\Delta t}{T_1} \leq 0.1 \times 0.125 = \underline{\underline{0.0125}}$$

c. Damped Newmark method ($\gamma = 0.6$, $\beta = 0.3025$)

$$\gamma > \frac{1}{2}, \quad \beta > \frac{\gamma}{2}$$

Therefore, unconditionally stable.

$$AD \cong 2\pi\bar{\xi}, \quad \bar{\xi} \leq \frac{0.05}{2\pi} \cong 0.008 \Rightarrow \frac{\Delta t}{T_6} \leq 0.03$$

Therefore,

$$\frac{\Delta t}{T_1} \leq \underline{\underline{0.003}}$$

Cases (d), (e), (f), and (g) involve *unconditionally stable* algorithms, so accuracy determines Δt :

d. α -method ($\alpha = -0.3$)

$$\bar{\xi} \leq 0.008 \Rightarrow \frac{\Delta t}{T_6} \leq 0.13 \Rightarrow \frac{\Delta t}{T_1} \leq 0.013$$

$$\frac{\bar{T} - T}{T} \leq 0.05 \Rightarrow \frac{\Delta t}{T_6} \leq 0.10 \Rightarrow \frac{\Delta t}{T_1} \leq \underline{\underline{0.010}}$$

e. Wilson-θ method

$$\bar{\xi} \leq 0.008 \Rightarrow \frac{\Delta t}{T_6} \leq 0.08 \Rightarrow \frac{\Delta t}{T_1} \leq 0.008$$

$$\frac{\bar{T} - T}{T} \leq 0.05 \Rightarrow \frac{\Delta t}{T_6} \leq 0.08 \Rightarrow \frac{\Delta t}{T_1} \leq \underline{0.008}$$

f. Houbolt method

$$\bar{\xi} \Rightarrow \frac{\Delta t}{T_1} \leq \underline{0.004}, \quad \frac{\bar{T} - T}{T} \Rightarrow \frac{\Delta t}{T_1} \leq 0.006$$

g. Park method

$$\bar{\xi} = \frac{\Delta t}{T_1} \leq 0.012, \quad \frac{\bar{T} - T}{T} \Rightarrow \frac{\Delta t}{T_1} \leq \underline{0.008}$$

Remark

From the results we see that the largest time step is permitted by the trapezoidal rule (0.0125). The second largest is attained by the α -method (0.010). The Wilson- θ and Park methods are tied for third (0.008). Houbolt's method is fifth (0.004), the damped Newmark is sixth (0.003), and the central-difference method is seventh (0.000318). Keep in mind that there may be other considerations: For example, the trapezoidal rule may produce some undesirable high-frequency oscillations; potential "overshoot" phenomena must be investigated for the Wilson- θ method; the fact that the central-difference method is explicit may compensate for the small time-step restriction.

Exercise 16. Consider the following system:

$$M\ddot{d} + Kd = 0$$

$$d = \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix}, \quad M = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}, \quad K = \begin{bmatrix} (k_1 + k_2) & -k_2 \\ -k_2 & k_2 \end{bmatrix}$$

Assume $k_1 = 10^4$, $k_2 = 1$, $m_1 = 1$, and $m_2 = 1$.

The intent of this two-degree-of-freedom model is to represent the character of typical large systems. The first mode is intended to represent those modes that are physically important and must be accurately integrated. The second mode represents the spurious high frequencies. It is desirable that the step-by-step integrator filter these high modes from the response of the system.

- Determine the natural frequencies ω_1 and ω_2 of this system.
- Consider the initial-value problem for the system above with initial data given by

$$d_0 = \begin{Bmatrix} 1 \\ 10 \end{Bmatrix}$$

$$v_0 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

(The initial condition is designed to exacerbate overshoot phenomena.)

Write a computer program to solve this problem employing methods 1–7.

1. Central difference method ($\beta = 0$, $\gamma = \frac{1}{2}$)
2. Trapezoidal rule ($\beta = \frac{1}{4}$, $\gamma = \frac{1}{2}$)
3. Damped Newmark method ($\beta = 0.3025$, $\gamma = 0.6$)
4. α -method ($\alpha = -0.3$)
5. Wilson- θ method ($\theta = 1.4$)
6. Houbolt method
7. Park method

Run the program at $\Delta t = T_i/20$ over a time interval of $[0, 5T_i]$. For the Houbolt and Park methods, use the trapezoidal rule to establish starting values. Obtain time-history plots for the displacements and velocities in each case. To facilitate data reduction use some form of computer plotting. Comment on the relative effectiveness of the algorithms.

Solution Computer response is presented in Figure 9.3.6. The calculations were performed by Alec Brooks. The reader should realize that $\Delta t = T_i/20$ leads to an unstable calculation with central differences. The critical time step for central differences is $2 \div 100.005 \approx 0.019999$. (Verify!) The central-difference calculations were run at 0.01999, just below the critical value. Note that this leads to a “beating” phenomenon in the high-frequency components (see Fig. 9.3.6(a)), which has been analytically explained by Hilber [23].

Exercise 17. Repeat Exercise 16 with $k_1 = 1$ and $k_2 = 10^4$ and the initial conditions

$$\mathbf{d}_0 = \begin{Bmatrix} 10 \\ 11 \end{Bmatrix}$$

$$\mathbf{v}_0 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

9.3.4 Some Recently Developed Algorithms for Structural Dynamics

In this section we briefly mention some more recently developed algorithms for structural dynamics.

Bossak's Method

Bossak's method [32] is defined by the Newmark formulas (i.e., (9.1.5) and (9.1.6)) and the following version of the time-discrete equation of motion (see page 551):

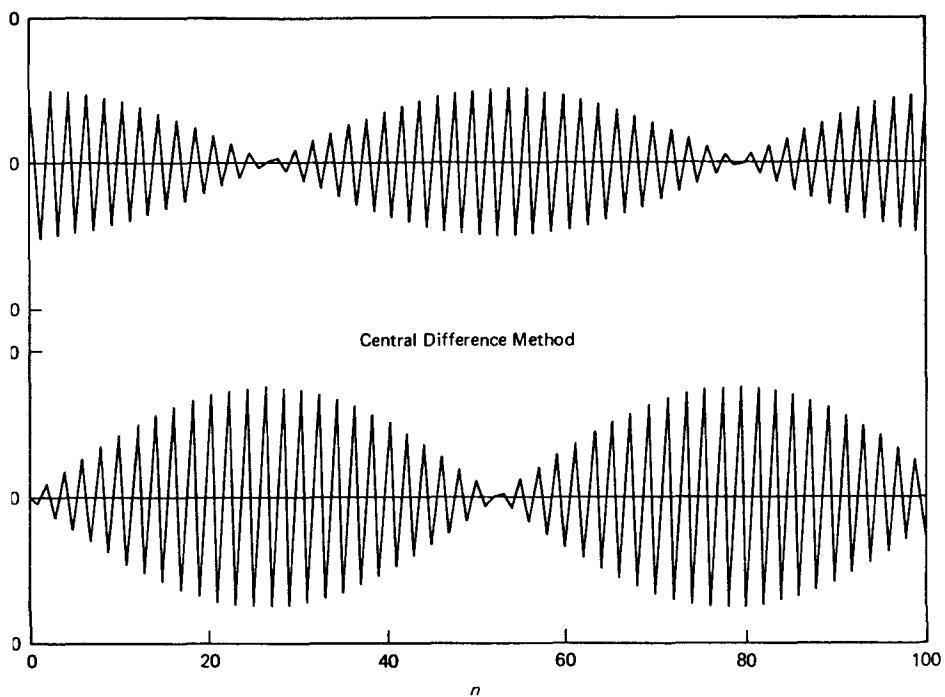


Figure 9.3.6(a)

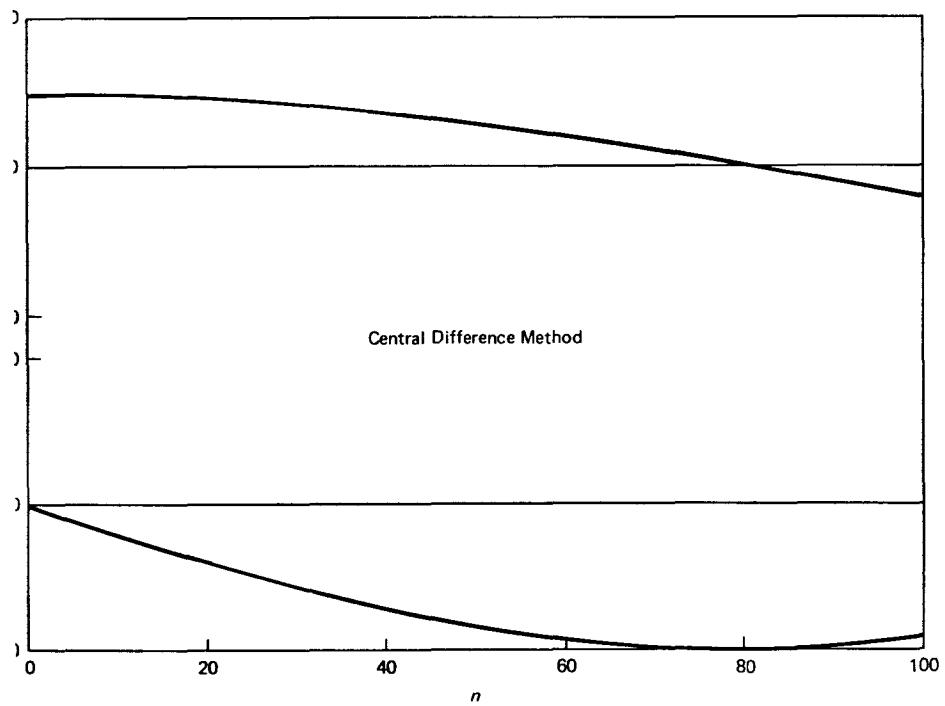


Figure 9.3.6(b)

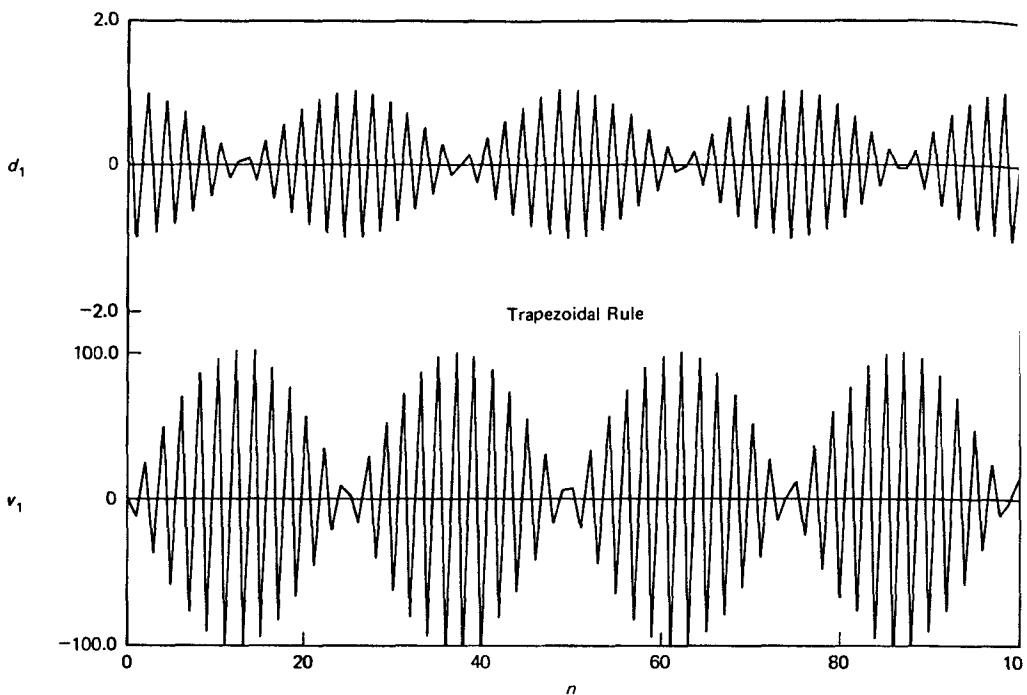


Figure 9.3.6(c)

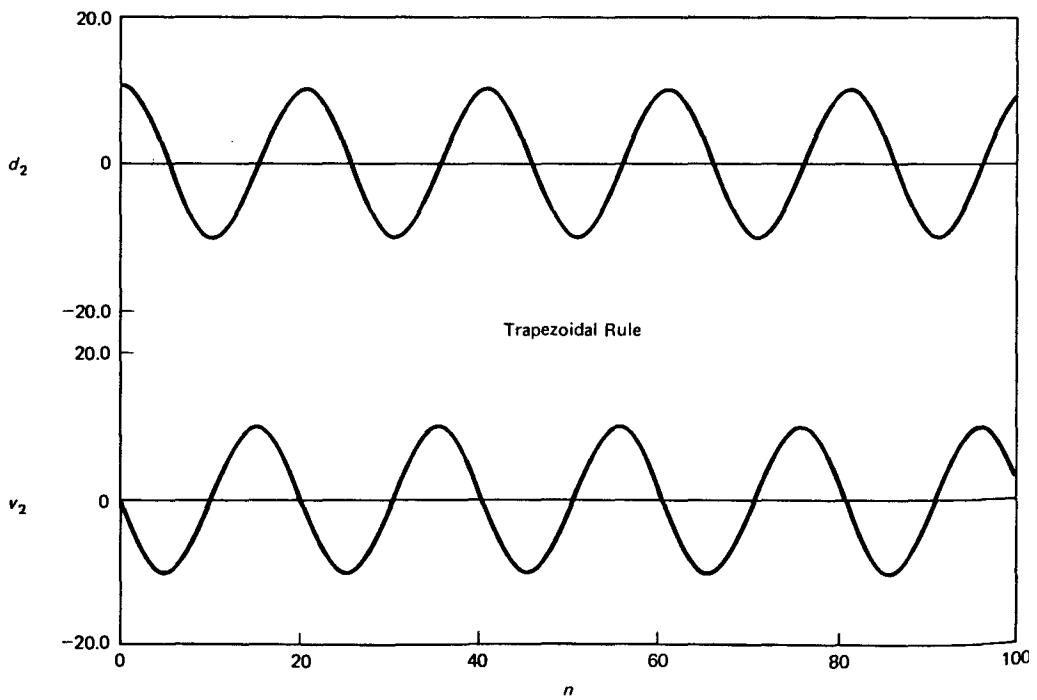


Figure 9.3.6(d)

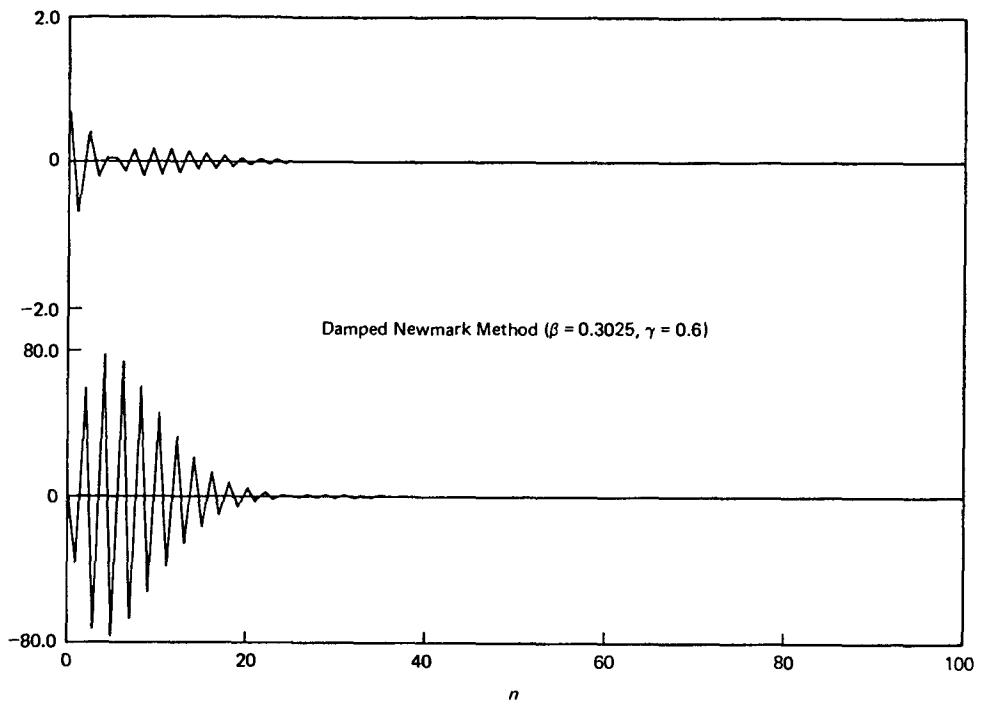


Figure 9.3.6(e)

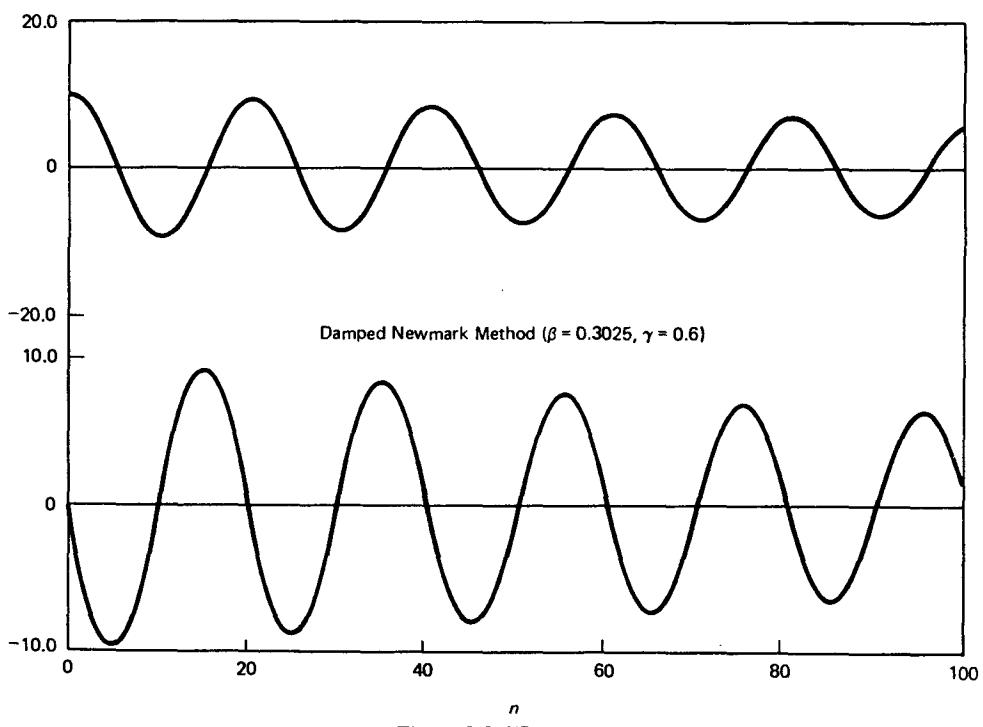


Figure 9.3.6(f)

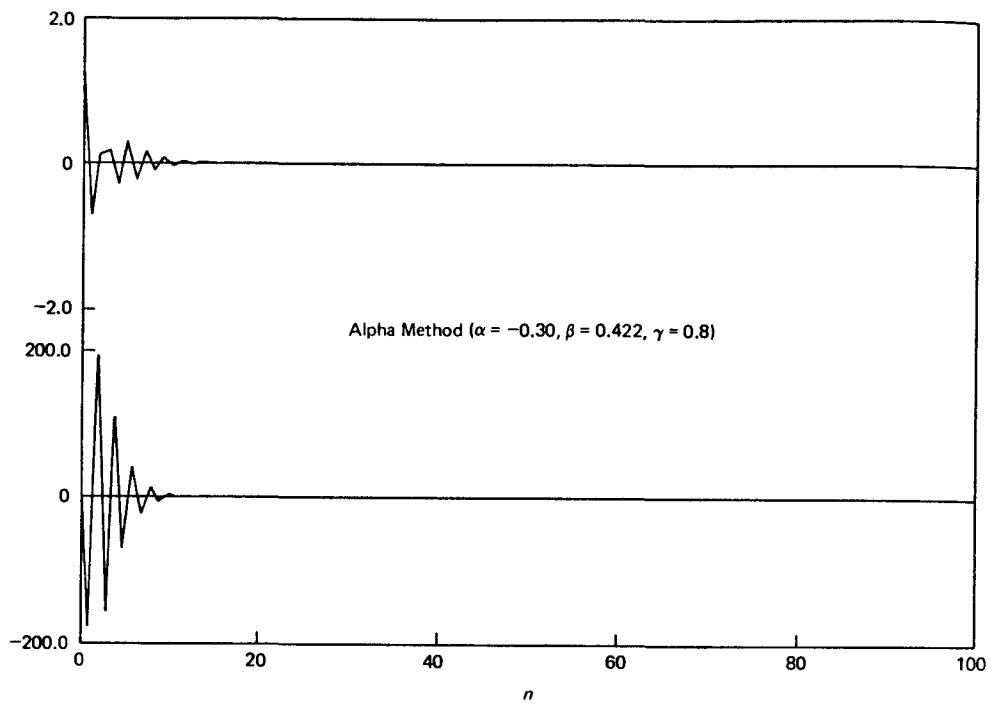


Figure 9.3.6(g)

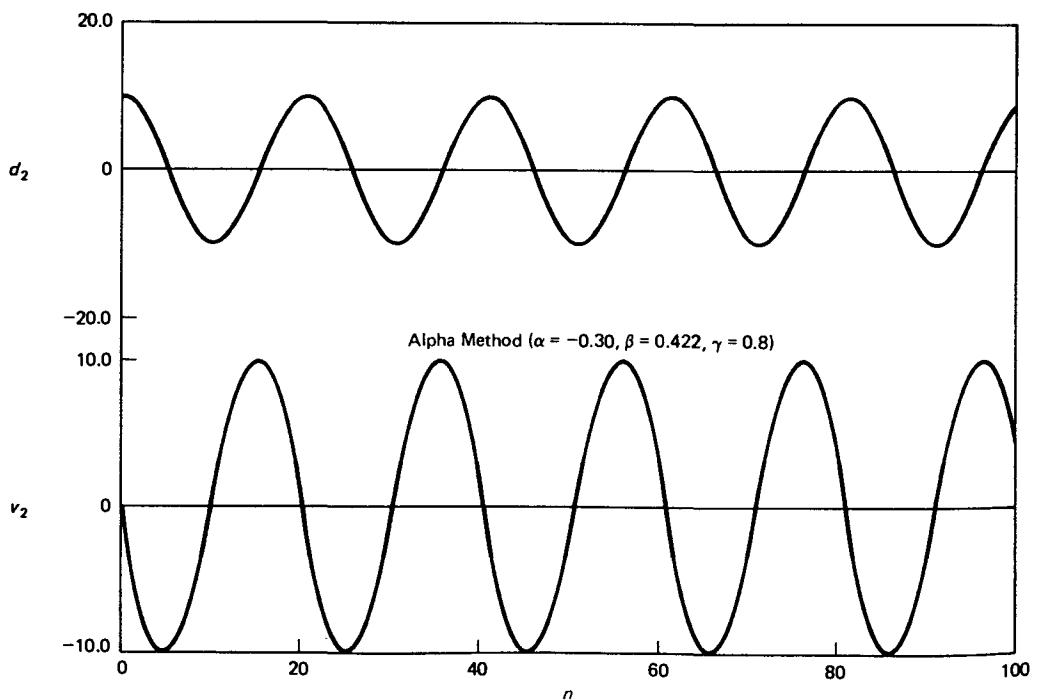


Figure 9.3.6(h)

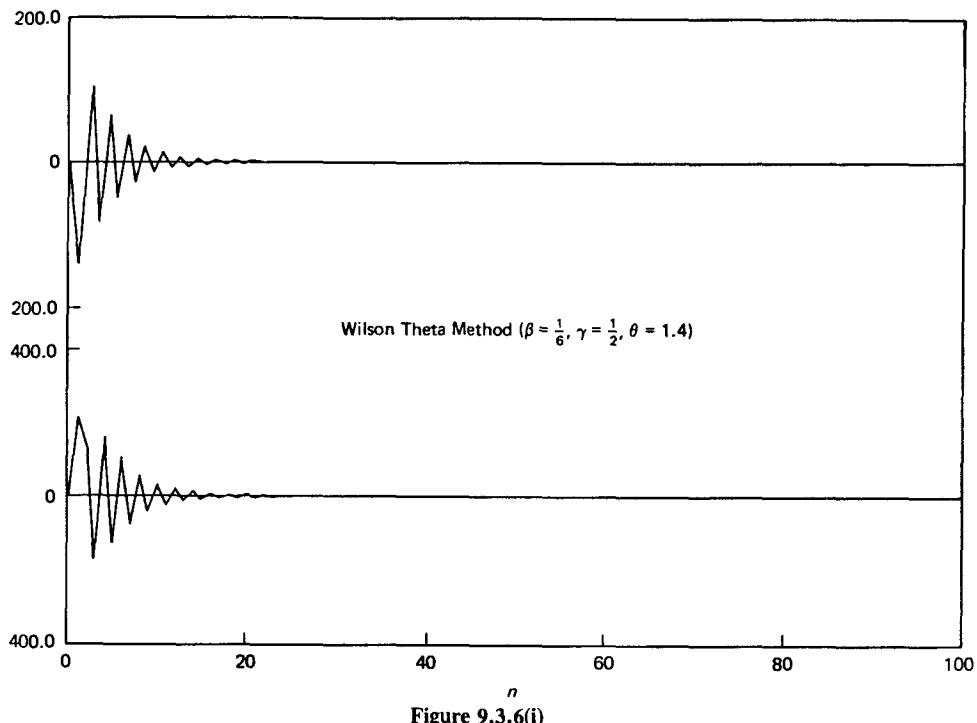


Figure 9.3.6(i)

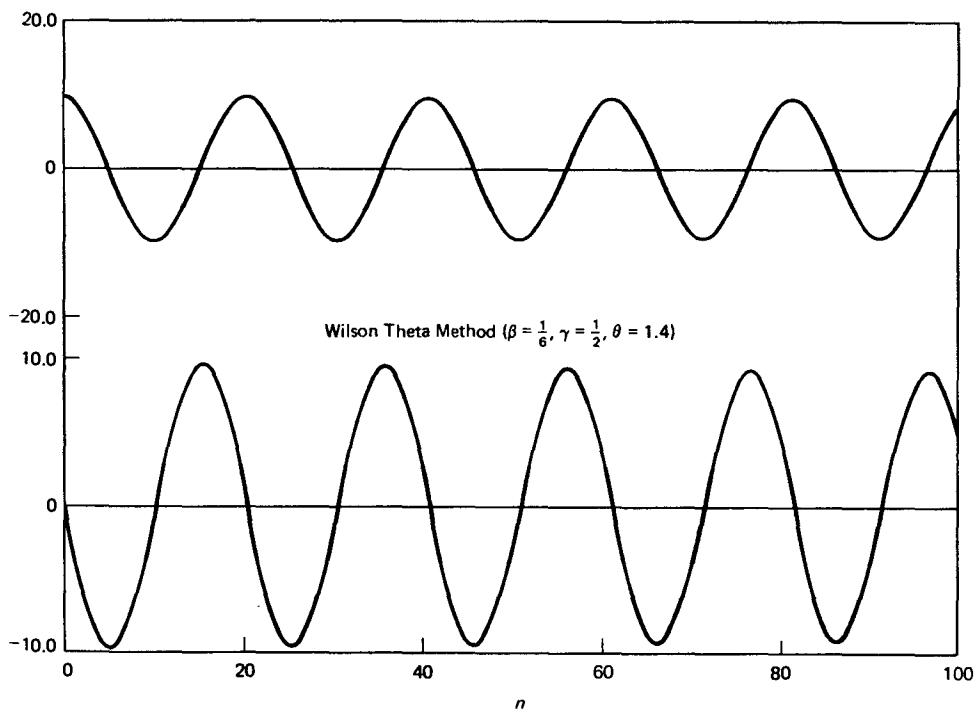


Figure 9.3.6(j)

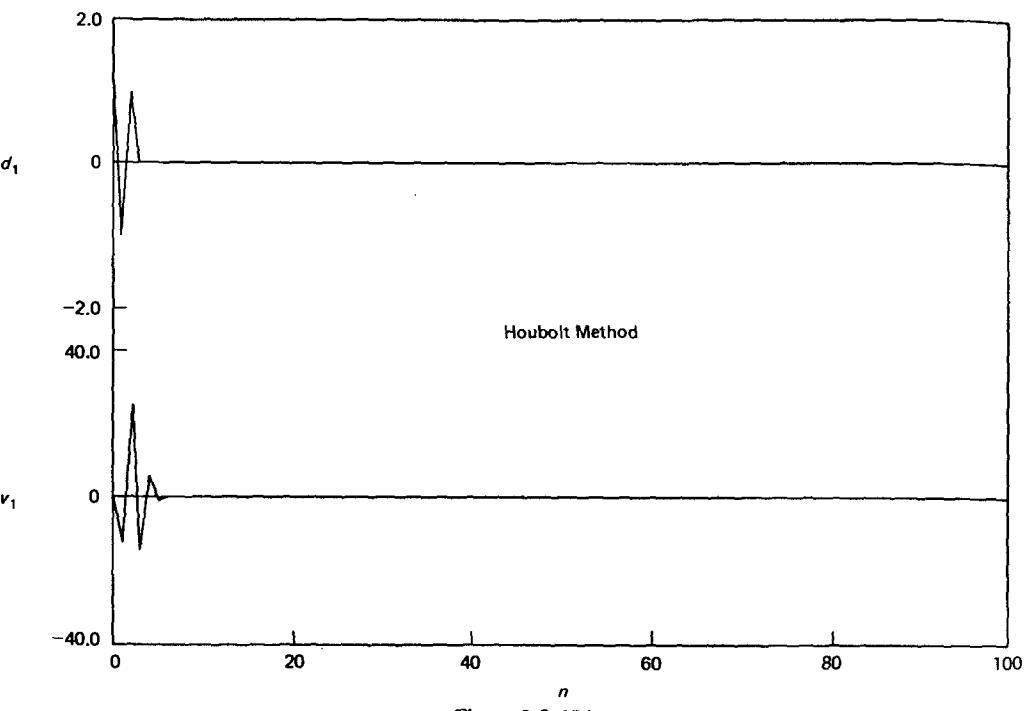


Figure 9.3.6(k)

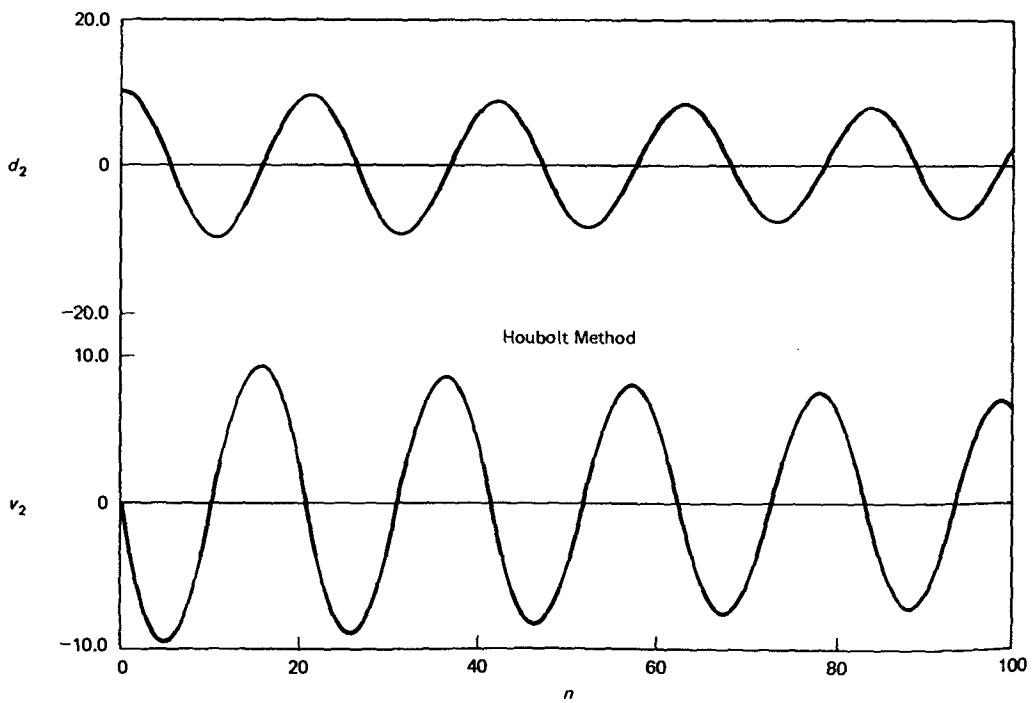


Figure 9.3.6(l)

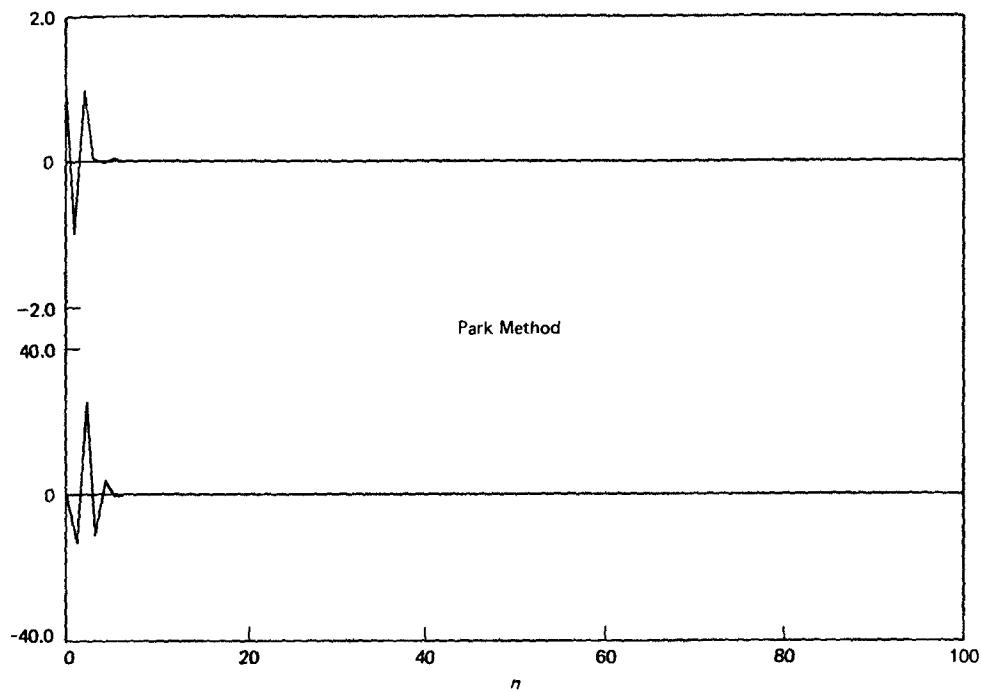


Figure 9.3.6(m)

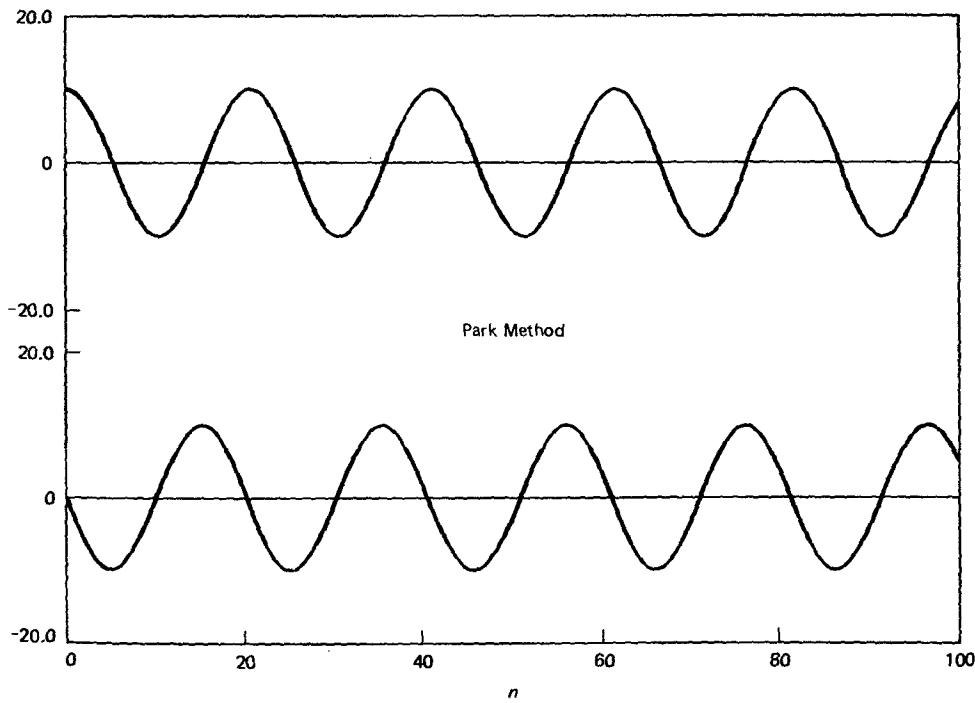


Figure 9.3.6(n)

$$(1 - \alpha_B)M\mathbf{a}_{n+1} + \alpha_B M\mathbf{a}_n + C\mathbf{v}_{n+1} + K\mathbf{d}_{n+1} = \mathbf{F}_{n+1} \quad (9.3.38)$$

where α_B is an algorithmic parameter. If $\alpha_B = 0$, we reduce to Newmark's method. Unconditional stability, second-order accuracy and maximal high-frequency dissipation are achieved if

$$\alpha_B \leq 0 \quad (9.3.39)$$

$$\gamma = \frac{1}{2} - \alpha_B \quad (9.3.40)$$

and

$$\beta = \frac{1}{4}(1 - \alpha_B)^2 \quad (9.3.41)$$

Bossak's method was inspired by and has features in common with the Hilber-Hughes-Taylor α -method. A comparison of the two methods is presented in Adams and Wood [33]. If $\alpha = \alpha_B$, both methods possess the same high-frequency dissipation properties, but the Hilber-Hughes-Taylor scheme registers an advantage in accuracy that becomes significant at values of $\alpha = \alpha_B$ near $-1/3$. For additional details the interested reader is urged to consult [32] and [33]. (The " B_1 -curve" in Fig. 1, page 1565, of [32] is wrong. A corrected version is presented in [33].)

Bazzi-Anderheggen ρ -method

The Bazzi-Anderheggen method [34] can be written in the following form:⁴

$$\mathbf{d}_{n+1} = \mathbf{d}_n + \Delta t \mathbf{v}_n + (1 - \alpha) \frac{\Delta t^2}{2} \{(1 - 2\beta)\mathbf{a}_n + 2\beta\mathbf{a}_{n+1}\} \quad (9.3.42)$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t \{(1 - \gamma)\mathbf{a}_n + \gamma\mathbf{a}_{n+1}\} \quad (9.3.43)$$

$$M[\mathbf{v}_n] + C[\mathbf{d}_n] + \Delta t K \mathbf{d}_{n+\alpha} = \Delta t \mathbf{F}_{n+\alpha} \quad (9.3.44)$$

where

$$[\mathbf{d}_n] = \mathbf{d}_{n+1} - \mathbf{d}_n, \dots \quad (9.3.45)$$

$$\mathbf{d}_{n+\alpha} = \alpha \mathbf{d}_{n+1} + (1 - \alpha) \mathbf{d}_n, \dots \quad (9.3.46)$$

$$\alpha = \frac{1}{1 + \rho} \quad (9.3.47)$$

$$\beta = \frac{1}{\rho(\rho^2 + 1)} \quad (9.3.48)$$

$$\gamma = \frac{2}{(\rho^2 + 1)(\rho + 1)} \quad (9.3.49)$$

⁴The style of presentation here is somewhat different than in [34].

where ρ is an algorithmic parameter that equals the value of the spectral radius of the amplification matrix as $\Delta t/T \rightarrow \infty$. Thus values of ρ that ensure unconditional stability are confined to the interval $[0, 1]$, with $\rho = 0$ reducing to the trapezoidal rule. If physical damping is absent, (i.e., $C = 0$), the algorithm is second-order accurate. However, if $C \neq 0$, the algorithm is limited to first-order accuracy. For the undamped case, the algorithm compares favorably with other commonly used methods. It is recommended that $\rho = 0.8$. See [34] for additional details.

Unified Set of Single-Step Methods

In [35], Zienkiewicz et al. present a unified formulation of single-step methods, which contains as special cases many of the popular structural dynamics schemes. The compact form of the algorithm enables a number of different methods to be easily implemented in one program. See [35] for further details.

9.4 ALGORITHMS BASED UPON OPERATOR SPLITTING AND MESH PARTITIONS

In recent years, methods outside the LMS realm have been introduced with considerable success. In this section we present the procedures developed in [36–38], which allow part of the mesh, or operator, to be treated implicitly and part to be treated explicitly. This has considerable practical advantages in that “stiff” subdomains of large finite element models can be treated economically with an implicit integrator. A stiff subdomain might arise from very fine local meshing, which reduces the Δt_{crit} considerably below what is allowable in the more coarsely meshed region. Penalty constraints, such as described in Sec. 4.2, also entail very small Δt_{crit} and thus it is advantageous to treat them with an unconditionally stable integrator even if the remainder of the model is to be treated explicitly. Earlier work on this subject endeavored to merge different integration methods, such as trapezoidal rule and central differences, but implementation proved somewhat cumbersome [39, 40]. In the approach presented in [36–38], only one integration procedure is used and the distinction between implicit and explicit zones is handled entirely by the finite element assembly algorithm. Implementation thus becomes trivially simple, and, at the same time, the procedure is amenable to rigorous stability [36] and convergence analyses [41]. Because the method does not fall within the LMS framework, the modal approach to stability seems inapplicable. However, an energy analysis provides necessary and sufficient conditions.

The basic ingredients in the implicit-explicit procedure advocated in [36–38] are (1) a given implicit integrator, (2) a predictor-corrector explicit scheme, which is constructed to be “compatible” with the given implicit integrator, and (3) a synthesis of the implicit and explicit schemes by way of a modified time-discrete equation of motion. We can start with *any* implicit integrator. For simplicity we shall employ the Newmark family with β assumed positive, which gives rise to implicit methods. We recall the integration formulas employed:

$$\left. \begin{aligned} \mathbf{d}_{n+1} &= \tilde{\mathbf{d}}_{n+1} + \beta \Delta t^2 \mathbf{a}_{n+1} \\ \mathbf{v}_{n+1} &= \tilde{\mathbf{v}}_{n+1} + \gamma \Delta t \mathbf{a}_{n+1} \end{aligned} \right\} \quad (9.4.1)$$

$$\left. \begin{aligned} \mathbf{v}_{n+1} &= \tilde{\mathbf{v}}_{n+1} + \gamma \Delta t \mathbf{a}_{n+1} \end{aligned} \right\} \quad (9.4.2)$$

$$\left. \begin{aligned} \tilde{\mathbf{d}}_{n+1} &= \mathbf{d}_n + \Delta t \mathbf{v}_n + \frac{\Delta t^2}{2} (1 - 2\beta) \mathbf{a}_n \\ \tilde{\mathbf{v}}_{n+1} &= \mathbf{v}_n + \Delta t (1 - \gamma) \mathbf{a}_n \end{aligned} \right\} \quad (9.4.3)$$

$$\left. \begin{aligned} \tilde{\mathbf{v}}_{n+1} &= \mathbf{v}_n + \Delta t (1 - \gamma) \mathbf{a}_n \end{aligned} \right\} \quad (9.4.4)$$

Implicit Newmark methods. The Newmark family is defined by (9.4.1) through (9.4.4) and the temporally discretized equation of motion in the form (cf. (9.1.4) through (9.1.6)):

$$\mathbf{M}\mathbf{a}_{n+1} + \mathbf{C}\mathbf{v}_{n+1} + \mathbf{K}\mathbf{d}_{n+1} = \mathbf{F}_{n+1} \quad (9.4.5)$$

For $\beta > 0$, (9.4.1) through (9.4.5) defines an implicit method. This is ingredient (1).

Explicit predictor-corrector methods. The “compatible” explicit scheme is defined by (9.1.1) through (9.1.4) and a temporally discrete equation of motion in which the stiffness and damping terms are rendered explicit:

$$\boxed{\mathbf{M}\mathbf{a}_{n+1} + \mathbf{C}\tilde{\mathbf{v}}_{n+1} + \mathbf{K}\tilde{\mathbf{d}}_{n+1} = \mathbf{F}_{n+1}} \quad (9.4.6)$$

Note that, compared with (9.4.5), (9.4.6) amounts to replacing corrector values of \mathbf{d}_{n+1} and \mathbf{v}_{n+1} with corresponding predictor values. Clearly, as long as \mathbf{M} is diagonal, \mathbf{a}_{n+1} may be determined from (9.4.6) without solving equations; \mathbf{d}_{n+1} and \mathbf{v}_{n+1} are then defined by (9.4.1) and (9.4.2). This defines ingredient (2).

Exercise 1. Assume $\gamma = \frac{1}{2}$ and $\mathbf{F} = \mathbf{0}$. Show that the displacement difference equation for the explicit predictor-corrector method is independent of β and that if in addition $\mathbf{C} = \mathbf{0}$ then it is the same as that for the Newmark method with $\beta = 0$ and $\gamma = \frac{1}{2}$. (This means that under the stated assumptions the explicit predictor-corrector scheme and central-difference method become identical up to the starting procedure.)

Implicit-explicit methods. Consider a finite element model in which the elements are divided into two groups: the implicit elements and the explicit elements. Let I and E superscripts refer to the implicit and explicit groups, respectively. In particular let \mathbf{M}^I , \mathbf{C}^I , \mathbf{K}^I , and \mathbf{F}^I (respectively, \mathbf{M}^E , \mathbf{C}^E , \mathbf{K}^E , and \mathbf{F}^E) be the assembled mass, damping, stiffness and load for the implicit (respectively, explicit) group. We assume \mathbf{M}^E is diagonal. Each of the aforementioned matrices is assumed positive-semidefinite. The implicit-explicit methods, composites of the implicit Newmark methods, and explicit, predictor-corrector methods, are given by (9.4.1) through (9.4.4) and

$$\boxed{\mathbf{M}\mathbf{a}_{n+1} + \mathbf{C}^I\mathbf{v}_{n+1} + \mathbf{C}^E\tilde{\mathbf{v}}_{n+1} + \mathbf{K}^I\mathbf{d}_{n+1} + \mathbf{K}^E\tilde{\mathbf{d}}_{n+1} = \tilde{\mathbf{F}}_{n+1}} \quad (9.4.7)$$

where

$$\mathbf{M} = \mathbf{M}^I + \mathbf{M}^E \quad (9.4.8)$$

$$\mathbf{C} = \mathbf{C}^I + \mathbf{C}^E \quad (9.4.9)$$

$$\mathbf{K} = \mathbf{K}^I + \mathbf{K}^E \quad (9.4.10)$$

$$\mathbf{F} = \mathbf{F}^I + \mathbf{F}^E \quad (9.4.11)$$

Note that in (9.4.7) the implicit arrays multiply corrector values, whereas the explicit arrays multiply predictor values. By using (9.4.1) and (9.4.2) in (9.4.7), we arrive at the equation that determines \mathbf{a}_{n+1} :

$$\boxed{\mathbf{M}^*\mathbf{a}_{n+1} = \mathbf{F}_{n+1} - \mathbf{C}\tilde{\mathbf{v}}_{n+1} - \mathbf{K}\tilde{\mathbf{d}}_{n+1}} \quad (9.4.12)$$

$$\boxed{\mathbf{M}^* = \mathbf{M} + \gamma\Delta t\mathbf{C}^I + \beta\Delta t^2\mathbf{K}^I} \quad (9.4.13)$$

Note that the only manifestation of the implicit-explicit partition is in the left-hand side coefficient matrix where only the implicit parts of \mathbf{C} and \mathbf{K} appear. The Newmark methods correspond to the choices:

$$\left. \begin{array}{l} \mathbf{C}^I = \mathbf{C}, \quad \mathbf{K}^I = \mathbf{K} \\ \mathbf{C}^E = \mathbf{0}, \quad \mathbf{K}^E = \mathbf{0} \end{array} \right\} \quad (9.4.14)$$

The explicit predictor-corrector methods correspond to:

$$\left. \begin{array}{l} \mathbf{C}^I = \mathbf{0}, \quad \mathbf{K}^I = \mathbf{0} \\ \mathbf{C}^E = \mathbf{C}, \quad \mathbf{K}^E = \mathbf{K} \end{array} \right\} \quad (9.4.15)$$

The implicit stiffness and damping contributions engender a band-profile structure, which corresponds to the connectivity of the implicit group *only*. Consequently, \mathbf{M}^* , which is symmetric and positive-definite, will have diagonal subregions emanating from the explicit elements. A simple mesh that illustrates these properties is shown in Fig. 9.4.1.

The structure of \mathbf{M}^ is fully exploited by active (compacted) column equation solvers, in which zeros outside the profile are neither stored nor operated upon; see Chapter 11.*

It may be clear to persons familiar with the computer implementation of implicit transient schemes that the implicit-explicit element methods advocated herein represent no additional complications and may be implemented into many existing implicit codes with relative ease.

Many other possibilities fit within this general framework. For example, (9.4.8) through (9.4.11) may be viewed as general definitions of “splitting” an operator. In

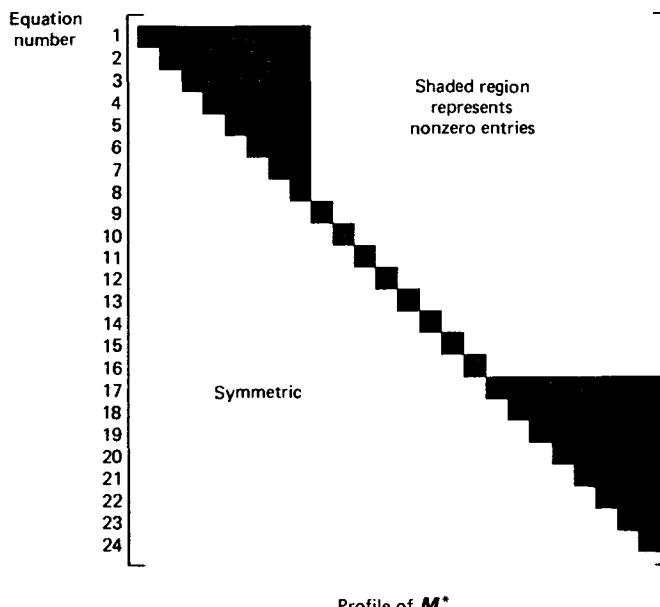
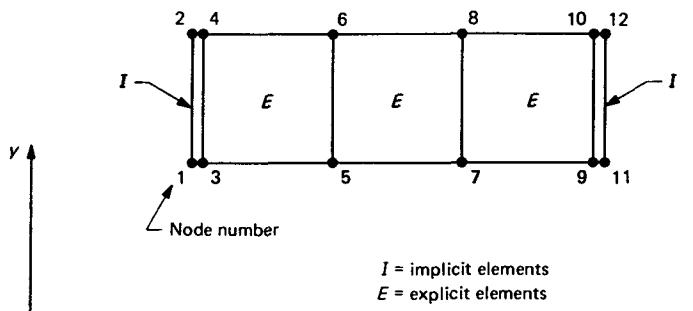


Figure 9.4.1 Profile of effective mass matrix for implicit-explicit finite element mesh partition.

this way, the procedures described in [42, 43] (sometimes referred to as “semi-implicit schemes,” or “triangular splittings”) may be subsumed. As observed by Park and Felippa [42], “nodal” partitions may also be encompassed by formulations of this type.

Exercise 2. Develop an implicit-explicit algorithm based upon the α -method described in Sec. 9.3.3. *Ans.:* The equations consist of (9.4.1) through (9.4.4) and

$$\begin{aligned} \mathbf{M}\mathbf{a}_{n+1} + (1 + \alpha)\mathbf{C}'\mathbf{v}_{n+1} + (1 + \alpha)\mathbf{C}^E\tilde{\mathbf{v}}_{n+1} - \alpha\mathbf{C}\mathbf{v}_n \\ + (1 + \alpha)\mathbf{K}'\mathbf{d}_{n+1} + (1 + \alpha)\mathbf{K}^E\tilde{\mathbf{d}}_{n+1} - \alpha\mathbf{K}\mathbf{d}_n = \mathbf{F}(t_{n+\alpha}) \end{aligned} \quad (9.4.16)$$

9.4.1 Stability via the Energy Method

The plan for obtaining stability conditions for the implicit-explicit algorithms is first to apply the energy method to the Newmark and predictor-corrector algorithms and then to show that the implicit-explicit algorithms may be reduced to a combination of the previous cases. It suffices to restrict attention to the case in which $\mathbf{F} = \mathbf{0}$ for purposes of stability.

Newmark Methods

Equations (9.4.1) through (9.4.5) may be combined to form the following identity:

$$\boxed{\begin{aligned}\mathbf{a}_{n+1}^T \mathbf{A} \mathbf{a}_{n+1} + \mathbf{v}_{n+1}^T \mathbf{K} \mathbf{v}_{n+1} &= \mathbf{a}_n^T \mathbf{A} \mathbf{a}_n + \mathbf{v}_n^T \mathbf{K} \mathbf{v}_n \\ &\quad - (2\gamma - 1)[\mathbf{a}_n]^T \mathbf{B} [\mathbf{a}_n] \\ &\quad - 2\Delta t \langle \mathbf{a}_n \rangle^T \mathbf{C} \langle \mathbf{a}_n \rangle\end{aligned}} \quad (9.4.17)$$

where

$$\mathbf{B} = \mathbf{M} + \Delta t \left(\gamma - \frac{1}{2} \right) \mathbf{C} + \Delta t^2 \left(\beta - \frac{\gamma}{2} \right) \mathbf{K} \quad (9.4.18)$$

$$\mathbf{A} = \mathbf{B} + \Delta t \left(\gamma - \frac{1}{2} \right) \mathbf{C} \quad (9.4.19)$$

(There should be no confusion over the present definition of the matrix \mathbf{A} and earlier use for denoting an amplification matrix.)

Theorem. If $\gamma \geq \frac{1}{2}$ and \mathbf{B} is positive definite, then \mathbf{a}_n and \mathbf{v}_n are bounded.

Proof: The hypotheses imply \mathbf{A} is positive definite and

$$\mathbf{a}_{n+1}^T \mathbf{A} \mathbf{a}_{n+1} + \mathbf{v}_{n+1}^T \mathbf{K} \mathbf{v}_{n+1} \leq \mathbf{a}_n^T \mathbf{A} \mathbf{a}_n + \mathbf{v}_n^T \mathbf{K} \mathbf{v}_n \quad (9.4.20)$$

which in turn implies

$$\mathbf{a}_n^T \mathbf{A} \mathbf{a}_n + \mathbf{v}_n^T \mathbf{K} \mathbf{v}_n \leq \mathbf{a}_0^T \mathbf{A} \mathbf{a}_0 + \mathbf{v}_0^T \mathbf{K} \mathbf{v}_0, \quad n = 1, 2, 3, \dots \quad (9.4.21)$$

from which the conclusions follow. ■

Remarks

1. If \mathbf{K}^{-1} exists, then \mathbf{d}_n is also bounded. This follows from (9.4.5).
2. To establish the stability conditions we need only to determine when \mathbf{B} is positive-definite. We may use the normal modes to diagonalize \mathbf{B} , from which it follows that

$$1 + 2\xi\left(\gamma - \frac{1}{2}\right)\omega^h\Delta t + \left(\beta - \frac{\gamma}{2}\right)(\omega^h\Delta t)^2 \quad (9.4.22)$$

must be positive for each mode in the system. Equation (9.4.22) leads directly to the conditions obtained earlier [cf. (9.1.16) through (9.1.20)].

Predictor-Corrector Methods

Equations (9.4.1) through (9.4.4) and (9.4.6) lead to the identity

$$\begin{aligned} \mathbf{a}_{n+1}^T \bar{\mathbf{A}} \mathbf{a}_{n+1} + \mathbf{v}_{n+1}^T \mathbf{K} \mathbf{v}_{n+1} &= \mathbf{a}_n^T \bar{\mathbf{A}} \mathbf{a}_n + \mathbf{v}_n^T \mathbf{K} \mathbf{v}_n \\ &\quad - (2\gamma - 1)[\mathbf{a}_n]^T \bar{\mathbf{B}} [\mathbf{a}_n] \\ &\quad - 2\Delta t \langle \mathbf{a}_n \rangle^T \mathbf{C} \langle \mathbf{a}_n \rangle \end{aligned} \quad (9.4.23)$$

where

$$\bar{\mathbf{B}} = \mathbf{B} - \Delta t \gamma \mathbf{C} - \Delta t^2 \beta \mathbf{K} \quad (9.4.24)$$

$$\bar{\mathbf{A}} = \bar{\mathbf{B}} + \Delta t \left(\gamma - \frac{1}{2} \right) \mathbf{C} \quad (9.4.25)$$

Theorem: If $\gamma \geq \frac{1}{2}$ and $\bar{\mathbf{B}}$ is positive-definite, then \mathbf{a}_n and \mathbf{v}_n are bounded.

Proof: The proof is identical to that for the previous theorem, with $\bar{\mathbf{A}}$ in place of \mathbf{A} . ■

Remarks

1. If \mathbf{K}^{-1} exists, then \mathbf{d}_n is also bounded. This follows from (9.4.6), where $\tilde{\mathbf{d}}_{n+1}$ and $\tilde{\mathbf{v}}_{n+1}$ are eliminated by (9.4.1) and (9.4.2), respectively.

2. The stability characteristics of the predictor-corrector methods are determined by the conditions that render $\bar{\mathbf{B}}$ positive-definite. The same argument as before leads to the condition that

$$1 - \xi\omega^h\Delta t - \frac{\gamma}{2}(\omega^h\Delta t)^2 \quad (9.4.26)$$

must be positive for each mode of the system. Thus the following two conditions must be satisfied:

$$\gamma \geq \frac{1}{2} \quad (9.4.27)$$

$$\Omega \leq \Omega_{\text{crit}} \stackrel{\text{def}}{=} \frac{(\xi^2 + 2\gamma)^{1/2} - \xi}{\gamma} \quad (9.4.28)$$

Observe that (9.4.27) and (9.4.28) are independent of β .

3. It is to be expected that there are no unconditionally stable predictor-corrector schemes, since they are all explicit linear multistep methods.

4. Spectral radii are presented in Fig. 9.4.2 for the case $\xi = 0$. The Ω_{crit} obtained from the diagram is consistent with (9.4.28). The slope discontinuity in the spectral radius curves corresponds to the point at which the complex conjugate, principal roots of the amplification matrix become real and bifurcate. The value of Ω at which bifurcation occurs is

$$\Omega_{\text{bif}} = \frac{2(1 - \xi)}{\gamma + \frac{1}{2}} \quad (9.4.29)$$

To ensure high-frequency numerical dissipation when $\zeta < 1$, Ω should be kept below Ω_{bif} , resulting in a somewhat smaller time step than that required by Ω_{crit} ; cf. (9.4.28).

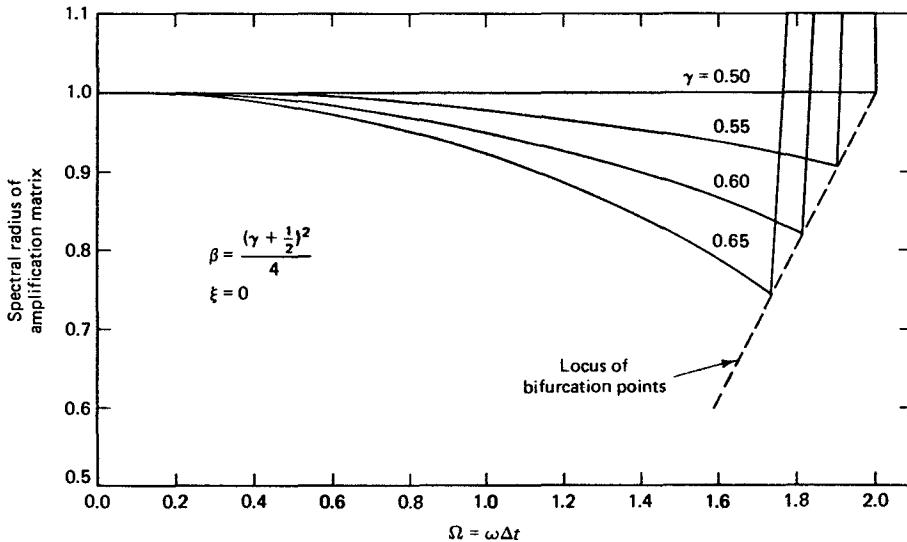


Figure 9.4.2 Spectral radius of amplification matrix for predictor-corrector algorithms [3].

5. Increasing ξ and γ decreases the critical time step for the predictor-corrector algorithms. Thus an estimate of ξ is essential in determining a stable time step.

6. When viscous damping is absent, $\Omega_{\text{crit}} = (2/\gamma)^{1/2}$. The maximum occurs for $\gamma = \frac{1}{2}$. That is $\Omega_{\text{crit}} = 2$, which is the same result as for the central difference method.

7. A local truncation-error analysis of the predictor-corrector schemes indicates that if $C = 0$, $\gamma = \frac{1}{2}$ is a necessary and sufficient condition for second-order accuracy. If $C \neq 0$, then first-order accuracy is attained for all $\gamma \geq \frac{1}{2}$. (A generalization of the basic scheme, which enables second-order accuracy to be regained under these circumstances, will be discussed subsequently.)

8. Stability conditions for the first-order case (i.e., when $K = 0$) are easily

deducible from the theorem. In this case, \bar{B} is positive-definite when $\lambda \Delta t < 2$, where λ is the maximum eigenvalue of $M^{-1}C$. (Note: $\gamma \geq \frac{1}{2}$ is still required.)

Implicit-Explicit Algorithms

An identity analogous to those for the previous cases may be derived from (9.4.1) through (9.4.4) and (9.4.7):

$$\begin{aligned} \mathbf{a}_{n+1}^T (\mathbf{A}^I + \bar{\mathbf{A}}^E) \mathbf{a}_{n+1} + \mathbf{v}_{n+1}^T \mathbf{K} \mathbf{v}_{n+1} &= \mathbf{a}_n^T (\mathbf{A}^I + \bar{\mathbf{A}}^E) \mathbf{a}_n + \mathbf{v}_n^T \mathbf{K} \mathbf{v}_n \\ &\quad - (2\gamma - 1) [\mathbf{a}_n]^T (\mathbf{B}^I + \bar{\mathbf{B}}^E) [\mathbf{a}_n] \\ &\quad - 2\Delta t \langle \mathbf{a}_n \rangle^T \mathbf{C} \langle \mathbf{a}_n \rangle \end{aligned} \quad (9.4.30)$$

where

$$\mathbf{B}^I = \mathbf{M}^I + \Delta t \left(\gamma - \frac{1}{2} \right) \mathbf{C}^I + \Delta t^2 \left(\beta - \frac{\gamma}{2} \right) \mathbf{K}^I \quad (9.4.31)$$

$$\mathbf{A}^I = \mathbf{B}^I + \Delta t \left(\gamma - \frac{1}{2} \right) \mathbf{C}^I \quad (9.4.32)$$

$$\bar{\mathbf{B}}^E = \mathbf{B}^E - \Delta t \gamma \mathbf{C}^E - \Delta t^2 \beta \mathbf{K}^E \quad (9.4.33)$$

$$\mathbf{B}^E = \mathbf{M}^E + \Delta t \left(\gamma - \frac{1}{2} \right) \mathbf{C}^E + \Delta t^2 \left(\beta - \frac{\gamma}{2} \right) \mathbf{K}^E \quad (9.4.34)$$

$$\bar{\mathbf{A}}^E = \bar{\mathbf{B}}^E + \Delta t \left(\gamma - \frac{1}{2} \right) \mathbf{C}^E \quad (9.4.35)$$

Theorem: If $\gamma \geq \frac{1}{2}$ and $\mathbf{B}^I + \bar{\mathbf{B}}^E$ is positive definite, then \mathbf{a}_n and \mathbf{v}_n are bounded.

Proof: Again, the proof is identical to that for the first theorem with $\mathbf{A}^I + \bar{\mathbf{A}}^E$ in place of \mathbf{A} . ■

Remarks

1. If \mathbf{K}^{-1} exists, then \mathbf{d}_n is also bounded.
2. $\mathbf{B}^I + \bar{\mathbf{B}}^E$ is rendered positive definite, and stability is thereby achieved, when β , γ , and Δt satisfy (9.1.16) or (9.1.17)–(9.1.20) and (9.4.28). The time-step restrictions, (9.1.19), (9.1.20), and (9.4.28), pertain only to the implicit and explicit element groups, respectively.
3. In practice, one might as well achieve unconditional stability in the implicit element group. Thus for a given value of $\gamma \geq \frac{1}{2}$, β could be selected according to (9.1.55). The time-step restriction then emanates from satisfaction of (9.4.28), or (9.4.29), in the explicit element group. For example, if $\gamma = \frac{1}{2}$, (9.1.55) yields $\beta = \frac{1}{4}$, resulting in the trapezoidal rule algorithm for the implicit group. In the absence of viscous damping in the explicit element group, (9.4.28) or (9.4.29) results in

$\Omega_{\text{crit}} = 2$. A sufficient condition for stability then is that $\Omega \leq 2$ for the maximum natural frequency of the explicit elements.

Derivation of (9.4.17), (9.4.23), and (9.4.30)

We work with the undivided forward-difference and mean-value operators (i.e., [] and ⟨ ⟩) introduced earlier. Assume we have a symmetric array A . Then it may be easily verified that

$$\mathbf{x}_{n+\alpha}^T A [\mathbf{x}_n] = \langle \mathbf{x}_n \rangle^T A [\mathbf{x}_n] + \left(\alpha - \frac{1}{2} \right) [\mathbf{x}_n]^T A [\mathbf{x}_n] \quad (9.4.36)$$

$$\langle \mathbf{x}_n \rangle^T A [\mathbf{x}_n] = \frac{1}{2} [\mathbf{x}_n^T A \mathbf{x}_n] \quad (9.4.37)$$

Equations (9.4.1) through (9.4.4) may be written in the form:

$$[\mathbf{v}_n] = \Delta t \mathbf{a}_{n+\gamma} \quad (9.4.38)$$

$$[\mathbf{d}_n] = \Delta t \langle \mathbf{v}_n \rangle + \frac{\Delta t^2}{2} (2\beta - \gamma) [\mathbf{a}_n] \quad (9.4.39)$$

Let

$$T(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T M \mathbf{x} \quad (9.4.40)$$

$$U(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T K \mathbf{x} \quad (9.4.41)$$

$$D(\mathbf{x}) = -\mathbf{x}^T C \mathbf{x} \quad (9.4.42)$$

Apply the []-operator to the time-discrete equation of motion, assuming $\mathbf{F} = \mathbf{0}$, and premultiply by $[\mathbf{v}_n]$:

$$\begin{aligned} 0 &= [\mathbf{v}_n]^T (M[\mathbf{a}_n] + C[\mathbf{v}_n] + K[\mathbf{d}_n]) \\ &= \Delta t \mathbf{a}_{n+\gamma}^T M[\mathbf{a}_n] + [\mathbf{v}_n]^T C[\mathbf{v}_n] + \langle \mathbf{v}_n^T \rangle K \Delta t [\mathbf{v}_n] \\ &\quad + \Delta t \mathbf{a}_{n+\gamma}^T \frac{\Delta t^2}{2} (2\beta - \gamma) [\mathbf{a}_n] \quad (\text{by (9.4.38) and (9.4.39)}) \\ &= \Delta t \left\{ [T(\mathbf{a}_n)] + 2 \left(\gamma - \frac{1}{2} \right) T([\mathbf{a}_n]) \right\} - D([\mathbf{v}_n]) + \Delta t [U(\mathbf{v}_n)] \\ &\quad + \frac{\Delta t^3}{2} (2\beta - \gamma) \left\{ [U(\mathbf{a}_n)] + 2 \left(\gamma - \frac{1}{2} \right) U([\mathbf{a}_n]) \right\} \\ &\quad (\text{by (9.4.36), (9.4.37), (9.4.40)–(9.4.42)}) \quad (9.4.43) \end{aligned}$$

Note that

$$\begin{aligned} -D([\mathbf{v}_n]) &= \Delta t^2 \mathbf{a}_{n+\gamma}^T C \mathbf{a}_{n+\gamma} \\ &= \Delta t^2 \left\{ \langle \mathbf{a}_n \rangle^T C \langle \mathbf{a}_n \rangle + 2 \left(\gamma - \frac{1}{2} \right) \langle \mathbf{a}_n \rangle^T C [\mathbf{a}_n] + \left(\gamma - \frac{1}{2} \right)^2 [\mathbf{a}_n]^T C [\mathbf{a}_n] \right\} \end{aligned}$$

$$= -\Delta t^2 \left\{ D(\langle \mathbf{a}_n \rangle) + \left(\gamma - \frac{1}{2} \right) [D(\mathbf{a}_n)] + \left(\gamma - \frac{1}{2} \right)^2 D([\mathbf{a}_n]) \right\} \quad (9.4.44)$$

Employing (9.4.44) in (9.4.43) enables us to write

$$\begin{aligned} & [T(\mathbf{a}_n)] + \frac{\Delta t^2}{2} (2\beta - \gamma) [U(\mathbf{a}_n)] - \Delta t \left(\gamma - \frac{1}{2} \right) [D(\mathbf{a}_n)] + [U(\mathbf{v}_n)] \\ & = -2 \left(\gamma - \frac{1}{2} \right) T([\mathbf{a}_n]) - \frac{\Delta t^2}{2} (2\beta - \gamma) 2 \left(\gamma - \frac{1}{2} \right) U([\mathbf{a}_n]) \\ & \quad + \Delta t \left(\gamma - \frac{1}{2} \right)^2 D([\mathbf{a}_n]) + \Delta t D(\langle \mathbf{a}_n \rangle) \end{aligned} \quad (9.4.45)$$

Rearranging:

$$\begin{aligned} & \langle \mathbf{a}_n \rangle^T \left\{ \mathbf{M} + \frac{\Delta t^2}{2} (2\beta - \gamma) \mathbf{K} + 2\Delta t \left(\gamma - \frac{1}{2} \right) \mathbf{C} \right\} [\mathbf{a}_n] + [U(\mathbf{v}_n)] \\ & = - \left(\gamma - \frac{1}{2} \right) [\mathbf{a}_n]^T \left\{ \mathbf{M} + \frac{\Delta t^2}{2} (2\beta - \gamma) \mathbf{K} + \Delta t \left(\gamma - \frac{1}{2} \right) \mathbf{C} \right\} [\mathbf{a}_n] \\ & \quad - \Delta t \langle \mathbf{a}_n \rangle^T \mathbf{C} \langle \mathbf{a}_n \rangle \end{aligned} \quad (9.4.46)$$

It follows directly from (9.4.46) that

$$\begin{aligned} & \frac{1}{2} \mathbf{a}_{n+1}^T \mathbf{A} \mathbf{a}_{n+1} + \frac{1}{2} \mathbf{v}_{n+1}^T \mathbf{K} \mathbf{v}_{n+1} \\ & = \frac{1}{2} \mathbf{a}_n^T \mathbf{A} \mathbf{a}_n + \frac{1}{2} \mathbf{v}_n^T \mathbf{K} \mathbf{v}_n - \left(\gamma - \frac{1}{2} \right) [\mathbf{a}_n]^T \mathbf{B} [\mathbf{a}_n] - \Delta t \langle \mathbf{a}_n \rangle^T \mathbf{C} \langle \mathbf{a}_n \rangle \end{aligned} \quad (9.4.47)$$

which is identical to (9.4.17) where \mathbf{A} and \mathbf{B} are defined by (9.4.19) and (9.4.18), respectively.

To derive (9.4.23) we observe that $\tilde{\mathbf{d}}_{n+1}$ and $\tilde{\mathbf{v}}_{n+1}$ can be eliminated from (9.4.6) by way of (9.4.1) and (9.4.2) respectively. This results in

$$\bar{\mathbf{M}} \mathbf{a}_{n+1} + \mathbf{C} \mathbf{v}_{n+1} + \mathbf{K} \mathbf{d}_{n+1} = \mathbf{F}_{n+1} \quad (9.4.48)$$

where

$$\bar{\mathbf{M}} = \mathbf{M} - \gamma \Delta t \mathbf{C} - \beta \Delta t^2 \mathbf{K} \quad (9.4.49)$$

Thus the predictor-corrector schemes consist of (9.4.1) through (9.4.4), (9.4.48) and (9.4.49). Note that (9.4.1) through (9.4.4) and (9.4.48) are Newmark's algorithm for an equation of motion in which the mass matrix is given by (9.4.49). Therefore the analysis leading to (9.4.47) applies to the predictor-corrector method if \mathbf{M} is replaced by $\bar{\mathbf{M}}$. This immediately gives rise to (9.4.23) through (9.4.25).

A similar strategy is used to derive (9.4.30). Substitute (9.4.1) and (9.4.2) in (9.4.7) to arrive at

$$\bar{\bar{\mathbf{M}}} \mathbf{a}_{n+1} + \mathbf{C} \mathbf{v}_{n+1} + \mathbf{K} \mathbf{d}_{n+1} = \mathbf{F}_{n+1} \quad (9.4.50)$$

where

$$\bar{\bar{\mathbf{M}}} = \mathbf{M} - \gamma \Delta t \mathbf{C}^E - \beta \Delta t^2 \mathbf{K}^E \quad (9.4.51)$$

Equation (9.4.47) with \mathbf{M} replaced by $\bar{\bar{\mathbf{M}}}$ results in (9.4.30). (The details are left as an exercise.)

9.4.2 Predictor/Multicorrector Algorithms [38]

Generalization of the previously described schemes to a predictor/multicorrector format leads to improved characteristics in some circumstances and is suggestive of further algorithmic generalizations. The implementational sequence is given as follows:

$$i = 0 \quad (i \text{ is the iteration counter}) \quad (9.4.52)$$

$$\mathbf{d}_{n+1}^{(i)} = \tilde{\mathbf{d}}_{n+1} \quad (9.4.53)$$

$$\left. \begin{array}{l} \mathbf{v}_{n+1}^{(i)} = \tilde{\mathbf{v}}_{n+1} \\ \mathbf{a}_{n+1}^{(i)} = \mathbf{0} \end{array} \right\} \quad (\text{predictor phase}) \quad (9.4.54)$$

$$\mathbf{a}_{n+1}^{(i)} = \mathbf{0} \quad (9.4.55)$$

$$\Delta \mathbf{F}_{n+1}^{(i)} = \sum_{\epsilon=1}^{n_{el}} (\mathbf{f}_{n+1}^{\text{ext}} - m^\epsilon \mathbf{a}_{n+1}^{(i)} - c^\epsilon \mathbf{v}_{n+1}^{(i)} - k^\epsilon \mathbf{d}_{n+1}^{(i)}) \quad (9.4.56)$$

$$\mathbf{M}^* \Delta \mathbf{a} = \Delta \mathbf{F}_{n+1}^{(i)} \quad (9.4.57)$$

$$\left. \begin{array}{l} \mathbf{a}_{n+1}^{(i+1)} = \mathbf{a}_{n+1}^{(i)} + \Delta \mathbf{a} \\ \mathbf{v}_{n+1}^{(i+1)} = \tilde{\mathbf{v}}_{n+1} + \Delta t \gamma \mathbf{a}_{n+1}^{(i+1)} \end{array} \right\} \quad (\text{corrector phase}) \quad (9.4.58)$$

$$\mathbf{d}_{n+1}^{(i+1)} = \tilde{\mathbf{d}}_{n+1} + \Delta t^2 \beta \mathbf{a}_{n+1}^{(i+1)} \quad (9.4.59)$$

$$\mathbf{d}_{n+1}^{(i+1)} = \tilde{\mathbf{d}}_{n+1} + \Delta t^2 \beta \mathbf{a}_{n+1}^{(i+1)} \quad (9.4.60)$$

If additional iterations are to be performed, i is replaced by $i + 1$, and calculations resume with (9.4.56). When the iterative phase is completed, the solution is defined by the last iterates (i.e., (9.4.58) through (9.4.60)).

The definition of \mathbf{M}^* emanates from the time-discrete equation of motion, which for the predictor/multicorrector algorithms (allowing for operator and/or mesh partitions), takes the form:

$$\mathbf{M} \mathbf{a}_{n+1}^{(i+1)} + \mathbf{C}^I \mathbf{v}_{n+1}^{(i+1)} + \mathbf{C}^E \mathbf{v}_{n+1}^{(i)} + \mathbf{K}' \mathbf{d}_{n+1}^{(i+1)} + \mathbf{K}^E \mathbf{d}_{n+1}^{(i)} = \mathbf{F}_{n+1} \quad (9.4.61)$$

In this case

$$\mathbf{M}^* = \mathbf{M} + \Delta t \gamma \mathbf{C}^I + \Delta t^2 \beta \mathbf{K}' \quad (9.4.62)$$

Remarks

1. If only one pass is made through (9.4.52) through (9.4.60), the algorithm is equivalent to the implicit-explicit scheme given earlier.

2. If any part of C is treated explicitly, an additional pass through (9.4.56) through (9.4.60) can increase accuracy.

To see this in a simple setting, it suffices to consider the following special case:

$$M\alpha_{n+1}^{(i+1)} + Cv_{n+1}^{(i)} = \mathbf{0} \quad (9.4.63)$$

It may then be shown that one additional pass through (9.4.56) through (9.4.60) results in

$$v_{n+1} = (I - \Delta t M^{-1} C + \gamma(\Delta t M^{-1} C)^2)v_n \quad (9.4.64)$$

which is clearly second-order accurate if $\gamma = \frac{1}{2}$.

The stability condition is easily determined from (9.4.64): $\Omega < \Omega_{\text{crit}} = 1/\gamma = 2$, where $\Omega = \lambda \Delta t$ and λ is the maximum eigenvalue of $M^{-1}C$. (This may be seen by noting that the eigenvectors of $M^{-1}C$ diagonalize all matrices of the form $M(M^{-1}C)^j$, where j is a positive integer. That is, if $(C - \lambda M)\psi = \mathbf{0}$, then $\psi^T M(M^{-1}C)^j \psi = \lambda^j$.) Thus there is no adverse effect on stability compared with a one-pass formulation, which amounts to the forward-difference algorithm (cf. Sec. 8.2.2).

3. A completely general stability analysis of partitioned predictor/multi-corrector algorithms is not available yet. The unpartitioned case is simpler, however, since all pertinent equations are diagonalizable. Thus one may work in terms of an SDOF model equation. The greater the number of passes, the higher the order of the stability polynomial.

4. Note the way in which the residual, or “out-of-balance,” force, (9.4.56), has been written. The term f_{n+1}^{ext} refers to the prescribed body and surface force contributions to f_{n+1}^e , but *not* the prescribed displacement, velocity, and acceleration contributions. These effects are accounted for in the remaining terms. That is, for all kinematic degrees of freedom that are prescribed, the appropriate quantities are included in the element vectors a_{n+1}^e , v_{n+1}^e , and d_{n+1}^e .

5. Kinematic boundary conditions can be specified in terms of displacement, velocity, or acceleration. Whichever quantity is given, the other two can be determined from the integration formulas which define the algorithm. For example, suppose acceleration is given; that is, assume we know q_n , \dot{q}_n , \ddot{q}_n , and \ddot{q}_{n+1} for a particular degree of freedom. Then q_{n+1} and \dot{q}_{n+1} are given by

$$q_{n+1} = q_n + \Delta t \dot{q}_n + \frac{\Delta t^2}{2} \{(1 - 2\beta)\ddot{q}_n + 2\beta\ddot{q}_{n+1}\} \quad (9.4.65)$$

$$\dot{q}_{n+1} = \dot{q}_n + \Delta t \{(1 - \gamma)\ddot{q}_n + \gamma\ddot{q}_{n+1}\} \quad (9.4.66)$$

Likewise, given any one of q_{n+1} , \dot{q}_{n+1} and \ddot{q}_{n+1} , the remaining two may be determined from (9.4.65) and (9.4.66). Quantities computed in this manner are set once and for all in the element vectors d_{n+1}^e , v_{n+1}^e , and a_{n+1}^e throughout the iterative phase during a time step (see Chapter 11).

6. Note that by ignoring contributions to K , the predictor/multicorrector schemes are applicable to the parabolic case considered in Chapter 8. Here C must be taken to the conductivity matrix.

7. An implicit-explicit predictor/multicorrector version of the Hilber-Hughes-Taylor α -method is implemented in program DLEARN (see Chapter 11). The reader may wish to study the details of this implementation in order fully to appreciate the data structure and treatment of initial and boundary conditions in transient analysis.

Exercise 3. Consider an algorithm governed by

$$X_{n+1} = AX_n$$

where A , the amplification matrix, is spectrally stable. Let us assume the eigenvalues of A are distinct so that A admits the representation

$$A = P \Lambda P^{-1}$$

where P is a matrix of eigenvectors and Λ is a diagonal matrix of eigenvalues.

Prove that if $\rho(A) \leq 1$, then $\|X_{n+1}\|_N \leq \|X_n\|_N$, where $\|X\|_N = (X^T N X)^{1/2}$, N is the symmetric real part of $(P^{-1})^* P^{-1}$, and $*$ denotes Hermitian transpose.

Solution

$$P^{-1} X_{n+1} = \Lambda P^{-1} X_n$$

and

$$X_{n+1}^T (P^{-1})^* P^{-1} X_{n+1} = X_n^T (P^{-1})^* \Lambda^* \Lambda P^{-1} X_n$$

where

$$\Lambda^* \Lambda = |\Lambda|^2$$

which implies

$$\|X_{n+1}\|_N \leq \|X_n\|_N$$

$$\|X\|_N^2 = X^T (P^{-1})^* P^{-1} X$$

$(P^{-1})^* P^{-1}$ is a Hermitian matrix. That is, it consists of a symmetric, real, positive definite matrix, plus imaginary, skew-symmetric matrix (which can be ignored in the norm).

Remark

The above result indicates that any spectrally stable algorithm possesses a conservation law, or growth inequality, in terms of a norm defined by the eigenvectors P , and thus a spectrally stable algorithm is stable with respect to the norm defined by P .

9.5 MASS MATRICES FOR SHELL ELEMENTS

In Chapter 6 we discussed a class of C^0 shell elements. Consistent- and lumped-mass matrices for these elements are presented next. The lumped matrix has scaled rota-

tional degrees of freedom to achieve effective critical time steps in explicit analysis. These ideas generalize those for the linear beam presented in Example 3 of Sec. 9.1.

Consistent Mass

The element consistent-mass matrix is defined by

$$\mathbf{m} = [\mathbf{m}_{ab}]$$

$$\mathbf{m}_{ab} = \int_{\square} \int_{-1}^{+1} N_a^T N_b \rho j \, d\xi \, d\square \quad (9.5.1)$$

where $1 \leq a, b \leq n_{en}$, the number of element nodes; ρ is the mass density; j is the Jacobian determinant; and

$$\int_{\square} \cdots \, d\square = \begin{cases} \int_{-1}^{+1} \int_{-1}^{+1} \cdots \, d\xi \, d\eta & \text{(three dimensions)} \\ \int_{-1}^{+1} \cdots \, d\eta & \text{(two dimensions)} \end{cases} \quad (9.5.2)$$

In (9.5.1), N_a is the shape function matrix (see (6.2.73) and Chapter 6 for further details).

Lumped Mass

Element lumped-mass matrices may be computed by performing the following steps

1. Calculate

$$\bar{j} = \int_{-1}^{+1} j \, d\xi \quad (9.5.3)$$

$$m_a^{\text{rot}} = \int_{\square} N_a^2 \rho \bar{j} \, d\square \quad (9.5.4)$$

(ρ is assumed to be independent of ξ .)

2. For elements other than *heterosis*:

$$m_a^{\text{disp}} = m_a^{\text{rot}}, \quad a = 1, 2, \dots, n_{en} \quad (9.5.5)$$

For *heterosis*:

$$m_a^{\text{disp}} = \int_{\square} (N_a + N_a^{\text{scr}}(0, 0)N_9)^2 \rho \bar{j} \, d\square \quad a = 1, 2, \dots, 8 \quad (9.5.6)$$

$$m_a^{\text{disp}} = 0$$

(N_a^{ser} is the eight-node serendipity shape function.) (9.5.7)

3.

$$V = \int_{\square} \bar{j} d\square, \quad M = \int_{\square} \rho \bar{j} d\square \quad (9.5.8)$$

4. Normalization

$$\tilde{M}^{\text{rot}} = \sum_{a=1}^{n_{en}} m_a^{\text{rot}} \quad (9.5.9)$$

$$\tilde{M}^{\text{disp}} = \sum_{a=1}^{n_{en}} m_a^{\text{disp}} \quad (9.5.10)$$

$$m_a^{\text{rot}} \leftarrow (M / \tilde{M}^{\text{rot}}) m_a^{\text{rot}} \quad (9.5.11)$$

$$m_a^{\text{disp}} \leftarrow (M / \tilde{M}^{\text{disp}}) m_a^{\text{disp}} \quad (9.5.12)$$

5. Adjustment to rotational inertia:

$$\langle z_a \rangle = \frac{z_a^+ + z_a^-}{2} \quad (9.5.13)$$

$$[z_a] := z_a^+ - z_a^- \quad (9.5.14)$$

$$\alpha_a = \int_{-1}^{+1} z_a^2 d\zeta / 2 = \langle z_a \rangle^2 + \frac{1}{12} [z_a]^2 \quad (9.5.15)$$

$$h = \frac{1}{n_{en}} \sum_{a=1}^{n_{en}} [z_a], \quad A = V/h \quad (9.5.16)$$

$$\alpha_a = \max \left\{ \alpha_a, \frac{A}{8} \right\} \quad (9.5.17)$$

$$m_a^{\text{rot}} \leftarrow m_a^{\text{rot}} \alpha_a \quad (\text{no sum}) \quad (9.5.18)$$

Remarks

1. In the axisymmetric case, j needs to be replaced throughout by rj , where r is the radial coordinate.
2. The mass matrices may be specialized for application to the plate elements of Chapter 5.

REFERENCES**Section 9.1**

1. N. M. Newmark, "A Method of Computation for Structural Dynamics," *Journal of the Engineering Mechanics Division, ASCE*, (1959), 67–94.
2. G. L. Goudreau and R. L. Taylor, "Evaluation of Numerical Integration Methods in Elastodynamics," *Computer Methods in Applied Mechanics and Engineering*, 2 (1972), 69–97.
3. R. D. Krieg and S. W. Key, "Transient Shell Response by Numerical Time Integration," *International Journal for Numerical Methods in Engineering*, 7 (1973), 273–286.
4. H. M. Hilber, "Analysis and Design of Numerical Integration Methods in Structural Dynamics," EERC Report No. 77-29, Earthquake Engineering Research Center, University of California, Berkeley, California, November 1976.
5. T. J. R. Hughes, "Analysis of Transient Algorithms with Particular Reference to Stability Behavior," pp. 67–155 in *Computational Methods for Transient Analysis*, eds. T. Belytschko and T. J. R. Hughes. Amsterdam: North-Holland, 1983.
6. H. Kolsky, *Stress Waves in Solids*. New York: Dover, 1963.
7. T. J. R. Hughes, R. L. Taylor, J. L. Sackman, A. Curnier, and W. Kanoknukulchai, "A Finite Element Method for a Class of Contact-Impact Problems," *Computer Methods in Applied Mechanics and Engineering*, 8 (1976), 249–276.

Section 9.2

8. I. Fried and D. S. Malkus, "Finite Element Mass Matrix Lumping by Numerical Integration Without Convergence Rate Loss," *International Journal of Solids and Structures*, 11 (1976), 461–466.
9. T. J. R. Hughes, R. L. Taylor, and W. Kanoknukulchai, "A Simple and Efficient Element for Plate Bending," *International Journal for Numerical Methods in Engineering*, 11, no. 10 (1977), 1529–1543.
10. T. J. R. Hughes, M. Cohen, and M. Haroun, "Reduced and Selective Integration Techniques in the Finite Element Analysis of Plates," *Nuclear Engineering and Design*, 46, no. 1 (1978), 203–222.
11. S. W. Key and Z. E. Beisinger, "The Transient Dynamic Analysis of Thin Shells by the Finite Element Method," *Proceedings of the 3rd Conference on Matrix Methods in Structural Mechanics*, Wright-Patterson Air Force Base, Ohio, 1971.
12. T. J. R. Hughes, W. K. Liu, and I. Levit, "Nonlinear Dynamic Finite Element Analysis," pp. 151–168 in *Nonlinear Finite Element Analysis in Structural Mechanics*, eds. W. Wunderlich et al. Berlin: Springer-Verlag, 1981. (Proceedings of the Europe-U.S. Workshop, Ruhr-Universität, Bochum, Germany, July 28–31, 1980.)
13. D. P. Flanagan and T. Belytschko, "A Uniform Strain Hexahedron and Quadrilateral with Orthogonal Hourglass Control," *International Journal for Numerical Methods in Engineering*, 17 (1981), 679–706.
14. T. Belytschko, "An Overview of Semidiscretization and Time Integration Procedures," pp. 1–65 in *Computational Methods for Transient Analysis*, eds. T. Belytschko and T. J. R. Hughes. Amsterdam: North-Holland, 1983.

10

Solution Techniques for Eigenvalue Problems

10.1 THE GENERALIZED EIGENPROBLEM

In this chapter we are concerned with algorithms for solving the so-called generalized eigenproblem arising from finite element discretizations. Recall that this takes the form:¹

$$(\mathbf{K} - \lambda \mathbf{M})\boldsymbol{\psi} = \mathbf{0} \quad (10.1.1)$$

where \mathbf{K} is symmetric and positive-semidefinite, \mathbf{M} is symmetric and positive-definite, and both matrices possess typical band/profile structure. For convenience we recall some basic properties of the solution of (10.1.1). We assume as always that the dimensions of \mathbf{K} and \mathbf{M} are $n_{eq} \times n_{eq}$. There exist n_{eq} eigenvalues and corresponding eigenvectors² that satisfy (10.1.1). That is,

$$(\mathbf{K} - \lambda_l \mathbf{M})\boldsymbol{\psi}_l = \mathbf{0}, \quad (\text{no sum}) \quad (10.1.2)$$

where $l = 1, 2, \dots, n_{eq}$ denotes the mode number. Furthermore,

$$0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{n_{eq}} \quad (10.1.3)$$

¹ Problems of this type are motivated and formulated in Chapter 7. To simplify subsequent writing we omit the superscript h on λ .

² The eigenvectors may be nonunique.

and

$$\Psi_k^T M \Psi_l = \delta_{kl} \quad (M \text{ orthonormality}) \quad (10.1.4)$$

$$\Psi_k^T K \Psi_l = \lambda_l \delta_{kl} \quad (\text{no sum}) \quad (K \text{ orthogonality}) \quad (10.1.5)$$

The eigenvectors $\{\Psi_l\}_{l=1}^{n_{eq}}$ constitute a *basis* for $\mathbb{R}^{n_{eq}}$. If, additionally, K is positive definite, as is often the case, then (10.1.3) may be strengthened as follows:

$$0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{n_{eq}} \quad (10.1.6)$$

In practice, we are typically interested only in the lower modes. These are the most important from a physical standpoint. For example, the lowest mode is the most important in a structural stability (i.e., “buckling”) calculation and the lower frequencies and corresponding mode shapes are usually the most important in considerations of dynamic response. Additionally, the higher modes of finite element discretizations are not accurate renditions of physical behavior but rather spurious artifacts of the discretization process. Consequently, they are of no engineering interest. For these reasons, we are interested in economical computational algorithms for extracting $\{\lambda_l, \Psi_l\}$, $1 \leq l \leq n_{\text{modes}}$, where $n_{\text{modes}} \ll n_{eq}$ is the number of desired eigenpairs. In practice n_{eq} may be very large, and thus solution of the eigenvalue problem, even for only a few eigenpairs, may entail an extensive and costly calculation.

Most procedures used for solving the large-scale generalized eigenproblem (10.1.1), involve a *reduced system* of the form

$$(K^* - \lambda^* M^*) \Psi^* = 0 \quad (10.1.7)$$

where K^* and M^* are *small, full, symmetric matrices*. Algorithms, such as the *generalized Jacobi method* [1], are available for directly solving (10.1.7).

Systems such as (10.1.7) may also be solved by first transforming to *standard form*:

$$(\bar{K}^* - \lambda^* I) \bar{\Psi}^* = 0 \quad (10.1.8)$$

where

$$\bar{K}^* = \bar{U}^{-T} K^* \bar{U}^{-1} \quad (10.1.9)$$

$$\bar{\Psi}^* = \bar{U} \Psi^* \quad (10.1.10)$$

and \bar{U} is the upper-triangular Cholesky factor of M^* , i.e.,

$$M^* = \bar{U}^T \bar{U} \quad (10.1.11)$$

(The Cholesky factor is related to the Crout factor by $\bar{U} = D^{1/2}U$, where $M^* = U^T D U$, U is upper-triangular with ones on the diagonal and D is a diagonal matrix of positive numbers; see Chapter 11 for further information on the Crout factorization.)

Observe that the eigenvalues of the standard form, (10.1.8), are identical to the generalized form, (10.1.7). However, the eigenvectors need to be transformed as indicated by (10.1.10)

Exercise 1. Derive (10.1.7) from (10.1.8) through (10.1.11).

There are many classical and widely available procedures for solving (10.1.8). For example, the *Jacobi*, *Givens*, and *Householder-QR methods* may be mentioned in this regard (e.g., see Bathe [1] and Noble [2]). For general matrices, it is currently felt that the most efficient strategy for solving the generalized eigenproblem, (10.1.7), is to first transform to standard form, (10.1.8), and then use the Householder-QR algorithm. However, under certain circumstances, such as when the subspace iteration procedure is employed (see Sec. 10.5), for example, direct use of the generalized Jacobi method proves very effective. It is very difficult to make sweeping statements about efficiency because the type of computer (e.g., sequential, vector, or parallel) strongly influences the performance of algorithms.

Exercise 2. Consider the undamped equation of motion,

$$M\ddot{d} + Kd = F \quad (10.1.12)$$

subject to zero initial displacement and velocity. Expand the solution in terms of the eigenvectors of the associated eigenproblem. Diagonalize the system and exactly solve the individual modal equations. Show that

$$d(t) = \sum_{i=1}^{n_{eq}} \left\{ \frac{1}{\omega_i} \int_0^t F_{(i)}(\tau) \sin \omega_i(t - \tau) d\tau \Psi_i \right\} \quad (10.1.13)$$

The $1/\omega_i$ factor in the expansion illustrates the diminishing influence of the higher modes. This analysis reveals why low mode response is viewed as "most important" and therefore, why, in practical calculations the summation in (10.1.13) is truncated at $n_{modes} \ll n_{eq}$.

Exercise 3. Obtain an exact solution for the static problem,

$$Kd = F \quad (10.1.14)$$

by way of an eigenvector expansion. Discuss the relative importance of low and high modes for this case.

Exercise 4. Generalize Exercise 2 to account for Rayleigh damping and non-zero initial conditions. Discuss the influence of low and high modes.

10.2 STATIC CONDENSATION

In practice one often encounters eigenvalue problems which can be written in the following *partitioned form*:

$$\left(\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} - \lambda \begin{bmatrix} M_{11} & 0 \\ 0 & 0 \end{bmatrix} \right) \begin{Bmatrix} \psi_1 \\ \psi_2 \end{Bmatrix} = 0 \quad (10.2.1)$$

where M_{11} is symmetric and positive-definite. That is, many degrees of freedom are “massless.” Problems of this type arise naturally when a relatively light structure is used to support heavy nonstructural masses which can be “lumped” at a few degrees of freedom. In such situations the structural mass is often insignificant and may be neglected, resulting in a system like (10.2.1). As it stands, (10.2.1) is ill-posed in the sense that the zero-diagonal masses give rise to infinite eigenvalues. The mass matrix can be “regularized” by the addition of small positive nonzero diagonal masses, in which case we return to the format of the generalized eigenvalue problem originally considered, or, alternatively, the zero-mass degrees of freedom can be eliminated by *static condensation*. To arrive at the statically condensed form, we expand (10.2.1):

$$K_{11}\psi_1 + K_{12}\psi_2 - \lambda M_{11}\psi_1 = 0 \quad (10.2.2)$$

$$K_{21}\psi_1 + K_{22}\psi_2 = 0 \quad (10.2.3)$$

The next step is to solve (10.2.3) for ψ_2 and then substitute in (10.2.2), which results in

$$(K_{11}^* - \lambda M_{11})\psi_1 = 0 \quad (10.2.4)$$

where

$$K_{11}^* = K_{11} - K_{12}K_{22}^{-1}K_{21} \quad (\text{statically condensed stiffness}) \quad (10.2.5)$$

The advantage of transforming to statically condensed form is that the problem size is reduced. However, K_{11}^* tends to be full. Thus, unless the size of the nonzero-mass matrix is rather small, the reduction to statically condensed form may be uneconomical because the profile structure of K is lost.

Note that to calculate the statically condensed stiffness (10.2.5) efficiently, K_{22} is never actually inverted.

The following steps may be used in practice:

$$K_{22} = U^T D U \quad (\text{Crout factorization}) \quad (10.2.6)$$

$$\mathbf{U} \mathbf{D}^{1/2} \mathbf{Z} = \mathbf{K}_{21} \quad (\text{solve for } \mathbf{Z}) \quad (10.2.7)$$

$$\mathbf{K}_{12} \mathbf{K}_{22}^{-1} \mathbf{K}_{21} = \mathbf{Z}^T \mathbf{Z} \quad (10.2.8)$$

Equation (10.2.7) amounts to solution of an equation system with upper triangular coefficient array and multiple right-hand sides (i.e., the columns of \mathbf{K}_{21}).

For further computational considerations regarding static condensation, see [3].

10.3 DISCRETE RAYLEIGH-RITZ REDUCTION

In the discrete Rayleigh-Ritz approach, *static load patterns*, \mathbf{P} , are selected and corresponding displacement vectors, \mathbf{R} , are calculated from:

$$\underbrace{\mathbf{K}}_{n_{eq} \times n_{eq}} \quad \underbrace{\mathbf{R}}_{n_{eq} \times n_{lp}} = \underbrace{\mathbf{P}}_{n_{eq} \times n_{lp}}, \quad n_{lp} < n_{eq} \quad (10.3.1)$$

where n_{lp} refers to the number of load patterns. The displacements \mathbf{R} , referred to as the *trial vectors*, are used to form the reduced eigenproblem, i.e., (10.1.7) by defining

$$\mathbf{K}^* = \mathbf{R}^T \mathbf{K} \mathbf{R} \quad (10.3.2)$$

$$\mathbf{M}^* = \mathbf{R}^T \mathbf{M} \mathbf{R} \quad (10.3.3)$$

The load patterns, \mathbf{P} , are usually subject to the following criteria:

- i. The columns of \mathbf{P} should be linearly independent.
- ii. The columns of \mathbf{P} should be selected to arouse the low modes by activating the heaviest masses and most flexible areas of the model.

Remarks

1. If n_{lp} is small, \mathbf{K}^* and \mathbf{M}^* will be full, but small.
2. Equation (10.3.1) amounts to solution of a multiple right-hand-side system with profile coefficient matrix.
3. The discrete Rayleigh-Ritz procedure is a general formalism for obtaining the reduced system. However, the guidelines for selecting \mathbf{P} are somewhat vague and thus a more systematic strategy is required for practical use.
4. The discrete Rayleigh-Ritz reduction is often referred to as a “projection method.”
5. The eigenvector approximations are defined by $\psi \cong \mathbf{R} \psi^*$.
6. By virtue of the fact that the calculation of trial vectors from load patterns requires solution of (10.3.1), \mathbf{K} must be nonsingular. This precludes application to cases in which there are zero eigenvalues (e.g., structures which possess rigid body modes). A simple reformulation of the original problem involving a positive *shifting* of the eigenvalues allows us to handle this case: Note that by adding and subtracting $\alpha \mathbf{M} \psi$, where α is a positive number, we produce an eigenproblem

$$(\bar{\mathbf{K}} - \bar{\lambda} \mathbf{M}) \boldsymbol{\psi} = \mathbf{0} \quad (10.3.1)$$

$$\bar{\mathbf{K}} = \mathbf{K} + \alpha \mathbf{M} \quad (10.3.2)$$

$$\bar{\lambda} = \lambda + \alpha \quad (10.3.3)$$

in which $\bar{\mathbf{K}}$ is positive-definite, and the eigenvectors are unchanged. The eigenvalues are related by (10.3.6).

This stratagem may also be employed if some of the λ 's are negative. Suppose an estimate of the smallest eigenvalue, λ_1 , is available. Then select $\alpha > 0$ such that $-\alpha < \lambda_1 \leq 0$. The $\bar{\lambda}$'s are then all positive and solution may proceed as usual.

7. Shifting is a particular example of a general class of *spectral transformations*. Let

$$\mathbf{S} = [\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \dots, \boldsymbol{\psi}_{n_{eq}}] \quad (10.3.4)$$

and

$$\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_{n_{eq}}) \quad (10.3.5)$$

Then it is easily verified that

$$\mathbf{K} = \mathbf{S} \mathbf{\Lambda} \mathbf{S}^T \quad (10.3.6)$$

and

$$\mathbf{M} = \mathbf{S} \mathbf{S}^T \quad (10.3.7)$$

Let $f = f(\lambda)$ represent a scalar-valued function of λ . Define a matrix-valued function $f = f(\mathbf{K})$ by way of

$$f(\mathbf{K}) = \mathbf{S} \text{diag}(f(\lambda_1), f(\lambda_2), \dots, f(\lambda_{n_{eq}})) \mathbf{S}^T \quad (10.3.8)$$

Then it is readily established that solutions of the eigenproblem

$$(f(\mathbf{K}) - \bar{\lambda} \mathbf{M}) \boldsymbol{\phi} = \mathbf{0} \quad (10.3.9)$$

are related to solutions of

$$(\mathbf{K} - \lambda \mathbf{M}) \boldsymbol{\psi} = \mathbf{0} \quad (10.3.10)$$

by

$$\bar{\lambda} = f(\lambda) \quad (10.3.11)$$

$$\boldsymbol{\phi} = \boldsymbol{\psi} \quad (10.3.12)$$

Exercise 1. Consider the partitioned eigenproblem defined by (10.2.1). Show that the statically condensed eigenproblem, (10.2.4) and (10.2.5), can be obtained from the discrete Rayleigh-Ritz approach by selecting the trial vectors to be

$$\mathbf{R} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{K}_{22}^{-1} \mathbf{K}_{21} \end{bmatrix} \quad (10.3.13)$$

10.4 IRONS-GUYAN REDUCTION

The Irons-Guyan reduction [4–6] fits into the discrete Rayleigh-Ritz framework described in the previous section. To understand the method it is helpful to rewrite the generalized eigenproblem in partitioned form:

$$\left(\begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} - \lambda \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \right) \begin{Bmatrix} \boldsymbol{\psi}_1 \\ \boldsymbol{\psi}_2 \end{Bmatrix} = \mathbf{0} \quad (10.4.1)$$

The degrees of freedom in $\boldsymbol{\psi}_1$ are to be retained in the reduced eigenproblem, whereas those in $\boldsymbol{\psi}_2$ are to be eliminated. For these purposes \mathbf{R} is defined by (10.3.16), which leads to the definitions of the reduced arrays (verify!):

$$\mathbf{K}^* = \mathbf{R}^T \mathbf{K} \mathbf{R} = \mathbf{K}_{11} - \mathbf{K}_{12} \mathbf{K}_{22}^{-1} \mathbf{K}_{21} \quad (10.4.2)$$

$$\mathbf{M}^* = \mathbf{R}^T \mathbf{M} \mathbf{R} = \mathbf{M}_{11} - \mathbf{M}_{12} \mathbf{K}_{22}^{-1} \mathbf{K}_{21} - \mathbf{K}_{12} \mathbf{K}_{22}^{-1} (\mathbf{M}_{21} - \mathbf{M}_{22} \mathbf{K}_{22}^{-1} \mathbf{K}_{21}) \quad (10.4.3)$$

Note that (10.4.2) is the statically condensed stiffness and, if $\mathbf{M}_{22} = \mathbf{0}$ and $\mathbf{M}_{12} = \mathbf{M}_{21}^T = \mathbf{0}$, then $\mathbf{M}^* = \mathbf{M}_{11}$ as in the statically condensed problem of Sec. 10.2.

The Irons-Guyan reduction may be thought of as invoking a “static relation,” namely

$$\boldsymbol{\psi}_2 = -\mathbf{K}_{22}^{-1} \mathbf{K}_{21} \boldsymbol{\psi}_1 \quad (10.4.4)$$

to eliminate the unwanted variables. Sometimes $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$ are referred to as the “master” and “slave” degrees of freedom, respectively. In practice, the following guidelines are suggested:

- i. Determine the number of eigenvalues and eigenvectors required, say n_{modes} .
- ii. Retain a multiple of n_{modes} degrees of freedom (e.g., $5 \times n_{\text{modes}}$ degrees of freedom).
- iii. The degrees of freedom to be retained are identified by calculating the ratios of diagonal elements in \mathbf{M} and \mathbf{K} , namely, K_{pp}/M_{pp} , $1 \leq p \leq n_{eq}$. The degrees of freedom with the smallest ratios are retained (i.e., they are the degrees of freedom in $\boldsymbol{\psi}_1$; [7]).

Remark

The calculation of the reduced mass, (10.4.3), is costly!

10.5 SUBSPACE ITERATION

A disadvantage of reduction techniques such as the Irons-Guyan procedure is that there is no guarantee that the eigenvalues and eigenvectors of the reduced problem,

namely, λ^* and $R\psi^*$, will be good approximations of those of the original problem, λ , and ψ_i , respectively. Consequently, methods have been developed in which a reduced problem is used along with an iterative strategy to obtain exactly the lower modes of the generalized eigenproblem. This is the underlying idea of the *subspace iteration*, or *block power, method*, which is widely used for large-scale finite element calculations. Roughly speaking, the procedure is as follows: Load patterns are selected and trial vectors are calculated. The trial vectors are used to form a reduced problem, which is solved. New load patterns are calculated from the “inertial” forces engendered by the eigenvectors of the reduced problem, namely,

$$\underbrace{P}_{n_{eq} \times n_{lp}} = MR[\psi_1^*, \psi_2^*, \dots, \psi_{n_{lp}}^*] \quad (10.5.1)$$

With these load patterns the process is repeated until convergence is achieved. The calculations are summarized in the flowchart contained in Table 10.5.1. Extensive discussion and further details are presented in [8].

TABLE 10.5.1 Subspace Iteration Procedure

| | |
|-----|--|
| I. | Initialization |
| 1. | Form K and M . |
| 2. | Factorize $K = U^T D U$. |
| 3. | Specify initial load patterns, P . |
| II. | Iteration |
| 1. | Solve for trial vectors: |
| | $(U^T D U)R = P$ |
| 2. | Compute reduced matrices: |
| | $K^* = R^T K R = R^T P$ |
| | $M^* = R^T M R$ |
| 3. | Solve the reduced eigenproblem: |
| | $(K^* - \lambda_k^* M^*)\psi_k^* = 0, \quad k = 1, 2, \dots, n_{lp}$ |
| 4. | Calculate approximations to the eigenvectors of the original system: |
| | $R[\psi_1^*, \psi_2^*, \dots, \psi_{n_{lp}}^*]$ |
| 5. | Perform convergence checks. If the desired eigenvalues have converged, stop. Otherwise continue. |
| 6. | Calculate improved load patterns: |
| | $P = MR[\psi_1^*, \psi_2^*, \dots, \psi_{n_{lp}}^*]$ |
| 7. | Go to Step II-1. |

Remarks

1. It is recommended from practical experience [8] that:

- The number of load patterns employed should be calculated from

$n_{lp} = \min \{2 n_{\text{modes}}, n_{\text{modes}} + 8\}$, where n_{modes} is the number of eigenpairs desired.

- ii. In the first load pattern (i.e., first column of \mathbf{P}) a 1 should be placed in the entry corresponding to the minimum value of K_{pp}/M_{pp} , and 0 in the remaining entries. Likewise, a 1 should be placed in the entry of the second column corresponding to the next smallest value of K_{pp}/M_{pp} , and so forth.
- 2. Wilson [9] recommends that one load pattern be generated from random numbers in the interval $[0, 1]$.

3. The convergence condition may be specified in terms of consecutive iterates:

$$\frac{(\lambda_k^*)_I - (\lambda_k^*)_I}{(\lambda_k^*)_I} \leq \epsilon, \quad k = 1, 2, \dots, n_{\text{modes}}$$

where I is the iteration number and ϵ is a preassigned error tolerance.

4. Solution of the generalized eigenproblem in the subspace iteration algorithm (Step II-3 in Table 10.5.1) is efficiently carried out by way of the *generalized Jacobi method* [8]. The reason for this is that after a number of iteration steps, the generalized eigenproblem possesses diagonally dominant matrices for which the generalized Jacobi method proves very effective.

10.5.1 Spectrum Slicing

The convergence of a procedure such as subspace iteration does not necessarily guarantee that the first n_{modes} eigenpairs of the original system have been found. An eigenvalue can be missed if the original trial vectors are orthogonal to the corresponding eigenvector. To ascertain whether or not this has occurred, a *spectrum slicing* may be performed.³ The steps involved are as follows: Perform a Crout factorization of the matrix $\mathbf{K} - \alpha \mathbf{M}$, where $\alpha = (1 + \delta)\lambda_{n_{\text{modes}}}$ and δ is a small positive number such that $(1 + \delta)\lambda_{n_{\text{modes}}} < \lambda_{n_{\text{modes}}+1}$. Let \mathbf{D}_α denote the diagonal matrix of pivots obtained. It follows from *Sylvester's inertia theorem* that the number of eigenvalues smaller than α will equal the number of negative entries in \mathbf{D}_α . If this number, say n_α , is greater than n_{modes} , then $n_\alpha - n_{\text{modes}}$ eigenvalues smaller than α were missed in the calculation. If this is the case, a revised set of load patterns must be employed and the eigensolution repeated. If $n_\alpha = n_{\text{modes}}$, then a degree of confidence in the solution is attained.

Remark

Spectrum slicing can be used to determine the number of eigenvalues in an interval $\left]\alpha, \beta\right[$, where $\beta > \alpha$. Factorize $\mathbf{K} - \alpha \mathbf{M}$ and $\mathbf{K} - \beta \mathbf{M}$. If \mathbf{D}_α and \mathbf{D}_β are the corresponding pivots and n_α and n_β are the number of negative entries of \mathbf{D}_α and \mathbf{D}_β , respectively, then $n_\beta - n_\alpha$ is the number of eigenvalues in $\left]\alpha, \beta\right[$.

³ Spectrum slicing is sometimes referred to as a *Sturm sequence check* [8].

10.5.2 Inverse Iteration

When the number of load patterns used in the subspace iteration procedure is one, the process is called *inverse iteration*. In this case convergence to the lowest eigenvalue occurs as long as the initial trial vector is not orthogonal to the corresponding eigenvector.

By reversing the roles of K and M , convergence to the largest eigenvalue, $\lambda_{n_{eq}}$, can be achieved. This may be seen from:

$$\mathbf{0} = (K - \lambda M)\psi = (M - \lambda^{-1}K)\psi$$

Note that when working with the latter form, subspace iteration requires M to be nonsingular.

Exercise 1. Consider the following eigenvalue problem:

$$(K - \lambda M)\psi = \mathbf{0}, \quad \lambda = \omega^2$$

where

$$\begin{aligned} M &= \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \\ K &= \begin{bmatrix} (k_1 + k_2) & -k_2 \\ -k_2 & k_2 \end{bmatrix} \\ \psi &= \begin{Bmatrix} \psi_1 \\ \psi_2 \end{Bmatrix} \end{aligned}$$

Assume $k_1 = k_2 = 1$, $m_1 = 3$, $m_2 = 2$.

- Calculate the frequencies (i.e., ω_1 , ω_2) and mode shapes (i.e., ψ_1 , ψ_2).
- Assuming that the fundamental mode load pattern is given approximately by

$$P = \left\{ \begin{array}{c} \frac{m_1}{(k_1 + k_2)} \\ \frac{m_2}{k_2} \end{array} \right\}$$

use the discrete Rayleigh-Ritz reduction procedure to obtain an estimate of the fundamental frequency and mode shape.

- Use the Irons-Guyan procedure to reduce the problem to one degree of freedom. Pick the degree of freedom to be retained according to the criterion presented in Sec. 10.4. Determine the approximate fundamental frequency and mode shape.
- Use the subspace iteration procedure to calculate the fundamental frequency and mode shape. Initialize the computations with the load pattern

$$P = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$$

Employ two iterations.

Solution

a.

$$\mathbf{M} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \quad \mathbf{K} = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$$

$$0 = \det \begin{bmatrix} 2 - 3\lambda & -1 \\ -1 & 1 - 2\lambda \end{bmatrix} = 6\lambda^2 - 7\lambda + 1$$

$$\lambda_{1,2} = \frac{1}{6}, 1$$

$$\boxed{\omega_{1,2} = \frac{1}{\sqrt{6}}, 1 = 0.40825, 1}$$

$$\Psi_1 = \begin{Bmatrix} 2 \\ 3 \\ 1 \end{Bmatrix} \quad \Psi_2 = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}$$

b.

$$\mathbf{P} = \begin{Bmatrix} 3 \\ 2 \\ 2 \end{Bmatrix}$$

$$\mathbf{K}\mathbf{R} = \mathbf{P}$$

$$\mathbf{R} = \begin{Bmatrix} \frac{7}{2} \\ \frac{2}{2} \\ \frac{11}{2} \end{Bmatrix} \quad (\text{the } \frac{1}{2} \text{ factors may be neglected})$$

$$\mathbf{K}^* = \mathbf{R}^T \mathbf{K} \mathbf{R} = \langle 7 \ 11 \rangle \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} 7 \\ 11 \end{Bmatrix} = 65$$

$$\mathbf{M}^* = \mathbf{R}^T \mathbf{M} \mathbf{R} = \langle 7 \ 11 \rangle \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{Bmatrix} 7 \\ 11 \end{Bmatrix} = 389$$

$$(\mathbf{K}^* - \lambda^* \mathbf{M}^*) \Psi^* = \mathbf{0} \Rightarrow \lambda^* = \frac{65}{389}, \quad \Psi^* = 1$$

$$\boxed{\Psi_1 \cong \mathbf{R} \Psi_1^* = \frac{1}{11} \begin{Bmatrix} 7 \\ 11 \end{Bmatrix} = \begin{Bmatrix} 0.64 \\ 1 \end{Bmatrix}; \quad \omega^* = \left(\frac{65}{389} \right)^{1/2} = 0.4087}$$

c.

$$\left. \begin{array}{l} \frac{K_{11}}{M_{11}} = \frac{2}{3} \\ \frac{K_{22}}{M_{22}} = \frac{1}{2} \end{array} \right\} \Rightarrow \text{retain degree of freedom number 2}$$

Reorder equations into the standard partitioned form.

$$\mathbf{K} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$

$$\mathbf{R} = \left\{ \begin{array}{l} 1 \\ -K_{22}^{-1} K_{21} \end{array} \right\} = \left\{ \begin{array}{l} 1 \\ \frac{1}{2} \end{array} \right\}$$

$$\mathbf{K}^* = \mathbf{R}^T \mathbf{K} \mathbf{R} = \frac{1}{2}$$

$$\mathbf{M}^* = \mathbf{R}^T \mathbf{M} \mathbf{R} = \frac{11}{4}$$

$$\lambda^* = \frac{2}{11}, \quad \boxed{\omega^* = \left(\frac{2}{11} \right)^{1/2} = 0.4264}, \quad \Psi_1^* = 1$$

$$\Psi_1 \cong \mathbf{R} \Psi_1^* = \left\{ \begin{array}{l} 1 \\ \frac{1}{2} \end{array} \right\}, \quad (\text{reorder}): \quad \boxed{\Psi_1 = \left\{ \begin{array}{l} 1 \\ \frac{1}{2} \\ 1 \end{array} \right\}}$$

$$\left. \begin{aligned} \text{d. } \mathbf{P} &= \left\{ \begin{array}{l} 0 \\ 1 \end{array} \right\}; & \mathbf{K} \mathbf{R} &= \mathbf{P}; & \mathbf{R} &= \left\{ \begin{array}{l} 1 \\ 2 \end{array} \right\} \\ \mathbf{K}^* &= 2; & \mathbf{M}^* &= 11; & \lambda^* &= \frac{2}{11} \quad (\text{same as (c)}) \end{aligned} \right\} \text{ iteration number 1}$$

$$\omega^* = 0.4264 \quad \Psi_1 \cong \left\{ \begin{array}{l} 1 \\ \frac{1}{2} \\ 1 \end{array} \right\}$$

$$\left. \begin{aligned} \mathbf{P} &= \mathbf{M} \Psi_1 = \left\{ \begin{array}{l} \frac{3}{2} \\ 2 \\ 2 \end{array} \right\} \quad (\text{same as (b)}) \\ \text{Therefore} \\ \omega^* &= 0.4087 \quad \Psi_1 \cong \left\{ \begin{array}{l} 0.64 \\ 1 \end{array} \right\} \end{aligned} \right\} \text{ iteration number 2}$$

THE LANCZOS ALGORITHM FOR SOLUTION OF LARGE GENERALIZED EIGENPROBLEMS

BAHRAM NOUR-OMID

10.6.1 Introduction

The Lanczos algorithm was first proposed in 1950. Lanczos intended his algorithm to be used for computing a few of the extreme eigenvalues and corresponding eigenvectors of a symmetric matrix. However, the algorithm was taken up as a method for reducing a symmetric matrix to tridiagonal form. Lanczos himself observed that round-off error has a significant effect on the algorithm's performance and expensive modifications seemed to be necessary to overcome this defect. By 1955 the Householder method replaced the Lanczos algorithm as a more efficient method for tri-diagonalizing a matrix.

Most engineering applications require only a few eigenvalues at one end of the spectrum, but for small problems the Householder-QR algorithm can find all the eigenvalues almost as fast as a few. For large problems however it is a waste to tridiagonalize the whole matrix. The Lanczos iteration has the property that well-isolated eigenvalues are approximated accurately after a comparatively small number of steps. For example, the largest eigenvalues will often be captured after 20 steps, almost independently of the order of the problem, n . Moreover, the number of steps taken by the Lanczos method to compute these eigenvalues will always be less than that for the power method. This property makes the Lanczos algorithm very efficient. However, the eigenvalues most frequently wanted are the smaller ones. The Lanczos algorithm is so powerful that it can compute the smaller eigenvalues of a matrix without any factorization. However, they will not be well approximated until nearly n steps have been taken. Consequently it is necessary, in these cases, to apply the iteration to an inverted form of the matrix. In this respect the Lanczos algorithm is just like the subspace iteration method. See [15] for a comparison of Lanczos and subspace iteration algorithms.

In this section we are concerned with the application of the Lanczos algorithm to the generalized symmetric eigenproblem

$$(K - \lambda M)z = 0 \quad (10.6.1)$$

where K and M are $n \times n$ real symmetric matrices. We assume for the moment that the eigenvalues of (10.6.1) are real. Later we state the conditions on K and M that must hold to ensure real eigenvalues. We will derive the generalized Lanczos algorithm for the solution of the eigenproblem (10.6.1). The relation between the Lanczos method and vector iteration methods will be established. We then look at the effect of round-off on the algorithm and the resulting loss of orthogonality. We consider two possible modifications to the algorithm that can maintain a desired level of orthogonality among the Lanczos vectors. Finally, we give a detailed description of the algorithm we prefer together with a listing of the computer subprograms.

10.6.2 Spectral Transformation

We turn now to an important misconception concerning the Lanczos algorithm. It is usually thought of as a way of computing eigenpairs, (λ, z) , of the standard eigenproblem

$$(A - \lambda I)z = 0 \quad (10.6.2)$$

The Lanczos method is so powerful that one can work directly with A to evaluate eigenvalues at both ends of the spectrum of (10.6.2) without solving any system of equations. Only products of A with a sequence of vectors need be computed. This virtue blinded certain users of the Lanczos method to the great advantages to be gained by a shift-and-invert procedure. The matrix $(A - \sigma I)^{-1}$ has the same eigenvectors as A . Using $(A - \sigma I)^{-1}$ instead of A will cause eigenvectors corresponding to eigenvalues close to σ to converge much more quickly. Of course there are cases when factoring of A is not possible (e.g., when the matrix is only known implicitly) or not desirable (e.g., when A has a given sparsity structure that can be destroyed when factored).

However, we are more interested in the generalized eigenproblem (10.6.1). For this problem some form of inversion or factoring of a matrix, either explicitly or implicitly, is required. It is interesting to note that when the structure of M is the same as that of K , as in the case of a consistent mass matrix, then the cost of one step of the power iteration method is exactly the same as that for the inverse iteration method. Each requires one triangular factorization.

There are two different procedures for transforming the generalized eigenproblem to the standard form:

- i. Factor M into $M = CC^T$. This is the Cholesky factorization of M . Then

$$(K - \lambda M) = C(C^{-1}KC^{-T} - \lambda I)C^T \quad (10.6.3)$$

and the eigenproblem of (10.6.1) reduces to the standard form $(A_1 - \lambda I)\hat{z} = 0$ where $A_1 = C^{-1}KC^{-T}$ and $\hat{z} = C^Tz$. The eigenvalues of the standard problem are the same as those of the generalized problem. Note that if M is positive-semidefinite, then its Cholesky factors are singular and this transformation can not be performed.

- ii. Using a similar procedure, (10.6.1) can be transformed into

$$(A_2 - \mu I)\hat{z} = 0 \quad (10.6.4)$$

where $A_2 = C^TK^{-1}C$ and $\mu = 1/\lambda$. Note that in this case both the eigenvalues and the eigenvectors of A_2 are different from those of (10.6.1).

A more general form of the second transformation is to first perform a shift of the origin, $(K_\sigma - (\lambda - \sigma)M)z = 0$, where $K_\sigma = K - \sigma M$, and then perform (ii). This results in a standard eigenproblem with $A_\sigma = C^T K_\sigma^{-1} C$. The spectrum of A_σ is related to the original spectrum through

$$\nu = \frac{1}{\lambda - \sigma}$$

(10.6.5)

where ν is the eigenvalue of A_σ [11].

The second procedure requires two triangular factorizations, one for M and one for K . It is possible to avoid the factorization of M by working with

$$(K_{\sigma}^{-1}M - \nu I)z = 0 \quad (10.6.6)$$

Although the matrix of this transformation is not symmetric, it is self-adjoint with respect to the *inertial inner product* defined by

$$(u, v)_M = v^T M u \quad (10.6.7)$$

which is shown in the following steps:

$$\begin{aligned} (K_{\sigma}^{-1}M u, v)_M &= v^T M K_{\sigma}^{-1} M u \\ &= (u, K_{\sigma}^{-1} M v)_M \end{aligned}$$

This unsymmetric form is particularly advantageous since it has the same eigenvectors as the original problem [16]. The algorithm that is derived later employs the transformation of equation (10.6.6).

10.6.3 Conditions for Real Eigenvalues

Contrary to common belief, the fact that K and M are symmetric is not a sufficient condition to ensure real eigenvalues for (10.6.1). This can be illustrated using the following simple example.

Example 1

Let $K = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ and $M = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. Then the eigenvalues of (10.6.1) with these matrices are $\frac{1}{2}(1 \pm i\sqrt{3})$, where $i^2 = -1$.

The eigenvalues of (10.6.1) are all real if some linear combination of K and M is positive-definite; that is $\rho K + \tau M$ is positive-definite for some choice of real ρ and τ . This is only a sufficient condition that guarantees real eigenvalues, but there is no easy way of computing ρ and τ .

Exercise 1. Prove that $\rho K + \tau M$ being positive definite for some choice of ρ and τ is only a sufficient condition for (10.6.1) to have real eigenvalues.

Solution Multiply (10.6.1) by ρ and shift the spectrum by τ :

$$\begin{aligned} 0 &= (\rho K - \rho \lambda M)z \\ &= (\rho K + \tau M - (\rho \lambda + \tau)M)z \end{aligned}$$

Because $\rho K + \tau M$ is positive definite, it possesses a Cholesky decomposition; i.e., $\rho K + \tau M = CC^T$. Thus, the above equation can be reduced to standard form:

$$(A - \mu I)\hat{z} = 0$$

where $A = C^{-1}MC^{-T}$, $\hat{z} = C^Tz$, and $\mu = 1/(\rho \lambda + \tau)$. By the symmetry of A , the μ 's

are real. Consequently, so are the λ 's. This establishes "sufficiency". Now consider the following case. Let

$$\mathbf{K} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \quad \text{and} \quad \mathbf{M} = \begin{bmatrix} -3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Clearly, the eigenvalues of this problem are real. However, there is no linear combination of \mathbf{K} and \mathbf{M} that is positive definite. This establishes that the above is not a necessary condition.

Fortunately, in many problems encountered in structural mechanics, either \mathbf{K} or \mathbf{M} or both are positive-definite.

10.6.4 The Rayleigh-Ritz Approximation

Consider a given set of vectors, $\mathbf{X}_m = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m]$, with $m \ll n$. We refer to these as *trial vectors*. We proceed to obtain an approximation to some of the eigenvectors of (10.6.1) by taking a linear combination of the trial vectors. Let $\mathbf{y} = \mathbf{X}_m \mathbf{s} = \sum_{i=1}^m \mathbf{x}_i s_i$ be the desired approximation. The i th component of \mathbf{s} is the coefficient of \mathbf{x}_i in this representation of \mathbf{y} . We denote the corresponding approximation to the eigenvalues of (10.6.1) by θ . The *residual*, \mathbf{r} , associated with the approximating pair $\{\theta, \mathbf{y}\}$ is given by

$$\boxed{\mathbf{r} = \mathbf{Ky} - \theta \mathbf{My}} \quad (10.6.8)$$

The Rayleigh-Ritz method requires the residual vector be orthogonal to each of the trial vectors; i.e.,

$$\mathbf{X}_m^T \mathbf{r} = \mathbf{X}_m^T \mathbf{Ky} - \theta \mathbf{X}_m^T \mathbf{My} = 0$$

Substituting the representation of \mathbf{y} in terms of the trial vectors, in the above orthogonality condition for \mathbf{r} , we obtain the reduced eigenproblem

$$(\mathbf{K}_m - \theta \mathbf{M}_m) \mathbf{s} = 0$$

where $\mathbf{K}_m = \mathbf{X}_m^T \mathbf{K} \mathbf{X}_m$ and $\mathbf{M}_m = \mathbf{X}_m^T \mathbf{M} \mathbf{X}_m$. The eigenvector of the reduced eigenproblem, $\mathbf{s}^{(k)}$, will determine the approximation, $\mathbf{y}^{(k)}$, to the eigenvector of (10.6.1). We refer to $\mathbf{y}^{(k)}$ as the *Ritz vector* and to its corresponding $\theta^{(k)}$ as the *Ritz value*, for each $k = 1, \dots, m$.⁴

⁴Some authors refer to the trial vectors as Ritz vectors. In keeping with more prevalent usage, we prefer to reserve the latter terminology for the \mathbf{y} 's.

10.6.5 Derivation of The Lanczos Algorithm

The Lanczos method can be thought of as a means of constructing an orthogonal set of vectors, known as Lanczos vectors, for use in the Rayleigh-Ritz procedure. The algorithm is closely related to the inverse iteration and power methods for calculating a single eigenpair. Given a pair of matrices $K_\sigma = K - \sigma M$ and M , and a starting vector r , these basic methods generate a sequence of vectors, $\{r, K_\sigma^{-1}Mr, (K_\sigma^{-1}M)^2r, \dots, (K_\sigma^{-1}M)^j r\}$, during j iterations. These vectors are referred to as the **Krylov sequence**; the sequence converges (as $j \rightarrow \infty$) to the eigenvector corresponding to the eigenvalue, λ , of (10.6.1) closest to the shift σ .

The basic difference between the Lanczos method and the other two is that the information contained in each successive vector of the Krylov sequence is used to obtain the best approximation to the wanted eigenvectors instead of using only the last vector in the sequence. To be more specific, the Lanczos algorithm is equivalent to obtaining the Rayleigh-Ritz approximation with the vectors in the Krylov sequence as the trial vectors. This involves supplementing the Krylov sequence with the Gram-Schmidt orthogonalization process at each step. The result is a set of M -orthonormal vectors (the **Lanczos vectors**) that is used in the Rayleigh-Ritz procedure to reduce the dimension of the eigenproblem. We show below that orthogonalization is required only with respect to two preceding vectors; a fact recognized by Lanczos. The Rayleigh-Ritz procedure with the M -orthonormal basis of the Krylov subspace leads to a standard eigenproblem with a tridiagonal matrix.

To derive the Lanczos algorithm it will be assumed for the moment that the first j Lanczos vectors, $\{q_1, q_2, \dots, q_j\}$ have been found, and the construction of the $j + 1$ vector will be described. The resulting vectors all satisfy the condition $q_i^T M q_j = \delta_{ij}$, where δ_{ij} is the Kronecker delta; that is, the vectors are orthonormal with respect to the mass matrix. To calculate q_{j+1} , we must orthogonalize $v_j = (K_\sigma^{-1}M)^j r$ against the j Lanczos vectors computed so far. From the definition of v_j we obtain $v_j = K_\sigma^{-1}M v_{j-1}$. Now, v_{j-1} is the vector that is M -orthonormalized against the first $j - 1$ Lanczos vectors to obtain q_j . Therefore

$$v_{j-1} = \sum_{i=1}^j v_i q_i$$

where v_i is the component of v_{j-1} along q_i . This result is used to eliminate v_{j-1} in the above recursive relation for v_j . We then get

$$\begin{aligned} v_j &= \sum_{i=1}^j v_i K_\sigma^{-1} M q_i \\ &= v_j K_\sigma^{-1} M q_j + \sum_{i=1}^{j-1} v_i K_\sigma^{-1} M q_i \end{aligned}$$

Observe that each vector, $K_\sigma^{-1} M q_i$, in the above summation can be written as a linear combination of the first $i + 1$ Lanczos vectors. Therefore the sum can be written as a linear combination of the first j Lanczos vectors. Consequently

$$v_j = v_j K_\sigma^{-1} M q_j + \sum_{i=1}^j \bar{v}_i q_i$$

The M -orthogonalization of v_j against the preceding j Lanczos vectors will purge the component of v_j along each of the q_i 's, and therefore the final result will be unaffected by the sum in the last equation. Therefore the next Lanczos vector, q_{j+1} , will be obtained by first computing a preliminary vector \bar{r}_j from the previous vector, q_j ,

$$\bar{r}_j = K_\sigma^{-1} M q_j \quad (10.6.9)$$

and M -orthonormalizing it against all the previous Lanczos vectors. Now, in general it may be assumed that this preliminary vector contains components from each of the preceding vectors. Thus

$$\bar{r}_j = r_j + \alpha_j q_j + \beta_j q_{j-1} + \gamma_j q_{j-2} + \dots \quad (10.6.10)$$

where r_j is the “pure” component of \bar{r}_j orthogonal to all previous Lanczos vectors, and $\alpha_j, \beta_j, \gamma_j, \dots$ are the amplitudes of the previous vectors contained in \bar{r}_j . These amplitude coefficients are evaluated from the orthonormality of the Lanczos vectors. Thus, if both sides of (10.6.10) are multiplied by $q_j^T M$, the result is

$$q_j^T M \bar{r}_j = q_j^T M r_j + \alpha_j q_j^T M q_j + \beta_j q_j^T M q_{j-1} + \gamma_j q_j^T M q_{j-2} + \dots \quad (10.6.11)$$

Here the first term on the right-hand side vanishes by *definition*, and all terms beyond the second vanish similarly due to M -orthogonality. The normalizing definition applied to the second term then reduces (10.6.11) to an expression for the amplitude of q_j along \bar{r}_j :

$$\boxed{\alpha_j = q_j^T M \bar{r}_j} \quad (10.6.12)$$

The amplitude of q_{j-1} contained in \bar{r}_j may be found similarly by multiplying (10.6.10) by $q_{j-1}^T M$. In this case all terms except the third vanish by orthogonality, and the coefficient of β_j is unity, so $\beta_j = q_{j-1}^T M \bar{r}_j$. But, using (10.6.9) to eliminate \bar{r}_j this gives $\beta_j = q_{j-1}^T M K_\sigma^{-1} M q_j$ and applying the transpose of (10.6.9) to the q_{j-1}^T vector gives

$$\beta_j = \bar{r}_{j-1}^T M q_j \quad (10.6.13)$$

Finally, expanding \bar{r}_{j-1} in terms of its pure component, r_{j-1} , and the preceding Lanczos vectors, as in (10.6.10), the transpose of (10.6.13) becomes

$$\beta_j = q_j^T M r_{j-1} + \alpha_{j-1} q_j^T M q_{j-1} + \beta_{j-1} q_j^T M q_{j-2} + \gamma_{j-1} q_j^T M q_{j-3} + \dots \quad (10.6.14)$$

It is evident that all terms except the first vanish on the right-hand side. Now q_j is the vector obtained by normalizing r_{j-1} , i.e.,

$$q_j = \frac{1}{\|r_{j-1}\|_M} r_{j-1} \quad (10.6.15)$$

where $\|r_{j-1}\|_M = (r_{j-1}^T M r_{j-1})^{1/2}$. Using this expression for q_j in (10.6.14), we obtain β_j as $\beta_j = (1/\|r_{j-1}\|_M) r_{j-1}^T M r_{j-1}$, or

$$\beta_j^2 = \mathbf{r}_{j-1}^T \mathbf{M} \mathbf{r}_{j-1} \quad (10.6.16)$$

This is an alternative to (10.6.13) for evaluating β_j . In [26] Scott established that computing β_j using (10.6.16) is preferable to (10.6.13) and numerical experiments also confirm his results. Using (10.6.16) ensures that the Lanczos vectors are properly normalized even if $\mathbf{q}_{j+1}^T \mathbf{M} \mathbf{q}_{j-1}$ is not exactly zero.

Continuing in the same way, the amplitude of \mathbf{q}_{j-2} contained in $\bar{\mathbf{r}}_j$ is found to be

$$\gamma_j = \mathbf{q}_{j-2}^T \mathbf{M} \bar{\mathbf{r}}_j \quad (10.6.17)$$

Following the procedure used to derive (10.6.14), this leads to

$$\gamma_j = \mathbf{q}_j^T \mathbf{M} \mathbf{r}_{j-2} + \alpha_{j-2} \mathbf{q}_j^T \mathbf{M} \mathbf{q}_{j-2} + \beta_{j-2} \mathbf{q}_j^T \mathbf{M} \mathbf{q}_{j-3} + \gamma_{j-2} \mathbf{q}_j^T \mathbf{M} \mathbf{q}_{j-4} + \dots \quad (10.6.18)$$

But, using the normalizing relationship equivalent to (10.6.15), $\mathbf{r}_{j-2} = \beta_{j-1} \mathbf{q}_{j-1}$. Hence, when this is substituted into (10.6.18) all terms on the right-hand side vanish, with the result that $\gamma_j = 0$. A corresponding procedure could be used to demonstrate that all further terms in the expansion for $\bar{\mathbf{r}}_j$, (10.6.10), vanish; in other words, *the orthogonalization procedure used in generating each Lanczos vector need be applied only to the previous two vectors*.

A summary of the Lanczos algorithm is presented in Table 10.6.1.

TABLE 10.6.1. The Lanczos Algorithm

Given an arbitrary vector \mathbf{r}_0 then:

1. Set
 - a. $\mathbf{q}_0 = \mathbf{0}$
 - b. $\beta_1 = (\mathbf{r}_0^T \mathbf{M} \mathbf{r}_0)^{1/2}$
 - c. $\mathbf{q}_1 = \frac{\mathbf{r}_0}{\beta_1}$
 - d. $\mathbf{p}_1 = \mathbf{M} \mathbf{q}_1$
2. For $j = 1, 2, \dots$, repeat:
 - a. $\bar{\mathbf{r}}_j = \mathbf{K}_\sigma^{-1} \mathbf{p}_j$
 - b. $\hat{\mathbf{r}}_j = \bar{\mathbf{r}}_j - \mathbf{q}_{j-1} \beta_j$
 - c. $\alpha_j = \mathbf{q}_j^T \mathbf{M} \hat{\mathbf{r}}_j = \mathbf{p}_j^T \hat{\mathbf{r}}_j$
 - d. $\mathbf{r}_j = \hat{\mathbf{r}}_j - \mathbf{q}_j \alpha_j$
 - e. $\bar{\mathbf{p}}_j = \mathbf{M} \mathbf{r}_j$
 - f. $\beta_{j+1} = (\mathbf{r}_j^T \mathbf{M} \mathbf{r}_j)^{1/2} = (\bar{\mathbf{p}}_j^T \mathbf{r}_j)^{1/2}$
 - g. if enough vectors, then terminate the loop
 - h. $\mathbf{q}_{j+1} = \frac{1}{\beta_{j+1}} \mathbf{r}_j$
 - i. $\mathbf{p}_{j+1} = \frac{1}{\beta_{j+1}} \bar{\mathbf{p}}_j$

The above process may be started from a random vector, r_0 , with $q_0 = \mathbf{0}$ and $\beta_1 = (r_0^T M r_0)^{1/2}$. At a typical step, j , the Lanczos algorithm computes α_j , β_{j+1} , and q_{j+1} , in order. In addition to the storage needs of K_σ and M , the algorithm requires storage for five vectors of length n ; one for each of the vectors, q_{j-1} , q_j , $M r_j$, p_j , and r_j . The total cost for one step of the algorithm involves a multiply with M , the solution of a system of equations with K_σ as the coefficient matrix, two inner products and four products of a scalar by a vector.

10.6.6 Reduction to Tridiagonal Form

Using the results of the previous section, (10.6.10) can be rewritten as the three-term relation

$$r_j = \beta_{j+1} q_{j+1} = K_\sigma^{-1} M q_j - q_j \alpha_j - q_{j-1} \beta_j \quad (10.6.19)$$

where $\alpha_j = q_j^T M K_\sigma^{-1} M q_j$ and r_j is normalized with respect to the mass matrix to obtain q_{j+1} with normalizing factor $\beta_{j+1} = (r_j^T M r_j)^{1/2}$. After m Lanczos steps all the quantities obtained from equation (10.6.19) can be arranged in a global matrix form

$$\left[K_\sigma^{-1} M \right] \left[Q_m \right] - \left[Q_m \right] \left[T_m \right] = \begin{bmatrix} 0 & 0 & \dots & 0 & r_m \end{bmatrix} = r_m e_m^T \quad (10.6.20)$$

Here $e_m^T = \langle 0, 0, \dots, 0, 1 \rangle$, Q_m is an $n \times m$ matrix with columns q_i , $i = 1, 2, \dots, m$, and T_m is a *tridiagonal matrix* of the form

$$T_m = \begin{bmatrix} \alpha_1 \beta_2 & & & & \\ \beta_2 \alpha_2 \beta_3 & & & & \\ \beta_3 & \ddots & & & \\ & \ddots & \ddots & & \\ & & & \ddots & \beta_m \\ & & & & \beta_m \alpha_m \end{bmatrix} \quad (10.6.21)$$

The orthogonality property of the Lanczos vectors, $Q_m^T M Q_m = I_m$, where I_m is the $m \times m$ identity matrix, can be used in (10.6.20) to obtain

$$Q_m^T M K_\sigma^{-1} M Q_m = T_m \quad (10.6.22)$$

Choosing the set of Lanczos vectors, Q_m , for the trial vectors, the Rayleigh-Ritz procedure can be used to obtain the best approximation to the eigenvectors of (10.6.6). The approximating Ritz vectors will then be of the form

$$y_i^{(m)} = Q_m s_i^{(m)}, \quad i = 1, \dots, m \quad (10.6.23)$$

When a residual vector, $\{K_{\sigma}^{-1}M\mathbf{y}_i^{(m)} - \theta_i^{(m)}\mathbf{y}_i^{(m)}\}$, associated with the pair $\{\theta_i^{(m)}, \mathbf{y}_i^{(m)}\}$ is M -orthogonal to the set of Lanczos vectors, then $\{\theta_i^{(m)}, \mathbf{y}_i^{(m)}\}$ is a Ritz pair. Accordingly

$$\mathbf{Q}_m^T M \{K_{\sigma}^{-1}M\mathbf{y}_i^{(m)} - \theta_i^{(m)}\mathbf{y}_i^{(m)}\} = \mathbf{0}$$

Using the relation between the Ritz vector, $\mathbf{y}_i^{(m)}$ and the Lanczos vectors, given in (10.6.23), together with the orthonormality condition of the q 's, and the tridiagonal properties of the Lanczos vectors, (10.6.22), the above equation reduces to the tridiagonal eigenproblem

$$\boxed{T_m s_i^{(m)} - \theta_i^{(m)} s_i^{(m)} = 0} \quad (10.6.24)$$

Thus $\{\theta_i^{(m)}, s_i^{(m)}\}$ is an eigenpair of the tridiagonal matrix, T_m . As the total number of Lanczos vectors increases, i.e., as we take more Lanczos steps, the size of the tridiagonal matrix increases and the eigenvalues of T_m converge to the eigenvalues of the transformed problem (10.6.6), $1/(\lambda_i - \sigma)$. When $m = n$, the order of K , then $\theta_i^{(n)} = 1/(\lambda_i - \sigma)$ for all i , but we hope to stop long before $m = n$. The steps enumerated in Step 2 of Table 10.6.1 are repeated until the Ritz pairs $\{\theta_i^{(m)}, \mathbf{y}_i^{(m)}\}$ have sufficiently converged to the desired eigenpairs.

Example 2

$$K = \frac{1}{80} \begin{bmatrix} 39 & -9 & 21 & -11 \\ -9 & 39 & -11 & 21 \\ 21 & -11 & 39 & -9 \\ -11 & 21 & -9 & 39 \end{bmatrix}$$

For this example we assume the mass matrix is the identity. Then the eigenvalue matrix for this problem is $\Lambda = \text{diag}(\frac{1}{3}, \frac{1}{4}, \frac{1}{2}, 1)$. The Lanczos algorithm presented here requires the solution of a linear system of equations with K . We therefore give K^{-1} explicitly for convenience. Accordingly,

$$K^{-1} = \frac{1}{2} \begin{bmatrix} 6 & 0 & -3 & 1 \\ 0 & 6 & 1 & -3 \\ -3 & 1 & 6 & 0 \\ 1 & -3 & 0 & 6 \end{bmatrix}$$

Choosing a starting vector $r_0 = \langle 1, 0, 0, 0 \rangle$, then

Step 1:

$$\beta_1 = \|r_0\| = 1, \quad q_1 = \frac{r_0}{\beta_1} = \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix}, \quad K^{-1} q_1 = \frac{1}{2} \begin{Bmatrix} 6 \\ 0 \\ -3 \\ 1 \end{Bmatrix}$$

$$\alpha_1 = \mathbf{q}_1^T K^{-1} \mathbf{q}_1 = 3, \quad r_1 = K^{-1} \mathbf{q}_1 - \alpha_1 \mathbf{q}_1 = \frac{1}{2} \begin{Bmatrix} 0 \\ 0 \\ -3 \\ 1 \end{Bmatrix}, \quad T_1 = [3]$$

Step 2:

$$\beta_2 = \|r_1\| = \frac{\sqrt{10}}{2}, \quad \mathbf{q}_2 = \frac{r_1}{\beta_2} = \frac{1}{\sqrt{10}} \begin{Bmatrix} 0 \\ 0 \\ -3 \\ 1 \end{Bmatrix}, \quad K^{-1} \mathbf{q}_2 = \frac{1}{\sqrt{10}} \begin{Bmatrix} 5 \\ -3 \\ -9 \\ 3 \end{Bmatrix}$$

$$\alpha_2 = \mathbf{q}_2^T K^{-1} \mathbf{q}_2 = 3, \quad r_2 = K^{-1} \mathbf{q}_2 - \alpha_2 \mathbf{q}_2 - \beta_2 \mathbf{q}_1 = \frac{1}{\sqrt{10}} \begin{Bmatrix} 0 \\ -3 \\ 0 \\ 0 \end{Bmatrix}$$

$$T_2 = \begin{bmatrix} & \frac{\sqrt{10}}{2} \\ 3 & & \\ & \frac{\sqrt{10}}{2} & 3 \end{bmatrix}$$

Step 3:

$$\beta_3 = \|r_2\| = \frac{3}{\sqrt{10}}, \quad \mathbf{q}_3 = \frac{r_2}{\beta_3} = \begin{Bmatrix} 0 \\ -1 \\ 0 \\ 0 \end{Bmatrix}, \quad K^{-1} \mathbf{q}_3 = \frac{1}{2} \begin{Bmatrix} 0 \\ -6 \\ -1 \\ 3 \end{Bmatrix}$$

$$\alpha_3 = \mathbf{q}_3^T K^{-1} \mathbf{q}_3 = 3, \quad r_3 = K^{-1} \mathbf{q}_3 - \alpha_3 \mathbf{q}_3 - \beta_3 \mathbf{q}_2 = \frac{1}{5} \begin{Bmatrix} 0 \\ 0 \\ 2 \\ 6 \end{Bmatrix}$$

$$T_3 = \begin{bmatrix} & \frac{\sqrt{10}}{2} & 0 \\ 3 & & \\ \frac{\sqrt{10}}{2} & 3 & \frac{3}{\sqrt{10}} \\ 0 & \frac{3}{\sqrt{10}} & 3 \end{bmatrix}$$

Step 4:

$$\beta_4 = \|r_3\| = \frac{4}{\sqrt{10}}, \quad \mathbf{q}_4 = \frac{r_3}{\beta_4} = \frac{1}{\sqrt{10}} \begin{Bmatrix} 0 \\ 0 \\ 1 \\ 3 \end{Bmatrix}, \quad K^{-1} \mathbf{q}_4 = \frac{1}{\sqrt{10}} \begin{Bmatrix} 0 \\ -4 \\ 3 \\ 9 \end{Bmatrix}$$

$$\alpha_4 = \mathbf{q}_4^T \mathbf{K}^{-1} \mathbf{q}_4 = 3, \quad \mathbf{r}_4 = \mathbf{K}^{-1} \mathbf{q}_4 - \alpha_4 \mathbf{q}_4 - \beta_4 \mathbf{q}_3 = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

$$T_4 = \begin{bmatrix} 3 & \frac{\sqrt{10}}{2} & 0 & 0 \\ \frac{\sqrt{10}}{2} & 3 & \frac{3}{\sqrt{10}} & 0 \\ 0 & \frac{3}{\sqrt{10}} & 3 & \frac{4}{\sqrt{10}} \\ 0 & 0 & \frac{4}{\sqrt{10}} & 3 \end{bmatrix}$$

The eigenvalues of the tridiagonal matrices converge to the *inverses* of the eigenvalues of \mathbf{K} in this example. This can be demonstrated by computing the eigenvalues of the tridiagonal at each step as shown in Table 10.6.2

TABLE 10.6.2 Convergence
of the Ritz Values for the Small
 4×4 Example

| j | Eigenvalues of T_j |
|-----|--------------------------------|
| 1 | 3.0000 |
| 2 | 1.4189, 4.5811 |
| 3 | 1.1561, 3.0000, 4.8439 |
| 4 | 1.0000, 2.0000, 4.0000, 5.0000 |

In Figure 10.6.1 we plot the Ritz values for a larger example. Notice the early convergence at both ends of the spectrum.

10.6.7 Convergence Criterion for Eigenvalues

The eigenvalues, $\theta_i^{(j)}$, $i = 1, \dots, j$, of the tridiagonal T_j are the Rayleigh-Ritz approximations to eigenvalues of (10.6.6). As j increases these Ritz values get closer to the eigenvalues they are approximating. Often users wait until the change in the computed quantities is within a specified tolerance; that is

$$\frac{|\theta^{(j-1)} - \theta^{(j)}|}{|\theta^{(j)}|} \leq \text{tol} \quad (10.6.25)$$

Here we demonstrate the weakness of such convergence tests. Suppose one is performing inverse iteration with a specified shift, σ , in an interval of interest $[\sigma_1, \sigma_2]$; i.e.,

$$\sigma = \gamma\sigma_1 + (1 - \gamma)\sigma_2$$

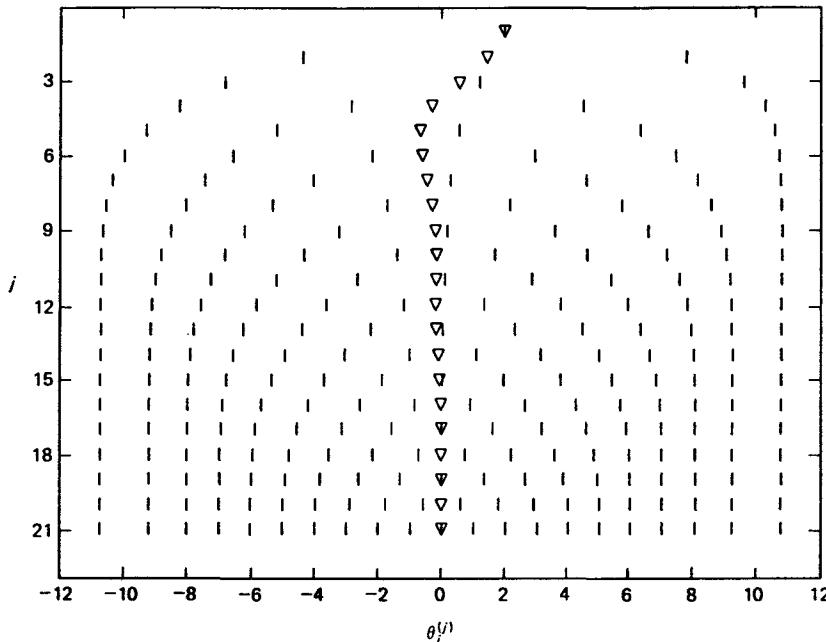


Figure 10.6.1 A typical pattern for the progress of the eigenvalues of the tridiagonal matrix, T_j . The symbol ∇ indicates the last diagonal entry of T_j .

where γ is a positive number less than 1. Further, assume that the problem has eigenvalues, λ_a and λ_b outside this interval with $\lambda_a < \sigma_1$ and $\lambda_b > \sigma_2$. Then the corresponding eigenvalues of the transformed problem, (10.6.6), are $\nu_a = 1/(\lambda_a - \sigma)$ and $\nu_b = 1/(\lambda_b - \sigma)$, with $\nu_a < 0$ and $\nu_b > 0$. Now, let us perform a step of the inverse iteration method with an unfortunate starting vector, $x^{(1)} = z_a \sin \psi + z_b \cos \psi$. Here, z_a and z_b are eigenvectors corresponding to λ_a and λ_b , respectively. We assume that the eigenvectors are normalized with respect to the mass matrix. Then $x^{(1)}$ will also be normalized. The Ritz value due to $x^{(1)}$ is

$$\begin{aligned}\theta^{(1)} &= x^{(1)T} K x^{(1)} \\ &= \lambda_a \sin^2 \psi + \lambda_b \cos^2 \psi\end{aligned}$$

After one step of inverse iteration the improved vector is

$$x^{(2)} = \frac{1}{[\nu_a^2 \sin^2 \psi + \nu_b^2 \cos^2 \psi]^{1/2}} (\nu_a z_a \sin \psi + \nu_b z_b \cos \psi)$$

and the Ritz value associated with $x^{(2)}$ is

$$\theta^{(2)} = \frac{1}{[\nu_a^2 \sin^2 \psi + \nu_b^2 \cos^2 \psi]^{1/2}} (\nu_a^2 \lambda_a \sin^2 \psi + \nu_b^2 \lambda_b \cos^2 \psi)$$

Now if $\lambda_a = \sigma - \tau$ and $\lambda_b = \sigma + \tau$, in which case $\nu_a = -1/\tau$ and $\nu_b = 1/\tau$, then

$$\theta^{(1)} = \theta^{(2)} = (\sigma - \tau) \sin^2 \psi + (\sigma + \tau) \cos^2 \psi = \sigma + \tau \cos 2\psi$$

These θ 's, referred to as "ghost" eigenvalues in [25], satisfy (10.6.25) and therefore would be accepted as eigenvalues. Further, an appropriate choice for ψ can result in a Ritz value that is in the interval of interest. This behavior, although rare, has been observed in certain implementations of the subspace iteration method [25] and reflects a serious difficulty. We refer to this as *misconvergence* in [23] and, as we observed therein, it cannot be detected by a natural criterion such as (10.6.25). A rigorous inexpensive termination criterion can be obtained using the residual error bound given later.

Consider a pair $\{\theta, y\}$, which approximates an eigenpair $\{\nu, z\}$, of (10.6.6), where ν is the closest eigenvalue of (10.6.6) to θ . For simplicity we consider only normalized Ritz vectors, $\|y\|_M = 1$ where $\|y\|_M = (y^T M y)^{1/2}$. The residual vector associated with this approximation is given by

$$r = K_\sigma^{-1} M y - \theta y \quad (10.6.26)$$

The norm of the residual vector can then be used to assess the accuracy of θ through the inequality

$$|\nu - \theta| \leq \|r\|_M \quad (10.6.27)$$

For the proof and an in-depth study see [21]. This result allows us to predict the accuracy of any candidate eigenpair of a matrix.

We use the residual vector associated with the Ritz pair $\{\theta_i^{(m)}, y_i^{(m)}\}$ given by (10.6.23) and (10.6.24) to check for their convergence. The tridiagonal property of the Lanczos algorithm, (10.6.20), greatly simplifies the computation of the residual norm. Postmultiplying (10.6.20) by $s_i^{(m)}$ leads to

$$K_\sigma^{-1} M Q_m s_i^{(m)} - Q_m T_m s_i^{(m)} = r_m e_m^T s_i^{(m)} \quad (10.6.28)$$

Since $s_i^{(m)}$ is an eigenvector of T_m (see (10.6.24)), we can replace $T_m s_i^{(m)}$ by $\theta_i^{(m)} s_i^{(m)}$. Further, by definition (10.6.23), $Q_m s_i^{(m)}$ is simply the Ritz vector $y_i^{(m)}$ and so (10.6.28) will reduce to

$$K_\sigma^{-1} M y_i^{(m)} - \theta_i^{(m)} y_i^{(m)} = r_m e_m^T s_i^{(m)} \quad (10.6.29)$$

Taking norms,

$$\begin{aligned} \|K_\sigma^{-1} M y_i^{(m)} - \theta_i^{(m)} y_i^{(m)}\|_M &= \|r_m e_m^T s_i^{(m)}\|_M \\ &= \|r_m\|_M |e_m^T s_i^{(m)}| \\ &= \beta_{m+1} |\zeta_i| \end{aligned} \quad (10.6.30)$$

where $\zeta_i = e_m^T s_i^{(m)}$ is the bottom element of $s_i^{(m)}$, the normalized eigenvector of T_m . β_{m+1} is a scalar quantity that is computed in the course of the Lanczos process. The bottom element of the eigenvector of T_m can be obtained at little cost once the

associated eigenvalue is found. Therefore, the residual norm associated with a Ritz pair can be computed as $\rho_{mi} = \beta_{m+1} |\zeta_i|$ and the Ritz value is considered converged once $\rho_{mi} < \text{tol}$. There is no need to compute $y_i^{(m)}$ until it has converged.

Multiple Eigenvalues

In exact arithmetic the simple Lanczos algorithm cannot compute a second copy of a multiple eigenvalue. Suppose λ is an eigenvalue with multiplicity two. Then there is no unique eigenvector associated with λ . In fact one can obtain two vectors, z_1 and z_2 , such that any linear combination of these vectors is also an eigenvector of λ . Moreover, it is possible to obtain a linear combination of z_1 and z_2 , $\bar{z} = \xi_1 z_1 + \xi_2 z_2$, that is orthogonal to q_1 . Then the eigenvector \bar{z} will be orthogonal to all the subsequent Lanczos vectors. Therefore \bar{z} will never be found.

Fortunately, this argument only holds in exact arithmetic. In finite precision, roundoff comes to the rescue. If q_1 is perfectly orthogonal to an eigenvector, \bar{z} , then because of roundoff error, q_2 will have a small component along \bar{z} . This component, although tiny, will eventually grow such that \bar{z} can be represented by a linear combination of the computed Lanczos vectors. However, the second copy of λ will converge some steps after the first has converged.

10.6.8 Loss of Orthogonality

In a previous section we derived the equations governing the Lanczos algorithm. These relations are only satisfied by quantities obtained in exact arithmetic. In finite precision, however, each computation introduces a small error and therefore each computed quantity will differ from its exact counterpart. Our objective here is to describe the effect of roundoff on the Lanczos process. For this purpose we need to introduce an important quantity that measures the accuracy of the arithmetic. Let ϵ be the smallest number in the computer such that $1 + \epsilon > 1$. It is known as the *unit roundoff error*.

Although the tridiagonal relation, (10.6.20), is preserved to within roundoff, the M -orthogonality property of the Lanczos vectors completely breaks down after a certain number of steps, depending on ϵ and the distribution of the eigenvalues of (10.6.6). The Lanczos vectors, which are orthogonal in exact arithmetic, not only lose their orthogonality, but may even become linearly dependent. Initially it was believed that loss of orthogonality is due to cancellations that occur each time r_j is evaluated using (10.6.19). This step is simply M -orthogonalization of $K_\sigma^{-1} M q_j$ against q_j and q_{j-1} . Comparing the final vector resulting from this computation to the starting vector, one can obtain a measure of the cancellation that occurs in this step; i.e., the ratio

$$\chi_j^2 = \frac{\|r_j\|_M^2}{\|K_\sigma^{-1} M q_j\|_M^2} = \frac{\beta_{j+1}^2}{\beta_j^2 + \alpha_j^2 + \beta_{j+1}^2} \quad (10.6.31)$$

indicates how much cancellation has occurred. χ_j is the sine of the angle between the vector $K_\sigma^{-1} M q_j$ and the plane containing the vectors q_j and q_{j-1} . When χ_j is zero it indicates that in the j th Lanczos step we are orthogonalizing a vector that is already

in this plane against \mathbf{q}_j and \mathbf{q}_{j-1} , and therefore complete cancellation occurs. In practice χ_j rarely drops below $\frac{1}{10}$. When χ_j drops to $\frac{1}{100}$, for instance, it indicates that $K_\sigma^{-1} M \mathbf{q}_j$ is nearly parallel to this plane, and therefore at the end of this Lanczos step the computed \mathbf{r}_j may not be orthogonal to the plane containing \mathbf{q}_j and \mathbf{q}_{j-1} to working accuracy. In such cases \mathbf{r}_j should be orthogonalized against \mathbf{q}_j and \mathbf{q}_{j-1} a second time. For a long time it was believed that loss of orthogonality was solely due to this cancellation. Indeed if this were the case, then we would have no option other than a complete reorthogonalization at each step. However, as we soon shall see, a completely different mechanism is at work.

Suppose for the moment the algorithm was executed in exact arithmetic for j steps, except that at some step $k < j$ a small error was introduced into the computation of \mathbf{q}_k . The first $k - 1$ Lanczos vectors will be perfectly M -orthonormal, but they will not be orthogonal to all the vectors computed after the k th step. The error introduced at step k will be amplified in the subsequent steps to such an extent that linear independency may also be lost.

Hence, the loss of orthogonality can be viewed as the subsequent amplification of the error introduced after each computation. To analyze the way in which orthogonality deteriorates, we let \mathbf{Q}_m denote the computed Lanczos vectors and define the following matrix:

$$\mathbf{H}_m = \mathbf{Q}_m^T M \mathbf{Q}_m \quad (10.6.32)$$

where the i, j -component of \mathbf{H}_m is $\eta_{i,j} = \mathbf{q}_i^T M \mathbf{q}_j$. In exact arithmetic \mathbf{H}_m is the identity matrix. The off-diagonals of \mathbf{H}_m will depend on ϵ , the unit roundoff error. Further, (10.6.19) will be satisfied by the computed quantities only to within roundoff error. Now, multiply (10.6.19) by $\mathbf{q}_i^T M$ and use the above definition to get the approximate relation

$$\mathbf{q}_i^T M K_\sigma^{-1} M \mathbf{q}_j \cong \alpha_j \eta_{i,j} + \beta_j \eta_{i,j-1} + \beta_{j+1} \eta_{i,j+1} \quad (10.6.33)$$

A similar equation can be obtained when the above procedure is repeated for the i th Lanczos step

$$\mathbf{q}_j^T M K_\sigma^{-1} M \mathbf{q}_i \cong \alpha_i \eta_{j,i} + \beta_i \eta_{j,i-1} + \beta_{i+1} \eta_{j,i+1} \quad (10.6.34)$$

By symmetry of $M K_\sigma^{-1} M$, the terms on the left-hand side of (10.6.33) and (10.6.34) are equal and therefore can be eliminated by subtraction, resulting in the relation

$$\beta_{j+1} \eta_{j+1,i} \cong \beta_{i+1} \eta_{j,i+1} + (\alpha_i - \alpha_j) \eta_{j,i} + \beta_i \eta_{j,i-1} - \beta_j \eta_{j-1,i} \quad (10.6.35)$$

This recursion holds for $j \geq 2$ and $1 \leq i \leq j - 1$ and starts by assuming that the diagonal of \mathbf{H}_m is the identity, $\eta_{j,j} = 1$ for all $j \geq 1$, and the first off-diagonal of \mathbf{H}_m is at roundoff level, $\eta_{j,j-1} = \epsilon$ for $j \geq 2$. The above relation is due to Simon [27]. It provides a means of estimating the elements of a column of \mathbf{H}_m from the elements of \mathbf{T}_m and the elements in the previous columns of \mathbf{H}_m . This recursion can be restated in the vector form

$$\beta_{j+1} \mathbf{h}_{j+1} \cong \mathbf{T}_{j-1} \mathbf{h}_j - \alpha_j \mathbf{h}_j - \beta_j \mathbf{h}_{j-1} \quad (10.6.36)$$

where \mathbf{h}_{j-1} , \mathbf{h}_j and \mathbf{h}_{j+1} are vectors of length $j - 1$ containing the top $j - 1$ elements of the $(j - 1)$ st, j th, and $(j + 1)$ st columns of $(\mathbf{H}_m - \mathbf{I}_m)$. Here, the bottom element of \mathbf{h}_{j-1} is zero. Then, all the terms in \mathbf{h}_j depend on ϵ . Taking norms we can show

$$\begin{aligned}\beta_{j+1} \|\mathbf{h}_{j+1}\| &\leq (\|\mathbf{T}_{j-1}\| + |\alpha_j|) \|\mathbf{h}_j\| + \beta_j \|\mathbf{h}_{j-1}\| \\ &\leq 2 \|\mathbf{T}_j\| \max(\|\mathbf{h}_j\|, \|\mathbf{h}_{j-1}\|)\end{aligned}\quad (10.6.37)$$

This result shows that the level of nonorthogonality can grow by at most a factor of $2 \|\mathbf{T}_j\| / \beta_{j+1}$ after each step. A drop in the value of β_{j+1} can result in a sudden loss of orthogonality but this rarely occurs.

An alternative characterization of the pattern in which orthogonality is lost was presented by Paige [18–20]. Instead of examining the vector $\mathbf{h}_{j+1} = \mathbf{Q}_j^T \mathbf{M} \mathbf{q}_{j+1}$, he looked at a linear combination of the components in this vector. To be more specific, he examined the inner product between each Ritz vector $\mathbf{y}_i^{(j)}$ given by (10.6.23) and \mathbf{q}_{j+1} . That is

$$\begin{aligned}\mathbf{y}_i^{(j)T} \mathbf{M} \mathbf{q}_{j+1} &= s_i^{(j)T} \mathbf{Q}_j^T \mathbf{M} \mathbf{q}_{j+1} \\ &= s_i^{(j)T} \mathbf{h}_{j+1}\end{aligned}\quad (10.6.38)$$

In exact arithmetic this value is zero. However, in his work Paige showed that

$$\mathbf{y}_i^{(j)T} \mathbf{M} \mathbf{q}_{j+1} = \frac{\gamma_{ji} \epsilon \|\mathbf{T}_j\|}{\beta_{j+1} |\zeta_i|} \quad (10.6.39)$$

Recall that ζ_i is the bottom element of $s_i^{(j)}$, the i th eigenvector of \mathbf{T}_j . γ_{ji} is a scalar quantity usually close to unity. We omit the derivation of this result and refer the interested reader to [21]. Note that Paige's result also shows that a sudden drop in β_{j+1} can result in a severe loss of orthogonality. Moreover, recall the quantity in the denominator is also a measure of the convergence of the Ritz value $\theta_i^{(j)}$ (see (10.6.30)). The only way the left side of (10.6.39) can rise up to values like 0.1 is for $\rho_{ji} (= \beta_{j+1} |\zeta_i|)$ to drop down to $10\gamma_{ji}\epsilon \|\mathbf{T}_j\|$, so

Loss of orthogonality \Rightarrow convergence of a Ritz value

When only a single Ritz value converges at step j , then \mathbf{q}_{j+1} loses orthogonality by tilting toward the converged Ritz vector, which in turn is a linear combination of the previous Lanczos vectors. When more than one Ritz vector converges simultaneously, the picture is more complicated. In this case \mathbf{q}_{j+1} tilts toward a linear combination of these vectors.

Return of Banished Ritz Vectors

In theory, if two successive Lanczos vectors, \mathbf{q}_{j-1} and \mathbf{q}_j , are orthogonal to an eigenvector, \mathbf{z} , then all the subsequent Lanczos vectors will also be orthogonal to \mathbf{z} . In practice, however, a converged Ritz vector, \mathbf{y}_i , will not remain orthogonal to all the subsequent Lanczos vectors. This is a consequence of the same mechanism by which

multiple eigenvalues are found. Roundoff errors will add to each Lanczos vector a small component of y_i , which will grow eventually to such a magnitude that orthogonality to y_i is lost. This phenomenon can also be observed in the inverse iteration method; i.e., orthogonalizing the starting vector against the first eigenvector does not guarantee convergence of the iteration vectors to the next eigenvector.

The state of orthogonality between a converged Ritz vector, y_i , and the current Lanczos vector, q_j , can be measured by the component of y_i along q_j . We define $\tau_j = y_i^T M q_j$. Then multiplying (10.6.19) by $y_i^T M$ and considering the effect of round-off on (10.6.19) yields

$$\beta_{j+1} y_i^T M q_{j+1} \approx y_i^T M K_\sigma^{-1} M q_j - \alpha_j y_i^T M q_j - \beta_j y_i^T M q_{j-1}$$

Using the fact that y_i is a converged Ritz value, i.e., $K_\sigma^{-1} M y_i = \theta_i y_i$, together with the definition of τ_j we obtain

$$\tau_{j+1} \cong \frac{(\theta_i - \alpha_j)\tau_j - \beta_j \tau_{j-1}}{\beta_{j+1}}$$

(10.6.40)

This recurrence can be updated for each converged Ritz value, θ_i . The magnitude of τ_j can be used as an indicator for loss of orthogonality against a converged Ritz vector. The growth of the components of the Lanczos vectors along a converged Ritz vector is referred to as the *return of a banished Ritz vector*.

10.6.9 Restoring Orthogonality

In this section we look at a number of preventive measures that we can adopt to maintain a certain level of orthogonality. Lanczos [14] was aware of the effects of round-off on the algorithm when he presented his work. He proposed that the newly computed Lanczos vector, q_{j+1} , be explicitly orthogonalized against all the preceding vectors at the end of each step j . We will refer to this technique as the “full reorthogonalization” method. This scheme is adopted in [12, 29]. With this procedure the Lanczos vectors will meet the stringent requirement

$$|q_i^T M q_j| < n\epsilon, \quad i \neq j \quad (10.6.41)$$

Although this scheme increases the overall cost of an eigenvalue computation, for short Lanczos runs (when the number of Lanczos steps is less than the half-bandwidth of K) the increase in cost is small compared to the cost of solving a system of equations with K_σ . For longer runs the cost of a full reorthogonalization step will begin to dominate the cost of a Lanczos step, although vector computers will delay this effect.

The orthogonality condition of (10.6.41) can be replaced by a more relaxed condition,

$$|q_i^T M q_j| < \sqrt{n\epsilon}, \quad i \neq j \quad (10.6.42)$$

We refer to this as the *semiorthogonality condition* and we refer to procedures that adopt the weaker condition as *selective orthogonalization methods*. Imposing the

more relaxed condition can result in considerable reduction in the number of operations in the reorthogonalization step and semiorthogonality is sufficient to make T_j exact to working precision.

We consider two different reorthogonalization schemes that adopt the more relaxed condition of (10.6.42).

A. Orthogonalization against Ritz vectors: This procedure is a consequence of the result of Paige [18–20]. As soon as a Ritz value converges, its corresponding Ritz vector is computed and the component of this vector along q_{j+1} is purged. Further, for each converged Ritz value, the three-term recurrence for τ_j is updated and whenever τ_j becomes greater than $\sqrt{\epsilon}$ in absolute value, it signals that the component of the corresponding eigenvector has grown too much. So the new Lanczos vector is orthogonalized against this known eigenvector [22, 25].

B. Orthogonalization against previous Lanczos vectors: This scheme can be based on either of the techniques given in [13, 27]. In [27] the vector h_{j+1} is updated using (10.6.36) and the magnitudes of its elements are monitored. Whenever the i th element of h_{j+1} is greater than $\sqrt{\epsilon}$ then semiorthogonality is lost between q_{j+1} and q_i . At this step the appropriate Lanczos vectors are brought in from secondary storage and their components along q_{j+1} are removed. This scheme is also adopted in [17].

Both schemes indicate loss of orthogonality at about the same step. The first method performs orthogonalization against fewer vectors and therefore costs less. However, alone it has some shortcomings. The result in (10.6.39) gives an accurate picture at the early stages of a Lanczos run when one or two Ritz values converge. When for a number of steps, a few Ritz values converge at the same time, then a gradual loss of semiorthogonality may occur. This appears most often in a Lanczos run with spectral transformation. For this reason we do not use scheme A on its own. However, for short Lanczos runs where only a very few (less than 10) eigenpairs are wanted, scheme A is very effective.

We advocate a combination of the above two schemes. Whenever possible we orthogonalize against a Ritz vector (scheme A) because of lower costs. We update h_{j+1} using (10.6.36) and monitor its elements. If any of the elements of h_{j+1} is greater than $\sqrt{\epsilon}$, then semiorthogonality may have been lost. This is the only procedure we use to determine loss of semiorthogonality, which can occur in two possible ways:

- i. Convergence of a Ritz value.
- ii. Growth of components of computed eigenvectors along the Lanczos vector.

We take different actions for each of these. If the observed orthogonality loss is due to (i), then we perform a step of scheme B. The Lanczos vectors are brought in from secondary storage and the newly computed Lanczos vector, r_j , is orthogonalized against them. Further, to minimize data transfer from secondary storage, we also compute some of the converged Ritz vectors at the same time. Therefore the computation of newly converged Ritz vectors is delayed until the Lanczos vectors are brought back for reorthogonalization. We should note that when (i) occurs, both methods A and B would require recalling the Lanczos vectors from secondary storage

but for different reasons: the first for computing Ritz vectors and the second for reorthogonalization. We perform both. Our method would clearly require more operations than scheme A but has the following two advantages:

- An eigenvector is computed in a few steps after the convergence of its eigenvalue and therefore has more than half correct digits.
- The gradual loss of orthogonality just mentioned will not occur with the combined procedure.

However, if loss of orthogonality is due to (ii), then we can use scheme A, i.e., orthogonalize against a computed Ritz vector. For this reason, the τ recurrence, (10.6.40), is also monitored to establish which Ritz vector, y_k , has contaminated q_{j+1} .

The orthogonalization of q_{j+1} against y_k alters the state of orthogonality among the Lanczos vectors. The modified \bar{q}_{j+1} is given by

$$\bar{q}_{j+1} = q_{j+1} - \xi_k y_k$$

where $\xi_k = y_k^T M q_{j+1}$. Let $\bar{h}_{j+1} = Q_j^T M \bar{q}_{j+1}$. Then

$$\begin{aligned}\bar{h}_{j+1} &= Q_j^T M (q_{j+1} - \xi_k y_k) \\ &= h_{j+1} - \xi_k s_k\end{aligned}\tag{10.6.43}$$

An approximation to ξ_k can be obtained via

$$\xi_k = y_k^T M q_{j+1} = s_k^T Q_j^T M q_{j+1} = s_k^T h_{j+1}$$

If the approximate result for ξ_k is used then (10.6.43) reduces to the orthogonalization of h_{j+1} against s_k . This simple step is used to modify h_{j+1} to reflect the changes caused by orthogonalizing q_{j+1} against a Ritz vector. The flowchart in Fig. 10.6.5 (see p. 615) gives a global view of our implementation of the selective orthogonalization algorithm.

When it becomes necessary to restore semiorthogonality, the computed vector q_{j+1} must be orthogonalized against some linear combination of the previous Lanczos vectors. q_j and q_{j-1} remain unchanged at the end of this step. But, at the next step, q_j appears again in the computation of q_{j+2} . If no action is taken, then q_{j+2} will be contaminated by q_j and the reorthogonalization efforts of the previous step would be wasted. Therefore a second reorthogonalization step must be performed. If the Lanczos vectors or the converged Ritz vectors reside in secondary storage, they must be retrieved in two successive steps.

Alternatively, at the same time the orthogonality of q_{j+1} is being restored, we can also perform similar modifications on q_j . Then at the end of the next step, no reorthogonalization of q_{j+2} will be necessary. The number of operations for this scheme is the same as that of the scheme above, but vectors are retrieved only once and therefore the I/O overhead is halved.

10.6.10 LANSEL Package

In this section we describe all the ingredients that go into the LANSEL eigenpackage and provide sufficient detail for installing LANSEL into a finite element program. The

steps in each subroutine are described in sequence followed by the listing. Higher level routines are described first.

User-Supplied Subroutines

The only interface between the algorithm and the eigenproblem is through two user-supplied subroutines. In general, the structures of the matrices K and M vary greatly from problem to problem. No single subroutine can be designed to take advantage of the special structure of K and M for all problems. Furthermore, the characteristics of a given equation solver depend strongly on the computing environment. Only the user is aware of the properties of his or her matrices and the computer system that is being used. Therefore, the job of solving the linear system of equations $K_\sigma \bar{r}_j = p_j$ and multiplication of a vector by the mass matrix, $\bar{p}_j = M r_j$ is relegated to the user. This has the added advantage that a shifting strategy may be implemented without modifying any part of the routines presented here. The two routines that connect K and M with our program must take the form

1. SUBROUTINE OPK (X, Y, N). This solves the linear system of equations $K_\sigma y = x$. N is the length of the vectors X and Y.
2. SUBROUTINE OPM (X, Y, N). This forms the matrix-vector product $y = Mx$. N is the length of the vectors X and Y.

A third subroutine must also be provided for the management of the Lanczos vectors. From time to time the program calls the subroutine STORE. The subroutine statement for STORE is

SUBROUTINE STORE (V, N, J, ISW)

When ISW = 1 the routine must store the column vector V of length N and identifier J in secondary storage. When ISW = 2, a vector V with identifier J that is in secondary storage must be fetched. This way the user can take full advantage of any data management system that is available.

TABLE 10.6.3 List of Vector Operation Routines used by LANSEL

| Name | Description |
|--------|--|
| IDAMAX | Finds the index of the element of a vector with maximum absolute value; $i = \arg \left(\max_{i=1,n} y_i \right)$. |
| DAXPY | Computes the product of a scalar, a , by a vector, x , adding the result to a vector, y ; $y = ax + y$. |
| DCOPY | Copies a vector, x , to a vector y ; $y = x$. |
| DDOT | Computes the Euclidean inner product of two vectors, x and y ; $\text{dot} = x^T y$. |
| DSCAL | Scales the elements of a vector, y , by a scalar factor, a ; $y = ay$. |
| DZERO | Resets the elements of a vector, y , to zero; $y = 0$. |

LANSEL also makes use of subroutines for performing basic vector operations, such as addition of two vectors. Some of these routines may be found in [10]. Many computers have libraries of carefully written assembly language implementations of these subroutines, which are much faster. In Table 10.6.3 we give a list of vector operations used by LANSEL. Alternatively, the user may wish to write his or her own subroutines to perform these tasks.

Project 1. Write a program to set up and solve the 4×4 example problem of Sec. 10.6.6.

The program should employ the LANSEL package described herein. You will have to provide "user-supplied subroutines" that are general enough to handle the data of the 4×4 problem. Print out all intermediate calculations and compare with the results presented in Sec. 10.6.6.

Project 2. Modify program DLEARN (see Chapter 11) to perform eigenvalue/eigenvector calculations by interfacing it with the LANSEL package. DLEARN contains several utility routines (e.g., equation-solving routines, a variety of matrix-vector manipulation routines, etc.) which can be helpful in developing the "user-supplied subroutines" for the LANSEL package. You will also need to write input and output routines for the computed eigenvalues and eigenvectors. (This project requires a fairly comprehensive knowledge of the DLEARN program.)

Subroutine LANDRV

LANDRV is the driver for the main subroutine LANSEL. Table 10.6.4 gives a brief description of the list of parameters for LANDRV.

TABLE 10.6.4 Description of the Parameters for LANDRV

| Name | Type | Dimension | Description |
|--------|------|-----------|--|
| N | I | scalar | The dimension of the eigenproblem. K and M are $N \times N$ matrices. |
| LANMAX | I | scalar | Upper limit to the number of Lanczos steps. LANMAX must not exceed N. |
| MAXPRS | I | scalar | Upper limit to the number of wanted eigenpairs. MAXPRS must not exceed LANMAX. |
| ENDL | F | scalar | $ENDL = v_L = 1/(\lambda_L - \sigma)$, where λ_L is the left end of the interval containing the wanted eigenvalues (see Fig. 10.6.2). |
| ENDR | F | scalar | $ENDR = v_R = 1/(\lambda_R - \sigma)$, where λ_R is the right end of the interval containing the wanted eigenvalues (see Fig. 10.6.2). |
| NW | I | scalar | Length of the work array W. NW must be at least $6 \times N + 2 \times MAXPRS + 10 \times LANMAX$. A larger NW can result in faster computation time. NW need not be greater than $6 \times N + 2 \times MAXPRS + LANMAX \times (6 + LANMAX)$. |
| W | F | NW | Work array of length NW. The first N words of W hold a user-supplied starting vector. |

| Name | Type | Dimension | Description |
|------|------|-----------|--|
| IW | I | MAXPRS | Work array of length MAXPRS. |
| EIG | F | MAXPRS | EIG will hold the converged Ritz values on return. The Ritz values are in the order they were computed. |
| Y | F | MAXPRS×N | Y will hold the converged Ritz vectors on return. Y is dimensioned as a two-dimensional array Y(N,MAXPRS) and the Ith column of Y holds the Ritz vector corresponding to EIG(I). |
| IERR | I | scalar | IERR is an error flag. A successful execution is indicated when IERR = 0. For a list of error flags see Table 10.6.5. |
| NEIG | I | scalar | The number of computed eigenpairs. |

LANDRV acts as an interface between the user and LANSEL. It performs the following tasks before calling LANSEL:

- i. Checks the control parameters for possible error.
- ii. Allocates the working memory for LANSEL.
- iii. If user does not supply a starting vector in W then LANDRV supplies a random vector.
- iv. Performs the first step of the Lanczos algorithm by calling STPONE.

If an error is detected, IERR is reset and returned as soon as the checking of the control parameters is complete. Each bit of the integer IERR is used as an error flag. IERR is set to zero when entering LANDRV. The Ith bit of IERR can then be reset to 1 using the command $IERR = IERR + 2^{*I}$. For example the third bit is set to 1 by adding 8 to IERR. This way all the errors in the control parameters can be detected at the same time. Table 10.6.5 gives a description of possible errors indicated by IERR.

TABLE 10.6.5 List of Errors
Indicated by IERR

| Bit | Indicates |
|-----|-------------------|
| 1 | $N < 0$ |
| 2 | $LANMAX < 0$ |
| 3 | $ENDR < ENDL$ |
| 4 | $MAXPRS < 0$ |
| 5 | $MAXPRS > LANMAX$ |
| 6 | $LANMAX > N$ |
| 7 | NW too small |

In addition, $IERR = -1$ indicates that the maximum number of Lanczos steps in LANSEL has been reached.

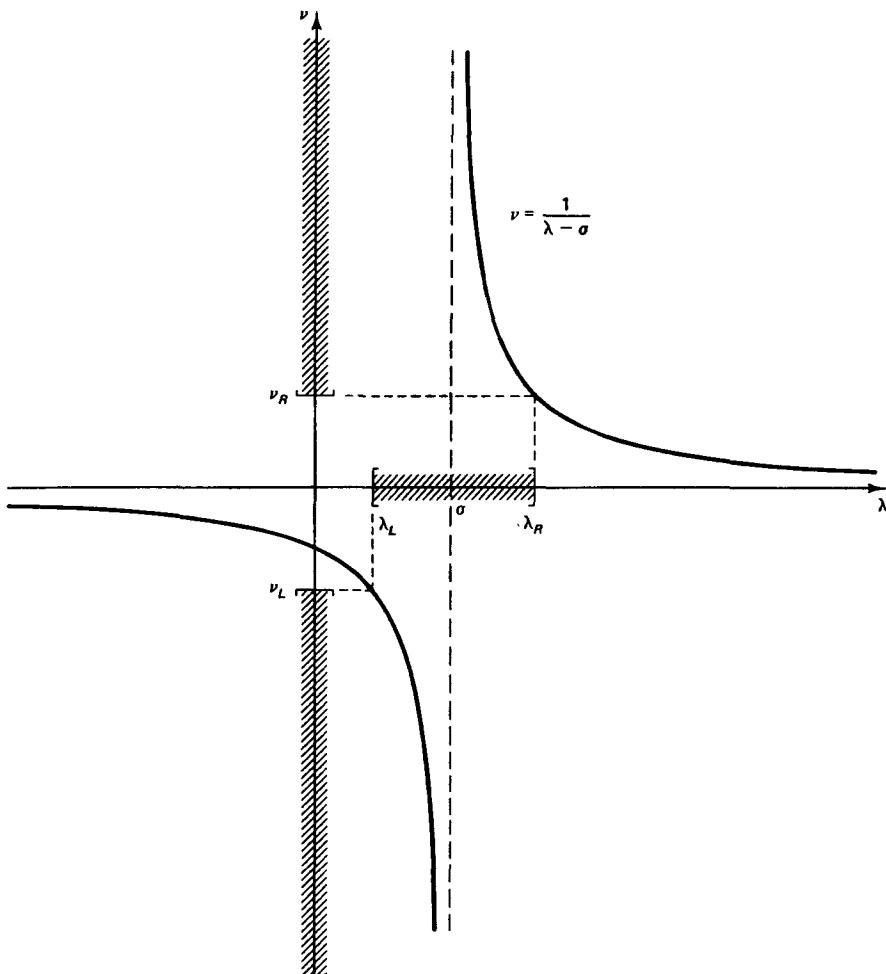


Figure 10.6.2 Spectral transformation of the interval containing the wanted eigenvalues;
 σ denotes the shift.

Remarks

1. LANSEL has no knowledge of the shift. The eigenvalues that it computes are those of the transformed problem (10.6.6). The eigenvalues of the original problem may be obtained by applying the inverse transformation $\lambda = \sigma + 1/\nu$. This is the responsibility of the user. The λ 's closest to σ usually converge first. In particular, for a system with positive-definite K and M , and $\sigma = 0$, the λ 's usually converge in ascending order starting with the smallest.

2. LANSEL may return only a single copy of multiple eigenvalues. To obtain additional copies, restart LANSEL with a new starting vector orthogonal to the computed eigenvectors. (The number of missed eigenvalues may be determined by the technique of spectrum slicing; see Sec. 10.5.1.).

3. Certain rare execution errors, beyond the control of the Lanczos algorithm

may occur in subsidiary routines such as GIVENS. These are indicated by a returned value of IERR greater than 127. The only recourse is to rerun LANSEL with a different shift and/or starting vector.

```

C      SUBROUTINE LANDRV(N, LANMAX, MAXPRS, ENDL, ENDR, NW, W, IW, EIG, Y, IERR,
1                      NEIG)
C
C***** L A N S E L *****
C***** LANCZOS ALGORITHM WITH *****
C***** SELECTIVE ORTHOGONALIZATION *****
C***** B. NOUR - O M I D *****
C***** *****
C.... INPUTS
C
C     N      DIMENSION OF THE EIGENPROBLEM
C     LANMAX  UPPER LIMIT TO THE NUMBER OF LANCZOS STEPS
C     MAXPRS  UPPER LIMIT TO THE NUMBER OF WANTED EIGENPAIRS
C     ENDL    LEFT END OF THE INTERVAL CONTAINING THE WANTED EIGENVALUES
C     ENDR    RIGHT END OF THE INTERVAL CONTAINING THE WANTED EIGENVALUES
C     NW      LENGTH OF THE WORK ARRAY W
C     W       WORK ARRAY OF LENGTH NW
C     IW      WORK ARRAY OF LENGTH MAXPRS
C.... OUTPUTS
C
C     EIG     ARRAY OF LENGTH MAXPRS TO HOLD THE CONVERGED RITZ VALUES
C     Y       ARRAY OF LENGTH MAXPRS*N TO HOLD THE CONVERGED RITZ VECTORS
C     IERR   ERROR FLAG
C     NEIG   TOTAL NUMBER OF CONVERGED EIGENPAIRS
C.... SUBROUTINES: DDOT, GETEPS, LANSEL, DPM, RAN, STPONE
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C DIMENSION NQ(5),Y(1),EIG(1),W(1),IW(1)
C INTEGER*2 IRANDOM1,IRANDOM2
C.... COMMON IDATA
C
C     EIGL   INNER MOST EIGENVALUE CONVERGED FROM LEFT END OF
C             TRANSFORMED SPECTRUM
C     EIGR   INNER MOST EIGENVALUE CONVERGED FROM RIGHT END OF
C             TRANSFORMED SPECTRUM
C     NEIG1  TOTAL NUMBER OF CONVEGED EIGENVALUES
C
C COMMON /IDATA/ EIGL, EIGR, NEIG1
C.... COMMON RDATA
C
C     RNM    NORM OF THE RESIDUAL VECTOR IN R(NQ(1))
C     RNM2   SQUARE OF RNM
C     SPREAD WIDTH OF THE INTERVAL CONTAINING THE UNCONVERGED
C             EIGENVALUES
C     TOL    TOLERANCE FOR CONVERGENCE OF THE EIGENVALUES
C     EPS    COMPUTER PRECISION
C     EPS1   EPS*SQRT(N)
C     REPS   SQUARE ROOT OF EPS
C
C COMMON /RDATA/ RNM, RNM2, SPREAD, TOL, EPS, EPS1, REPS
C
C COMMON /LANCON/ ZERO, TENTH, EIGHTH, FOURTH, HALF, ONE, TWO,
C                 FOUR, TEN, ONE28, TWO56, FIVE12, ORTFAC, OVRF LW
C
C DATA ZERO, TENTH, EIGHTH, FOURTH, HALF, ONE, TWO, FOUR, TEN,
C 1   ONE28, TWO56, FIVE12, ORTFAC, OVRF LW /
C 2   0.0D0, 0.1D0, 0.125D0, 0.25D0, 0.5D0, 1.0D0, 2.0D0, 4.0D0,
C 3   10.0D0, 128.0D0, 256.0D0, 512.0D0, 4.0D0, 1.7014D+38/
C.... CHECK INPUT DATA
C
C     IERR = 0
C     MT = 6*N + 2*MAXPRS + 10*LANMAX
C     IF (N .LE. 0) IERR = IERR + 1
C     IF (LANMAX .LE. 0 ) IERR = IERR + 2
C     IF (ENDR .LE. ENDL ) IERR = IERR + 4
C     IF (MAXPRS .LE. 0 ) IERR = IERR + 8
C     IF (MAXPRS .GT. LANMAX ) IERR = IERR + 16

```

```

IF ( LANMAX .GT. N )           IERR = IERR + 32
IF ( MT .GT. NW )             IERR = IERR + 64
IF ( IERR .GT. 0 ) RETURN

COMPUTE THE MACHINE PRECISION
EPS = GETEPS(IBETA,IT,IRND)
REPS = SQRT(EPS)
EPS1 = EPS*SQRT(FLOAT(N))

SET POINTERS AND INITIALIZE
M1 = 1 + 6*N
M2 = MAXPRS + M1
M3 = MAXPRS + M2
M4 = LANMAX + M3
M5 = LANMAX + M4
M6 = LANMAX + M5
M7 = LANMAX + M6
M8 = LANMAX + M7
M9 = LANMAX + M8
M10 = LANMAX + M9
M11 = LANMAX + M10

IF MORE STORAGE IS TO BE ALLOCATED, MUST CHANGE "MT" ABOVE
NS = NW - MT + 2*LANMAX
NQ(1) = N + 1
DO 20 I = 2,5
    NQ(I) = NQ(I-1) + N
CONTINUE

CHECK FOR STARTING VECTOR
RNM2 = ZERO
DO 30 I = 1,N
    RNM2 = RNM2 + ABS(W(I))
CONTINUE

IF (RNM2 .EQ. ZERO) THEN
    GET RANDOM STARTING VECTOR
    IRAND = N + LANMAX + MAXPRS + NW
    IRANDOM1=IRAND
    DO 40 I = 1,N
        IRANDOM2=I
        W(I) = RAN(IRANDOM1,IRANDOM2)
    CONTINUE
END IF

CALL DPM(W,W(NQ(3)),N)
RNM2 = DDOT(N,W,1,W(NQ(3)),1)
CALL STPMNE(N,W(M3),W(M4),W(M5),W(M6),W,NQ)
CALL LANSEL(N,LANMAX,MAXPRS,NS,ENDL,ENDR,W,W(M3),W(M4),W(M5),
1          W(M6),EIG,W(M1),W(M2),W(M10),W(M11),W(M7),W(M8),
2          IW,Y,W(M9),NQ,IERR)
IF (IERR.GT.0) THEN
    RETURN
ENDIF
NEIG = NEIG1

RETURN
END

```

Function GETEPS

GETEPS determines the precision of the computer being used. It evaluates the unit roundoff error (i.e., the smallest number, ϵ , in the computer such that $1 + \epsilon > 1$).

```

FUNCTION GETEPS(IBETA, IT, IRND)
IMPLICIT DOUBLE PRECISION (A-H,D-Z)
COMMON /LANCON/ ZERO, TENTH, EIGHTH, FOURTH, HALF, ONE, TWO,
1                   FOUR, TEN, ONE28, TWO56, FIVE12, ORTFAC, OVRFLOW
DETERMINE IBETA, BETA

```

```

10   A = ONE
    A = A + A
    IF (((A + ONE) - A) - ONE .EQ. ZERO) GO TO 10
    B = ONE
20   B = B + B
    IF ((A + B) - A .EQ. ZERO) GO TO 20
    IBETA = INT(SNGL((A + B) - A))
    BETA = FLOAT(IBETA)
C.... DETERMINE IT, IRND
C
30   IT = 0
    B = ONE
    IT = IT + 1
    B = B * BETA
    IF (((B + ONE) - B) - ONE .EQ. ZERO) GO TO 30
    IRND = 0
    BETAM1 = BETA - ONE
    IF ((A + BETAM1) - A .NE. ZERO) IRND = 1
C.... DETERMINE EPS
C
40   BETAIN = ONE / BETA
    A = ONE
    DO 40 I = 1, IT + 3
        A = A * BETAIN
    CONTINUE
C
50   IF ((ONE + A) - ONE .NE. ZERO) GO TO 60
    A = A * BETA
    GO TO 50
60   EPS = A
    IF ((IBETA .EQ. 2) .OR. (IRND .EQ. 0)) GO TO 70
    A = (A*(ONE + A)) / (ONE + ONE)
    IF ((ONE + A) - ONE .NE. ZERO) EPS = A
    GETEPS = EPS
70
C
    RETURN
END

```

Subroutine STPONE

This routine performs the initialization of ALF, BET, ALPH and BET2 as well as the first step of the Lanczos algorithm. See Sec. 10.6.5 for notation. Since $q_{j-1} = 0$ at the first step the algorithm reduces to an orthogonalization of $K_\sigma^{-1}Mq_1$ against q_1 . q_1 is obtained by normalizing r and is stored in the array R starting at location NQ(1). Mr , stored in starting location NQ(3) of R, is then normalized to get Mq_1 which is put in R starting from NQ(4). NQ(2) and NQ(5) are the location of q_{j-1} and a temporary vector in R, respectively.

```

C
C SUBROUTINE STPONE(N,ALF,BET,ALPH,BET2,R,NQ)
C... THIS ROUTINE PERFORMS THE FIRST STEP OF THE LANCZOS ALGORITHM.
C... IT PERFORMS A STEP OF LOCAL REORTHOGONALIZATION IF NEEDED.
C... INPUT/OUTPUT
C
C     N      DIMENSION OF THE EIGENPROBLEM
C     ALF    THE NEW DIAGONAL OF T
C     BET    THE NEW OFF-DIAGONAL OF T
C     ALPH   THE NEW DIAGONAL OF THE DEFLATED T
C     BET2   THE NEW OFF-DIAGONAL SQUARED OF THE DEFLATED T
C     R      AN ARRAY CONTAINING [R(J),Q(J),Q(J-1),P(J),MR(J)]
C     NQ(5)  LOCATION POINTERS FOR THE ARRAY R
C... SUBROUTINES: DAXPY,DCOPY,DDOT,DSCAL,OPK,OPM
C
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C     DIMENSION NQ(1),R(1)
C
C     COMMON /RDATA/ RNM,RNM2,SPREAD,TOL,EPS,EPS1,REPS
C     COMMON /LANCON/ ZERO,TENTH,EIGHTH,FOURTH,HALF,ONE,TWO,
C                      FOUR,TEN,ONE28,TWO56,FIVE12,ORTFAC,ORTFLW
C
C... MODIFY R TO SATISFY THE CORRECT CONDITIONS FOR SINGULAR M
C

```

```

T = ONE/SQRT(RNM2)
CALL DCOPY(N,R(NQ(3)),1,R(NQ(4)),1)
CALL DSCAL(N,T,R(NQ(4)),1)

CALL OPK(R(NQ(4)),R,N)
CALL OPM(R,R(NQ(3)),N)
RNM2 = DDOT(N,R,1,R(NQ(3)),1)
RNM = SQRT(RNM2)

BET2(1) OUGHT TO BE ZERO.
BET(1) STORES THE M-NORM OF THE STARTING VECTOR.

BET2 = ZERO
BET = RNM
T = ONE/RNM
CALL DCOPY(N,R,1,R(NQ(1)),1)
CALL DSCAL(N,T,R(NQ(1)),1)
CALL DCOPY(N,R(NQ(3)),1,R(NQ(4)),1)
CALL DSCAL(N,T,R(NQ(4)),1)
CALL OPK(R(NQ(4)),R,N)

ALF = ZERO
DALF = DDOT(N,R,1,R(NQ(4)),1)
DO 10 I=1,2
  CALL DAXPY(N,-DALF,R(NQ(1)),1,R,1)
  ALF = ALF + DALF
  ALPH = ALF
  CALL OPM(R,R(NQ(3)),N)
  RNM2 = DDOT(N,R,1,R(NQ(3)),1)
  IF (I .EQ. 2) RETURN
  DALF = DDOT(N,R(NQ(1)),1,R(NQ(3)),1)
  DBET = DDOT(N,R(NQ(2)),1,R(NQ(3)),1)
  CALL DAXPY(N,-DBET,R(NQ(2)),1,R,1)
CONTINUE

RETURN
END

```

Subroutine LANSEL

LANSEL, LANCzos algorithm with SELective orthogonalization, performs the main steps of the procedure described in the previous sections. The flowchart of Fig. 10.6.3 gives a global structure of LANSEL. After some initializations the algorithm checks if any orthogonalization is necessary by calling PURGE. Then it calls LANSIM to perform a step of simple Lanczos. The orthogonality estimates and the eigenvalues of the tridiagonal matrix are updated by calling ORTBND and ANALZT, respectively. When enough eigenvalues have been computed loop 10 is terminated and finally the wanted eigenvectors are computed using RITVEC.

LANSEL terminates the iteration if any of the following conditions are satisfied:

- The norm of r_j becomes small (indicating an invariant subspace).
- The number of Lanczos steps reaches LANMAX, the maximum allowed. In this case IERR is set to -1 .
- The desired eigenvalues are computed (see LOGICAL FUNCTION ENOUGH).

If the interval $[ENDL, ENDR]$ contains no eigenvalue, then LANSEL will continue until an eigenvalue has converged outside each end of this interval. If the first MAXPRS eigenvalues are wanted, then set $ENDL = -\infty$, and $ENDR = +\infty$ (i.e., set to very large negative and positive numbers, respectively).

Structure of R: The first vector starting at location 1 is r_j when entering the subroutine. The remaining five vectors are stored at locations NQ(1) through NQ(5).

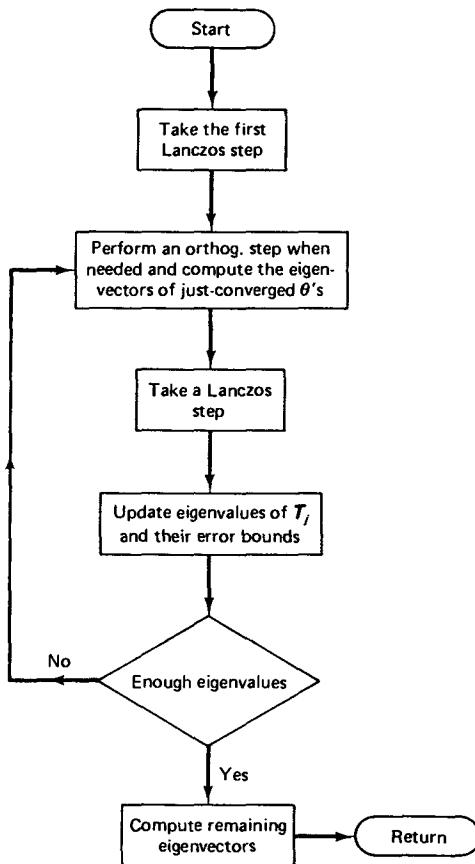


Figure 10.6.3 A global picture of the Lanczos method with selective orthogonalization indicating the way the individual modules are used.

This eliminates the need for moving the content of one array to another for swapping. Only the pointers of the arrays are changed. The vectors q_j , q_{j-1} , Mr_j and Mq_j are held in R starting from locations NQ(1) through NQ(4), respectively. The space in R starting from NQ(5) is used for a working vector in other parts of the program.

```

C      SUBROUTINE LANSEL(N, LANMAX, MAXPRS, NS, ENDL, ENDR, R, ALF, BET, ALPH,
1                      BET2, EIG, TAU, OLDTAU, RHO, WORK, ETA, OLDETA, INFO,
2                      Y, S, NQ, IERR)
C
C..... INPUTS
C
C      N      DIMENSION OF THE EIGENPROBLEM
C      LANMAX UPPER LIMIT TO THE NUMBER OF LANCZOS STEPS
C      MAXPRS UPPER LIMIT TO THE NUMBER OF WANTED EIGENPAIRS
C      NS     LENGTH OF THE ARRAY S
C      ENDL   LEFT END OF THE INTERVAL CONTAINING THE WANTED EIGENVALUES
C      ENDR   RIGHT END OF THE INTERVAL CONTAINING THE WANTED EIGENVALUES
C
C..... WORK SPACE
C
C      R      HOLDS 6 VECTORS OF LENGTH N. SEE THE TEXT FOR DETAILS.
C      NQ(5)  CONTAINS THE POINTERS TO THE BEGINNING OF EACH VECTOR IN R.
C      ALF    ARRAY OF LENGTH LANMAX TO HOLD DIAGONAL OF THE TRIDIAGONAL T
C      BET    ARRAY OF LENGTH LANMAX TO HOLD OFF-DIAGONAL OF T
C      ALPH   DIAGONAL OF THE DEFLATED TRIDIAGONAL
C      BET2   SQUARE OF THE OFF-DIAGONALS OF THE DEFLATED TRIDIAGONAL
C      TAU   ORTHOGONALITY ESTIMATE OF RITZ VECTORS AT STEP J
  
```

```

C      OLDTAU ORTHOGONALITY ESTIMATE OF RITZ VECTORS AT STEP J-1
C      RHO WORKING ARRAY USED IN DEFLAT()
C      ETA ORTHOGONALITY ESTIMATE OF LANCZOS VECTORS AT STEP J
C      OLDETA ORTHOGONALITY ESTIMATE OF LANCZOS VECTORS AT STEP J-1
C      INFO INFORMATION ARRAY ABOUT EIGENVECTORS OF T
C      S ARRAY FOR COMPUTING THE EIGENVECTORS OF THE TRIDIAGONAL
C      WORK WORKING ARRAY TO HOLD SQUARES OF ARRAY BETA

C.... OUTPUTS
C      EIGE ARRAY OF LENGTH MAXPRS TO HOLD THE CONVERGED RITZ VALUES
C      Y ARRAY OF LENGTH MAXPRS*N TO HOLD THE CONVERGED RITZ VECTORS
C      NEIG NUMBER OF COMPUTED EIGENPAIRS
C      IERR ERROR FLAG

C.... SUBROUTINES: ANALZT, ENOUGH, LANSIM, ORTBND, PURGE, RITVEC
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION R(1),Y(N,1),EIG(1),TAU(1),OLDTAU(1),ALF(1),BET(1)
C      DIMENSION S(1),NQ(1),ALPH(1),BET2(1),ETA(1),OLDETA(1),INFO(1)
C      DIMENSION RHO(1),WORK(1)
C      LOGICAL LASTEP,ENOUGH

C      COMMON /RDATA/ RNM,RNM2,SPREAD,TOL,EPS,EPS1,REPS
C      COMMON /IDATA/ EIGL,EIGR,NEIG

C      JJ = 1
C      ETA(1) = EPS1
C      EIGL = ENDL - (ENDR - ENDL)
C      EIGR = ENDL + (ENDR - ENDL)
C      NEIG = 0

C.... LANCZOS LOOP
C
C      DO 10 J = 2,LANMAX
C      NBUF = NS/J
C      RNM = SQRT(RNM2)

C.... RESTORE THE ORTHOGONALITY STATE WHEN NEEDED
C
C      1 CALL PURGE(R,R(NQ(1)),R(NQ(3)),R(NQ(4)),R(NQ(5)),Y,ALF,BET,S,
C                   EIG,ETA,OLDETA,TAU,OLDTAU,WORK,INFO,N,J-1,NBUF,IERR)
C      IF (IERR .GT. 0) RETURN

C.... UPDATE THE RITZ VALUES
C
C      CALL ANALZT(JJ,ALPH,BET2,EIG,TAU,OLDTAU,RHO,INFO)
C
C      IF (ENOUGH(ENDL,ENDR,MAXPRS) ) GO TO 30
C
C      IF (RNM .LT. REPS*SPREAD) GO TO 20
C      JJ = JJ + 1

C.... TAKE A LANCZOS STEP
C
C      1 CALL LANSIM(R,ALF(J),BET(J),ALPH(JJ),BET2(JJ),RNM,RNM2,NQ,
C                   N,J)

C.... UPDATE THE ORTHOGONALITY BOUNDS
C
C      1 CALL ORTBND(ALF,BET,J,EPS1,ETA,OLDETA,TAU,OLDTAU,EIG,INFO,
C                   RNM,NEIG,N)

10      CONTINUE
20      J = J - 1

C.... COMPUTE THE REMAINING RITZ VECTORS
C
30      DO 50 I=1,NEIG
C          M = 1
C          DO 40 K = 1,NEIG
C              IF (INFO(K) .GT. 0) INFO(K) = -INFO(K)
C              M = MIN(ABS(INFO(K)),M)
C
40      CONTINUE
C
C          IF (M .EQ. 0) THEN
C              CALL RITVEC(R,R(NQ(1)),R(NQ(3)),R(NQ(4)),R(NQ(5)),Y,ALF,
C                          BET,EIG,S,INFO,N,J,NEIG,NBUF,.TRUE.,WORK,IERR)
C              IF (IERR .GT. 0) THEN
C                  RETURN
C              ENDIF
C          ELSE
C              GO TO 60
C          END IF
C
50      CONTINUE

```

```

C
60  CONTINUE
IF( J .EQ. LANMAX ) THEN
IERR = -1
END IF
C
RETURN
END

```

Logical Function ENOUGH

ENOUGH determines if all the desired eigenvalues have converged. This is established if one of the following conditions is satisfied:

- The number of computed eigenvalues exceeds the maximum number requested, MAXPRS.
- An eigenvalue is found outside the interval [ENDL, ENDR] at each end.

```

C      LOGICAL FUNCTION ENOUGH(ENDL,ENDR,MAXPRS)
C.... EXAMINE IF ENOUGH EIGENVALUES HAVE CONVERGED
C.... INPUT
C      ENDL    LEFT END OF THE INTERVAL CONTAINING THE WANTED EIGENVALUES
C      ENDR    RIGHT END OF THE INTERVAL CONTAINING THE WANTED EIGENVALUES
C      MAXPRS  UPPER LIMIT TO THE NUMBER OF WANTED EIGENPAIRS
C      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C      PARAMETER (NMAX = 128)
C      COMMON/ATDATA/THET(NMAX), BJ(NMAX), WINDOW, NBD(2), NDST
C      COMMON/IDATA/EIGL, EIGR, NEIG
C      ENOUGH = .TRUE.
C      IF ( NEIG .GT. MAXPRS ) RETURN
C      ENOUGH = ( THET(1) .GT. ENDL .AND. THET(NDST) .LT. ENDR ) .AND.
1      (( EIGL .GT. ENDL .AND. EIGL .LT. ENDR ) .OR.
2      ( EIGR .GT. ENDL .AND. EIGR .LT. ENDR ))
C      RETURN
END

```

Subroutine LANSIM

This routine performs a single step of the simple Lanczos process. The flowchart in Fig. 10.6.4 gives a global view of the algorithm. All the vectors required in this routine are stored in array R.

```

C      SUBROUTINE LANSIM(R,ALF,BET,ALPH,BET2,RNM,RNM2,NQ,N,J)
C.... THIS ROUTINE PERFORMS A SINGLE STEP OF THE LANCZOS ALGORITHM,
C FOLLOWED BY A STEP OF LOCAL REORTHOGONALIZATION IF NEEDED.
C.... INPUT/OUTPUT
C      R      AN ARRAY CONTAINING [R(J),Q(J),Q(J-1),P(J),MR(J)]
C      ALF    THE NEW DIAGONAL OF T
C      BET    THE NEW OFF-DIAGONAL OF T
C      ALPH   THE NEW DIAGONAL OF THE DEFLATED T
C      BET2   THE NEW OFF-DIAGONAL SQUARED OF THE DEFLATED T
C      RNM   NORM OF R(J)
C      RNM2  RNM**2
C      NQ(5) LOCATION POINTERS FOR THE ARRAY R
C      N      DIMENSION OF THE EIGENPROBLEM
C      J      CURRENT LANCZOS STEP
C.... SUBROUTINES: DAXPY,DCOPY,DDOT,DSCAL,DPK,DPM,STORE
C

```

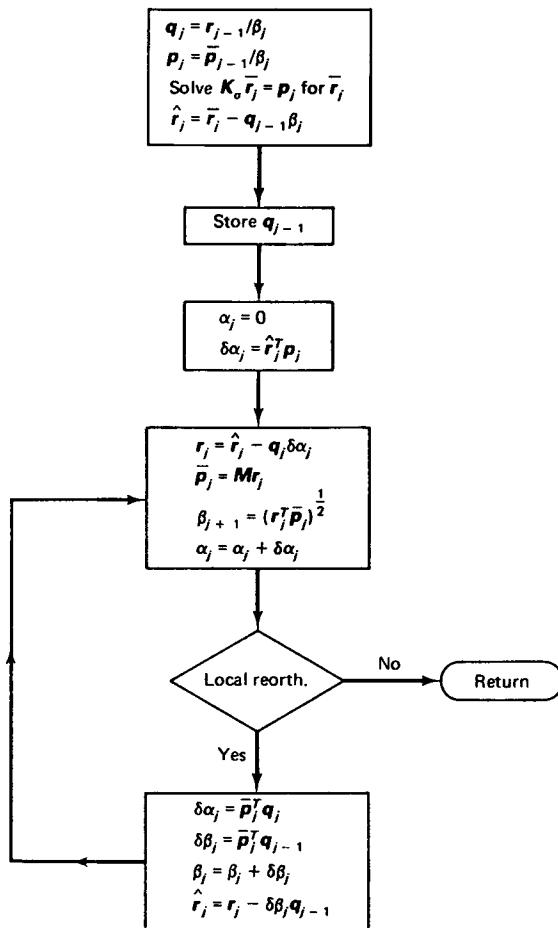


Figure 10.6.4 Flowchart description of the j th step of the simple Lanczos algorithm.

```

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DIMENSION NQ(1),R(1)
1 COMMON /LANCON/ ZERO, TENTH, EIGHTH, FOURTH, HALF, ONE, TWO,
     FOUR, TEN, ONE28, TWO56, FIVE12, ORTFAC, OVRFLW
.. SWAP Q(J) AND Q(J-1)
NTMP = NQ(2)
NQ(2) = NQ(1)
NQ(1) = NTMP
.. Q = R/BETA
T = ONE/RNM
CALL DCOPY(N,R,1,R(NQ(1)),1)
CALL DSCAL(N,T,R(NQ(1)),1)
.. P = PBAR/BETA
CALL DCOPY(N,R(NQ(3)),1,R(NQ(4)),1)
CALL DSCAL(N,T,R(NQ(4)),1)
.. R = ( K INVERSE ) * Q
CALL DPK(R(NQ(4)),R,N)
.. R = R - Q(J-1)*BETA
  
```

```

C      T = -RNM
C      CALL DAXPY(N,T,R(NQ(2)),1,R,1)
C.... STORE Q(J-1)
C      CALL STORE(R(NQ(2)),N,J-1,1)
C.... START LOCAL REORTHOGONALIZATION
C      BET = RNM
C      BET2 = RNM2
C      ALF = ZERO
C.... ALF = ( R TRANSPOSE ) * P
C      DALF = DDOT(N,R,1,R(NQ(4)),1)
C      DO 10 I=1,2
C         CALL DAXPY(N,-DALF,R(NQ(1)),1,R,1)
C         ALF = ALF + DALF
C         CALL OPM(R,R(NQ(3)),N)
C         RNM2 = DDOT(N,R,1,R(NQ(3)),1)
C         ALPH = ALF
C
C         IF (RNM2*ORTFAC .GT. (ALF**2 + BET2) .OR. I .EQ. 2) RETURN
C.... REPEAT LOCAL REORTHOGONALIZATION WHEN WARRANTED
C
C      DALF = DDOT(N,R(NQ(1)),1,R(NQ(3)),1)
C      DBET = DDOT(N,R(NQ(2)),1,R(NQ(3)),1)
C      CALL DAXPY(N,-DBET,R(NQ(2)),1,R,1)
C      BET = BET + DBET
C      BET2 = BET**2
10    CONTINUE
C
END

```

Subroutine ORTBND

The subroutine ORTBND updates the orthogonality bounds, h_j , of (10.6.36), and the τ recurrence of (10.6.40). These quantities are later examined in subroutine PURGE for possible loss of semiorthogonality. LANSEL is the only routine that calls ORTBND.

```

C
1   SUBROUTINE ORTBND (ALF,BET,J,EP1,ETA,OLDETA,TAU,OLDTAU,EIG,INFO,
RNM,NEIG,N)
C.... UPDATE THE ETA AND TAU RECURRENCES.
C.... INPUTS
C
C      ALF(J)      DIAGONAL OF THE TRIDIAGONAL T
C      BET(J)      OFF-DIAGONAL OF T
C      J           DIMENSION OF T
C
C      EP1         ROUNDOFF ESTIMATE FOR DOT PRODUCT OF TWO UNIT VECTORS
C      ETA(J)      ORTHOGONALITY ESTIMATE OF LANCZOS VECTORS AT STEP J
C      OLDETA(J)   ORTHOGONALITY ESTIMATE OF LANCZOS VECTORS AT STEP J-1
C      TAU(NEIG)   ORTHOGONALITY ESTIMATE OF RITZ VECTORS AT STEP J
C      OLDTAU(NEIG) ORTHOGONALITY ESTIMATE OF RITZ VECTORS AT STEP J-1
C      EIG(NEIG)   ARRAY OF CONVERGED EIGENVALUES
C      INFO(NEIG)  INFORMATION ARRAY ABOUT EIGENVECTORS OF T
C      RNM         NORM OF THE NEXT RESIDUAL VECTOR
C      NEIG        NUMBER OF CONVERGED EIGENVALUES
C      N           DIMENSION OF THE EIGENPROBLEM
C
C.... OUTPUTS
C
C      ETA(J)      ORTHOGONALITY ESTIMATE OF LANCZOS VECTORS AT STEP J+1
C      OLDETA(J)   ORTHOGONALITY ESTIMATE OF LANCZOS VECTORS AT STEP J
C      TAU(NEIG)   ORTHOGONALITY ESTIMATE OF RITZ VECTORS AT STEP J+1
C      OLDTAU(NEIG) ORTHOGONALITY ESTIMATE OF RITZ VECTORS AT STEP J
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION ALF(1),BET(1),ETA(1),OLDETA(1),TAU(1),OLDTAU(1)
C      DIMENSION EIG(1),INFO(1)
C      COMMON /LANCON/ ZERO, TENTH, EIGHTH, FOURTH, HALF, ONE, TWO,
C                      FOUR, TEN, ONE28, TWO56, FIVE12, ORTFAC, ORFLW
1

```

```

IF ( J .GT. 1 ) THEN
  OLDETA(i) = ( BET(2)*ETA(2) + (ALF(1) - ALF(J))*ETA(1)
  |           - BET(J)*OLDETA(1) ) / RNM
  J1 = J - 1
  IF ( J1 .GT. 2 ) THEN
    DO 100 K=2,J1
      OLDETA(K) = ( BET(K+1)*ETA(K+1) + (ALF(K) - ALF(J))*ETA(K)
      |           + BET(K)*ETA(K-1) - BET(J)*OLDETA(K) ) / RNM
    CONTINUE
  END IF
  DO 200 K=1,J1
    T = OLDETA(K)
    OLDETA(K) = ETA(K)
    ETA(K) = T
  CONTINUE
END IF
ETA(J) = EPS1*MAX(BET(2)/RNM, ONE)
UPDATE THE TAU RECURRENCE.

DO 300 I=1,NEIG
  IF ( INFO(I) .NE. 0 ) THEN
    T = TAU(I)
    TAU(I) = (EIG(I) - ALF(J))*TAU(I) - BET(J)*OLDTAU(I)
    OLDTAU(I) = T
  END IF
CONTINUE
RETURN
END

```

Subroutine PURGE

PURGE first examines the array ETA, which holds the vector h_{j+1} . See Sec. 10.6.8. If the element of ETA with largest absolute value is less than REPS, $\sqrt{\epsilon}$, indicating no loss of semiorthogonality, then PURGE does nothing and returns. Otherwise, the elements of TAU are examined to determine if loss of orthogonality might be due to the return of a previously banished Ritz vector. If TAU(I) is greater than REPS in absolute value, it indicates loss of semiorthogonality against the Ritz vector with index I. The corresponding eigenvector of T_j is computed using GIVENS. ETA and OLD-ETA are orthogonalized against the computed eigenvector of T_j . See Fig. 10.6.5.

The elements of ETA are examined for a second time. If ETA still holds an element with absolute value greater than REPS, then loss of orthogonality is also due to the convergence of a Ritz value. Then RITVEC is called and the contents of TAU, OLDTAU, ETA, and OLDETA are all set to $\sqrt{n}\epsilon$. Otherwise, q_j and r_j , held in Q and R, are orthogonalized against those Ritz vectors in columns of Y with TAU value greater than REPS in absolute value. These elements of TAU and OLDTAU are then set to EPS1.

All orthogonalizations of the Lanczos vectors are performed with respect to the inertial inner product, and therefore we require two vectors, QA and RA, that hold Mq_j and Mr_j , respectively. At the end of PURGE, Mq_j and Mr_j are recomputed if q_j and r_j are modified.

```

SUBROUTINE PURGE(R,Q,RA,QA,T,Y,ALF,BET,S,EIG,ETA,OLDETA,TAU,
  |           OLDTAU,WORK,INFO,N,J,NBUF,IERR)
THIS ROUTINE EXAMINES ETA, OLDETA, TAU AND OLDTAU TO DECIDE
WHICH FORM OF REORTHOGONALIZATION IF ANY SHOULD BE PERFORMED.

```

INPUT/OUTPUT

| | |
|----|---|
| R | THE RESIDUAL VECTOR TO BECOME THE NEXT LANCZOS VECTOR |
| Q | THE CURRENT LANCZOS VECTOR |
| RA | THE PRODUCT OF THE MASS MATRIX AND R |
| QA | THE PRODUCT OF THE MASS MATRIX AND Q |

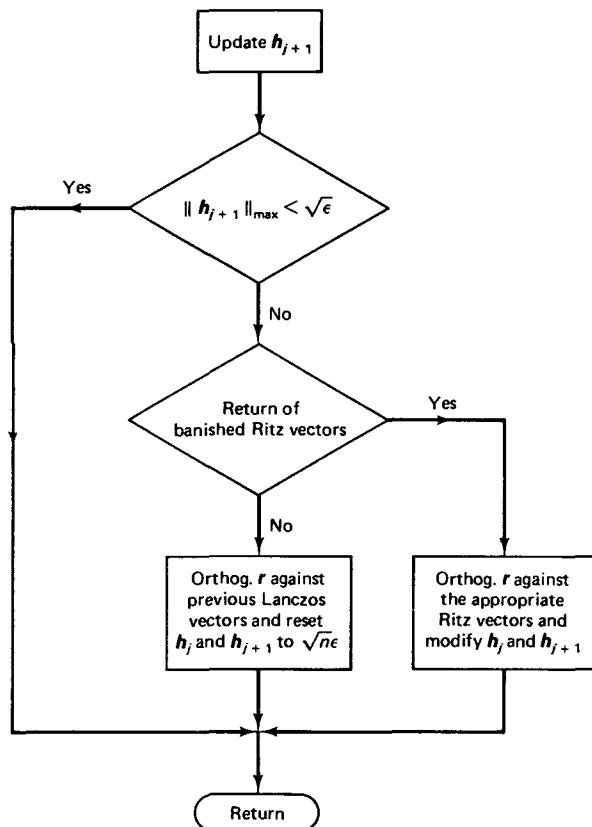


Figure 10.6.5 A flowchart for the selective orthogonalization method.

```

C      T      A TEMPORARY VECTOR TO HOLD THE PREVIOUS LANCZOS VECTORS
C      Y      CONTAINS THE COMPUTED RITZ VECTORS
C      ALF     THE NEW DIAGONAL OF T
C      BET     THE NEW OFF-DIAGONAL OF T
C      S      VECTOR FOR COMPUTING EIGENVECTORS OF T(J)
C      EIG     HOLDS THE CONVERGED RITZ VALUES
C      ETA     STATE OF ORTHOGONALITY BETWEEN R AND PREVIOUS LANCZOS VECTORS
C      OLDETA  STATE OF ORTHOGONALITY BETWEEN Q AND PREVIOUS LANCZOS VECTORS
C      TAU     STATE OF ORTHOGONALITY BETWEEN R AND COMPUTED RITZ VECTORS
C      OLDTAU  STATE OF ORTHOGONALITY BETWEEN Q AND COMPUTED RITZ VECTORS
C      WORK    WORKING ARRAY EXPLAINED IN LANSERL
C      INFO    INFORMATION ABOUT THE EIGENVECTORS OF T(J)
C      N      DIMENSION OF THE EIGENPROBLEM
C      J      CURRENT LANCZOS STEP
C      NEIG   NUMBER OF RITZ VALUES
C      NBUF   NUMBER OF VECTORS IN S
C
C..... SUBROUTINES: DAXPY, DZERO, DDOT, GIVENS, IDAMAX, DPM, RITVEC, SUBTJ
C
C      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C      DIMENSION R(1),Q(1),RA(1),QA(1),T(1),Y(N,1),ALF(1),BET(1),S(J,1)
C      DIMENSION EIG(1),ETA(1),OLDETA(1),TAU(1),OLDTAU(1),INFO(1),WORK(1)
C      LOGICAL ORTHO
C
C      COMMON /IDATA/EIGL,EIGR,NEIG
C      COMMON /RDATA/RNM,RNM2,SPREAD,DUMMY,EPS,EPS1,REPS
C      COMMON /LANCON/ ZERO, TENTH, EIGHTH, FOURTH, HALF, ONE, TWO,
C      1        FOUR, TEN, ONE28, TWO56, FIVE12, ORTFAC, OVRFILW
C
C      ORTHO = .FALSE.
C      TOL = REPS*SPREAD
C      REPS0J = SQRT(EPS/FLDAT(J))
C      K = IDAMAX(J-2,ETA,1)
C      IF (ABS(ETA(K)) .GT. REPS0J) THEN
  
```

```

DO 10 I=1,NEIG
  IF (INFO(I) .LT. 0 .AND. ABS(TAU(I)) .GT. REPSQJ) THEN
    CALL DZERD(J,S,1)
    CALL SUBTJ(ALF,BET,EIG,INFO,TOL,I,N,NEIG,J,L,K,M,
              WORK,IERR)
    IF (IERR .GT. 0) RETURN
    CALL GIVENS(K-M+1,L-M+1,ALF(M),BET(M),EIG(I),EPS,
                S,RESID,RAYCOR,IERR)
    IF (IERR .GT. 0) RETURN
    ZETA = -DDOT(J,ZETA,S,1)
    CALL DAXPY(J,ZETA,S,1,ZETA,1)
    ZETA = -DDOT(J,OLDETA,S,1)
    CALL DAXPY(J,ZETA,S,1,OLDETA,1)
  END IF
CONTINUE
K = IDAMAX(J-2,ETA,1)
IF (ABS(ETA(K)) .GT. REPSQJ) THEN
  GRAM-SCHMID NEEDED
  ORTHO = .TRUE.
  CALL RITVEC(R,Q,RA,QA,T,Y,ALF,BET,EIG,S,INFO,N,J,NEIG,
              NBUF,FALSE.,WORK,IERR)
  IF (IERR.GT.0) RETURN
  DO 20 I=1,NEIG
    TAU(I) = EPS1
    OLDTAU(I) = EPS1
  CONTINUE
  DO 30 I=1,J-1
    ETA(I) = EPS1
    OLDETA(I) = EPS1
  CONTINUE
ELSE
  REMOVE COMPONENTS OF A RITZ VECTOR
  ORTHO = .TRUE.
  DO 40 I = 1,NEIG
    IF (ABS(TAU(I)) .GT. REPSQJ) THEN
      TAU(I) = EPS1
      OLDTAU(I) = EPS1
      ZETA = DDOT(N,RA,1,Y(1,I),1)
      CALL DAXPY(N,-ZETA,Y(1,I),1,R,1)
      ZETA = DDOT(N,QA,1,Y(1,I),1)
      CALL DAXPY(N,-ZETA,Y(1,I),1,Q,1)
    END IF
  CONTINUE
END IF
END IF
IF (ORTHO) THEN
  CALL OPM(Q,QA,N)
  QNORM = SQRT(DDOT(N,Q,1,QA,1))
  CALL DSCAL(N,ONE/QNORM,Q,1)
  CALL DSCAL(N,ONE/QNORM,QA,1)
  BET(J) = BET(J) * QNORM
  ZETA = DDOT(N,R,1,QA,1)
  ALF(J) = ALF(J) + ZETA
  CALL DAXPY(N,-ZETA,Q,1,R,1)
  CALL OPM(R,RA,N)
  RMN2 = DDOT(N,R,1,RA,1)
  RMN = SQRT(RMN2)
END IF
RETURN
END

```

Subroutine RITVEC

The purpose of this routine is twofold:

1. Compute the Ritz vector corresponding to a converged Ritz value.
2. Perform a full reorthogonalization step.

The first step of the program is to compute some eigenvectors of the tridiagonal T_j . The eigenvectors of T_j are stored in an array S, and the number of computed vectors

depends on the available storage indicated by NBUF. GIVENS is used to obtain the wanted eigenvectors. The next step is to recall all the Lanczos vectors from secondary storage (by calling STORE) and compute the Ritz vectors using (10.6.23), accumulating the result in columns of Y. The old Lanczos vector that is brought in is stored in the temporary vector T. In the same loop, when EVONLY is false, the two current Lanczos vectors, q_j and r_j , held in Q and R, are orthogonalized against the previous Lanczos vectors held in T. In this way the number of I/O transfers is minimized. Arrays QA and RA that hold Mq_j and Mr_j , respectively, are also needed by RITVEC to perform the orthogonalization with respect to the inertial inner product.

```

C      SUBROUTINE RITVEC(R,Q,RA,QA,T,Y,ALF,BET,EIG,S,INFO,N,J,NEIG,NBUF,
1      EVONLY,WORK,IERR)
C..... THIS ROUTINE COMPUTES SOME RITZ VECTORS AND PERFORMS A
C..... REORTHOGONALIZATION OF THE LANCZOS VECTORS.
C..... INPUT/OUTPUT
C
C      R      THE RESIDUAL VECTOR TO BECOME THE NEXT LANCZOS VECTOR
C      Q      THE CURRENT LANCZOS VECTOR
C      RA     THE PRODUCT OF THE MASS MATRIX AND R
C      QA     THE PRODUCT OF THE MASS MATRIX AND Q
C      T      A TEMPORARY VECTOR TO HOLD THE PREVIOUS LANCZOS VECTORS
C      Y      CONTAINS THE COMPUTED RITZ VECTORS
C      ALF    THE NEW DIAGONAL OF T
C      BET    THE NEW OFF-DIAGONAL OF T
C      EIG    HOLDS THE CONVERGED RITZ VALUES
C      S      VECTOR FOR COMPUTING EIGENVECTORS OF T(J)
C      INFO   INFORMATION ABOUT THE EIGENVECTORS OF T(J)
C      N      DIMENSION OF THE EIGENPROBLEM
C      J      CURRENT LANCZOS STEP
C      NEIG   NUMBER OF RITZ VALUES
C      NBUF   NUMBER OF VECTORS IN S
C      EVONLY IF .TRUE. NO REORTHO. IS PERFORMED, COMPUTES ONLY RITZ VECTORS
C..... SUBROUTINES: DAXPY,DDOT,DZERO,GIVENS,IDAMAX,NUMLES,STORE
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION R(1),Q(1),RA(1),QA(1),T(1),Y(N,1),ALF(1),BET(1),S(J,1)
C      DIMENSION EIG(1),INFO(1),WORK(1)
C      LOGICAL EVONLY
C
C      COMMON /RDATA/ RNM,RNM2,SPREAD,DUMMY,EPS,EPS1,REPS
C      COMMON /LANCON/ ZERO,TENTH,EIGHTH,FOURTH,HALF,ONE,TWO,
C1      FOUR,TEN,ONE28,TWO56,FIVE12,ORTFAC,ORTFLW
C
C      EPS14 = EPS**1/4
C      TOL14 = EPS**1/4*SPREAD
C      TOL34 = EPS**3/4*SPREAD
C
C      EPS14 = SQRT(REPS)
C      TOL14 = EPS14*SPREAD
C      TOL34 = REPS*TOL14
C
C      IBUF = 0
C      TOL = REPS*SPREAD
C      RNEPS = RNM*EPS
C
C..... COMPUTE EIGENVECTORS OF T AND PUT IN THE BUFFER.
C
C      DO 30 I=NEIG,1,-1
C          IF (INFO(I).GE.0 .AND. IBUF.LT.NBUF) THEN
C              IF (EVONLY .AND. INFO(I).NE.0) GO TO 30
C              IBUF = IBUF + 1
C              CALL DZERO(N,Y(1,I),1)
C
C..... EIG(I) ISOLATED
C
C      M = 1
C      L = J
C      K = M
C      CALL DZERO(J,S(1,IBUF),1)
C      1      CALL GIVENS(K-M+1,L-M+1,ALF(M),BET(M),EIG(I),EPS1,
C                  S(M,IBUF),RESID,RAYCR,IERR)
C      IF (IERR.GT.0) GO TO 900
C      IF (RESID.LE.TOL14) GO TO 15
C

```

```

CHECK THAT EIG(I) IS AN EIGENVALUE OF T
DO 5 IDUM = M, L
  WORK(IDUM) = BET(IDUM)*BET(IDUM)
CONTINUE
ZETA = EIG(I)*(ONE - EPS14)
NUL = NUMLES(ALF(M),WORK(M),ZETA,L-M+1,1,EPS)
ZETA = EIG(I)*(ONE + EPS14)
NUR = NUMLES(ALF(M),WORK(M),ZETA,L-M+1,1,EPS)

EIG(I) IS NOT AN EIGENVALUE OF T, GOODBYE.

IF (NUR .EQ. NUL) THEN
  IERR = IERR + 1024
  GO TO 900
ENDIF

EIG(I) IS O.K., NOW FIND A GOOD K

DO 10 K=M+1,L
  CALL DZERO(J,S(1,IBUF),1)
  CALL GIVENS(K-M+1,L-M+1,ALF(M),BET(M),EIG(I),EPS1,
              S(M,IBUF),RESID,RAYCOR,IERR)
  IF (IERR.GT.0) GO TO 900
  IF (RESID.LE.TOL14) GO TO 15
CONTINUE

NO SUITABLE K WAS FOUND, GOODBYE.

IERR = IERR + 512
GO TO 900

NOW WE CAN REFINE THE EIGENVECTOR

DO 20 LOOP = 1,10
  K = IDAMAX(L-M+1,S(M,IBUF),1)+M-1
  CALL DZERO(J,S(1,IBUF),1)
  CALL GIVENS(K-M+1,L-M+1,ALF(M),BET(M),EIG(I),EPS1,
              S(M,IBUF),RESID,RAYCOR,IERR)
  IF (RESID.LE.TOL) GO TO 25
  EIG(I) = EIG(I) + RAYCOR
CONTINUE

RESID FAILED TO DIMINISH, GOODBYE.

IF (RESID .GT. TOL) THEN
  IERR = IERR + 2048
  GO TO 900
ENDIF

INFO(I) = N*IDAMAX(J,S(1,IBUF),1)+J-1
END IF
CONTINUE

COMPUTE THE RITZ VECTORS AND PERFORM G-S ORTHOGONALIZATION.

DO 50 I=1,J-1
  . RETRIEVING THE LANCZOS VECTOR AND PUT IT IN T
  CALL STORE(T,N,I,2)
  KBUF = 0
  DO 40 K=NEIG,1,-1
    IF (INFO(K) .GE. 0 .AND. KBUF .LT. IBUF) THEN
      KBUF = KBUF + 1
      SI = S(I,KBUF)
      IF (ABS(SI) .GT. EPS) CALL DAXPY(N,SI,T,1,Y(1,K),1)
    END IF
  CONTINUE
  IF (.NOT. EVONLY) THEN
    ZETOLD = -DDOT(N,QA,1,T,1)
    IF ( ABS(ZETOLD) .GT. EPS ) CALL DAXPY(N,ZETOLD,T,1,Q,1)
    ZETA = -DDOT(N,RA,1,T,1)
    IF ( ABS(ZETA) .GT. RNEPS ) CALL DAXPY(N,ZETA,T,1,R,1)
  END IF
CONTINUE

ADD IN CONTRIBUTION OF QJ TO Y

KBUF = 0
DO 60 I=NEIG,1,-1
  IF (INFO(I) .GE. 0 .AND. KBUF .LT. IBUF) THEN
    KBUF = KBUF + 1
    IF (ABS(S(J,KBUF)) .LT. EPS) INFO(I) = -INFO(I)
    CALL DAXPY(N,S(J,KBUF),Q,1,Y(1,I),1)
  END IF

```

```
60  CONTINUE
C
900 RETURN
END
```

Subroutine GIVENS

This routine computes the eigenvector of a tridiagonal corresponding to an eigenvalue stored in THET. It first assumes a value for the bottom element of the eigenvector and solves for the rest using the Givens recurrence [28] going backward. This phase is terminated once the Kth element of S has been obtained. Then all the terms computed thus far are scaled to make S(K) unity. A similar procedure is performed starting from the top element and running the recurrence in the forward direction. Then the computed vector is normalized, and finally the residual and the Rayleigh correction to THETA are computed and returned in RES and COR, respectively. The algorithm scales S to avoid overflow whenever it becomes necessary.

```
C      SUBROUTINE GIVENS(K,J,ALF,BET,THET,EPS,S,RES,COR,IERR)
C.... GIVENS RECURRENCE FOR COMPUTING EIGENVECTORS OF A TRIDIAGONAL T.
C.... INPUTS
C      K      INDEX OF THE RIGHT HAND SIDE E(K)
C      J      DIMENSION OF THE TRIDIAGONAL MATRIX
C      ALF(J)  DIAGONALS OF T
C      BET(J)  OFF-DIAGONALS OF T
C      THET   EIGENVALUE OF T
C      EPS    COMPUTER PRECISION
C.... OUTPUTS
C      S(J)   COMPUTED EIGENVECTOR
C      RES   NORM OF THE RESIDUAL
C      COR   RAYLEIGH CORRECTION FOR THET
C.... SUBROUTINES : DSCAL
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION ALF(J),BET(J),S(J)
C.... OVRFLW IS THE MACHINE OVERFLOW THRESHOLD
C      COMMON /LANCON/ ZERO, TENTH, EIGHTH, FOURTH, HALF, ONE, TWO,
C      1        FOUR, TEN, ONE28, TWO56, FIVE12, ORTFAC, OVRFLW
C.... BACKWARD RECURRENCE
C      BIG = SQRT(OVRFLW/FLOAT(J+1))
C      RES = ZERO
C
C      S(J) = ONE
C      SUM2 = ONE
C      IF ( K .LT. J ) THEN
C          S(J-1) = -(ALF(J) - THET)*S(J)/BET(J)
C          SUM2 = SUM2 + S(J-1)**2
C      END IF
C      DO 10 I = J-1,K+1,-1
C          S(I-1) = -( (ALF(I) - THET)*S(I) + BET(I+1)*S(I+1))/BET(I)
C
C.... SCALE TO AVOID OVERFLOW
C
C      IF ( ABS(S(I-1)) .GT. BIG ) THEN
C          F = ONE/S(I-1)
C          S(I-1) = ONE
C          CALL DSCAL(J-I+1,F,S(I),1)
C          SUM2 = (SUM2*F)*F
C      END IF
C      SUM2 = SUM2 + S(I-1)**2
10    CONTINUE
C
C.... STOP EXECUTION GRACEFULLY IF S(K) IS EXACTLY ZERO
C
C      IF ( S(K) .EQ. ZERO) THEN
C          IERR = IERR + 256
```

```

      RETURN
ENDIF
F = ONE/S(K)
S(K) = ONE
SUM2 = (SUM2*F)*F
CALL DSCAL(J-K,F,S(K+1),1)
IF ( K .LE. 1 ) GO TO 30

FORWARD RECURRENCE

X = ZERO
S(1) = ONE
SUM1 = ONE
IF ( K .GT. 2 ) THEN
  S(2) = -((ALF(1) - THET)*S(1))/BET(2)
  SUM1 = SUM1 + S(2)**2
  DO 20 I = 2, K-2
    S(I+1) = -((ALF(I) - THET)*S(I) + BET(I)*S(I-1))/BET(I+1)
    IF ( ABS(S(I+1)) .GT. BIG ) THEN
      F = ONE/S(I+1)
      S(I+1) = ONE
      CALL DSCAL(I,F,S,1)
      SUM1 = (SUM1*F)*F
    END IF
    SUM1 = SUM1 + S(I+1)**2
  CONTINUE
  X = BET(K-1)*S(K-2)
END IF
X = -(X + (ALF(K-1) - THET)*S(K-1))/BET(K)

MATCH X WITH S(K)
IF ( X .EQ. ZERO ) THEN
  RES = ONE
  RETURN
END IF
F = S(K)/X
CALL DSCAL(K-1,F,S,1)
SUM2 = SUM2 + SUM1*F**2
RES = BET(K)*S(K-1)

NORMALIZE S

F = ONE/SQRT(SUM2)
CALL DSCAL(J,F,S,1)
RES = RES*F + (ALF(K) - THET)*S(K)
IF ( K .LT. J ) RES = RES + BET(K+1)*S(K+1)
COR = S(K)*RES
RES = ABS(RES)

RETURN
END

```

Subroutine SUBTJ

In theory, unreduced tridiagonal matrices do not have equal eigenvalues, but in practice they can have two eigenvalues that are so close to one another that they are indistinguishable in finite precision. This creates certain difficulties when computing eigenvectors with close eigenvalues. The problem is solved by working with a submatrix $T_{l,m}$ of the tridiagonal T_j . This routine obtains an estimate of the indices l and m that defines a submatrix for which EIG(I) is simple. INFO(I) = N × K + J stores the two indices J and K. J is the step at which the eigenvector corresponding to EIG(I) was computed and K is the index of the element of this eigenvector with largest absolute value. When INFO(I) is negative it indicates that the Ritz vector in Y(I) is converged.

```
SUBROUTINE SUBTJ(ALF,BET,EIG,INFO,TOL,I,N,NEIG,J,L,K,M,
, U,IERR)
```

THIS ROUTINE SCANS BACK THROUGH THE CONVERGED EIGENVALUES
FOR COPIES OF EIG(I) TO DETERMINE THE SUBMATRIX T(M,L) AND
THE RIGHT HAND SIDE E(K) FOR THE GIVENS RECURRENCE.

INPUTS

```

C      EIG(NEIG)      LIST OF THE CONVERGED EIGENVALUES
C      INFO(NEIG)     INFORMATION ABOUT THE EIGENVECTORS
C      TOL           TOLERANCE FOR FINDING COPIES OF EIG(I)
C      I              INDEX OF THE EIGENVALUE CONSIDERED
C      N              DIMENSION OF THE EIGENPROBLEM
C      NEIG          NUMBER OF EIGENVALUES IN EIG
C.... OUTPUTS
C      K              INDEX OF THE RIGHT HAND SIDE FOR GIVENS
C      L              INDEX OF THE LAST ELEMENT OF THE SUBMATRIX
C      M              INDEX OF THE FIRST ELEMENT OF THE SUBMATRIX
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION ALF(1),BET(1),EIG(1),INFO(1),U(1)
C
C      1 COMMON /LANCON/ ZERO, TENTH, EIGHTH, FOURTH, HALF, ONE, TWO,
C                         FOUR, TEN, ONE28, TWO56, FIVE12, ORTFAC, OVRFLW
C
C      L = J
C      EIGI = EIG(I)
C.... CHECK SPECIAL CASE OF EIGI = ALL ALTERNATE ALF'S
C
C      DO 10 JM = J,1,-2
C         IF (ABS(ALF(JM))-EIGI) .GE. EPS*SPREAD) GO TO 20
10    CONTINUE
C.... FALSE RITZ VALUE ACCEPTED, STOP EXECUTION GRACEFULLY.
C
C      IERR = IERR + 2048
C      RETURN
C.... RUN RECURRENCE UNTIL "PIVOT = DIAGONAL" TO FIND M
C
20    U(JM) = (ALF(JM)-EIGI)*(ONE - TEN*EPS)
C      UMIN = ABS(U(JM))
C      IUMIN = JM
C      DO 30 M = JM-1,1,-1
C         U(M) = ALF(M)-EIGI-BET(M+1)**2/U(M+1)
C         IF (ABS(U(M)).LT.TOL*EIGHTH) GO TO 40
C         IF (ABS(U(M)).LE.UMIN) THEN
C            UMIN = ABS(U(M))
C            IUMIN = M
C         ENDIF
30    CONTINUE
C.... TEMPORARILY
C
C      M = IUMIN
C.... NOW SET K
C
40    IF (M .EQ. 1) THEN
C         K = M
C     ELSE
C         K = M+1
C     ENDIF
C.... NOW CHECK FOR CLOSE EIGENVALUES
C
C      IF (M .GT. 1) THEN
C         DO 50 I1 = I-1,1,-1
C            IF (ABS(EIGI-EIG(I1)).LT.ONE28*EPS*SPREAD)
1          GO TO 60
50    CONTINUE
C
C      RETURN
C
60    KPREV = MOD(ABS(INFO(I1)),N)
C      IF (KPREV .LE. K) K = KPREV+1
C    ENDIF
C.... FIND L
C
C      DO 200 I2 = I+1,NEIG
C         IF (ABS(EIGI - EIG(I2)) .LT. TOL) GO TO 210
200  CONTINUE
210  IF (I2 .LE. NEIG) THEN
C         L = ABS(INFO(I2))/N - 1
C         IF (INFO(I2) .EQ. 0) L = MOD(ABS(INFO(I)),N)
C     END IF
C
C      RETURN
END

```

Subroutine ANALZT

The goal of this routine is to obtain the smallest possible interval that contains a single eigenvalue of T_j (see [24] for more details). It makes full use of the corresponding information that was obtained at the previous step. When $J = 2$ it simply computes the eigenvalues of the 2×2 tridiagonal matrix. Phase I of the routine updates the data structure (THETA, BJ) containing those eigenvalues of T_{j-1} that are about to converge to eigenvalues of the big problem. Their residual bounds go in BJ. Phase II of the routine appends some new items to this data structure. Any converged eigenvalue is removed from THETA, put into EIG, and then explicitly deflated from the tridiagonal. This routine is called by LANSEL. For a more detailed description see [24].

```

SUBROUTINE ANALZT(J,ALF,BET2,EIG,TAU,OLDTAU,RHO,INFO)
. THIS ROUTINE UPDATES SOME EIGENVALUES OF A TRIDIAGONAL T(J) USING
. THE EIGENVALUES OF T(J-1).
. INPUTS
J          ORDER OF THE TRIDIAGONAL T.
ALF         DIAGONAL OF T.
BET2        SQUARES OF THE OFFDIAGONAL TERMS, BET2(1) = ZERO
EIG         ARRAY OF CONVERGED EIGENVALUES
TAU         ORTHOGONALITY ESTIMATE OF RITZ VECTORS AT STEP J
OLDTAU      ORTHOGONALITY ESTIMATE OF RITZ VECTORS AT STEP J-1
RHO         WORKING ARRAY USED IN DEFLAT
INFO        INFORMATION ARRAY ABOUT EIGENVECTORS OF T

. INTERNAL VARIABLES
THET        EXTERIOR EIGENVALUES OF T, NEARLY CONVERGED
.           RITZ VALUES. THET(1)=LEFTMOST, THET(NDST)=RIGHTMOST.
NDST        SIZE OF THET AND BJ
BJ          ERROR BOUND ON THET
.           BJ(I) IS SET TO -1 IF THET(I) DISAPPEARS.
NBD         CONTAINS L AND R IN THE TEXT.
SPREAD      THET(NDST) - THET(1)
EPS         PRECISION OF ARITHMETIC OPERATIONS
IP          IP = 1 FOR UPDATING LEFT END, IP = 2 FOR THE RIGHT END.
INC         INC=1 FOR UPDATING LEFT END, INC=-1 FOR THE RIGHT END.
IS          STARTING INDEX (EITHER 1 OR NDST)
START       LEFT BOUND ON EIGENVALUES (INC=1), RIGHT BOUND (INC=-1)
PROBE       THE OUTER END OF THE NEXT SUBINTERVAL TO BE UPDATED.
INDXOK     TRUE, IF THERE ARE I-INC RITZ VALUES EXTERIOR TO THE
.           NEW THET(I).

. SUBROUTINES: DEFLAT,MOVE1,NEWCOR,NUMLES
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
PARAMETER (NMAX = 128)

DIMENSION ALF(1),BET2(1),EIG(1),TAU(1),OLDTAU(1),RHO(1),INFO(1)
LOGICAL INSERT,INDXOK,APPEND

COMMON /RDATA/ RNM,RNM2,SPREAD,TOL,EPS,EPS1,REPS
COMMON /ATDATA/ THET(NMAX),BJ(NMAX),WINDOW,NBD(2),NDST
COMMON /IDATA/ EIGL,EIGR,NEIG
COMMON /LANCON/ ZERO,TENTH,EIGHTH,FOURTH,HALF,ONE,TWO,
.             FOUR,TEN,ONE28,TW056,FIVE12,BRTFAC,OVRFW
1   IF (J.LE.1) RETURN
IF (J.EQ.2) THEN
  NDST = 16
  WINDOW = FOURTH/SQRT(REPS)
  THET(1) = (ALF(1) + ALF(2) - SQRT(FOUR*BET2(2) +
.           (ALF(1) - ALF(2))*2))*HALF
  THET(NDST) = ALF(1) + ALF(2) - THET(1)
  BJ(1) = SQRT(RNM2/(ONE + BET2(2)/(THET(1) - ALF(1))*2))
  BJ(NDST) = SQRT(RNM2/(ONE + BET2(2)/(THET(NDST) - ALF(1))*2))
  NBD(1) = 1
  NBD(2) = NDST
  SPREAD = THET(NDST) - THET(1)
  RETURN
END IF
SPREAD = THET(NDST) - THET(1)

```

```

TOL = TWO*REPS*SPREAD
W = WINDOW*TOL
C.... BEGIN PHASE 1.
C.... LOOP FOR LEFT END, THEN RIGHT
C
DO 100 IP = 1,2
INC = 3 - 2*IP
IS = (NDST-1)*IP - (NDST-2)
I = IS
INSERT = FALSE
1 START = (THET(I) + ALF(J) - INC*SQRT(BET2(J)*FOUR +
      ALF(J) - THET(I)**2))*HALF
PROBE = THET(I) - INC*BJ(I)
INDEXOK = NUMLES(ALF,BET2,PROBE,J,INC,EPS) .EQ. 0
C
DO 50 IDUMMY =1,NDST
C
      IF (I-NBD(IP)).EQ.INC) GO TO 100
C.... EXAMINE I-TH SUBINTERVAL
C
      IF (INDEXOK) THEN
          IF (INSERT) THEN
              START = THET(I)
              THET(I) = START + INC*MIN(B**2/
      1           ABS(START-THET(I-INC)),B)
          ELSE
              IF (INT(SIGN(ONE,PROBE-START)).EQ.INC) START = PROBE
          END IF
C.... CHECK FOR DISJOINT SUBINTERVALS
C
      IF (I .EQ. NBD(IP)) THEN
          PROBE=THET(I)+INC*(THET(NBD(2))-THET(NBD(1)))/(FOUR*j)
      ELSE
          PROBE = THET(I+INC) - INC*BJ(I+INC)
      END IF
      IF (INT(SIGN(ONE,PROBE-THET(I))).EQ.INC) THEN
          CHECK FOR AN EXTRA RITZ VALUE
          K = NUMLES(ALF,BET2,PROBE,J,INC,EPS)
          IF (K.LT. ABS(I-IS+INC)) THEN
              THET(I) DISAPPEARS
              BJ(I) = -ONE
          ELSE
              RECORD INDEXOK FOR NEXT LOOP. USE REFINED BOUNDS.
          C
              IF (.NOT.INSERT) THEN
                  B = BJ(I)
                  INDEXOK = (K .LE. ABS(I-IS+INC))
                  BND = MIN(B**2/ABS(PROBE-THET(I)),B)
                  IF (INDEXOK.AND.BND.LT.ABS(THET(I)-START)) THEN
                      START = THET(I) - INC*BND
                  END IF
              END IF
          END IF
      ELSE
          PREPARE FOR AN INTRUDING RITZ VALUE
          1 IF ((IS.EQ.NBD(IP)).OR.BJ(NBD(IP)-INC).LT.W).AND.
              NBD(2)-NBD(1).GT.1) NBD(IP) = NBD(IP) + INC
          CALL MOVE1(THET,I,NBD(IP),-INC,PROBE)
          CALL MOVE1(BJ,I,NBD(IP),-INC,TWO*TOL)
          INSERT = .TRUE.
          INDEXOK = .TRUE.
      END IF
      IF (BJ(I).GT.TOL) THEN
C.... USE NEWTON ITERATION TO FIND NEW THET(I)
          CALL NEWCOR(ALF,BET2,START,THET,BJ,INC,I,J,NDST)
      END IF
      IF (BJ(I).LT.0) THEN

```

```

THET(I) DISAPPEARS
CALL MOVE1(THET,NBD(IP),I,INC,ZERO)
CALL MOVE1(BJ,NBD(IP),I,INC,ZERO)
NBD(IP) = NBD(IP) - INC
INSERT = .FALSE.
INDEXOK = .TRUE.

      I = I - INC
END IF
I = I + INC
CONTINUE
. END OF PHASE 1
CONTINUE
. BEGIN PHASE 2.
. APPEND MORE RITZ VALUES AND CHECK FOR CONVERGED RITZ VALUES.

DO 200 IP = 1,2
  INC = 3 - 2*IP
  IS = (NDST-1)*IP - (NDST-2)
  I = IS
  DO 150 IDUMMY = 1,J
    NREM = J - NBD(1) - ((NDST+1) - NBD(2))
    IF ((I-NBD(IP))*INC.GT.0) GO TO 200
    APPEND = I.EQ.NBD(IP).AND.(BJ(I).LT.W.OR.(J.EQ.4
1           .AND.NBD(IP).EQ.IS)).AND.NREM.GT.0
    IF (APPEND) THEN
      START = THET(I) + FLOAT(INC)*BJ(I)
      PROBE = INC*(THET(NBD(2)) - THET(NBD(1)))/NREM
    END IF
    IF (BJ(I) .LE. TOL) THEN
      APPLY QR ALGOR. TO DEFLATE
      CALL DEFLAT(ALF,BET2,THET(I),J)
      INSERT THET(I) INTO EIG
      NEIG = NEIG + 1
      IF (IP.EQ.1) THEN
        EIGL = MAX(EIGL,THET(I))
      ELSE
        EIGR = MIN(EIGR,THET(I))
      END IF
      EIG(NEIG) = THET(I)
      INFO(NEIG) = 0
      REMOVE STABILIZED RITZ VALUES
      CALL MOVE1(THET,NBD(IP),I,INC,ZERO)
      CALL MOVE1(BJ,NBD(IP),I,INC,ZERO)
      NBD(IP) = NBD(IP) - INC
      I = I - INC
    END IF
    IF (APPEND.AND.NBD(2)-NBD(1).GT.1) THEN
      T = START + PROBE
      NBD(IP) = NBD(IP) + INC
      IK = ABS(IS - NBD(IP))
      DO 110 IDUM = 1,J
        IF (NUMLES(ALF,BET2,T,J,INC,EPS).NE.IK) GO TO 120
        T = T + PROBE
      CONTINUE
      THET(NBD(IP)) = T
      START = T - PROBE
      CALL NEWCOR(ALF,BET2,START,THET,BJ,INC,NBD(IP),J,NDST)

    END IF
    IF (J.GT.NDST.AND.I.EQ.NBD(IP).AND.I.NE.IS.AND.BJ(I).GT.
      BJ(I-INC).AND.BJ(I-INC).GT.W) NBD(IP) = NBD(IP) - INC
    I = I + INC
  CONTINUE
CONTINUE

RE-ESTABLISH AN END MARKER, IF NECESSARY, AT EARLY STAGE
DO 300 IP = 1,2
  INC = 3 - 2*IP
  IS = (NDST-1)*IP - (NDST-2)
  IF (NBD(IP).EQ.IS-INC) THEN
    THET(IS) = THET(NBD(3-IP))
    BJ(IS) = BJ(NBD(3-IP))

```

```

      NBD(IP) = IS
      NBD(3-IP) = NBD(3-IP) + INC
  END IF
300  CONTINUE
C
      RETURN
END

```

Subroutine NEWCOR

The object of this routine is to find an eigenvalue of T_j in a given interval. It uses a combination of bisection and Newton's method. For a detailed description see [24]. INDEX points to the position in THETA that will eventually hold the computed eigenvalue. On entry THETA(INDEX) contains the eigenvalue of T_{j-1} that is also one end of the interval; ZETA is the other. ZETA is also the starting value for Newton's method. If this interval is too large then the algorithm performs a few steps of bisection to obtain a better estimate, ZETA, to the desired eigenvalue. The eigenvalue counts are obtained by calling NUMLES. Then Newton's iteration is used to compute the eigenvalue to the full computer precision. The Newton correction is computed using a recurrence that is given in [24]. The convergence of Newton's method is guaranteed by deflating the eigenvalues exterior to ZETA implicitly. NEWCOR is called only by ANALZT.

```

C
      SUBROUTINE NEWCOR(ALF,BET2,ZETA,THET,BJ,INC,indx,J,NDST)
C.... COMPUTES EXTERIOR EIGENVALUES OF A TRIDIAGONAL USING A
C..... COMBINATION OF BISECTIONS AND NEWTON'S METHOD
C..... INPUT
C
      ALF      DIAGONAL OF T
      BET2     SQUARES OF THE OFFDIAGONAL TERMS, BET2(1) = ZERO
      SPREAD   THET(NDST) - THET(1)
      INC      INC=1 FOR UPDATING LEFT END, INC=-1 FOR THE RIGHTEND.
      indx     INDEX OF TO-BE-UPDATED THET.
      J        ORDER OF THE TRIDIAGONAL T.
      NDST    SIZE OF THET AND BJ
C..... INPUT/OUTPUT
C
      ZETA     EXTERIOR BOUND FOR EIGENVALUE OF T IN THET[INDX]
      THET     EXTERIOR EIGENVALUES OF T NEARLY CONVERGED RITZ VALUES;
C           THET(1)=LEFTMOST, THET(NDST)=RIGHTMOST.
      BJ       ERROR BOUND ON THET
C
      SUBROUTINES: NUMLES,QLBOT
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      PARAMETER (MAXBIS = 15, MAXNEW = 40)
      DIMENSION ALF(1),BET2(1),THET(1),BJ(1)
C
      COMMON /RDATA/ RNM,RNM2,SPREAD,TOL,EPS,EPS1,REPS
      COMMON /LANCON/ ZERO, TENTH, EIGHTH, FOURTH, HALF, ONE, TWO,
C                      FOUR, TEN, ONE28, TWO56, FIVE12, ORTFAC, OVRFLW
C
      1      ZOLD = ZETA
      IF (J .EQ. 1) THEN
          ZETA = THET(indx)
          THET(indx) = ALF(J)
          BJ(indx) = SQRT(RNM2)
          RETURN
      END IF
C..... PERFORM BISECTION FOR AN IMPROVED ZETA
C
      IS = ((NDST+1) - (NDST-1)*INC)/2
      FACT = FIVE12*FLOAT(J)*LOG(FLOAT(J))
      WIDTH = (THET(indx) - ZETA)*HALF
      IF (WIDTH .EQ. ZERO) THEN
          WIDTH = BJ(indx)*HALF
      ENDIF
      IOLD = ABS(IS - INDX)

```

```

DO 10 IDUMMY = 1,MAXBIS
  IF (ABS(WIDTH)*FACT .LE. ABS(THET((NDST+1)-IS)-THET(INDX)))
1   GO TO 20
  ZNEW = ZETA + WIDTH
  INEW = NUMLES(ALF,BET2,ZNEW,J,INC,EPS)
  WIDTH = WIDTH*HALF
  IF (INEW .EQ. IOLD) THEN
    ZETA = ZNEW
  ENDIF

CONTINUE
CONTINUE

DO 50 IDUMMY = 1,MAXNEW
  U = ALF(1) - ZETA
  IF (U .EQ. ZERO) U = TENTH*EPS*BET2(2)
  RAT = ONE/U
  SUM = RAT
  DO 30 I = 2,J
    H = BET2(I)/U
    U = ALF(I) - ZETA - H
    IF (U .EQ. ZERO) U = TENTH*EPS*(H + BET2(I))
    RAT = (ONE + H*RAT)/U
    SUM = SUM + RAT
  CONTINUE
  BOT2 = U*SUM

DEFLATION

DO 40 I = IS,INDX-INC,INC
  DEL = ZETA - THET(I)
  IF (ABS(DEL).LT.EPS*ABS(ZETA)) THEN
    DEL = EPS*ABS(ZETA)
  ENDIF
  SUM = SUM + ONE/DEL
CONTINUE

CHECK FOR CONVERGENCE

ZNEW = ZETA + ONE/SUM
ZETA = ZNEW
1  IF (SPREAD+TENTH/SUM.EQ.SPREAD.OR.
     FLOAT(INC)/SUM.LT.ZERO) THEN
    GO TO 60
  ENDIF
CONTINUE
CONTINUE

CALL QLBOT(ALF,BET2,ZETA,BOT2,J)

ZNEW = THET(INDX)
THET(INDX) = ZETA
ZETA = ZNEW
BJ(INDX) = SQRT(RNM2*BOT2)

RETURN
END

```

Subroutine DEFLAT

The subroutine DEFLAT deflates the tridiagonal matrix T_j using an eigenvalue THETA. One step of the QR algorithm is used to perform the deflation. Each of the last few QR rotations will be almost a row and column interchange. For a more detailed description of the algorithm see [24]. ANALZT is the only routine that calls DEFLAT.

```

SUBROUTINE DEFLAT(ALF,BET2,THET,J)
.. THIS ROUTINE PERFORMS DEFLATION OF T USING SHIFT THET.
.. INPUTS
ALF(J)  DIAGONALS OF T
BET2(J)  SQUARES OF OFF-DIAGONALS OF T
THET    EIGENVALUE OF T TO BE DEFATED EXPLICITLY
J       DIMENSION OF THE TRIDIAGONAL MATRIX
.. OUTPUTS
ALF(J-1) MODIFIED DIAGONALS OF T

```

```

C      BET2(J-1) MODIFIED OFF-DIAGONALS OF T
C
C      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C      PARAMETER (MAXITR = 3)
C      DIMENSION ALF(J),BET2(J)
C
C      COMMON /RDATA/ RNM,RNM2,SPREAD,TOL,EPS,EPS1,REPS
C      1           ZERO, TENTH, EIGHTH, FOURTH, HALF, ONE, TWO,
C                  FOUR, TEN, ONE28, TWO56, FIVE12, ORTFAC, OVRFLW
C
C..... PWK ALGORITHM
C
C      DO 200 LOOP=1,MAXITR
C      C = ONE
C      S = ZERO
C      G = ALF(1) - THET
C      P = G**2
C      DO 100 I=1,J-1
C          B = BET2(I+1)
C          R = P + B
C          BET2(I) = S*R
C          OLDC = C
C          C = P/R
C          S = B/R
C          OLDG = G
C          A = ALF(I+1)
C          G = C*(A - THET) - S*OLDG
C          ALF(I) = OLDG + (A - G)
C          IF (C .EQ. ZERO) THEN
C              P = OLDC*B
C          ELSE
C              P = G*(G/C)
C          END IF
C 100    CONTINUE
C          BET2(J) = S*P
C          ALF(J) = G + THET
C          IF (BET2(J) .LE. EPS*SPREAD) THEN
C              GOTO 300
C          ENDIF
C 200    CONTINUE
C 300    J = J - 1
C
C      RETURN
C      END

```

Function NUMLES

This routine performs the LDL^T factorization of the tridiagonal matrix T_j to obtain eigenvalue counts for $T_j - \zeta$. By Sylvester's inertia theorem [21], D has the same signature as $T_j - \zeta$. L is not needed and therefore is not computed. The diagonal elements of D are not preserved, but a record of the number of negative or positive terms is kept and returned in NUMLES. NUMLES is called by ANALZT and NEWCOR.

```

C      INTEGER FUNCTION NUMLES(ALF,BET2,ZETA,N,INC,EPS)
C
C..... ROUTINE TO PERFORM THE SPECTRUM SLICING OF A TRIDIAGONAL MATRIX.
C
C      IF INC = 1, NUMLES RETURNS THE NUMBER OF EIGENVALUES BELOW ZETA.
C      IF INC = -1, NUMLES RETURNS THE NUMBER OF EIGENVALUES ABOVE ZETA.
C
C..... INPUTS
C
C      ALF(N)   DIAGONALS OF T
C      BET2(N)  SQUARE OF THE OFF-DIAGONALS OF T
C      ZETA     THE SHIFT TO BE APPLIED TO T
C      N        DIMENSION OF THE TRIDIAGONAL MATRIX
C      INC      INDEX TO INDICATE ABOVE OR BELOW
C      EPS      COMPUTER PRECISION
C
C..... OUTPUTS
C
C      NUMLES   THE NUMBER OF EIGENVALUES ABOVE/BELLOW ZETA.
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION ALF(1),BET2(1)
C

```

```

COMMON /LANCON/ ZERO, TENTH, EIGHTH, FOURTH, HALF, ONE, TWO,
FOUR, TEN, ONE28, TWO56, FIVE12, ORTFAC, OVRFLW
SAVE = BET2(1)
BET2(1) = ZERO
DEL = ONE
K = 0
DO 10 J=1,N
  DEL = (ALF(J) - ZETA) - BET2(J)/DEL
  IF ( DEL :EQ: ZERO ) DEL = EPS*BET2(J+1)*INC
  IF ( DEL :LT: ZERO ) K = K + 1
CONTINUE
NUMLES = K
IF ( INC :LT: 0 ) NUMLES = N - K
BET2(1) = SAVE
RETURN
END

```

Subroutine QLBOT

This routine computes the bottom element of the eigenvector of the tridiagonal matrix T_j corresponding to the eigenvalue contained in THET. The algorithm performs a step of the QL factorization. The product of the sines of the rotation angles is the desired bottom element and is returned in BOT. NEWCOR is the only routine that calls QLBOT.

```

SUBROUTINE QLBOT(ALF,BET2,THET,BOT,J)
COMPUTE THE BOTTOM ELEMENT OF THE NORMALIZED EIGENVECTOR OF A
TRIDIAGONAL MATRIX CORRESPONDING TO EIGENVALUE THET.
INPUTS
ALF(J)  DIAGONALS OF T
BET2(J)  SQUARE OF THE OFF-DIAGONALS OF T
THET    EIGENVALUE OF T
J        DIMENSION OF THE TRIDIAGONAL MATRIX
OUTPUTS
BOT      BOTTOM ELEMENT OF THE NORMALIZED EIGENVECTOR
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
DIMENSION ALF(1),BET2(1)
COMMON /LANCON/ ZERO, TENTH, EIGHTH, FOURTH, HALF, ONE, TWO,
FOUR, TEN, ONE28, TWO56, FIVE12, ORTFAC, OVRFLW
BOT = ONE
C = ONE
S = ZERO
G = ALF(J) - THET
P = G**2
DO 100 I=J-1,1,-1
  B = BET2(I+1)
  R = P + B
  OLDC = C
  C = P/R
  S = B/R
  OLDG = G
  A = ALF(I)
  G = C*(A - THET) - S*OLDG
  IF ( C :EQ: ZERO ) THEN
    P = OLDC*B
  ELSE
    P = G**2/C
  END IF
  BOT = BOT*S
CONTINUE
RETURN
END

```

Subroutine MOVE1

This routine inserts the value of T into the Kth location of the array Y. Elements of Y are shifted up or down depending on the sign of MINC. MOVE1 is only called by ANALZT for data management.

```

C      SUBROUTINE MOVE1(Y,K,L,MINC,T)
C..... MOVES THE CONTENT OF Y TO OPEN A SPACE FOR T.
C..... INPUT/OUTPUT
C      Y      THE ARRAY TO BE REORGANIZED
C      K      THE POSITION IN Y OF THE NEW ELEMENT T
C      L      END OF THE DATA IN Y
C      MINC   THE INCREMENT +1 OR -1
C      T      THE NEW ELEMENT TO BE INSERTED
C
C      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C      DIMENSION Y(1)
C
C      DO 100 I=L,K-MINC,MINC
C      Y(I) = Y(I+MINC)
100    CONTINUE
      Y(K) = T
C
C      RETURN
END

```

REFERENCES**Section 10.1**

1. K. J. Bathe, *Finite Element Procedures in Engineering Analysis*. Englewood Cliffs, N. J.: Prentice-Hall, 1982.
2. B. Noble, *Applied Linear Algebra*. Englewood Cliffs, N.J.: Prentice-Hall, 1969.

Section 10.2

3. E. L. Wilson, "The Static Condensation Algorithm," *International Journal for Numerical Methods in Engineering*, 8 (1974), 199–203.

Section 10.4

4. B. M. Irons, "Eigenvalue Economisers in Vibration Problems," *Journal of the Royal Aeronautical Society*, 67 (1963), 526.
5. B. M. Irons, "Structural Eigenvalue Problems: Elimination of Unwanted Variables," *AIAA Journal*, 3 (1965), 961.
6. R. J. Guyan, "Reduction of Stiffness and Mass Matrices," *AIAA Journal*, 3 (1965), 380.
7. R. D. Henshell and J. H. Ong, "Automatic Masters for Eigenvalue Economization," *Earthquake Engineering and Structural Dynamics*, 3 (1975), 375–383.

Section 10.5

8. K. J. Bathe, *Finite Element Procedures in Engineering Analysis*. Englewood Cliffs, N. J.: Prentice-Hall, 1982.
9. E. L. Wilson, "Numerical Methods for Dynamic Analysis," *International Symposium on Numerical Methods in Offshore Engineering*, Swansea, January 11–15, 1977.

Section 10.6

10. J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, *LINPACK Users' Guide*, SIAM, Philadelphia, 1979.
11. T. Ericsson and A. Ruhe, "The Spectral Transformation Lanczos Method for the Numerical Solution of Large Sparse Generalized Symmetric Eigenvalue Problems," *Mathematics of Computation*, 35 (1980), 1251–1268.
12. G. Golub, R. Underwood, and J. H. Wilkinson, "The Lanczos Algorithm for the Symmetric $Ax = \lambda Bx$ Problem," Tech. Rep. STAN-CS-72-720, Computer Science Department, Stanford University, 1972.
13. J. Grcar, "Analyses of the Lanczos Algorithm and of the Approximation Problem in Richardson's Method," Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1981.
14. C. Lanczos, "An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators," *Journal of Research of the National Bureau of Standards*, 45 (1950), 255–281.
15. B. Nour-Omid, B. N. Parlett, and R. L. Taylor, "Lanczos versus Subspace Iteration for Solution of Eigenvalue Problems," *International Journal for Numerical Methods in Engineering*, 19 (1983), 859–871.
16. B. Nour-Omid and B. N. Parlett, "How to Implement the Spectral Transformation," Tech. Rep. PAM-224, Center for Pure and Applied Mathematics, University of California, Berkeley, 1984.
17. B. Nour-Omid and R. W. Clough, "Dynamic Analysis of Structures Using Lanczos Coordinates," *Earthquake Engineering and Structural Dynamics*, 12 (1984), 565–577.
18. C. C. Paige, "Computational Variants of the Lanczos Method for the Eigenproblem," *Journal of the Institute for Mathematics and its Applications*, 10 (1972), 373–381.
19. C. C. Paige, "Error Analysis of the Lanczos Algorithm for Tridiagonalizing a Symmetric Matrix," *Journal of the Institute for Mathematics and its Applications*, 18 (1976), 341–349.
20. C. C. Paige, "Accuracy and Effectiveness of the Lanczos Algorithm for Symmetric Eigenproblems," *Linear Algebra and its Applications*, 34 (1980), 235–258.
21. B. N. Parlett, *The Symmetric Eigenvalue Problem*. Englewood Cliffs, N.J.: Prentice-Hall, 1980.
22. B. N. Parlett and D. Scott, "The Lanczos Algorithm with Selective Orthogonalization," *Mathematics of Computation*, 33, no. 145 (1979), 217–238.
23. B. N. Parlett, H. D. Simon, and L. M. Stringer, "On Estimating the Largest Eigenvalue with the Lanczos Algorithm," *Mathematics of Computation*, 38, no. 157 (1982), 153–165.
24. B. N. Parlett and B. Nour-Omid, "The Use of Refined Error Bounds When Updating Eigenvalues of Tridiagonals," *Linear Algebra and its Applications*, 68 (1985), 179–219.
25. R. L. Taylor, private communication, 1978.
26. D. S. Scott, "Analysis of the Symmetric Lanczos Process," Tech. Rep. ERL-M78/40, Electronics Research Laboratory, University of California, Berkeley, 1978.
27. H. D. Simon, "The Lanczos Algorithm with Partial Reorthogonalization," *Mathematics of Computation*, 42, no. 165 (1984), 115–142.
28. H. D. Wilkinson, *The Algebraic Eigenvalue Problem*. Oxford: Clarendon, 1965.
29. E. L. Wilson, M. Yuan, and J. M. Dickens, "Dynamic Analysis by Direct Superposition of Ritz Vectors," *Earthquake Engineering and Structural Dynamics*, 10 (1982), 813–821.

11

DLEARN—A Linear Static and Dynamic Finite Element Analysis Program

Thomas J. R. Hughes, Robert M. Ferencz,
and Arthur M. Raefsky

11.1. INTRODUCTION

This chapter describes DLEARN, a finite element code for linear static and dynamic analysis. For instructional purposes it may be considered a “first example” in that the use of auxiliary disk storage to reduce memory requirements is omitted, and blocking (segmenting large jobs via auxiliary storage) and overlay structures are avoided. Also, the class of capabilities—linear static and dynamic analysis—is relatively simple. Nevertheless, many sophisticated procedures are employed (e.g., compacted column solution and data structures, dynamic storage allocation, a memory manager, element routines, data generation routines, etc.) that are used in more advanced situations.

An important feature of DLEARN is that it has been written to be an integral constituent of this book. Methods and data-processing techniques developed throughout the first 10 chapters are employed in DLEARN. The names of variables and arrays are kept as close as possible to those used in the development of the theory. It is believed that this will expedite the learning process. Finite element methods find their ultimate manifestations in computer programs. Unless the student comprehends the intricacies of finite element programs, he or she will never fully appreciate the finite element method.

The major features and capabilities of DLEARN are listed below along with relevant subsections of the Input Instructions (these are denoted by “Sec. In.” followed by a decimal number):

- Modular program structure to enhance readability and extendability.
- Memory pointer dictionary printed with each execution to provide storage information and illustrate the concept of dynamic storage allocation.
- Simple, but labor-saving, nodal generation capabilities in one, two, and three

dimensions (Sec. In.5.0) as well as element generation capabilities (Sec. In.10.0).

- Complex loadings can be synthesized by combination of multiple load vectors and load-time functions (Secs. In.7.0 and In.8.0).
- Static analysis with full stress recovery (Sec. In.2.0).
- Dynamic analysis with an α -method predictor/corrector algorithm allowing calculation of consistent initial accelerations (Sec. In.2.0). Includes Rayleigh viscous damping with mixed implicit-explicit element groups (Sec. In.10.0).
- Four-node quadrilateral, elastic continuum element, with plane stress, plane strain, and torsionless axisymmetric capabilities. Includes a mean-dilatational \bar{B} capability for applications to nearly incompressible media. (Sec. In.10.1).
- Three-dimensional, elastic truss element (Sec. In.10.2).

DLEARNS is not intended to be a “full-service” analysis package with a broad range of capabilities. For instance, the number of element types included has been purposely limited. However, major emphasis has been placed on providing a flexible and comprehensible program structure to which the student or researcher can profitably add new features of his or her own choosing. Moreover, the code has been structured around techniques having natural extensions to nonlinear analysis.

DLEARNS may be used for problem solution in finite element courses and serves the following additional purposes:

1. It is felt that the first step in learning to implement finite element methods is to add a new element to an existing program. In this regard, DLEARNS is written so that it is very easy to add new elements. (Instructions are given in Sec.11.4 entitled “Adding an Element to DLEARNS.”) Students may elect to perform projects of this kind. For this reason, only two sets of element routines are included with the listing so that many standard element types (e.g., three-dimensional elasticity; beam, plate, and shell elements; heat conduction elements; etc.) are available for project topics.

2. DLEARNS makes available to students a good, first, finite element “software package.” The subroutines are efficiently and simply coded and, as the student becomes more advanced, the subroutines may be used as a set of building blocks for more-sophisticated applications. With this idea in mind it was decided to include a very efficient equation solver in DLEARNS, which employs a compacted column storage mode.

The reader is forewarned that the level of documentation is not considered self-sufficient (nor is it intended to be) for fully understanding all aspects of the program but rather is supplementary to the remainder of the text.

1.2. DESCRIPTION OF CODING TECHNIQUES USED IN DLEARNS

In keeping with contemporary programming practice, DLEARNS has been written with the objectives of readability, maintainability and extendability. The 1000-line subroutines of an earlier generation of finite element codes have been eschewed in

favor of compact subroutines with a limited number of tasks to perform. While large subroutines have a place in codes dedicated to efficient solution of large-scale problems, we believe smaller program units will be more easily comprehended by students first learning how to implement finite element methodologies. To this end, utility subroutines are widely used to remove “in-line” code, which, though faster, might obscure the algorithmic outline of a procedure. To understand DLEARN and ultimately extend its capabilities, the user must become familiar with the data structure.

Definition. The *data structure* is the mode of storage, location, and history of existence of the data employed.

In DLEARN, all the data are stored in central (“core”) memory in the unnamed common block COMMON A(MTOT), where the size parameter MTOT is set in the main program. Many operating systems, such as those for Control Data Corporation (CDC) and Cray Research computers, allow greater run-time flexibility in memory allocation if the storage vector A is kept in an unnamed COMMON block. This fact is responsible for the commonly made remark that finite element programs store data in “blank common.” On virtual memory machines such as the Digital Equipment Corporation (DEC) VAX-11/7xx series, however, no penalty is incurred by using a named common block. The basic features of the data structure are:

Compacted column architecture This feature is engendered by the method of equation solving employed. The solver is called variously a “profile,” “skyline,” or “active column” solver. Since, in large problems, the dominant portion of storage is usually that devoted to the effective mass matrix M^* , the storage scheme for M^* is the most important feature of the storage.

Dynamic storage allocation Because of the variable sizes of the arrays needed in different problems, it would be very inefficient to “fix” the dimensioning. In the dynamic storage allocation concept the dimensions of arrays are set in the program at the time of execution.

Element groups Many features of finite element computer programs are the same even though the intended applications may be quite different. To some extent this is also true of the individual finite element subroutines. However, the data structure and options available may vary considerably from one element to another. To accommodate these differences, the elements are read in, stored and operated upon in groups. The data structure for the group is set up via dynamic storage allocation in individual element subroutines.

11.2.1 Compacted Column Storage Scheme

Assume the symmetric effective mass matrix M^* has dimension 8×8 and has zero and nonzero elements indicated by 0 and X, respectively, in the figure on page 634. The dashed line enveloping the nonzero terms is sometimes called the “profile” or “skyline.”

$$M^* = [M_{ij}^*] = \begin{bmatrix} X & X & 0 & 0 & 0 & 0 & X \\ X & X & 0 & X & 0 & 0 & 0 \\ X & X & X & 0 & 0 & 0 & 0 \\ X & 0 & X & 0 & X & X & X \\ X & X & 0 & 0 & X & X & X \\ X & X & 0 & 0 & X & X & X \\ X & X & 0 & 0 & X & X & X \\ X & X & 0 & 0 & X & X & X \end{bmatrix}$$

M^* is stored in a one-dimensional array, ALHS, column-wise beginning with the first nonzero element in each column and ending with the diagonal term. The locations of the diagonal terms in ALHS are stored in a one-dimensional integer array IDIAG. The dimension of IDIAG is NEQ, the number of equations (e.g., for M^* , NEQ = 8). The dimension of ALHS is NALHS, the sum of the "column heights." The column height is the number of terms in a column beginning with the first nonzero term and ending with the diagonal term (e.g., for the above matrix M^* , column 1 has height 1, column 4 has height 2, column 6 has height 3, etc.). NALHS for M^* is 24. The arrays ALHS and IDIAG corresponding to M^* are given explicitly as follows:

| | | |
|-----------------------|-----|----------|
| ALHS(1) = M_{11}^* | } | column 1 |
| ALHS(2) = M_{12}^* | { } | column 2 |
| ALHS(3) = M_{22}^* | | |
| ALHS(4) = M_{23}^* | { } | column 3 |
| ALHS(5) = M_{33}^* | | |
| ALHS(6) = M_{34}^* | { } | column 4 |
| ALHS(7) = M_{44}^* | | |
| ALHS(8) = M_{25}^* | { } | column 5 |
| ALHS(9) = M_{35}^* | | |
| ALHS(10) = M_{45}^* | { } | column 5 |
| ALHS(11) = M_{55}^* | | |
| ALHS(12) = M_{46}^* | { } | column 6 |
| ALHS(13) = M_{56}^* | | |
| ALHS(14) = M_{66}^* | { } | column 6 |
| ALHS(15) = M_{67}^* | | |
| ALHS(16) = M_{77}^* | { } | column 7 |
| | | |

$$\begin{aligned}
 & \text{ALHS}(17) = M_{18}^* \\
 & \text{ALHS}(18) = M_{28}^* \\
 & \text{ALHS}(19) = M_{38}^* \\
 & \text{ALHS}(20) = M_{48}^* \\
 & \text{ALHS}(21) = M_{58}^* \\
 & \text{ALHS}(22) = M_{68}^* \\
 & \text{ALHS}(23) = M_{78}^* \\
 & \text{ALHS}(24) = M_{88}^* \\
 \\
 & \text{IDIAG}(1) = 1 \\
 & \text{IDIAG}(2) = 3 \\
 & \text{IDIAG}(3) = 5 \\
 & \text{IDIAG}(4) = 7 \\
 & \text{IDIAG}(5) = 11 \\
 & \text{IDIAG}(6) = 14 \\
 & \text{IDIAG}(7) = 16 \\
 & \text{IDIAG}(8) = 24
 \end{aligned}
 \quad \left. \right\} \text{column 8}$$

The zero terms beneath the skyline must be stored because they become nonzero during the factorization process. All information in the original M^* is now contained in the arrays ALHS and IDIAG in compact fashion. For large finite element effective mass matrices, the saving of storage is considerable.

The heights of the columns in M^* are determined by subroutine COLHT from the element group LM arrays. The column heights are initially stored in IDIAG. When the calculation of column heights is completed, subroutine DIAG determines the diagonal address and “overwrites” them into IDIAG.

The right-hand-side vector (residual, or out-of-balance, force) in

$$M^* \Delta a = R$$

is stored in an array BRHS of dimension NEQ in the obvious way:

$$\text{BRHS}(1) = R_1$$

$$\text{BRHS}(2) = R_2$$

$$\vdots$$

$$\text{BRHS}(8) = R_8$$

The elimination scheme employed is based upon the following theorem.

Theorem. Let M^* be symmetric and positive-definite. Then there exists a

nonsingular upper triangular matrix U , with unit diagonal entries, and a diagonal matrix D such that

$$M^* = U^T D U$$

Remarks

1. In the case when M^* is positive-definite, the diagonal entries of D are all strictly positive. Thus examining the diagonal entries of D enables us to determine the rank of M^* . This is the purpose of the *rank check* option in the program.

2. The matrix U has the same profile as M^* . The fact that U has diagonal entries equal to one allows us to save storage by placing the diagonal entries of D in the diagonal entries of the array ALHS. Thus after factorization the nonzero entries of both U and D are stored in array ALHS. During the factorization procedure *no* auxiliary storage is needed.

3. Upper triangular matrices with unit diagonal entries are sometimes referred to as *unit upper triangular matrices*.

11.2.2 Crout Elimination

The procedure used for performing the factorization of the left-hand-side matrix is called *Crout elimination*, a convenient variant of Gauss elimination. In the Crout algorithm, one column at a time is completely factorized, beginning with the first column and not involving subsequent columns. This feature proves convenient in nonlinear analysis in which only a small zone of the entire matrix is nonlinear. The linear portion can be located in the first columns, factorized once and retained for the duration of the calculation.

After factorization, the solution is carried out in three steps:

$$U^T z = R \quad \text{forward reduction}$$

$$Dy = z \quad \text{diagonal scaling}$$

$$U \Delta a = y \quad \text{back substitution}$$

In each step the solution is an explicit process, the coefficient matrix being triangular or diagonal. The intermediate vectors y and z are in turn stored in the array BRHS. Thus again no additional storage besides that for ALHS and BRHS (and IDIAG) is needed to carry out the factorization and solution processes.

Derivation of Computational Formulas for Crout Elimination

To simplify the notation, let us write

$$Ax = b$$

and assume x is n -dimensional. We wish to derive algorithms for

$$A = U^T D U$$

$$U^T z = b$$

$$Dy = z$$

and

$$Ux = y$$

Factorization

The index version of the Crout factorization is expressed as

$$A_{ij} = \sum_{k=1}^j U_{ki} D_{kk} U_{kj} \quad (1 \leq i \leq j)$$

where

$$U_{ii} = 1,$$

$$U_{ij} = 0, \quad \text{for } i > j$$

Useful formulas for programming may be derived by expanding the above for $j = 1, 2, \dots$, as follows:

$$A_{11} = U_{11} D_{11} U_{11} = D_{11}$$

$$\begin{aligned} A_{12} &= U_{11} D_{11} U_{12} + U_{21} D_{22} U_{22} \\ &= D_{11} U_{12} \end{aligned}$$

$$\begin{aligned} A_{22} &= U_{12} D_{11} U_{12} + U_{22} D_{22} U_{22} \\ &= U_{12} D_{11} U_{12} + D_{22} \end{aligned}$$

$$\begin{aligned} A_{13} &= U_{11} D_{11} U_{13} + U_{21} D_{22} U_{23} + U_{31} D_{33} U_{33} \\ &= D_{11} U_{13} \end{aligned}$$

$$\begin{aligned} A_{23} &= U_{12} D_{11} U_{13} + U_{22} D_{22} U_{23} + U_{32} D_{33} U_{33} \\ &= U_{12} D_{11} U_{13} + D_{22} U_{23} \end{aligned}$$

$$\begin{aligned} A_{33} &= U_{13} D_{11} U_{13} + U_{23} D_{22} U_{23} + U_{33} D_{33} U_{33} \\ &= U_{13} D_{11} U_{13} + U_{23} D_{22} U_{23} + D_{33} \end{aligned}$$

$$\begin{aligned} A_{14} &= U_{11} D_{11} U_{14} + U_{21} D_{22} U_{24} + U_{31} D_{33} U_{34} + U_{41} D_{44} U_{44} \\ &= D_{11} U_{14} \end{aligned}$$

$$\begin{aligned} A_{24} &= U_{12} D_{11} U_{14} + U_{22} D_{22} U_{24} + U_{32} D_{33} U_{34} + U_{42} D_{44} U_{44} \\ &= U_{12} D_{11} U_{14} + D_{22} U_{24} \end{aligned}$$

$$\begin{aligned} A_{34} &= U_{13} D_{11} U_{14} + U_{23} D_{22} U_{24} + U_{33} D_{33} U_{34} + U_{43} D_{44} U_{44} \\ &= U_{13} D_{11} U_{14} + U_{23} D_{22} U_{24} + D_{33} U_{34} \end{aligned}$$

$$\begin{aligned} A_{44} &= U_{14} D_{11} U_{14} + U_{24} D_{22} U_{24} + U_{34} D_{33} U_{34} + U_{44} D_{44} U_{44} \\ &= U_{14} D_{11} U_{14} + U_{24} D_{22} U_{24} + U_{34} D_{33} U_{34} + D_{44} \end{aligned}$$

And so on. These formulas may be solved for the U_{ij} 's and D_{jj} 's:

$$D_{11} = A_{11}$$

$$U_{12} = \frac{A_{12}}{D_{11}}$$

$$D_{22} = A_{22} - D_{11}U_{12}^2$$

$$U_{13} = \frac{A_{13}}{D_{11}}$$

$$U_{23} = \frac{A_{23} - U_{12}D_{11}U_{13}}{D_{22}}$$

$$D_{33} = A_{33} - D_{11}U_{13}^2 - D_{22}U_{23}^2$$

$$U_{14} = \frac{A_{14}}{D_{11}}$$

$$U_{24} = \frac{A_{24} - U_{12}D_{11}U_{14}}{D_{22}}$$

$$U_{34} = \frac{A_{34} - U_{13}D_{11}U_{14} - U_{23}D_{22}U_{24}}{D_{33}}$$

$$D_{44} = A_{44} - D_{11}U_{14}^2 - D_{22}U_{24}^2 - D_{33}U_{34}^2$$

And so on. For purposes of efficient programming, it is worthwhile to introduce the auxiliary variable

$$L_{ji} \stackrel{\text{def}}{=} D_{ii}U_{ij}$$

Then, equivalent to the above, we have

$$D_{11} = A_{11}$$

$$L_{21} = A_{12}$$

$$U_{12} = L_{21}/D_{11}$$

$$D_{22} = A_{22} - L_{21}U_{12}$$

$$L_{31} = A_{13}$$

$$L_{32} = A_{23} - U_{12}L_{31}$$

$$U_{13} = L_{31}/D_{11}$$

$$U_{23} = L_{32}/D_{22}$$

$$D_{33} = A_{33} - L_{31}U_{13} - L_{32}U_{23}$$

$$L_{41} = A_{14}$$

$$L_{42} = A_{24} - U_{12}L_{41}$$

$$L_{43} = A_{34} - U_{13}L_{41} - U_{23}L_{42}$$

$$U_{14} = L_{41}/D_{11}$$

$$U_{24} = L_{42}/D_{22}$$

$$U_{34} = L_{43}/D_{33}$$

$$D_{44} = A_{44} - L_{41}U_{14} - L_{42}U_{24} - L_{43}U_{34}$$

And so on. Summarizing, for $j = 1, 2, \dots, n$

$$L_{ji} = A_{ij} - \sum_{k=1}^{i-1} U_{ki}L_{jk}, \quad 1 \leq i \leq j-1$$

$$U_{ij} = L_{ji}/D_{ii}$$

$$D_{jj} = A_{jj} - \sum_{i=1}^{j-1} L_{ji}U_{ij}$$

Observe that no additional storage besides that needed for A is necessary in the factorization process if A is overwritten by U and D , viz.,

For $i = 2, 3, \dots, j-1$

$$A_{ij} \leftarrow A_{ij} - \sum_{k=1}^{i-1} A_{ki}A_{kj}$$

For $i = 1, 2, \dots, j-1$

$$T \leftarrow A_{ij}$$

$$A_{ij} \leftarrow T/A_{ii}$$

$$A_{ij} \leftarrow A_{jj} - TA_{ij}$$

This is the procedure coded in subroutine FACTOR in DLEARN. Additionally, FACTOR takes account of the profile storage of the coefficient matrix. The indexing necessary to account for the profile somewhat complicates the procedure but is necessary to achieve optimal efficiency. Figures 11.2.1(a) and 11.2.1(b) are presented as an aid for understanding the indexing and one-dimensional storage of the coefficient matrix in FACTOR. Note that if $A_{ii} = 0$, subroutine FACTOR skips the operations in the second do loop.

Forward Reduction

The indicial form of the forward reduction is

$$\sum_{i=1}^n U_{ij}z_i = b_j$$

$ISTART = J - JCOLHT + 2$

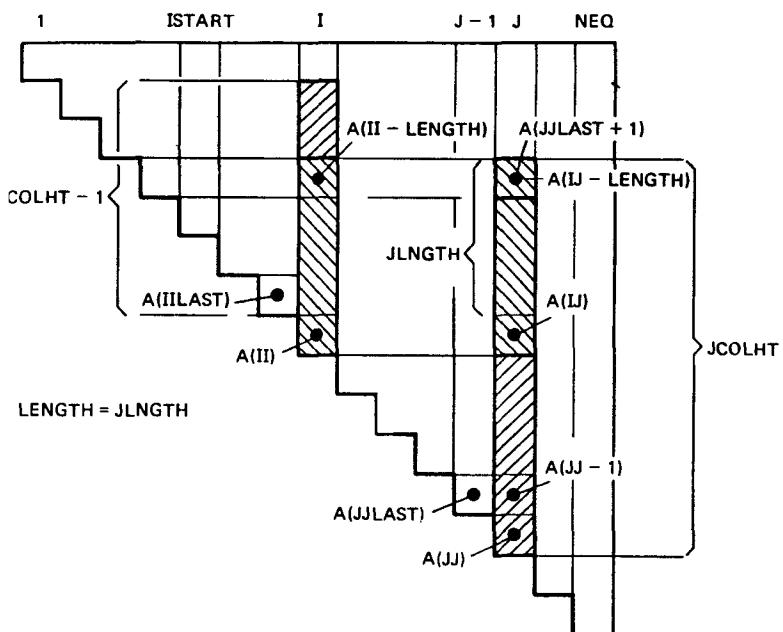


Figure 11.2.1(a) Indexing and storage for subroutine FACTOR; the case $ICOLHT - 1 > JLNGTH$.

$ISTART = J - JCOLHT + 2$

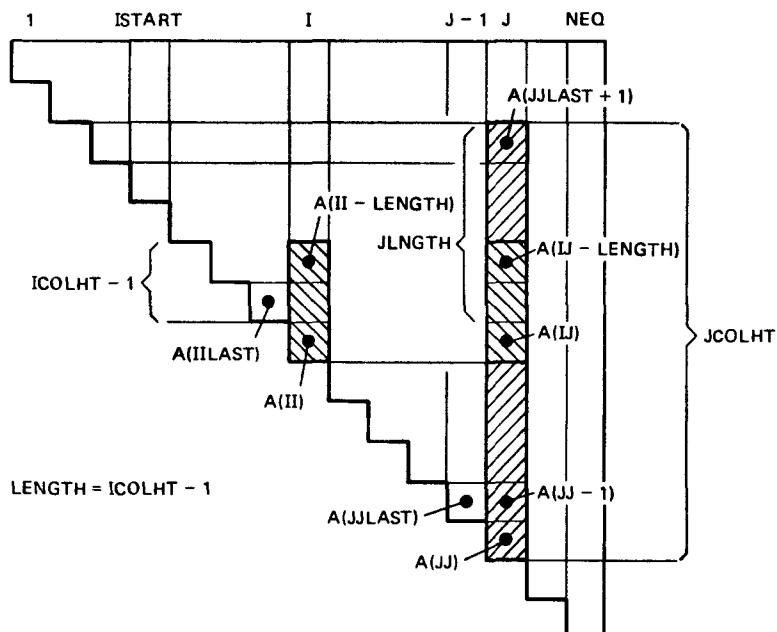


Figure 11.2.1(b) Indexing and storage for subroutine FACTOR; the case $ICOLHT - 1 < JLNGTH$.

Using the properties of the U_{ij} 's, we can immediately derive

$$z_1 = b_1$$

$$z_2 = b_2 - U_{12}z_1$$

$$z_3 = b_3 - U_{13}z_1 - U_{23}z_2$$

.

.

.

from which we conclude that

$$z_j = b_j - \sum_{i=1}^{j-1} U_{ij}z_i, \quad (2 \leq j \leq n)$$

It is clear from this formula that b_j may be overwritten by z_j , viz.,

$$b_j \leftarrow b_j - \sum_{i=1}^{j-1} A_{ij}b_i \quad (2 \leq j \leq n)$$

where we have used the fact that A_{ij} has been overwritten by U_{ij} . This procedure is coded in subroutine BACK, which takes account of the profile storage of the coefficient matrix. The indexing is illustrated in Fig. 11.2.2(a).

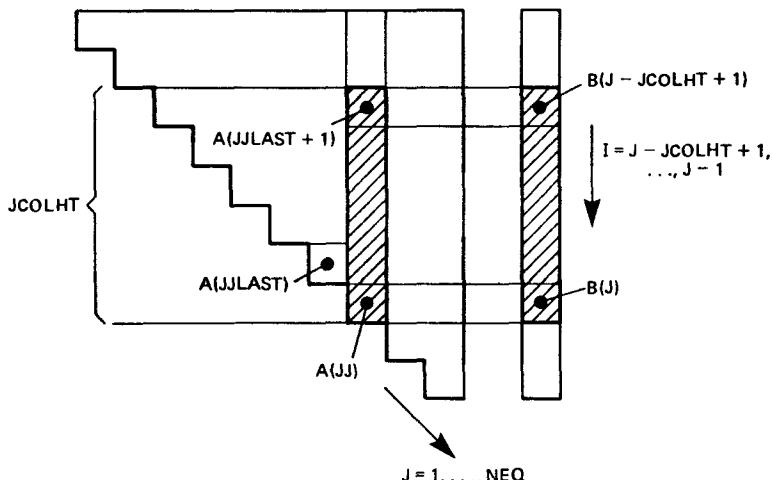


Figure 11.2.2(a) Indexing for forward reduction phase of subroutine BACK.

Diagonal Scaling

Diagonal scaling is also performed in subroutine BACK. Again b_j is overwritten:

$$y_j = \frac{z_j}{D_{jj}} \Leftrightarrow b_j \leftarrow b_j / A_{jj} \quad (1 \leq j \leq n)$$

If $A_{jj} = 0$, diagonal scaling is skipped in subroutine BACK.

Back Substitution

The indicial form of the back substitution is

$$\sum_{j=1}^n U_{ij} x_j = y_i$$

Expanding this expression, we obtain

$$\begin{aligned}
 x_1 &= y_1 - U_{1n} x_n & x_n &= U_{1,n-1} x_{n-1} \cdots - U_{13} x_3 - U_{12} x_2 \\
 x_2 &= y_2 - U_{2n} x_n & x_n &= U_{2,n-1} x_{n-1} \cdots - U_{23} x_3 \\
 \vdots & & & \\
 x_{n-2} &= y_{n-2} - U_{n-2,n} x_n & x_n &= U_{n-2,n-1} x_{n-1} \\
 x_{n-1} &= y_{n-1} - U_{n-1,n} x_n & & \\
 x_n &= y_n & &
 \end{aligned}$$

(n - 1)st column

3rd column

2nd column

nth column

There are several ways of organizing the calculations. The most computationally efficient proceeds from left to right, subtracting multiples of the columns of U . Proceeding in this manner allows b to be overwritten with x :

```

For j = n, n - 1, ..., 2
  For i = 1, 2, ..., j - 1
    b_i ← b_i - A_ij b_j.
  
```

The indexing used in subroutine BACK is illustrated in Fig. 11.2.2(b). The number of “flops” (floating-point operations¹) involved in Crout factorization is approximately $m^2 n/2$, where m is the mean half-bandwidth (defined to be the sum of the column heights divided by the number of equations). In obtaining this result we assume $1 \ll m \ll n$, and thus lower-order terms may be ignored. Given the factorized matrix, a “solution” (which consists of a forward reduction, diagonal scaling, and back

¹ Roughly speaking, one flop consists of a floating-point multiply, an addition, and some subscripting. For example, $X = X + Y(I)*Z(J)$ would amount to one flop.

ISTART = J - JCOLHT + 1

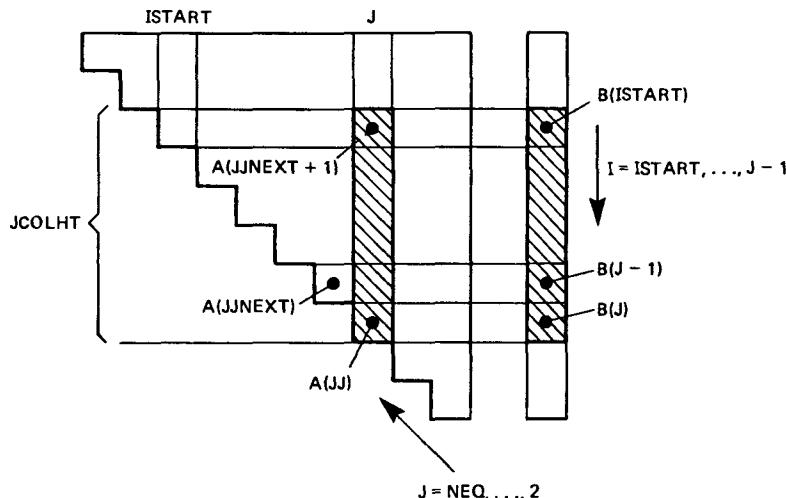


Figure 11.2.2(b) Indexing for back substitution phase of subroutine BACK.
(Note: ISTART has a different significance in this routine than in subroutine FACTOR.)

substitution) involves approximately $2mn$ flops. Clearly, for large systems, factorization of the coefficient matrix is much more expensive than solution. Factorization costs vary quadratically with m and thus there is considerable interest in schemes for ordering the equations to minimize m . For additional information on this subject, see [1–4]. An excellent reference for equation solving in general is [5].

Exercise 1. Consider the system

$$Kd = F$$

where

$$K = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

$$d = \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{Bmatrix} \quad F = \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

- Obtain the Crout factorization of K (i.e., define the diagonal matrix D and the unit upper triangular matrix U).
- Obtain the solution d by performing the following steps:

$$U^T z = F \quad \text{forward reduction}$$

$$Dy = z \quad \text{diagonal scaling}$$

$$Ud = y \quad \text{back substitution}$$

- Check parts (i) and (ii) by verifying $U^T D U = K$ and $Kd = F$.

Exercise 2. The *Cholesky decomposition* of a symmetric positive-definite matrix A takes the form

$$A = G^T G$$

where G is an upper triangular matrix with positive diagonal entries. The Cholesky decomposition may be obtained from the Crout factorization by way of

$$G = D^{1/2} U$$

Develop an algorithm for directly determining G along the lines of the Crout factorization presented above.

11.2.3 Dynamic Storage Allocation

Examples

In the initialization subroutine STATIN, memory storage is allocated for the global arrays required for static analysis. On lines 24 and 25 the “pointers” are determined for the X and ID arrays:

```
MPX = MPOINT('X      ',NSD ,NUMNP,0,IPREC)
MPID = MPOINT('ID     ',NDOF,NUMNP,0,1)
```

FUNCTION MPOINT manages memory allocation within the unnamed or “blank” COMMON A(MTOT). The first argument is a 2A4 string, which serves as the array’s label when the storage dictionary is printed. The next three arguments are array dimensions; in this case we see that both ID and X are two-dimensional arrays. The last argument determines the precision of the array (IPREC = 1 or 2 corresponding to single or double precision, respectively). Notice that as an integer array, ID is always defined to be single precision.

The calling statements:

```
CALL COORD(A(MPX),NSD,NUMNP,IPRTIN)
CALL BC(A(MPID),NDOF,NUMNP,NEQ,IPRTIN)
```

and the corresponding subroutine statements:

```
SUBROUTINE COORD(X,NSD,NUMNP, IPRTIN)
SUBROUTINE BC(ID,NDOF,NUMNP,NEQ,IPRTIN)
```

mean array X begins at address MPX of blank common and array ID begins at address MPID of blank common, respectively.

In COORD and BC, the arrays X and ID are treated as *two-dimensional*, as evidenced by the DIMENSION statements:

```
DIMENSION X(NSD,1)
```

and

```
DIMENSION ID(NDOF,1)
```

respectively. It does not matter that X and ID correspond to portions of a one-dimensional array in the calling program STATIN. The significance of the dimensioning is as follows: *The arrays X and ID are stored in blank common in consecutively addressed cells by columns.* Consequently the addresses of X(M,N) and ID(M,N) in A are given by

$$\text{IPREC} * (\text{NSD} * (N - 1) + M - 1) + \text{MPX}$$

and

$$\text{NDOF} * (N - 1) + M - 1 + \text{MPID}$$

Thus the only information necessary to determine the appropriate addresses in A are the starting addresses (i.e., MPX and MPID, respectively), which are provided by the CALL statements, and the number of rows (i.e., NSD and NDOF, respectively), which are provided by the DIMENSION statements in the subroutines. This explains why it is not necessary to define the number of columns of two-dimensional arrays "passed" to subroutines. The 1's in the column slots could be replaced by any positive integer; they only serve notice that these arrays are to be regarded as two-dimensional in the subroutine—the integer is never used. For three-dimensional arrays, the first two dimensions must be precisely defined, whereas the third may be set to 1, etc.

The array names, pointers, and other dictionary entries are stored at the end of blank common. Thus for the first two pointers declared during execution:

$$A(\text{MTOT} - 4) = \text{MPSTEP} = 1$$

$$A(\text{MTOT} - 11) = \text{MPDPRT}, \dots$$

Note that seven words of storage are required for each entry in the dictionary. The memory FUNCTION MPOINT monitors the blank common allocation to ensure that data arrays starting from the beginning of blank common do not overwrite the array pointer information. If the user receives a memory error, he or she must reduce the size of the problem or increase MTOT. Given the availability of sufficient machine memory size, *the latter action requires only recompiling the main program (which amounts to only a few statements), followed by linking to create a new executable file.*

The following tables present the storage arrays and associated pointers for all global data used during static or dynamic analysis. Also, an example of local element data storage requirements is presented for the four-node quadrilateral, elastic continuum element QUADC.

TABLE 11.2.1 Storage in Blank Common—Time Sequence and Time History Data*

| Array Pointer | Array |
|---------------|---------------|
| MPSTEP | NSTEP(NUMSEQ) |
| MPDPRT | NDPRT(NUMSEQ) |
| MPSPRT | NSPRT(NUMSEQ) |
| MPHPLT | NHPLT(NUMSEQ) |
| MPITER | NITER(NUMSEQ) |
| MPALPH | ALPHA(NUMSEQ) |
| MPBETA | BETA(NUMSEQ) |

* Required for static and dynamic analysis, storage allocated in SUBROUTINE TSEQ.

TABLE 11.2.1 (cont.) Storage in Blank Common—Time Sequence and Time History Data

| Array Pointer | Array |
|---------------|-----------------------------------|
| MPGAMM | GAMMA(NUMSEQ) |
| MPDT | DT(NUMSEQ) |
| MPIDHS | IDHIST(3,NDOUT) [†] |
| MPDOUT | DOUT(NDOUT+1,NPLPTS) [†] |

where, for each time sequence I=1,NUMSEQ:

| | | |
|----------|---|---|
| NSTEP(I) | = | number of time steps |
| NDPRT(I) | = | number of time steps between printing of kinematic output |
| NSPRT(I) | = | number of time steps between printing of element stress/strain output |
| NHPLT(I) | = | number of time steps between plotting of output time-history data |
| NITER(I) | = | number of iterations in corrector loop |
| ALPHA(I) | = | α coefficient for time-integration algorithm |
| BETA(I) | = | β coefficient for time-integration algorithm |
| GAMMA(I) | = | γ coefficient for time-integration algorithm |
| DT(I) | = | time step Δt |

End time sequence data.

| | | |
|-------------|---|---|
| IDHIST(1,N) | = | node number for output history N |
| IDHIST(2,N) | = | degree of freedom number for output history N |
| IDHIST(3,N) | = | kinematic-variable-type number for output history N |
| DOUT(1,J) | = | value of time at plot step J (stored as single precision for IPREC = 1 or 2) |
| DOUT(N+1,J) | = | output history N at plot step J (stored as single precision for IPREC = 1 or 2) |
| NDOUT | = | number of nodal output time-histories |
| NPLPTS | = | number of points in output time-history plots |
| IPREC | = | precision flag: 1 \Rightarrow single, 2 \Rightarrow double |

[†]If NDOUT=0, no storage is allocated for IDHIST and DOUT.

TABLE 11.2.2 Storage in Blank Common—Dynamic Analysis Data*

| Array Pointer | Array |
|---------------|--|
| MPVPRD | VPRED(NDOF,NUMNP) |
| MPDPRD | DPRED(NDOF,NUMNP) |
| MPA | A(NDOF,NUMNP) |
| MPV | V(NDOF,NUMNP) |
| where: | |
| VPRED(I,J) | = Ith predictor velocity component of node J |
| DPRED(I,J) | = Ith predictor displacement component of node J |
| A(I,J) | = Ith acceleration component of node J |
| V(I,J) | = Ith velocity component of node J |
| NDOF | = number of degrees of freedom per node |
| NUMNP | = number of nodal points |

* Required only for dynamic analysis; storage allocated in SUBROUTINE DYNPTS.

TABLE 11.2.3 Storage in Blank Common—Static Analysis Data*

| Array Pointer | Array |
|---------------|-----------------------------------|
| MPD | D(NDOF,NUMNP) |
| MPX | X(NSD,NUMNP) |
| MPID | ID(NDOF,NUMNP) |
| MPF | F(NDOF,NUMNP,NLVECT) [†] |
| MPG | G(NPTSLF,2,NLTFTN) [‡] |
| MPG1 | G1(NLTFTN) [‡] |
| MPDIAG | IDIAG(NEQ) |
| MPNGRP | NGRP(NUMEQ) |

where:

| | |
|----------|--|
| D(I,J) | = Ith displacement component of node J |
| X(I,J) | = Ith coordinate of node J |
| ID(I,J) | = equation number for degree of freedom I, node J |
| F(I,J,L) | = Ith prescribed force/kinematic-variable [§] component for node J, load vector L |
| G(K,1,L) | = time t_k for load-time function L |
| G(K,2,L) | = function value at time t_k , load-time function L |
| G1(L) | = value of load-time function L at current time t |
| IDIAG(N) | = location of Nth diagonal element in coefficient array ALHS |
| NGRP(N) | = pointer to beginning address of element group N |
| NDOF | = number of degrees of freedom per node |
| NUMNP | = number of nodal points |
| NSD | = number of space dimensions |
| NLVECT | = number of load vectors |
| NPTSLF | = number of points in a load-time function |
| NLTFTN | = number of load-time functions |
| NEQ | = number of equations |
| NUMEG | = number of element groups |

* Required for static and dynamic analysis; storage allocated in SUBROUTINE STATIN.

[†] If NLVECT = 0, no storage is allocated for F.

[‡] If NLTFTN = 0, no storage is allocated for G and G1.

[§] Displacements for static analysis; displacements, velocities or accelerations for dynamic analysis.

TABLE 11.2.4 Storage Requirements for Four-Node Quadrilateral, Elastic Continuum Element (NTYPE = 1)

Each element group has an associated array NPAR with pointer MPNPAR; group control data can be stored in positions 1 through 16. This is followed immediately in blank common by the MP array, which stores the memory pointers for the element group data. This allows each element type to have a unique data structure. For the present element, MP contains 35 words and stores the following pointers:

| Array Pointer | Array |
|---------------|----------------------|
| MP(MW) | W(NINT) |
| MP(MDET) | DET(NINT) |
| MP(MR) | R(NINT) |
| MP(MSHL) | SHL(NROWSH,NEN,NINT) |

TABLE 11.2.4 (cont.) Storage Requirements for Four-Node Quadrilateral, Elastic Continuum Element (NTYPE = 1)

| Array Pointer | Array |
|---------------|--------------------------|
| MP(MSHG) | SHG(NROWSH,NEN,NINT) |
| MP(MSHGBR) | SHGBAR(NROWSH,NEN,NRINT) |
| MP(MRHO) | RHO(NUMAT) |
| MP(MRDPM) | RDAMPM(NUMAT) |
| MP(MRDPK) | RDAMPK(NUMAT) |
| MP(MTH) | TH(NUMAT) |
| MP(MC) | C(NROWB,NROWB,NUMAT) |
| MP(MGRAV) | GRAV(NESD) |
| MP(MIEN) | IEN(NEN,NUMEL) |
| MP(MMAT) | MAT(NUMEL) |
| MP(MLM) | LM(NED,NEN,NUMEL) |
| MP(MIELNO) | IELNO(NSURF) |
| MP(MISIDE) | ISIDE(NSURF) |
| MP(MPRESS) | PRESS(2,NSURF) |
| MP(MSHEAR) | SHEAR(2,NSURF) |
| MP(MISHST) | ISHIST(3,NSOUT)* |
| MP(MSOUT) | SOUT(NSOUT+1,NPLPTS)* |
| MP(MELEFM) | ELEFFM(NEE,NEE) |
| MP(MXL) | XL(NESD,NEN) |
| MP(MWORK) | WORK(16) |
| MP(MB) | B(NROWB,NEE) |
| MP(MDMAT) | DMAT(NROWB,NROWB) |
| MP(MDB) | DB(NROWB,NEE) |
| MP(MVL) | VL(NED,NEN) |
| MP(MAL) | AL(NED,NEN) |
| MP(MELRES) | ELRESF(NEE) |
| MP(MDL) | DL(NED,NEN) |
| MP(MSTRN) | STRAIN(NROWB) |
| MP(MSTRS) | STRESS(NROWB) |
| MP(MPSTRN) | PSTRN(NROWB) |
| MP(MPSTRS) | PSTRS(NROWB) |

where, for each quadrature point L=1,NINT:

| | | |
|------------|---|--|
| W(L) | = | integration-rule weight |
| DET(L) | = | Jacobian determinant |
| R(L) | = | radius at quadrature point (axisymmetric option) |
| SHL(1,J,L) | = | local derivative $N_{j,\xi}$ |
| SHL(2,J,L) | = | local derivative $N_{j,\eta}$ |
| SHL(3,J,L) | = | N_j |
| SHG(1,J,L) | = | global derivative N_{j,x_1} |
| SHG(2,J,L) | = | global derivative N_{j,x_2} |
| SHG(3,J,L) | = | N_j |

End quadrature-point data.

* If NSOUT = 0, no storage is allocated for ISHIST and SOUT.

TABLE 11.2.4 (cont.) Storage Requirements for Four-Node Quadrilateral, Elastic Continuum Element (NTYPE = 1)

For mean-dilatational \bar{B} formulation:

$$\text{SHGBAR}(1,J) = \int_{\Omega^*} N_{J,x_1} d\Omega / \int_{\Omega^*} d\Omega$$

$$\text{SHGBAR}(2,J) = \int_{\Omega^*} N_{J,x_2} d\Omega / \int_{\Omega^*} d\Omega$$

$$\text{SHGBAR}(3,J) = \int_{\Omega^*} N_J d\Omega / \int_{\Omega^*} d\Omega$$

For each material set M=1,NUMAT:

| | |
|-----------|---|
| RHO(M) | = mass density |
| RDAMP(M) | = Rayleigh mass-proportional damping coefficient |
| RDAMPK(M) | = Rayleigh stiffness-proportional damping coefficient |
| TH(M) | = thickness |
| C(I,J,M) | = I,Jth component of material moduli matrix |

End material data.

| | |
|-------------|---|
| GRAV(I) | = gravity-load multiplier in x_i direction |
| IEN(J,NEL) | = global node number for local node J of element NEL |
| MAT(NEL) | = material set number for element NEL |
| LM(I,J,NEL) | = global equation number for Ith degree of freedom, Jth local node of element NEL |
| IELNO(N) | = element number of Nth applied surface force |
| ISIDE(N) | = element-side number of Nth applied surface force |
| PRESS(1,N) | = pressure at local node I for Nth applied surface force [†] |
| PRESS(2,N) | = pressure at local node J for Nth applied surface force [†] |
| SHEAR(1,N) | = shear at local node I for Nth applied surface force [†] |
| SHEAR(2,N) | = shear at local node J for Nth applied surface force [†] |
| ISHIST(1,N) | = element number for output history N |
| ISHIST(2,N) | = quadrature-point number for output history N |
| ISHIST(3,N) | = stress/strain-component number for output history N |
| SOUT(1,J) | = time at plot step J (stored as single precision for IPREC = 1 or 2) |
| SOUT(N+1,J) | = output history N at plot step J (stored as single precision for IPREC = 1 or 2) |
| ELEFFM(I,J) | = element effective mass matrix |
| XL(I,J) | = Ith coordinate component of Jth element node |
| WORK(I) | = work space used for temporary calculations |
| B(I,J) | = strain-displacement matrix |
| DMAT(I,J) | = scalar multiple of element material moduli matrix |
| DB(I,J) | = matrix product of DMAT and B |
| VL(I,J) | = Ith velocity component of Jth element node |
| AL(I,J) | = Ith acceleration component of Jth element node |
| ELRESF(I) | = Ith entry of element residual force vector |
| DL(I,J) | = Ith displacement component of Jth element node |
| STRAIN(I) | = strains |
| STRESS(I) | = stresses |
| PSTRN(I) | = principal strains |

[†]For the four-node quadrilateral, $1 \leq I \leq 4$; if $I \leq 3$, $J = I + 1$; if $I = 4$, $J = 1$.

TABLE 11.2.4 (cont.) Storage Requirements for Four-Node Quadrilateral, Elastic Continuum Element (NTYPE = 1)

| | |
|--------------------|---|
| PSTRS(I) | = principal stresses |
| NINT | = number of integration points |
| NUMAT | = number of materials in this group |
| NROWSH | = number of rows in shape-function arrays (=3) |
| NEN | = number of element nodes (=4) |
| NRINT | = number of reduced-integration points; for mean-dilatational \bar{B} formulation = 1 |
| NROWB [‡] | = number of rows in strain-displacement matrix (=4) |
| NUMEL | = number of elements in this group |
| NSURF | = number of element surface loads in this group |
| NSOUT | = number of stress/strain time-histories in this group |
| NPLPTS | = number of points in output time-histories. |
| NEE | = number of element equations (=8) |
| NED | = number of element degrees of freedom per node (=2) |

[‡]Note that while the strain-displacement matrix B has a constant row dimension of 4, the parameter NSTR specified in SUBROUTINE QUADC denotes the number of nonzero strain-energy components in the element. For plane stress (IOPT = 0), and plane strain (IOPT = 1) without mean-dilatational \bar{B} formulation (IBBAR = 0), NSTR equals 3. However, for plane strain with \bar{B} (IBBAR = 1), and torsionless axisymmetry (IOPT = 2) with or without \bar{B} , NSTR equals 4.

TABLE 11.2.5 Storage in Blank Common—Equation System Data*

| Array Pointer | Array |
|---------------|--|
| MPALHS | ALHS(NALHS) |
| MPBRHS | BRHS(NEQ) |
| where: | |
| ALHS(N) | = Nth term of coefficient matrix stored in compacted column form |
| BRHS(N) | = Nth active residual (dynamic analysis) or force (static analysis) component [†] |
| NALHS | = number of terms in ALHS |
| NEQ | = number of equations |

* Required for static and dynamic analysis; storage allocated in SUBROUTINE EQSET.

[†]Upon back substitution BRHS contains acceleration increments (dynamic analysis) or displacements (static analysis) for active degrees of freedom.

1.3 PROGRAM STRUCTURE

In this section we present an organizational description of DLEARNS, including algorithmic equations when they are deemed illustrative. All FORTRAN subroutines appear in boldface and variable names are capitalized. In the example element loops, the term “generic subroutine” identifies a fundamental program unit which could be accessed by any element type, e.g., **COLHT**, or by a class of elements, e.g., **PROP2D** for two-dimensional, elastic continuum elements.

11.3.1 Global Control

G1. MAIN PROGRAM

1. Set blank common capacity MTOT, and initialize addresses of first and last available words.
2. Specify numerical precision IPREC.
3. Specify input/output unit numbers and open files.
4. Call global driver DLEARN.
5. Close files and exit program.

G2. DLEARN: Global driver

1. **ECHO:** IF (IECHO.EQ.1) write copy of input data (uninterpreted) to output file.
2. Read TITLE: IF (TITLE(1).EQ.'*END') RETURN to G1.5.
3. Read execution control card. If static analysis LDYN = .FALSE., otherwise .TRUE. .
4. Call initialization drivers, GO TO I1.
5. IF (IEXEC.EQ.1) **DRIVER:** solution driver, GO TO S1.
6. IF (IWRITR.EQ.1) **RSOUT:** write restart file.
7. **PRTDC:** print memory pointer dictionary.
8. GO TO 2 (begin new problem).

11.3.2 Initialization Phase

I1. TSEQ: driver to input time sequence control data.

1. Set memory pointers for time sequence data.
2. **TSEQIN:** read time sequence data and compute storage for time histories. Set defaults for static analysis.
3. Set memory pointers for nodal time history data arrays.

I2. IF (LDYN) DYNPTS: set pointers for dynamic analysis data arrays.

I3. IF (NDOUT.GT.0) DHIST: read nodal output time-history data, store in IDHIST(3,NDOUT).

I4. STATIN: driver to set pointers for static analysis data arrays and read geometric and boundary condition input data. IF (.NOT.LDYN) MPDPRD = MPD, then element task 'FORM_RHS' accesses proper displacement array during static analysis. (cf. S10.1.a).

1. **COORD:** read/generate nodal coordinates, store in X(NSD,NUMNP).

2. **BC:** read/generate nodal boundary condition codes into ID(NDOF,NUMNP), then overwrite array with equation numbers.
 3. **IF (NLVECT.GT.0) INPUT:** read/generate prescribed nodal force/kinematic boundary condition data and store in F(NDOF, NUMNP, NLVECT).
 4. **IF (NLTFTN.GT.0) LTIMEF:** read load-time functions, store in G(NPTSLF,2,NLTFTN).
- 15. IF (LDYN):** input kinematic initial conditions.
1. **IF (IREADR .EQ. 1) THEN**
 - RSIN:** read initial conditions from restart file.
 - ELSE**
 - INPUT:** read initial conditions from input file.
 - ENDIF**
- 16. IF (LDYN .AND. NDOUT.GT.0) STORED:** store kinematic initial conditions for time histories in DOUT(NDOUT+1,NPLPTS).
- 17. ELEMNT('INPUT__'):** loop over element groups NEG=1,NUMEG and read/generate element definition data; element task 1.
-

Example

QUADC: four-node quadrilateral, elastic continuum element.

1. Set memory pointers.
 2. **QDCT1:** element task 1.
 - a. **QDCSHL:** compute bilinear shape functions N_a and their local derivatives $N_{a,\xi}$, $N_{a,\eta}$ at integration points, store in SHL(NROWSH,NEN,NINT); generic subroutine.
 - b. **PROP2D:** read material property data for two-dimensional elastic continuum. Compute material moduli matrix, store in C(NROWB,NROWB, NUMAT); generic subroutine.
 - c. Read body-force vector GRAV(NESD).
 - d. **GENEL:** read/generate element data; store element node numbers in IEN(NEN,NUMEL) and material set numbers in MAT(NUMEL).
 - e. **FORMLM:** form assembly mapping LM(NED,NEN,NUMEL); generic subroutine.
 - f. If element group contributes to global left-hand-side matrix, then **COLHT:** determine column heights of each element; generic subroutine.
 - g. **IF (NSURF.GT.0) QDCRSF:** read surface force data.
 - h. **IF (NSOUT.GT.0) SHIST:** read stress/strain output time-history data, store in ISHIST(3,NSOUT); generic subroutine.
-

I8. EQSET: complete specification of data structure for global equation system.

1. **DIAG:** determine addresses of diagonal entries of global left-hand-side matrix ALHS, store in IDIAG(NEQ). Set NALHS equal to number of terms in ALHS.
2. Allocate memory for ALHS(NALHS) and global right-hand-side vector BRHS(NEQ).

I9. Resume global driver DLEARN; GO TO G2.5.

11.3.3 Solution Phase

Time sequence loop (DO S19 NSQ = 1,NUMSEQ)

S1. TIMCON: set current time sequence parameters: NSTEP1, NDPRT1, NSPRT1, NHPLT1, NITER1, ALPHA1, BETA1, GAMMA1, DT1 and compute coefficients for α -method transient algorithm:

$$\begin{aligned} \text{COEFF1} &= (1 + \alpha), \\ \text{COEFF2} &= \gamma\Delta t, \\ \text{COEFF3} &= \beta\Delta t^2 \\ \text{COEFF4} &= (1 + \alpha)\gamma\Delta t, \\ \text{COEFF5} &= (1 + \alpha)\beta\Delta t^2, \\ \text{COEFF6} &= (1 + \alpha)\Delta t, \\ \text{COEFF7} &= \frac{1}{2}(1 + \alpha)(1 - 2\beta)\Delta t^2, \\ \text{COEFF8} &= (1 + \alpha)(1 - \gamma)\Delta t. \end{aligned}$$

S2. ELEMNT('FORM_LHS'): loop over element groups NEG=1,NUMEG and form effective mass matrix M^* ; element task 2.

$$M^* = M^E + M^I + (1 + \alpha)\gamma\Delta t C^I + (1 + \alpha)\beta\Delta t^2 K^I \quad (\text{dynamic analysis})$$

$$M^* = K \quad (\text{static analysis})$$

where superscripts E and I denote explicit and implicit element group contributions, respectively.

Example

QUADC: four-node quadrilateral, elastic continuum element.

1. **QDCT2:** element task 2.

```

DO h NEL=1,NUMEL
  a. CLEAR: zero element effective mass array ELEFFM(NEE,NEE).
  b. LOCAL: localize nodal coordinates X, store in XL(NESD,NEN); generic
    subroutine.
  c. QDCSHG: compute shape function global derivatives  $N_{a,x}$ ,  $N_{a,y}$  at integration
    points, store in SHG(NROWSH,NEN,NINT); generic subroutine.
  d. IF (IOPT.EQ.2), compute radial coordinates of integration points for
    torsionless axisymmetry option, store in R(NINT). Include radial weighting
    with Jacobian determinants DET(NINT).
  e. IF (LDYN .AND. IMASS.NE.2) THEN
    CONTM: form consistent or lumped mass matrix for continuum
    element and store in ELEFFM(NEE,NEE); generic subroutine.
     $m^* = m^* + [1 + RDAMPM(MAT(NEL))*(1 + \alpha)\gamma\Delta t]m^*$ 
  ENDIF
  f. IF ((NOT.LDYN) .OR. IMPEXP.EQ.0) THEN
    QDCK: form stiffness matrix  $k^* = \int B^T D B d\Omega$  and sum to
    ELEFFM(NEE,NEE).
     $m^* = m^* + [RDAMPK(MAT(NEL))*(1 + \alpha)\gamma\Delta t + (1 + \alpha)\beta\Delta t^2]k^*$ 
  ENDIF
  g. ADDLHS: assemble ELEFFM into ALHS; generic subroutine.
  h. CONTINUE to next element.

```

S3. FACTOR: perform Crout factorization of ALHS.

$$M^* = U^T D U$$

S4. Perform rank check if requested.

1. IF (IRANK.EQ.1) PIVOTS: print numbers of negative and zero pivots; RETURN to S19.
2. IF (IRANK.EQ.2) PRINTP: print all pivots; RETURN to S19.

Time-step loop (DO S18 N=1,NSTEP1)

S5. IF (LDYN) THEN

PREDCT: predictor update of all degrees of freedom.

$$\tilde{d}^{(1)} = d_n + (1 + \alpha)\Delta t v_n$$

$$+ \frac{1}{2}(1 + \alpha)(1 - 2\beta)\Delta t^2 a_n \quad DPRED = D + COEFF6 * V + COEFF7 * A$$

$$\tilde{v}^{(1)} = v_n + (1 + \alpha)(1 - \gamma)\Delta t a_n \quad VPRED = V + COEFF8 * A$$

$$\tilde{a}^{(1)} = 0 \quad A = ZERO$$

ELSE

CLEAR: zero displacement array D.

ENDIF

S6. IF (NLVECT.GT.0) LFAC: evaluate load-time functions at time t_{n+1} , store values in G1(NLTFTN).

S7. IF (DT1.NE.ZERO) COMPBC: overwrite predictors to account for kinematic boundary conditions.

1. If d_{n+1} given:

IF (LDYN) THEN

$$d^* = (1 + \alpha)d_{n+1} - \alpha d_n \quad TEMP = COEFF1 * VAL - ALPHA1 * D$$

$$\tilde{a}^{(1)} = \frac{d^* - \tilde{d}^{(1)}}{(1 + \alpha)\beta\Delta t^2} \quad A = (TEMP - DPRED)/COEFF5$$

$$\tilde{d}^{(1)} = d^* \quad DPRED = TEMP$$

$$\tilde{v}^{(1)} = \tilde{v}^{(1)} + (1 + \alpha)\gamma\Delta t \tilde{a}^{(1)} \quad VPRED = VPRED + COEFF4 * A$$

ELSE

$$\tilde{d} = d_{n+1} \quad DPRED = VAL$$

ENDIF

2. If v_{n+1} given:

$$v^* = (1 + \alpha)v_{n+1} - \alpha v_n \quad TEMP = COEFF1 * VAL - ALPHA1 * V$$

$$\tilde{a}^{(1)} = \frac{v^* - \tilde{v}^{(1)}}{(1 + \alpha)\gamma\Delta t} \quad A = (TEMP - VPRED)/COEFF4$$

$$\tilde{v}^{(1)} = v^* \quad VPRED = TEMP$$

$$\tilde{d}^{(1)} = \tilde{d}^{(1)} + (1 + \alpha)\beta\Delta t^2 \tilde{a}^{(1)} \quad DPRED = DPRED + COEFF5 * A$$

3. If a_{n+1} given:

$$\tilde{d}^{(1)} = \tilde{d}^{(1)} + (1 + \alpha)\beta\Delta t^2 a_{n+1} \quad DPRED = DPRED + COEFF5 * VAL$$

$$\tilde{v}^{(1)} = \tilde{v}^{(1)} + (1 + \alpha)\gamma\Delta t a_{n+1} \quad VPRED = VPRED + COEFF4 * VAL$$

$$\tilde{a}^{(1)} = a_{n+1} \quad A = VAL$$

Multicorrector iteration loop (DO S13 I=1,NITER1)

S8. IF (NLTFTN.GT.0) LFAC: evaluate load-time functions at time $t_{n+1+\alpha}$, store values in G1(NLTFTN).

S9. IF (NLVECT.GT.0) LOAD: form nodal contribution to residual force, $\mathbf{F}_{n+1+\alpha}$, store in BRHS(NEQ).

S10. ELEMNT('FORM_RHS'): loop over element groups NEG=1,NUMEG and form element contributions to residual force \mathbf{R} ; element task 3.

$$\mathbf{R} = \underbrace{\mathbf{F}_{n+1+\alpha}}_{\text{Nodal contribution}} - \underbrace{\mathbf{M}\tilde{\mathbf{a}}^{(i)} - \mathbf{C}\tilde{\mathbf{v}}^{(i)} - \mathbf{K}\tilde{\mathbf{d}}^{(i)}}_{\text{Element contribution}} + \underbrace{\sum_{e=1}^{n_{el}} \mathbf{f}_e^{n+1+\alpha}}_{\text{Element contribution}} \quad (\text{dynamic analysis})$$

$$\mathbf{R} = \underbrace{\mathbf{F}}_{\text{Nodal contribution}} + \underbrace{\sum_{e=1}^{n_{el}} \mathbf{f}_e^e}_{\text{Element contribution}} \quad (\text{static analysis})$$

Example

QUADC: four-node quadrilateral, elastic continuum element.

1. QDCT3: element task 3.

DO h NEL=1,NUMEL

- a. **LOCAL:** localize displacements DPRED, store in DL(NED,NEN); generic subroutine. Note: *For static analysis MPDPRD = MPD, so reference to DPRED accesses the array D.*
- b. IF (LDYN) THEN

- i. **LOCAL:** localize VPRED and A, store in VL(NED,NEN) and AL(NED,NEN), respectively; generic subroutine.

- ii. Compute effective displacement and acceleration accounting for Rayleigh damping:

$$\begin{aligned}\tilde{\mathbf{d}}^e &= \tilde{\mathbf{d}}^e + RDAMPK * \tilde{\mathbf{v}}^e, \\ \tilde{\mathbf{a}}^e &= \tilde{\mathbf{a}}^e + RDAMPM * \tilde{\mathbf{v}}^e.\end{aligned}$$

ENDIF

- c. Determine if element makes inertial (FORMMA = .TRUE.) and/or stiffness (FORMKD = .TRUE.) contribution to right-hand-side vector.
- d. IF (FORMMA .OR. FORMKD) THEN

- i. **LOCAL:** localize nodal coordinates X, store in XL(NESD,NEN); generic subroutine.

- ii. **QDCSHG:** compute shape function global derivatives $N_{a,x}$, $N_{a,y}$ at integration points, store in SHG(NROWSH,NEN,NINT); generic subroutine.

- iii. IF (IOPT.EQ.2) compute radial coordinates of integration points for torsionless axisymmetry option, store in R(NINT). Include radial weighting with Jacobian determinants DET(NINT).
- ENDIF
- e. IF (FORMMA) CONTMA: compute $-m^\epsilon(\tilde{a}^\epsilon - g^\epsilon)$ for continuum element and store in ELRESF(NEE); generic subroutine.
- f. IF (FORMKD) QDCKD: compute internal force $-k^\epsilon \tilde{d}^\epsilon = -\int B^T \sigma d\Omega$, and add to ELRESF(NEE).
- g. ADDRHS: assemble ELRESF into BLHS; generic subroutine.
- h. CONTINUE to next element.
- i. IF (NSURF.GT.0) QDCSUF: compute element surface forces and assemble into BRHS.
-

S11. BACK: perform forward reduction, diagonal scaling, and back substitution to find $\Delta a^{(i)}$ (dynamic analysis) or $d^{(i)}$ (static analysis).

S12. ITERUP: intermediate update of active degrees of freedom.

IF (LDYN) THEN

$$\begin{aligned}\tilde{d}^{(i+1)} &= \tilde{d}^{(i)} + (1 + \alpha)\beta\Delta t^2\Delta a^{(i)} & DPRED &= DPRED + COEFF5 * BRHS \\ \tilde{v}^{(i+1)} &= \tilde{v}^{(i)} + (1 + \alpha)\gamma\Delta t\Delta a^{(i)} & VPRED &= VPRED + COEFF4 * BRHS \\ \tilde{a}^{(i+1)} &= \tilde{a}^{(i)} + \Delta a^{(i)} & A &= A + BRHS\end{aligned}$$

ELSE

$$d = d^{(i)} \quad D = BRHS$$

ENDIF

S13. CONTINUE to next iteration (GO TO S8).

S14. IF (LDYN) CORRCT: corrector update of all degrees of freedom.

$$\begin{aligned}d_{n+1} &= \frac{\tilde{d}^{(i+1)} - d_n}{1 + \alpha} + d_n, & D &= (DPRED - D)/COEFF1 + D \\ v_{n+1} &= \frac{\tilde{v}^{(i+1)} - v_n}{1 + \alpha} + v_n & V &= (VPRED - V)/COEFF1 + V \\ a_{n+1} &= \tilde{a}^{(i+1)}\end{aligned}$$

S15. IF (LOUT(N,NDPRT1)) PRINTD: write kinematic output.

S16. IF (LOUT(N,NSPRT1)) ELEMNT('STR_PRNT'): loop over element groups NEG=1,NUMEG and perform stress recovery for printing; element task 4.

Example

QUADC: four-node quadrilateral, elastic continuum element.

1. **QDCT4:** element task 4.

DO h NEL=1,NUMEL

- a. **LOCAL:** localize coordinates X and displacements D, store in XL(NESD,NEN) and DL(NED,NEN), respectively; generic subroutine.
- b. **QDCSHG:** compute shape function global derivatives $N_{a,x}, N_{a,y}$ at integration points, store in SHG(NROWSH,NEN,NINT); generic subroutine.
- c. Compute coordinates of integration points, store in XINT(NESD,NINT). IF (IOPT.EQ.2) set R(L) = XINT(1,L) and include radial weighting in Jacobian determinants DET(NINT).
- d. IF (IBBAR.EQ.1) **MEANSH:** compute volume average of global shape function derivatives for mean-dilatational \bar{B} ; IF (IOPT.EQ.2), compute volume average of shape functions; store in SHGBR(NROWSH,NEN).

DO g L=1,NINT

- e. **QDCSTR:** compute stress and strain for two-dimensional continuum element, store in STRESS(NROWB) and STRAIN(NROWB), respectively.
 - f. **PRTS2D:** print stresses and strains for two-dimensional element; generic subroutine.
 - g. **CONTINUE** to next quadrature point.
 - h. **CONTINUE** to next element.
-

S17. IF (LDYN .AND. LOUT(N,NHPLT1)) THEN

1. **STORED:** store kinematic time history data in DOUT(NDOUT+1,NPLPTS).
 2. **ELEMNT('STR_STOR'):** loop over element groups NEG=1,NUMEG and perform stress recovery for element time-histories; element task 5.
-

Example

QUADC: four-node quadrilateral, elastic continuum element.

1. **QDCT5:** element task 5.

- a. Store current time value in SOUT(1,LOCPLT).

- DO h I=1,NSOUT
- b. For active element **LOCAL**: localize coordinates X and displacements D, store in XL(NESD,NEN) and DL(NED,NEN), respectively; generic subroutine.
 - c. **QDCSHG**: Compute shape function global derivatives $N_{a,x}$, $N_{a,y}$ at integration points, store in SHG(NROWSH,NEN,NINT); generic subroutine.
 - d. IF (IOPt.EQ.2) compute radial coordinates of integration points, store in R(NINT). Include radial weighting in Jacobian determinants DET(NINT).
 - e. IF (IBBAR.EQ.1) **MEANSH**: compute volume average of global shape function derivatives for mean-dilatational \bar{B} ; IF (IOPt.EQ.2), compute volume average of shape functions; store in SHGBR(NROWSH,NEN).
 - f. For active integration point **QDCSTR**: compute stress and strain for two-dimensional continuum element, store in STRESS(NROWB) and STRAIN(NROWB), respectively.
 - g. Store required stress/strain component in SOUT(I+1,LOCPLT).
 - h. CONTINUE to next active element/quadrature point pair.
-

ENDIF

S18. CONTINUE to next time step (GO TO S5).

S19. CONTINUE to next time sequence (GO TO S1).

S20. IF (LDYN) THEN

1. **HPLOT**: plot nodal time-histories.
2. **ELEMNT('STR_PLOT')**: loop over element groups NEG=1,NUMEG and plot element time-histories with **HPLOT**; element task 6.

ENDIF

S21. RETURN to **DLEARN** (GO TO G2.6).

11.4 ADDING AN ELEMENT TO DLEARN

Example

Suppose we wish to add a shell element to the program. Let us call the subroutine that sets element group parameters and allocates blank common storage **SUBROUTINE SHELL**. The steps are as follows:

1. Study and understand thoroughly the subroutines QUADC and TRUSS before beginning. Notice the great similarity in their structure. Although a two-node truss

could be programmed more explicitly, i.e., in the spirit of matrix structural analysis, we feel the present implementation emphasizes more generalized finite element methodologies.

2. When studying the elements QUADC and TRUSS, note that the elements are broken into six calls to different subroutines; the variable ITASK defines which routine will be called. In the global routines the call to ELEMNT defines the value of ITASK. For example, in DRIVER, the statement CALL ELEMNT ('FORM_LHS') results in ITASK having the value 2 (see routine ELEMNT). In the routine QUADC, if ITASK has the value 2, the routine QDCT2 is executed. Therefore, to make any new element compatible with the "global element functions," the new element should be designed with these six functions in mind. However, if the user is extending the capabilities of DLEARN, he or she is totally free to define additional element functions. The six current functions are defined as follows:

1. 'INPUT____' : Read/generate element input data.
2. 'FORM_LHS' : Compute element contribution to global left-hand-side matrix.
3. 'FORM_RHS' : Compute element contribution to global right-hand-side vector.
4. 'STR_PRNT' : Compute and print element stress/strain output.
5. 'STR_STOR' : Compute element stress/strain for output time histories.
6. 'STR_PLOT' : Plot element time histories.

As an example, we present an outline of SUBROUTINE QUADC.

```

SUBROUTINE QUADC(ITASK,NPAR,MP,NEG)
PROGRAM TO SET STORAGE AND CALL TASKS FOR THE
FOUR-NODE QUADRILATERAL, ELASTIC CONTINUUM ELEMENT
*
*
DEFINE STORAGE POINTER NAMES
*
*
EQUIVALENCE GROUP PARAMETERS FROM NPAR ARRAY
*
*
SET ELEMENT PARAMETERS
*
*
IF (ITASK.EQ.1) THEN
.. SET MEMORY POINTERS
*
*
ENDIF *
TASK CALLS
IF (ITASK.GT.6) RETURN
GO TO (100,200,300,400,500,600),ITASK

```

```

C 100 CONTINUE
C.... INPUT ELEMENT DATA ('INPUT__')
C      CALL QDCT1(A(MP(MSHL )),A(MP(MW    )),A(MP(MRH0  )),
C                  *
C      *
C      RETURN
C 200 CONTINUE
C.... FORM ELEMENT EFFECTIVE MASS AND ASSEMBLE INTO GLOBAL
C      LEFT-HAND-SIDE MATRIX ('FORM_LHS')
C      CALL QDCT2(A(MP(MELEFM)),A(MP(MIEN )),A(MPX      ),
C                  *
C                  *
C      RETURN
C 300 CONTINUE
C.... FORM ELEMENT RESIDUAL-FORCE VECTOR AND ASSEMBLE INTO GLOBAL
C      RIGHT-HAND-SIDE VECTOR ('FORM_RHS')
C      CALL QDCT3(A(MP(MMAT )),A(MP(MIEN )),A(MPDPRD  ),
C                  *
C                  *
C      RETURN
C 400 CONTINUE
C.... CALCULATE AND PRINT ELEMENT STRESS/STRAIN OUTPUT ('STR_PRNT')
C      IF (ISTPRT.EQ.0)
C      & CALL QDCT4(A(MP(MMAT )),A(MP(MIEN )),A(MPD      ),
C                  *
C                  *
C      RETURN
C 500 CONTINUE
C.... CALCULATE AND STORE ELEMENT TIME-HISTORIES ('STR_STORE')
C      IF (NSOUT.GT.0)
C      & CALL QDCT5(A(MP(MISHST)),A(MP(MSOUT )),A(MP(MMAT  )),
C                  *
C                  *
C      RETURN
C 600 CONTINUE
C.... PLOT ELEMENT TIME-HISTORIES ('STR_PLOT')
C      IF (NSOUT.GT.0)
C      & CALL HPLOT(A(MP(MISHST)),A(MP(MSOUT )),NSOUT ,3,NTYPE )
C      RETURN
C      END

```

3. When coding SHELL and its task routines, use existing utility routines, such as MULTAB, whenever possible.²
4. Code a new version of subroutine ELMLIB to include a call to SHELL. The necessary modifications are indicated:

²The experienced programmer will recognize that the use of utility routines such as MULTAB is highly inefficient. Nevertheless, this approach facilitates implementation and results in very readable coding. Significant gains in efficiency can be made by avoiding subroutine calls and explicitly writing out the matrix products line by line, taking account of zeros and repeated calculations. See Sec.3.10.

```

GO TO (100,200,[300]),NTYPE
C
100 CALL QUADC(ITASK,A(MPNPAR),A(MPNPAR+16),NEG)
      RETURN
C
200 CALL TRUSS(ITASK,A(MPNPAR),A(MPNPAR+16),NEG)
      RETURN

C
300 CALL SHELL(ITASK,A(MPNPAR),A(MPNPAR+16),NEG)
      RETURN

```

5. Define an array LABEL3 in BLOCK DATA and specify the desired labels for stress, strain, and any other time-histories. Include the appropriate WRITE statements in HPLOT and SHIST.
 6. Write a description for the Input Instructions to be included in Sec.In.10, ELEMENT DATA. Mimic the style of Secs. In.10.1 and In.10.2, and use the same variable names as far as possible.
-

Instructions for Debugging³ and Running the New Element

1. Compile all your new routines separately before including them in DLEARN.
2. Link all routines compiled and execute with simple, static, one-element problems (e.g., homogeneous deformation states), which are easy to check. Check all options (e.g., gravity loads, etc.) on the one-element problems.
3. Next, proceed to “patch tests,” which also should be easy to check. Finally, after you are convinced your element is working properly, try some interesting static and dynamic test problems for which you have exact analytical, or reliable experimental data. Assess the accuracy of the element you have programmed.

I.5 DLEARN USER'S MANUAL

11.5.1 Remarks for the New User

This section presents the necessary input instructions for the user to execute DLEARN. The new user is advised to read *all* the instructions, paying particular

³ Debugging is boring and frustrating, though the use of interactive debug utilities can reduce the tedium. When you are feeling particularly depressed by this experience, remember *everyone* who has ever programmed has experienced what you are feeling now. Misery loves company.

attention to the appended notes, to gain an initial overview of the data required to specify an analysis task to DLEARN. Readers without prior finite element computational experience will benefit from careful study of the example problems; they are intended to help clarify the use of the data generation features. The amount of data required to specify even a small problem can become burdensome if the user does not learn to exercise these capabilities.

Upon examining the input instructions, the reader should recognize that no system of units is assumed by the program. Rather, *it is the user's responsibility to specify data in the consistent units of his or her choice*. Overlooking this requirement is a common source of erroneous results, especially in dynamic analysis.

It is often the case that new users have difficulty in getting a program to successfully run. It is tempting to assume that there are "bugs" in the program and that the fault does not lie with oneself. DLEARN has undergone extensive testing to minimize the possibility of any "bugs" being present. Thus, the new user should normally avoid this alibi, for the vast majority of aborted runs are the result of input specification errors. Train yourself to look closely at generation sequences; a single slipped digit can lead to unbelievably chaotic input data and frequently the legendary "floating point overflow/divide by zero."

Unfortunately, input errors need not be this apparent, though nonetheless catastrophic. For example, an error in generating initial displacements for a dynamic analysis may lead INPUT to write beyond the storage allocated for the D array. The D array is followed in COMMON A(MTOT) by the nodal coordinates array X(NSD,NUMNP), thus INPUT would unknowingly overwrite the coordinate data. This may later lead to an error termination, e.g., calculation of a nonpositive Jacobian determinant in an element or the successful solution of the *wrong* initial/boundary-value problem. All this would occur after the coordinate data are written to the output file, making detection difficult. This can be a frustrating task for the new user. However, with a little experience, efficiency in recognizing errors will be gained. Take heart, even the most seasoned analyst occasionally wastes time (and money) searching for such anomalies.

11.5.2 Input Instructions

In.0.0 Input Data Echo Card⁴ (I5)

| Note | Columns | Variable | Description |
|------|---------|----------|---|
| (1) | 1-5 | IECHO | Input file data echo EQ.0, no data echo EQ.1, data echo |

Notes

1. Data from input file are read and written, without interpretation, at beginning of output file before execution continues.

⁴ Recognizing the preponderance of interactive computing today, the term "card" as used here refers to an 80-character line in the input file.

In.1.0 Title Card (20A4)

| Note | Columns | Variable | Description |
|------|---------|-----------|----------------------------------|
| (1) | 1–80 | TITLE(20) | Job title for heading the output |

Notes

1. Each data set begins with a title card. *To terminate execution, the last card in the input file, which will be read as a new title card, must contain the characters *END in the first four columns.*

In.2.0 Execution Control Card (15/5)

| Note | Columns | Variable | Description |
|------|---------|----------|---|
| (1) | 1–5 | IEXEC | Execution code EQ.0, data check only EQ.1, execution |
| (2) | 6–10 | IACODE | Analysis code EQ.0, dynamic analysis EQ.1, static analysis |
| (3) | 11–15 | IREADR | Read restart file code EQ.0, do not read restart file EQ.1, read restart file |
| (4) | 16–20 | IWRITR | Write restart file code EQ.0, do not write restart file EQ.1, write restart file |
| (5) | 21–25 | IPRTIN | Input data print code EQ.0, print nodal and element input data EQ.1, do not print nodal and element input data |
| (6) | 26–30 | IRANK | Rank check code EQ.0, do not perform rank check EQ.1, print numbers of zero and negative pivots EQ.2, print all pivots |
| (7) | 31–35 | NUMSEQ | Number of time sequences, GE.1 |
| (8) | 36–40 | NDOUT | Number of nodal output time histories |
| (9) | 41–45 | NSD | Number of space dimensions |
| | 46–50 | NUMNP | Number of nodal points |
| (10) | 51–55 | NDOF | Number of nodal degrees of freedom |
| (11) | 56–60 | NLVECT | Number of load vectors |
| (12) | 61–65 | NLTFTN | Number of load-time functions |
| (13) | 66–70 | NPTSLF | Number of points defining a load-time function |
| (14) | 71–75 | NUMEG | Number of element groups |

Notes

1. In the data check mode, input data are printed and storage requirements are indicated. This mode should be employed prior to making expensive calculations.
2. A static analysis consists of solution of the equilibrium equations, resulting in displacement and stress output. A dynamic analysis consists of time-integration of the semi-discrete equations of motion, resulting in displacement, velocity, acceleration, and stress output.
3. If IREADR .EQ. 1, then initial displacement, velocity, and acceleration data will be read from the file associated with FORTRAN unit IRSIN at the beginning of a dynamic analysis (i.e. IACODE .EQ. 0).
4. If IWWRITR .EQ. 1, then final displacement, velocity, and acceleration data will be written to the file associated with FORTRAN unit IRSOUT at the conclusion of a dynamic analysis (i.e., IACODE .EQ. 0).
5. If IPRTIN .EQ. 1, then only control input data, e.g., time-sequence data, is printed in the output file. This option is useful in reducing output for very large amounts of generated input data. However, it should only be employed after thorough checking of data has been performed prior to execution.
6. A rank check consists of factorization of the effective mass matrix, $M^* = U^T D U$, and either printing the numbers of zero and negative entries along the diagonal of D (IRANK .EQ. 1), or printing the entire diagonal of D (IRANK .EQ. 2), for each time sequence. The number of zero-energy modes can be deduced from the number of zeroes along the diagonal of D . This option allows for the checking of elements and meshes suspected of containing spurious zero-energy modes, such as those arising in so-called "under-integrated" finite elements. If IRANK .GT. 0, then no other executable calculations are performed (i.e., dynamic or static analysis).
7. For static analysis, a single time sequence should be adequate, although the use of multiple sequences is not prohibited. A dynamic analysis can be subdivided into multiple contiguous time intervals in which different time-integration constants and output parameters may be specified. See Sec. In.3.0.
8. Output histories are ignored in static analysis as denoted by IACODE .EQ. 1. See Sec. In.4.0.
9. Consult individual element routines in Sec. In.10.0 for requirements.
10. In problems for which there are different numbers of degrees of freedom at different nodes, NDOF should be set equal to the maximum number. Consult individual element routines in Sec. In.10.0 for requirements. Superfluous degrees of freedom can be eliminated by employing boundary condition codes. See Sec. In.6.0.
11. The prescribed nodal forces and kinematic boundary conditions are synthesized by NLVECT vectors and corresponding load-time functions. Details are presented in Secs. In.7.0 and In.8.0.
12. The user must define NLTFN .GE. NLVECT. The first NLVECT load-time functions scale the corresponding load vectors. Additional load-time functions may be defined to scale element surface and body forces specified in the element group data.
13. The load-time functions are discretized by NPTSLF points. See Sec. In.8.0 for details.
14. The elements are read in groups.

In.3.0 Time Sequence Data Cards (6I5,4F10.0)

| Note | Columns | Variable | Description |
|------|---------|----------|--|
| (1) | 1–5 | N | Sequence number |
| | 6–10 | NSTEP(N) | Number of time steps for sequence N |
| (2) | 11–15 | NDPRT(N) | Print kinematic output every NDPRT(N) time steps |
| (2) | 16–20 | NSPRT(N) | Print stress/strain output every NSPRT(N) time steps |
| (2) | 21–25 | NHPLT(N) | Plot time history data every NHPLT(N) time steps |
| | 26–30 | NITER(N) | Number of iterations in corrector loop; GE. 1 |
| (3) | 31–40 | ALPHA(N) | α parameter for sequence N, LE. 0.0 |
| (3) | 41–50 | BETA(N) | β parameter for sequence N, GT. 0.0 |
| (3) | 51–60 | GAMMA(N) | γ parameter for sequence N, GE. 0.0 |
| (3) | 61–70 | DT(N) | Time step Δt for sequence N |

Notes

1. NUMSEQ time sequence data cards must be defined but need not be input in any particular order. NUMSEQ .GE. 1 is defined on the Execution Control Card; see Sec. In.2.0.
2. If set Eq. 0, no output. In each sequence, the final print/plot state is output at *sequence* time step $\text{NSTEP}(N) - \text{MOD}(\text{NSTEP}(N), \text{NDPRT}(N))$, etc., and the output logic will reinitialize at the beginning of the following sequence.
3. Many different transient algorithms can be realized by the proper choice of parameters α , β , and γ . The choice of $\alpha \in [-\frac{1}{3}, 0]$, $\beta = \frac{1}{4}(1 - \alpha)^2$, $\gamma = \frac{1}{2}(1 - 2\alpha)$ leads to a second-order accurate method, which is unconditionally stable if *all element groups are specified to be implicit*. The user should be cognizant of any stability restrictions upon the time step size Δt ; *such limitations can be severe when using explicit element groups*. To compute the consistent initial acceleration for all *active* degrees of freedom, specify the first time sequence to consist of one time-step with $\alpha = 0$ and $\Delta t = 0$. The code will then solve $M\Delta a = F_0 - Ma_0 - Cv_0 - Kd_0$ and update: $a_0 = a_0 + \Delta a$, $v_0 = v_0$, and $d_0 = d_0$. The values of β and γ are irrelevant, but the user should specify $\text{NDPRT} = \text{NSPRT} = \text{NHPLT} = 1$ if he or she wishes to examine the results of this calculation. Note that there is no rational way to *calculate* the initial acceleration for any node controlled by kinematic boundary conditions (see Sec. In.6.0): These data must be specified by the user (See Sec. In.9.0).

For *static analysis*, the following defaults are set internally:

$$\text{NSTEP}(N) = \text{MAXO}(1, \text{NSTEP}(N))$$

$$\text{NDPRT}(N) = 1 \quad \alpha = 0.0$$

$$\text{NSPRT}(N) = 1 \quad \beta = 1.0$$

$$\text{NHPLT}(N) = 0 \quad \gamma = 0.0$$

$$\text{NITER}(N) = 1 \quad \Delta t = 1.0$$

In.4.0 Nodal History Data Cards (3/5)

If there are no nodal output time-histories (i.e., NDOUT .EQ. 0 on the Execution Control Card; see Sec. In.2.0), then no nodal history data cards should be input; proceed directly to Sec. In.5.0.

If the analysis is static (i.e., IACODE .EQ. 1 on the Execution Control Card), then output history cards are read, but subsequently ignored.

Each component requested is plotted versus time in the output file. Plots of this type are useful in providing quick information concerning the time histories of important components. Data for line plots of nodal variables are stored in array DOUT. The user may wish to modify DLEARN to write this information to a disk file for subsequent off-line plotting.

| Note | Columns | Variable | Description |
|------|-------------|----------------------------|--|
| (1) | 1-5 6-10 | IDHIST(1,N) IDHIST(2,N) | Node number; GE. 1 and LE. NUMNP Degree of freedom number; LE. NDOF |
| (2) | 11-15 | IDHIST(3,N) | Kinematic quantity specifier |

Notes

1. History output data must be input for each node/degree of freedom combination at which a time history output is desired. The index N in the array IDHIST corresponds to the order in which the cards are read: 1 .LE. N .LE. NDOUT. The cards need not be specified in any particular order. *The total number of degrees of freedom to be plotted for all nodes must equal NDOUT, defined on the Execution Control Card* (see Sec. In.2.0).
2. IDHIST(3,N) = 1, displacement
IDHIST(3,N) = 2, velocity
IDHIST(3,N) = 3, acceleration

In.5.0 Coordinate Data**In.5.1 Nodal Coordinate Cards (2/5,NSD×F10.0)**

| Note | Columns | Variable | Description |
|------|-------------------------|----------------------------|---|
| (1) | 1-5 | N | Node number; GE. 1 and LE. NUMNP |
| (2) | 6-10 | NUMGP | Number of generation points; EQ.0, no generation GT.0, generate nodal data |
| (3) | 11-20 21-30 31-40 | X(1,N) X(2,N) X(3,N) | x_1 -coordinate of node N x_2 -coordinate of node N x_3 -coordinate of node N |

Notes

1. The coordinates of each node must be defined, but need not be input in any particular

order. If NSD is specified as 2 on the Execution Control Card (See Sec. In.2.0), any nonzero x_3 -coordinates will be ignored. *Terminate with a blank card.*

- If NUMGP is greater than zero, this card initiates an isoparametric data generation sequence. Cards 2 to NUMGP of the sequence define the coordinates of the additional generation points (see Sec. In.5.2). The final card of the sequence defines the nodal increment information (see Sec. In.5.3). After the generation sequence is completed, additional nodal coordinate cards, or generation sequences, may follow.

The generation may be performed along a line, over a surface, or within a volume. A description of each of these options follows.

Generation along a line

The line may be defined by two or three generation points (see Fig. 11.5.1), and the physical space may be one-, two-, or three-dimensional.

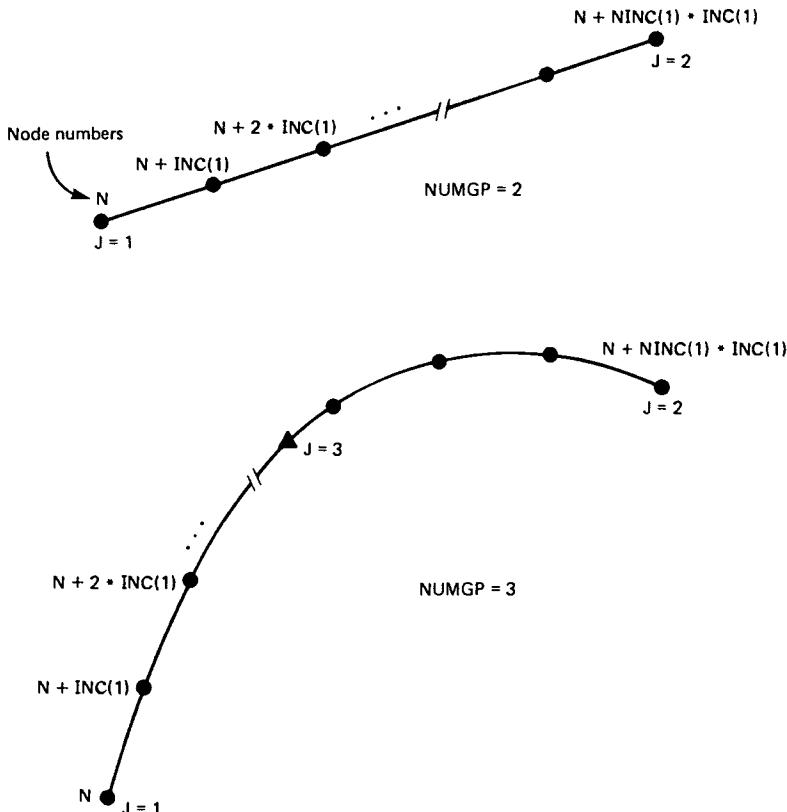


Figure 11.5.1 Nodal generation along a line.

In the case $\text{NUMGP} = 2$, linear interpolation takes place, resulting in equally spaced nodal points.

In the case $\text{NUMGP} = 3$, quadratic interpolation is employed and graded nodal spacing may be achieved by placing the third generation point ($J = 3$) off center. Note that the third generation point does not generally coincide with any nodal point. The spacing in this case may be determined from the following mapping:

$$x_A = x(\xi_A) = \frac{1}{2} \xi_A (\xi_A - 1)x_1^e + \frac{1}{2} \xi_A (\xi_A + 1)x_2^e + (1 - \xi_A^2)x_3^e$$

where ξ_A is the location of node number A in ξ -space. The nodes are placed at equal intervals in ξ -space; x_1^e , x_2^e and x_3^e are the coordinates of the three generation points in x -space; and x_A denotes the coordinates of the A th node in x -space (see Fig. 11.5.2).

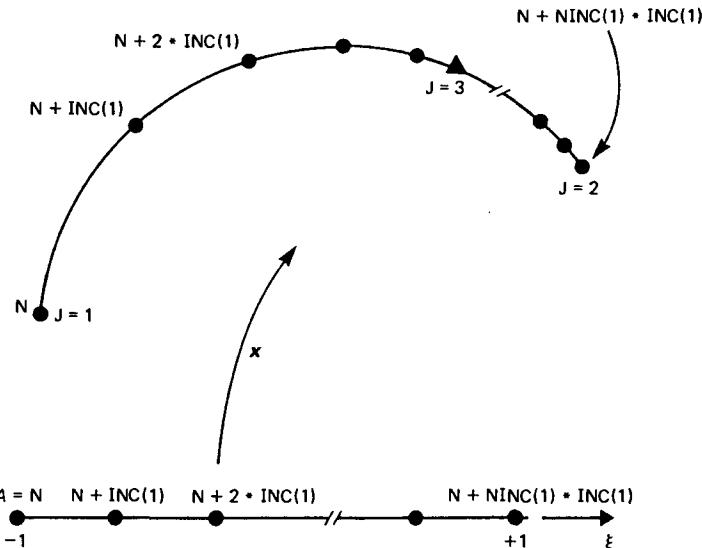


Figure 11.5.2 Nodal generation along a line: mapping from local interval to physical space.

Generation over a surface

The surface may be defined by four or eight generation points (see Fig. 11.5.3) and the physical space may be two- or three-dimensional. In the three-dimensional case, the surfaces may be curved.

In the case NUMGP = 4, bilinear interpolation is employed, resulting in equally spaced nodal points along generating lines.

In the case NUMGP = 8, biquadratic serendipity interpolation is employed, and graded nodal spacing may be achieved by placing generation points 5–8 off center. Note that generation points 5–8 do not generally coincide with any nodal points. The spacing of the nodal points may be determined from the serendipity mapping.

Generation within a volume

The volume is brick shaped and may be defined by 8 or 20 generation points (see Fig. 11.5.4). In this case the physical space must be three-dimensional.

If NUMGP = 8, trilinear interpolation is employed, resulting in equally spaced nodal points along generating lines.

If NUMGP = 20, triquadratic serendipity interpolation is employed and graded nodal spacing may be achieved by placing generation points 9–20 off center. Note that generation points 9–20 do not generally coincide with any nodal points. The spacing of the nodal points may be determined by the serendipity mapping.

3. If the coordinates of node N are input and/or generated more than one time, the last values take precedence.

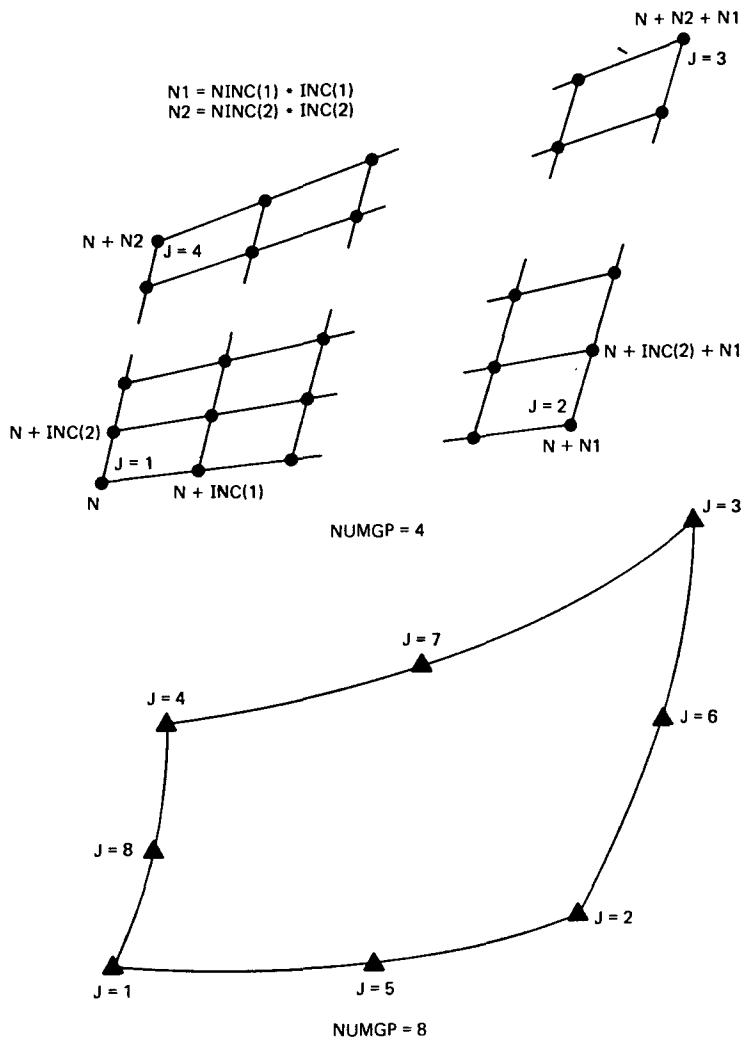


Figure 11.5.3 Nodal generation over a surface.

In.5.2 Generation Point Coordinate Cards (2I5,NSD×F10.0)

| Note | Columns | Variable | Description |
|------|-------------|-----------|--|
| (1) | 1–5 6–10 | M MGEN | Node number Generation parameter; EQ.0, coordinates of the Jth generation point are input on this card; M is ignored. |

| Note | Columns | Variable | Description |
|-------|-----------|---|--|
| 11–20 | TEMP(1,J) | x_1 -coordinate of generation point J | EQ.1, coordinates of the Jth generation point are set equal to coordinates of the Mth node which was previously defined; coordinates on this card are ignored. |
| 21–30 | TEMP(2,J) | x_2 -coordinate of generation point J | |
| 31–40 | TEMP(3,J) | x_3 -coordinate of generation point J | |

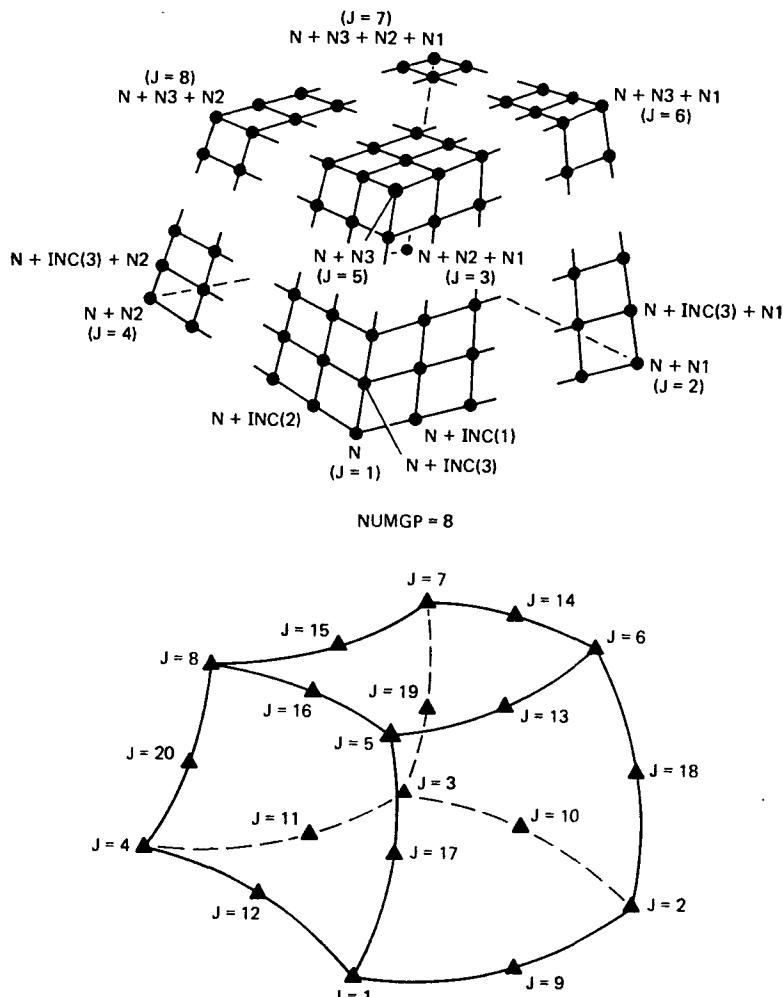


Figure 11.5.4 Nodal generation within a volume.

Notes

1. The coordinates of each generation point are defined by a generation point coordinate card. The cards must be read in order ($J = 2, 3, \dots, \text{NUMGP}$) following the nodal coordinate card which initiated the generation sequence ($J = 1$). A nodal increments card (see Sec. In.5.3), which completes the sequence, follows the last generation point card ($J = \text{NUMGP}$).

In.5.3 Nodal Increments Card (6I5)

| Note | Columns | Variable | Description |
|------|---------|----------|---|
| (1) | 1–5 | NINC(1) | Number of node increments for direction 1; GE.0 |
| | 6–10 | INC(1) | Node number increment for direction 1 |
| (1) | 11–15 | NINC(2) | Number of node increments for direction 2; GE.0 |
| | 16–20 | INC(2) | Node number increment for direction 2 |
| (1) | 21–25 | NINC(3) | Number of node increments for direction 3; GE.0 |
| | 26–30 | INC(3) | Node number increment for direction 3 |

Notes

1. Each generation option is assigned an option code (IOPT) as follows:

| IOPT | Option |
|------|----------------------------|
| 1 | generation along a line |
| 2 | generation over a surface |
| 3 | generation within a volume |

IOPT is determined by the following logic:

$$\begin{aligned} \text{IOPT} &= 3 \\ \text{IF } (\text{NINC}(3).\text{EQ}.0) \text{ IOPT} &= 2 \\ \text{IF } (\text{NINC}(2).\text{EQ}.0) \text{ IOPT} &= 1 \end{aligned}$$

In.6.0 Boundary Conditions Data Cards($(3+NDOF) \times 15$)

| Note | Columns | Variable | Description |
|------|---------|------------|---|
| (1) | 1–5 | N | Number of first node in sequence GE.1 and LE. NUMNP |
| (2) | 6–10 | NE | Number of last node in generation sequence LE. NUMNP |
| (3) | 11–15 | NG | Generation increment |
| | 16–20 | ID(1,N) | Degree of freedom 1 boundary code |
| | 21–25 | ID(2,N) | Degree of freedom 2 boundary code |
| | : | : | : |
| | | ID(NDOF,N) | Degree of freedom NDOF boundary code |

Notes

1. Boundary condition data must be input for each node which has one or more specified displacements, velocities or accelerations. Cards need not be specified in any particular order. *Terminate with a blank card.*
2. Each data card will generate identical boundary condition codes for the following sequence of nodes:

$N, N+NG, N+2\times NG, \dots, NE-\text{MOD}(NE-N, NG)$

If NG is blank or zero, no generation takes place and only boundary conditions for node N are defined. In this case NE is ignored.

3. Boundary condition codes may be assigned the following values:

$ID(I,N) = 0$, unspecified displacement (active degree of freedom)

$ID(I,N) = 1$, specified displacement

$ID(I,N) = 2$, specified velocity

$ID(I,N) = 3$, specified acceleration

Upon exiting subroutine BC, the ID array holds the following values:

$ID(I,N) = M$ (> 0), global equation number for active degree of freedom I of node N

$ID(I,N) = 0$, displacement specified for degree of freedom I of node N

$ID(I,N) = -1$, velocity specified for degree of freedom I of node N

$ID(I,N) = -2$, acceleration specified for degree of freedom I of node N

In.7.0 Prescribed Nodal Forces and Kinematic Boundary Conditions

If NLVECT.EQ.0 on the Execution Control Card (see Sec. In.2.0), no prescribed nodal forces and boundary condition cards should be input; proceed directly to Sec. In.8.0.

Prescribed nodal forces and kinematic boundary conditions, i.e., displacements, velocities or accelerations, are defined by expansions of the form:

$$\mathbf{F}(\mathbf{x}_A, t) = \sum_{i=1}^{\text{NLVECT}} G_i(t) \mathbf{F}_i(\mathbf{x}_A)$$

where $\mathbf{F}(\mathbf{x}_A, t)$ is the resultant force or kinematic boundary condition at node A at time t ; G_i is the i th load-time function; \mathbf{F}_i is the i th load vector; and NLVECT is the total number of load vectors as specified on the Execution Control Card. The data preparation for the load-time functions is described in Sec. In.8.0. In this section, the data specification for each \mathbf{F}_i is described.

The load vectors should be read in the order $\mathbf{F}_1, \mathbf{F}_2, \dots, \mathbf{F}_{\text{NLVECT}}$. Note that each entry of a single vector can be interpreted as a force or kinematic quantity depending on its corresponding boundary condition code defined in Sec. In.6.0.

Data cards for a typical load vector are described next.

**In.7.1 Nodal Prescribed Forces/Kinematic Boundary Conditions
Cards (215,NDOF×F10.0)**

| Note | Columns | Variable | Description |
|------|---------|-----------|---|
| (1) | 1–5 | N | Node number; GE.1 and LE.NUMNP |
| (2) | 6–10 | NUMGP | Number of generation points EQ.0, no generation GT.1, generate data |
| (3) | 11–20 | F(1,N) | Degree of freedom 1 prescribed force/ kinematic boundary condition |
| | 21–30 | F(2,N) | Degree of freedom 2 prescribed force/ kinematic boundary condition |
| : | : | : | : |
| | | F(NDOF,N) | Degree of freedom NDOF prescribed force/ kinematic boundary condition |

Notes

1. Data must be included for each node subjected to a nonzero prescribed force or kinematic boundary condition. Cards need not be specified in any particular order. *Terminate input for each load vector with a blank card.*
2. If NUMGP is greater than zero, this card initiates an isoparametric data generation sequence. The scheme used is the same as that employed for coordinate generation described previously in Sec. In.5. Cards 2 to NUMGP of the sequence define the prescribed forces/kinematic boundary condition data of the additional generation points (see Sec. In.7.2). The final card of the sequence defines the nodal increment information (see Sec. In.5.3). After the generation sequence is completed, additional nodal prescribed forces/kinematic boundary conditions cards, or generation sequences, may follow.

The generation may be performed along a line, over a surface, or within a volume. A description of each of these options is given below.

Generation along a line

The line may be defined by two or three generation points (see Fig. 11.5.5), and the physical space (x -space) may be one-, two-, or three-dimensional.

In the case NUMGP = 2, linear interpolation takes place with respect to ξ -space. If the nodes are equally spaced in x -space, then the variation will also be linear in x -space. Otherwise, a nonlinear variation will be induced by the unequal nodal spacing.

In the case NUMGP = 3, quadratic interpolation is performed with respect to ξ -space. Note that the third generation point does not generally coincide with any nodal point. The variation of the data may be determined from the following mapping:

$$\mathbf{d}_A = \mathbf{d}(\xi_A) = \frac{1}{2}\xi_A(\xi_A - 1)\mathbf{d}_1^\xi + \frac{1}{2}\xi_A(\xi_A + 1)\mathbf{d}_2^\xi + (1 - \xi_A^2)\mathbf{d}_3^\xi$$

where ξ_A is the location of node number A in ξ -space (recall that nodes are assumed to be placed at equal intervals in ξ -space); \mathbf{d}_1^ξ , \mathbf{d}_2^ξ , and \mathbf{d}_3^ξ are the data assigned to the three generation points in x -space; and \mathbf{d}_A denotes the prescribed force/kinematic boundary condition data of the A th node in x -space.

The case in which NUMGP = 2 may be deduced from the case NUMGP = 3 by setting $\mathbf{d}_3^\xi = (\mathbf{d}_1^\xi + \mathbf{d}_2^\xi)/2$.

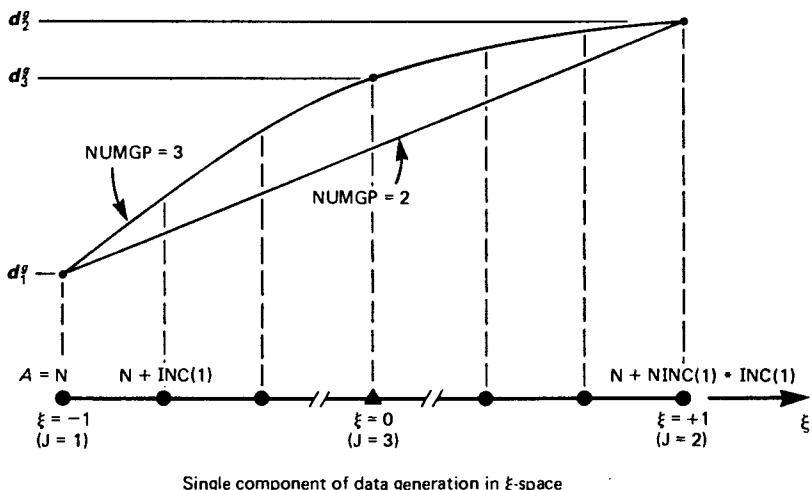
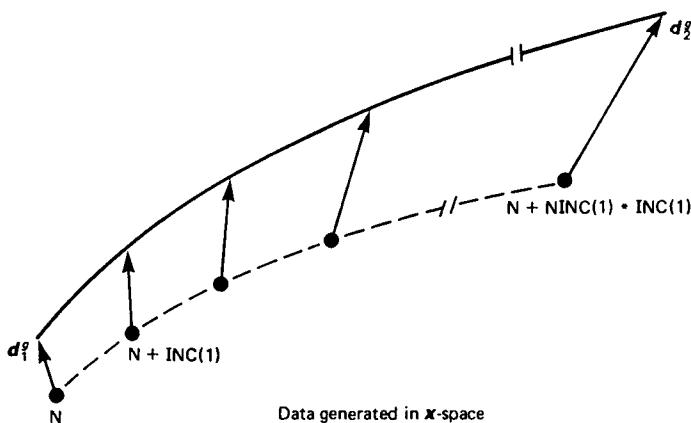


Figure 11.5.5 Load vector generation.

Generation over a surface

The surface may be defined by 4 or 8 generation points. The generation points and nodal patterns are the same as in coordinate generation (see Fig. 11.5.3).

In the case $NUMGP = 4$, bilinear interpolation is performed; for $NUMGP = 8$, biquadratic serendipity interpolation is performed. Note that generation points 5–8 do not generally coincide with any nodal points.

Generation within a volume

Generation of data within a brick-shaped volume may be performed using 8 or 20 generation points. The generation points and nodal patterns are the same as for coordinate generation (see Fig. 11.5.4).

If $NUMGP = 8$, trilinear interpolation is performed; if $NUMGP = 20$, triquadratic

serendipity interpolation is performed. Note that generation points 9–20 do not generally coincide with any nodal points.

3. The elements of the array F(NDOF,NUMNP,NLVECT) are initialized to zero. Prescribed forces/kinematic boundary conditions for node N are written within this array. If values for node N are defined more than once, the last value will take precedence.

In.7.2 Generation Point Prescribed Forces/Kinematic Boundary Conditions Cards (2I5,NDOFxF10.0)

| Note | Columns | Variable | Description |
|-------|--------------|-----------|---|
| 1 | 1–5 6–10 | M MGEN | Node number Generation parameter; EQ.0, prescribed forces/kinematic boundary conditions of the Jth generation point are input on this card; M is ignored. EQ.1, prescribed forces/kinematic boundary conditions of the Jth generation point are set equal to the prescribed forces/kinematic boundary conditions of the Mth node which were previously defined; prescribed forces/ kinematic boundary conditions on this card are ignored. |
| 11–20 | TEMP(1,J) | | Degree of freedom 1 prescribed force/kinematic boundary condition for generation point J |
| 21–30 | TEMP(2,J) | | Degree of freedom 2 prescribed force/kinematic boundary condition for generation point J |
| : | : | | : |
| | TEMP(NDOF,J) | | Degree of freedom NDOF prescribed force/kinematic boundary condition for generation point J |

Notes

1. The prescribed forces/kinematic boundary conditions of each generation point are defined by a generation point prescribed forces/kinematic boundary conditions card. The cards must be read in order ($J = 2, 3, \dots, \text{NUMGP}$) following the nodal prescribed forces/kinematic boundary conditions card which initiated the generation sequence ($J = 1$). A nodal increments card (see Sec. In.5.3) follows the last generation point card ($J = \text{NUMGP}$) and completes the sequence.

In.7.3 Nodal Increments Card (6I5)

See instructions in Sec. In.5.3.

In.8.0 LOAD-TIME FUNCTIONS

If NLTFN .EQ. 0 on the Execution Control Card (see Sec. In.2.0), no load-time function cards should be input; proceed directly to Sec. In.9.0.

If NLTFN .GT. 0, the user must input NLTFN load-time functions, each defined by NPTSLF pairs of time instants and function values, where NLTFN and NPTSLF are defined on the Execution Control Card. A schematic representation of a typical load-time function is shown in Fig. 11.5.6. The time instants must be in

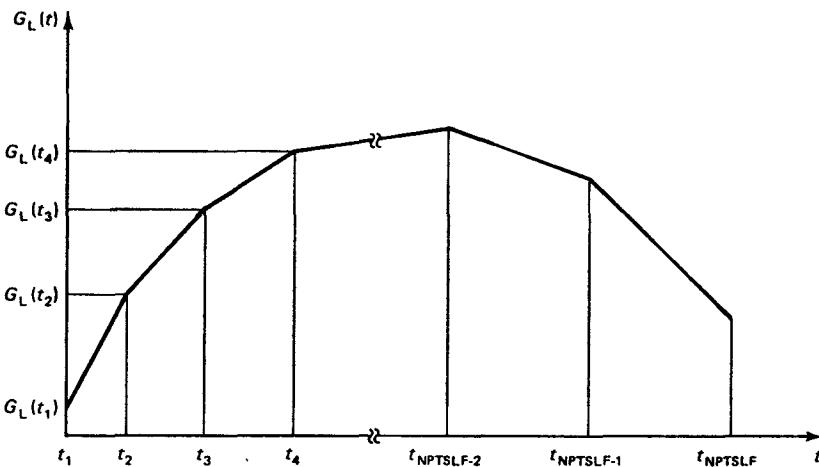


Figure 11.5.6 Schematic representation of load-time function L.

ascending order (i.e., $t_k < t_{k+1}$, $1 \leq k \leq \text{NPTSLF} - 1$). Load step intervals need not be equal and need not be the same from one load-time function to another. However, there must be the same number of load steps (i.e., NPTSLF) for each load-time function. As may be seen from Fig. 11.5.6, the load-time function is assumed to behave in a piecewise linear fashion between data points. For values of t outside the interval $[t_1, t_{\text{NPTSLF}}]$ we define G_i by constant extrapolation: $G_i(t) = G_i(t_1)$ for all $t \leq t_1$ and $G_i(t) = G_i(t_{\text{NPTSLF}})$ for all $t \geq t_{\text{NPTSLF}}$. For example, if the user wishes to define all load-time functions with a constant magnitude for all time, he may set NPTSLF = 1 and define a single point of $G_i(t)$ for any time t .

The load-time functions must be read in the order $G_1, G_2, \dots, G_{\text{NLTFN}}$, where the first NLVECT functions scale the corresponding load vectors specified in Sec. In.7.0. Element loads (e.g., pressure, gravity, etc.) are also multiplied by load-time functions. The applicable function number is defined in the element group data (see Sec. In.10).

Data cards for a typical load-time function are described next.

In.8.1 Load-time Function Cards (2F10.0)

| Note | Columns | Variable | Description |
|------|---------|----------|---|
| (1) | 1–10 | G(K,1) | Time value K, i.e., t_k |
| | 11–20 | G(K,2) | Value of load-time function at time t_k |

Notes

1. The local array G(K,I) is equivalent to global array G(K,I,L) for load-time function L.

In.9.0 KINEMATIC INITIAL CONDITION DATA

If the analysis is static (i.e., if IACODE .EQ. 1 on the Execution Control Card, see Sec. In.2.0) or if the program is instructed to read a restart file (i.e., if IREADR .EQ. 1 on the Execution Control Card), then no nodal initial condition data cards should be input; proceed directly to Sec. In.10.0.

Initial nodal displacements, velocities and accelerations are specified in separate data blocks having identical form. Each block is terminated with a blank card, which must be present even if zero initial conditions are desired at all nodes for one of the kinematic variables. For example, if the user wishes only to specify nonzero initial velocities, the data will have the form:

1. A blank card representing the absence of nonzero initial displacement data.
2. A block of initial velocity data input in the manner specified by Secs. In.9.1 to In.9.3 and terminating with a blank card.
3. A blank card representing the absence of nonzero initial acceleration data.

In.9.1 Nodal Initial Condition Cards (2I5,NSD×F10.0)

| Note | Columns | Variable | Description |
|------|---------|-----------|---|
| (1) | 1–5 | N | Node number; GE.1 and LE. NUMNP |
| (2) | 6–10 | NUMGP | Number of generation points EQ.0, no generation GT.0, generate initial conditions |
| (3) | 11–20 | D(1,N)* | Degree of freedom 1 initial condition |
| | 21–30 | D(2,N) | Degree of freedom 2 initial condition |
| | : | : | : |
| | : | : | : |
| | : | : | : |
| | | D(NDOF,N) | Degree of freedom NDOF initial condition |

* Here the array D is used generically to refer to either displacements (D), velocities (V), or accelerations (A).

Notes

1. Data must be included for each node subjected to nonzero initial displacements, velocities, or accelerations but need not be specified in any particular nodal order. *Terminate each data block with a blank card.*
2. If NUMGP is greater than zero, this card initiates an isoparametric data generation sequence. The scheme used is the same as that for prescribed nodal force/boundary condition data generation described in Sec. In.7.0. Cards 2 to NUMGP of the sequence define the initial condition data of the additional generation points and are described in Sec. In.9.2. The final card of the sequence defines the nodal increment information and is identical to the one used for coordinate generation (see Sec. In.5.3). After the generation sequence is completed, additional nodal initial condition cards, or generation sequences, may follow.
The generation may be performed along a line, over a surface, or within a volume. For additional information concerning these options see Note (2) of Sec. In.7.1.
3. The elements of the arrays D(NDOF,NUMNP), V(NDOF,NUMNP) and A(NDOF,NUMNP) are initialized to zero. If the initial condition data of node N is input and/or generated more than one time, the last value takes precedence.

**In.9.2 Generation Point Initial Condition Cards
(2I5,NDOFxF10.0)**

| Note | Columns | Variable | Description |
|------|---------|--------------|--|
| (1) | 1–5 | M | Node number |
| | 6–10 | MGEN | Generation parameter EQ.0, initial condition data of the Jth generation point are input on this card; M is ignored. EQ.1, initial condition data of the Jth generation point are set equal to the initial condition data of the Mth node which were previously defined; initial condition data on this card are ignored. |
| | 11–20 | TEMP(1,J) | Degree of freedom 1 initial condition for generation point J |
| | 21–30 | TEMP(2,J) | Degree of freedom 2 initial condition for generation point J |
| | . | . | . |
| | . | . | . |
| | . | . | . |
| | | TEMP(NDOF,J) | Degree of freedom NDOF initial condition for generation point J |

Notes

1. The initial condition data of each generation point are defined by a generation point initial condition card. The cards must be read in order ($J = 2, 3, \dots, \text{NUMGP}$) following the nodal initial condition card that initiated the generation sequence ($J = 1$). A nodal increments card (see Sec. In.5.3), which completes the sequence, follows the last generation point card ($J = \text{NUMGP}$).

In.9.3 Nodal Increments Card (615)

See the discussion in Sec. In.5.3.

In.10.0 ELEMENT DATA

In.10.1 Two-Dimensional, Isotropic Elasticity Element

The present element may be used in triangular (three-node) or quadrilateral (four-node) form for plane stress, plane strain, or torsionless axisymmetric analysis. Two displacement degrees of freedom, in the x_1 - and x_2 -directions, are assigned to each node. Therefore, the user must specify NSD .GE. 2 and NDOF .GE. 2 on the Execution Control Card (see Sec. In.2.0).

Stress and strain output is computed at each of the integration points. Printed results include stresses and strains in the global coordinate system, principal stresses and strains, maximum shear stress or strain in the x_1 , x_2 -plane and angle of inclination, in degrees, of principal states (see Fig. 11.5.7). Any one of these quantities may also be output as a time history. All shear strains are reported according to the "engineering" convention (i.e., twice the value of the tensor components). See Sec. In.10.1.6 for precise definitions of the output quantities.

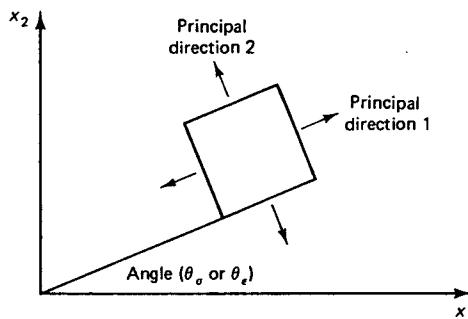


Figure 11.5.7 Orientation of principal axes.

The following sequence of cards is used to describe the elements in a group.

In.10.1.1 Element Group Control Card (1315)

| Note | Columns | Variable | Description |
|------|---------|----------|---|
| (1) | 1–5 | NTYPE | The number 1 |
| | 6–10 | NUMEL | Number of elements in this group; GE. 1 |
| | 11–15 | NUMAT | Number of material sets in this group; GE. 1 |
| | 16–20 | NSURF | Number of surface force cards |
| (2) | 21–25 | NSOUT | Number of stress/strain time-history cards |
| | 26–30 | IOPT | Analysis option EQ.0, plane stress EQ.1, plane strain EQ.2, torsionless axisymmetric |

| Note | Columns | Variable | Description |
|------|---------|----------|---|
| (3) | 31–35 | ISTPRT | Stress output print code EQ.0, print stress output EQ.1, do not print stress output |
| (4) | 36–40 | LFSURF | Surface force load function number |
| (4) | 41–45 | LFBODY | Body force load function number |
| (5) | 46–50 | NICODE | Numerical integration code EQ.0, 2×2 Gaussian quadrature EQ.1, one-point Gaussian quadrature |
| (6) | 51–55 | IBBAR | Strain-displacement option EQ.0, standard formulation EQ.1, mean-dilatational \bar{B} formulation |
| (7) | 56–60 | IMASS | Mass type code EQ.0, consistent mass matrix EQ.1, lumped mass matrix EQ.2, no mass matrix |
| | 61–65 | IMPEXP | Implicit/explicit code EQ.0, implicit element group EQ.1, explicit element group |

Notes

1. All data on this card are stored in an array (NPAR(I),I=1,16):

NPAR(1)=NTYPE
NPAR(2)=NUMEL

NPAR(13)=IMPEXP

2. Output histories are ignored in static analysis as denoted by IACODE .EQ.1 on the Execution Control Card, Sec. In.2.0. See also Sec. In.10.1.6.
3. This option is useful, especially during dynamic analysis, for suppressing output from element groups in physically uninteresting regions of the mesh.
4. The user may specify any of the load-time functions defined in Sec. In.8.0.
5. The standard four-node quadrilateral employs 2×2 Gaussian quadrature. One-point Gaussian quadrature produces a dangerous element in that zero energy modes of deformation, so-called hourglass or keystone modes, may be present, resulting in a singular stiffness matrix. If enough displacement boundary conditions are present these modes may be eliminated. *However, one-point Gaussian quadrature should only be used if you know exactly what you are doing.* Internally, NINT denotes the number of integration points; NICODE .EQ. 0 \Rightarrow NINT .EQ. 4, and NICODE .EQ. 1 \Rightarrow NINT .EQ. 1.
6. The use of the mean-dilatational \bar{B} strain-displacement matrix is effective in the analysis of nearly incompressible materials, where standard elements are susceptible to mesh “locking.” Use *only* if IOPT .EQ. 1 or 2 (i.e., plane strain or axisymmetry).
7. IMASS .EQ. 2 will cause inertial and body forces to be neglected for this element group in both static and dynamic analysis.

In. 10.1.2 Material Properties Cards (15,5X,6F10.0)

| Note | Columns | Variable | Description |
|------|---------|-----------|--|
| (1) | 1–5 | M | Material identification number; GE, 1 and LE. NUMAT |
| | 11–20 | E | Young's modulus |
| (2) | 21–30 | POIS | Poisson's ratio |
| | 31–40 | RHO(M) | Mass density |
| (3) | 41–50 | RDAMPM(M) | Mass matrix Rayleigh damping coefficient |
| (3) | 51–60 | RDAMPK(M) | Stiffness matrix Rayleigh damping coefficient |
| (4) | 61–70 | TH(M) | Thickness |

Notes

1. Material property sets may be specified in any order. Note that E and POIS are not stored directly; rather they are used to compute the entries of the material stiffness matrix C .
2. Poisson's ratio cannot be set equal to 0.5 since it results in division by zero. A value close to 0.5—for instance, 0.4999—can be employed for incompressible applications.
3. The element damping matrix is computed as:

$$c^e = RDAMPM(MAT(NEL)) * m^e + RDAMPK(MAT(NEL)) * k^e.$$

4. For IOPT .EQ. 1 or 2, the thickness will default to 1.0.

In. 10.1.3 Gravity Vector Card (2F10.0)

| Note | Columns | Variable | Description |
|------|---------|----------|---|
| (1) | 1–10 | GRAV(1) | Component of gravity vector in x_1 -direction |
| | 11–20 | GRAV(2) | Component of gravity vector in x_2 -direction |

Notes

1. These components will be scaled by the load-time function LFBODY when computing the element body force.

In. 10.1.4 Element Definition**In. 10.1.4.1 Nodal data cards (7I5)**

| Note | Columns | Variable | Description |
|------|---------|----------|-------------------------|
| (1) | 1–5 | N | Element number |
| | 6–10 | M | Element material number |
| (2) | 11–15 | IEN(1,N) | Number of first node |

| Note | Columns | Variable | Description |
|------|---------|----------|--|
| | 16–20 | IEN(2,N) | Number of second node |
| | 21–25 | IEN(3,N) | Number of third node |
| (3) | 26–30 | IEN(4,N) | Number of fourth node |
| (4) | 31–35 | NG | Generation parameter EQ.0, no generation EQ.1, generate data |

Notes

1. All elements must be input on a nodal data card or generated. Each group begins with an element number 1. *Terminate with a blank card.*
2. Element nodes must be listed in counterclockwise order. See Fig. 11.5.8.
3. For triangular elements, set node number IEN (4,N) equal to IEN(3,N).
4. If the generation parameter is set to 1, an Element Generation Data Card must be input next.

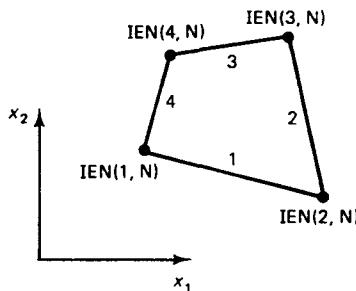


Figure 11.5.8 Nodal ordering for element number N.

In. 10.1.4.2 Element generation data cards (6!5)

| Note | Columns | Variable | Description |
|------|---------|----------|--|
| (1) | 1–5 | NEL(1) | Number of elements in direction 1; GE. 0 If EQ.0, set internally to 1. |
| | 6–10 | INCEL(1) | Element number increment for direction 1 If EQ.0, set internally to 1. |
| | 11–15 | INC(1) | Node number increment for direction 1 If EQ.0, set internally to 1. |
| | 16–20 | NEL(2) | Number of elements in direction 2; GE. 0 If EQ.0, set internally to 1. |
| | 21–25 | INCEL(2) | Element number increment for direction 2 If EQ.0, set internally to NEL(1). |
| | 26–30 | INC(2) | Node number increment for direction 2 If EQ.0, set internally to (NEL(1) + 1)*INC(1). |

Notes

1. See Fig. 11.5.9 for a schematic representation of the generation algorithm.

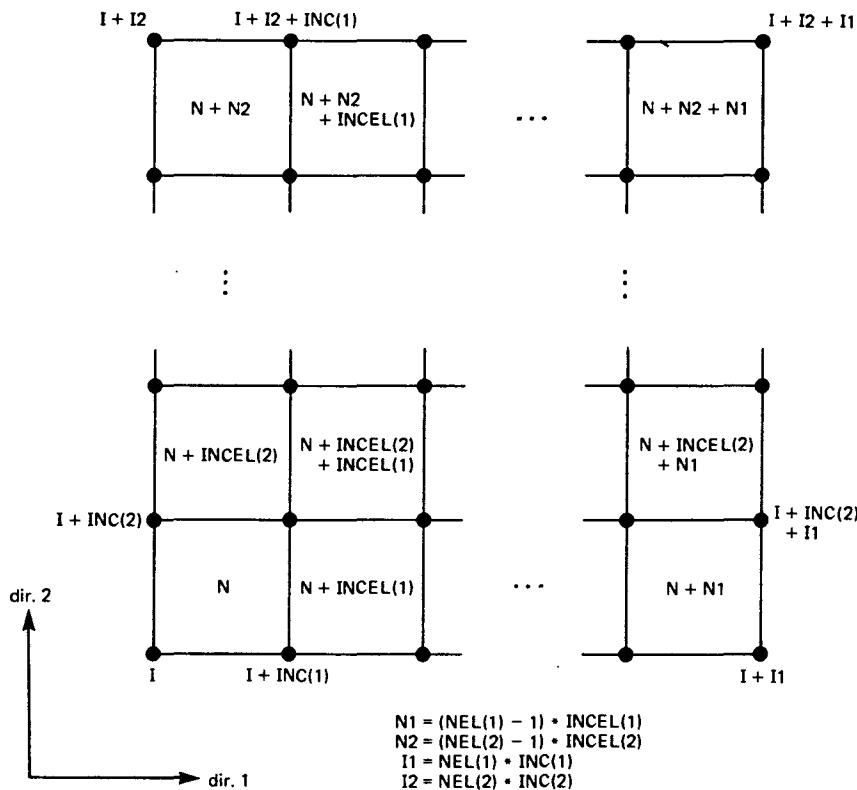


Figure 11.5.9 Element generation over a logically rectangular region.

In.10.1.5 Element Surface Load Cards (2I5,4F10.0)

| Note | Columns | Variable | Description |
|------|---------|------------|--------------------------------------|
| (1) | 1-5 | IELNO(K) | Element number; GE. 1 and LE. NUMEL |
| (2) | 6-10 | ISIDE(K) | Element side number; GE. 1 and LE. 4 |
| (3) | 11-20 | PRESS(1,K) | Pressure at node I |
| | 21-30 | PRESS(2,K) | Pressure at node J |
| (4) | 31-40 | SHEAR(1,K) | Shear stress at node I |
| | 41-50 | SHEAR(2,K) | Shear stress at node J |

Notes

1. Each element side subjected to a surface load must be specified. If more than one side of the element is loaded, one card must be input for each loaded side. The index K in the arrays IELNO and ISIDE corresponds to the order in which the cards are read: 1.LE. K.LE. NSURF. The cards need not be specified in any particular order. *The total number*

of Element Surface Load Cards must equal NSURF, as specified on the Element Group Control Card (see Sec. In.10.1.1).

2. The element side number is deduced as follows: Consider the element of Fig. 11.5.8. Let $N = \text{IELNO}(K)$. Side 1 connects nodes $\text{IEN}(1,N)$ and $\text{IEN}(2,N)$, side 2 connects nodes $\text{IEN}(2,N)$ and $\text{IEN}(3,N)$, etc. For the purpose of element surface force specification, locally we refer to the node beginning the loaded side as I and the node ending the side as J.
3. The pressure is assumed to be positive pointing inward, and is linearly interpolated between nodal values; see Fig. 11.5.10. In computing the element surface forces, pressure is scaled by load-time function LFSURF.

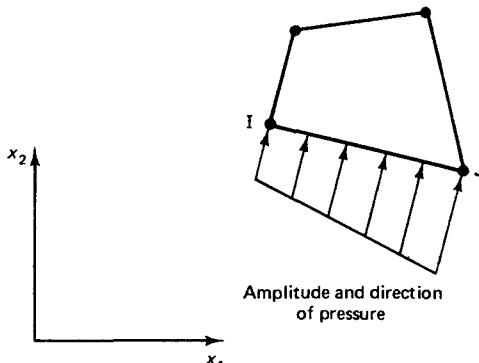


Figure 11.5.10

4. The shear stress is assumed to be positive pointing in a counterclockwise fashion and is linearly interpolated between nodal values; see Fig. 11.5.11. In computing the element surface forces, the shear stress is scaled by load-time function LFSURF.

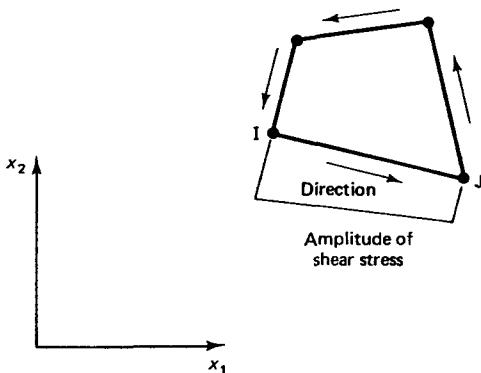


Figure 11.5.11

In.10.1.6 Element Time History Cards (315)

If the analysis is static (i.e., IACODE .EQ. 1 on the Execution Control Card; see Sec. In.2.0), then output history cards are read but subsequently ignored.

| Note | Columns | Variable | Description |
|------|---------|-------------|--|
| (1) | 1–5 | ISHIST(1,N) | Element number; GE. 1 and LE. NUMEL |
| (2) | 6–10 | ISHIST(2,N) | Quadrature point number; GE. 1 and LE. NINT |
| (3) | 11–15 | ISHIST(3,N) | Stress/strain component number, GE. 1 and LE. 16 |

Notes

1. Each desired element time history must be specified; thus an element for which the user requests multiple time histories will engender multiple data cards. The index N in the array ISHIST corresponds to the order in which the cards are read: 1 .LE. N .LE. NSOUT. The cards need not be specified in any particular order. *The total number of Element Time History Cards must equal NSOUT as specified on the Element Group Control Card* (see Sec. In.10.1.1).
2. NINT denotes the number of integration points; NICODE .EQ. 0 \Rightarrow NINT .EQ. 4, and NICODE .EQ. 1 \Rightarrow NINT .EQ. 1. In local coordinates (i.e., ξ , η), the quadrature point number defines the following locations:

| NINT | l | $\tilde{\xi}_l$ | $\tilde{\eta}_l$ |
|------|-----|-----------------------|-----------------------|
| 1 | 1 | 0 | 0 |
| 4 | 1 | $\frac{-1}{\sqrt{3}}$ | $\frac{-1}{\sqrt{3}}$ |
| | 2 | $\frac{1}{\sqrt{3}}$ | $\frac{-1}{\sqrt{3}}$ |
| | 3 | $\frac{1}{\sqrt{3}}$ | $\frac{1}{\sqrt{3}}$ |
| | 4 | $\frac{-1}{\sqrt{3}}$ | $\frac{1}{\sqrt{3}}$ |

3. The following element components may be specified:

| NCOMP | Description | Output Label |
|-------|--|--------------|
| 1 | σ_{11} , stress 11 | S 11 |
| 2 | σ_{22} , stress 22 | S 22 |
| 3 | σ_{12} , stress 12 | S 12 |
| 4 | σ_{33} , stress 33 | S 33 |
| 5 | σ_1 , principal stress 1 | PS 1 |
| 6 | σ_2 , principal stress 2 | PS 2 |
| 7 | $\tau = \frac{1}{2} \sigma_1 - \sigma_2 $, shear stress | TAU |
| 8 | θ_σ , stress angle (i.e., angle between x_1 and σ_1 axes) | SANG |
| 9 | ϵ_{11} , strain 11 | E 11 |
| 10 | ϵ_{22} , strain 22 | E 22 |
| 11 | $\gamma_{12} = 2\epsilon_{12}$ (i.e., engineering shear strain) | G 12 |
| 12 | ϵ_{33} , strain 33 | E 33 |

| | | |
|----|--|------|
| 13 | ϵ_1 , principal strain 1 | PE 1 |
| 14 | ϵ_2 , principal strain 2 | PE 2 |
| 15 | $\gamma = \epsilon_1 - \epsilon_2 $, (i.e., engineering shear strain) | GAM |
| 16 | θ_ϵ , strain angle (i.e., angle between x_1 and ϵ_1 axes) | EANG |

In.10.2 Three-Dimensional, Elastic Truss Element

The present element connects two or three nodes in space and transmits axial force only. There are three degrees of freedom at each node, i.e., the x_1 , x_2 , and x_3 translations. When employing truss elements, the user must specify NSD .EQ. 3 and NDOF .GE. 3 on the Execution Control Card (see Sec. In.2.0). The following sequence of cards is used to describe truss elements:

In.10.2.1 Element Group Control Card (10!5)

| Note | Columns | Variable | Description |
|------|---------|----------|---|
| (1) | 1–5 | NTYPE | The number 2 |
| | 6–10 | NUMEL | Number of elements in this group; GE. 1 |
| | 11–15 | NUMAT | Number of material sets in this group; GE. 1 |
| | 16–20 | NEN | Maximum number of nodes for all elements in this group EQ. 2 or 3 |
| (2) | 21–25 | NSOUT | Number of stress/strain time-history cards |
| (3) | 26–30 | ISTPRT | Stress output print code EQ.0, print stress output EQ.1, do not print stress output |
| (4) | 31–35 | LFBODY | Body force load function number |
| (5) | 36–40 | NINT | Number of integration points EQ.1, one-point Gaussian quadrature EQ.2, two-point Gaussian quadrature EQ.3, three-point Gaussian quadrature |
| (6) | 41–45 | IMASS | Mass type code EQ.0, consistent mass matrix EQ.1, lumped mass matrix EQ.2, no mass matrix |
| | 46–50 | IMPEXP | Implicit/explicit code EQ.0, implicit element group EQ.1, explicit element group |

Notes

- All data on this card are stored in an array (NPAR(I), I = 1,16):

$$\text{NPAR}(1) = \text{NTYPE}$$

$$\text{NPAR}(2) = \text{NUMEL}$$

$$\vdots$$

$$\text{NPAR}(10) = \text{IMPEXP}$$

2. Output histories are ignored in static analysis as denoted by IACODE .EQ. 1 on the Execution Control Card, Sec. In.2.0. See also Sec. In.10.2.5.
3. This option is useful, especially during dynamic analysis, for suppressing output from element groups in physically uninteresting regions of the mesh.
4. The user may specify any of the load-time functions defined in Sec. In.8.0.
5. The following facts may be used to guide the selection of an appropriate integration rule:
 - For two-node truss elements, the stiffness is exactly integrated by the one-point Gauss rule and the mass is exactly integrated by the two-point Gauss rule.
 - For three-node truss elements, if the location of the “middle node,” x_3 , satisfies $x_3 = (x_1 + x_2)/2$, then the stiffness is exactly integrated by the two-point Gauss rule and the mass is exactly integrated by the three-point Gauss rule.
6. IMASS .EQ. 2 will cause inertial and body forces to be neglected for this element group in both static and dynamic analysis.

In.10.2.2 Material Properties Cards (15,5X,5F10.0)

| Note | Columns | Variable | Description |
|------|---------|-----------|---|
| (1) | 1–5 | M | Material identification number GE. 1 and LE. NUMAT |
| | 11–20 | E | Young’s modulus |
| | 21–30 | RHO(M) | Mass density |
| (2) | 31–40 | RDAMPM(M) | Mass matrix Rayleigh damping coefficient |
| (2) | 41–50 | RDAMPK(M) | Stiffness matrix Rayleigh damping coefficient |
| | 51–60 | AREA(M) | Area |

Notes

1. Material property sets may be specified in any order. Note that E is stored as the single entry in the material stiffness matrix C .
2. The element damping matrix is computed as

$$c^e = RDAMPM(MAT(NEL)) * m^e + RDAMPK(MAT(NEL)) * k^e.$$

In.10.2.3 Gravity Vector Card (3F10.0)

| Note | Columns | Variable | Description |
|------|---------|----------|---|
| (1) | 1–10 | GRAV(1) | Component of gravity vector in x_1 -direction |
| | 11–20 | GRAV(2) | Component of gravity vector in x_2 -direction |
| | 21–30 | GRAV(3) | Component of gravity vector in x_3 -direction |

Notes

1. These components will be scaled by the load-time function LFBODY when computing the element body force.

In.10.2.4 Element Definition**In.10.2.4.1. Nodal Data Cards (615)**

If NEN .EQ. 2, all nodal data cards in this group have the following form.

| Note | Columns | Variable | Description |
|------|---------|----------|--|
| (1) | 1–5 | N | Element number |
| | 6–10 | M | Element material number |
| (2) | 11–15 | IEN(1,N) | Number of first node |
| | 16–20 | IEN(2,N) | Number of second node |
| (3) | 21–25 | NG | Generation parameter |
| | | | EQ.0, no generation EQ.1, generate data |

If NEN .EQ. 3, all nodal data cards in this group have the following form.

| Note | Columns | Variable | Description |
|------|---------|----------|--|
| (1) | 1–5 | N | Element number |
| | 6–10 | M | Element material number |
| (2) | 11–15 | IEN(1,N) | Number of first node |
| | 16–20 | IEN(2,N) | Number of second node |
| (4) | 21–25 | IEN(3,N) | Number of third node |
| | 26–30 | NG | Generation parameter |
| (3) | | | EQ.0, no generation EQ.1, generate data |

Notes

1. All elements must be input on a nodal data card or generated. Each group begins with an element number 1. *Terminate with a blank card.*
2. See Fig. 11.5.12 for definition of nodal ordering.
3. If the generation parameter is set to 1, a generation data card must be input next.
4. Two-node elements may be defined within a group of three-node elements (i.e., NEN = 3) by setting IEN(3,N) equal to IEN(2,N).

NEN = 2

NEN = 3

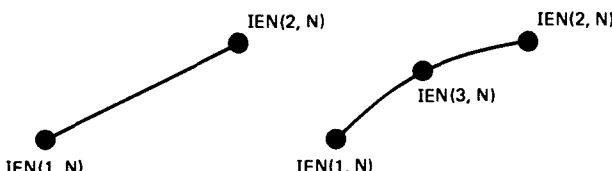


Figure 11.5.12 Nodal ordering for the truss element.

In. 10.2.4.2 Element Generation Data Cards (315)

| Note | Columns | Variable | Description |
|------|---------|----------|---|
| (1) | 1–5 | NEL(1) | Number of elements to be generated |
| | 6–10 | INCEL(1) | Element number increment If EQ.0, set internally to 1. |
| | 11–15 | INC(1) | Node number increment If EQ.0, set internally to 1. |

Notes

- See Fig. 11.5.13 for a schematic representation of the generation algorithm.

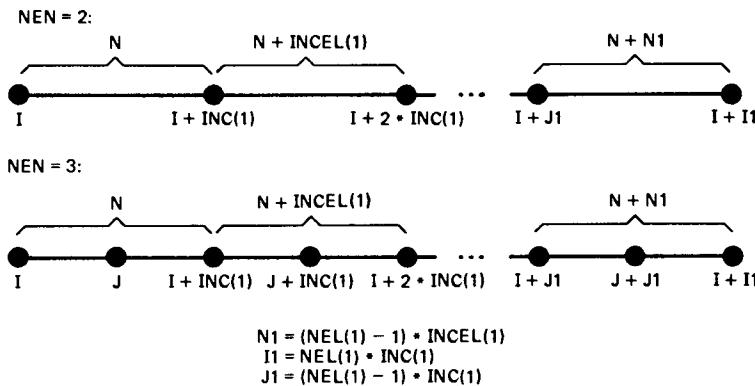


Figure 11.5.13 Generation of truss elements along a line.

In. 10.2.5 Element Time History Cards (315)

If the analysis is static (i.e., IACODE .EQ. 1 on the Execution Control Card; see Sec. In.2.0), then output history cards are read, but subsequently ignored.

| Note | Columns | Variable | Description |
|------|---------|-------------|---|
| (1) | 1–5 | ISHIST(1,N) | Element number; GE. 1 and LE. NUMEL |
| (2) | 6–10 | ISHIST(2,N) | Quadrature point number; GE. 1 and LE. NINT |
| (3) | 11–15 | ISHIST(3,N) | Stress/strain component number, GE. 1 and LE. 3 |

Notes

- Each desired element time history must be specified; thus an element for which the user requests multiple time histories will engender multiple data cards. The index N in the array ISHIST corresponds to the order in which the cards are read: 1 .LE. N .LE. NSOUT. The cards need not be specified in any particular order. *The total number of Element Time History Cards must equal NSOUT, as specified on the Element Group Control Card* (see Sec. In.10.2.1).
- In the local coordinate (i.e., ξ), the quadrature point number corresponds to the following locations:

| NINT | <i>l</i> | $\tilde{\xi}_l$ |
|------|----------|-------------------------|
| 1 | 1 | 0 |
| 2 | 1 | $\frac{-1}{\sqrt{3}}$ |
| | 2 | $\frac{1}{\sqrt{3}}$ |
| 3 | 1 | $\frac{-1}{\sqrt{3/5}}$ |
| | 2 | $\frac{1}{\sqrt{3/5}}$ |
| | 3 | 0 |

3. The following element components may be specified:

| NCOMP | Description | Output Label |
|-------|--------------|--------------|
| 1 | Axial stress | STRS |
| 2 | Axial force | FORC |
| 3 | Axial strain | STRN |

11.5.3 Examples

Example 1 (Planar Truss)

Here we consider a planar truss, shown schematically with node numbers in Fig. 11.5.14, subjected to multiple static loadings. Looking at the input file given in

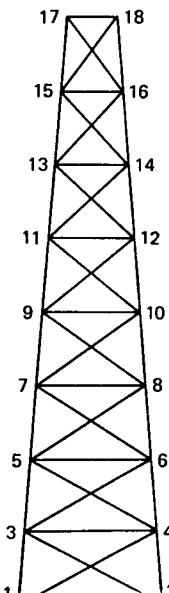


Table 11.5.1, note the use of generation capabilities, including the definition of the truss elements. In the time-sequence data we request three time steps and have set $\Delta t = 1.0$, although the code would default to this value. Two load vectors and load-time functions have been specified. The first load vector corresponds to a linearly varying lateral load, as might be used for preliminary seismic design. The second defines a uniform vertical load acting at every node. The load-time functions, shown schematically in Fig. 11.5.15, produce three different load cases combining varying proportions of the two load vectors. The output on pages 693 through 704 can be used to verify DLEARNS upon installation on a new computer system.

TABLE 11.5.1 Input File for Planar Truss Problem

| | | | | | | | | | | | | | |
|-------------|----|-------|------------|------------|---------|----------|---------|---------|-------|------|------|------|------|
| In 0.0 | 0 | TRUSS | TOWER | WITH | STATIC | VERTICAL | AND | LATERAL | LOADS | | | | |
| In 1.0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0.00 | 2 | 1.00 |
| In 2.0 | 1 | 3 | 1 | 1 | 1 | 0 | 1 | 1 | 0.00 | 18 | 1.00 | 3 | 0.00 |
| In 3.0 | 1 | 2 | 2 | 0 | 0 | 48 | 0 | 0 | 768 | | | | |
| In 5.1 | 1 | 2 | 2 | 2 | 2 | 144 | 0 | 0 | 768 | | | | |
| In 5.2 | 17 | 0 | 0 | 0 | 0 | 96 | 0 | 0 | 768 | | | | |
| In 5.3 | 8 | 2 | 2 | 2 | 2 | 96 | 0 | 0 | 768 | | | | |
| In 5.3 | 18 | 2 | 2 | 2 | 2 | 96 | 0 | 0 | 768 | | | | |
| In 6.0 | { | 1 | 18 | 1 | 0 | 0 | 0 | 1 | | | | | |
| In 6.0 | | 2 | 0 | 0 | 0 | 1 | 1 | 1 | | | | | |
| In 7.1 | { | 3 | 2 | 500.0 | 0.00 | 0.00 | 0.00 | 0.00 | | | | | |
| In 7.2 | | 17 | 0 | 4000.0 | 0.00 | 0.00 | 0.00 | 0.00 | | | | | |
| In 7.3 | | 7 | 2 | | | | | | | | | | |
| In 7.1 | { | 3 | 2 | 0.00 | -3000.0 | 0.00 | -3000.0 | 0.00 | | | | | |
| In 7.2 | | 18 | 0 | 0.00 | -3000.0 | 0.00 | -3000.0 | 0.00 | | | | | |
| In 7.3 | | 15 | 1 | | | | | | | | | | |
| In 8.1 | { | 0.00 | 0.0 | | | | | | | | | | |
| In 8.1 | | 1.00 | 1.0 | | | | | | | | | | |
| In 8.1 | | 2.00 | 0.0 | | | | | | | | | | |
| In 8.1 | | 3.00 | 1.7 | | | | | | | | | | |
| In 8.1 | | 0.00 | 0.0 | | | | | | | | | | |
| In 8.1 | | 1.00 | 0.0 | | | | | | | | | | |
| In 8.1 | | 2.00 | 0.0 | | | | | | | | | | |
| In 8.1 | | 3.00 | 1.3 | | | | | | | | | | |
| In 10.2.1 | { | 2 | 40 | 2 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 10.0 | |
| In 10.2.2 | | 1 | 300000000. | 300000000. | 7.30E-4 | 7.30E-4 | 0 | 0.00 | 0.00 | 0.00 | 0.00 | 6.0 | |
| In 10.2.3 | | 2 | 0.00 | -386.4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | | |
| In 10.2.4.1 | { | 1 | 1 | 1 | 3 | 1 | | | | | | | |
| In 10.2.4.1 | | 4 | 1 | 2 | 3 | 1 | | | | | | | |
| In 10.2.4.1 | | 5 | 1 | 2 | 4 | 1 | | | | | | | |
| In 10.2.4.1 | | 4 | 1 | 2 | 4 | 1 | | | | | | | |
| In 10.2.4.1 | | 9 | 1 | 3 | 4 | 1 | | | | | | | |
| In 10.2.4.1 | | 4 | 1 | 2 | 4 | 1 | | | | | | | |
| In 10.2.4.1 | | 13 | 2 | 1 | 4 | 1 | | | | | | | |
| In 10.2.4.1 | | 4 | 1 | 2 | 4 | 1 | | | | | | | |
| In 10.2.4.1 | | 17 | 2 | 2 | 3 | 1 | | | | | | | |
| In 10.2.4.1 | | 4 | 1 | 2 | 3 | 1 | | | | | | | |
| In 10.2.4.1 | | 21 | 2 | 9 | 11 | 1 | | | | | | | |
| In 10.2.4.1 | | 4 | 1 | 2 | 11 | 1 | | | | | | | |
| In 10.2.4.1 | | 25 | 2 | 10 | 12 | 1 | | | | | | | |
| In 10.2.4.1 | | 4 | 1 | 2 | 12 | 1 | | | | | | | |
| In 10.2.4.1 | | 29 | 2 | 11 | 12 | 1 | | | | | | | |
| In 10.2.4.1 | | 4 | 1 | 2 | 12 | 1 | | | | | | | |
| In 10.2.4.1 | | 33 | 2 | 9 | 12 | 1 | | | | | | | |
| In 10.2.4.1 | | 4 | 1 | 2 | 12 | 1 | | | | | | | |
| In 10.2.4.1 | | 37 | 2 | 10 | 11 | 1 | | | | | | | |
| In 10.2.4.1 | | 4 | 1 | 2 | 11 | 1 | | | | | | | |
| }*END | | | | | | | | | | | | | |

Coordinates
 Boundary conditions
 Load vector 1
 Load vector 2
 Load-time function 1
 Load-time function 2
 Element group
 Element nodes

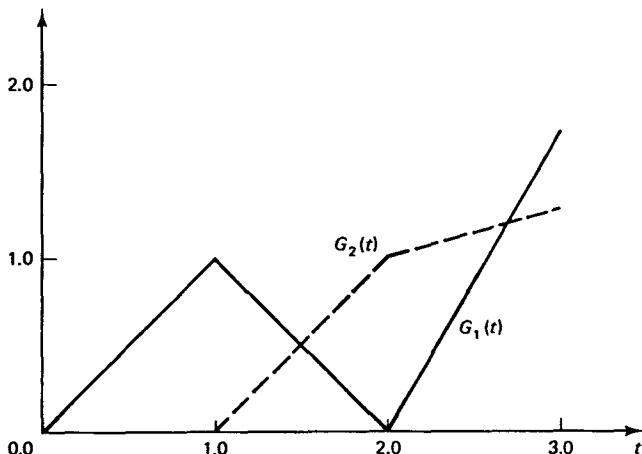


Figure 11.5.15 Load-time functions for planar truss problem.

TRUSS TOWER WITH STATIC VERTICAL AND LATERAL LOADS

EXECUTION CONTROL INFORMATION

EXECUTION CODE (IEXEC) .

ANALYSIS CODE (IACODE) :
EQ. 0, DYNAMIC ANALYSIS
EQ. 1, STATIC ANALYSIS

READ RESTART FILE CODE (IREADR) :
EQ. 0, DO NOT READ RESTART FILE
EQ. 1, READ RESTART FILE

WRITE RESTART FILE CODE (IWRITR) =
EQ. 0, DO NOT WRITE RESTART FILE
EQ. 1, WRITE RESTART FILE

INPUT DATA PRINT CODE (IPRTIN) :
EQ. 0, PRINT NODAL AND ELEMENT INPUT DATA
EQ. 1, DO NOT PRINT NODAL AND ELEMENT INPUT DATA

RANK CHECK CODE (IRANK) =
EQ. 0, DO NOT PERFORM RANK CHECK
EQ. 1, PRINT NUMBERS OF ZERO AND NEGATIVE PIVOTS
EQ. 2, PRINT ALL PIVOTS

NUMBER OF TIME SEQUENCES (NUMSEQ) =

NUMBER OF NODAL OUTPUT TIME-HISTORIES . . . (NDOUT) :

NUMBER OF SPACE DIMENSIONS (NSD) :

NUMBER OF NODAL POINTS (NUMNP) :

NUMBER OF NODAL DEGREES-OF-FREEDOM . . . (NDOF) :

NUMBER OF LOAD VECTORS (NLVECT) :

NUMBER OF LOAD-TIME FUNCTIONS (NLTFN) :

NUMBER OF POINTS ON LOAD-TIME FUNCTIONS . . (NPTSLF) :

NUMBER OF ELEMENT GROUPS (NUMEG) =

TIME SEQUENCE DATA

NUMBER OF TIME SEQUENCES (NUMSEQ) = ^ 1

TIME SEQUENCE NUMBER (N) = 1
 NUMBER OF TIME STEPS (NSTEP(N)) = 3
 KINEMATIC PRINT INCREMENT (NDPRT(N)) = 1
 STRESS/STRAIN PRINT INCREMENT (NSPRT(N)) = 1
 TIME HISTORY PLOT INCREMENT (NHPLT(N)) = 0
 NUMBER OF ITERATIONS (NITER(N)) = 1
 FIRST INTEGRATION PARAMETER (ALPHA(N)) = 0.00000E+00
 SECOND INTEGRATION PARAMETER (BETA(N)) = 1.00000E+00
 THIRD INTEGRATION PARAMETER (GAMMA(N)) = 0.00000E+00
 TIME STEP (DT(N)) = 1.00000E+00

NODE COORDINATE DATA

| NODE NO. | X1 | X2 | X3 |
|----------|----------------|----------------|----------------|
| 1 | 0.00000000E+00 | 0.00000000E+00 | 0.00000000E+00 |
| 2 | 1.44000000E+02 | 0.00000000E+00 | 0.00000000E+00 |
| 3 | 6.00000000E+00 | 9.60000000E+01 | 0.00000000E+00 |
| 4 | 1.38000000E+02 | 9.60000000E+01 | 0.00000000E+00 |
| 5 | 1.20000000E+01 | 1.92000000E+02 | 0.00000000E+00 |
| 6 | 1.32000000E+02 | 1.92000000E+02 | 0.00000000E+00 |
| 7 | 1.80000000E+01 | 2.88000000E+02 | 0.00000000E+00 |
| 8 | 1.26000000E+02 | 2.88000000E+02 | 0.00000000E+00 |
| 9 | 2.40000000E+01 | 3.84000000E+02 | 0.00000000E+00 |
| 10 | 1.20000000E+02 | 3.84000000E+02 | 0.00000000E+00 |
| 11 | 3.00000000E+01 | 4.80000000E+02 | 0.00000000E+00 |
| 12 | 1.14000000E+02 | 4.80000000E+02 | 0.00000000E+00 |
| 13 | 3.60000000E+01 | 5.76000000E+02 | 0.00000000E+00 |
| 14 | 1.08000000E+02 | 5.76000000E+02 | 0.00000000E+00 |
| 15 | 4.20000000E+01 | 6.72000000E+02 | 0.00000000E+00 |
| 16 | 1.02000000E+02 | 6.72000000E+02 | 0.00000000E+00 |
| 17 | 4.80000000E+01 | 7.68000000E+02 | 0.00000000E+00 |
| 18 | 9.60000000E+01 | 7.68000000E+02 | 0.00000000E+00 |

NODE BOUNDARY CONDITION CODES

| NODE NO. | DOF1 | DOF2 | DOF3 |
|----------|------|------|------|
| 1 | 1 | 1 | |
| 2 | 1 | 1 | |
| 3 | 0 | 1 | |
| 4 | 0 | 1 | |
| 5 | 0 | 1 | |
| 6 | 0 | 1 | |
| 7 | 0 | 1 | |
| 8 | 0 | 1 | |
| 9 | 0 | 1 | |
| 10 | 0 | 1 | |
| 11 | 0 | 1 | |
| 12 | 0 | 1 | |
| 13 | 0 | 1 | |
| 14 | 0 | 1 | |
| 15 | 0 | 1 | |
| 16 | 0 | 1 | |
| 17 | 0 | 1 | |
| 18 | 0 | 0 | 1 |

PRESCRIBED FORCES AND KINEMATIC BOUNDARY CONDITIONS

LOAD VECTOR NUMBER = 1

| NODE NO. | DOF1 | DOF2 | DOF3 |
|----------|----------------|----------------|----------------|
| 3 | 5.00000000E+02 | 0.00000000E+00 | 0.00000000E+00 |
| 5 | 1.00000000E+03 | 0.00000000E+00 | 0.00000000E+00 |
| 7 | 1.50000000E+03 | 0.00000000E+00 | 0.00000000E+00 |
| 9 | 2.00000000E+03 | 0.00000000E+00 | 0.00000000E+00 |
| 11 | 2.50000000E+03 | 0.00000000E+00 | 0.00000000E+00 |
| 13 | 3.00000000E+03 | 0.00000000E+00 | 0.00000000E+00 |
| 15 | 3.50000000E+03 | 0.00000000E+00 | 0.00000000E+00 |
| 17 | 4.00000000E+03 | 0.00000000E+00 | 0.00000000E+00 |

P R E S C R I B E D F O R C E S A N D K I N E M A T I C
B O U N D A R Y C O N D I T I O N S

L O A D V E C T O R N U M B E R = 2

| NODE NO. | DDF1 | DDF2 | DDF3 |
|----------|----------------|-----------------|----------------|
| 3 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 4 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 5 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 6 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 7 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 8 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 9 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 10 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 11 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 12 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 13 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 14 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 15 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 16 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 17 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |
| 18 | 0.00000000E+00 | -3.00000000E+03 | 0.00000000E+00 |

L O A D - T I M E F U N C T I O N D A T A

N U M B E R O F L O A D - T I M E F U N T I O N S (N L T F T N) = 2

F U N C T I O N N U M B E R 1

| T I M E | L O A D F A C T O R |
|----------------|---------------------|
| 0.00000000E+00 | 0.00000000E+00 |
| 1.00000000E+00 | 1.00000000E+00 |
| 2.00000000E+00 | 0.00000000E+00 |
| 3.00000000E+00 | 1.70000000E+00 |

F U N C T I O N N U M B E R 2

| T I M E | L O A D F A C T O R |
|----------------|---------------------|
| 0.00000000E+00 | 0.00000000E+00 |
| 1.00000000E+00 | 0.00000000E+00 |
| 2.00000000E+00 | 1.00000000E+00 |
| 3.00000000E+00 | 1.30000000E+00 |

E L E M E N T G R O U P D A T A

E L E M E N T G R O U P N U M B E R (N E G) = 1

T W O / T H R E E - N O D E T R U S S E L E M E N T S

E L E M E N T T Y P E N U M B E R (N T Y P E) = 2

N U M B E R O F E L E M E N T S (N U M E L) = 40

N U M B E R O F E L E M E N T M A T E R I A L S E T S (N U M A T) = 2

N U M B E R O F E L E M E N T N O D E S (N E N) = 2

N U M B E R O F S T R E S S / S T R A I N T I M E H I S T O R I E S . . . (N S O U T) = 0

S T R E S S O U T P U T P R I N T C O D E (I S T P R T) = 0

E Q . 0 , S T R E S S O U T P U T P R I N T E D
E Q . 1 , S T R E S S O U T P U T N O T P R I N T E D

B O D Y F O R C E L O A D - T I M E F U N C T I O N N U M B E R . . . (L F B O D Y) = 0

I N T E G R A T I O N C O D E (N I N T) = 1

E Q . 1 , 1 - P O I N T G A U S S I A N Q U A D R A T U R E
E Q . 2 , 2 - P O I N T G A U S S I A N Q U A D R A T U R E
E Q . 3 , 3 - P O I N T G A U S S I A N Q U A D R A T U R E

MATERIAL SET DATA

NUMBER OF MATERIAL SETS (NUMAT^) = 2

| SET NUMBER | YOUNG'S MODULUS | MASS DENSITY | MASS DAMPING | STIFFNESS DAMPING | AREA |
|------------|-----------------|--------------|--------------|-------------------|------------|
| 1 | 3.0000E+07 | 7.3000E-04 | 0.0000E+00 | 0.0000E+00 | 1.0000E+01 |
| 2 | 3.0000E+07 | 7.3000E-04 | 0.0000E+00 | 0.0000E+00 | 6.0000E+00 |

GRAVITY VECTOR COMPONENTS

X-1 DIRECTION = 0.00000000E+00

X-2 DIRECTION = -3.86400000E+02

X-3 DIRECTION = 0.00000000E+00

ELEMENT DATA

| ELEMENT | MATERIAL | NODE 1 | NODE 2 |
|---------|----------|--------|--------|
| 1 | 1 | 1 | 3 |
| 2 | 1 | 5 | 5 |
| 3 | 1 | 7 | 9 |
| 4 | 1 | 2 | 4 |
| 5 | 1 | 4 | 6 |
| 6 | 1 | 6 | 8 |
| 7 | 1 | 8 | 10 |
| 8 | 1 | 3 | 4 |
| 9 | 1 | 5 | 6 |
| 10 | 1 | 7 | 8 |
| 11 | 1 | 9 | 10 |
| 12 | 1 | 1 | 4 |
| 13 | 2 | 3 | 6 |
| 14 | 2 | 5 | 8 |
| 15 | 2 | 7 | 10 |
| 16 | 2 | 2 | 3 |
| 17 | 2 | 4 | 5 |
| 18 | 2 | 6 | 7 |
| 19 | 2 | 8 | 9 |
| 20 | 2 | 9 | 11 |
| 21 | 2 | 11 | 13 |
| 22 | 2 | 13 | 15 |
| 23 | 2 | 15 | 17 |
| 24 | 2 | 10 | 12 |
| 25 | 2 | 12 | 14 |
| 26 | 2 | 14 | 16 |
| 27 | 2 | 16 | 18 |
| 28 | 2 | 11 | 12 |
| 29 | 2 | 13 | 14 |
| 30 | 2 | 15 | 16 |
| 31 | 2 | 17 | 18 |
| 32 | 2 | 9 | 12 |
| 33 | 2 | 11 | 14 |
| 34 | 2 | 13 | 16 |
| 35 | 2 | 15 | 18 |
| 36 | 2 | 10 | 11 |
| 37 | 2 | 12 | 13 |
| 38 | 2 | 14 | 15 |
| 39 | 2 | 16 | 17 |
| 40 | 2 | | |

TRUSS TOWER WITH STATIC VERTICAL AND LATERAL LOADS

EQUATION SYSTEM DATA

NUMBER OF EQUATIONS (NEQ) = 32
 NUMBER OF TERMS IN LEFT-HAND-SIDE MATRIX . (NALHS) = 192
 MEAN HALF BANDWIDTH' (MEANBW) = 6
 TOTAL LENGTH OF BLANK COMMON REQUIRED . . . (NWORDS) = 2012

DISPLACEMENTS

STEP NUMBER = 1
 TIME = 1.000E+00

| NODE NO. | DOF1 | DOF2 | DOF3 |
|----------|-----------------|-----------------|----------------|
| 3 | -3.55458742E-02 | -3.80776188E-02 | 0.00000000E+00 |
| 4 | -3.39701332E-02 | 2.82149641E-02 | 0.00000000E+00 |
| 5 | -1.18072704E-01 | -6.85865605E-02 | 0.00000000E+00 |
| 6 | -1.16727240E-01 | 4.99146888E-02 | 0.00000000E+00 |
| 7 | -2.46087075E-01 | -9.10080677E-02 | 0.00000000E+00 |
| 8 | -2.44821303E-01 | 6.49165276E-02 | 0.00000000E+00 |
| 9 | -4.15844342E-01 | -1.05185369E-01 | 0.00000000E+00 |
| 10 | -4.14547637E-01 | 7.29927351E-02 | 0.00000000E+00 |
| 11 | -6.33329338E-01 | -1.22704195E-01 | 0.00000000E+00 |
| 12 | -6.31285639E-01 | 8.31843360E-02 | 0.00000000E+00 |
| 13 | -9.00356267E-01 | -1.27236953E-01 | 0.00000000E+00 |
| 14 | -8.98780674E-01 | 8.21529828E-02 | 0.00000000E+00 |
| 15 | -1.20224430E+00 | -1.20066166E-01 | 0.00000000E+00 |
| 16 | -1.20101352E+00 | 7.13913000E-02 | 0.00000000E+00 |
| 17 | -1.52011008E+00 | -1.04034181E-01 | 0.00000000E+00 |
| 18 | -1.51917308E+00 | 5.35084016E-02 | 0.00000000E+00 |

ENT STRESSES AND STRAINS

ENT GROUP NUMBER (NEG) = 1

| INT. PT. NUMBER | X1 COORD. | X2 COORD. | X3 COORD. | STRESS | FORCE | STRAIN |
|--------------------|--------------|--------------|--------------|-----------|-----------|-----------|
| 1 | 3.00E+00 | 4.80E+01 | 0.00E+00 | -1.25E+04 | -1.25E+05 | -4.18E-04 |
| 1 | 9.00E+00 | 1.44E+02 | 0.00E+00 | -1.11E+04 | -1.11E+05 | -3.70E-04 |
| 1 | 1.50E+01 | 2.40E+02 | 0.00E+00 | -9.47E+03 | -9.47E+04 | -3.16E-04 |
| 1 | 2.10E+01 | 3.36E+02 | 0.00E+00 | -7.72E+03 | -7.72E+04 | -2.57E-04 |
| 1 | 1.41E+02 | 4.80E+01 | 0.00E+00 | 9.44E+03 | 9.44E+04 | 3.15E-04 |
| 1 | 1.35E+02 | 1.44E+02 | 0.00E+00 | 8.36E+03 | 8.36E+04 | 2.79E-04 |
| 1 | 1.29E+02 | 2.40E+02 | 0.00E+00 | 7.16E+03 | 7.16E+04 | 2.39E-04 |
| 1 | 1.23E+02 | 3.36E+02 | 0.00E+00 | 5.82E+03 | 5.82E+04 | 1.94E-04 |
| 1 | 7.20E+01 | 9.60E+01 | 0.00E+00 | 3.58E+02 | 3.58E+03 | 1.19E-05 |
| 1 | 7.20E+01 | 1.92E+02 | 0.00E+00 | 3.36E+02 | 3.36E+03 | 1.12E-05 |
| 1 | 7.20E+01 | 2.88E+02 | 0.00E+00 | 3.52E+02 | 3.52E+03 | 1.17E-05 |
| 1 | 7.20E+01 | 3.84E+02 | 0.00E+00 | 4.05E+02 | 4.05E+03 | 1.35E-05 |
| 1 | 6.90E+01 | 4.80E+01 | 0.00E+00 | -2.10E+03 | -1.26E+04 | -7.00E-05 |
| 1 | 6.90E+01 | 1.44E+02 | 0.00E+00 | -2.13E+03 | -1.28E+04 | -7.10E-05 |
| 1 | 6.90E+01 | 2.40E+02 | 0.00E+00 | -2.21E+03 | -1.32E+04 | -7.35E-05 |
| 1 | 6.90E+01 | 3.36E+02 | 0.00E+00 | -2.20E+03 | -1.32E+04 | -7.33E-05 |
| 1 | 7.50E+01 | 4.80E+01 | 0.00E+00 | 1.33E+03 | 7.96E+03 | 4.42E-05 |
| 1 | 7.50E+01 | 1.44E+02 | 0.00E+00 | 1.56E+03 | 9.35E+03 | 5.20E-05 |
| 1 | 7.50E+01 | 2.40E+02 | 0.00E+00 | 1.65E+03 | 9.87E+03 | 5.49E-05 |
| 1 | 7.50E+01 | 3.36E+02 | 0.00E+00 | 1.70E+03 | 1.02E+04 | 5.68E-05 |
| 1 | 2.70E+01 | 4.32E+02 | 0.00E+00 | -9.68E+03 | -5.81E+04 | -3.23E-04 |
| 1 | 3.30E+01 | 5.28E+02 | 0.00E+00 | -6.61E+03 | -3.96E+04 | -2.20E-04 |
| 1 | 3.90E+01 | 6.24E+02 | 0.00E+00 | -3.64E+03 | -2.18E+04 | -1.21E-04 |
| 1 | 4.50E+01 | 7.20E+02 | 0.00E+00 | -1.19E+03 | -7.16E+03 | -3.98E-05 |

ELEMENT STRESSES AND STRAINS

ELEMENT GROUP NUMBER (NEG) = 1

| ELEMENT NUMBER | INT. PT. NUMBER | X1 COORD. | X2 COORD. | X3 COORD. | STRESS | FORCE | STRA |
|----------------|-----------------|-----------|-----------|-----------|-----------|-----------|--------|
| 25 | 1 | 1.17E+02 | 4.32E+02 | 0.00E+00 | 7.39E+03 | 4.43E+04 | 2.46E |
| 26 | 1 | 1.11E+02 | 5.28E+02 | 0.00E+00 | 4.88E+03 | 2.93E+04 | 1.63E |
| 27 | 1 | 1.05E+02 | 6.24E+02 | 0.00E+00 | 2.53E+03 | 1.52E+04 | 8.43E |
| 28 | 1 | 9.90E+01 | 7.20E+02 | 0.00E+00 | 6.23E+02 | 3.74E+03 | 2.08E |
| 29 | 1 | 7.20E+01 | 4.80E+02 | 0.00E+00 | 7.30E+02 | 4.38E+03 | 2.43E |
| 30 | 1 | 7.20E+01 | 5.76E+02 | 0.00E+00 | 6.61E+02 | 3.96E+03 | 2.20E |
| 31 | 1 | 7.20E+01 | 6.72E+02 | 0.00E+00 | 6.15E+02 | 3.69E+03 | 2.05E |
| 32 | 1 | 7.20E+01 | 7.68E+02 | 0.00E+00 | 5.86E+02 | 3.51E+03 | 1.95E |
| 33 | 1 | 6.90E+01 | 4.32E+02 | 0.00E+00 | -2.26E+03 | -1.36E+04 | -7.54E |
| 34 | 1 | 6.90E+01 | 5.28E+02 | 0.00E+00 | -2.04E+03 | -1.22E+04 | -6.79E |
| 35 | 1 | 6.90E+01 | 6.24E+02 | 0.00E+00 | -1.71E+03 | -1.03E+04 | -5.71E |
| 36 | 1 | 6.90E+01 | 7.20E+02 | 0.00E+00 | -1.12E+03 | -6.69E+03 | -3.72E |
| 37 | 1 | 7.50E+01 | 4.32E+02 | 0.00E+00 | 1.57E+03 | 9.39E+03 | 5.22E |
| 38 | 1 | 7.50E+01 | 5.28E+02 | 0.00E+00 | 1.54E+03 | 9.27E+03 | 5.15E |
| 39 | 1 | 7.50E+01 | 6.24E+02 | 0.00E+00 | 1.36E+03 | 8.16E+03 | 4.53E |
| 40 | 1 | 7.50E+01 | 7.20E+02 | 0.00E+00 | 9.65E+02 | 5.79E+03 | 3.22E |

DISPLACEMENTS

STEP NUMBER = 2

TIME = 2.000E+00

| NODE NO. | DOF1 | DOF2 | DOF3 |
|----------|-----------------|------------------|----------------|
| 3 | -1.03904457E-03 | -7.03705120E-03 | 0.00000000E+00 |
| 4 | 1.03904457E-03 | -7.03705120E-03 | 0.00000000E+00 |
| 5 | -7.97810718E-04 | -1.32947618E-02 | 0.00000000E+00 |
| 6 | 7.97810718E-04 | -1.32947618E-02 | 0.00000000E+00 |
| 7 | -6.72223927E-04 | -1.85330487E-02 | 0.00000000E+00 |
| 8 | 6.72223927E-04 | -1.85330487E-02 | 0.00000000E+00 |
| 9 | -6.43972712E-04 | -2.28047225E-02 | 0.00000000E+00 |
| 10 | 6.43972712E-04 | -2.28047225E-02 | 0.00000000E+00 |
| 11 | -9.38067044E-04 | -2.78375422E-02 | 0.00000000E+00 |
| 12 | 9.38067044E-04 | -2.78375422E-02 | 0.00000000E+00 |
| 13 | -5.6420278E-04 | -3.15277555E-02 | 0.00000000E+00 |
| 14 | 5.6420278E-04 | -3.15277555E-02 | 0.00000000E+00 |
| 15 | -2.80376309E-04 | -3.380777792E-02 | 0.00000000E+00 |
| 16 | 2.80376309E-04 | -3.380777792E-02 | 0.00000000E+00 |
| 17 | -6.37576302E-05 | -3.48593216E-02 | 0.00000000E+00 |
| 18 | 6.37576302E-05 | -3.48593216E-02 | 0.00000000E+00 |

INT STRESSES AND STRAINS

INT GROUP NUMBER (NEG) = 1

| INT. PT. NUMBER | X1 COORD. | X2 COORD. | X3 COORD. | STRESS | FORCE | STRAIN |
|--------------------|--------------|--------------|--------------|-----------|-----------|-----------|
| 1 | 3.00E+00 | 4.80E+01 | 0.00E+00 | -2.21E+03 | -2.21E+04 | -7.37E-05 |
| 1 | 9.00E+00 | 1.44E+02 | 0.00E+00 | -1.94E+03 | -1.94E+04 | -6.48E-05 |
| 1 | 1.50E+01 | 2.40E+02 | 0.00E+00 | -1.63E+03 | -1.63E+04 | -5.43E-05 |
| 1 | 2.10E+01 | 3.36E+02 | 0.00E+00 | -1.33E+03 | -1.33E+04 | -4.43E-05 |
| 1 | 1.41E+02 | 4.80E+01 | 0.00E+00 | -2.21E+03 | -2.21E+04 | -7.37E-05 |
| 1 | 1.35E+02 | 1.44E+02 | 0.00E+00 | -1.94E+03 | -1.94E+04 | -6.48E-05 |
| 1 | 1.29E+02 | 2.40E+02 | 0.00E+00 | -1.63E+03 | -1.63E+04 | -5.43E-05 |
| 1 | 1.23E+02 | 3.36E+02 | 0.00E+00 | -1.33E+03 | -1.33E+04 | -4.43E-05 |
| 1 | 7.20E+01 | 9.60E+01 | 0.00E+00 | 4.72E+02 | 4.72E+03 | 1.57E-05 |
| 1 | 7.20E+01 | 1.92E+02 | 0.00E+00 | 3.99E+02 | 3.99E+03 | 1.33E-05 |
| 1 | 7.20E+01 | 2.88E+02 | 0.00E+00 | 3.73E+02 | 3.73E+03 | 1.24E-05 |
| 1 | 7.20E+01 | 3.84E+02 | 0.00E+00 | 4.02E+02 | 4.02E+03 | 1.34E-05 |
| 1 | 6.90E+01 | 4.80E+01 | 0.00E+00 | -5.65E+02 | -3.39E+03 | -1.88E-05 |
| 1 | 6.90E+01 | 1.44E+02 | 0.00E+00 | -4.42E+02 | -2.65E+03 | -1.47E-05 |
| 1 | 6.90E+01 | 2.40E+02 | 0.00E+00 | -4.53E+02 | -2.72E+03 | -1.51E-05 |
| 1 | 6.90E+01 | 3.36E+02 | 0.00E+00 | -4.22E+02 | -2.53E+03 | -1.41E-05 |
| 1 | 7.50E+01 | 4.80E+01 | 0.00E+00 | -5.65E+02 | -3.39E+03 | -1.88E-05 |
| 1 | 7.50E+01 | 1.44E+02 | 0.00E+00 | -4.42E+02 | -2.65E+03 | -1.47E-05 |
| 1 | 7.50E+01 | 2.40E+02 | 0.00E+00 | -4.53E+02 | -2.72E+03 | -1.51E-05 |
| 1 | 7.50E+01 | 3.36E+02 | 0.00E+00 | -4.22E+02 | -2.53E+03 | -1.41E-05 |
| 1 | 2.70E+01 | 4.32E+02 | 0.00E+00 | -1.57E+03 | -9.43E+03 | -5.24E-05 |
| 1 | 3.30E+01 | 5.28E+02 | 0.00E+00 | -1.14E+03 | -6.85E+03 | -3.80E-05 |
| 1 | 3.90E+01 | 6.24E+02 | 0.00E+00 | -7.04E+02 | -4.23E+03 | -2.35E-05 |
| 1 | 4.50E+01 | 7.20E+02 | 0.00E+00 | -3.23E+02 | -1.94E+03 | -1.08E-05 |

ELEMENT STRESSES AND STRAINS

ELEMENT GROUP NUMBER (NEG) = 1

| ELEMENT NUMBER | INT. PT. NUMBER | X1 COORD. | X2 COORD. | X3 COORD. | STRESS | FORCE | STRAI |
|----------------|-----------------|-----------|-----------|-----------|-----------|-----------|---------|
| 25 | 1 | 1.17E+02 | 4.32E+02 | 0.00E+00 | -1.57E+03 | -9.43E+03 | -5.24E- |
| 26 | 1 | 1.11E+02 | 5.28E+02 | 0.00E+00 | -1.14E+03 | -6.85E+03 | -3.80E- |
| 27 | 1 | 1.05E+02 | 6.24E+02 | 0.00E+00 | -7.04E+02 | -4.23E+03 | -2.35E- |
| 28 | 1 | 9.90E+01 | 7.20E+02 | 0.00E+00 | -3.23E+02 | -1.94E+03 | -1.08E- |
| 29 | 1 | 7.20E+01 | 4.80E+02 | 0.00E+00 | 6.70E+02 | 4.02E+03 | 2.23E- |
| 30 | 1 | 7.20E+01 | 5.76E+02 | 0.00E+00 | 4.70E+02 | 2.82E+03 | 1.57E- |
| 31 | 1 | 7.20E+01 | 6.72E+02 | 0.00E+00 | 2.80E+02 | 1.68E+03 | 9.35E- |
| 32 | 1 | 7.20E+01 | 7.68E+02 | 0.00E+00 | 7.97E+01 | 4.78E+02 | 2.66E- |
| 33 | 1 | 6.90E+01 | 4.32E+02 | 0.00E+00 | -5.90E+02 | -3.54E+03 | -1.97E- |
| 34 | 1 | 6.90E+01 | 5.28E+02 | 0.00E+00 | -4.65E+02 | -2.79E+03 | -1.55E- |
| 35 | 1 | 6.90E+01 | 6.24E+02 | 0.00E+00 | -3.61E+02 | -2.16E+03 | -1.20E- |
| 36 | 1 | 6.90E+01 | 7.20E+02 | 0.00E+00 | -2.04E+02 | -1.22E+03 | -6.79E- |
| 37 | 1 | 7.50E+01 | 4.32E+02 | 0.00E+00 | -5.90E+02 | -3.54E+03 | -1.97E- |
| 38 | 1 | 7.50E+01 | 5.28E+02 | 0.00E+00 | -4.65E+02 | -2.79E+03 | -1.55E- |
| 39 | 1 | 7.50E+01 | 6.24E+02 | 0.00E+00 | -3.61E+02 | -2.16E+03 | -1.20E- |
| 40 | 1 | 7.50E+01 | 7.20E+02 | 0.00E+00 | -2.04E+02 | -1.22E+03 | -6.79E- |

DISPLACEMENTS

STEP NUMBER = 3

TIME = 3.000E+00

| NODE NO. | DOF1 | DOF2 | DOF3 |
|----------|-----------------|-----------------|----------------|
| 3 | 3.34677851E-02 | 2.40035164E-02 | 0.00000000E+00 |
| 4 | 3.60482223E-02 | -4.22890665E-02 | 0.00000000E+00 |
| 5 | 1.16477083E-01 | 4.19970369E-02 | 0.00000000E+00 |
| 6 | 1.183222861E-01 | -7.65042123E-02 | 0.00000000E+00 |
| 7 | 2.44742627E-01 | 5.39459704E-02 | 0.00000000E+00 |
| 8 | 2.46169251E-01 | -1.01982625E-01 | 0.00000000E+00 |
| 9 | 4.14556339E-01 | 5.95759242E-02 | 0.00000000E+00 |
| 10 | 4.15335582E-01 | -1.18022200E-01 | 0.00000000E+00 |
| 11 | 6.31453204E-01 | 6.70291111E-02 | 0.00000000E+00 |
| 12 | 6.33161773E-01 | -1.38859420E-01 | 0.00000000E+00 |
| 13 | 8.99237866E-01 | 6.41814424E-02 | 0.00000000E+00 |
| 14 | 8.99909074E-01 | -1.45212494E-01 | 0.00000000E+00 |
| 15 | 1.201638354E+00 | 5.24506079E-02 | 0.00000000E+00 |
| 16 | 1.20157427E+00 | -1.39006859E-01 | 0.00000000E+00 |
| 17 | 1.51998256E+00 | 3.43155377E-02 | 0.00000000E+00 |
| 18 | 1.51930059E+00 | -1.23227045E-01 | 0.00000000E+00 |

ELEMENT STRESSES AND STRAINS

ELEMENT GROUP NUMBER (NEG) = 1

| ELEMENT NUMBER | INT. PT. NUMBER | X1 COORD. | X2 COORD. | X3 COORD. | STRESS | FORCE | STRAIN |
|----------------|-----------------|-----------|-----------|-----------|-----------|-----------|-----------|
| 1 | 1 | 3.00E+00 | 4.80E+01 | 0.00E+00 | 8.12E+03 | 8.12E+04 | 2.71E-04 |
| 2 | 1 | 9.00E+00 | 1.44E+02 | 0.00E+00 | 7.22E+03 | 7.22E+04 | 2.41E-04 |
| 3 | 1 | 1.50E+01 | 2.40E+02 | 0.00E+00 | 6.21E+03 | 6.21E+04 | 2.07E-04 |
| 4 | 1 | 2.10E+01 | 3.36E+02 | 0.00E+00 | 5.06E+03 | 5.06E+04 | 1.69E-04 |
| 5 | 1 | 1.41E+02 | 4.80E+01 | 0.00E+00 | -1.39E+04 | -1.39E+05 | -4.62E-04 |
| 6 | 1 | 1.35E+02 | 1.44E+02 | 0.00E+00 | -1.23E+04 | -1.23E+05 | -4.08E-04 |
| 7 | 1 | 1.29E+02 | 2.40E+02 | 0.00E+00 | -1.04E+04 | -1.04E+05 | -3.47E-04 |
| 8 | 1 | 1.23E+02 | 3.36E+02 | 0.00E+00 | -8.47E+03 | -8.47E+04 | -2.82E-04 |
| 9 | 1 | 7.20E+01 | 9.60E+01 | 0.00E+00 | 5.86E+02 | 5.86E+03 | 1.95E-05 |
| 10 | 1 | 7.20E+01 | 1.92E+02 | 0.00E+00 | 4.61E+02 | 4.61E+03 | 1.54E-05 |
| 11 | 1 | 7.20E+01 | 2.88E+02 | 0.00E+00 | 3.95E+02 | 3.95E+03 | 1.32E-05 |
| 12 | 1 | 7.20E+01 | 3.84E+02 | 0.00E+00 | 4.00E+02 | 4.00E+03 | 1.33E-05 |
| 13 | 1 | 6.90E+01 | 4.80E+01 | 0.00E+00 | 9.71E+02 | 5.83E+03 | 3.24E-05 |
| 14 | 1 | 6.90E+01 | 1.44E+02 | 0.00E+00 | 1.25E+03 | 7.48E+03 | 4.16E-05 |
| 15 | 1 | 6.90E+01 | 2.40E+02 | 0.00E+00 | 1.30E+03 | 7.80E+03 | 4.33E-05 |
| 16 | 1 | 6.90E+01 | 3.36E+02 | 0.00E+00 | 1.36E+03 | 8.14E+03 | 4.52E-05 |
| 17 | 1 | 7.50E+01 | 4.80E+01 | 0.00E+00 | -2.46E+03 | -1.47E+04 | -8.19E-05 |
| 18 | 1 | 7.50E+01 | 1.44E+02 | 0.00E+00 | -2.44E+03 | -1.47E+04 | -8.14E-05 |
| 19 | 1 | 7.50E+01 | 2.40E+02 | 0.00E+00 | -2.55E+03 | -1.53E+04 | -8.50E-05 |
| 20 | 1 | 7.50E+01 | 3.36E+02 | 0.00E+00 | -2.55E+03 | -1.53E+04 | -8.49E-05 |
| 21 | 1 | 2.70E+01 | 4.32E+02 | 0.00E+00 | 6.54E+03 | 3.92E+04 | 2.18E-04 |
| 22 | 1 | 3.30E+01 | 5.28E+02 | 0.00E+00 | 4.32E+03 | 2.59E+04 | 1.44E-04 |
| 23 | 1 | 3.90E+01 | 6.24E+02 | 0.00E+00 | 2.23E+03 | 1.34E+04 | 7.44E-05 |
| 24 | 1 | 4.50E+01 | 7.20E+02 | 0.00E+00 | 5.47E+02 | 3.28E+03 | 1.82E-05 |

ELEMENT STRESSES AND STRAINS

ELEMENT GROUP NUMBER (NEG) = 1

| ELEMENT NUMBER | INT. PT. NUMBER | X1 COORD. | X2 COORD. | X3 COORD. | STRESS | FORCE | STRAI |
|-------------------|-----------------------|--------------|--------------|--------------|-----------|-----------|---------|
| 25 | 1 | 1.17E+02 | 4.32E+02 | 0.00E+00 | -1.05E+04 | -6.32E+04 | -3.51E- |
| 26 | 1 | 1.11E+02 | 5.28E+02 | 0.00E+00 | -7.17E+03 | -4.30E+04 | -2.39E- |
| 27 | 1 | 1.05E+02 | 6.24E+02 | 0.00E+00 | -3.94E+03 | -2.36E+04 | -1.31E- |
| 28 | 1 | 9.90E+01 | 7.20E+02 | 0.00E+00 | -1.27E+03 | -7.62E+03 | -4.23E- |
| 29 | 1 | 7.20E+01 | 4.80E+02 | 0.00E+00 | 6.10E+02 | 3.66E+03 | 2.03E- |
| 30 | 1 | 7.20E+01 | 5.76E+02 | 0.00E+00 | 2.80E+02 | 1.68E+03 | 9.32E- |
| 31 | 1 | 7.20E+01 | 6.72E+02 | 0.00E+00 | -5.46E+01 | -3.28E+02 | -1.82E- |
| 32 | 1 | 7.20E+01 | 7.68E+02 | 0.00E+00 | -4.26E+02 | -2.56E+03 | -1.42E- |
| 33 | 1 | 6.90E+01 | 4.32E+02 | 0.00E+00 | 1.08E+03 | 6.49E+03 | 3.61E- |
| 34 | 1 | 6.90E+01 | 5.28E+02 | 0.00E+00 | 1.11E+03 | 6.64E+03 | 3.69E- |
| 35 | 1 | 6.90E+01 | 6.24E+02 | 0.00E+00 | 9.91E+02 | 5.94E+03 | 3.30E- |
| 36 | 1 | 6.90E+01 | 7.20E+02 | 0.00E+00 | 7.08E+02 | 4.25E+03 | 2.36E- |
| 37 | 1 | 7.50E+01 | 4.32E+02 | 0.00E+00 | -2.75E+03 | -1.65E+04 | -9.15E- |
| 38 | 1 | 7.50E+01 | 5.28E+02 | 0.00E+00 | -2.47E+03 | -1.48E+04 | -8.25E- |
| 39 | 1 | 7.50E+01 | 6.24E+02 | 0.00E+00 | -2.08E+03 | -1.25E+04 | -6.94E- |
| 40 | 1 | 7.50E+01 | 7.20E+02 | 0.00E+00 | -1.37E+03 | -8.24E+03 | -4.58E- |

| A M I C S T O R A G E | | ALLOCATION | INFORMATION | | | |
|----------------------------------|--------|------------|-------------|------|------|-------|
| ARRAY NO. | ARRAY | ADDRESS | DIM1 | DIM2 | DIM3 | PREC. |
| 1 | NSTEP | 1 | 1 | 0 | 0 | 1 |
| 2 | NDPRT | 3 | 1 | 0 | 0 | 1 |
| 3 | NSPRT | 5 | 1 | 0 | 0 | 1 |
| 4 | NHPLT | 7 | 1 | 0 | 0 | 1 |
| 5 | NITER | 9 | 1 | 0 | 0 | 1 |
| 6 | ALPHA | 11 | 1 | 0 | 0 | 2 |
| 7 | BETA | 13 | 1 | 0 | 0 | 2 |
| 8 | GAMMA | 15 | 1 | 0 | 0 | 2 |
| 9 | DT | 17 | 1 | 0 | 0 | 2 |
| 10 | X | 19 | 1 | 18 | 0 | 2 |
| 11 | ID | 127 | 1 | 18 | 0 | 1 |
| 12 | FF | 235 | 1 | 18 | 0 | 2 |
| 13 | G | 289 | 4 | 18 | 0 | 2 |
| 14 | G1 | 505 | 2 | 2 | 0 | 2 |
| 15 | IDIAG | 537 | 32 | 0 | 0 | 2 |
| 16 | NGRP | 541 | 1 | 0 | 0 | 1 |
| 17 | | 573 | 1 | 0 | 0 | 1 |
| ***** BEGIN ELEMENT GROUP NUMBER | | 1 | | | | |
| 18 | NPAR | 575 | 16 | 0 | 0 | 1 |
| 19 | MP | 591 | 29 | 0 | 0 | 1 |
| 20 | W | 621 | 1 | 0 | 0 | 2 |
| 21 | DET | 623 | 1 | 0 | 0 | 1 |
| 22 | SHL | 625 | 1 | 0 | 0 | 1 |
| 23 | SHG | 633 | 1 | 0 | 0 | 1 |
| 24 | XS | 641 | 1 | 0 | 0 | 1 |
| 25 | RHO | 647 | 1 | 0 | 0 | 1 |
| 26 | RDAMPM | 651 | 2 | 0 | 0 | 1 |
| 27 | RDAMPK | 655 | 2 | 0 | 0 | 1 |
| 28 | AREA | 659 | 1 | 0 | 0 | 1 |
| 29 | C | 663 | 1 | 0 | 0 | 2 |
| 30 | GRAV | 667 | 1 | 0 | 0 | 2 |
| 31 | TEN | 673 | 1 | 40 | 0 | 1 |
| 32 | MAT | 753 | 40 | 0 | 0 | 1 |
| 33 | LM | 793 | 3 | 0 | 0 | 1 |
| 34 | ELEFFM | 1033 | 6 | 0 | 0 | 1 |
| 35 | XL | 1105 | 3 | 0 | 0 | 2 |
| 36 | NORK | 1117 | 16 | 0 | 0 | 2 |
| 37 | B | 1149 | 1 | 0 | 0 | 1 |
| 38 | DMAT | 1161 | 1 | 0 | 0 | 1 |
| 39 | DB | 1163 | 1 | 0 | 0 | 1 |
| 40 | VL | 1175 | 3 | 0 | 0 | 1 |
| 41 | AL | 1187 | 3 | 0 | 0 | 1 |
| 42 | ELRESF | 1199 | 6 | 0 | 0 | 1 |
| 43 | DL | 1211 | 3 | 0 | 0 | 1 |
| 44 | STRAIN | 1223 | 1 | 0 | 0 | 1 |
| 45 | STRESS | 1225 | 1 | 0 | 0 | 1 |
| 46 | FORCE | 1227 | 1 | 0 | 0 | 1 |
| ***** END ELEMENT GROUP DATA | | | | | | |
| 47 | ALHS | 1229 | 192 | 0 | 0 | 2 |
| 48 | BRHS | 1613 | 32 | 0 | 0 | 2 |

TRUSS TOWER WITH STATIC VERTICAL AND LATERAL LOADS

EXECUTION TIMING INFORMATION

INITIALIZATION PHASE = 1.578E+00

SOLUTION PHASE = 2.461E+00

FORMATION OF LEFT-HAND-SIDE MATRICES = 1.797E-01

FACTORIZATIONS = 1.953E-02

FORMATION OF RIGHT-HAND-SIDE VECTORS = 5.859E-02

FORWARD REDUCTIONS/BACK SUBSTITUTIONS = 1.953E-02

CALCULATION OF ELEMENT OUTPUT = 1.781E+00

SUBTOTAL = 2.059E+00

Example 2 (Static Analysis of a Plane Strain Cantilever Beam)

This problem illustrates the use of the four-node quadrilateral, elastic continuum element. The mathematical model and corresponding mesh are shown in Figs. 4.4.1 and 4.4.2. In Table 11.5.2 we present the input file needed for one of the static analyses discussed in Chapter 4. Notice that the nodes have been numbered in a manner that minimizes the bandwidth of the effective mass matrix M^* (i.e., the stiffness matrix K). Observe that the boundary condition codes are set to allow warping of the root section of the cantilever, which is consistent with the analytic solution. The single load vector specified represents the consistent nodal loads $f_a^* = \int N_a f d\Omega$ computed from the surface tractions defined in Chapter 4. In this example, the nodal loads were computed by hand. An alternative procedure is to use the element surface loads described in Sec. In.10.1.5. However, note that the element surface loads are restricted to piecewise linear variation. The parabolic variation of end shear cannot be exactly represented by *continuous* piecewise linear variation, but it can be exactly represented by *discontinuous* piecewise linear interpolation (see Sec. 1.15, Exercise 1).

TABLE 11.5.2 Input File for Static Analysis of Cantilever Beam

| | | | |
|---------------|------|-------------------------------|--|
| In 0.0 | 0 | PLANE STRAIN CANTILEVER BEAM: | STATIC ANALYSIS WITH CONSISTENT NODAL FORCES |
| In 1.0 | 1 | 1 | 1 |
| In 2.0 | 2 | 0 | 0 |
| In 3.0 | 3 | 0 | 2 |
| In 5.1 | 4 | 0.0 | 0.0 |
| In 5.2 | 16.0 | 0.0 | 0.0 |
| | 16.0 | 2.0 | 0.0 |
| | 0.0 | 2.0 | 0.0 |
| In 5.3 | 8 | 5 | 4 |
| In 6.0 | 1 | 1 | 1 |
| | 5 | 41 | 5 |
| | 6 | 1 | 0 |
| In 7.1 | 2 | -7.500E-01 | 0.17383 |
| | 3 | -1.500E+00 | 0.13867 |
| | 4 | -2.250E+00 | 0.08008 |
| | 5 | 0.000E+00 | 0.01465 |
| | 41 | 0.000E+00 | -0.09280 |
| | 42 | 0.000E+00 | -0.17383 |
| | 43 | 0.000E+00 | -0.13867 |
| | 44 | 0.000E+00 | -0.08008 |
| | 45 | 0.000E+00 | -0.01465 |
| | 0.0 | 1.0 | 0.300 |
| In 8.1 | 1 | 0.32 | 1.00 |
| In 10.1.1 | 1 | 1.00 | 0.300 |
| In 10.1.2 | 0 | 0.0 | 0.0 |
| In 10.1.3 | 0 | 0.0 | 0.0 |
| In 10.1.4.1 | 1 | 1 | 2 |
| In 10.1.4.2 | 8 | 4 | 5 |
| Element nodes | | | |
| *END | | | |

Example 3 (Dynamic Analysis of a Plane Strain Cantilever Beam)

This problem, like the previous one, employs the configuration illustrated in Figs. 4.4.1 and 4.4.2. However, here the beam is forced dynamically. The beam is assumed initially at rest and all loads, namely, those of the static linear elasticity solution, are applied instantaneously at $t = 0^+$. The following data were employed in this calculation:

$$P(t) = 1000 H(t) \quad (\text{load factor, see Sec. 4.4})$$

$$\lambda = 1.2 \times 10^6 \quad (\text{Lamé parameters})$$

$$\mu = 0.8 \times 10^6$$

$$\rho = 2.0 \times 10^{-3} \quad (\text{density})$$

where $H(t)$ denotes the Heaviside step function (i.e., $H(t) = 1$ if $t > 0$; otherwise, $H(t) = 0$).

The dynamic response is calculated with the Hilber-Hughes-Taylor α -method. The parameters employed are

$$\alpha = -0.1, \quad \beta = 0.3025, \quad \gamma = 0.60$$

which results in an implicit, second-order accurate, unconditionally stable algorithm with some high-frequency numerical dissipation. One hundred equal time steps of length $\Delta t = 2.5 \times 10^{-4}$ were used in the analysis. This amounts to about 50 time steps for the fundamental period* and ensures an accurate integration of the response in the fundamental mode. By comparing the input files of Tables 11.5.2 and 11.5.3, it may be seen that little additional information is required to specify the dynamic problem. The same load vector is utilized as in the static case, but it is now scaled by a factor of 1000 by way of a constant load-time function. Three blank lines are inserted to indicate zero kinematic initial conditions. Notice that setting NDPR and NSPRT equal to zero minimizes the printed output. Time-history plots of the vertical displacement of the tip (i.e., node 45) and the σ_{11} -component of stress in element 20 are requested. The output file is presented on pages 707 through 715.

TABLE 11.5.3 Input File for Dynamic Analysis of a Plane Strain Cantilever Beam

* The fundamental period can be estimated from classical beam theory:

$$T_1 = \frac{2\pi}{\omega_1} \cong \frac{2\pi}{1.875^2} \sqrt{\frac{ml^4}{EI}} = \frac{2\pi}{1.875^2} \sqrt{\frac{(0.008)(16)^4}{(2.08 \times 10^6)(5.3333)}} = 1.23 \times 10^{-2}$$

where E is Young's modulus, I is the cross-sectional moment of inertia, l is the length and m denotes the mass per unit length. (See W. C. Hurty and M. F. Rubinstein, *Dynamics of Structures*, p. 203. Englewood Cliffs, N.J.: Prentice-Hall, 1964.)

PLANE STRAIN CANTILEVER BEAM: DYNAMIC ANALYSIS WITH ALPHA METHOD

EXECUTION CONTROL INFORMATION

EXECUTION CODE (IEXEC) = 1

EQ. 0; DATA CHECK

EQ. 1; EXECUTION

ANALYSIS CODE (IACODE) = 0

EQ. 0, DYNAMIC ANALYSIS

EQ. 1, STATIC ANALYSIS

READ RESTART FILE CODE (IREADR) = 0

EQ. 0, DO NOT READ RESTART FILE

EQ. 1, READ RESTART FILE

WRITE RESTART FILE CODE (IWRITR) = 0

EQ. 0, DO NOT WRITE RESTART FILE

EQ. 1, WRITE RESTART FILE

INPUT DATA PRINT CODE (IPRTIN) = 0

EQ. 0, PRINT NODAL AND ELEMENT INPUT DATA

EQ. 1, DO NOT PRINT NODAL AND ELEMENT INPUT DATA

RANK CHECK CODE (IRANK) = 0

EQ. 0, DO NOT PERFORM RANK CHECK

EQ. 1, PRINT NUMBERS OF ZERO AND NEGATIVE PIVOTS

EQ. 2, PRINT ALL PIVOTS

NUMBER OF TIME SEQUENCES (NUMSEQ) = 1

NUMBER OF NODAL OUTPUT TIME-HISTORIES . . . (NDOUT) = 1

NUMBER OF SPACE DIMENSIONS (NSD) = 2

NUMBER OF NODAL POINTS (NUMNP) = 45

NUMBER OF NODAL DEGREES-OF-FREEDOM (NDOF) = 2

NUMBER OF LOAD VECTORS (NLVECT) = 1

NUMBER OF LOAD-TIME FUNCTIONS (NLTFTN) = 1

NUMBER OF POINTS ON LOAD-TIME FUNCTIONS . . (NPTSLF) = 1

NUMBER OF ELEMENT GROUPS (NUMEG) = 1

TIME SEQUENCE DATA

NUMBER OF TIME SEQUENCES (NUMSEQ) = 1

TIME SEQUENCE NUMBER (N) = 1

NUMBER OF TIME STEPS (NSTEP(N)) = 100

KINEMATIC PRINT INCREMENT (NDPRT(N)) = 0

STRESS/STRAIN PRINT INCREMENT . . . (NSPRT(N)) = 0

TIME HISTORY PLOT INCREMENT . . . (NHPLT(N)) = 2

NUMBER OF ITERATIONS (NITER(N)) = 1

FIRST INTEGRATION PARAMETER (ALPHA(N)) = -1.00000E-01

SECOND INTEGRATION PARAMETER (BETA(N)) = 3.02500E-01

THIRD INTEGRATION PARAMETER (GAMMA(N)) = 6.00000E-01

TIME STEP (DT(N)) = 2.50000E-04

NODAL TIME-HISTORY INFORMATION

NUMBER OF NODAL TIME HISTORIES (NDOUT) = 1

| NODE NUMBER | DOF NUMBER | KINEMATIC TYPE |
|----------------|---------------|-------------------|
|----------------|---------------|-------------------|

45

2

DISP

N O D A L C O O R D I N A T E D A T A

| NODE NO. | X1 | X2 |
|----------|----------------|----------------|
| 1 | 0.00000000E+00 | 0.00000000E+00 |
| 2 | 0.00000000E+00 | 5.00000000E-01 |
| 3 | 1.00000000E+00 | 1.00000000E+00 |
| 4 | 0.00000000E+00 | 1.50000000E+00 |
| 5 | 0.00000000E+00 | 2.00000000E+00 |
| 6 | 2.00000000E+00 | 0.00000000E+00 |
| 7 | 2.00000000E+00 | 5.00000000E-01 |
| 8 | 2.00000000E+00 | 1.00000000E+00 |
| 9 | 2.00000000E+00 | 1.50000000E+00 |
| 10 | 2.00000000E+00 | 2.00000000E+00 |
| 11 | 4.00000000E+00 | 0.00000000E+00 |
| 12 | 4.00000000E+00 | 5.00000000E-01 |
| 13 | 4.00000000E+00 | 1.00000000E+00 |
| 14 | 4.00000000E+00 | 1.50000000E+00 |
| 15 | 4.00000000E+00 | 2.00000000E+00 |
| 16 | 6.00000000E+00 | 0.00000000E+00 |
| 17 | 6.00000000E+00 | 5.00000000E-01 |
| 18 | 6.00000000E+00 | 1.00000000E+00 |
| 19 | 6.00000000E+00 | 1.50000000E+00 |
| 20 | 6.00000000E+00 | 2.00000000E+00 |
| 21 | 8.00000000E+00 | 0.00000000E+00 |
| 22 | 8.00000000E+00 | 5.00000000E-01 |
| 23 | 8.00000000E+00 | 1.00000000E+00 |
| 24 | 8.00000000E+00 | 1.50000000E+00 |
| 25 | 8.00000000E+00 | 2.00000000E+00 |
| 26 | 1.00000000E+01 | 0.00000000E+00 |
| 27 | 1.00000000E+01 | 5.00000000E-01 |
| 28 | 1.00000000E+01 | 1.00000000E+00 |
| 29 | 1.00000000E+01 | 1.50000000E+00 |
| 30 | 1.00000000E+01 | 2.00000000E+00 |
| 31 | 1.20000000E+01 | 0.00000000E+00 |
| 32 | 1.20000000E+01 | 5.00000000E-01 |
| 33 | 1.20000000E+01 | 1.00000000E+00 |
| 34 | 1.20000000E+01 | 1.50000000E+00 |
| 35 | 1.20000000E+01 | 2.00000000E+00 |
| 36 | 1.40000000E+01 | 0.00000000E+00 |
| 37 | 1.40000000E+01 | 5.00000000E-01 |
| 38 | 1.40000000E+01 | 1.00000000E+00 |
| 39 | 1.40000000E+01 | 1.50000000E+00 |
| 40 | 1.40000000E+01 | 2.00000000E+00 |
| 41 | 1.60000000E+01 | 0.00000000E+00 |
| 42 | 1.60000000E+01 | 5.00000000E-01 |
| 43 | 1.60000000E+01 | 1.00000000E+00 |
| 44 | 1.60000000E+01 | 1.50000000E+00 |
| 45 | 1.60000000E+01 | 2.00000000E+00 |

N O D A L B O U N D A R Y C O N D I T I O N C O D E S

| NODE NO. | DOF1 | DOF2 |
|----------|------|------|
| 1 | 1 | 0 |
| 5 | 1 | 0 |
| 6 | 1 | 0 |
| 11 | 1 | 0 |
| 16 | 1 | 0 |
| 21 | 1 | 0 |
| 26 | 1 | 0 |
| 31 | 1 | 0 |
| 36 | 1 | 0 |
| 41 | 1 | 0 |

P R E S C R I B E D F O R C E S A N D K I N E M A T I C
B O U N D A R Y C O N D I T I O N S

LOAD VECTOR NUMBER = 1

| NODE NO. | DOF1 | DOF2 |
|----------|-----------------|-----------------|
| 2 | -7.50000000E-01 | 1.73830000E-01 |
| 3 | -1.50000000E+00 | 1.38670000E-01 |
| 4 | -2.25000000E+00 | 8.00800000E-02 |
| 5 | 0.00000000E+00 | 1.46500000E-02 |
| 41 | 0.00000000E+00 | -9.28000000E-02 |
| 42 | 0.00000000E+00 | -1.73830000E-01 |
| 43 | 0.00000000E+00 | -1.38670000E-01 |
| 44 | 0.00000000E+00 | -8.00800000E-02 |
| 45 | 0.00000000E+00 | -1.46500000E-02 |

L O A D - T I M E F U N C T I O N D A T A

NUMBER OF LOAD-TIME FUNTIONS (NLTFN) = 1

FUNCTION NUMBER 1

| TIME | LOAD FACTOR |
|----------------|----------------|
| 0.00000000E+00 | 1.00000000E+03 |

THERE ARE NO NONZERO INITIAL DISPLACEMENTS

THERE ARE NO NONZERO INITIAL VELOCITIES

THERE ARE NO NONZERO INITIAL ACCELERATIONS

ELEMENT GROUP DATA

ELEMENT GROUP NUMBER (NEG) = 1

FOUR - NOD E QUADRILATERAL ELEMENTS

ELEMENT TYPE NUMBER (NTYPE) = 1

NUMBER OF ELEMENTS (NUMEL) = 32

NUMBER OF ELEMENT MATERIAL SETS (NUMAT) = 1

NUMBER OF SURFACE FORCE CARDS (NSURF) = 0

NUMBER OF STRESS/STRAIN TIME HISTORIES . . . (NSOUT) = 1

ANALYSIS OPTION (IOPT) = 1

EQ.0, PLANE STRESS

EQ.1, PLANE STRAIN

EQ.2, AXISYMMETRIC

STRESS OUTPUT PRINT CODE (ISTPRT) = 0

EQ.0, STRESS OUTPUT PRINTED

EQ.1, STRESS OUTPUT NOT PRINTED

SURFACE FORCE LOAD-TIME FUNCTION NUMBER . . (LFSURF) = 0

BODY FORCE LOAD-TIME FUNCTION NUMBER . . (LFBODY) = 0

NUMERICAL INTEGRATION CODE (NICODE) = 0

EQ.0, 2 X 2 GAUSSIAN QUADRATURE

EQ.1, 1-POINT GAUSSIAN QUADRATURE

STRAIN-DISPLACEMENT OPTION (IBBAR) = 0

EQ.0, STANDARD FORMULATION

EQ.1, B-BAR FORMULATION

MASS TYPE CODE (IMASS) = 0

EQ.0, CONSISTENT MASS MATRIX

EQ.1, LUMPED MASS MATRIX

EQ.2, NO MASS MATRIX

IMPLICIT/EXPLICIT CODE (IMPEXP) = 0

EQ.0, IMPLICIT ELEMENT GROUP

EQ.1, EXPLICIT ELEMENT GROUP

MATERIAL SET DATA

NUMBER OF MATERIAL SETS (NUMAT) = 1

| SET NUMBER | YOUNG'S MODULUS | POISSON'S RATIO | MASS DENSITY | MASS DAMPING | STIFFNESS DAMPING | THICKNESS |
|------------|-----------------|-----------------|--------------|--------------|-------------------|------------|
| 1 | 2.0800E+06 | 3.0000E-01 | 2.0000E-03 | 0.0000E+00 | 0.0000E+00 | 1.0000E+00 |

GRAVITY VECTOR COMPONENTS

X-1 DIRECTION = 0.00000000E+00

X-2 DIRECTION = 0.00000000E+00

ELEMENT DATA

| ELEMENT | MATERIAL | NODE 1 | NODE 2 | NODE 3 | NODE 4 |
|---------|----------|--------|--------|--------|--------|
| 1 | 1 | 2 | 6 | 7 | 2 |
| 2 | 1 | 2 | 7 | 8 | 3 |
| 3 | 1 | 3 | 8 | 9 | 4 |
| 4 | 1 | 4 | 9 | 10 | 5 |
| 5 | 1 | 5 | 11 | 12 | 6 |
| 6 | 1 | 6 | 12 | 13 | 7 |
| 7 | 1 | 7 | 13 | 14 | 8 |
| 8 | 1 | 8 | 14 | 15 | 9 |
| 9 | 1 | 9 | 16 | 17 | 10 |
| 10 | 1 | 11 | 17 | 18 | 12 |
| 11 | 1 | 12 | 18 | 19 | 13 |
| 12 | 1 | 13 | 19 | 20 | 14 |
| 13 | 1 | 14 | 21 | 22 | 15 |
| 14 | 1 | 16 | 22 | 23 | 17 |
| 15 | 1 | 17 | 23 | 24 | 18 |
| 16 | 1 | 18 | 24 | 25 | 19 |
| 17 | 1 | 19 | 25 | 26 | 20 |
| 18 | 1 | 21 | 26 | 27 | 22 |
| 19 | 1 | 22 | 27 | 28 | 23 |
| 20 | 1 | 23 | 28 | 29 | 24 |
| 21 | 1 | 24 | 29 | 30 | 25 |
| 22 | 1 | 26 | 31 | 32 | 27 |
| 23 | 1 | 27 | 32 | 33 | 28 |
| 24 | 1 | 28 | 33 | 34 | 29 |
| 25 | 1 | 29 | 34 | 35 | 30 |
| 26 | 1 | 31 | 36 | 37 | 32 |
| 27 | 1 | 32 | 37 | 38 | 33 |
| 28 | 1 | 33 | 38 | 39 | 34 |
| 29 | 1 | 34 | 39 | 40 | 35 |
| 30 | 1 | 36 | 41 | 42 | 37 |
| 31 | 1 | 37 | 42 | 43 | 38 |
| 32 | 1 | 38 | 43 | 44 | 39 |
| | | 39 | 44 | 45 | 40 |

ELEMENT TIME HISTORY INFORMATION

NUMBER OF STRESS/STRAIN TIME HISTORIES . . (NSOUT) = 1

| ELEMENT NUMBER | INT PT NUMBER | COMPONENT |
|----------------|---------------|-----------|
| 20 | 4 | S 11 |

PLANE STRAIN CANTILEVER BEAM: DYNAMIC ANALYSIS WITH ALPHA METHOD

EQUATION SYSTEM DATA

NUMBER OF EQUATIONS (NEQ) = 79
 NUMBER OF TERMS IN LEFT-HAND-SIDE MATRIX . (NALHS) = 872
 MEAN HALF BANDWIDTH (MEANBW) = 11
 TOTAL LENGTH OF BLANK COMMON REQUIRED . . (NWORDS) = 5196

PLANE STRAIN CANTILEVER BEAM: DYNAMIC ANALYSIS WITH ALPHA METHOD

NODE NUMBER = 45

DOF NUMBER = 2 OUTPUT: DISP

| TIME | VALUE | -2.1102E-01 | 0.0000E+00 |
|------------|-------------|-------------|------------|
| 0.0000E+00 | 0.0000E+00 | | * |
| 5.0000E-04 | -5.4078E-03 | | * |
| 1.0000E-03 | -1.8582E-02 | | * |
| 1.5000E-03 | -3.3184E-02 | | * |
| 2.0000E-03 | -4.6195E-02 | | * |
| 2.5000E-03 | -6.2725E-02 | | * |
| 3.0000E-03 | -8.8417E-02 | | * |
| 3.5000E-03 | -1.1853E-01 | | * |
| 4.0000E-03 | -1.4416E-01 | * | * |
| 4.5000E-03 | -1.6146E-01 | * | * |
| 5.0000E-03 | -1.7541E-01 | * | * |
| 5.5000E-03 | -1.9146E-01 | * | * |
| 6.0000E-03 | -2.0595E-01 | * | * |
| 6.5000E-03 | -2.1102E-01 | * | * |
| 7.0000E-03 | -2.0382E-01 | * | * |
| 7.5000E-03 | -1.9046E-01 | * | * |
| 8.0000E-03 | -1.7812E-01 | * | * |
| 8.5000E-03 | -1.6520E-01 | * | * |
| 9.0000E-03 | -1.4554E-01 | * | * |
| 9.5000E-03 | -1.1748E-01 | * | * |
| 1.0000E-02 | -8.8231E-02 | * | * |
| 1.0500E-02 | -6.5911E-02 | * | * |
| 1.1000E-02 | -4.9822E-02 | * | * |
| 1.1500E-02 | -3.4154E-02 | * | * |
| 1.2000E-02 | -1.6705E-02 | * | * |
| 1.2500E-02 | -3.9318E-03 | * | * |
| 1.3000E-02 | -2.8680E-03 | * | * |
| 1.3500E-02 | -1.1958E-02 | * | * |
| 1.4000E-02 | -2.3960E-02 | * | * |
| 1.4500E-02 | -3.4801E-02 | * | * |
| 1.5000E-02 | -4.9081E-02 | * | * |
| 1.5500E-02 | -7.2129E-02 | * | * |
| 1.6000E-02 | -1.0136E-01 | * | * |
| 1.6500E-02 | -1.2838E-01 | * | * |
| 1.7000E-02 | -1.4806E-01 | * | * |
| 1.7500E-02 | -1.6424E-01 | * | * |
| 1.8000E-02 | -1.8210E-01 | * | * |
| 1.8500E-02 | -1.9972E-01 | * | * |
| 1.9000E-02 | -2.0945E-01 | * | * |
| 1.9500E-02 | -2.0710E-01 | * | * |
| 2.0000E-02 | -1.9753E-01 | * | * |
| 2.0500E-02 | -1.8743E-01 | * | * |
| 2.1000E-02 | -1.7690E-01 | * | * |
| 2.1500E-02 | -1.6005E-01 | * | * |
| 2.2000E-02 | -1.3415E-01 | * | * |
| 2.2500E-02 | -1.0517E-01 | * | * |
| 2.3000E-02 | -8.0990E-02 | * | * |
| 2.3500E-02 | -6.2825E-02 | * | * |
| 2.4000E-02 | -4.5339E-02 | * | * |
| 2.4500E-02 | -2.5752E-02 | * | * |
| 2.5000E-02 | -9.3527E-03 | * | * |

PLANE STRAIN CANTILEVER BEAM: DYNAMIC ANALYSIS WITH ALPHA METHOD

ELEMENT NUMBER = 20

INTEGRATION POINT NUMBER = 4 OUTPUT: S 11

| TIME | VALUE | 0.0000E+000 | 4.4685E+03 |
|------------|-------------|-------------|------------|
| 0.0000E+00 | 0.0000E+000 | * | |
| 5.0000E-04 | 4.683E+02 | * | |
| 1.0000E-03 | 1.7634E+03 | * | |
| 1.5000E-03 | 2.1571E+03 | * | * |
| 2.0000E-03 | 2.5223E+03 | * | * |
| 2.5000E-03 | 3.1955E+02 | * | * |
| 3.0000E-03 | 1.5578E+03 | * | * |
| 3.5000E-03 | 2.9402E+03 | * | * |
| 4.0000E-03 | 3.6048E+03 | * | * |
| 4.5000E-03 | 3.0491E+03 | * | * |
| 5.0000E-03 | 2.2760E+03 | * | * |
| 5.5000E-03 | 2.8644E+03 | * | * |
| 6.0000E-03 | 4.0393E+03 | * | * |
| 6.5000E-03 | 4.4685E+03 | * | * |
| 7.0000E-03 | 3.5778E+03 | * | * |
| 7.5000E-03 | 2.5135E+03 | * | * |
| 8.0000E-03 | 2.6804E+03 | * | * |
| 8.5000E-03 | 3.4800E+03 | * | * |
| 9.0000E-03 | 3.6342E+03 | * | * |
| 9.5000E-03 | 2.4908E+03 | * | * |
| 1.0000E-02 | 1.2421E+03 | * | * |
| 1.0500E-02 | 1.2176E+03 | * | * |
| 1.1000E-02 | 1.9710E+03 | * | * |
| 1.1500E-02 | 2.2058E+03 | * | * |
| 1.2000E-02 | 1.2270E+03 | * | * |
| 1.2500E-02 | 1.9409E+02 | * | * |
| 1.3000E-02 | 3.8807E+02 | * | * |
| 1.3500E-02 | 1.4569E+03 | * | * |
| 1.4000E-02 | 2.0683E+03 | * | * |
| 1.4500E-02 | 1.4862E+03 | * | * |
| 1.5000E-02 | 8.0191E+02 | * | * |
| 1.5500E-02 | 1.2554E+03 | * | * |
| 1.6000E-02 | 2.5711E+03 | * | * |
| 1.6500E-02 | 3.3970E+03 | * | * |
| 1.7000E-02 | 2.9637E+03 | * | * |
| 1.7500E-02 | 2.3049E+03 | * | * |
| 1.8000E-02 | 2.6465E+03 | * | * |
| 1.8500E-02 | 3.8000E+03 | * | * |
| 1.9000E-02 | 4.4250E+03 | * | * |
| 1.9500E-02 | 3.7470E+03 | * | * |
| 2.0000E-02 | 2.7592E+03 | * | * |
| 2.0500E-02 | 2.6999E+03 | * | * |
| 2.1000E-02 | 3.4782E+03 | * | * |
| 2.1500E-02 | 3.7809E+03 | * | * |
| 2.2000E-02 | 2.8405E+03 | * | * |
| 2.2500E-02 | 1.6044E+03 | * | * |
| 2.3000E-02 | 1.3247E+03 | * | * |
| 2.3500E-02 | 1.9881E+03 | * | * |
| 2.4000E-02 | 2.2995E+03 | * | * |
| 2.4500E-02 | 1.4800E+03 | * | * |
| 2.5000E-02 | 4.0438E+02 | * | * |

| I C S T O R A G E | A L L O C A T I O N | I N F O R M A T I O N | | | | |
|-------------------|----------------------------|-----------------------|------|------|------|-------|
| ARRAY NO. | ARRAY | ADDRESS | DIM1 | DIM2 | DIM3 | PREC. |
| 1 | NSTEP | 1 | 1 | 0 | 0 | 1 |
| 2 | NDPT | 3 | 1 | 0 | 0 | 1 |
| 3 | NSPT | 5 | 1 | 0 | 0 | 1 |
| 4 | NHPLT | 7 | 1 | 0 | 0 | 1 |
| 5 | NITER | 9 | 1 | 0 | 0 | 1 |
| 6 | ALPHA | 11 | 1 | 0 | 0 | 2 |
| 7 | BETA | 13 | 1 | 0 | 0 | 2 |
| 8 | GAMMA | 15 | 1 | 0 | 0 | 2 |
| 9 | DT | 17 | 1 | 0 | 0 | 2 |
| 10 | IDHIST | 19 | 3 | 1 | 0 | 1 |
| 11 | DDUT | 23 | 2 | 1 | 0 | 1 |
| 12 | VPRED | 125 | 2 | 45 | 0 | 2 |
| 13 | DPRED | 305 | 2 | 45 | 0 | 2 |
| 14 | A | 4855 | 2 | 45 | 0 | 2 |
| 15 | V | 6655 | 2 | 45 | 0 | 2 |
| 16 | D | 8455 | 2 | 45 | 0 | 2 |
| 17 | X | 1025 | 2 | 45 | 0 | 2 |
| 18 | ID | 1205 | 2 | 45 | 0 | 1 |
| 19 | F | 1295 | 2 | 45 | 1 | 2 |
| 20 | G | 1475 | 1 | 0 | 0 | 2 |
| 21 | G1 | 1479 | 1 | 0 | 0 | 2 |
| 22 | IDIAG | 1481 | 79 | 0 | 0 | 1 |
| 23 | NGRP | 1561 | 1 | 0 | 0 | 1 |
| ***** | BEGIN ELEMENT GROUP NUMBER | 1 | | | | |
| 24 | NPAR | 1563 | 16 | 0 | 0 | 1 |
| 25 | MP | 1579 | 35 | 0 | 0 | 1 |
| 26 | W | 1615 | 4 | 0 | 0 | 2 |
| 27 | DET | 1623 | 4 | 0 | 0 | 2 |
| 28 | R | 1631 | 4 | 0 | 0 | 2 |
| 29 | SHL | 1639 | 3 | 4 | 4 | 4 |
| 30 | SHG | 1735 | 3 | 4 | 4 | 4 |
| 31 | SHGBAR | 1831 | 3 | 1 | 0 | 2 |
| 32 | RHO | 1855 | 1 | 0 | 0 | 2 |
| 33 | RDAMP | 1857 | 1 | 0 | 0 | 2 |
| 34 | RDAMPK | 1859 | 1 | 0 | 0 | 2 |
| 35 | TH | 1861 | 1 | 0 | 0 | 2 |
| 36 | C | 1863 | 1 | 0 | 0 | 2 |
| 37 | GRAV | 1895 | 4 | 0 | 0 | 2 |
| 38 | IEN | 1899 | 2 | 0 | 0 | 2 |
| 39 | MAT | 2027 | 32 | 32 | 0 | 1 |
| 40 | LM | 2059 | 2 | 4 | 0 | 2 |
| 41 | IELNO | 2315 | 0 | 0 | 0 | 1 |
| 42 | ISIDE | 2315 | 0 | 0 | 0 | 1 |
| 43 | PRESS | 2315 | 2 | 0 | 0 | 2 |
| 44 | SHEAR | 2319 | 2 | 0 | 0 | 2 |
| 45 | ISHIST | 2323 | 3 | 1 | 0 | 2 |
| 46 | SOUT | 2327 | 2 | 0 | 0 | 2 |
| 47 | ELEFFM | 2429 | 8 | 8 | 0 | 2 |
| 48 | XL | 2557 | 2 | 4 | 0 | 2 |
| 49 | WORK | 2573 | 16 | 0 | 0 | 2 |
| 50 | B | 2605 | 4 | 8 | 0 | 2 |

| DYNAMIC STORAGE | | ALLOCATION | INFORMATION | | | |
|------------------------------|--------|------------|-------------|------|------|-------|
| ARRAY NO. | ARRAY | ADDRESS | DIM1 | DIM2 | DIM3 | PREC. |
| 51 | DMAT | 2669 | 4 | 4 | 0 | 2 |
| 52 | DB | 2701 | 4 | 8 | 0 | 2 |
| 53 | VL | 2765 | 2 | 4 | 0 | 2 |
| 54 | AL | 2781 | 2 | 4 | 0 | 2 |
| 55 | ELRESF | 2797 | 8 | 0 | 0 | 2 |
| 56 | DL | 2813 | 2 | 4 | 0 | 2 |
| 57 | STRAIN | 2829 | 4 | 0 | 0 | 2 |
| 58 | STRESS | 2837 | 4 | 0 | 0 | 2 |
| 59 | PSTRN | 2845 | 4 | 0 | 0 | 2 |
| 60 | PSTRS | 2853 | 4 | 0 | 0 | 2 |
| ***** END ELEMENT GROUP DATA | | | | | | |
| 61 | ALHS | 2861 | 872 | 0 | 0 | 2 |
| 62 | BRHS | 4605 | 79 | 0 | 0 | 2 |

PLANE STRAIN CANTILEVER BEAM: DYNAMIC ANALYSIS WITH ALPHA METHOD

EXECUTION TIMING INFORMATION

INITIALIZATION PHASE = 1.340E+00

SOLUTION PHASE = 9.855E+01

FORMATION OF LEFT-HAND-SIDE MATRICES = 1.891E+00

FACTORIZATIONS = 1.289E-01

FORMATION OF RIGHT-HAND-SIDE VECTORS = 8.767E+01

FORWARD REDUCTIONS/BACK SUBSTITUTIONS = 3.773E+00

CALCULATION OF ELEMENT OUTPUT = 1.006E+00

SUBTOTAL = 9.447E+01

Example 4 (Implicit-Explicit Dynamic Analysis of a Rod)

The initial-value problem consists of a rod, modeled with two-node truss elements, fixed at the left end, and subjected to an initial displacement. The initial velocity and external forces are assumed equal to 0. The length of the rod is 10.5 and there are 21 equal-length elements. Young's modulus equals 100 in all elements except the last, in which it equals 10⁷. The initial displacement varies linearly between 0 on the left end of the rod (i.e., $x = 0$) and 0.1 at $x = 10.0$ and is constant over the last element. The cross-sectional area of the rod is 1.0 and its density is 0.01. The implicit-explicit algorithm described in Sec. 9.4 is employed. The algorithmic parameters used are

$$\alpha = 0.0, \quad \beta = 0.3025, \quad \gamma = 0.60$$

The critical time step of explicit integration may be computed from (9.4.29), namely,

$$\omega^h \Delta t \leq \Omega_{\text{bif}} = \frac{2(1 - \xi)}{\gamma + \frac{1}{2}}$$

There is no viscous damping, so $\xi = 0$. Assuming lumped mass, the maximum element frequency is (see (9.2.1)):

$$\omega_{\max}^h = \frac{2c}{h} = \frac{2\sqrt{E/\rho}}{h}$$

Evaluating these formulas reveals that the stable critical time step for the last element is $\frac{1}{316}$ that of the first 20 elements. This suggests a mesh partition in which the last element is treated implicitly and the first 20 elements are treated explicitly, thus enabling the

larger critical time step to be employed. Specifically, $\Delta t = 4.5 \times 10^{-3}$ was used, and the analysis was run for 100 steps. Output histories are requested for displacement at the end of the rod (i.e., node 22) and stress in element 10. The initial displacements require that the initial accelerations be computed. This is facilitated by a one-step, “zero Δt ” time sequence as described in Sec. In.3.0. The data for the analysis is presented in Table 11.5.4 and the output follows on pages 717 through 729.

Exercise 1. The addition of Rayleigh damping can significantly reduce critical time steps in explicit integration. This can be seen from (9.4.28):

$$\Omega_{\text{crit}} = \frac{(\xi^2 + 2\gamma)^{1/2} - \xi}{\gamma}$$

In the physically undamped case,

$$\Omega_{\text{crit}} = \sqrt{\frac{2}{\gamma}}$$

Estimate the fundamental mode of the rod from

$$\omega_1 = \frac{c\pi}{2L}$$

Select the stiffness-proportional damping factor RDAMPK such that the damping ratio in the fundamental mode, namely, ξ_1 , is 10%. Determine the maximum damping factor from

$$\xi_{\max} = \frac{1}{2} (\text{RDAMPM}/\omega_{\max}^h + \text{RDAMPK} * \omega_{\max}^h)$$

Consider the setup of Example 4. Due to *implicit* treatment of the single stiff element, it may be ignored in calculating ω_{\max}^h , i.e., employ $E = 100.0$ and $h = 0.5$. Estimate ω_1 using $E = 100.0$ and $L = 10.0$. Assuming $\gamma = \frac{1}{2}$, calculate the reduction in critical time step compared with the physically undamped case, namely,

$$\frac{[(\xi_{\max}^2 + 2\gamma)^{1/2} - \xi_{\max}]/\gamma}{\sqrt{2}/\gamma}$$

[Ans. 0.189]

Repeat the calculation assuming the rod is discretized by 210 elements of length 0.05.

TABLE 11.5.4 Input File for Implicit-Explicit Dynamic Analysis of a Rod

IMPLICIT-EXPLICIT DYNAMIC ANALYSIS OF A ROD

```

E X E C U T I O N   C O N T R O L   I N F O R M A T I O N
EXECUTION CODE . . . . . (IEXEC) = 1
  EQ. 0, DATA CHECK
  EQ. 1, EXECUTION

ANALYSIS CODE . . . . . (IACODE) = 0
  EQ. 0, DYNAMIC ANALYSIS
  EQ. 1, STATIC ANALYSIS

READ RESTART FILE CODE . . . . . (IREADR) = 0
  EQ. 0, DO NOT READ RESTART FILE
  EQ. 1, READ RESTART FILE

WRITE RESTART FILE CODE . . . . . (IWRITR) = 0
  EQ. 0, DO NOT WRITE RESTART FILE
  EQ. 1, WRITE RESTART FILE

INPUT DATA PRINT CODE . . . . . (IPRTIN) = 0
  EQ. 0, PRINT NODAL AND ELEMENT INPUT DATA
  EQ. 1, DO NOT PRINT NODAL AND ELEMENT INPUT DATA

RANK CHECK CODE . . . . . (IRANK) = 0
  EQ. 0, DO NOT PERFORM RANK CHECK
  EQ. 1, PRINT NUMBERS OF ZERO AND NEGATIVE PIVOTS
  EQ. 2, PRINT ALL PIVOTS

NUMBER OF TIME SEQUENCES . . . . . (NUMSEQ) = 2
NUMBER OF NODAL OUTPUT TIME-HISTORIES . . (NDOUT) = 1
NUMBER OF SPACE DIMENSIONS . . . . . (NSD) = 3
NUMBER OF NODAL POINTS . . . . . (NUMNP) = 22
NUMBER OF NODAL DEGREES-OF-FREEDOM . . . (NDOF) = 3
NUMBER OF LOAD VECTORS . . . . . (NLVECT) = 0
NUMBER OF LOAD-TIME FUNCTIONS . . . . . (NLTFTN) = 0
NUMBER OF POINTS ON LOAD-TIME FUNCTIONS . (NPTSLF) = 0
NUMBER OF ELEMENT GROUPS . . . . . (NUMEG) = 2

```

TIME SEQUENCE DATA

```

NUMBER OF TIME SEQUENCES . . . . . (NUMSEQ ) =      2

TIME SEQUENCE NUMBER . . . . . (N      ) =      1
NUMBER OF TIME STEPS . . . . . (NSTEP(N)) =      1
INEMATIC PRINT INCREMENT . . . . . (NDPRT(N)) =      1
TRESS/STRAIN PRINT INCREMENT . . . . (NSPRT(N)) =      1
TIME HISTORY PLOT INCREMENT . . . . (NHPLT(N)) =      1
NUMBER OF ITERATIONS . . . . . (NITER(N)) =      1
IRST INTEGRATION PARAMETER . . . . (ALPHA(N)) =  0.00000E+00
ECOND INTEGRATION PARAMETER . . . . (BETA(N) ) =  0.00000E+00
HIRD INTEGRATION PARAMETER . . . . (GAMMA(N)) =  0.00000E+00
IME STEP . . . . . . . . . . (DT(N) ) =  0.00000E+00

```

```

IME SEQUENCE NUMBER . . . . . (N      ) =      2
NUMBER OF TIME STEPS . . . . . (NSTEP(N)) =    100
INEMATIC PRINT INCREMENT . . . . . (NDPRT(N)) =      0
TRESS/STRAIN PRINT INCREMENT . . . . (NSPRT(N)) =      0
IME HISTORY PLOT INCREMENT . . . . (NHPLT(N)) =      2
NUMBER OF ITERATIONS . . . . . (NITER(N)) =      1
IRST INTEGRATION PARAMETER . . . . (ALPHA(N)) =  0.00000E+00
ECOND INTEGRATION PARAMETER . . . . (BETA(N) ) =  3.02500E-01
HIRD INTEGRATION PARAMETER . . . . (GAMMA(N)) =  6.00000E-01
IME STEP . . . . . . . . . . (DT(N) ) =  4.50000E-03

```

ALL TIME-HISTORY INFORMATION

```
NUMBER OF NODAL TIME HISTORIES . . . . . (NDOUT ) =      1
```

| NODE NUMBER | DOF NUMBER | KINEMATIC TYPE |
|----------------|---------------|-------------------|
| 22 | 1 | DISP |

Sec. 11.5 DLEARN User's Manual

N O D A L C O O R D I N A T E D A T A

| NODE NO. | X1 | X2 | X3 |
|----------|-----------------|-----------------|-----------------|
| 1 | 0.000000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 2 | 5.000000000E-01 | 0.000000000E+00 | 0.000000000E+00 |
| 3 | 1.000000000E+00 | 0.000000300E+00 | 0.000000000E+00 |
| 4 | 1.500000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 5 | 2.000000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 6 | 2.500000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 7 | 3.000000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 8 | 3.500000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 9 | 4.000000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 10 | 4.500000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 11 | 5.000000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 12 | 5.500000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 13 | 6.000000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 14 | 6.500000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 15 | 7.000000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 16 | 7.500000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 17 | 8.000000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 18 | 8.500000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 19 | 9.000000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 20 | 9.500000000E+00 | 0.000000000E+00 | 0.000000000E+00 |
| 21 | 1.000000000E+01 | 0.000000000E+00 | 0.000000000E+00 |
| 22 | 1.050000000E+01 | 0.000000000E+00 | 0.000000000E+00 |

N O D A L B O U N D A R Y C O N D I T I O N C O D E S

| NODE NO. | DOF1 | DOF2 | DOF3 |
|----------|------|------|------|
| 1 | 1 | 1 | 1 |
| 2 | 0 | 1 | 1 |
| 3 | 0 | 1 | 1 |
| 4 | 0 | 1 | 1 |
| 5 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 |
| 7 | 0 | 1 | 1 |
| 8 | 0 | 1 | 1 |
| 9 | 0 | 1 | 1 |
| 10 | 0 | 1 | 1 |
| 11 | 0 | 1 | 1 |
| 12 | 0 | 1 | 1 |
| 13 | 0 | 1 | 1 |
| 14 | 0 | 1 | 1 |
| 15 | 0 | 1 | 1 |
| 16 | 0 | 1 | 1 |
| 17 | 0 | 1 | 1 |
| 18 | 0 | 1 | 1 |
| 19 | 0 | 1 | 1 |
| 20 | 0 | 1 | 1 |
| 21 | 0 | 1 | 1 |
| 22 | 0 | 1 | 1 |

INITIAL DISPLACEMENTS

STEP NUMBER = 0
 TIME = 0.000E+00

| NODE NO. | Ddf1 | Ddf2 | Ddf3 |
|----------|---------------|---------------|---------------|
| 2 | 5.0000000E-03 | 0.0000000E+00 | 0.0000000E+00 |
| 3 | 1.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 4 | 1.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 5 | 2.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 6 | 2.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 7 | 3.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 8 | 3.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 9 | 4.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 10 | 4.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 11 | 5.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 12 | 5.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 13 | 6.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 14 | 6.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 15 | 7.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 16 | 7.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 17 | 8.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 18 | 8.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 19 | 9.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 20 | 9.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 21 | 1.0000000E-01 | 0.0000000E+00 | 0.0000000E+00 |
| 22 | 1.0000000E-01 | 0.0000000E+00 | 0.0000000E+00 |

THERE ARE NO NONZERO INITIAL VELOCITIES

THERE ARE NO NONZERO INITIAL ACCELERATIONS

ELEMENT GROUP DATA

ELEMENT GROUP NUMBER (NEG) = 1

TWO / THREE - NODE TRUSS ELEMENTS

ELEMENT TYPE NUMBER (NTYPE) = 2

NUMBER OF ELEMENTS (NUMEL) = 1

NUMBER OF ELEMENT MATERIAL SETS (NUMAT) = 1

NUMBER OF ELEMENT NODES (NEN) = 2

NUMBER OF STRESS/STRAIN TIME HISTORIES . . (NSOUT) = 0

Sec. 11.5 DLEARN User's Manual

ELEMENT GROUP DATA

ELEMENT GROUP NUMBER (NEG) = 2

TWO / THREE - NODE TRUSS ELEMENTS

ELEMENT TYPE NUMBER (NTYPE) = 2

NUMBER OF ELEMENTS (NUMEL) = 20

NUMBER OF ELEMENT MATERIAL SETS (NUMAT) = 1

NUMBER OF ELEMENT NODES (NEN) = 2

NUMBER OF STRESS/STRAIN TIME HISTORIES . . . (NSOUT) = 1

STRESS OUTPUT PRINT CODE (ISTPRT) = 0

EQ.0, STRESS OUTPUT PRINTED

EQ.1, STRESS OUTPUT NOT PRINTED

BODY FORCE LOAD-TIME FUNCTION NUMBER . . . (LFBODY) = 0

INTEGRATION CODE (NINT) = 1

EQ.1, 1-POINT GAUSSIAN QUADRATURE

EQ.2, 2-POINT GAUSSIAN QUADRATURE

EQ.3, 3-POINT GAUSSIAN QUADRATURE

MASS TYPE CODE (IMASS) = 1

EQ.0, CONSISTENT MASS MATRIX

EQ.1, LUMPED MASS MATRIX

EQ.2, NO MASS MATRIX

IMPLICIT/EXPLICIT CODE (IMPEXP) = 1

EQ.0, IMPLICIT ELEMENT GROUP

EQ.1, EXPLICIT ELEMENT GROUP

MATERIAL SET DATA

NUMBER OF MATERIAL SETS (NUMAT) = 1

| SET NUMBER | YOUNG'S MODULUS | MASS DENSITY | MASS DAMPING | STIFFNESS DAMPING | AREA |
|------------|-----------------|--------------|--------------|-------------------|------------|
| 1 | 1.0000E+02 | 1.0000E-02 | 0.0000E+00 | 0.0000E+00 | 1.0000E+00 |

GRAVITY VECTOR COMPONENTS
 X-1 DIRECTION = 0.00000000E+00
 X-2 DIRECTION = 0.00000000E+00
 X-3 DIRECTION = 0.00000000E+00

ELEMENT DATA

| ELEMENT | MATERIAL | NODE 1 | NODE 2 |
|---------|----------|--------|--------|
| 1 | 1 | 1 | 2 |
| 2 | 1 | 2 | 3 |
| 3 | 1 | 3 | 4 |
| 4 | 1 | 4 | 5 |
| 5 | 1 | 5 | 6 |
| 6 | 1 | 6 | 7 |
| 7 | 1 | 7 | 8 |
| 8 | 1 | 8 | 9 |
| 9 | 1 | 9 | 10 |
| 10 | 1 | 10 | 11 |
| 11 | 1 | 11 | 12 |
| 12 | 1 | 12 | 13 |
| 13 | 1 | 13 | 14 |
| 14 | 1 | 14 | 15 |
| 15 | 1 | 15 | 16 |
| 16 | 1 | 16 | 17 |
| 17 | 1 | 17 | 18 |
| 18 | 1 | 18 | 19 |
| 19 | 1 | 19 | 20 |
| 20 | 1 | 20 | 21 |

ELEMENT TIME HISTORY INFORMATION
 NUMBER OF STRESS/STRAIN TIME HISTORIES . . (NSOUT) = 1

| ELEMENT NUMBER | INT PT NUMBER | COMPONENT |
|-------------------|------------------|-----------|
| 10 | 1 | STRS |

IMPLICIT-EXPLICIT DYNAMIC ANALYSIS OF A ROD

EQUATION SYSTEM DATA

| | | |
|--|------------|----|
| NUMBER OF EQUATIONS | (NEQ) = | 21 |
| NUMBER OF TERMS IN LEFT-HAND-SIDE MATRIX . (NALHS) = | 22 | |
| MEAN HALF BANDWIDTH | (MEANBW) = | 1 |
| TOTAL LENGTH OF BLANK COMMON REQUIRED . . . (NWORDS) = | 2542 | |

D I S P L A C E M E N T S

STEP NUMBER = 1
 TIME = 0.000E+00

| NODE NO. | DOF1 | DOF2 | DOF3 |
|----------|---------------|---------------|---------------|
| 2 | 5.0000000E-03 | 0.0000000E+00 | 0.0000000E+00 |
| 3 | 1.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 4 | 1.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 5 | 2.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 6 | 2.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 7 | 3.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 8 | 3.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 9 | 4.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 10 | 4.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 11 | 5.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 12 | 5.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 13 | 6.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 14 | 6.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 15 | 7.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 16 | 7.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 17 | 8.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 18 | 8.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 19 | 9.0000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 20 | 9.5000000E-02 | 0.0000000E+00 | 0.0000000E+00 |
| 21 | 1.0000000E-01 | 0.0000000E+00 | 0.0000000E+00 |
| 22 | 1.0000000E-01 | 0.0000000E+00 | 0.0000000E+00 |

V E L O C I T I E S

STEP NUMBER = 1
 TIME = 0.000E+00

| NODE NO. | DOF1 | DOF2 | DOF3 |
|----------|------|------|------|
| | | | |

THERE ARE NO NONZERO COMPONENTS

A C C E L E R A T I O N S

STEP NUMBER = 1
 TIME = 0.000E+00

| NODE NO. | DOF1 | DOF2 | DOF3 |
|----------|-----------------|---------------|---------------|
| 3 | -5.55111512E-15 | 0.0000000E+00 | 0.0000000E+00 |
| 4 | 5.35111512E-15 | 0.0000000E+00 | 0.0000000E+00 |
| 5 | -2.22044605E-14 | 0.0000000E+00 | 0.0000000E+00 |
| 6 | -1.94289029E-14 | 0.0000000E+00 | 0.0000000E+00 |
| 7 | 3.60822483E-14 | 0.0000000E+00 | 0.0000000E+00 |
| 8 | -1.66533454E-14 | 0.0000000E+00 | 0.0000000E+00 |
| 9 | -3.60822483E-14 | 0.0000000E+00 | 0.0000000E+00 |
| 10 | 6.93889390E-14 | 0.0000000E+00 | 0.0000000E+00 |
| 11 | -1.02695630E-13 | 0.0000000E+00 | 0.0000000E+00 |
| 12 | 1.02695630E-13 | 0.0000000E+00 | 0.0000000E+00 |
| 13 | -7.21644966E-14 | 0.0000000E+00 | 0.0000000E+00 |
| 14 | 7.21644966E-14 | 0.0000000E+00 | 0.0000000E+00 |
| 16 | -6.93889390E-14 | 0.0000000E+00 | 0.0000000E+00 |
| 17 | -1.41553436E-13 | 0.0000000E+00 | 0.0000000E+00 |
| 18 | -1.41553436E-13 | 0.0000000E+00 | 0.0000000E+00 |
| 19 | 6.93889390E-14 | 0.0000000E+00 | 0.0000000E+00 |
| 21 | -2.00000000E+02 | 0.0000000E+00 | 0.0000000E+00 |
| 22 | 2.77555556E-08 | 0.0000000E+00 | 0.0000000E+00 |

ELEMENT STRESSES AND STRAINS

ELEMENT GROUP NUMBER (NEG) = 1

| ELEMENT NUMBER | INT. PT. NUMBER | X1 COORD. | X2 COORD. | X3 COORD. | STRESS | FORCE | STRAIN |
|----------------|-----------------|-----------|-----------|-----------|-----------|-----------|-----------|
| 1 | 1 | 1.02E+01 | 0.00E+00 | 0.00E+00 | -6.94E-11 | -6.94E-11 | -6.94E-18 |

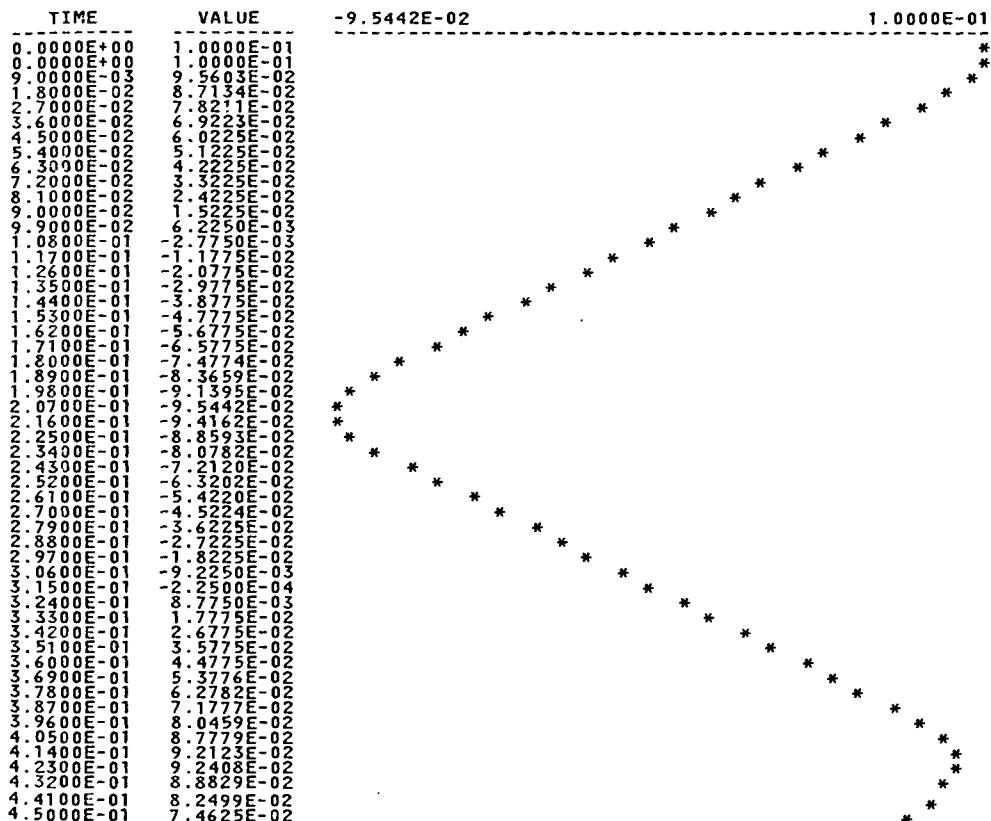
ELEMENT STRESSES AND STRAINS

ELEMENT GROUP NUMBER (NEG) = 2

| ELEMENT NUMBER | INT. PT. NUMBER | X1 COORD. | X2 COORD. | X3 COORD. | STRESS | FORCE | STRAIN |
|----------------|-----------------|-----------|-----------|-----------|----------|----------|----------|
| 1 | 1 | 2.50E-01 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 2 | 1 | 7.50E-01 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 3 | 1 | 1.25E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 4 | 1 | 1.75E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 5 | 1 | 2.25E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 6 | 1 | 2.75E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 7 | 1 | 3.25E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 8 | 1 | 3.75E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 9 | 1 | 4.25E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 10 | 1 | 4.75E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 11 | 1 | 5.25E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 12 | 1 | 5.75E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 13 | 1 | 6.25E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 14 | 1 | 6.75E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 15 | 1 | 7.25E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 16 | 1 | 7.75E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 17 | 1 | 8.25E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 18 | 1 | 8.75E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 19 | 1 | 9.25E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |
| 20 | 1 | 9.75E+00 | 0.00E+00 | 0.00E+00 | 1.00E+00 | 1.00E+00 | 1.00E-02 |

IMPLICIT-EXPLICIT DYNAMIC ANALYSIS OF A ROD

NODE NUMBER = 22
 DOF NUMBER = 1 OUTPUT: DISP



IMPLICIT-EXPLICIT DYNAMIC ANALYSIS OF A ROD

ELEMENT NUMBER = 10
 INTEGRATION POINT NUMBER = 1 OUTPUT: STRS

| TIME | VALUE | -1.0018E+00 | 1.0021E+00 |
|------------|-------------|-------------|------------|
| 0.0000E+00 | 1.0000E+00 | | * |
| 0.0000E+00 | 1.0000E+00 | | * |
| 9.0000E-03 | 1.0000E+00 | | * |
| 1.8000E-02 | 1.0000E+00 | | * |
| 2.7000E-02 | 1.0000E+00 | | * |
| 3.6000E-02 | 1.0000E+00 | | * |
| 4.5000E-02 | 9.8613E-01 | | * |
| 5.4000E-02 | 6.9567E-01 | | * |
| 6.3000E-02 | 2.1295E-01 | | * |
| 7.2000E-02 | 7.5574E-02 | * | * |
| 8.1000E-02 | 5.5897E-03 | * | * |
| 9.0000E-02 | 8.4809E-04 | * | * |
| 9.9000E-02 | 1.4168E-04 | * | * |
| 1.0800E-01 | 2.8074E-05 | * | * |
| 1.1700E-01 | 6.3156E-06 | * | * |
| 1.2600E-01 | 8.1778E-07 | * | * |
| 1.3500E-01 | -1.4638E-02 | * | * |
| 1.4400E-01 | -1.7536E-01 | * | * |
| 1.5300E-01 | -5.3637E-01 | * | * |
| 1.6200E-01 | -8.3211E-01 | * | * |
| 1.7100E-01 | -9.5730E-01 | * | * |
| 1.8000E-01 | -9.9151E-01 | * | * |
| 1.8900E-01 | -9.9855E-01 | * | * |
| 1.9800E-01 | -9.9977E-01 | * | * |
| 2.0700E-01 | -9.9996E-01 | * | * |
| 2.1600E-01 | -9.9999E-01 | * | * |
| 2.2500E-01 | -1.0000E+00 | * | * |
| 2.3400E-01 | -1.0018E+00 | * | * |
| 2.4300E-01 | -9.9973E-01 | * | * |
| 2.5200E-01 | -9.2160E-01 | * | * |
| 2.6100E-01 | -6.9174E-01 | * | * |
| 2.7000E-01 | -3.9669E-01 | * | * |
| 2.7900E-01 | -1.7502E-01 | * | * |
| 2.8800E-01 | -6.1613E-02 | * | * |
| 2.9700E-01 | -1.8053E-02 | * | * |
| 3.0600E-01 | -4.5810E-03 | * | * |
| 3.1500E-01 | -1.1041E-03 | * | * |
| 3.2400E-01 | -1.2314E-03 | * | * |
| 3.3300E-01 | 1.1990E-03 | * | * |
| 3.4200E-01 | 4.6505E-02 | * | * |
| 3.5100E-01 | 1.9879E-01 | * | * |
| 3.6000E-01 | 4.4648E-01 | * | * |
| 3.6900E-01 | 6.9268E-01 | * | * |
| 3.7800E-01 | 8.6165E-01 | * | * |
| 3.8700E-01 | 9.4836E-01 | * | * |
| 3.9600E-01 | 9.8359E-01 | * | * |
| 4.0500E-01 | 9.9545E-01 | * | * |
| 4.1400E-01 | 9.9886E-01 | * | * |
| 4.2300E-01 | 9.9975E-01 | * | * |
| 4.3200E-01 | 1.0009E+00 | * | * |
| 4.4100E-01 | 1.0021E+00 | * | * |
| 4.5000E-01 | 9.8302E-01 | * | * |

| I C | S T O R A G E | A L L O C A T I O N | I N F O R M A T I O N | | | |
|-----------|----------------------------|---------------------|-----------------------|------|------|-------|
| ARRAY NO. | ARRAY | ADDRESS | DIM1 | DIM2 | DIM3 | PREC. |
| 1 | NSTEP | 1 | 2 | 0 | 0 | 1 |
| 2 | NDPRT | 3 | 2 | 0 | 0 | 1 |
| 3 | NSPRT | 5 | 2 | 0 | 0 | 1 |
| 4 | NHPLT | 7 | 2 | 0 | 0 | 1 |
| 5 | NITER | 9 | 2 | 0 | 0 | 1 |
| 6 | ALPHA | 11 | 2 | 0 | 0 | 2 |
| 7 | BETA | 15 | 2 | 0 | 0 | 2 |
| 8 | GAMMA | 19 | 2 | 0 | 0 | 2 |
| 9 | DT | 23 | 2 | 0 | 0 | 2 |
| 10 | IDHIST | 27 | 2 | 0 | 0 | 1 |
| 11 | DOUT | 31 | 2 | 0 | 0 | 1 |
| 12 | VPRED | 135 | 2 | 0 | 0 | 2 |
| 13 | DPRED | 267 | 2 | 0 | 0 | 2 |
| 14 | A | 399 | 2 | 0 | 0 | 2 |
| 15 | V | 531 | 2 | 0 | 0 | 2 |
| 16 | D | 663 | 2 | 0 | 0 | 2 |
| 17 | X | 795 | 2 | 0 | 0 | 2 |
| 18 | ID | 927 | 2 | 0 | 0 | 1 |
| 19 | IDIAG | 993 | 21 | 0 | 0 | 1 |
| 20 | NGRP | 1015 | 2 | 0 | 0 | 1 |
| ***** | BEGIN ELEMENT GROUP NUMBER | | 1 | | | |
| 21 | NPAR | 1017 | 16 | 0 | 0 | 1 |
| 22 | MP | 1033 | 29 | 0 | 0 | 1 |
| 23 | W | 1063 | 1 | 0 | 0 | 2 |
| 24 | DET | 1065 | 1 | 0 | 0 | 2 |
| 25 | SHL | 1067 | 2 | 0 | 0 | 2 |
| 26 | SHG | 1075 | 2 | 0 | 0 | 2 |
| 27 | XS | 1083 | 3 | 0 | 0 | 2 |
| 28 | RHO | 1089 | 1 | 0 | 0 | 2 |
| 29 | RDAMPM | 1091 | 1 | 0 | 0 | 2 |
| 30 | RDAMPK | 1093 | 1 | 0 | 0 | 2 |
| 31 | AREA | 1095 | 1 | 0 | 0 | 2 |
| 32 | C | 1097 | 1 | 0 | 0 | 2 |
| 33 | GRAV | 1099 | 3 | 0 | 0 | 2 |
| 34 | IEN | 1105 | 2 | 0 | 0 | 1 |
| 35 | MAT | 1107 | 1 | 0 | 0 | 1 |
| 36 | LM | 1109 | 1 | 0 | 0 | 1 |
| 37 | ELEFFM | 1115 | 3 | 0 | 0 | 1 |
| 38 | XL | 1182 | 3 | 0 | 0 | 2 |
| 39 | WORK | 1199 | 16 | 0 | 0 | 2 |
| 40 | B | 1231 | 1 | 0 | 0 | 2 |
| 41 | DMAT | 1243 | 1 | 0 | 0 | 2 |
| 42 | DB | 1245 | 1 | 0 | 0 | 2 |
| 43 | VL | 1257 | 3 | 0 | 0 | 2 |
| 44 | AL | 1269 | 3 | 0 | 0 | 2 |
| 45 | ELRESF | 1281 | 6 | 0 | 0 | 2 |
| 46 | DL | 1293 | 3 | 0 | 0 | 2 |
| 47 | STRAIN | 1305 | 1 | 0 | 0 | 2 |
| 48 | STRESS | 1307 | 1 | 0 | 0 | 2 |
| 49 | FORCE | 1309 | 1 | 0 | 0 | 2 |
| ***** | BEGIN ELEMENT GROUP NUMBER | | 2 | | | |
| 50 | NPAR | 1311 | 16 | 0 | 0 | 1 |

| DYNAMIC STORAGE | | ALLOCATION | INFORMATION | | | |
|------------------------------|--------|------------|-------------|------|------|-------|
| ARRAY NO. | ARRAY | ADDRESS | DIM1 | DIM2 | DIM3 | PREC. |
| 51 | MP | 1327 | 29 | 0 | 0 | 1 |
| 52 | W | 1357 | 1 | 0 | 0 | 2 |
| 53 | DET | 1359 | 1 | 0 | 0 | 2 |
| 54 | SHL | 1361 | 2 | 2 | 1 | 2 |
| 55 | SHG | 1369 | 2 | 2 | 1 | 2 |
| 56 | XS | 1377 | 3 | 1 | 0 | 2 |
| 57 | RHO | 1383 | 1 | 0 | 0 | 2 |
| 58 | RDANPM | 1385 | 1 | 0 | 0 | 2 |
| 59 | RDAMPK | 1387 | 1 | 0 | 0 | 2 |
| 60 | AREA | 1389 | 1 | 0 | 0 | 2 |
| 61 | C | 1391 | 1 | 1 | 1 | 2 |
| 62 | GRAV | 1393 | 3 | 0 | 0 | 2 |
| 63 | IEN | 1399 | 2 | 2 | 0 | 1 |
| 64 | MAT | 1439 | 20 | 20 | 0 | 1 |
| 65 | LM | 1459 | 2 | 2 | 20 | 1 |
| 66 | ISHIST | 1579 | 3 | 1 | 20 | 1 |
| 67 | SOUT | 1583 | 2 | 2 | 52 | 1 |
| 68 | ELEFFM | 1687 | 6 | 6 | 52 | 1 |
| 69 | XL | 1759 | 3 | 2 | 0 | 2 |
| 70 | WORK | 1771 | 16 | 0 | 0 | 2 |
| 71 | B | 1803 | 1 | 6 | 0 | 2 |
| 72 | DMAT | 1815 | 1 | 1 | 0 | 2 |
| 73 | DB | 1817 | 1 | 6 | 0 | 2 |
| 74 | VL | 1829 | 3 | 2 | 0 | 2 |
| 75 | AL | 1841 | 3 | 2 | 0 | 2 |
| 76 | ELRESF | 1853 | 6 | 0 | 0 | 2 |
| 77 | DL | 1865 | 3 | 2 | 0 | 2 |
| 78 | STRAIN | 1877 | 1 | 0 | 0 | 2 |
| 79 | STRESS | 1879 | 1 | 0 | 0 | 2 |
| 80 | FORCE | 1881 | 1 | 0 | 0 | 2 |
| ***** END ELEMENT GROUP DATA | | | | | | |
| 81 | ALHS | 1883 | 22 | 0 | 0 | 2 |
| 82 | BRHS | 1927 | 21 | 0 | 0 | 2 |

IMPLICIT-EXPLICIT DYNAMIC ANALYSIS OF A ROD

| EXECUTION TIMING INFORMATION | |
|---------------------------------------|-------------|
| INITIALIZATION PHASE | = 1.711E+00 |
| SOLUTION PHASE | = 1.584E+01 |
| FORMATION OF LEFT-HAND-SIDE MATRICES | = 1.211E-01 |
| FACTORIZATIONS | = 0.000E+00 |
| FORMATION OF RIGHT-HAND-SIDE VECTORS | = 1.122E+01 |
| FORWARD REDUCTIONS/BACK SUBSTITUTIONS | = 1.973E-01 |
| CALCULATION OF ELEMENT OUTPUT | = 4.473E-01 |
| SUBTOTAL | = 1.199E+01 |

11.5.4 Subroutine Index for Program Listing

| Routine | Description | Page |
|-------------------|---|------|
| MAIN | Program to set storage capacity, precision and input/output units | 734 |
| SUBROUTINE DLEARN | Global driver program | 735 |
| SUBROUTINE DRIVER | Solution driver program | 737 |
| SUBROUTINE ADDLHS | Program to add element left-hand-side matrix to global left-hand-side matrix | 739 |
| SUBROUTINE ADDRHS | Program to add element residual-force vector to global right-hand-side vector | 740 |
| SUBROUTINE BACK | Program to perform forward reduction and back substitution | 740 |

11.5.4 Subroutine Index for Program Listing (cont.)

| Routine | Description | Page |
|------------------|---|------|
| JBROUTINE BC | Program to read, generate and write boundary condition data and establish equation numbers | 741 |
| JOCK DATA | Program to define output labels and numerical constants | 741 |
| JBROUTINE BTDB | Program to multiply $B^T DB$, taking account of symmetry, and accumulate into element stiffness matrix | 742 |
| JBROUTINE CLEAR | Program to clear a floating-point array | 742 |
| JNCTION COLDOT | Program to compute the dot product of vectors stored column-wise | 742 |
| JBROUTINE COLHT | Program to compute column heights in global left-hand-side matrix | 742 |
| JBROUTINE COMPBC | Program to compute displacement, velocity and acceleration boundary conditions | 743 |
| JBROUTINE CONTM | Program to form mass matrix for a continuum element with NEN nodes | 743 |
| JBROUTINE CONTMA | Program to calculate inertial and gravity/body force $(-m^e * (a^e - g^e))$ for a continuum element with NEN nodes | 745 |
| JBROUTINE COORD | Program to read, generate, and write coordinate data | 745 |
| JBROUTINE CORRCT | Program to perform corrector update of displacements and velocities | 746 |
| JBROUTINE DCTNRY | Program to store pointer information in dictionary | 746 |
| JBROUTINE DHIST | Program to read, write, and store nodal time history input | 746 |
| JBROUTINE DIAG | Program to compute diagonal addresses of left-hand-side matrix | 747 |
| JBROUTINE DYNPTS | Program to set memory pointers for dynamic analysis data arrays | 747 |
| JBROUTINE ECHO | Program to echo input data | 747 |
| JBROUTINE ELEMNT | Program to calculate element task number | 747 |
| JBROUTINE ELCARD | Program to read element group control card | 748 |
| JBROUTINE ELMLIB | Program to call element routines | 748 |
| JBROUTINE EQSET | Program to allocate storage for global equation system | 748 |
| JBROUTINE FACTOR | Program to perform Crout factorization $A = U^T DU$ | 749 |
| JBROUTINE FORMLM | Program to form LM array | 750 |
| JBROUTINE GENEL | Program to read and generate element node and material numbers | 750 |
| JBROUTINE GENEL1 | Program to generate element node and material numbers | 750 |
| UBROUTINE GENELD | Program to set defaults for element node and material number generation | 751 |
| UBROUTINE GENELI | Program to increment element node numbers | 751 |
| UBROUTINE GENFL | Program to read and generate floating-point nodal data | 751 |

11.5.4 Subroutine Index for Program Listing (cont.)

| Routine | Description | Page |
|-------------------|---|------|
| SUBROUTINE GENFL1 | Program to generate floating-point nodal data via isoparametric interpolation | 752 |
| SUBROUTINE GENSH | Program to call shape function routines for isoparametric generation | 753 |
| SUBROUTINE GENSH1 | Program to compute one-dimensional shape functions for isoparametric generation | 753 |
| SUBROUTINE GENSH2 | Program to compute two-dimensional shape functions for isoparametric generation | 753 |
| SUBROUTINE GENSH3 | Program to compute three-dimensional shape functions for isoparametric generation | 754 |
| SUBROUTINE HPLOT | Program to plot output histories | 754 |
| SUBROUTINE ICLEAR | Program to clear an integer array | 755 |
| SUBROUTINE IGEN | Program to read and generate integer nodal data | 755 |
| SUBROUTINE IMOVE | Program to move an integer array | 756 |
| SUBROUTINE INPUT | Program to read, generate and write nodal input data | 756 |
| SUBROUTINE INTERP | Program to perform linear interpolation | 757 |
| SUBROUTINE ITERUP | Program to perform intermediate update of displacements, velocities, and accelerations during iterative loop in predictor/corrector algorithm | 757 |
| SUBROUTINE LFAC | Program to compute load factors at time t | 758 |
| SUBROUTINE LOAD | Program to accumulate nodal forces and transfer into right-hand-side vector | 758 |
| SUBROUTINE LOCAL | Program to localize a global array | 758 |
| FUNCTION LOUT | Program to determine logical switch | 758 |
| SUBROUTINE LTIMEF | Program to read, write, and store load-time functions | 758 |
| SUBROUTINE MATADD | Program to add rectangular matrices | 759 |
| SUBROUTINE MEANSH | Program to calculate mean values of shape function global derivatives for mean-dilatational \bar{B} formulation | 760 |
| SUBROUTINE MINMAX | Program to compute the min and max in the row of a matrix | 760 |
| SUBROUTINE MOVE | Program to move a floating-point array | 760 |
| FUNCTION MPOINT | Program to calculate storage pointer | 760 |
| SUBROUTINE MULTAB | Program to multiply two matrices | 761 |
| SUBROUTINE PIVOTS | Program to determine the numbers of zero and negative terms in diagonal of array D of Crout factorization $A = U^T D U$ | 761 |
| SUBROUTINE PREDCT | Program to calculate predictor for displacements, velocities, and accelerations | 762 |
| SUBROUTINE PRINC | Program to compute principal values of symmetric second-rank tensor | 762 |
| SUBROUTINE PRINTD | Program to print kinematic data | 763 |
| SUBROUTINE PRINTF | Program to print prescribed force and boundary condition data | 763 |

11.5.4 Subroutine Index for Program Listing (cont.)

| Routine | Description | Page |
|-----------------|--|------|
| BROUTINE PRINTP | Program to print diagonal of array D after Crout factorization $A = U^T D U$ | 764 |
| BROUTINE PRNTEL | Program to print data for element with NEN nodes | 764 |
| BROUTINE PROP2D | Program to read, write and store properties for two-dimensional continuum elements | 764 |
| BROUTINE PRTDC | Program to print memory-pointer dictionary | 765 |
| BROUTINE PRTDC1 | Program to print memory-pointer information for an array | 765 |
| BROUTINE PRTS2D | Program to print stress, strain, and principal values for two-dimensional continuum elements | 766 |
| NCTION RCDOT | Program to compute the dot product of a vector stored row-wise with a vector stored column-wise | 766 |
| NCTION ROWDOT | Program to compute the dot product of vectors stored row-wise | 766 |
| BROUTINE RSIN | Program to read restart file | 767 |
| BROUTINE RSOUT | Program to write restart file | 767 |
| BROUTINE SERROR | Program to print error message if available storage is exceeded | 767 |
| BROUTINE SETUPD | Program to calculate the D matrix | 767 |
| BROUTINE SHIST | Program to read, write, and store element time-history input data | 768 |
| BROUTINE SMULT | Program to perform scalar multiplication of a matrix | 768 |
| BROUTINE STATIN | Program to set memory pointers for static analysis data arrays and call associated input routines | 768 |
| BROUTINE STORED | Program to store nodal time histories as single-precision data | 769 |
| BROUTINE TIMCON | Program to compute current time sequence parameters and time-integration coefficients | 770 |
| BROUTINE TIMING | Program to determine elapsed cpu time | 770 |
| BROUTINE TIMLOG | Program to print log of execution times | 770 |
| BROUTINE TSEQ | Program to set memory pointers for time sequence and nodal time history data arrays | 770 |
| BROUTINE TSEQIN | Program to read, write and store time sequence data | 771 |
| BROUTINE ZTEST | Program to determine if an array contains only zero entries | 772 |
| BROUTINE QUADC | Program to set storage and call tasks for the four-node quadrilateral, elastic continuum element | 772 |
| BROUTINE QDCT1 | Program to read, generate, and write data for the four-node quadrilateral, elastic continuum element | 775 |
| BROUTINE QDCT2 | Program to calculate effective mass matrix for the four-node quadrilateral, elastic continuum element and assemble into the global left-hand-side matrix | 776 |

11.5.4 Subroutine Index for Program Listing (cont.)

| Routine | Description | Page |
|-------------------|---|------|
| SUBROUTINE QDCT3 | Program to calculate residual-force vector for the four-node quadrilateral, elastic continuum element and assemble into the global right-hand-side vector | 777 |
| SUBROUTINE QDCT4 | Program to calculate and print stress, strain, and principal values for the four-node quadrilateral, elastic continuum element | 779 |
| SUBROUTINE QDCTS | Program to calculate and store element time histories for the four-node quadrilateral, elastic continuum element | 780 |
| SUBROUTINE QDCB | Program to set up the strain-displacement matrix B for two-dimensional continuum elements | 780 |
| SUBROUTINE QDCK | Program to form stiffness matrix for a continuum element with NEN nodes | 781 |
| SUBROUTINE QDCKD | Program to form internal force ($-k^e * d^e$) for a continuum element with NEN nodes | 782 |
| SUBROUTINE QDCRSF | Program to read, write and store surface force data for the four-node quadrilateral, elastic continuum element | 782 |
| SUBROUTINE QDCSHG | Program to calculate global derivatives of shape functions and Jacobian determinants for a four-node quadrilateral element | 783 |
| SUBROUTINE QDCSHL | Program to calculate integration-rule weights, shape functions, and local derivatives for a four-node quadrilateral element | 784 |
| SUBROUTINE QDCSTR | Program to calculate stress, strain, and principal values at an integration point for a two-dimensional continuum element | 784 |
| SUBROUTINE QDCSUF | Program to compute consistent surface loads for the four-node quadrilateral, elastic continuum element | 785 |
| SUBROUTINE TRUSS | Program to set storage and call tasks for the three-dimensional, elastic truss element | 785 |
| SUBROUTINE TRUST1 | Program to read, generate, and write element data for the three-dimensional, elastic truss element | 788 |
| SUBROUTINE TRUST2 | Program to calculate effective mass matrix for the three-dimensional, elastic truss element and assemble into the global left-hand-side matrix | 789 |
| SUBROUTINE TRUST3 | Program to calculate residual-force vector for the three-dimensional, elastic truss element and assemble into the global right-hand-side vector | 790 |
| SUBROUTINE TRUST4 | Program to calculate and print stress, strain, and force for the three-dimensional, elastic truss element | 791 |
| SUBROUTINE TRUST5 | Program to calculate and store element time histories for the three-dimensional, elastic truss element | 792 |
| SUBROUTINE TRUSB | Program to set up the strain-displacement matrix B for the three-dimensional, elastic truss element | 792 |

11.5.4 Subroutine Index for Program Listing (cont.)

| Routine | Description | Page |
|-----------------|---|------|
| BROUTINE TRUSHG | Program to calculate global derivatives of shape functions and Jacobian determinants for the three-dimensional, elastic truss element | 793 |
| BROUTINE TRUSHL | Program to calculate integration-rule weights, shape functions and local derivatives for a two or three node one-dimensional element | 793 |
| BROUTINE TRUSK | Program to form stiffness matrix for the three-dimensional, elastic truss element | 794 |
| BROUTINE TRUSKD | Program to form internal force ($-k^e * d^e$) for the three-dimensional, elastic truss element | 795 |
| BROUTINE TRUSPR | Program to read, write, and store properties for the three-dimensional, elastic truss element | 795 |
| BROUTINE TRUSPT | Program to print stress, strain, and force for the three-dimensional, elastic truss element | 795 |
| BROUTINE TRUSTR | Program to calculate stress, strain and force at an integration point for the three-dimensional, elastic truss element | 796 |

11.5.5 Program Listing

```

PROGRAM MAIN
*****
*          * * * D L E A R N * * *
*
*          A LINEAR STATIC AND DYNAMIC
*          FINITE ELEMENT ANALYSIS PROGRAM
*
*          T. J. R. H U G H E S
*          R. M. F E R E N C Z
*          A. M. R A E F S K Y
*
*****                                         MAIN      1
                                         MAIN      2
                                         MAIN      3
                                         MAIN      4
                                         MAIN      5
                                         MAIN      6
                                         MAIN      7
                                         MAIN      8
                                         MAIN      9
                                         MAIN     10
                                         MAIN     11
                                         MAIN     12
                                         MAIN     13
                                         MAIN     14
                                         MAIN     15
                                         MAIN     16
                                         MAIN     17
                                         MAIN     18
                                         MAIN     19
                                         MAIN     20
PROGRAM TO SET STORAGE CAPACITY, PRECISION AND INPUT/OUTPUT UNITS MAIN      21
COMMON /BPOINT/ MFIRST,MLAST,MTOT,IPREC
COMMON /IOUNIT/ IIN,IOUT,IRSI,IRSO
COMMON A(50000)                                         MAIN      22
                                         MAIN      23
                                         MAIN      24
                                         MAIN      25
                                         MAIN      26
                                         MAIN      27
                                         MAIN      28
                                         MAIN      29
                                         MAIN      30
                                         MAIN      31
                                         MAIN      32
                                         MAIN      33
                                         MAIN      34
                                         MAIN      35
                                         MAIN      36
                                         MAIN      37
                                         MAIN      38
                                         MAIN      39
                                         MAIN      40
                                         MAIN      41
MFIRST = ADDRESS OF FIRST AVAILABLE WORD IN BLANK COMMON
MLAST = ADDRESS OF LAST AVAILABLE WORD IN BLANK COMMON
MTOT = TOTAL STORAGE ALLOCATED TO BLANK COMMON
IPREC = PRECISION FLAG; EQ.1, SINGLE PRECISION
        EQ.2, DOUBLE PRECISION
MFIRST = 1
MTOT = 50000
MLAST = MTOT
IPREC = 2
IIN = INPUT UNIT NUMBER
IOUT = OUTPUT UNIT NUMBER
IRSI = RESTART INPUT UNIT NUMBER
IRSO = RESTART OUTPUT UNIT NUMBER

```

```

C      IIN      = 5          MAIN      42
C      IOUT     = 6          MAIN      43
C      IRSIN    = 7          MAIN      44
C      IRSOUT   = 8          MAIN      45
C
C.... SYSTEM-DEPENDENT UNIT/FILE SPECIFICATIONS
C
C      THE FOLLOWING LINES ARE APPROPRIATE FOR THE VMS OPERATING
C      SYSTEM -- CHANGE AS NECESSARY FOR OTHER OPERATING SYSTEMS
C
C      CALL ASSIGN( IIN, 'INPUT.DAT', 9)
C      CALL ASSIGN( IOUT, 'OUTPUT.DAT', 10)
C      CALL ASSIGN( IRSIN, 'RSINPUT.DAT', 11)
C      CALL ASSIGN( IRSOUT,'RSOUTPUT.DAT',12)
C
C      CALL DLEARN
C
C.... SYSTEM-DEPENDENT UNIT/FILE SPECIFICATIONS
C
C      THE FOLLOWING LINES ARE APPROPRIATE FOR THE VMS OPERATING
C      SYSTEM -- CHANGE AS NECESSARY FOR OTHER OPERATING SYSTEMS
C
C      CALL CLOSE(IIN)
C      CALL CLOSE(IOUT)
C      CALL CLOSE(IRGIN)
C      CALL CLOSE(IRGOUT)
C
C      STOP
C      END
C*****
***** SUBROUTINE DLEARN ***** DLEARN 1
C
C.... DLEARN - A LINEAR STATIC AND DYNAMIC FINITE ELEMENT
C      ANALYSIS PROGRAM: GLOBAL DRIVER DLEARN 2
C
C      DOUBLE PRECISION DLEARN 3
C      & TIME,ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE,TEMPF, DLEARN 4
C      & COEFF1,COEFF2,COEFF3,COEFF4,COEFF5,COEFF6, DLEARN 5
C      & COEFF7,COEFF8,ALPHA1,BETA1,GAMMA1,DT1 DLEARN 6
C
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION DLEARN 7
C
C      LOGICAL LDYN DLEARN 8
C      CHARACTER*4 TITLE,CIAO DLEARN 9
C
C.... CATALOG OF COMMON STATEMENTS DLEARN 10
C
C      COMMON /BPOINT/ MFIRST,MLAST,MTOT,IPREC DLEARN 11
C      COMMON /COEFFS/ COEFF1,COEFF2,COEFF3,COEFF4,COEFF5,COEFF6, DLEARN 12
C      & COEFF7,COEFF8,ALPHA1,BETA1,GAMMA1,DT1 DLEARN 13
C      COMMON /COLHTC/ NEQ DLEARN 14
C      COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE DLEARN 15
C      COMMON /DPOINT/ MPSTEP,MPDPRT,MPSPRT,MPHPLT,MPITER,MPALPH,MPBETA, DLEARN 16
C      & MPGAMM,MPDT ,MPIDHS,MPDOUT,MPVPRD,MPDPRD,MPA,MPV DLEARN 17
C      COMMON /ETIMEC/ ETIME(7) DLEARN 18
C      COMMON /GENELC/ N,NEI(3),INCEL(3),INC(3) DLEARN 19
C      COMMON /GENFLC/ TEMPF(6,20),NF,NUMGPF,NINCF(3),INCF(3) DLEARN 20
C      COMMON /HPLOTC/ NPLPTS,LOCPLT,TIME DLEARN 21
C      COMMON /INFO/ IEXEC,IACODE,LDYN,IREADR,IWRITR,IPRTIN,IRANK, DLEARN 22
C      & NUMSEQ,NDOUT,NSD,NUMNP,NDOF,NLVECT,NLTFTN,NPTSLF, DLEARN 23
C      & NUMEG DLEARN 24
C      COMMON /IOUNIT/ IIN,IOUT,IRGIN,IRGOUT DLEARN 25
C      COMMON /LABELS/ LABELO(3),LABEL1(16),LABEL2(3) DLEARN 26
C      COMMON /SPOINT/ MPD,MPX,MPID,MPF,MPG,MPG1,MPDIAG,MPNGRP, DLEARN 27
C      & MPALHS,MPBRHS DLEARN 28
C      COMMON /TITLEC/ TITLE(20) DLEARN 29
C      COMMON A(1) DLEARN 30
C      DATA CIAO/*END*/
C
C.... INPUT PHASE DLEARN 31
C
C      CALL ECHO DLEARN 32
C      100 CONTINUE DLEARN 33
C      DO 200 I=1,7 DLEARN 34
C      200 ETIME(I) = 0.0 DLEARN 35
C      CALL TIMING(T1)
C      READ(IIN,1000) TITLE
C      IF (TITLE(1).EQ.CIAO) RETURN DLEARN 36
C      READ(IIN,2000) IEXEC,IACODE,IREADR,IWRITR,IPRTIN,IRANK,NUMSEQ, DLEARN 37
C      & NDOUT,NSD,NUMNP,NDOF,NLVECT,NLTFTN,NPTSLF,NUMEG DLEARN 38
C      LDYN = .TRUE.
C      IF (IACODE.EQ.1) LDYN = .FALSE.
C      WRITE(IOUT,3000) TITLE , IEXEC,IACODE,IREADR,IWRITR,IPRTIN, DLEARN 39
C      WRITE(IOUT,4000) IRANK , NUMSEQ,NDOUT,NSD,NUMNP,NDOF, DLEARN 40
C      & NLVECT,NLTFTN,NPTSLF,NUMEG DLEARN 41
C
C.... INITIALIZATION PHASE DLEARN 42

```

```

CALL TSEQ          DLEARN 58
IF (LDYN) CALL DYNPTS      DLEARN 59
IF (NDOUT.GT.0) CALL DHIST(A(MPIDHS),NDOUT)      DLEARN 60
CALL STATIN(NEQ)      DLEARN 61
NTSTEP = 0           DLEARN 62
TIME = ZERO         DLEARN 63
IF (LDYN) THEN      DLEARN 64
  IF (IREADR.EQ.1) THEN      DLEARN 65
    ..... READ INITIAL CONDITIONS FROM RESTART FILE      DLEARN 66
    CALL RSIN(A(MPD),A(MPV),A(MPA),NDOF,NUMNP,NTSTEP,TIME)      DLEARN 67
    IF (IPRTIN.EQ.0) THEN      DLEARN 68
      CALL PRINTD('RESTART DISPLACEMENTS ',A(MPD),NDOF,NUMNP,NTSTEP,TIME)      DLEARN 69
      CALL PRINTD('RESTART VELOCITIES ',A(MPV),NDOF,NUMNP,NTSTEP,TIME)      DLEARN 70
      CALL PRINTD('RESTART ACCELERATIONS ',A(MPA),NDOF,NUMNP,NTSTEP,TIME)      DLEARN 71
    ENDIF      DLEARN 72
  ELSE      DLEARN 73
    ..... READ INITIAL CONDITIONS FROM INPUT FILE      DLEARN 74
    CALL INPUT(A(MPD),NDOF,NUMNP,1,1,IPRTIN,TIME)      DLEARN 75
    CALL INPUT(A(MPV),NDOF,NUMNP,2,1,IPRTIN,TIME)      DLEARN 76
    CALL INPUT(A(MPA),NDOF,NUMNP,3,1,IPRTIN,TIME)      DLEARN 77
  ENDIF      DLEARN 78
ENDIF      DLEARN 79
STORE INITIAL KINEMATIC DATA FOR TIME HISTORIES      DLEARN 80
IF (LDYN .AND. NDOUT.GT.0 ) THEN      DLEARN 81
  LOCPLT = 1           DLEARN 82
  CALL STORED(A(MPIDHS),A(MPD),A(MPV),A(MPA),A(MPDOUT),      DLEARN 83
  NDOF,NDOUT)      DLEARN 84
ENDIF      DLEARN 85
INPUT ELEMENT DATA      DLEARN 86
CALL ELEMNT('INPUT___',A(MPNGRP))      DLEARN 87
STORE INITIAL STRESS/STRAIN DATA FOR ELEMENT TIME HISTORIES      DLEARN 88
IF (LDYN) THEN      DLEARN 89
  LOCPLT = 1           DLEARN 90
  CALL ELEMNT('STR_STOR',A(MPNGRP))      DLEARN 91
ENDIF      DLEARN 92
ALLOCATE MEMORY FOR GLOBAL EQUATION SYSTEM      DLEARN 93
CALL EQSET(NEQ,NALHS)      DLEARN 94
CALL TIMING(T2)      DLEARN 95
ETIME(1) = T2 - T1      DLEARN 96
SOLUTION PHASE      DLEARN 97
IF (IEXEC.EQ.1) CALL DRIVER(NTSTEP,NEQ,NALHS)      DLEARN 98
WRITE RESTART FILE      DLEARN 99
IF (LDYN .AND. (IWRITR.EQ.1) )      DLEARN 100
  CALL RSOUT(A(MPD),A(MPV),A(MPA),NDOF,NUMNP,NTSTEP,TIME)      DLEARN 101
PRINT MEMORY-POINTER DICTIONARY      DLEARN 102
CALL PRTDC      DLEARN 103
CALL TIMING(T1)      DLEARN 104
ETIME(2) = T1 - T2      DLEARN 105
PRINT ELAPSED TIME SUMMARY      DLEARN 106
CALL TIMLOG      DLEARN 107
GO TO 100      DLEARN 108

```

```

C
1000 FORMAT(20A4)                                DLEARN 139
2000 FORMAT(16I5)                                DLEARN 140
3000 FORMAT('1',20A4///)                          DLEARN 141
& EXECUTION CONTROL INFORMATION //5X,DLEARN 142
& EXECUTION CODE . . . . . (IEXEC) = , I5//5X,DLEARN 143
& EQ. 0, DATA CHECK . . . . . , /5X,DLEARN 144
& EQ. 1, EXECUTION . . . . . , /5X,DLEARN 145
& ANALYSIS CODE . . . . . (IACODE) = , I5//5X,DLEARN 146
& EQ. 0, DYNAMIC ANALYSIS . . . . . , /5X,DLEARN 147
& EQ. 1, STATIC ANALYSIS . . . . . , /5X,DLEARN 148
& READ RESTART FILE CODE . . . . . (IREADR) = , I5//5X,DLEARN 149
& EQ. 0, DO NOT READ RESTART FILE . . . . . , /5X,DLEARN 150
& EQ. 1, READ RESTART FILE . . . . . , /5X,DLEARN 151
& WRITE RESTART FILE CODE . . . . . (IWRITR) = , I5//5X,DLEARN 152
& EQ. 0, DO NOT WRITE RESTART FILE . . . . . , /5X,DLEARN 153
& EQ. 1, WRITE RESTART FILE . . . . . , /5X,DLEARN 154
& INPUT DATA PRINT CODE . . . . . (IPRTIN) = , I5//5X,DLEARN 155
& EQ. 0, PRINT NODAL AND ELEMENT INPUT DATA . . . . . , /5X,DLEARN 156
& EQ. 1, DO NOT PRINT NODAL AND ELEMENT INPUT DATA . . . . . , /5X,DLEARN 157
4000 FORMAT(5X)                                  DLEARN 159
& RANK CHECK CODE . . . . . (IRANK) = , I5//5X,DLEARN 160
& EQ. 0, DO NOT PERFORM RANK CHECK . . . . . , /5X,DLEARN 161
& EQ. 1, PRINT NUMBERS OF ZERO AND NEGATIVE PIVOTS . . . . . , /5X,DLEARN 162
& EQ. 2, PRINT ALL PIVOTS . . . . . , /5X,DLEARN 163
& NUMBER OF TIME SEQUENCES . . . . . (NUMSEQ) = , I5//5X,DLEARN 164
& NUMBER OF NODAL OUTPUT TIME-HISTORIES . . . . . (NDOUT) = , I5//5X,DLEARN 165
& NUMBER OF SPACE DIMENSIONS . . . . . (NSD) = , I5//5X,DLEARN 166
& NUMBER OF NODAL POINTS . . . . . (NUMNP) = , I5//5X,DLEARN 167
& NUMBER OF NODAL DEGREES-OF-FREEDOM . . . . . (NDOF) = , I5//5X,DLEARN 168
& NUMBER OF LOAD VECTORS . . . . . (NLVECT) = , I5//5X,DLEARN 169
& NUMBER OF LOAD-TIME FUNCTIONS . . . . . (NLTFN) = , I5//5X,DLEARN 170
& NUMBER OF POINTS ON LOAD-TIME FUNCTIONS . . . . . (NPTSLF) = , I5//5X,DLEARN 171
& NUMBER OF ELEMENT GROUPS . . . . . (NUMEG) = , I5//5X,DLEARN 172
C
C      END                                         DLEARN 173
***** SUBROUTINE DRIVER(NTSTEP,NEQ,NALHS)          *****
C
C.... SOLUTION DRIVER PROGRAM                   DRIVER 1
C
C      DOUBLE PRECISION                         DRIVER 2
&      TIME,Coeff1,Coeff2,Coeff3,Coeff4,Coeff5,Coeff6,    DRIVER 3
&      Coeff7,Coeff8,Alpha1,Beta1,Gamma1,DT1           DRIVER 4
C
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C
C      LOGICAL LDYN,LOUT                         DRIVER 5
COMMON /COEFFS/ COEFF1,COEFF2,COEFF3,COEFF4,COEFF5,COEFF6,    DRIVER 6
& COEFF7,COEFF8,ALPHA1,BETA1,GAMMA1,DT1           DRIVER 7
COMMON /DPOINT/ MPSTEP,MPDPRT,MPSVRT,MPHPLT,MPITER,MPALPH,MPBETA,   DRIVER 8
& MPGAMM,NPDT,MPIDHS,MPDOUT,MPVPRD,MPDPRD,MPA,MPV   DRIVER 9
COMMON /ETIMEC/ ETIME(7)                           DRIVER 10
COMMON /HPLOTC/ NPLPTS,LOCPLT,TIME               DRIVER 11
COMMON /INFO / IEXEC,IACODE,LDYN,IREADR,IWRITR,IPRTIN,IRANK,    DRIVER 12
& NUMSEQ,NDOUT,NSD,NUMNP,NDOF,NLVECT,NLTFN,NPTSLF,    DRIVER 13
& NUMEG                                            DRIVER 14
& COMMON /SPOINT/ MPD,MPX,MPID,MPF,MPG,MPG1,MPDIAG,MPNGRP,    DRIVER 15
& MPALHS,MPBRHS                                     DRIVER 16
COMMON A(1)                                       DRIVER 17
C
C.... TIME SEQUENCE LOOP                         DRIVER 18
C
DO 300 NSQ=1,NUMSEQ                            DRIVER 19
C
C.... SET CURRENT TIME SEQUENCE PARAMETERS      DRIVER 20
C
CALL TIMCON(NSQ,A(MPSTEP),A(MPDPRT),A(MPSVRT),A(MPHPLT),A(MPITER),    DRIVER 21
& NSTEP1,NDPRT1,NSPRT1,NHPLT1,NITER1,DRIVER 22
& A(MPALPH),A(MPBETA),A(MPGAMM),A(MPDT))        DRIVER 23
C
C.... FORM EFFECTIVE MASS MATRIX                DRIVER 24
C
CALL CLEAR(A(MPALHS),NALHS)                     DRIVER 25
CALL TIMING(T1)                                 DRIVER 26
CALL ELEMNT('FORM_LHS',A(MPNGRP))              DRIVER 27
CALL TIMING(T2)                                 DRIVER 28
ETIME(3) = ETIME(3) + T2 - T1                  DRIVER 29
C
C.... PERFORM FACTORIZATION OF EFFECTIVE MASS MATRIX
C
CALL FACTOR(A(MPALHS),A(MPDIAG),NEQ)           DRIVER 30
CALL TIMING(T1)                                 DRIVER 31
ETIME(4) = ETIME(4) + T1 - T2                  DRIVER 32
C
C.... RANK CHECK (NOTE: SUBROUTINES "PIVOTS" AND "PRINTP"
C
RETURN TO STATEMENT 300)                         DRIVER 33

```

| | |
|--|------------|
| IF (IRANK.EQ.1) CALL PIVOTS(A(MPALHS),A(MPDIAG),NEQ,NSQ,*300) | DRIVER 51 |
| IF (IRANK.EQ.2) CALL PRINTP(A(MPALHS),A(MPDIAG),NEQ,NSQ,*300) | DRIVER 52 |
| TIME STEP LOOP | DRIVER 53 |
| DO 200 N=1,NSTEP1 | DRIVER 54 |
| TIME = TIME + DT1 | DRIVER 55 |
| NTSTEP = NTSTEP + 1 | DRIVER 56 |
| IF (LDYN) THEN | DRIVER 57 |
| ... PREDICTOR UPDATE OF ALL DEGREES-OF-FREEDOM | DRIVER 58 |
| CALL PREDCT(A(MPD),A(MPV),A(MPA),A(MPDPRD),A(MPVPRD), | DRIVER 59 |
| NDOF,NUMNP) | DRIVER 60 |
| ELSE | DRIVER 61 |
| CALL CLEAR(A(MPD),NDOF*NUMNP) | DRIVER 62 |
| ENDIF | DRIVER 63 |
| IF (NLVECT.GT.0) | DRIVER 64 |
| EVALUATE LOAD-TIME FUNCTIONS AT TIME N+1 | DRIVER 65 |
| CALL Lfac(A(MPG),TIME,A(MPG1),NLTFTN,NPTSLF) | DRIVER 66 |
| IF (DT1.NE.0) | DRIVER 67 |
| ... OVERWRITE PREDICTORS TO ACCOUNT FOR KINEMATIC | DRIVER 68 |
| BOUNDARY CONDITIONS | DRIVER 69 |
| CALL COMPBC(A(MPID),A(MPD),A(MPV),A(MPA),A(MPDPRD),A(MPVPRD), | DRIVER 70 |
| A(MPF),A(MPG1),NDOF,NUMNP,NLVECT,LDYN) | DRIVER 71 |
| MULTI-CORRECTOR ITERATION LOOP | DRIVER 72 |
| DO 100 I=1,NITER1 | DRIVER 73 |
| CALL CLEAR(A(MPBRHS),NEQ) | DRIVER 74 |
| IF (NLTFTN.GT.0) | DRIVER 75 |
| ... EVALUATE LOAD-TIME FUNCTIONS AT TIME N+1+ALPHA | DRIVER 76 |
| CALL Lfac(A(MPG),TIME+ALPHA1*DT1,A(MPG1),NLTFTN,NPTSLF) | DRIVER 77 |
| IF (NLVECT.GT.0) | DRIVER 78 |
| ... FORM NODAL CONTRIBUTION TO RESIDUAL FORCE VECTOR | DRIVER 79 |
| CALL LOAD(A(MPID),A(MPF),A(MPBRHS),A(MPG1),NDOF,NUMNP,NLVECT) | DRIVER 80 |
| FORM ELEMENT CONTRIBUTION TO RESIDUAL FORCE VECTOR | DRIVER 81 |
| CALL TIMING(T1) | DRIVER 82 |
| CALL ELEMNT('FORM_RHS',A(MPNGRP)) | DRIVER 83 |
| CALL TIMING(T2) | DRIVER 84 |
| ETIME(5) = ETIME(5) + T2 - T1 | DRIVER 85 |
| SOLVE EQUATION SYSTEM | DRIVER 86 |
| CALL BACK(A(MPALHS),A(MPBRHS),A(MPDIAG),NEQ) | DRIVER 87 |
| CALL TIMING(T1) | DRIVER 88 |
| ETIME(6) = ETIME(6) + T1 - T2 | DRIVER 89 |
| PERFORM INTERMEDIATE UPDATE OF ACTIVE DEGREES-OF-FREEDOM | DRIVER 90 |
| CALL ITERUP(A(MPID),A(MPD),A(MPDPRD),A(MPVPRD),A(MPA),A(MPBRHS), | DRIVER 91 |
| NDOF,NUMNP,LDYN) | DRIVER 92 |
| CONTINUE | DRIVER 93 |
| IF (LDYN) | DRIVER 94 |
| PERFORM CORRECTOR UPDATE OF ALL DEGREES-OF-FREEDOM | DRIVER 95 |
| CALL CORRCT(A(MPID),A(MPD),A(MPV),A(MPDPRD),A(MPVPRD), | DRIVER 96 |
| NDOF,NUMNP) | DRIVER 97 |
| IF (LOUT(N,NDPRT1)) THEN | DRIVER 98 |
| ... WRITE KINEMATIC OUTPUT | DRIVER 99 |
| | DRIVER 100 |
| | DRIVER 101 |
| | DRIVER 102 |
| | DRIVER 103 |
| | DRIVER 104 |
| | DRIVER 105 |
| | DRIVER 106 |
| | DRIVER 107 |
| | DRIVER 108 |
| | DRIVER 109 |
| | DRIVER 110 |
| | DRIVER 111 |
| | DRIVER 112 |
| | DRIVER 113 |
| | DRIVER 114 |
| | DRIVER 115 |
| | DRIVER 116 |
| | DRIVER 117 |
| | DRIVER 118 |
| | DRIVER 119 |
| | DRIVER 120 |
| | DRIVER 121 |
| | DRIVER 122 |
| | DRIVER 123 |
| | DRIVER 124 |
| | DRIVER 125 |
| | DRIVER 126 |
| | DRIVER 127 |
| | DRIVER 128 |
| | DRIVER 129 |
| | DRIVER 130 |
| | DRIVER 131 |
| | DRIVER 132 |

```

C     CALL PRINTD(' D I S P L A C E M E N T S
&           A(MPD),NDOF,NUMNP,NTSTEP,TIME)
&     IF (LDYN) THEN
&       CALL PRINTD(' V E L O C I T I E S
&           A(MPV),NDOF,NUMNP,NTSTEP,TIME)
&       CALL PRINTD(' A C C E L E R A T I O N S
&           A(MPA),NDOF,NUMNP,NTSTEP,TIME)
ENDIF
ENDIF

C     CALL TIMING(T1)

C.... CALCULATE AND WRITE ELEMENT OUTPUT
C     IF (LOUT(N,NSPRT1)) CALL ELEMNT('STR_PRNT',A(MPNGRP))
C     IF (LDYN.AND.LOUT(N,NHPLT1)) THEN
C       LOCPLT = LOCPLT + 1
C..... NOTE: VARIABLES "LOCPLT" AND "TIME" ARE PASSED INTO SUB-
C       ROUTINE "STORED" AND ELEMENT ROUTINES PERFORMING
C       TASKS ('STR_STOR') BY WAY OF COMMON /HPLOT/C.
C..... STORE KINEMATIC TIME HISTORY DATA
C     CALL STORED(A(MPIDHS),A(MPD),A(MPV),A(MPA),A(MPDDOUT),
&           NDOF,NDOUT)
C..... CALCULATE AND STORE ELEMENT TIME HISTORY DATA
C
C       CALL ELEMNT('STR_STOR',A(MPNGRP))
ENDIF
CALL TIMING(T2)
ETIME(7) = ETIME(7) + T2 - T1

200 CONTINUE
300 CONTINUE
IF (LDYN) THEN
C..... PLOT NODAL TIME-HISTORIES
IF (NDOUT.GT.0) CALL HPLOT(A(MPIDHS),A(MPDDOUT),NDOUT,3,0)
C..... PLOT ELEMENT TIME-HISTORIES
CALL ELEMNT('STR_PLOT',A(MPNGRP))
ENDIF
RETURN
END
*****SUBROUTINE ADDLHS(ALHS,ELEFFM,IDIAG,LM,NEE,LDIAG)
C.... PROGRAM TO ADD ELEMENT LEFT-HAND-SIDE MATRIX TO
GLOBAL LEFT-HAND-SIDE MATRIX
LDIAG = .TRUE., ADD DIAGONAL ELEMENT MATRIX
LDIAG = .FALSE., ADD UPPER TRIANGLE OF FULL ELEMENT MATRIX
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
LOGICAL LDIAG
DIMENSION ALHS(1),ELEFFM(NEE,1),IDIAG(1),LM(1)
IF (LDIAG) THEN
DO 100 J=1,NEE
K = LM(J)
IF (K.GT.0) THEN
L = IDIAG(K)
ALHS(L) = ALHS(L) + ELEFFM(J,J)
ENDIF
100 CONTINUE
ELSE
DO 300 J=1,NEE
K = LM(J)
IF (K.GT.0) THEN

```

```

      DO 200 I=1,J
      M = LM(I)
      IF (M.GT.0) THEN
        L = IDIAG(K) - K + M
      ELSE
        L = IDIAG(M) - M + K
      ENDIF
      ALHS(L) = ALHS(L) + ELEFFM(I,J)
  0   ENDIF
CONTINUE
 0
      ENDIF
 0   CONTINUE
      ENDIF
      RETURN
END
*****SUBROUTINE ADDRHS(BRHS,ELRESF,LM,NEE)
. PROGRAM TO ADD ELEMENT RESIDUAL-FORCE VECTOR TO
  GLOBAL RIGHT-HAND-SIDE VECTOR
  IMPLICIT DOUBLE PRECISION (A-H,O-Z)
. DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
  DIMENSION BRHS(1),ELRESF(1),LM(1)
  DO 100 J=1,NEE
  K = LM(J)
  IF (K.GT.0) BRHS(K) = BRHS(K) + ELRESF(J)
  0 CONTINUE
  RETURN
END
*****SUBROUTINE BACK(A,B,IDIAG,NEQ)
. PROGRAM TO PERFORM FORWARD REDUCTION AND BACK SUBSTITUTION
  IMPLICIT DOUBLE PRECISION (A-H,O-Z)
. DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
  DIMENSION A(1),B(1),IDIAG(1)
  COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
. FORWARD REDUCTION
  JJ = 0
  DO 100 J=1,NEQ
    JJLAST = JJ
    JJ = IDIAG(J)
    JCOLHT = JJ - JJLAST
    IF (JCOLHT.GT.1)
    & B(J) = B(J) - COLDOT(A(JJLAST+1),B(J-JCOLHT+1),JCOLHT-1)
  0 CONTINUE
. DIAGONAL SCALING
  DO 200 J=1,NEQ
    AJJ = A(IDIAG(J))
. WARNING: DIAGONAL SCALING IS NOT PERFORMED IF AJJ EQUALS ZERO
  IF (AJJ.NE.ZERO) B(J) = B(J)/AJJ
  0 CONTINUE
. BACK SUBSTITUTION
  IF (NEQ.EQ.1) RETURN
  JJNEXT = IDIAG(NEQ)
  DO 400 J=NEQ,2,-1
    JJ = JJNEXT
    JJNEXT = IDIAG(J-1)
    JCOLHT = JJ - JJNEXT
    IF (JCOLHT.GT.1) THEN
      BJ = B(J)
      ISTART = J - JCOLHT + 1
      JTEMP = JJNEXT - ISTART + 1
      ADDLHS 32
      ADDLHS 33
      ADDLHS 34
      ADDLHS 35
      ADDLHS 36
      ADDLHS 37
      ADDLHS 38
      ADDLHS 39
      ADDLHS 40
      ADDLHS 41
      ADDLHS 42
      ADDLHS 43
      ADDLHS 44
      ADDLHS 45
      ADDLHS 46
      ADDLHS 47
      ADDLHS 48
      ADDLHS 49
      ADDLHS 50
      ADDLHS 51
      ADDRHS 1
      ADDRHS 2
      ADDRHS 3
      ADDRHS 4
      ADDRHS 5
      ADDRHS 6
      ADDRHS 7
      ADDRHS 8
      ADDRHS 9
      ADDRHS 10
      ADDRHS 11
      ADDRHS 12
      ADDRHS 13
      ADDRHS 14
      ADDRHS 15
      ADDRHS 16
      ADDRHS 17
      ADDRHS 18
      BACK 1
      BACK 2
      BACK 3
      BACK 4
      BACK 5
      BACK 6
      BACK 7
      BACK 8
      BACK 9
      BACK 10
      BACK 11
      BACK 12
      BACK 13
      BACK 14
      BACK 15
      BACK 16
      BACK 17
      BACK 18
      BACK 19
      BACK 20
      BACK 21
      BACK 22
      BACK 23
      BACK 24
      BACK 25
      BACK 26
      BACK 27
      BACK 28
      BACK 29
      BACK 30
      BACK 31
      BACK 32
      BACK 33
      BACK 34
      BACK 35
      BACK 36
      BACK 37
      BACK 38
      BACK 39
      BACK 40
      BACK 41
      BACK 42
      BACK 43
      BACK 44
      BACK 45
      BACK 46

```

```

C          DO 300 I=ISTART,J-1
C          B(I) = B(I) - A(JTEMP+I)*BJ
C          CONTINUE
C        ENDIF
C 400  CONTINUE
C        RETURN
C      END
C*****SUBROUTINE BC(ID,NDOF,NUMNP,NEQ,IPRTIN)
C... PROGRAM TO READ, GENERATE AND WRITE BOUNDARY CONDITION DATA
C     AND ESTABLISH EQUATION NUMBERS
C     DIMENSION ID(NDOF,1)
C     COMMON /IOUNIT/ IIN,IOUT,IRSIM,IRSOUT
C     LOGICAL PFLAG
C     CALL ICLEAR(ID,NDOF*NUMNP)
C     CALL IGEN(ID,NDOF)
C     IF (IPRTIN.EQ.0) THEN
C       NN=0
C       DO 200 N=1,NUMNP
C         PFLAG = .FALSE.
C         DO 100 I=1,NDOF
C           IF (ID(I,N).NE.0) PFLAG = .TRUE.
C 100    CONTINUE
C           IF (PFLAG) THEN
C             NN = NN + 1
C             IF ((MOD(NN,50).EQ.1)) WRITE(IOUT,1000) (I,I=1,NDOF)
C             WRITE(IOUT,2000) N,(ID(I,N),I=1,NDOF)
C           ENDIF
C 200    CONTINUE
C         ENDIF
C     .... ESTABLISH EQUATION NUMBERS
C     NEQ = 0
C     DO 400 N=1,NUMNP
C       DO 300 I=1,NDOF
C         IF (ID(I,N).EQ.0) THEN
C           NEQ = NEQ + 1
C           ID(I,N) = NEQ
C         ELSE
C           ID(I,N) = 1 - ID(I,N)
C         ENDIF
C 300    CONTINUE
C 400    CONTINUE
C        RETURN
C
C 1000 FORMAT('1', ' N O D A L   B O U N D A R Y   C O N D I T I O N   C O '
C & DES///,
C & 5X,' NODE NO.',3X,6(6X,'DOF',I1:)//)
C 2000 FORMAT(6X,I5,5X,6(5X,I5))
C
C      END
C*****BLOCK DATA
C... PROGRAM TO DEFINE OUTPUT LABELS AND NUMERICAL CONSTANTS
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C
C     CHARACTER*4 LABELD,LABEL1,LABEL2
C     COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
C     COMMON /LABELS/ LABELD(3),LABEL1(16),LABEL2(3)
C
C     LABELD(3) = DISPLACEMENT, VELOCITY AND ACCELERATION LABELS
C     LABEL1(16) = OUTPUT LABELS FOR ELEMENT-TYPE 1
C     LABEL2(3) = OUTPUT LABELS FOR ELEMENT-TYPE 2
C
C... NOTE: ADD LABEL ARRAYS FOR ANY ADDITIONAL ELEMENTS

```

BACK 47
BACK 48
BACK 49
BACK 50
BACK 51
BACK 52
BACK 53
BACK 54
BACK 55
BACK 56
BACK 57
BC 1
BC 2
BC 3
BC 4
BC 5
BC 6
BC 7
BC 8
BC 9
BC 10
BC 11
BC 12
BC 13
BC 14
BC 15
BC 16
BC 17
BC 18
BC 19
BC 20
BC 21
BC 22
BC 23
BC 24
BC 25
BC 26
BC 27
BC 28
BC 29
BC 30
BC 31
BC 32
BC 33
BC 34
BC 35
BC 36
BC 37
BC 38
BC 39
BC 40
BC 41
BC 42
BC 43
BC 44
BC 45
BC 46
BC 47
BC 48
BC 49
BC 50
BC 51
BC 52
BC 53
BC 54
BC 55
BC 56
BLOCK 1
BLOCK 2
BLOCK 3
BLOCK 4
BLOCK 5
BLOCK 6
BLOCK 7
BLOCK 8
BLOCK 9
BLOCK 10
BLOCK 11
BLOCK 12
BLOCK 13
BLOCK 14
BLOCK 15
BLOCK 16
BLOCK 17
BLOCK 18

```

C      DATA    ZERO,PT1667,PT25,PT5
C      &     /0.00,0.16666666666667,0.25,0.50/,
C      &     ONE,TWO,THREE,FOUR,FIVE
C      &     /1.00,2.00,3.00,4.00,5.00/
C      DATA LABELD//'DISP','VEL ','ACC '//          BLOCK 19
C      DATA LABEL1//'S 11','S 22','S 12','S 33','PS 1','PS 2',
C      &     TAU,'SANG','E 11','E 22','G 12','E 33',
C      &     'PE 1','PE 2','GAM','EANG'//             BLOCK 20
C      DATA LABEL2//'STRS','FORC','STRN'//          BLOCK 21
C      END                                         BLOCK 22
C*****SUBROUTINE BTDB(ELSTIF,B,DB,NEE,NROWB,NSTR)          BLOCK 23
C      BTDB 1
C      BTDB 2
C      BTDB 3
C      BTDB 4
C      BTDB 5
C      BTDB 6
C      BTDB 7
C      BTDB 8
C      BTDB 9
C      BTDB 10
C      BTDB 11
C      BTDB 12
C      BTDB 13
C      BTDB 14
C      BTDB 15
C      BTDB 16
C      BTDB 17
C      BTDB 18
C      BTDB 19
C      BTDB 20
C      BTDB 21
C      BTDB 22
C      BTDB 23
C      BTDB 24
C      BTDB 25
C      BTDB 26
C      BTDB 27
C      BTDB 28
C      BTDB 29
C      BTDB 30
C      BTDB 31
C      BTDB 32
C      BTDB 33
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C      DIMENSION ELSTIF(NEE,1),B(NROWB,1),DB(NROWB,1)
C      DO 200 J=1,NEE
C      DO 100 I=1,J
C      100 CONTINUE
C      200 CONTINUE
C      RETURN
C      END
C*****SUBROUTINE CLEAR(A,M)
C      CLEAR 1
C      CLEAR 2
C      CLEAR 3
C      CLEAR 4
C      CLEAR 5
C      CLEAR 6
C      CLEAR 7
C      CLEAR 8
C      CLEAR 9
C      CLEAR 10
C      CLEAR 11
C      CLEAR 12
C      CLEAR 13
C      CLEAR 14
C      CLEAR 15
C      CLEAR 16
C      CLEAR 17
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C      DIMENSION A(1)
C      COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
C      DO 100 I=1,M
C      A(I) = ZERO
C      100 CONTINUE
C      RETURN
C      END
C*****FUNCTION COLDOT(A,B,N)
C      COLDOT 1
C      COLDOT 2
C      COLDOT 3
C      COLDOT 4
C      COLDOT 5
C      COLDOT 6
C      COLDOT 7
C      COLDOT 8
C      COLDOT 9
C      COLDOT 10
C      COLDOT 11
C      COLDOT 12
C      COLDOT 13
C      COLDOT 14
C      COLDOT 15
C      COLDOT 16
C      COLDOT 17
C      COLDOT 18
C      COLDOT 19
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C      DIMENSION A(1),B(1)
C      COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
C      COLDOT = ZERO
C      DO 100 I=1,N
C      COLDOT = COLDOT + A(I)*B(I)
C      100 CONTINUE
C      RETURN
C      END
C*****SUBROUTINE COLHT(IDIAG,LM,NED,NEN,NUMEL)
C      COLHT 1
C      COLHT 2
C      COLHT 3
C      COLHT 4
C      COLHT 5
C      COLHT 6
C      COLHT 7
C      COLHT 8
C      COLHT 9
C      COLHT 10
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C      DIMENSION IDIAG(1),LM(NED,NEN,1)
C      COMMON /COLHTC/ NEQ
C      DO 500 K=1,NUMEL
C      MIN = NEQ
C      DO 200 J=1,NEN

```

```

C      DO 100 I=1,NED          COLHT
      NUM = LM(I,J,K)          COLHT
      IF (NUM.GT.0) MIN = MIN0(MIN,NUM)  COLHT
100  CONTINUE                 COLHT
C      200 CONTINUE                 COLHT
C      DO 400 J=1,NEN          COLHT
C      DO 300 I=1,NED          COLHT
      NUM = LM(I,J,K)          COLHT
      IF (NUM.GT.0) THEN        COLHT
         M = NUM - MIN          COLHT
         IF (M.GT.IDIAG(NUM)) IDIAG(NUM) = M  COLHT
      ENDIF                     COLHT
C      300 CONTINUE                 COLHT
      400 CONTINUE                 COLHT
C      500 CONTINUE                 COLHT
C      RETURN                     COLHT
END                                COLHT
C***** SUBROUTINE COMPBC(ID,D,V,A,DPRED,VPRED,F,G1,          COMPBC
& ND0F,NUMNP,NLVECT,Ldyn)          COMPBC
C.... PROGRAM TO COMPUTE DISPLACEMENT, VELOCITY AND          COMPBC
ACCELERATION BOUNDARY CONDITIONS          COMPBC
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)          COMPBC
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION  COMPBC
C      LOGICAL LDYN          COMPBC
      DIMENSION ID(ND0F,1),D(ND0F,1),V(ND0F,1),A(ND0F,1),          COMPBC
& DPRED(ND0F,1),VPRED(ND0F,1),F(ND0F,NUMNP,1),G1(1)          COMPBC
C      COMMON /COEFFS/ COEFF1,COEFF2,COEFF3,COEFF4,COEFF5,COEFF6,          COMPBC
& COEFF7,COEFF8,ALPHA1,BETA1,GAMMA1,DT1          COMPBC
C      COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE  COMPBC
C      DO 700 I=1,ND0F          COMPBC
C      DO 600 J=1,NUMNP          COMPBC
C      K = ID(I,J)          COMPBC
      IF (K.GT.0) GO TO 500          COMPBC
      VAL = ZERO          COMPBC
      DO 100 LV=1,NLVECT          COMPBC
      VAL = VAL + F(I,J,LV)*G1(LV)          COMPBC
100  CONTINUE                 COMPBC
C      M = 1 - K          COMPBC
      GO TO (200,300,400),M          COMPBC
C      200 CONTINUE                 COMPBC
      IF (LDYN) THEN          COMPBC
         TEMP = COEFF1*VAL - ALPHA1*D(I,J)          COMPBC
         A(I,J) = (TEMP - DPRED(I,J))/COEFF5          COMPBC
         DPRED(I,J) = TEMP          COMPBC
         VPRED(I,J) = VPRED(I,J) + COEFF4*A(I,J)          COMPBC
      ELSE          COMPBC
         D(I,J) = VAL          COMPBC
      ENDIF          COMPBC
      GO TO 500          COMPBC
C      300 TEMP = COEFF1*VAL - ALPHA1*V(I,J)          COMPBC
      A(I,J) = (TEMP - VPRED(I,J))/COEFF4          COMPBC
      VPRED(I,J) = TEMP          COMPBC
      DPRED(I,J) = DPRED(I,J) + COEFF5*A(I,J)          COMPBC
      GO TO 500          COMPBC
C      400 DPRED(I,J) = DPRED(I,J) + COEFF5*VAL          COMPBC
      VPRED(I,J) = VPRED(I,J) + COEFF4*VAL          COMPBC
      A(I,J) = VAL          COMPBC
C      500 CONTINUE                 COMPBC
C      600 CONTINUE                 COMPBC
    700 CONTINUE                 COMPBC
      RETURN                     COMPBC
END                                COMPBC
C***** SUBROUTINE CONTM(SHG,XL,W,DET,ELMASS,WORK,CONSTM,IMASS,NINT,  CONTM
& NROWSH,NESD,NEN,NED,NEE,COLUMN)          CONTM
C

```

PROGRAM TO FORM MASS MATRIX FOR A CONTINUUM ELEMENT
WITH "NEN" NODES

IMASS = MASS CODE, EQ. 0, CONSISTENT MASS
EQ. 1, LUMPED MASS
OTHERWISE RETURN

IMPLICIT DOUBLE PRECISION (A-H,O-Z)

DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION

LOGICAL COLUMN

DIMENSION SHG(NROWSH,NEN,1),XL(NESD,1),W(1),DET(1),
& ELMMASS(NEE,1),WORK(1)
& COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE

IF (IMASS.EQ.0) THEN

.... CONSISTENT MASS

DO 400 L=1,NINT
TEMP1 = CONSTM*N(L)*DET(L)

DO 300 J=1,NEN
N = (J - 1)*NED

DO 200 I=1,J
M = (I - 1)*NED
TEMP2 = TEMP1*SHG(NROWSH,I,L)*SHG(NROWSH,J,L)

DO 100 K=1,NED
ELMASS(M + K,N + K) = ELMMASS(M + K,N + K) + TEMP2

0 CONTINUE

0 CONTINUE

0 CONTINUE

0 CONTINUE

ENDIF

IF (IMASS.EQ.1) THEN

.... LUMPED MASS

DSUM = ZERO
TOTMAS = ZERO
CALL CLEAR(WORK,NEN)

DO 600 L=1,NINT
TEMP1 = CONSTM*N(L)*DET(L)
TOTMAS = TOTMAS + TEMP1

DO 500 J=1,NEN
TEMP2 = TEMP1*SHG(NROWSH,J,L)**2
DSUM = DSUM - TEMP2
WORK(J) = WORK(J) + TEMP2

10 CONTINUE

10 CONTINUE

.... SCALE DIAGONAL TO CONSERVE TOTAL MASS

TEMP1 = TOTMAS/DSUM

IF (COLUMN) THEN

..... STORE TERMS IN FIRST COLUMN OF MATRIX

DO 800 J=1,NEN
TEMP2 = TEMP1*WORK(J)
N = (J - 1)*NED

DO 700 K=1,NED
ELMASS(N + K,1) = ELMMASS(N + K,1) + TEMP2

10 CONTINUE

10 CONTINUE

ELSE

..... STORE TERMS ALONG DIAGONAL OF MATRIX

DO 1000 J=1,NEN
TEMP2 = TEMP1*WORK(J)
N = (J - 1)*NED

CONTM 4
CONTM 5
CONTM 6
CONTM 7
CONTM 8
CONTM 9
CONTM 10
CONTM 11
CONTM 12
CONTM 13
CONTM 14
CONTM 15
CONTM 16
CONTM 17
CONTM 18
CONTM 19
CONTM 20
CONTM 21
CONTM 22
CONTM 23
CONTM 24
CONTM 25
CONTM 26
CONTM 27
CONTM 28
CONTM 29
CONTM 30
CONTM 31
CONTM 32
CONTM 33
CONTM 34
CONTM 35
CONTM 36
CONTM 37
CONTM 38
CONTM 39
CONTM 40
CONTM 41
CONTM 42
CONTM 43
CONTM 44
CONTM 45
CONTM 46
CONTM 47
CONTM 48
CONTM 49
CONTM 50
CONTM 51
CONTM 52
CONTM 53
CONTM 54
CONTM 55
CONTM 56
CONTM 57
CONTM 58
CONTM 59
CONTM 60
CONTM 61
CONTM 62
CONTM 63
CONTM 64
CONTM 65
CONTM 66
CONTM 67
CONTM 68
CONTM 69
CONTM 70
CONTM 71
CONTM 72
CONTM 73
CONTM 74
CONTM 75
CONTM 76
CONTM 77
CONTM 78
CONTM 79
CONTM 80
CONTM 81
CONTM 82
CONTM 83
CONTM 84
CONTM 85
CONTM 86
CONTM 87
CONTM 88
CONTM 89
CONTM 90

```

C
      DO 900 K=1,NED
      ELMASS(N + K,N + K) = ELMASS(N + K,N + K) + TEMP2
      CONTINUE
900
1000      CONTINUE
C
      ENDIF
C
      ENDIF
C
      RETURN
END
*****
SUBROUTINE CONTMA(SHG,XL,W,DET,AL,ELMASS,WORK,ELRESF,CONSTM,IMASS,
&           NINT,NROWSH,NESD,NEN,NED,NEE)
C.... PROGRAM TO CALCULATE INERTIAL AND GRAVITY/BODY FORCE (" -M*(A-G") )
C..... FOR A CONTINUUM ELEMENT WITH "NEN" NODES
CCC
      IMASS = MASS CODE, EQ. 0, CONSISTENT MASS
      EQ. 1, LUMPED MASS
      OTHERWISE RETURN
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C
      & DIMENSION SHG(NROWSH,NEN,1),XL(NESD,1),W(1),DET(1),
      &           AL(NED,1),ELMASS(NEE,1),WORK(1),ELRESF(NED,1)
C
      IF (IMASS.EQ.0) THEN
C..... CONSISTENT MASS
C
      DO 300 L=1,NINT
      TEMP = CONSTM*W(L)*DET(L)
C
      DO 200 I=1,NED
      ACC = ROWDOT(SHG(NROWSH,1,L),AL(I,1),NROWSH,NED,NEN)
C
      DO 100 J=1,NEN
      ELRESF(I,J) = ELRESF(I,J) + TEMP*ACC*SHG(NROWSH,J,L)
100      CONTINUE
200      CONTINUE
300      CONTINUE
C
      ENDIF
C
      IF (IMASS.EQ.1) THEN
C..... LUMPED MASS
C
      & CALL CLEAR(ELMASS,NEE)
      & CALL CONTM(SHG,XL,W,DET,ELMASS,WORK,CONSTM,IMASS,NINT,
      &           NROWSH,NESD,NEN,NED,NEE,.TRUE.)
C
      DO 500 J=1,NEN
      K = (J - 1)*NED
C
      DO 400 I=1,NED
      ELRESF(I,J) = ELRESF(I,J) + AL(I,J)*ELMASS(K + I,1)
400      CONTINUE
500      CONTINUE
C
      ENDIF
C
      RETURN
END
*****
SUBROUTINE COORD(X,NSD,NUMNP,IPRTIN)
C.... PROGRAM TO READ, GENERATE AND WRITE COORDINATE DATA
C
      X(NSD,NUMNP) = COORDINATE ARRAY
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C
      DIMENSION X(NSD,1)
      COMMON /IOUNIT/ IIN,IOUT,IRSI,IRSO
C
      CALL GENFL(X,NSD)
C
      IF (IPRTIN.EQ.1) RETURN
      COORD   91
      COORD   92
      COORD   93
      COORD   94
      COORD   95
      COORD   96
      COORD   97
      COORD   98
      COORD   99
      COORD  100
      COORD  101
      COORD  102
      COORD  103
      COORD   1
      COORD   2
      COORD   3
      COORD   4
      COORD   5
      COORD   6
      COORD   7
      COORD   8
      COORD   9
      COORD  10
      COORD  11
      COORD  12
      COORD  13
      COORD  14
      COORD  15
      COORD  16

```

```

DO 100 N=1,NUMNP          COORD    17
IF (MOD(N,50).EQ.1) WRITE(1000,1000) (I,I=1,NSD)      COORD    18
WRITE(1000,2000) N,(X(I,N),I=1,NSD)      COORD    19
100 CONTINUE               COORD    20
                                         COORD    21
                                         COORD    22
                                         COORD    23
                                         COORD    24
                                         COORD    25
                                         COORD    26
                                         COORD    27
                                         COORD    28
                                         ****
SUBROUTINE CORRCT(ID,D,V,DPRED,VPRED,NDOF,NUMNP)      CORRCT    1
... PROGRAM TO PERFORM CORRECTOR UPDATE OF DISPLACEMENTS   CORRCT    2
     AND VELOCITIES                                     CORRCT    3
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)                 CORRCT    4
... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION CORRCT    5
     DIMENSION ID(NDOF,1),D(NDOF,1),V(NDOF,1),DPRED(NDOF,1),   CORRCT    6
&           VPRED(NDOF,1)                                CORRCT    7
& COMMON /COEFFS/ COEFF1,COEFF2,COEFF3,COEFF4,COEFF5,COEFF6,   CORRCT    8
&           COEFF7,COEFF8,ALPHA1,BETA1,GAMMA1,DT             CORRCT    9
& COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE CORRCT   10
     TEMP = ONE/COEFF1                                 CORRCT   11
DO 200 I=1,NDOF          CORRCT   12
     DO 100 J=1,NUMNP          CORRCT   13
     DN = D(I,J)                CORRCT   14
     VN = V(I,J)                CORRCT   15
     D(I,J) = (DPRED(I,J) - DN)*TEMP + DN          CORRCT   16
     V(I,J) = (VPRED(I,J) - VN)*TEMP + VN          CORRCT   17
100 CONTINUE               CORRCT   18
200 CONTINUE               CORRCT   19
                                         CORRCT   20
                                         CORRCT   21
                                         CORRCT   22
                                         CORRCT   23
                                         CORRCT   24
                                         CORRCT   25
                                         CORRCT   26
                                         CORRCT   27
                                         CORRCT   28
                                         CORRCT   29
                                         CORRCT   30
                                         CORRCT   31
                                         ****
SUBROUTINE DCTNRY(NAME,NDIM1,NDIM2,NDIM3,MPOINT,IPR,MLAST) DCTNRY    1
... PROGRAM TO STORE POINTER INFORMATION IN DICTIONARY       DCTNRY    2
     DIMENSION NAME(2)                               DCTNRY    3
     COMMON IA(1)                                  DCTNRY    4
     MLAST = MLAST - 7                            DCTNRY    5
     IA(MLAST+1) = NAME(1)                         DCTNRY    6
     IA(MLAST+2) = NAME(2)                         DCTNRY    7
     IA(MLAST+3) = MPOINT                          DCTNRY    8
     IA(MLAST+4) = NDIM1                           DCTNRY    9
     IA(MLAST+5) = NDIM2                           DCTNRY   10
     IA(MLAST+6) = NDIM3                           DCTNRY   11
     IA(MLAST+7) = IPR                             DCTNRY   12
                                         DCTNRY   13
                                         DCTNRY   14
                                         DCTNRY   15
                                         DCTNRY   16
                                         DCTNRY   17
                                         DCTNRY   18
                                         ****
SUBROUTINE DHIST(IDHIST,NDOUT)                         DHIST    1
... PROGRAM TO READ, WRITE AND STORE NODAL TIME-HISTORY INPUT DATA DHIST    2
     DIMENSION IDHIST(3,1)                           DHIST    3
     COMMON /IOUNIT/ IIN,IOUT,IR SIN,IR SOUT        DHIST    4
     COMMON /LABELS/ LABE LD(3),LABEL1(16),LABEL2(3) DHIST    5
                                         DHIST    6
                                         DHIST    7
                                         DHIST    8
DO 100 N=1,NDOUT          DHIST    9
READ(IIN,2000) NODE,IDO F, IDVA                  DHIST   10
IF (MOD(N,50).EQ.1) WRITE(1000,1000) NDOUT        DHIST   11
WRITE(1000,3000) NODE,IDO F,LABE LD(IDVA)        DHIST   12
IDHIST(1,N) = NODE                           DHIST   13
IDHIST(2,N) = IDO F                          DHIST   14
IDHIST(3,N) = IDVA                           DHIST   15
100 CONTINUE               DHIST   16
                                         DHIST   17
                                         DHIST   18
                                         RETURN

```

```

C
1000 FORMAT('1', ' N O D A L T I M E - H I S T O R Y ', DHIST 19
  & ' I N F O R M A T I O N //5X, DHIST 20
  & ' N U M B E R O F N O D A L T I M E H I S T O R I E S DHIST 21
  & ' 5X, ' NODE DOF KINEMATIC :; . . . (NDOUT ) = ',I5// DHIST 22
  & ' 5X, ' NUMBER NUMBER TYPE :; /) DHIST 23
2000 FORMAT(3I5) DHIST 24
3000 FORMAT(7X,I5,5X,I5,7X,A4) DHIST 25
DHIST 26
DHIST 27
DHIST 28
C      END
C*****SUBROUTINE IDIAG(IDIAG,NEQ,N)
C      PROGRAM TO COMPUTE DIAGONAL ADDRESSES OF LEFT-HAND-SIDE MATRIX
C      DIMENSION IDIAG(1)
C
C      N = 1
C      IDIAG(1) = 1
C      IF (NEQ.EQ.1) RETURN
C
C      DO 100 I=2,NEQ
C      IDIAG(I) = IDIAG(I) + IDIAG(I-1) + 1
100    CONTINUE
C      N = IDIAG(NEQ)
C
C      RETURN
C
C*****SUBROUTINE DYNPTS
C      PROGRAM TO SET MEMORY POINTERS FOR DYNAMIC ANALYSIS DATA ARRAYS
C
C      LOGICAL LDYN
C      COMMON /BPOINT/ MFIRST,MLAST,MTOT,IPREC
C      COMMON /DPOINT/ MPSTEP,MPDPRT,MPSVRT,MPHPLT,MPITER,MPALPH,MPBETA,
&                  MPGAMM,MPDT,MPIDHS,MPDOUT,MPVPRD,MPDPRD,MPA,MPV
&                  COMMON /INFO / IEXEC,IACODE,LDYN,IREADR,INRITR,IPRTIN,IRANK,
&                  NUMSEQ,NDOUT,NSD,NUMNP,NDOF,NLVECT,NLTFTN,NPTSFL,
&                  NUMEG
C      COMMON A(1)
C
C      MPVPRD = MPOINT('VPRED   ',NDOF ,NUMNP ,0,IPREC)
C      MPDPRD = MPOINT('DPRED   ',NDOF ,NUMNP ,0,IPREC)
C      MPA   = MPOINT('A     ',NDOF ,NUMNP ,0,IPREC)
C      MPV   = MPOINT('V     ',NDOF ,NUMNP ,0,IPREC)
C
C      RETURN
C
C*****SUBROUTINE ECHO
C      PROGRAM TO ECHO INPUT DATA
C
C      DIMENSION IA(20)
C      COMMON /IOUNIT/ IIN,IOUT,IRSIN,IRSOUT
C
C      READ(IIN,1000) IECHO
C      IF (IECHO.EQ.0) RETURN
C
C      WRITE(IOUT,2000) IECHO
C      BACKSPACE IIN
C
C      DO 100 I=1,100000
C      READ(IIN,3000,END=200) IA
C      IF ((MOD(I,50).EQ.1) WRITE(IOUT,4000)
C      WRITE(IOUT,5000) IA
100    CONTINUE
C
C      200 CONTINUE
C      REWIND IIN
C      READ(IIN,1000) IECHO
C
C      RETURN
C
1000 FORMAT(16I5)
2000 FORMAT('1', I N P U T D A T A F I L E
  & ' E C H O P R I N T C O D E
  & ' E Q . 0 , N O E C H O O F I N P U T D A T A . . . . . (IECHO ) = ',I5//5X,ECHO 26
  & ' E Q . 1 , E C H O I N P U T D A T A
3000 FORMAT(20A4)
4000 FORMAT(' ',8('123456789*'),//)
5000 FORMAT(' ',20A4)
END
C*****SUBROUTINE ELEMNT(TASK,NGRP)
C      PROGRAM TO CALCULATE ELEMENT TASK NUMBER
ELEMNT 1
ELEMNT 2
ELEMNT 3

```

```

CHARACTER*8 TASK,ELTASK(6)
LOGICAL LDYN
DIMENSION NGRP(1)
COMMON /INFO/ IEXEC,IACODE,LDYN,IREADR,IWRITR,IPRTIN,IRANK,
&           NUMSEQ,NDOUT,NSD,NUMNP,NDOF,NLVECT,NLTFTN,NPTSFLF,
&           NUMEG
& COMMON IA(1)
& DATA NTASK,      ELTASK
& /       6, INPUT
&        'FORM_LHS',
&        'FORM_RHS',
&        'STR_PRNT',
&        'STR_STOR',
&        'STR_PLOT'
& DO 100 I=1,NTASK
& IF (TASK.EQ.ELTASK(I)) ITASK = I
10 CONTINUE
DO 200 NEG=1,NUMEG
IF (ITASK.EQ.1) THEN
  MPNPAR = MPOINT('NPAR      ',16    ,0,0,1)
  NGRP(NEG) = MPNPAR
  CALL ELCARD(IA(MPNPAR),NEG)
ELSE
  MPNPAR = NGRP(NEG)
ENDIF
NTYPE = IA(MPNPAR)
CALL ELMLIB(NTYPE,MPNPAR,ITASK,NEG)
10 CONTINUE
RETURN
END
***** SUBROUTINE ELCARD(NPAR,NEG)
. PROGRAM TO READ ELEMENT GROUP CONTROL CARD
DIMENSION NPAR(1)
COMMON /IOUNIT/ IIN,IOUT,IRSIN,IRSOUT
READ(IIN,1000) (NPAR(I),I=1,16)
WRITE(IOUT,2000) NEG
RETURN
10 FORMAT(16I5)
10 FORMAT(' ELEMENT G R O U P   D A T A      ',//5X,
& ' ELEMENT GROUP NUMBER . . . . . (NEG ) = ',I5///)
END
***** SUBROUTINE ELMLIB(NTYPE,MPNPAR,ITASK,NEG)
. PROGRAM TO CALL ELEMENT ROUTINES
COMMON A(1)
GO TO (100,200),NTYPE
10 CONTINUE
CALL QUADC(ITASK,A(MPNPAR),A(MPNPAR+16),NEG)
RETURN
10 CONTINUE
CALL TRUSS(ITASK,A(MPNPAR),A(MPNPAR+16),NEG)
RETURN
. ADD ADDITIONAL ELEMENTS FOR FUN AND VALUABLE PRIZES
END
***** SUBROUTINE EQSET(NEQ,NALHS)
. PROGRAM TO ALLOCATE STORAGE FOR GLOBAL EQUATION SYSTEM
CHARACTER*4 TITLE
COMMON /BPOINT/ MFIRST,MLAST,MTOT,IPREC
COMMON /IOUNIT/ IIN,IOUT,IRSIN,IRSOUT
COMMON /SPOINT/ MPD,MPX,MPID,MPF,MPG,MPG1,MPDIAG,MNGRP,
&               MPALHS,MPBRHS
& COMMON /TITLEC/ TITLE(20)
COMMON A(1)
. DETERMINE ADDRESSES OF DIAGONALS IN LEFT-HAND-SIDE MATRIX
ELEMNT 4
ELEMNT 5
ELEMNT 6
ELEMNT 7
ELEMNT 8
ELEMNT 9
ELEMNT 10
ELEMNT 11
ELEMNT 12
ELEMNT 13
ELEMNT 14
ELEMNT 15
ELEMNT 16
ELEMNT 17
ELEMNT 18
ELEMNT 19
ELEMNT 20
ELEMNT 21
ELEMNT 22
ELEMNT 23
ELEMNT 24
ELEMNT 25
ELEMNT 26
ELEMNT 27
ELEMNT 28
ELEMNT 29
ELEMNT 30
ELEMNT 31
ELEMNT 32
ELEMNT 33
ELEMNT 34
ELEMNT 35
ELEMNT 36
ELEMNT 37
ELEMNT 38
ELEMNT 39
ELCARD 1
ELCARD 2
ELCARD 3
ELCARD 4
ELCARD 5
ELCARD 6
ELCARD 7
ELCARD 8
ELCARD 9
ELCARD 10
ELCARD 11
ELCARD 12
ELCARD 13
ELCARD 14
ELCARD 15
ELCARD 16
ELCARD 17
ELMLIB 1
ELMLIB 2
ELMLIB 3
ELMLIB 4
ELMLIB 5
ELMLIB 6
ELMLIB 7
ELMLIB 8
ELMLIB 9
ELMLIB 10
ELMLIB 11
ELMLIB 12
ELMLIB 13
ELMLIB 14
ELMLIB 15
ELMLIB 16
ELMLIB 17
ELMLIB 18
ELMLIB 19
EQSET 1
EQSET 2
EQSET 3
EQSET 4
EQSET 5
EQSET 6
EQSET 7
EQSET 8
EQSET 9
EQSET 10
EQSET 11
EQSET 12
EQSET 13

```

```

C
C      CALL DIAG(A(MPDIAG),NEQ,NALHS)
C      MPALHS = MPOINT('ALHS',NALHS,0,0,IPREC)
C      MPBRHS = MPOINT('BRHS',NEQ,0,0,IPREC)
C      MEANBW = NALHS/NEQ
C      NWORDS = MTOT - MLAST + MFIRST - 1
C
C.... WRITE EQUATION SYSTEM DATA
C      WRITE(IOUT,1000) TITLE,NEQ,NALHS,MEANBW,NWORDS
C
C      RETURN
1000 FORMAT('1',20A4///
&' E Q U A T I O N   S Y S T E M   D A T A   : (NEQ ) = ',I8,'/5X,EQSET 14
&' NUMBER OF EQUATIONS : (NALHS ) = ',I8,'/5X,EQSET 15
&' NUMBER OF TERMS IN LEFT-HAND-SIDE MATRIX : (NALHS ) = ',I8,'/5X,EQSET 16
&' MEAN HALF BANDWIDTH : (MEANBW) = ',I8,'/5X,EQSET 17
&' TOTAL LENGTH OF BLANK COMMON REQUIRED : (NWORDS) = ',I8,EQSET 18
C
C      END
C*****SUBROUTINE FACTOR(A,IDIAG,NEQ)
C.... PROGRAM TO PERFORM CRUT FACTORIZATION: A = U(TRANSPOSE) * D * U
C      A(I): COEFFICIENT MATRIX STORED IN COMPACTED COLUMN FORM;
C      AFTER FACTORIZATION CONTAINS D AND U
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C
C      DIMENSION A(1),IDIAG(1)
C      COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
C
C      JJ = 0
C      DO 300 J=1,NEQ
C          JJLAST = JJ
C          JJ = IDIAG(J)
C          JCOLHT = JJ - JJLAST
C
C          IF (JCOLHT.GT.2) THEN
C              ..... FOR COLUMN J AND I.LE.J-1, REPLACE A(I,J) WITH D(I,I)*U(I,J)
C              ISTART = J - JCOLHT + 2
C              JM1 = J - 1
C              IJ = JJLAST + 2
C              II = IDIAG(ISTART-1)
C
C              DO 100 I=ISTART,JM1
C                  IILAST = II
C                  II = IDIAG(I)
C                  ICOLHT = II - IILAST
C                  JLNGTH = I - ISTART + 1
C                  LENGTH = MIN0(ICOLHT-1,JLNGTH)
C                  IF (LENGTH.GT.0)
C                      & A(IJ) = A(IJ) - COLDOT(A(II-LENGTH),A(IJ-LENGTH),LENGTH)
C                  IJ = IJ + 1
C 100          CONTINUE
C
C          ENDIF
C
C          IF (JCOLHT.GE.2) THEN
C              ..... FOR COLUMN J AND I.LE.J-1, REPLACE A(I,J) WITH U(I,J);
C              REPLACE A(J,J) WITH D(J,J).
C              JTEMP = J - JJ
C              DO 200 IJ=JJLAST+1,JJ-1
C                  II = IDIAG(JTEMP + IJ)
C
C              ..... WARNING: THE FOLLOWING CALCULATIONS ARE SKIPPED
C                  IF A(II) EQUALS ZERO
C
C                  IF (A(II).NE.ZERO) THEN
C                      TEMP = A(IJ)
C                      A(IJ) = TEMP/A(II)
C                      A(JJ) = A(JJ) - TEMP*A(IJ)
C                  ENDIF
C 200          CONTINUE
C
C          ENDIF

```

```

DO CONTINUE
  RETURN
END
***** SUBROUTINE FORMLM (ID,IEN,LM,NDOF,NED,NEN,NUMEL)
. PROGRAM TO FORM LM ARRAY
DIMENSION ID(NDOF,1),IEN(NEN,1),LM(NED,NEN,1)
DO 300 K=1,NUMEL
DO 200 J=1,NEN
  NODE=IEN(J,K)
  DO 100 I=1,NDOF
    LM(I,J,K) = ID(I,NODE)
DO CONTINUE
DO CONTINUE
DO CONTINUE
  RETURN
END
***** SUBROUTINE GENEL(IEN,MAT,NEN)
. PROGRAM TO READ AND GENERATE ELEMENT NODE AND MATERIAL NUMBERS
IEN(NEN,NUMEL) = ELEMENT NODE NUMBERS
MAT(NUMEL) = ELEMENT MATERIAL NUMBERS
NEN = NUMBER OF ELEMENT NODES (LE.27)
N = ELEMENT NUMBER
NG = GENERATION PARAMETER
NEL(I) = NUMBER OF ELEMENTS IN DIRECTION I
INCEL(I) = ELEMENT NUMBER INCREMENT FOR DIRECTION I
INC(I) = NODE NUMBER INCREMENT FOR DIRECTION I
DIMENSION IEN(NEN,1),MAT(1),ITEMP(27)
COMMON /IOUNIT/ IIN,IOUT,IR$IN,IR$OUT
COMMON /GENELC/ N,NEL(3),INCEL(3),INC(3)

DO CONTINUE
  READ(IIN,1000) N,M,(ITEMP(I),I=1,NEN),NG
  IF (N.EQ.0) RETURN
  CALL IMOVE(IEN(1,N),ITEMP,NEN)
  MAT(N)=M
  IF (NG.NE.0) THEN
    .... GENERATE DATA
      READ(IIN,1000) (NEL(I),INCEL(I),INC(I),I=1,3)
      CALL GENEL1(IEN,MAT,NEN)
    ENDIF
    GO TO 100
  DO FORMAT(16I5,10X,14I5)
  END
***** SUBROUTINE GENEL1(IEN,MAT,NEN)
. PROGRAM TO GENERATE ELEMENT NODE AND MATERIAL NUMBERS
DIMENSION IEN(NEN,1),MAT(1)
COMMON /GENELC/ N,NEL(3),INCEL(3),INC(3)

. SET DEFAULTS
  CALL GENELD
. GENERATION ALGORITHM
  IE = N
  JE = N
  KE = N
  II = NEL(1)
  JJ = NEL(2)
  KK = NEL(3)
  DO 300 K=1,KK
  DO 200 J=1,JJ
  DO 100 I=1,II
    FACTOR 68
    FACTOR 69
    FACTOR 70
    FACTOR 71
    FACTOR 72
    FORMLM 1
    FORMLM 2
    FORMLM 3
    FORMLM 4
    FORMLM 5
    FORMLM 6
    FORMLM 7
    FORMLM 8
    FORMLM 9
    FORMLM 10
    FORMLM 11
    FORMLM 12
    FORMLM 13
    FORMLM 14
    FORMLM 15
    FORMLM 16
    FORMLM 17
    FORMLM 18
    FORMLM 19
    FORMLM 20
    FORMLM 21
    GENEL 1
    GENEL 2
    GENEL 3
    GENEL 4
    GENEL 5
    GENEL 6
    GENEL 7
    GENEL 8
    GENEL 9
    GENEL 10
    GENEL 11
    GENEL 12
    GENEL 13
    GENEL 14
    GENEL 15
    GENEL 16
    GENEL 17
    GENEL 18
    GENEL 19
    GENEL 20
    GENEL 21
    GENEL 22
    GENEL 23
    GENEL 24
    GENEL 25
    GENEL 26
    GENEL 27
    GENEL 28
    GENEL 29
    GENEL 30
    GENEL 31
    GENEL 32
    GENEL 33
    GENEL 34
    GENEL1 1
    GENEL1 2
    GENEL1 3
    GENEL1 4
    GENEL1 5
    GENEL1 6
    GENEL1 7
    GENEL1 8
    GENEL1 9
    GENEL1 10
    GENEL1 11
    GENEL1 12
    GENEL1 13
    GENEL1 14
    GENEL1 15
    GENEL1 16
    GENEL1 17
    GENEL1 18
    GENEL1 19
    GENEL1 20
    GENEL1 21
    GENEL1 22
    GENEL1 23
    GENEL1 24
    GENEL1 25
    GENEL1 26
  
```

```

C      IF (I.NE.II) THEN          GENEL1  27
C          LE = IE               GENEL1  28
C          IE = IE + INC(1)       GENEL1  29
C          CALL GENELI(IEN(1),IE),IEN(1,LE),INC(1),NEN
C          MAT(IE) = MAT(LE)
C      ENDIF                      GENEL1  30
C 100  CONTINUE                  GENEL1  31
C
C      IF (J.NE.JJ) THEN          GENEL1  32
C          LE = JE               GENEL1  33
C          JE = LE + INC(2)       GENEL1  34
C          CALL GENELI(IEN(1),JE),IEN(1,LE),INC(2),NEN
C          MAT(JE) = MAT(LE)
C          IE = JE
C      ENDIF                      GENEL1  35
C 200  CONTINUE                  GENEL1  36
C
C      IF (K.NE.KK) THEN          GENEL1  37
C          LE = KE               GENEL1  38
C          KE = LE + INC(3)       GENEL1  39
C          CALL GENELI(IEN(1),KE),IEN(1,LE),INC(3),NEN
C          MAT(KE) = MAT(LE)
C          IE = KE
C          JE = KE
C      ENDIF                      GENEL1  40
C 300  CONTINUE                  GENEL1  41
C
C      RETURN                     GENEL1  42
C
C***** SUBROUTINE GENELD
C
C.... PROGRAM TO SET DEFAULTS FOR ELEMENT NODE
C     AND MATERIAL NUMBER GENERATION
C
C      COMMON /GENELC/ N,NEL(3),INC(3),INC(3)
C
C      IF (NEL(1).EQ.0) NEL(1) = 1           GENELD 1
C      IF (NEL(2).EQ.0) NEL(2) = 1           GENELD 2
C      IF (NEL(3).EQ.0) NEL(3) = 1           GENELD 3
C
C      IF (INC(1).EQ.0) INC(1) = 1           GENELD 4
C      IF (INC(2).EQ.0) INC(2) = (1+NEL(1))*INC(1) GENELD 5
C      IF (INC(3).EQ.0) INC(3) = (1+NEL(2))*INC(2) GENELD 6
C
C      IF (INC(1).EQ.0) INC(1) = 1           GENELD 7
C      IF (INC(2).EQ.0) INC(2) = (1+NEL(1))*INC(1) GENELD 8
C      IF (INC(3).EQ.0) INC(3) = (1+NEL(2))*INC(2) GENELD 9
C
C      RETURN                         GENELD 10
C
C***** SUBROUTINE GENELI(IEN2,IEN1,INC,NEN)
C
C.... PROGRAM TO INCREMENT ELEMENT NODE NUMBERS
C
C      DIMENSION IEN1(1),IEN2(1)
C
C      DO 100 I=1,NEN
C      IF (IEN1(I).EQ.0) THEN             GENELI 11
C          IEN2(I) = 0                  GENELI 12
C      ELSE
C          IEN2(I) = IEN1(I) + INC      GENELI 13
C      ENDIF
C 100  CONTINUE
C
C      RETURN                         GENELI 14
C
C***** SUBROUTINE GENFL(A,NRA)
C
C.... PROGRAM TO READ AND GENERATE FLOATING-POINT NODAL DATA
C
C      A      = INPUT ARRAY          GENFL 1
C      NRA   = NUMBER OF ROWS IN A (LE.6) GENFL 2
C      N      = NODE NUMBER         GENFL 3
C      NUMGP = NUMBER OF GENERATION POINTS GENFL 4
C      NINC(I) = NUMBER OF INCREMENTS FOR DIRECTION I GENFL 5
C      INC(I) = INCREMENT FOR DIRECTION I GENFL 6
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z) GENFL 7
C
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C
C      DIMENSION A(NRA,1)             GENFL 8
C      COMMON /IOUNIT/ IIN,IOUT,IR$IN,IR$OUT
C      COMMON /GENFLC/ TEMP(6,20),N,NUMGP,NINC(3),INC(3) GENFL 9

```

```

00 CONTINUE
  READ(IIN,1000) N,NUMGP,(TEMP(I,1),I=1,NRA)
  IF (N.EQ.0) RETURN
  CALL MOVE(A(1,N),TEMP,NRA)
  IF (NUMGP.NE.0) THEN
    DO 200 J=2,NUMGP
      READ(IIN,1000) M,MGEN,(TEMP(I,J),I=1,NRA)
      IF (MGEN.NE.0) CALL MOVE(TEMP(1,J),A(1,M),NRA)
  100 CONTINUE
  READ(IIN,2000) (NINC(I),INC(I),I=1,3)
  CALL GENFL1(A,NRA)
  ENDF
  GO TO 100
100 FORMAT(2I5,6F10.0)
100 FORMAT(16I5)
END
*****SUBROUTINE GENFL1(A,NRA)
.. PROGRAM TO GENERATE FLOATING-POINT NODAL DATA
  VIA ISOPARAMETRIC INTERPOLATION
    IOPT = 1, GENERATION ALONG A LINE
    = 2, GENERATION OVER A SURFACE
    = 3, GENERATION WITHIN A VOLUME
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
.. DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
DIMENSION A(NRA,1),SH(20)
COMMON /GENFLC/ TEMP(6,20),N,NUMGP,NINC(3),INC(3)
COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
  IOPT = 3
  IF (NINC(3).EQ.0) IOPT = 2
  IF (NINC(2).EQ.0) IOPT = 1
  DR = ZERO
  DS = ZERO
  DT = ZERO
  IF (NINC(1).NE.0) DR = TWO/NINC(1)
  IF (NINC(2).NE.0) DS = TWO/NINC(2)
  IF (NINC(3).NE.0) DT = TWO/NINC(3)
  II = NINC(1)+1
  JJ = NINC(2)+1
  KK = NINC(3)+1
  NI = N
  NJ = N
  NK = N
  T = -ONE
  DO 300 K=1,KK
    S = -ONE
    DO 200 J=1,JJ
      R = -ONE
      DO 100 I=1,II
        CALL GENSHP(R,S,T,SH,NUMGP,IOPT)
        CALL MULTAB(TEMP,SH,A(1,NI),6,20,NRA,NUMGP,NRA,1,1)
        NI = NI + INC(1)
        R = R + DR
100  CONTINUE
        NJ = NJ + INC(2)
        NI = NJ
        S = S + DS
300  CONTINUE
        NK = NK + INC(3)
        NI = NK
        T = T + DT
400  CONTINUE
        RETURN
END

```

| | |
|--------|----|
| GENFL | 19 |
| GENFL | 20 |
| GENFL | 21 |
| GENFL | 22 |
| GENFL | 23 |
| GENFL | 24 |
| GENFL | 25 |
| GENFL | 26 |
| GENFL | 27 |
| GENFL | 28 |
| GENFL | 29 |
| GENFL | 30 |
| GENFL | 31 |
| GENFL | 32 |
| GENFL | 33 |
| GENFL | 34 |
| GENFL | 35 |
| GENFL | 36 |
| GENFL | 37 |
| GENFL | 38 |
| GENFL | 39 |
| GENFL1 | 1 |
| GENFL1 | 2 |
| GENFL1 | 3 |
| GENFL1 | 4 |
| GENFL1 | 5 |
| GENFL1 | 6 |
| GENFL1 | 7 |
| GENFL1 | 8 |
| GENFL1 | 9 |
| GENFL1 | 10 |
| GENFL1 | 11 |
| GENFL1 | 12 |
| GENFL1 | 13 |
| GENFL1 | 14 |
| GENFL1 | 15 |
| GENFL1 | 16 |
| GENFL1 | 17 |
| GENFL1 | 18 |
| GENFL1 | 19 |
| GENFL1 | 20 |
| GENFL1 | 21 |
| GENFL1 | 22 |
| GENFL1 | 23 |
| GENFL1 | 24 |
| GENFL1 | 25 |
| GENFL1 | 26 |
| GENFL1 | 27 |
| GENFL1 | 28 |
| GENFL1 | 29 |
| GENFL1 | 30 |
| GENFL1 | 31 |
| GENFL1 | 32 |
| GENFL1 | 33 |
| GENFL1 | 34 |
| GENFL1 | 35 |
| GENFL1 | 36 |
| GENFL1 | 37 |
| GENFL1 | 38 |
| GENFL1 | 39 |
| GENFL1 | 40 |
| GENFL1 | 41 |
| GENFL1 | 42 |
| GENFL1 | 43 |
| GENFL1 | 44 |
| GENFL1 | 45 |
| GENFL1 | 46 |
| GENFL1 | 47 |
| GENFL1 | 48 |
| GENFL1 | 49 |
| GENFL1 | 50 |
| GENFL1 | 51 |
| GENFL1 | 52 |
| GENFL1 | 53 |
| GENFL1 | 54 |
| GENFL1 | 55 |
| GENFL1 | 56 |
| GENFL1 | 57 |
| GENFL1 | 58 |
| GENFL1 | 59 |
| GENFL1 | 60 |
| GENFL1 | 61 |
| GENFL1 | 62 |
| GENFL1 | 63 |
| GENFL1 | 64 |

```

***** SUBROUTINE GENSH(R,S,T,SH,NUMGP,IOPT) *****
C.... PROGRAM TO CALL SHAPE FUNCTION ROUTINES
C..... FOR ISOPARAMETRIC GENERATION
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C.... MODIFY ABOVE CARD FOR SINGLE-PRECISION OPERATION
C
C DIMENSION SH(1)
C GO TO (100,200,300),IOPT
C
C 100 CALL GENSH1(R,SH,NUMGP)
C RETURN
C
C 200 CALL GENSH2(R,S,SH,NUMGP)
C RETURN
C
C 300 CALL GENSH3(R,S,T,SH,NUMGP)
C RETURN
C
C END
*****
***** SUBROUTINE GENSH1(R,SH,N) *****
C.... PROGRAM TO COMPUTE 1D SHAPE FUNCTIONS
C..... FOR ISOPARAMETRIC GENERATION
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C.... MODIFY ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C
C DIMENSION SH(1)
C COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
C
C SH(2) = PT5*R
C SH(1) = PT5 - SH(2)
C SH(2) = PT5 + SH(2)
C IF (N.EQ.3) THEN
C     SH(3) = ONE - R*R
C     SH(1) = SH(1) - PT5*SH(3)
C     SH(2) = SH(2) - PT5*SH(3)
C ENDIF
C
C RETURN
C END
*****
***** SUBROUTINE GENSH2(R,S,SH,N) *****
C.... PROGRAM TO COMPUTE 2D SHAPE FUNCTIONS
C..... FOR ISOPARAMETRIC GENERATION
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C.... MODIFY ABOVE CARD FOR SINGLE-PRECISION OPERATION
C
C DIMENSION SH(1)
C COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
C
C R2 = PT5*R
C R1 = PT5 - R2
C R2 = PT5 + R2
C S2 = PT5*S
C S1 = PT5 - S2
C S2 = PT5 + S2
C SH(1) = R1*S1
C SH(2) = R2*S1
C SH(3) = R2*S2
C SH(4) = R1*S2
C IF (N.EQ.4) RETURN
C
C R3 = ONE - R*R
C S3 = ONE - S*S
C SH(5) = R3*S1
C SH(6) = S3*R2
C SH(7) = R3*S2
C SH(8) = S3*R1
C SH(1) = SH(1) - PT5*(SH(5) + SH(8))
C SH(2) = SH(2) - PT5*(SH(6) + SH(5))
C SH(3) = SH(3) - PT5*(SH(7) + SH(6))
C SH(4) = SH(4) - PT5*(SH(8) + SH(7))
C
C RETURN
C END

```

```
*****
* SUBROUTINE GENSH3(R,S,T,SH,N)
* PROGRAM TO COMPUTE 3D SHAPE FUNCTIONS
* FOR ISOPARAMETRIC GENERATION
* IMPLICIT DOUBLE PRECISION (A-H,O-Z)
* MODIFY ABOVE CARD FOR SINGLE-PRECISION OPERATION
* DIMENSION SH(1)
* COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
R2 = PT5*R          GENSH3 1
R1 = PT5 - R2      GENSH3 2
R2 = PT5 + R2      GENSH3 3
S2 = PT5*S          GENSH3 4
S1 = PT5 - S2      GENSH3 5
S2 = PT5 + S2      GENSH3 6
T2 = PT5*T          GENSH3 7
T1 = PT5 - T2      GENSH3 8
T2 = PT5 + T2      GENSH3 9
RS1 = R1*S1         GENSH3 10
RS2 = R2*S1         GENSH3 11
RS3 = R2*S2         GENSH3 12
RS4 = R1*S2         GENSH3 13
SH(1) = RS1*T1      GENSH3 14
SH(2) = RS2*T1      GENSH3 15
SH(3) = RS3*T1      GENSH3 16
SH(4) = RS4*T1      GENSH3 17
SH(5) = RS1*T2      GENSH3 18
SH(6) = RS2*T2      GENSH3 19
SH(7) = RS3*T2      GENSH3 20
SH(8) = RS4*T2      GENSH3 21
IF (N.EQ.8) RETURN  GENSH3 22
RS1 = ONE - R*R     GENSH3 23
S3 = ONE - S*S     GENSH3 24
T3 = ONE - T*T     GENSH3 25
SH(17) = T3*RS1    GENSH3 26
SH(18) = T3*RS2    GENSH3 27
SH(19) = T3*RS3    GENSH3 28
SH(20) = T3*RS4    GENSH3 29
RS1 = R3*S1         GENSH3 30
RS2 = S3*R2         GENSH3 31
RS3 = R3*S2         GENSH3 32
RS4 = S3*R1         GENSH3 33
SH( 9) = RS1*T1    GENSH3 34
SH(10) = RS2*T1    GENSH3 35
SH(11) = RS3*T1    GENSH3 36
SH(12) = RS4*T1    GENSH3 37
SH(13) = RS1*T2    GENSH3 38
SH(14) = RS2*T2    GENSH3 39
SH(15) = RS3*T2    GENSH3 40
SH(16) = RS4*T2    GENSH3 41
SH( 1) = SH( 1) - PT5*(SH( 9) + SH(12) + SH(17))  GENSH3 42
SH( 2) = SH( 2) - PT5*(SH( 9) + SH(10) + SH(18))  GENSH3 43
SH( 3) = SH( 3) - PT5*(SH(10) + SH(11) + SH(19))  GENSH3 44
SH( 4) = SH( 4) - PT5*(SH(11) + SH(12) + SH(20))  GENSH3 45
SH( 5) = SH( 5) - PT5*(SH(13) + SH(16) + SH(17))  GENSH3 46
SH( 6) = SH( 6) - PT5*(SH(13) + SH(14) + SH(18))  GENSH3 47
SH( 7) = SH( 7) - PT5*(SH(14) + SH(15) + SH(19))  GENSH3 48
SH( 8) = SH( 8) - PT5*(SH(15) + SH(16) + SH(20))  GENSH3 49
RETURN
END
*****
* SUBROUTINE HPLOT(IH,XT,NPLOTS,NROWS,IO)
* PROGRAM TO PLOT OUTPUT HISTORIES
IH(NROWS,NPLOTS) = DOF/COMPONENT INFORMATION
XT(NPLOTS+1,NPLPTS) = OUTPUT HISTORY DATA
XT(      1,NPLPTS) = TIME RECORD
NPLOTS = NUMBER OF HISTORIES TO BE PLOTTED
NPLPTS = NUMBER OF TIME POINTS AT WHICH
        DATA IS TO BE PLOTTED
NROWS = NUMBER OF ROWS IN IH ARRAY
IO = OUTPUT CODE
        EQ.0, NODAL OUTPUT HISTORIES
        EQ.N.GT.0, ELEMENT OUTPUT HISTORIES
        (N = NTYPE IN CALLING ROUTINE)
DOUBLE PRECISION TIME
DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
HPLOT 1
HPLOT 2
HPLOT 3
HPLOT 4
HPLOT 5
HPLOT 6
HPLOT 7
HPLOT 8
HPLOT 9
HPLOT 10
HPLOT 11
HPLOT 12
HPLOT 13
HPLOT 14
HPLOT 15
HPLOT 16
HPLOT 17
HPLOT 18
HPLOT 19

```

```

C
CHARACTER*1 IBLANK,ISTAR,LINE(53)
CHARACTER*4 TITLE,LABELD,LABEL1,LABEL2
DIMENSION IH(NROWS,1),XT(NPLOTS+1,1)
COMMON /HPLOTC/ NPLPTS,LOCPLT,TIME
COMMON /IOUNIT/ IIN,IOUT,IRSI,N,IRSOUT
COMMON /LABELS/ LABELD(3),LABEL1(16),LABEL2(3)
COMMON /TITLEC/ TITLE(20)
HPLOT 20
HPLOT 21
HPLOT 22
HPLOT 23
HPLOT 24
HPLOT 25
HPLOT 26
HPLOT 27
HPLOT 28
HPLOT 29
HPLOT 30
HPLOT 31
HPLOT 32
HPLOT 33
HPLOT 34
HPLOT 35
HPLOT 36
HPLOT 37
HPLOT 38
HPLOT 39
HPLOT 40
HPLOT 41
HPLOT 42
HPLOT 43
HPLOT 44
HPLOT 45
HPLOT 46
HPLOT 47
HPLOT 48
HPLOT 49
HPLOT 50
HPLOT 51
HPLOT 52
HPLOT 53
HPLOT 54
HPLOT 55
HPLOT 56
HPLOT 57
HPLOT 58
HPLOT 59
HPLOT 60
HPLOT 61
HPLOT 62
HPLOT 63
HPLOT 64
HPLOT 65
HPLOT 66
HPLOT 67
HPLOT 68
HPLOT 69
HPLOT 70
HPLOT 71
HPLOT 72
HPLOT 73
HPLOT 74
HPLOT 75
HPLOT 76
HPLOT 77
HPLOT 78
HPLOT 79
HPLOT 80
HPLOT 81
HPLOT 82
HPLOT 83
***** SUBROUTINE ICLEAR(IA,M) ***** ICLEAR 1
ICLEAR 2
ICLEAR 3
ICLEAR 4
ICLEAR 5
ICLEAR 6
ICLEAR 7
ICLEAR 8
ICLEAR 9
ICLEAR 10
ICLEAR 11
ICLEAR 12
***** SUBROUTINE IGEN(IA,M) ***** IGEN 1
IGEN 2
IGEN 3
IGEN 4
IGEN 5
IGEN 6
IGEN 7
IGEN 8
IGEN 9
***** PROGRAM TO READ AND GENERATE INTEGER NODAL DATA *****
CC  IA = INPUT ARRAY
CC  M = NUMBER OF ROWS IN IA
CC  N = NODE NUMBER
CC  NE = END NODE IN GENERATION SEQUENCE
CC  NG = GENERATION INCREMENT

```

```

DIMENSION IA(M,1),IB(13)
COMMON /IOUNIT/ IIN,IOUT,IR$IN,IR$OUT
) CONTINUE
READ(IIN,1000) N,NE,NG,(IB(I),I=1,M)
IF (N.EQ.0) RETURN
IF (NG.EQ.0) THEN
  NE = N
  NG = 1
ELSE
  NE = NE - MOD(NE-N,NG)
ENDIF
DO 200 I=N,NE,NG
CALL IMOVE(IA(1,I),IB,M)
) CONTINUE
GO TO 100
) FORMAT(16I5)
END
***** SUBROUTINE IMOVE(IA,IB,N)
PROGRAM TO MOVE AN INTEGER ARRAY
DIMENSION IA(1),IB(1)
DO 100 I=1,N
IA(I)=IB(I)
) CONTINUE
RETURN
END
***** SUBROUTINE INPUT(F,NDOF,NUMNP,J,NLVECT,IPRTIN,TIME)
PROGRAM TO READ, GENERATE AND WRITE NODAL INPUT DATA
F(NDOF,NUMNP,NLVECT) = PRESCRIBED FORCES/KINEMATIC DATA (J=0)
= INITIAL DISPLACEMENTS (J=1)
= INITIAL VELOCITIES(J=2)
= INITIAL ACCELERATIONS (J=3)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
LOGICAL LZERO
DIMENSION F(NDOF,NUMNP,1)
COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
COMMON /IOUNIT/ IIN,IOUT,IR$IN,IR$OUT
CALL CLEAR(F,NLVECT*NUMNP*NDOF)
DO 100 NLV=1,NLVECT
CALL GENFL(F(1,1,NLV),NDOF)
CALL ZTEST(F(1,1,NLV),NDOF*NUMNP,LZERO)
IF (IPRTIN.EQ.0) THEN
  IF (LZERO) THEN
    IF (J.EQ.0) WRITE(IOUT,1000) NLV
    IF (J.EQ.1) WRITE(IOUT,2000)
    IF (J.EQ.2) WRITE(IOUT,3000)
    IF (J.EQ.3) WRITE(IOUT,4000)
  ELSE
    IF (J.EQ.0) CALL PRINTF(F,NDOF,NUMNP,NLV)
    IF (J.EQ.1)
      & CALL PRINTD(' I N I T I A L   D I S P L A C E M E N T S ',F,NDOF,NUMNP,0,TIME)
    IF (J.EQ.2)
      & CALL PRINTD(' I N I T I A L   V E L O C I T I E S ',F,NDOF,NUMNP,0,TIME)
    IF (J.EQ.3)
      & CALL PRINTD(' I N I T I A L   A C C E L E R A T I O N S ',F,NDOF,NUMNP,0,TIME)
  ENDIF
ENDIF
) CONTINUE
      IGEN 10
      IGEN 11
      IGEN 12
      IGEN 13
      IGEN 14
      IGEN 15
      IGEN 16
      IGEN 17
      IGEN 18
      IGEN 19
      IGEN 20
      IGEN 21
      IGEN 22
      IGEN 23
      IGEN 24
      IGEN 25
      IGEN 26
      IGEN 27
      IGEN 28
      IGEN 29
      IGEN 30
      IGEN 31
IMOVE 1
IMOVE 2
IMOVE 3
IMOVE 4
IMOVE 5
IMOVE 6
IMOVE 7
IMOVE 8
IMOVE 9
IMOVE 10
IMOVE 11
IMOVE 12
INPUT 1
INPUT 2
INPUT 3
INPUT 4
INPUT 5
INPUT 6
INPUT 7
INPUT 8
INPUT 9
INPUT 10
INPUT 11
INPUT 12
INPUT 13
INPUT 14
INPUT 15
INPUT 16
INPUT 17
INPUT 18
INPUT 19
INPUT 20
INPUT 21
INPUT 22
INPUT 23
INPUT 24
INPUT 25
INPUT 26
INPUT 27
INPUT 28
INPUT 29
INPUT 30
INPUT 31
INPUT 32
INPUT 33
INPUT 34
INPUT 35
INPUT 36
INPUT 37
INPUT 38
INPUT 39
INPUT 40
INPUT 41
INPUT 42
INPUT 43
INPUT 44
INPUT 45
INPUT 46
INPUT 47
INPUT 48
INPUT 49
INPUT 50

```

```

C
C***** RETURN ***** INPUT 5
1000 FORMAT('1///', ' THERE ARE NO NONZERO PRESCRIBED FORCES AND ',;I5) INPUT 5
&   'KINEMATIC BOUNDARY CONDITIONS FOR LOAD VECTOR NUMBER.',;I5) INPUT 5
2000 FORMAT('1///', ' THERE ARE NO NONZERO INITIAL DISPLACEMENTS.') INPUT 5
3000 FORMAT('1///', ' THERE ARE NO NONZERO INITIAL VELOCITIES.') INPUT 5
4000 FORMAT('1///', ' THERE ARE NO NONZERO INITIAL ACCELERATIONS.') INPUT 5
END INPUT 5
C***** SUBROUTINE INTERP(X,Y,XX,YY,N) ***** INPUT 5
C
C.... PROGRAM TO PERFORM LINEAR INTERPOLATION INPUT 5
C
C      X(I) = ABSCISSAS INPUT 5
C      Y(I) = ORDINATES INPUT 5
C      XX = INPUT ABSCISSA INPUT 5
C      YY = OUTPUT ORDINATE INPUT 5
C      N = TOTAL NUMBER OF DATA POINTS (1.LE.I.LE.N) INPUT 5
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z) INPUT 10
C
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION INPUT 10
C
C      DIMENSION X(1),Y(1) INPUT 10
C
C      IF (XX.LE.X(1)) THEN INPUT 10
C          YY = Y(1) INPUT 10
C
C      ELSE IF (XX.GE.X(N)) THEN INPUT 20
C          YY = Y(N) INPUT 20
C
C      ELSE INPUT 20
C          DO 100 I=1,N INPUT 20
C              IF (X(I).GE.XX) INPUT 20
C                  YY = Y(I-1) + (XX - X(I-1))*(Y(I) - Y(I-1))/(X(I) - X(I-1)) INPUT 20
C 100 CONTINUE INPUT 20
C
C      ENDIF INPUT 20
C
C      RETURN INPUT 30
C
C***** SUBROUTINE ITERUP(ID,D,DPRED,VPRED,A,BRHS,NDOF,NUMNP,LDYN) ***** INPUT 1
C
C.... PROGRAM TO PERFORM INTERMEDIATE UPDATE OF DISPLACEMENTS, INPUT 1
C      VELOCITIES AND ACCELERATIONS DURING ITERATIVE LOOP IN INPUT 1
C      PREDICTOR/CORRECTOR ALGORITHM INPUT 1
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z) INPUT 6
C
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION INPUT 8
C
C      LOGICAL LDYN INPUT 10
C      DIMENSION ID(NDOF,1),D(NDOF,1),DPRED(NDOF,1),VPRED(NDOF,1), INPUT 11
C      & A(NDOF,1),BRHS(1) INPUT 12
C      & COMMON /COEFFS/ COEFF1,COEFF2,COEFF3,COEFF4,COEFF5,COEFF6, INPUT 13
C      & COEFF7,COEFF8,ALPHA1,BETA1,GAMMA1,DT1 INPUT 14
C
C      IF (LDYN) THEN INPUT 15
C
C          DO 200 I=1,NDOF INPUT 16
C
C              DO 100 J=1,NUMNP INPUT 17
C                  K = ID(I,J) INPUT 18
C
C                  IF (K.GT.0) THEN INPUT 19
C                      DPRED(I,J) = DPRED(I,J) + COEFF5*BRHS(K) INPUT 20
C                      VPRED(I,J) = VPRED(I,J) + COEFF4*BRHS(K) INPUT 21
C                      A(I,J) = A(I,J) + BRHS(K) INPUT 22
C
C                  ENDIF INPUT 23
C 100 CONTINUE INPUT 24
C 200 CONTINUE INPUT 25
C
C      ELSE INPUT 30
C
C          DO 400 I=1,NDOF INPUT 31
C
C              DO 300 J=1,NUMNP INPUT 32
C                  K = ID(I,J) INPUT 33
C
C                  IF (K.GT.0) D(I,J) = BRHS(K) INPUT 34
C
C 300 CONTINUE INPUT 35
C 400 CONTINUE INPUT 36
C
C      ENDIF INPUT 41
C
C      RETURN INPUT 43
C
C***** 
```

```
*****
SUBROUTINE LFAC(G,T,G1,NLTFTN,NPTSLF) *****1
.. PROGRAM TO COMPUTE LOAD FACTORS AT TIME T *****2
IMPLICIT DOUBLE PRECISION (A-H,O-Z) *****3
.. DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION *****4
DIMENSION G(NPTSLF,2,1),G1(1) *****5
DO 100 NLF=1,NLTFTN *****6
CALL INTERP(G(1,1,NLF),G(1,2,NLF),T,G1(NLF),NPTSLF) *****7
00 CONTINUE *****8
RETURN *****9
END *****10
*****
SUBROUTINE LOAD(ID,F,BRHS,G1,NDOF,NUMNP,NLVECT) *****11
.. PROGRAM TO ACCUMULATE NODAL FORCES AND TRANSFER INTO *****12
RIGHT-HAND-SIDE VECTOR *****13
IMPLICIT DOUBLE PRECISION (A-H,O-Z) *****14
.. DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION *****15
DIMENSION ID(NDOF,1),F(NDOF,NUMNP,1),BRHS(1),G1(1) *****16
DO 300 I=1,NDOF *****17
DO 200 J=1,NUMNP *****18
K = ID(I,J) *****19
IF (K.GT.0) THEN *****20
    DO 100 NLV=1,NLVECT *****21
    BRHS(K) = BRHS(K) + F(I,J,NLV)*G1(NLV) *****22
00 CONTINUE *****23
ENDIF *****24
00 CONTINUE *****25
00 CONTINUE *****26
RETURN *****27
END *****28
*****
SUBROUTINE LOCAL(IEN,X,XL,NEN,NROWX,NROWXL) *****29
.. PROGRAM TO LOCALIZE A GLOBAL ARRAY *****1
NOTE: IT IS ASSUMED NROWXL.LE.NROWX *****2
IMPLICIT DOUBLE PRECISION (A-H,O-Z) *****3
.. DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION *****4
DIMENSION IEN(1),X(NROWX,1),XL(NROWXL,1) *****5
DO 200 J=1,NEN *****6
NODE = IEN(J) *****7
DO 100 I=1,NROWXL *****8
XL(I,J)= X(I,NODE) *****9
00 CONTINUE *****10
00 CONTINUE *****11
RETURN *****12
END *****13
*****
FUNCTION LOUT(I,J) *****14
.. PROGRAM TO DETERMINE LOGICAL SWITCH *****15
LOGICAL LOUT *****16
LOUT = .FALSE. *****17
IF (J.EQ.0) RETURN *****18
IF ((MOD(I,J).EQ.0)) LOUT = .TRUE. *****19
RETURN *****20
END *****21
*****
SUBROUTINE LTIMEF(G,NPTSLF,NLTFTN,IPRTIN) *****22
.. PROGRAM TO READ, WRITE AND STORE LOAD-TIME FUNCTIONS *****23
LTIMEF *****24
```

```

C      G(I,1,L) = TIME I FOR LOAD-TIME FUNCTION L          LTIMEF
C      G(I,2,L) = LOAD FACTOR AT TIME I FOR LOAD-TIME FUNCTION L  LTIMEF
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                  LTIMEF
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION  LTIMEF
C      DIMENSION G(NPTSLF,2,NLTFTN)                         LTIMEF
C      COMMON /IOUNIT/ IIN,IOUT,IRSI,IRSO
C      DO 200 L=1,NLTFTN                                     LTIMEF
C          DO 100 I=1,NPTSLF                                 LTIMEF
C              READ(IIN,1000) G(I,1,L),G(I,2,L)             LTIMEF
C 100    CONTINUE                                         LTIMEF
C 200    CONTINUE                                         LTIMEF
C      IF (IPRTIN.EQ.1) RETURN                            LTIMEF
C      WRITE(IOUT,2000) NLTFTN                           LTIMEF
C      DO 400 L=1,NLTFTN                                 LTIMEF
C          DO 300 I=1,NPTSLF                         LTIMEF
C              IF (MOD(I,50).EQ.1) WRITE(IOUT,3000) L  LTIMEF
C                  WRITE(IOUT,4000) G(I,1,L),G(I,2,L)  LTIMEF
C 300    CONTINUE                                         LTIMEF
C 400    CONTINUE                                         LTIMEF
C      RETURN                                            LTIMEF
C
C 1000 FORMAT(2F10.0)                                     LTIMEF
C 2000 FORMAT('1',L0,A,D-TIME  F U N C T I O N   D A T A ',//5X) LTIMEF
C     & ' NUMBER OF LOAD-TIME FUNTIONS   : (NLTFTN ) = ,I5    LTIMEF
C 3000 FORMAT(//5X,FUNCTION NUMBER : I5,/)               LTIMEF
C     & 16X,'TIME',13X,'LOAD FACTOR')                 LTIMEF
C 4000 FORMAT(5X,2(1PE20.8))                           LTIMEF
C
C      END                                              LTIMEF
C*****SUBROUTINE MATADD(A,B,C,MA,MB,MC,M,N,IOPT)          *****MATADD
C
C.... PROGRAM TO ADD RECTANGULAR MATRICES                MATADD
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                  MATADD
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION  MATADD
C      DIMENSION A(MA,1),B(MB,1),C(MC,1)                  MATADD
C      GO TO (1000,2000,3000),IOPT                         MATADD
C.... IOPT = 1, ADD ENTIRE MATRICES                      MATADD
C
C 1000 DO 1200 J=1,N                                     MATADD
C      DO 1100 I=1,M                                     MATADD
C          C(I,J) = A(I,J) + B(I,J)                   MATADD
C 1100 CONTINUE                                         MATADD
C 1200 CONTINUE                                         MATADD
C      RETURN                                            MATADD
C.... IOPT = 2, ADD LOWER TRIANGULAR AND DIAGONAL ELEMENTS  MATADD
C
C 2000 DO 2200 J=1,N                                     MATADD
C      DO 2100 I=J,M                                     MATADD
C          C(I,J) = A(I,J) + B(I,J)                   MATADD
C 2100 CONTINUE                                         MATADD
C 2200 CONTINUE                                         MATADD
C      RETURN                                            MATADD
C.... IOPT = 3, ADD UPPER TRIANGULAR AND DIAGONAL ELEMENTS  MATADD
C
C 3000 DO 3200 J=1,N                                     MATADD
C      DO 3100 I=1,J                                     MATADD
C          C(I,J) = A(I,J) + B(I,J)                   MATADD
C 3100 CONTINUE                                         MATADD
C 3200 CONTINUE                                         MATADD
C      RETURN                                            MATADD
C
C      END                                              MATADD

```

```
*****
SUBROUTINE MEANSH(SHGBAR,W,DET,R,SHG,NEN,NINT,IOPt,NEsd,NRowsh) MEANsh 1
. PROGRAM TO CALCULATE MEAN VALUES OF SHAPE FUNCTION MEANsh 2
. GLOBAL DERIVATIVES FOR B-BAR METHOD MEANsh 3
. NOTE: IF IOPt.EQ.2, DET(L) = DET(L)*R(L) UPON ENTRY MEANsh 4
. IMPLICIT DOUBLE PRECISION (A-H,O-Z) MEANsh 5
. DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION MEANsh 6
. DIMENSION SHGBAR(3,1),W(1),DET(1),R(1),SHG(NRowsh,NEN,1) MEANsh 7
. COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE MEANsh 8
. CALL CLEAR(SHGBAR,3*NEN) MEANsh 9
. VOLINV = ONE/COLDOT(W,DET,NINT) MEANsh 10
. DO 300 L=1,NINT MEANsh 11
. TEMP1 = W(L)*DET(L)*VOLINV MEANsh 12
. IF (IOPt.EQ.2) TEMP2 = TEMP1/R(L) MEANsh 13
. DO 200 J=1,NEN MEANsh 14
. DO 100 I=1,NEsd MEANsh 15
. SHGBAR(I,J) = SHGBAR(I,J) + TEMP1*SHG(I,J,L) MEANsh 16
. 00 CONTINUE MEANsh 17
. IF (IOPt.EQ.2) SHGBAR(3,J) = SHGBAR(3,J) + TEMP2*SHG(3,J,L) MEANsh 18
. 00 CONTINUE MEANsh 19
. 00 CONTINUE MEANsh 20
. RETURN MEANsh 21
. END MEANsh 22
. *****
SUBROUTINE MINMAX(X,XMAX,XMIN,L,M,N) MINMAX 1
. PROGRAM TO COMPUTE THE MIN AND MAX IN THE ROW OF A MATRIX MINMAX 2
. X = MATRIX MINMAX 3
. L = NUMBER OF ROWS IN X MINMAX 4
. M = NUMBER OF COLUMNS IN X MINMAX 5
. N = ROW NUMBER MINMAX 6
. DIMENSION X(L,1) MINMAX 7
. XMAX = X(N,1) MINMAX 8
. XMIN = X(N,1) MINMAX 9
. DO 100 I = 2,M MINMAX 10
. IF (X(N,I).GT.XMAX) XMAX = X(N,I) MINMAX 11
. IF (X(N,I).LT.XMIN) XMIN = X(N,I) MINMAX 12
. 00 CONTINUE MINMAX 13
. RETURN MINMAX 14
. END MINMAX 15
. *****
SUBROUTINE MOVE(A,B,N) MOVE 1
. PROGRAM TO MOVE A FLOATING-POINT ARRAY MOVE 2
. IMPLICIT DOUBLE PRECISION (A-H,O-Z) MOVE 3
. DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION MOVE 4
. DIMENSION A(1),B(1) MOVE 5
. DO 100 I=1,N MOVE 6
. A(I) = B(I) MOVE 7
. 00 CONTINUE MOVE 8
. RETURN MOVE 9
. END MOVE 10
. *****
FUNCTION MPOINT(NAME,NDIM1,NDIM2,NDIM3,IPR) MPOINT 1
. PROGRAM TO CALCULATE STORAGE POINTER MPOINT 2
. DIMENSION NAME(2) MPOINT 3
. COMMON /BPOINT/ MFIRST,MLAST,MTOT,IPREC MPOINT 4
. MPOINT = MFIRST MPOINT 5
. IF (IPREC.EQ.2 .AND. MOD(MPOINT,2).EQ.0 ) MPOINT = MPOINT + 1 MPOINT 6
. CALL DCTNRY(NAME,NDIM1,NDIM2,NDIM3,MPOINT,IPR,MLAST) MPOINT 7
. MFIRST = MPOINT + NDIM1*MAX0(1,NDIM2)*MAX0(1,NDIM3)*IPR MPOINT 8
. IF (MFIRST.GE.MLAST) CALL SERROR(NAME,MFIRST-MLAST) MPOINT 9
. MPOINT = 10
. MPOINT = 11
. MPOINT = 12
```

```

C      RETURN
C      END
C*****SUBROUTINE MULTAB(A,B,C,MA,MB,MC,L,M,N,IOPT)
C
C..... PROGRAM TO MULTIPLY TWO MATRICES
C
C      L = RANGE OF DOT-PRODUCT INDEX
C      M = NUMBER OF ACTIVE ROWS IN C
C      N = NUMBER OF ACTIVE COLUMNS IN C
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C..... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C
C      DIMENSION A(MA,1),B(MB,1),C(MC,1)
C
C      GO TO (1000,2000,3000,4000),IOPT
C
C..... IOPT = 1, C(I,J) = A(I,K)*B(K,J) , (C = A * B)
C
C1000 DO 1200 I=1,M
C
C      DO 1100 J=1,N
C      C(I,J) = RCDOT(A(I,1),B(1,J),MA,L)
C1100 CONTINUE
C
C1200 CONTINUE
C
C      RETURN
C
C..... IOPT = 2, C(I,J) = A(K,I)*B(K,J) (C = AT * B)
C
C2000 DO 2200 I=1,M
C
C      DO 2100 J=1,N
C      C(I,J) = COLDOT(A(1,I),B(1,J),L)
C2100 CONTINUE
C
C2200 CONTINUE
C
C      RETURN
C
C..... IOPT = 3, C(I,J) = A(I,K)*B(J,K) (C = A * BT)
C
C3000 DO 3200 I=1,M
C
C      DO 3100 J=1,N
C      C(I,J) = ROWDOT(A(I,1),B(J,1),MA,MB,L)
C3100 CONTINUE
C
C3200 CONTINUE
C
C      RETURN
C
C..... IOPT = 4, C(I,J) = A(K,I)*B(J,K) (C = AT * BT)
C
C4000 DO 4200 I=1,M
C
C      DO 4100 J=1,N
C      C(I,J) = RCDOT(B(J,1),A(1,I),MB,L)
C4100 CONTINUE
C
C4200 CONTINUE
C
C      RETURN
C
C*****SUBROUTINE PIVOTS(A,IDIAG,NEQ,NSQ,*)
C
C..... PROGRAM TO DETERMINE THE NUMBER OF ZERO AND NEGATIVE TERMS IN
C       ARRAY D OF FACTORIZATION A = U(TRANSPOSE) * D * U
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C..... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C
C      DIMENSION A(1),IDIAG(1)
C      COMMON /IOUNIT/ IIN,IOUT,IRSIM,IRSOOUT
C
C      IZ = 0
C      IN = 0
C
C      DO 100 N=1,NEQ
C      I = IDIAG(N)
C      IF (A(I).EQ.0.) IZ = IZ + 1
C      IF (A(I).LT.0.) IN = IN + 1
C100 CONTINUE
C
C      WRITE(IOUT,1000) NSQ,IZ,IN

```

```

RETURN 1                                PIVOTS 23
) FORMAT( ,                                PIVOTS 24
& ZERO AND/OR NEGATIVE PIVOTS ENCOUNTERED   PIVOTS 25
& TIME SEQUENCE NUMBER . . . . . (NSQ ) = ., I5//5X,PIVOTS 26
& NUMBER OF ZEROES . . . . . = ., I5//5X,PIVOTS 28
& NUMBER OF NEGATIVES . . . . . = ., I5//5X,PIVOTS 29
& PIVOTS 30
& PIVOTS 31
END PIVOTS 32
*****SUBROUTINE PREDCT(D,V,A,DPRED,VPRED,NDOF,NUMNP) PREDCT 1
PROGRAM TO CALCULATE PREDICTOR FOR DISPLACEMENTS, VELOCITIES PREDCT 2
AND ACCELERATIONS PREDCT 3
IMPLICIT DOUBLE PRECISION (A-H,O-Z) PREDCT 4
DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION PREDCT 5
DIMENSION D(NDOF,1),V(NDOF,1),A(NDOF,1), PREDCT 6
& DPRED(NDOF,1),VPRED(NDOF,1) PREDCT 9
& COMMON /COEFFS/ COEFF1,COEFF2,COEFF3,COEFF4,COEFF5,COEFF6, PREDCT 10
& COEFF7,COEFF8,ALPHA1,BETA1,GAMMA1,DT1 PREDCT 11
& COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE PREDCT 12
DO 200 I=1,NDOF PREDCT 13
DO 100 J=1,NUMNP PREDCT 14
DPRED(I,J) = D(I,J) + COEFF6*V(I,J) + COEFF7*A(I,J) PREDCT 15
VPRED(I,J) = V(I,J) + COEFF8*A(I,J) PREDCT 16
A(I,J) = ZERO PREDCT 17
I CONTINUE PREDCT 18
I CONTINUE PREDCT 19
RETURN PREDCT 20
END PREDCT 21
*****SUBROUTINE PRINC(N,S,P) PRINC 1
PROGRAM TO COMPUTE PRINCIPAL VALUES OF SYMMETRIC 2ND-RANK TENSOR PRINC 2
S = SYMMETRIC SECOND-RANK TENSOR STORED AS A VECTOR PRINC 3
N = NUMBER OF DIMENSIONS (2 OR 3) PRINC 4
P = VECTOR OF PRINCIPAL VALUES PRINC 5
THE COMPONENTS OF S MUST BE STORED IN THE FOLLOWING ORDERS PRINC 6
2-D PROBLEMS: S11,S22,S12 PRINC 7
3-D PROBLEMS: S11,S22,S33,S12,S23,S31 PRINC 8
IMPLICIT DOUBLE PRECISION (A-H,O-Z) PRINC 9
DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION PRINC 10
DIMENSION S(1),P(1) PRINC 11
COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE PRINC 12
DATA RT2/1.41421356237309/, PI23/2.09439510239321/, PRINC 13
& TWO2/22.50/, FOUR5/45.0/ PRINC 14
IF (N.EQ.2) THEN PRINC 15
... 2-D PROBLEM PRINC 16
A = TWO2/ATAN(ONE) PRINC 17
X = PT5*(S(1) + S(2)) PRINC 18
Y = PT5*(S(1) - S(2)) PRINC 19
R = SQRT(Y*Y + S(3)*S(3)) PRINC 20
P(1) = X + R PRINC 21
P(2) = X - R PRINC 22
P(3) = R PRINC 23
P(4) = FOUR5 PRINC 24
IF (Y.NE.ZERO .OR. S(3).NE.ZERO) P(4) = A*ATAN2(S(3),Y) PRINC 25
ENDIF PRINC 26
IF (N.EQ.3) THEN PRINC 27
... 3-D PROBLEM PRINC 28

```

```

C   100   R = ZERO          PRINC
        X = (S(1) + S(2) + S(3))/THREE
        Y = S(1)*(S(2) + S(3)) + S(2)*S(3)
&       - S(4)*S(4) - S(6)*S(6) - S(5)*S(5)
&       Z = S(1)*S(2)*S(3) - TWO*S(4)*S(6)*S(5) - S(1)*S(5)*S(5)
&       - S(2)*S(6)*S(6) - S(3)*S(4)*S(4)
        T = THREE*X*X - Y
        U = ZERO          PRINC
        IF (T.NE.ZERO) THEN PRINC
        U = SQRT(TWO*T/THREE)
        UCUBED = U*U*U      PRINC
        IF (UCUBED.NE.ZERO) THEN PRINC
        A = (Z + (T - X*X)*X)*RT2/UCUBED
        R = SQRT(ABS(ONE - A*A)) PRINC
        IF ((R.NE.ZERO) .OR. (A.NE.ZERO)) THEN PRINC
        R = ATAN2(R,A)/THREE PRINC
        ELSE
        R = ZERO          PRINC
        ENDIF
        ELSE
        U = ZERO          PRINC
        ENDIF
        ENDIF
        P(1) = X + U*RT2*COS(R) PRINC
        P(2) = X + U*RT2*COS(R - PI23) PRINC
        P(3) = X + U*RT2*COS(R + PI23) PRINC
    ENDIF
C   RETURN          PRINC
C   END          PRINC
C*****SUBROUTINE PRINTD(NAME,DVA,NDOF,NUMNP,NTSTEP,TIME)*****
C   .PROGRAM TO PRINT KINEMATIC DATA
C   .IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C   .DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION
C   .LOGICAL LZERO,LSKIP
C   .DIMENSION NAME(11),DVA(NDOF,1)
C   .COMMON /IOUNIT/ IIN,IOUT,IRSI,IRSO
C
C   NN = 0          PRINTD
C   LSKIP = .TRUE. PRINTD
C
C   DO 100 N=1,NUMNP PRINTD
C   CALL ZTEST(DVA(1,N),NDOF,LZERO) PRINTD
C   IF (.NOT.LZERO) THEN PRINTD
C       NN = NN + 1 PRINTD
C       IF (MOD(NN,50).EQ.1) PRINTD
C           & WRITE(IOUT,1000) NAME,NTSTEP,TIME,(I,I=1,NDOF) PRINTD
C           & WRITE(IOUT,2000) N,(DVA(I,N),I=1,NDOF) PRINTD
C           LSKIP = .FALSE. PRINTD
C   ENDIF
C   100 CONTINUE PRINTD
C
C   IF (LSKIP) THEN PRINTD
C       WRITE(IOUT,1000) NAME,NTSTEP,TIME,(I,I=1,NDOF) PRINTD
C       WRITE(IOUT,3000) PRINTD
C   ENDIF
C
C   RETURN          PRINTD
C
C   1000 FORMAT('1',11A4//5X, PRINTD
C   & ' STEP NUMBER = ',I10//5X, PRINTD
C   & ' TIME = ',1PE10.3//5X, PRINTD
C   & ' NODE NO. ',6(13X,'DOF',I1,:)//) PRINTD
C   2000 FORMAT(6X,I5,10X,6(1PE15.8,2X)) PRINTD
C   3000 FORMAT(' ',' THERE ARE NO NONZERO COMPONENTS') PRINTD
C
C*****SUBROUTINE PRINTF(F,NDOF,NUMNP,NLV)*****
C   .PROGRAM TO PRINT PRESCRIBED FORCE AND BOUNDARY CONDITION DATA PRINTF
C   .IMPLICIT DOUBLE PRECISION (A-H,O-Z) PRINTF
C   .DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION PRINTF
C   .LOGICAL LZERO PRINTF
C   .DIMENSION F(NDOF,NUMNP,1) PRINTF
C   .COMMON /IOUNIT/ IIN,IOUT,IRSI,IRSO PRINTF
C
C   NN = 0          PRINTF 1

```

```

DO 100 N=1,NUMNP
CALL ZTEST(F(1,N,NLV),NDOF,LZERO)
IF (.NOT.LZERO) THEN
  NN = NN + 1
  IF (MOD(NN,50).EQ.1)
&    WRITE(IOUT,1000) NLV,(I,I=1,NDOF)
ENDIF
10 CONTINUE
RETURN

10 FORMAT('1',
& PRES CRIBED FORCES AND KINEMATIC',
& BOUNDARY CONDITIONS//5X,
& LOAD VECTOR NUMBER = ,I5//5X,
& NODE NO. 6(13X,DOF,I1,:)/)
10 FORMAT(6X,I5,10X,6(1PE15.8,2X))
END
***** SUBROUTINE PRINP(A,IDIAG,NEQ,NSQ,*)
10 PROGRAM TO PRINT ARRAY D AFTER CRUT FACTORIZATION
A = U(TRANSPOSE) * D * U
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
10 DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION
DIMENSION A(1),IDIAG(1)
COMMON /IOUNIT/ IIN,IOUT,IRSIM,IRSOOUT
DO 100 N=1,NEQ
IF (MOD(N,50).EQ.1) WRITE(IOUT,1000) NSQ
I = IDIAG(N)
WRITE(IOUT,2000) N,A(I)
10 CONTINUE
RETURN 1
10 FORMAT('1', 'ARRAY D OF FACTORIZATION',//,
& A = U(TRANSPOSE) * D * U ', //5X,PRINTP
& TIME SEQUENCE NUMBER . . . . . (NSQ) = ,I5//5X)PRINTP
10 FORMAT(1X,I5,4X,1PE20.8) . . . . .
END
***** SUBROUTINE PRNTL(MAT,IEN,NEN,NUMEL)
10 PROGRAM TO PRINT DATA FOR ELEMENT WITH "NEN" NODES
NOTE: PRESENTLY THE LABEL FORMATS ARE LIMITED TO
ELEMENTS WITH ONE TO NINE NODES
DIMENSION MAT(1),IEN(NEN,1)
COMMON /IOUNIT/ IIN,IOUT,IRSIM,IRSOOUT
DO 100 N=1,NUMEL
IF (MOD(N,50).EQ.1) WRITE(IOUT,1000) (I,I=1,NEN)
WRITE(IOUT,2000) N,MAT(N),(IEN(I,N),I=1,NEN)
10 CONTINUE
RETURN
10 FORMAT('1',
& ELEMENT DATA',//5X,
& ELEMENT MATERIAL,9(NODE,,I1,:,2X),/5X,
& NUMBER NUMBER//)
10 FORMAT(6X,I5,9(5X,I5))
END
***** SUBROUTINE PROPD2D(RHO,RDAMP,RDAMPK,TH,C,NUMAT,IOPT,NROWB)
10 PROGRAM TO READ, WRITE AND STORE PROPERTIES FOR TWO-DIMENSIONAL
CONTINUUM ELEMENTS
NOTE: THIS ROUTINE IS PRESENTLY RESTRICTED TO THE
ISOTROPIC LINEARLY-ELASTIC CASE
IOPT = 0; PLANE STRESS
      = 1; PLANE STRAIN
      = 2; TORSIONLESS AXISYMMETRIC
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
10 DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION

```

```

C
DIMENSION RH0(1), RDAMPM(1), RDAMPK(1), TH(1), C(NROWB,NROWB,1) PROP2D
COMMON /CONSTS/ ZERO, PT1667, PT25, PT5, ONE, TWO, THREE, FOUR, FIVE PROP2D
COMMON /IOUNIT/ IIN, IOUT, IRSIN, IRSOUT PROP2D
C
DO 100 N=1,NUMAT PROP2D
IF (MOD(N,50).EQ.1) WRITE(IOUT,1000) NUMAT PROP2D
READ(IIN,2000) M,E,POIS,RH0(M),RDAMPM(M),RDAMPK(M),TH(M) PROP2D
IF (TH(M).EQ.ZERO) TH(M)=ONE PROP2D
WRITE(IOUT,3000) M,E,POIS,RH0(M),RDAMPM(M),RDAMPK(M),TH(M) PROP2D
C.... SET MATERIAL CONSTANTS FOR OUT-OF-PLANE COMPONENTS PROP2D
AMU2 = E/(ONE + POIS) PROP2D
ALAM = AMU2*POIS/(ONE - TWO*POIS) PROP2D
C
C(1,4,M) = ALAM PROP2D
C(2,4,M) = ALAM PROP2D
C(3,4,M) = ZERO PROP2D
C(4,4,M) = ALAM + AMU2 PROP2D
C
C(4,1,M) = C(1,4,M) PROP2D
C(4,2,M) = C(2,4,M) PROP2D
C(4,3,M) = C(3,4,M) PROP2D
C.... SET MATERIAL CONSTANTS FOR IN-PLANE COMPONENTS PROP2D
C
IF (IOPT.EQ.0) ALAM = ALAM*AMU2/(ALAM + AMU2) PROP2D
C
C(1,1,M) = ALAM + AMU2 PROP2D
C(1,2,M) = ALAM PROP2D
C(2,2,M) = C(1,1,M) PROP2D
C(1,3,M) = ZERO PROP2D
C(2,3,M) = ZERO PROP2D
C(3,3,M) = PT5*AMU2 PROP2D
C
C(2,1,M) = C(1,2,M) PROP2D
C(3,1,M) = C(1,3,M) PROP2D
C(3,2,M) = C(2,3,M) PROP2D
C
100 CONTINUE PROP2D
C
RETURN PROP2D
C
1000 FORMAT('1', //5X,
'& M A T E R I A L S E T   D A T A ', //5X, PROP2D
'& N U M B E R O F M A T E R I A L S E T S ', 5X, '(NUMAT ) = ', I5//, PROP2D
'& 7X, 'SET', '5X, 'YOUNG', '5X, 'POISSON', '5X, 'MASS', '8X, 'MASS', //5X, PROP2D
'& 6X, 'STIFFNESS', '3X, 'THICKNESS', '6X, 'NUMBER', '3X, 'MODULUS', //5X, PROP2D
'& 6X, 'RATIO', '6X, 'DENSITY', '5X, 'DAMPING', '5X, 'DAMPING', //) PROP2D
2000 FORMAT(I5,5X,7F10.0) PROP2D
3000 FORMAT(4X,I5,3X,6(2X,1PE10.4)) PROP2D
END PROP2D
***** SUBROUTINE PRTDC PRTDC
C
C.... PROGRAM TO PRINT MEMORY-POINTER DICTIONARY PRTDC
C
COMMON /BPOINT/ MFIRST,MLAST,MTOT,IPREC PRTDC
COMMON /IOUNIT/ IIN,IOUT,IRSIN,IRSOUT PRTDC
COMMON IA(1) PRTDC
C
N = (MTOT-MLAST)/7 PRTDC
J = MTOT + 1 PRTDC
C
DO 100 I=1,N PRTDC
IF (MOD(I,50).EQ.1) WRITE(IOUT,1000) PRTDC
J = J - 7 PRTDC
CALL PRTDC1(I,IA(J),IA(J+2),IA(J+3),IA(J+4),IA(J+5),IA(J+6)) PRTDC
100 CONTINUE PRTDC
C
RETURN PRTDC
C
1000 FORMAT('1', //5X,
'& D Y N A M I C   S T O R A G E   A L L O C A T I O N ', PRTDC
'& I N F O R M A T I O N //', PRTDC
'& 12X, 'ARRAY NO.', '5X, 'ARRAY', '8X, 'ADDRESS', '6X, 'DIM1', '6X, 'DIM2', PRTDC
'& 6X, 'DIM3', '6X, 'PREC.', //) PRTDC
C
END PRTDC
***** SUBROUTINE PRTDC1(I,INAME,IADD,NDIM1,NDIM2,NDIM3,IPR) PRTDC1
C
C.... PROGRAM TO PRINT MEMORY-POINTER INFORMATION FOR AN ARRAY PRTDC1
C
DIMENSION INAME(2) PRTDC1
COMMON /IOUNIT/ IIN,IOUT,IRSIN,IRSOUT PRTDC1
SAVE NEG PRTDC1
DATA NELPAR,LEFTHS/'NPAR','ALHS'/

```

```

IF (I.EQ.1) NEG = 1 PRTDC1 9
IF (INAME(1).EQ.NELPAR) THEN PRTDC1 10
  WRITE (IOUT,1000) NEG PRTDC1 11
  NEG = NEG + 1 PRTDC1 12
ENDIF PRTDC1 13
IF (INAME(1).EQ.LEFTHS) WRITE (IOUT,2000) PRTDC1 14
  WRITE(IOUT,3000) I,INAME,IADD,NDIM1,NDIM2,NDIM3,IPR PRTDC1 15
  PRTDC1 16
  PRTDC1 17
  PRTDC1 18
  PRTDC1 19
  PRTDC1 20
00 FORMAT(14X,'*****',7X,'BEGIN ELEMENT GROUP NUMBER',I5//') PRTDC1 21
00 FORMAT(14X,'*****',7X,'END ELEMENT GROUP DATA',/,') PRTDC1 22
00 FORMAT(14X,I5,7X,2A4,1X,6I10) PRTDC1 23
END PRTDC1 23
***** SUBROUTINE PRTS2D(XINT,STRESS,PSTRS,STRAIN,PSTRN, PRTS2D 1
  & NN,NNTOT,NEG,NEL,LINT) PRTS2D 2
PRTS2D 3
.. PROGRAM TO PRINT STRESS, STRAIN, AND PRINCIPAL VALUES PRTS2D 4
  FOR TWO-DIMENSIONAL CONTINUUM ELEMENTS PRTS2D 5
IMPLICIT DOUBLE PRECISION (A-H,O-Z) PRTS2D 6
.. DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION PRTS2D 7
DIMENSION XINT(2),STRESS(4),PSTRS(4),S,RAIN(4),PSTRN(4) PRTS2D 8
COMMON /IOUNIT/ IN,IOUT,IRSI,IRSOUT PRTS2D 9
PRTS2D 10
NN = NN+1 PRTS2D 11
IF (MOD(NN,NNTOT).EQ.1) WRITE(IOUT,1000) NEG PRTS2D 12
WRITE(IOUT,2000) NEL,LINT,XINT,STRESS,PSTRS,STRAIN,PSTRN PRTS2D 13
PRTS2D 14
PRTS2D 15
PRTS2D 16
PRTS2D 17
PRTS2D 18
PRTS2D 19
PRTS2D 20
)0 FORMAT('1', PRTS2D 21
  & ELEMENT STRESSES AND STRAINS :,//5X, PRTS2D 22
  & ELEMENT GROUP NUMBER . . . . . (NEG ) = :,I5/// PRTS2D 23
  & ELEMENT INT. PT. X1 . . . . . X2 . . . . ,5X, PRTS2D 24
  & STRESS STRESS STRESS STRESS :, /,5X, PRTS2D 25
  & PRINC. PRINC. SHEAR STRESS :, /,5X, PRTS2D 26
  & NUMBER NUMBER :, /,5X, PRTS2D 27
  & 11 22 12 33 :, /,49X, PRTS2D 28
  & STRESS 1 STRESS 2 STRESS ANGLE :, /,49X, PRTS2D 29
  & STRAIN STRAIN STRAIN STRAIN :, /,49X, PRTS2D 30
  & PRINC. PRINC. SHEAR STRAIN :, /,49X, PRTS2D 31
  & 11 22 12 33 :, /,49X, PRTS2D 32
  & STRAIN 1 STRAIN 2 STRAIN ANGLE :, /,49X, PRTS2D 33
)0 FORMAT(2X,I5,6X,I2,8X,2(1PE10.2),5X,8(1PE10.2)/48X,8(1PE10.2)) PRTS2D 34
END PRTS2D 34
***** FUNCTION RCDOT(A,B,MA,N) RCDOT 1
RCDOT 2
.. PROGRAM TO COMPUTE THE DOT PRODUCT OF A VECTOR STORED ROW-WISE RCDOT 3
  WITH A VECTOR STORED COLUMN-WISE RCDOT 4
IMPLICIT DOUBLE PRECISION (A-H,O-Z) RCDOT 5
.. DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION RCDOT 6
DIMENSION A(MA,1),B(1) RCDOT 7
COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE RCDOT 8
RCDOT 9
RCDOT = ZERO RCDOT 10
DO 100 I=1,N RCDOT 11
  RCDOT = RCDOT + A(1,I)*B(I) RCDOT 12
)0 CONTINUE RCDOT 13
RCDOT 14
RETURN RCDOT 15
END RCDOT 16
***** FUNCTION ROWDOT(A,B,MA,MB,N) ROWDOT 1
ROWDOT 2
.. PROGRAM TO COMPUTE THE DOT PRODUCT OF VECTORS STORED ROW-WISE ROWDOT 3
IMPLICIT DOUBLE PRECISION (A-H,O-Z) ROWDOT 4
.. DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION ROWDOT 5
DIMENSION A(MA,1),B(MB,1) ROWDOT 6
COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE ROWDOT 7
ROWDOT = ZERO ROWDOT 8
DO 100 I=1,N ROWDOT 9
  ROWDOT = ROWDOT + A(1,I)*B(1,I) ROWDOT 10
)0 CONTINUE ROWDOT 11
ROWDOT 12
ROWDOT 13
ROWDOT 14
ROWDOT 15
ROWDOT 16

```

```

C      RETURN
C      END
***** SUBROUTINE RSIN(D,V,A,NDOF,NUMNP,NTSTEP,TIME)
C
C..... PROGRAM TO READ RESTART FILE
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C..... DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION
C
C      DIMENSION D(NDOF,1),V(NDOF,1),A(NDOF,1)
C      COMMON /IOUNIT/ IIN,IOUT,IR$IN,IR$OUT
C
C      READ(IR$IN,1000) NTSTEP,TIME,IJUNK
C
C      DO 100 J=1,NUMNP
C      READ(IR$IN,2000) (D(I,J),V(I,J),A(I,J),I=1,NDOF)
100   CONTINUE
C
C      RETURN
C
1000  FORMAT(//,15X,I5,/,15X,E12.5/,I1)
2000  FORMAT(3(6E16.8/))
END
***** SUBROUTINE RSOUT(D,V,A,NDOF,NUMNP,NTSTEP,TIME)
C
C..... PROGRAM TO WRITE RESTART FILE
C
C      DOUBLE PRECISION A,D,V,TIME
C
C..... DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION
C
C      CHARACTER*4 TITLE
C      DIMENSION D(NDOF,1),V(NDOF,1),A(NDOF,1)
C      COMMON /IOUNIT/ IIN,IOUT,IR$IN,IR$OUT
C      COMMON /TITLEC/ TITLE(20)
C
C      WRITE(IR$OUT,1000) TITLE,NTSTEP,TIME
C
C      DO 100 J=1,NUMNP
C      WRITE(IR$OUT,2000) (D(I,J),V(I,J),A(I,J),I=1,NDOF)
100   CONTINUE
C
C      RETURN
C
1000  FORMAT(' ',20A4//,' STEP NUMBER = ',I5/' ',TIME = ',1PE12.5/' ')
2000  FORMAT(3(6E16.8/))
END
***** SUBROUTINE SERRQ(NAME,I)
C
C..... PROGRAM TO PRINT ERROR MESSAGE IF AVAILABLE STORAGE IS EXCEEDED
C
C      DIMENSION NAME(2)
C      COMMON /IOUNIT/ IIN,IOUT,IR$IN,IR$OUT
C
C      CALL PRTDC
C      WRITE(IOUT,1000) I,NAME
C      STOP
C
1000  FORMAT(1X,5('*'),'STORAGE EXCEEDED BY ',I10,
& ' WORDS IN ATTEMPTING TO STORE ARRAY ',2A4)
END
***** SUBROUTINE SETUPD(C,DMAT,CONST,NSTR,NROWB)
C
C..... PROGRAM TO CALCULATE THE D MATRIX
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C..... DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION
C
C      DIMENSION C(NROWB,1),DMAT(NROWB,1)
C
C      DO 200 J=1,NSTR
C
C          DO 100 I=1,J
C              DMAT(I,J) = CONST*C(I,J)
C              DMAT(J,I) = DMAT(I,J)
100      CONTINUE
C
C          200 CONTINUE
C
C      RETURN
C

```

```
*****
SUBROUTINE SHIST(ISHIST,NSOUT,NTYPE) SHIST 1
. PROGRAM TO READ, WRITE AND STORE ELEMENT TIME-HISTORY INPUT DATA SHIST 2
. DIMENSION ISHIST(3,1) SHIST 3
COMMON /IOUNIT/ IIN,IOUT,IRSIN,IRSOUP SHIST 5
COMMON /LABELS/ LABELD(3),LABEL1(16),LABEL2(3) SHIST 6
DO 100 N=1, NSOUT SHIST 7
IF (MOD(N,50).EQ.1) WRITE(IOUT,1000) NSOUT SHIST 8
READ(IIN,2000) NEL,INTPT,NCOMP SHIST 9
IF (INTPT.EQ.0) INTPT = 1 SHIST 10
IF (NTYPE.EQ.1) WRITE(IOUT,3000) NEL,INTPT,LABEL1(NCOMP) SHIST 11
IF (NTYPE.EQ.2) WRITE(IOUT,3000) NEL,INTPT,LABEL2(NCOMP) SHIST 12
. ADD IF/WRITE STATEMENTS AS ABOVE FOR ADDITIONAL ELEMENT TYPES SHIST 13
ISHIST(1,N) = NEL SHIST 14
ISHIST(2,N) = INTPT SHIST 15
ISHIST(3,N) = NCOMP SHIST 16
0 CONTINUE SHIST 17
      RETURN SHIST 18
0 FORMAT('1' SHIST 19
&' E L E M E N T   T I M E   H I S T O R Y ', SHIST 20
&' I N F O R M A T I O N ',//5X, SHIST 21
&' N U M B E R   O F   S T R E S S / S T R A I N   T I M E   H I S T O R I E S   . . . ( N S O U T ) = ',I5/// SHIST 22
&5X,' E L E M E N T   I H T P T   C O M P O N E N T ',// SHIST 23
&5X,' N U M B E R   N U M B E R ',// SHIST 24
0 FORMAT(3I5) SHIST 25
0 FORMAT(7X,I5,5X,I5,7X,1A4) SHIST 26
END SHIST 27
*****
SUBROUTINE SMULT(A,B,C,MB,MC,M,N,IOPT) SMULT 1
. PROGRAM TO PERFORM SCALAR MULTIPLICATION OF A MATRIX SMULT 2
      C(I,J) = A*B(I,J) SMULT 3
      IMPLICIT DOUBLE PRECISION (A-H,O-Z) SMULT 4
. DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION SMULT 5
DIMENSION B(MB,1),C(MC,1) SMULT 6
GO TO (1000,2000,3000),IOPT SMULT 7
. IOPT = 1, MULTIPLY ENTIRE MATRIX SMULT 8
) DO 1200 J=1,N SMULT 9
      DO 1100 I=1,M SMULT 10
      C(I,J) = A*B(I,J) SMULT 11
) CONTINUE SMULT 12
) CONTINUE SMULT 13
      RETURN SMULT 14
. IOPT = 2, MULTIPLY LOWER TRIANGULAR AND DIAGONAL ELEMENTS SMULT 15
) DO 2200 J=1,N SMULT 16
      DO 2100 I=J,M SMULT 17
      C(I,J) = A*B(I,J) SMULT 18
) CONTINUE SMULT 19
) CONTINUE SMULT 20
      RETURN SMULT 21
. IOPT = 3, MULTIPLY UPPER TRIANGULAR AND DIAGONAL ELEMENTS SMULT 22
) DO 3200 J=1,N SMULT 23
      DO 3100 I=1,J SMULT 24
      C(I,J) = A*B(I,J) SMULT 25
) CONTINUE SMULT 26
) CONTINUE SMULT 27
      RETURN SMULT 28
END SMULT 29
*****
SUBROUTINE STATIN(NEQ) STATIN 1
. PROGRAM TO SET MEMORY POINTERS FOR STATIC ANALYSIS DATA ARRAYS, STATIN 2
AND CALL ASSOCIATED INPUT ROUTINES STATIN 3
STATIN 4
```

```

C      DOUBLE PRECISION ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE      STATIN
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION      STATIN
C
C      LOGICAL LDYN      STATIN
C      COMMON /BPOINT/ MFIRST,MLAST,MTOT,IPREC      STATIN
C      COMMON /CONSTS/ ZERO,P1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE      STATIN
C      COMMON /DPOINT/ MPSTEPM,MPDPRT,MPSPRT,MPHPLT,MPITER,MPALPH,MPBETA,      STATIN
C      & MPGAMM,MPDT,MPIDHS,MPDOUT,MPVPRD,MPDPRD,MPA,MPV      STATIN
C      & IEXEC,IACODE,LDYN,IREADR,IWRITR,IPRTIN,IRANK,      STATIN
C      & NUMSEQ,NDOUT,NSD,NUMNP,NDOF,NLVECT,NLTFTN,NPTSFL,      STATIN
C      & NUMEG      STATIN
C      COMMON /SPOINT/ MPD,MPX,MPID,MPF,MPG,MPG1,MPDIAG,MPNGRP,      STATIN
C      & MPALHS,MPBRHS      STATIN
C      COMMON A(1)      STATIN
C
C      MPD = MPPOINT('D      ,NDOF ,NUMNP ,0,IPREC)      STATIN
C      IF (.NOT.LDYN) MPDPRD = MPD      ,NDOF ,NUMNP ,0 ,IPREC)      STATIN
C      MPX = MPPOINT('X      ,NSD ,NUMNP ,0 ,IPREC)      STATIN
C      MPID = MPPOINT('ID      ,NDOF ,NUMNP ,0 ,1)      STATIN
C
C      IF (NLVECT.EQ.0) THEN      STATIN
C          MPF = 1      STATIN
C      ELSE      STATIN
C          MPF = MPPOINT('F      ,NDOF ,NUMNP ,NLVECT,IPREC)      STATIN
C      ENDIF      STATIN
C
C      IF (NLTFTN.EQ.0) THEN      STATIN
C          MPG = 1      STATIN
C          MPG1 = 1      STATIN
C      ELSE      STATIN
C          MPG = MPPOINT('G      ,NPTSFL,2      ,NLTFTN,IPREC)      STATIN
C          MPG1 = MPPOINT('G1      ,NLTFTN,0      ,0      ,IPREC)      STATIN
C      ENDIF      STATIN
C
C.... INPUT COORDINATE DATA      STATIN
C      CALL COORD(A(MPX),NSD,NUMNP,IPRTIN)      STATIN
C
C.... INPUT BOUNDARY CONDITION DATA AND ESTABLISH EQUATION NUMBERS      STATIN
C      CALL BC(A(MPID),NDOF,NUMNP,NEQ,IPRTIN)      STATIN
C
C.... INPUT NODAL FORCE AND PRESCRIBED KINEMATIC BOUNDARY-VALUE DATA      STATIN
C
C      IF (NLVECT.GT.0) CALL INPUT(A(MPF),NDOF,NUMNP,0,NLVECT,      STATIN
C          & IPRTIN,ZERO)      STATIN
C
C.... INPUT LOAD-TIME FUNCTIONS      STATIN
C      IF (NLTFTN.GT.0) CALL LTIMEF(A(MPG),NPTSFL,NLTFTN,IPRTIN)      STATIN
C
C.... ALLOCATE MEMORY FOR IDIAG ARRAY AND CLEAR      STATIN
C
C      MPDIAG = MPPOINT('IDIAG      ,NEQ ,0      ,0      ,1)      STATIN
C      CALL ICLEAR(A(MPDIAG),NEQ)      STATIN
C
C      MPNGRP = MPPOINT('NGRP      ,NUMEG ,0      ,0      ,1)      STATIN
C
C      RETURN      STATIN
C      END      STATIN
*****
***** SUBROUTINE STORED(IDHIST,D,V,A,DOUT,NDOF,NDOUT)      STORED
C
C.... PROGRAM TO STORE NODAL TIME HISTORIES AS SINGLE-PRECISION DATA      STORED
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)      STORED
C
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION      STORED
C
C      REAL DOUT(NDOUT+1,1)      STORED
C      DIMENSION IDHIST(3,1),D(NDOF,1),V(NDOF,1),A(NDOF,1)      STORED
C      COMMON /HPLOTC/ NPLPTS,LOCPLT,TIME      STORED
C
C      DOUT(1,LOCPLT) = REAL(TIME)      STORED
C
C      DO 100 I=1,NDOUT      STORED
C      NODE = IDHIST(1,I)      STORED
C      IDOF = IDHIST(2,I)      STORED
C      IDVA = IDHIST(3,I)      STORED
C      IF (IDVA .EQ. 1) DOUT(I+1,LOCPLT) = REAL(D(IDOF,NODE))      STORED
C      IF (IDVA .EQ. 2) DOUT(I+1,LOCPLT) = REAL(V(IDOF,NODE))      STORED
C      IF (IDVA .EQ. 3) DOUT(I+1,LOCPLT) = REAL(A(IDOF,NODE))      STORED
C 100 CONTINUE      STORED
C
C      RETURN      STORED
C      END      STORED

```

```
*****
SUBROUTINE TIMCON(NSQ,NSTEP,NDPRT,NSPRT,NHPLT,NITER,
&           NSTEP1,NDPRT1,NSPRT1,NHPLT1,NITER1, ~
&           ALPHA,BETA,GAMMA,DT)           TIMCON 1
&           ALPHA(1),BETA(1),GAMMA(1),DT1)   TIMCON 2
&           ALPHA1,BETA1,GAMMA1,DT1)        TIMCON 3
&           ALPHA ,BETA ,GAMMA ,DT)        TIMCON 4
. PROGRAM TO COMPUTE CURRENT TIME SEQUENCE PARAMETERS
AND TIME-INTEGRATION COEFFICIENTS          TIMCON 5
IMPLICIT DOUBLE PRECISION (A-H,O-Z)        TIMCON 6
DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION TIMCON 7
DIMENSION NSTEP(1),NDPRT(1),NSPRT(1),NHPLT(1),NITER(1),
&           ALPHA(1),BETA(1),GAMMA(1),DT1)   TIMCON 8
& COMMON /COEFFS/ COEFF1,COEFF2,COEFF3,COEFF4,COEFF5,COEFF6,
& COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE TIMCON 9
NSTEP1 = NSTEP(NSQ)                      TIMCON 10
NDPRT1 = NDPRT(NSQ)                     TIMCON 11
NSPRT1 = NSPRT(NSQ)                     TIMCON 12
NHPLT1 = NHPLT(NSQ)                     TIMCON 13
NITER1 = NITER(NSQ)                     TIMCON 14
ALPHA1 = ALPHA(NSQ)                     TIMCON 15
BETA1 = BETA(NSQ)                       TIMCON 16
GAMMA1 = GAMMA(NSQ)                     TIMCON 17
DT1 = DT(NSQ)                          TIMCON 18
COEFF1 = ONE + ALPHA1                  TIMCON 19
COEFF2 = GAMMA1*DT1                   TIMCON 20
COEFF3 = BETA1*DT1*DT1                TIMCON 21
COEFF4 = COEFF1*COEFF2                TIMCON 22
COEFF5 = COEFF1*COEFF3                TIMCON 23
COEFF6 = COEFF1*DT1                  TIMCON 24
COEFF7 = PT5*COEFF1*(ONE - TWO*BETA1)*DT1*DT1 TIMCON 25
COEFF8 = COEFF1*(ONE - GAMMA1)*DT1    TIMCON 26
RETURN
END
*****
SUBROUTINE TIMING(TIME)
PROGRAM TO DETERMINE ELAPSED CPU TIME
**** THIS IS A SYSTEM-DEPENDENT ROUTINE ****
NOTE: CAN ONLY ACCESS CLOCK TIME ON VAX/VMS
TIME = SECNDS(0.0)
RETURN
END
*****
SUBROUTINE TIMLOG
PROGRAM TO PRINT LOG OF EXECUTION TIMES
CHARACTER*4 TITLE
COMMON /ETIMEC/ ETIME(7)
COMMON /IOUNIT/ IIN,IOUT,IR$IN,IR$OUT
COMMON /TITLEC/ TITLE(20)

SUBTOT = 0.0
DO 100 I=3,7
SUBTOT = SUBTOT + ETIME(I)
CONTINUE
WRITE(IOUT,1000) TITLE,ETIME,SUBTOT
RETURN
FORMAT('1',20A4//5X,
& EXECUTION TIMING INFORMATION //5X,TIMLOG 19
& INITIALIZATION PHASE      = ,1PE10.3//5X,TIMLOG 20
& SOLUTION PHASE           = ,1PE10.3//5X,TIMLOG 21
& FORMATION OF LEFT-HAND-SIDE MATRICES = ,1PE10.3//5X,TIMLOG 22
& FACTORIZATIONS           = ,1PE10.3//5X,TIMLOG 23
& FORMATION OF RIGHT-HAND-SIDE VECTORS = ,1PE10.3//5X,TIMLOG 24
& FORWARD REDUCTIONS/BACK SUBSTITUTIONS = ,1PE10.3//5X,TIMLOG 25
& CALCULATION OF ELEMENT OUTPUT     = ,1PE10.3//5X,TIMLOG 26
& SUBTOTAL                  = ,1PE10.3//5X,TIMLOG 27
& -SUBTOTAL                  = ,1PE10.3//5X,TIMLOG 28
END
*****
SUBROUTINE TSEQ
PROGRAM TO SET MEMORY POINTERS FOR TIME SEQUENCE AND
NODAL TIME HISTORY DATA ARRAYS
TSEQ
TSEQ
TSEQ
TSEQ

```

```

C      LOGICAL LDYN          TSEQ
COMMON /BPOINT/ MFIRST,MLAST,MTOT,IPREC   TSEQ
COMMON /DPOINT/ MPSTEP,MPDPRT,MPSRT,MPHPLT,MPITER,MPALPH,MPBETA,  TSEQ
&           MPGAMM,MPDT ,MPIDHS,MPDOUT,MPVPRD,MPDPRD,MPA,MPV   TSEQ
&           COMMON /HPLOTC/ NPLPTS,LOCPLT,TIME   TSEQ
&           COMMON /INFO  / IEXEC,IACODE,LDYN,IREADR,IWRITR,IPRTIN,IRANK,  TSEQ
&           NUMSEQ,NDOUT,NSD,NUMNP,NDOF,NLVECT,NLTFTN,NPTSLF,  TSEQ
&           NUMEG   TSEQ
C      COMMON A(1)          TSEQ
C      MPSTEP = MPOINT('NSTEP    ',NUMSEQ,0,0,1)          TSEQ
MPDPRT = MPOINT('NDPRT    ',NUMSEQ,0,0,1)          TSEQ
MPSRT = MPOINT('NSPRT    ',NUMSEQ,0,0,1)          TSEQ
MPHPLT = MPOINT('NHPLT    ',NUMSEQ,0,0,1)          TSEQ
MPITER = MPOINT('NITER    ',NUMSEQ,0,0,1)          TSEQ
MPALPH = MPOINT('ALPHA    ',NUMSEQ,0,0,IPREC)      TSEQ
MPBETA = MPOINT('BETA     ',NUMSEQ,0,0,IPREC)      TSEQ
MPGAMM = MPOINT('GAMMA    ',NUMSEQ,0,0,IPREC)      TSEQ
MPDT  = MPOINT('DT       ',NUMSEQ,0,0,IPREC)      TSEQ
C      CALL TSEQIN(A(MPSTEP),A(MPDPRT),A(MPSRT),A(MPHPLT),  TSEQ
&           A(MPITER),A(MPALPH),A(MPBETA),A(MPGAMM),  TSEQ
&           A(MPDT ),NUMSEQ,NPLPTS,LDYN)          TSEQ
C      IF (NDOUT.EQ.0) THEN          TSEQ
  MPIDHS = 1          TSEQ
  MPDOUT = 1          TSEQ
ELSE          TSEQ
  MPIDHS = MPOINT('IDHIST  ',3,NDOUT,0,1)          TSEQ
  MPDOUT = MPOINT('DOUT    ',NDOUT+1,NPLPTS,0,1)          TSEQ
ENDIF          TSEQ
C      RETURN          TSEQ
END          TSEQ
*****SUBROUTINE TSEQIN(NSTEP,NDPRT,NSPRT,NHPLT,NITER,ALPHA,BETA,  TSEQIN
&           GAMMA,DT,NUMSEQ,NPLPTS,LDYN)          TSEQIN
C.... PROGRAM TO READ, WRITE AND STORE TIME SEQUENCE DATA  TSEQIN
C.... NOTE: "NPLPTS" IS PASSED TO SUBROUTINE HPLOT BY WAY OF  TSEQIN
C           COMMON /HPLOTC/          TSEQIN
C           IMPLICIT DOUBLE PRECISION (A-H,O-Z)          TSEQIN
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION  TSEQIN 1
C           LOGICAL LDYN          TSEQIN
DIMENSION NSTEP(1),NDPRT(1),NSPRT(1),NHPLT(1),NITER(1)  TSEQIN 1
&           ALPHA(1),BETA(1),GAMMA(1),DT(1)          TSEQIN 1
COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE  TSEQIN 1
COMMON /IOUNIT/ IIN,IOUT,IR$IN,IRSOUT          TSEQIN 1
C           NPLPTS = 1          TSEQIN 1
C           DO 100 I=1,NUMSEQ          TSEQIN 1
  READ(IIN,1000) N,NSTEP(N),NDPRT(N),NSPRT(N),NHPLT(N),NITER(N),  TSEQIN 1
&           ALPHA(N),BETA(N),GAMMA(N),DT(N)          TSEQIN 1
  IF (NHPLT(N).GT.0) NPLPTS = NPLPTS + NSTEP(N)/NHPLT(N)  TSEQIN 1
100 CONTINUE          TSEQIN 1
C.... SET DEFAULT SEQUENCE PARAMETERS FOR STATIC ANALYSIS  TSEQIN
C           IF (.NOT.LDYN) THEN          TSEQIN
  DO 200 I=1,NUMSEQ          TSEQIN
  NSTEP(I) = MAX0(1,NSTEP(I))          TSEQIN
  NDPRT(I) = 1          TSEQIN
  NSPRT(I) = 1          TSEQIN
  NHPLT(I) = 0          TSEQIN
  NITER(I) = 1          TSEQIN
  ALPHA(I) = ZERO          TSEQIN
  BETA(I) = ONE          TSEQIN
  GAMMA(I) = ZERO          TSEQIN
  DT(I) = ONE          TSEQIN
200 CONTINUE          TSEQIN 4
C           ENDIF          TSEQIN 4
C           DO 300 N=1,NUMSEQ          TSEQIN 4
  IF ((MOD(N,2).EQ.1)) WRITE(IOUT,2000) NUMSEQ          TSEQIN 4
  WRITE(IOUT,3000) N,NSTEP(N),NDPRT(N),NSPRT(N),NHPLT(N),NITER(N),  TSEQIN 4
&           ALPHA(N),BETA(N),GAMMA(N),DT(N)          TSEQIN 4
300 CONTINUE          TSEQIN 4
C           RETURN          TSEQIN 4

```

```

000 FORMAT(6I5,4F10.0) TSEQIN 50
000 FORMAT(1X, TIME SEQUENCE DATA , ',` I5//5X, TSEQIN 51
& NUMBER OF TIME SEQUENCES . . . . (NUMSEQ ) = ,` I5//5X, TSEQIN 52
000 FORMAT(5X, TSEQIN 53
& TIME SEQUENCE NUMBER . . . . (N ) = ,` I5//5X, TSEQIN 54
& NUMBER OF TIME STEPS . . . . (NSTEP(N)) = ,` I5//5X, TSEQIN 55
& KINEMATIC PRINT INCREMENT . . . . (NDPRT(N)) = ,` I5//5X, TSEQIN 56
& STRESS/STRAIN PRINT INCREMENT . . . . (NSPRT(N)) = ,` I5//5X, TSEQIN 57
& TIME HISTORY PLOT INCREMENT . . . . (NHPLT(N)) = ,` I5//5X, TSEQIN 58
& NUMBER OF ITERATIONS . . . . (NITER(N)) = ,` I5//5X, TSEQIN 59
& FIRST INTEGRATION PARAMÉTÉR . . . . (ALPHA(N)) = ,` 1PE12.5//5X, TSEQIN 60
& SECOND INTEGRATION PARAMETER . . . . (BETA(N)) = ,` 1PE12.5//5X, TSEQIN 61
& THIRD INTEGRATION PARAMETER . . . . (GAMMA(N)) = ,` 1PE12.5//5X, TSEQIN 62
& TIME STEP . . . . (DT(N)) = ,` 1PE12.5//5X, TSEQIN 63
                                         TSEQIN 64
                                         TSEQIN 65
END TSEQIN 66
*****SUBROUTINE ZTEST(A,N,LZERO) ZTEST 1
... PROGRAM TO DETERMINE IF AN ARRAY CONTAINS ONLY ZERO ENTRIES ZTEST 2
IMPLICIT DOUBLE PRECISION (A-H,O-Z) ZTEST 4
... DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION ZTEST 6
DIMENSION A(1) ZTEST 8
COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE ZTEST 9
LOGICAL LZERO ZTEST 10
LZERO = .TRUE. ZTEST 11
DO 100 I=1,N ZTEST 12
IF (A(I).NE.ZERO) THEN ZTEST 13
  LZERO = .FALSE. ZTEST 14
  RETURN ZTEST 15
ENDIF ZTEST 16
100 CONTINUE ZTEST 17
RETURN ZTEST 18
END ZTEST 19
*****SUBROUTINE QUADCC(ITASK,NPAR,MP,NEG) QUADC 1
... PROGRAM TO SET STORAGE AND CALL TASKS FOR THE QUADC 2
FOUR-NODE QUADRILATERAL, ELASTIC CONTINUUM ELEMENT QUADC 3
DOUBLE PRECISION TIME QUADC 4
... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION QUADC 5
LOGICAL LDYN QUADC 6
DIMENSION NPAR(1),MP(1) QUADC 7
COMMON /BPOINT/ MFIRST,MLAST,MTOT,IPREC QUADC 8
COMMON /DPOINT/ MPSTEP,MPDPRT,MPSVRT,MPHPLT,MPITER,MPALPH,MPBETA, QUADC 9
& MPGAMM,MPDT,MPIDHS,MPDOUT,MPVPRD,MPDPRD,MPA,MPV QUADC 10
& COMMON /HPLOTC/ NPLPTS,LOCPLT,TIME QUADC 11
COMMON /INFO / IEXEC,IACODE,LDYN,IREADR,IWRITR,IPRTIN,IRANK, QUADC 12
& NUMSEQ,NDOUT,NSD,NUMNP,NDOF,NLVECT,NLTFTN,NPTSFL, QUADC 13
& NUMEG QUADC 14
& COMMON /SPOINT/ MPD,MPX,MPID,MPF,MPG,MPG1,MPDIAG,MPNGRP, QUADC 15
& MPALHS,MPBRHS QUADC 16
COMMON A(1) QUADC 17
                                         QUADC 18
                                         QUADC 19
                                         QUADC 20
                                         QUADC 21

```

```

C
MW      = 1          QUADC 2
MDET    = 2          QUADC 2
MR      = 3          QUADC 2
MSHL    = 4          QUADC 2
MSHG    = 5          QUADC 2
MSHGBR  = 6          QUADC 2
MRHO    = 7          QUADC 2
MRDPM   = 8          QUADC 2
MRDPK   = 9          QUADC 2
MTH     = 10         QUADC 2
MC      = 11         QUADC 2
MGRAV   = 12         QUADC 2
MIEN    = 13         QUADC 2
MMAT    = 14         QUADC 2
MLM     = 15         QUADC 2
MIELNO  = 16         QUADC 2
MISIDE  = 17         QUADC 2
MPRESS  = 18         QUADC 2
MSHEAR  = 19         QUADC 2
MISHST  = 20         QUADC 2
MSOUT   = 21         QUADC 2
MELEFM  = 22         QUADC 2
MXL     = 23         QUADC 2
MWORK   = 24         QUADC 2
MB      = 25         QUADC 2
MDMAT   = 26         QUADC 2
MDB     = 27         QUADC 2
MVL     = 28         QUADC 2
MAL     = 29         QUADC 2
MELRES  = 30         QUADC 2
MDL     = 31         QUADC 2
MSTRN   = 32         QUADC 2
MSTRS   = 33         QUADC 2
MPSTRN  = 34         QUADC 2
MPSTRS  = 35         QUADC 2
C
NTYPE   = NPAR( 1 )  QUADC 5
NUMEL   = NPAR( 2 )  QUADC 5
NUMAT   = NPAR( 3 )  QUADC 5
NSURF   = NPAR( 4 )  QUADC 6
NSOUT   = NPAR( 5 )  QUADC 6
IOPT    = NPAR( 6 )  QUADC 6
ISTPRT  = NPAR( 7 )  QUADC 6
LFSURF  = NPAR( 8 )  QUADC 6
LFBODY  = NPAR( 9 )  QUADC 6
NICODE  = NPAR(10)  QUADC 6
IBBAR   = NPAR(11)  QUADC 6
IMASS   = NPAR(12)  QUADC 7
IMPEXP  = NPAR(13)  QUADC 7
C.... SET ELEMENT PARAMETERS
C
NEN     = 4          QUADC 7
NED     = 2          QUADC 7
NEE     = NEN*NED    QUADC 7
NESD    = 2          QUADC 7
NROWSH  = 3          QUADC 7
NEEQS   = NEE*NEE    QUADC 8
NRQWB   = 4          QUADC 8
NSTR    = 3          QUADC 8
IF ( (IOPT.EQ.2) .OR. (IBBAR.EQ.1) ) NSTR = 4  QUADC 8
NINT    = 1          QUADC 8
IF (NICODE.EQ.0) NINT = 4  QUADC 8
NRINT   = 1          QUADC 8
IF (ITASK.EQ.1) THEN QUADC 8
C..... SET MEMORY POINTERS
CCC
C
NOTE: THE MP ARRAY IS STORED DIRECTLY AFTER THE NPAR ARRAY,
      BEGINNING AT LOCATION MPNPAR + 16 OF BLANK COMMON. QUADC 9
      THE VARIABLE "JUNK" IS NOT USED SUBSEQUENTLY. QUADC 9
C
JUNK     = MPOINT('MP      ',35      ,0      ,0      ,1)  QUADC 9

```

```

MP(MW) = MPOINT('W',NINT,0,0,IPREC) QUAD C 97
MP(MDET) = MPOINT('DET',NINT,0,0,IPREC) QUAD C 98
MP(MR) = MPOINT('R',NINT,0,0,IPREC) QUAD C 99
MP(MSHL) = MPOINT('SHL',NROWSH,NEN,NINT,IPREC) QUAD C 100
MP(MSHG) = MPOINT('SHG',NROWSH,NEN,NINT,IPREC) QUAD C 101
MP(MSHGBR) = MPOINT('SHGBAR',NROWSH,NEN,NINT,IPREC) QUAD C 102
MP(MRH0) = MPOINT('RHO',NUMAT,0,0,IPREC) QUAD C 104
MP(MRDPM) = MPOINT('RDAMP',NUMAT,0,0,IPREC) QUAD C 105
MP(MRDPK) = MPOINT('RDAMPK',NUMAT,0,0,IPREC) QUAD C 106
MP(MTH) = MPOINT('TH',NUMAT,0,0,IPREC) QUAD C 107
MP(MC) = MPOINT('C',NROWB,NROWB,NUMAT,IPREC) QUAD C 108
MP(MGRAV) = MPOINT('GRAV',NESD,0,0,IPREC) QUAD C 109
MP(MIEN) = MPOINT('IEN',NEN,NUMEL,1), QUAD C 110
MP(MMAT) = MPOINT('MAT',NUMEL,0,0,1), QUAD C 111
MP(MLM) = MPOINT('LM',NED,NEN,NUMEL,1), QUAD C 112
MP(MIELNO) = MPOINT('IELNO',NSURF,0,0,1), QUAD C 113
MP(MISIDE) = MPOINT('ISIDE',NSURF,0,0,1), QUAD C 114
MP(MPRESS) = MPOINT('PRESS',2,NSURF,0,IPREC) QUAD C 115
MP(MSHEAR) = MPOINT('SHEAR',2,NSURF,0,IPREC) QUAD C 116
IF (NSOUT.EQ.0) THEN QUAD C 117
  MP(MISHST) = JUNK QUAD C 118
  MP(MSOUT) = JUNK QUAD C 119
ELSE QUAD C 120
  MP(MISHST) = MPOINT('ISHIST',3,NSOUT,0,1), QUAD C 121
  MP(MSOUT) = MPOINT('SOUT',NSOUT+1,NPLPTS,0,1), QUAD C 122
ENDIF QUAD C 123
MP(MELEFM) = MPOINT('ELEFFM',NEE,NEE,0,IPREC) QUAD C 124
MP(MXL) = MPOINT('XL',NESD,NEN,0,IPREC) QUAD C 125
MP(MWORK) = MPOINT('WORK',16,0,0,IPREC) QUAD C 126
MP(MB) = MPOINT('B',NROWB,NEE,0,IPREC) QUAD C 127
MP(MDMAT) = MPOINT('DMAT',NROWB,NROWB,0,IPREC) QUAD C 128
MP(MDB) = MPOINT('DB',NROWB,NEE,0,IPREC) QUAD C 129
MP(MVL) = MPOINT('VL',NED,NEN,0,IPREC) QUAD C 130
MP(MAL) = MPOINT('AL',NED,NEN,0,IPREC) QUAD C 131
MP(MELRES) = MPOINT('ELRESF',NEE,0,0,IPREC) QUAD C 132
MP(MDL) = MPOINT('DL',NED,NEN,0,IPREC) QUAD C 133
MP(MSTRN) = MPOINT('STRAIN',NROWB,0,0,IPREC) QUAD C 134
MP(MSTRS) = MPOINT('STRESS',NROWB,0,0,IPREC) QUAD C 135
MP(MPSTRN) = MPOINT('PSTRN',NROWB,0,0,IPREC) QUAD C 136
MP(MPSTRS) = MPOINT('PSTRS',NROWB,0,0,IPREC) QUAD C 137
ENDIF QUAD C 138
.. TASK CALLS QUAD C 139
IF (ITASK.GT.6) RETURN QUAD C 140
GO TO (100,200,300,400,500,600),ITASK QUAD C 141
10 CONTINUE QUAD C 142
.. INPUT ELEMENT DATA ('INPUT__')
CALL QDCT1(A(MP(MSHL)),A(MP(MW)),A(MP(MRH0)), QUAD C 143
  A(MP(MRDPM)),A(MP(MRDPK)),A(MP(MTH)), QUAD C 144
  A(MP(MC)),A(MP(MGRAV)),A(MP(MIEN)), QUAD C 145
  A(MP(MMAT)),A(MPID),A(MP(MLM)), QUAD C 146
  A(MPDIAG),A(MP(MIELNO)),A(MP(MISIDE)), QUAD C 147
  A(MP(MPRESS)),A(MP(MSHEAR)),A(MP(MISHST)), QUAD C 148
  NTYPE,NUMEL,NUMAT,NSURF,NSOUT,IOPT, QUAD C 149
  ISTRPRT,LFSURF,LFBODY,NICODE,NINT,IBBAR, QUAD C 150
  & IMASS,IMPEXP,NROWSH,NROWB,NESD,NEN, QUAD C 151
  & NDOF,NED,IPRTIN,LDYN), QUAD C 152
RETURN QUAD C 153
10 CONTINUE QUAD C 154
.. FORM ELEMENT EFFECTIVE MASS AND ASSEMBLE INTO GLOBAL QUAD C 155
LEFT-HAND-SIDE MATRIX ('FORM_LHS')
CALL QDCT2(A(MP(MELEFM)),A(MP(MIEN)),A(MPX), QUAD C 156
  A(MP(MXL)),A(MP(MMAT)),A(MP(MDET)), QUAD C 157
  A(MP(MSHL)),A(MP(MSHG)),A(MP(MR)), QUAD C 158
  A(MP(MRDPM)),A(MP(MRDPK)),A(MP(MTH)), QUAD C 159
  A(MP(MRH0)),A(MP(MW)),A(MP(MWORK)), QUAD C 160
  A(MP(MSHGBR)),A(MP(MC)),A(MPALHS), QUAD C 161
  A(MP(MDMAT)),A(MP(MDB)),A(MPALHS), QUAD C 162
  A(MPDIAG),A(MP(MLM)),QUAD C 163
  IMPEXP,IMASS,NUMEL,NEESQ,NEN,NSD, QUAD C 164
  NESD,NINT,NEG,NROWSH,LDYN,NED, QUAD C 165
  IOPT,IBBAR,NROWB,NSTR,NEE), QUAD C 166
RETURN QUAD C 167
10 CONTINUE QUAD C 168
.. FORM ELEMENT RESIDUAL-FORCE VECTOR AND ASSEMBLE INTO GLOBAL QUAD C 169

```

```

C      RIGHT-HAND-SIDE VECTOR ('FORM_RHS')
C
CALL QDCT3(A(MP(MMAT)),A(MP(MIEN)),A(MPDPRD),
&           A(MP(MDL)),A(MPVPRD),A(MP(MVL)),
&           A(MPA),A(MP(MAL)),A(MP(MRDPK)),
&           A(MP(MRDPM)),A(MP(MRH0)),A(MP(MGRAV)),
&           A(MP(MELRES)),A(MPX),A(MP(MXL)),
&           A(MP(MDET)),A(MP(MSHL)),A(MP(MSHG)),
&           A(MP(MR)),A(MPG1),A(MP(MWORK)),
&           A(MP(MTH)),A(MP(MW)),A(MP(MELEM)),
&           A(MP(MSHGBR)),A(MP(MB)),A(MP(MSTRN)),
&           A(MP(MC)),A(MP(MSTRS)),A(MPBRHS),
&           A(MP(MLM)),A(MP(MIELNO)),A(MP(MISIDE)),
&           A(MP(MPRESS)),A(MP(MSHEAR)),
&           NUMEL,NED,NEN,NDOF,Ldyn,NEE
&           IMASS,NESD,LFBODY,NSD,NINT,NROWSH,
&           NEG,IOPT,NROWB,NSTR,IBBAR,NSURF,
&           LFSURF)
C      RETURN
C 400 CONTINUE
C.... CALCULATE AND PRINT ELEMENT STRESS/STRAIN OUTPUT ('STR_PRNT')
C
IF (ISTPRT.EQ.0)
& CALL QDCT4(A(MP(MMAT)),A(MP(MIEN)),A(MPD),
&           A(MP(MDL)),A(MPX),A(MP(MXL)),
&           A(MP(MDET)),A(MP(MSHL)),A(MP(MSHG)),
&           A(MP(MWORK)),A(MP(MR)),A(MP(MSHGBR)),
&           A(MP(MW)),A(MP(MB)),A(MP(MSTRN)),
&           A(MP(MC)),A(MP(MSTRS)),A(MP(MPSTRN)),
&           A(MP(MPSTRS)),
&           NINT,NUMEL,NEN,NDOF,NED,NSD,
&           NESD,NROWSH,NEG,IOPT,IBBAR,NROWB,
&           NEE,NSTR)
C      RETURN
C 500 CONTINUE
C.... CALCULATE AND STORE ELEMENT TIME-HISTORIES ('STR_STOR')
C
IF (NSOUT.GT.0)
& CALL QDCT5(A(MP(MISHST)),A(MP(MSOUT)),A(MP(MMAT)),
&           A(MP(MIEN)),A(MPD),A(MP(MDL)),
&           A(MPX),A(MP(MXL)),A(MP(MDET)),
&           A(MP(MSHL)),A(MP(MSHG)),A(MP(MR)),
&           A(MP(MSHGBR)),A(MP(MW)),A(MP(MB)),
&           A(MP(MSTRN)),A(MP(MC)),A(MP(MSTRS)),
&           A(MP(MPSTRN)),A(MP(MPSTRS)),A(MP(MWORK)),
&           NSOUT,NEN,NDOF,NED,NSD,NEED,
&           NESD,NROWSH,NINT,NEG,IOPT,IBBAR,NROWB,
&           NEE,NSTR)
C      RETURN
C 600 CONTINUE
C.... PLOT ELEMENT TIME-HISTORIES ('STR_PLOT')
C
IF (NSOUT.GT.0)
& CALL HPLOT(A(MP(MISHST)),A(MP(MSOUT)),NSOUT,3,NTYPE)
RETURN
C      END
***** SUBROUTINE QDCT1(SHL,W,RHO,RDAMPM,RDAMPK,TH
&           C,GRAV,IEN,MAT,IDL,LM
&           IDIAG,IELNO,ISIDE,PRESS,SHEAR,ISHIST,
&           NTYPE,NUMEL,NUMAT,NSURF,NSOUT,IOPT,
&           IMASS,LFSURF,LFBODY,NICODE,NINT,IBBAR,
&           IMPEXP,NROWSH,NROWB,NEED,NEN,
&           NDOF,NED,IPRTIN,Ldyn)
C.... PROGRAM TO READ, GENERATE AND WRITE DATA FOR THE
C      FOUR-NODE QUADRILATERAL, ELASTIC CONTINUUM ELEMENT
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C
LOGICAL LDYN
DIMENSION SHL(NROWSH,NEN,1),W(1),RHO(1),RDAMPM(1),RDAMPK(1),
&           TH(1),C(NROWB,NROWB,1),GRAV(NESD),IEN(NEN,1),MAT(1),
&           ID(NDOF,1),LM(NED,NEN,1),IDIAG(1),IELNO(1),ISIDE(1),
&           PRESS(2,1),SHEAR(2,1),ISHIST(3,1)
COMMON /IOUNIT/IIN,IOUT,IJSIN,IRSOUT

```



```

C LOGICAL LDYN,LDIAG,LQUAD
C DIMENSION ELEFFM(NEE,1),IEN(NEN,1),X(NSD,1),XL(NESD,1),MAT(1),
C & DET(1),SHL(NR0WSH,NEN,1),SHG(NR0WSH,NEN,1),R(1),
C & RDAMPM(1),RDAMPK(1),TH(1),RH0(1),W(1),WORK(1),
C & SHGBAR(3,1),B(NROWB,1),C(NROWB,NROWB,1),DMAT(NROWB,1),
C & DB(NROWB,1),ALHS(1),IDIA(1),LMINED(NEN,1)
C COMMON /COEFFS/ COEFF1,COEFF2,COEFF3,COEFF4,COEFF5,COEFF6,
C & COEFF7,COEFF8,ALPHA1,BETA1,GAMMA1,DT1
C COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
C
C LDIAG = .FALSE.
C IF ( (IMPEXP.EQ.1) .AND. (IMASS.EQ.1) ) LDIAG = .TRUE.
C
C DO 200 NEL=1,NUMEL
C
C CALL CLEAR(ELEFFM,NEESQ)
C CALL LOCAL(IEN(1,NEL),X,XL,NEN,NSD,NESD)
C M = MAT(NEL)
C LQUAD = .TRUE.
C IF ( IEN(3,1).EQ.IEN(4,NEL) ) LQUAD = .FALSE.
C CALL QDCSHG(XL,DET,SHL,SHG,NINT,NEL,NEG,LQUAD)
C
C IF ( IOPT.EQ.2) THEN
C
C   DO 100 L=1,NINT
C     R(L) = ROWDOT(SHG(NR0WSH,1,L),XL,NR0WSH,NESD,NEN)
C     DET(L) = DET(L)*R(L)
C 100  CONTINUE
C
C ENDIF
C
C IF ( LDYN .AND. (IMASS.NE.2) ) THEN
C
C..... FORM MASS MATRIX
C
C CONSTM = (ONE + RDAMPM(M)*COEFF4)*TH(M)*RH0(M)
C IF (CONSTM.NE.ZERO) CALL CONTM(SHG,XL,W,DET,ELEFFM,WORK,
C & CONSTM,IMASS,NINT,NR0WSH,NESD,NEN,NED,NEE,.FALSE.)
C
C ENDIF
C
C IF ( (.NOT.LDYN) .OR. (IMPEXP.EQ.0) ) THEN
C
C..... FORM STIFFNESS MATRIX
C
C CONSTK = (COEFF4*RDAMPK(M) + COEFF5)*TH(M)
C CALL QDCK(SHGBAR,W,DET,R,SHG,B,C(1,1,M),DMAT,DB,ELEFFM,CONSTK,
C & IBBAR,NEN,NINT,IOPT,NESD,NR0WSH,NROWB,NSTR,NEE)
C
C ENDIF
C
C.... ASSEMBLE ELEMENT EFFECTIVE MASS MATRIX INTO GLOBAL
C LEFT-HAND-SIDE MATRIX
C
C CALL ADDLHS(ALHS,ELEFFM,IDIAG,LM(1,1,NEL),NEE,LDIA)
C
C 200 CONTINUE
C
C RETURN
C END
C **** SUBROUTINE QDCT3(MAT ,IEN ,DPRED ,DL ,VPRED ,VL ,
C & A ,AL ,RDAMPK ,RDAMPM ,RHO ,GRAV , )
C & ELRESF,X ,XL ,DET ,SHL ,SHG , )
C & R ,G1 ,WORK ,TH ,W ,ELEFFM , )
C & SHGBAR,B ,STRAIN,C ,PRESS ,STRESS ,BRHS , )
C & LM ,IELNO ,ISIDE ,PRESS ,SHEAR , )
C & NUMEL ,NED ,NEN ,NDOF ,LDYN ,NEE , )
C & IMASS ,NESD ,LFBODY ,NSD ,NINT ,NR0WSH , )
C & NEG ,IOPT ,NROWB ,NSTR ,IBBAR ,NSURF , )
C & LFSURF )
C
C.... PROGRAM TO CALCULATE RESIDUAL-FORCE VECTOR FOR THE
C FOUR-NODE QUADRILATERAL, ELASTIC CONTINUUM ELEMENT AND
C ASSEMBLE INTO THE GLOBAL RIGHT-HAND-SIDE VECTOR
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION

```

```

LOGICAL LDYN, FORMMA, FORMKD, ZEROAL, ZERODL, ZEROG, LQUAD      QDCT3   19
DIMENSION MAT(1), IEN(NEN,1), DPRED(NDOF,1), DL(NED,1), VPRED(NDOF,1), QDCT3   20
&          VL(NED,1), A(NDOF,1), AL(NED,1), RDAMPK(1), RDAMPM(1)      QDCT3   21
&          RHO(1), GRAV(1), ELRESF(1), X(NSD,1), XL(NED,1), DET(1), QDCT3   22
&          SHL(NR0NSH,NEN,1), SHG(NR0NSH,NEN,1), R(1), G1(1), WORK(1), QDCT3   23
&          TH(1), W(1), ELEFFM(NEE,1), SHGBAR(3,1), BNROWB,1,      QDCT3   24
&          STRAIN(1), C(NR0WB,NR0WB,1), STRESS(1), BRHS(1)      QDCT3   25
&          LM(NED,NEN,1), IELNO(1), ISIDE(1), PRESS(2,1), SHEAR(2,1) QDCT3   26
8 COMMON /CONSTS/ ZERO, PT1667, PT25, PT5, ONE, TWO, THREE, FOUR, FIVE QDCT3   27
DO 600 NEL=1,NUMEL QDCT3   28
  FORMMA = .FALSE. QDCT3   29
  FORMKD = .FALSE. QDCT3   30
  M = MAT(NEL) QDCT3   31
  NOTE: FOR STATIC ANALYSIS MPDPRD = MPD, HENCE REFERENCE TO QDCT3   32
        ARRAY "DPRED" WILL ACCESS THE CONTENTS OF ARRAY "D". QDCT3   33
CALL LOCAL(IEN(1,NEL),DPRED,DL,NEN,NDOF,NED) QDCT3   34
IF (LDYN) THEN QDCT3   35
  CALL LOCAL(IEN(1,NEL),VPRED,VL,NEN,NDOF,NED) QDCT3   36
  CALL LOCAL(IEN(1,NEL),A,AL,NEN,NDOF,NED) QDCT3   37
  DO 200 J=1,NEN QDCT3   38
    DO 100 I=1,NED QDCT3   39
      DL(I,J) = DL(I,J) + RDAMPK(M)*VL(I,J) QDCT3   40
      AL(I,J) = AL(I,J) + RDAMPM(M)*VL(I,J) QDCT3   41
10    CONTINUE QDCT3   42
10    CONTINUE QDCT3   43
  CALL ZTEST(AL,NEE,ZEROAL) QDCT3   44
  IF ((.NOT.ZEROAL).AND.(IMASS.NE.2).AND.(RHO(M).NE.ZERO)) QDCT3   45
&    FORMMA = .TRUE. QDCT3   46
  ELSE QDCT3   47
    CALL CLEAR(AL,NEE) QDCT3   48
  ENDIF QDCT3   49
  CALL ZTEST(DL,NEE,ZERODL) QDCT3   50
  IF (.NOT.ZERODL) FORMKD = .TRUE. QDCT3   51
  CALL ZTEST(GRAV,NESD,ZEROG) QDCT3   52
  IF ((.NOT.ZEROG).AND.(LFBODY.NE.0).AND.(RHO(M).NE.ZERO) QDCT3   53
& .AND.(IMASS.NE.2)) THEN QDCT3   54
    FORMMA = .TRUE. QDCT3   55
    DO 400 I=1,NED QDCT3   56
      TEMP = GRAV(I)*G1(LFBODY) QDCT3   57
      DO 300 J=1,NEN QDCT3   58
        AL(I,J) = AL(I,J) - TEMP QDCT3   59
300    CONTINUE QDCT3   60
400    CONTINUE QDCT3   61
  CONTINUE QDCT3   62
  ENDIF QDCT3   63
  IF (FORMMA.OR.FORMKD) THEN QDCT3   64
    CALL CLEAR(ELRESF,NEE) QDCT3   65
    CALL LOCAL(TEN(1,NEL),X,XL,NEN,NSD,NESD) QDCT3   66
    LQUAD = .TRUE. QDCT3   67
    IF (IEN(3,NEL).EQ.IEN(4,NEL)) LQUAD = .FALSE. QDCT3   68
    CALL QDCSHG(XL,DET,SHL,SHG,NINT,NEL,NEG,LQUAD) QDCT3   69
    IF (IOPT.EQ.2) THEN QDCT3   70
      DO 500 L=1,NINT QDCT3   71
        R(L) = ROWDOT(SHG(NR0NSH,1,L),XL,NR0NSH,NESD,NEN) QDCT3   72
        DET(L) = DET(L)*R(L) QDCT3   73
500    CONTINUE QDCT3   74
    ENDIF QDCT3   75
    IF (FORMMA) THEN QDCT3   76
      ..... FORM INERTIAL AND/OR BODY FORCE QDCT3   77
      CONSTM = - TH(M)*RHO(M) QDCT3   78
      CALL CONTM(A,SHG,XL,W,DET,AL,ELEFFM,WORK,ELRESF,CONSTM,IMASS, QDCT3   79
&           NINT,NR0NSH,NESD,NEN,NED,NEE) QDCT3   80
    ENDIF QDCT3   81
    IF (FORMKD) THEN QDCT3   82
      ..... FORM INERTIAL AND/OR BODY FORCE QDCT3   83
      CONSTM = - TH(M)*RHO(M) QDCT3   84
      CALL CONTM(A,SHG,XL,W,DET,AL,ELEFFM,WORK,ELRESF,CONSTM,IMASS, QDCT3   85
&           NINT,NR0NSH,NESD,NEN,NED,NEE) QDCT3   86
    ENDIF QDCT3   87
    IF (FORMMA) THEN QDCT3   88
      ..... FORM INERTIAL AND/OR BODY FORCE QDCT3   89
      CONSTM = - TH(M)*RHO(M) QDCT3   90
      CALL CONTM(A,SHG,XL,W,DET,AL,ELEFFM,WORK,ELRESF,CONSTM,IMASS, QDCT3   91
&           NINT,NR0NSH,NESD,NEN,NED,NEE) QDCT3   92
    ENDIF QDCT3   93
    IF (FORMKD) THEN QDCT3   94
      ..... FORM INERTIAL AND/OR BODY FORCE QDCT3   95
      CONSTM = - TH(M)*RHO(M) QDCT3   96
      CALL CONTM(A,SHG,XL,W,DET,AL,ELEFFM,WORK,ELRESF,CONSTM,IMASS, QDCT3   97
&           NINT,NR0NSH,NESD,NEN,NED,NEE) QDCT3   98
    ENDIF QDCT3   99
    IF (FORMMA) THEN QDCT3  100
      ..... FORM INERTIAL AND/OR BODY FORCE QDCT3  101
      CONSTM = - TH(M)*RHO(M) QDCT3  102
      CALL CONTM(A,SHG,XL,W,DET,AL,ELEFFM,WORK,ELRESF,CONSTM,IMASS, QDCT3  103
&           NINT,NR0NSH,NESD,NEN,NED,NEE) QDCT3  104
    ENDIF QDCT3  105
    IF (FORMKD) THEN QDCT3  106
      ..... FORM INERTIAL AND/OR BODY FORCE QDCT3  107
      CONSTM = - TH(M)*RHO(M) QDCT3  108
      CALL CONTM(A,SHG,XL,W,DET,AL,ELEFFM,WORK,ELRESF,CONSTM,IMASS, QDCT3  109
&           NINT,NR0NSH,NESD,NEN,NED,NEE) QDCT3  110
    ENDIF QDCT3  111
  ENDIF QDCT3  112
END QDCT3  113

```

```

C..... FORM INTERNAL FORCE
C
C      CONSTK = - TH(M)
C      CALL QDCKD(SHGBAR,W,DET,R,SHG,B,DL,STRAIN,C(1,1,M),STRESS,
C      &           WORK,ELRESF,CONSTK,IBBAR,NEN,NINT,IOPt,NROWSH,
C      &           NESD,NROWB,NEE,NSTR)
C
C      ENDIF
C
C      CALL ADDRHS(BRHS,ELRESF,LM(1,1,NEL),NEE)
C
C      ENDIF
C
C 600 CONTINUE
C
C.... FORM SURFACE FORCE
C
C      NOTE: ASSEMBLY OF SURFACE LOADS IS PERFORMED INSIDE QDCSUf
C
C      IF ( (NSURF.GT.0) .AND. (LFSURF.GT.0) )
C      &      CALL QDCSUf(IELN0,IEN,X,XL,ISIDE,MAT,TH,PRESS,SHEAR,ELRESF,
C      &           BRHS,LM,G1(LFSURF),NSURF,NEN,NSD,NESD,NED,NEE,IOPt)
C
C      RETURN
C
C*****
C***** SUBROUTINE QDCD4(MAT ,IEN ,D ,DL ,X ,XL ,
C***** &           DET ,SHL ,SHG ,XINT ,R ,SHGBAR ,
C***** &           W ,B ,STRAIN,C ,STRESS,PSTRN ,
C***** &           PSTRS
C***** &           NINT ,NUMEL ,NEN ,NDOF ,NED ,NSD
C***** &           NESD ,NROWSH ,NEG ,IOPt ,IBBAR ,NROWB ,
C***** &           NEE ,NSTR )
C
C.... PROGRAM TO CALCULATE AND PRINT STRESS, STRAIN AND
C      PRINCIPAL VALUES FOR THE FOUR-NODE QUADRILATERAL,
C      ELASTIC CONTINUUM ELEMENT
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION
C
C      LOGICAL LQUAD
C      DIMENSION MAT(1),IEN(NEN,1),D(NDOF,1),DL(NED,1),X(NSD,1),
C      &           XL(NESD,1),DET(1),SHL(NROWSH,NEN,1),SHG(NROWSH,NEN,1),
C      &           XINT(NESD,1),R(1),SHGBAR(3,1),W(1),B(NROWB,1),STRAIN(1),
C      &           C(NROWB,NROWB,1),STRESS(1),PSTRN(1),PSTRS(1)
C      COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
C
C      NNNTOT = 16
C      NN = 0
C
C      DO 300 NEL=1,NUMEL
C
C      M = MAT(NEL)
C      CALL LOCAL(IEN(1,NEL),D,DL,NEN,NDOF,NED)
C      CALL LOCAL(IEN(1,NEL),X,XL,NEN,NSD,NESD)
C      LQUAD = TRUE
C      IF (IEN(3,NEL).EQ.IEN(4,NEL)) LQUAD = .FALSE.
C      CALL QDCSHG(XL,DET,SHL,SHG,NINT,NEL,NEG,LQUAD)
C
C.... CALCULATE COORDINATES OF INTEGRATION POINTS
C
C      DO 100 L=1,NINT
C      XINT(1,L) = RONDOT(SHG(NROWSH,1,L),XL(1,1),NROWSH,NSD,NEN)
C      XINT(2,L) = RONDOT(SHG(NROWSH,1,L),XL(2,1),NROWSH,NSD,NEN)
C      IF (IOPt.EQ.2) THEN
C          R(L) = XINT(1,L)
C          DET(L) = DET(L)*R(L)
C      ENDIF
C 100 CONTINUE
C
C      IF (IBBAR.EQ.1)
C      &      CALL MEANSH(SHGBAR,W,DET,R,SHG,NEN,NINT,IOPt,NESD,NROWSH)
C
C.... LOOP OVER INTEGRATION POINTS
C
C      DO 200 L=1,NINT
C
C.... CALCULATE STRESS, STRAIN AND PRINCIPAL VALUES
C
C      CALL QDCSTR(SHG(1,1,L),SHGBAR,B,R(L),DL,STRAIN,C(1,1,M),STRESS,
C      &           PSTRN,PSTRS,NROWSH,NSD,NROWB,IBBAR,NEN,NED,NEE,NSTR,IOPt)
C
C.... PRINT STRESS, STRAIN AND PRINCIPAL VALUES
C
C      CALL PRTS2D(XINT(1,L),STRESS,PSTRS,STRAIN,PSTRN,
C      &           NN,NNNTOT,NEG,NEL,L)
C 200 CONTINUE

```

```

300 CONTINUE
RETURN
END
***** SUBROUTINE QDCT5(IHIST,SOUT ,MAT ,IEN ,D ,DL ,&
& X ,XL ,DET ,SHL ,SHG ,R ,&
& SHGBAR,W ,B ,PSTRN ,PSTRS ,WORK ,STRAIN,C ,STRESS ,&
& NSOUT ,NEN ,NDOF ,NED ,NSD ,NESD ,&
& NROWSH ,NINT ,NEG ,IOPT ,IBBAR ,NROWB ,&
& NEE ,NSTR ) ; QDCT5 1
QDCT5 2
QDCT5 3
QDCT5 4
QDCT5 5
QDCT5 6
QDCT5 7
QDCT5 8
QDCT5 9
QDCT5 10
QDCT5 11
QDCT5 12
QDCT5 13
QDCT5 14
QDCT5 15
QDCT5 16
QDCT5 17
QDCT5 18
QDCT5 19
QDCT5 20
QDCT5 21
QDCT5 22
QDCT5 23
QDCT5 24
QDCT5 25
QDCT5 26
QDCT5 27
QDCT5 28
QDCT5 29
QDCT5 30
QDCT5 31
QDCT5 32
QDCT5 33
QDCT5 34
QDCT5 35
QDCT5 36
QDCT5 37
QDCT5 38
QDCT5 39
QDCT5 40
QDCT5 41
QDCT5 42
QDCT5 43
QDCT5 44
QDCT5 45
QDCT5 46
QDCT5 47
QDCT5 48
QDCT5 49
QDCT5 50
QDCT5 51
QDCT5 52
QDCT5 53
QDCT5 54
QDCT5 55
QDCT5 56
QDCT5 57
QDCT5 58
QDCT5 59
QDCT5 60
QDCT5 61
QDCT5 62
QDCT5 63
QDCT5 64
QDCT5 65
QDCT5 66
QDCT5 67
QDCT5 68
QDCT5 69
QDCT5 70
***** SUBROUTINE QDCB(SHG,SHGBAR,B,R,IOPT,NROWSH,NROWB,NEN,IBBAR) QDCB 1
QDCB 2
QDCB 3
QDCB 4
QDCB 5
QDCB 6
QDCB 7
QDCB 8
QDCB 9
QDCB 10
QDCB 11
QDCB 12
***** PROGRAM TO CALCULATE AND STORE ELEMENT TIME-HISTORIES FOR THE FOUR-NODE QUADRILATERAL, ELASTIC CONTINUUM ELEMENT QDCT5 10
IMPLICIT DOUBLE PRECISION (A-H,O-Z) QDCT5 11
. DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION QDCT5 12
REAL SOUT
LOGICAL LQUAD
DIMENSION ISHIST(3,1),SOUT(NSOUT+1,1),MAT(1),IEN(NEN,1),D(NDOF,1),&
& DL(NEL,1),X(NSD,1),XL(NESD,1),DET(1),SHL(NROWSH,NEN,1),&
& SHG(NROWSH,NEN,1),R(1),SHGBAR(3,NEN,1),W(1),B(NROWB,1),&
& STRAIN(1),C(NROWB,NROWB,1),STRESS(1),PSTRN(1),PSTRS(1),&
& WORK(1)
COMMON /HPLOT/C/NPLPTS,LOCPLT,TIME
SOUT(1,LOCPLT) = REAL(TIME)
DO 300 I=1,NSOUT
NEL = ISHIST(1,I)
INTPT = ISHIST(2,I)
NCOMP = ISHIST(3,I)
M = MAT(NEL)
CALL LOCAL(IEN(1,NEL),D,DL,NEN,NDOF,NED)
CALL LOCAL(IEN(1,NEL),X,XL,NEN,NSD,NESD)
LQUAD = .TRUE.
IF (IEN(3,NEL).EQ.IEN(4,NEL)) LQUAD = .FALSE.
CALL QDCSHG(XL,DET,SHL,SHG,NINT,NEL,NEG,LQUAD)
IF (IOPT.EQ.2) THEN
DO 100 L=1,NINT
R(L) = RROWSH(SHGBAR,W,DET,R,SHG,NEN,NINT,IOPT,NESD,NROWSH)
DET(L) = DET(.)*R(L)
CONTINUE
ENDIF
IF (IBBAR.EQ.1) & CALL MEANSHS(SHGBAR,W,DET,R,SHG,NEN,NINT,IOPT,NESD,NROWSH)
. CALCULATE STRESS, STRAIN AND PRINCIPAL VALUES
CALL QDCSTR(SHG(1,1,INTPT),SHGBAR,B,R(INTPT),DL,STRAIN,C(1,1,M),&
& STRESS,PSTRN,PSTRS,NROWSH,NESD,NROWB,IBBAR,NEN,NED,&
& NEE,NSTR,IOPT)
DO 200 J=1,4
WORK(J) = STRESS(J)
WORK(J+4) = PSTRS(J)
WORK(J+8) = STRAIN(J)
WORK(J+12) = PSTRN(J)
10 CONTINUE
SOUT(I+1,LOCPLT) = REAL(WORK(NCOMP))
10 CONTINUE
RETURN
END
***** SUBROUTINE QDCB(SHG,SHGBAR,B,R,IOPT,NROWSH,NROWB,NEN,IBBAR) QDCB 1
QDCB 2
QDCB 3
QDCB 4
QDCB 5
QDCB 6
QDCB 7
QDCB 8
QDCB 9
QDCB 10
QDCB 11
QDCB 12
. PROGRAM TO SET UP THE STRAIN-DISPLACEMENT MATRIX "B" FOR THG-DIMENSIONAL CONTINUUM ELEMENTS
IBBAR = 0, STANDARD B-MATRIX
IBBAR = 1, MEAN-DILATATIONAL B-MATRIX
IMPLICIT DOUBLE PRECISION (A-H,O-Z)

```

```

C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION QDCB 13
C
C      DIMENSION SHG(NROWSH,1),SHGBAR(3,1),B(NROWB,1) QDCB 14
C      COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE QDCB 15
C
C      DO 100 J=1,NEN QDCB 16
C
C      J2      = 2*j QDCB 17
C      J2M1   = J2 - 1 QDCB 18
C
C      B(1,J2M1) = SHG(1,J) QDCB 19
C      B(1,J2)   = ZERO QDCB 20
C      B(2,J2M1) = ZERO QDCB 21
C      B(2,J2)   = SHG(2,J) QDCB 22
C      B(3,J2M1) = SHG(2,J) QDCB 23
C      B(3,J2)   = SHG(1,J) QDCB 24
C
C      IF (IOPT.EQ.2) THEN QDCB 25
C          B(4,J2M1) = SHG(3,J)/R QDCB 26
C          B(4,J2)   = ZERO QDCB 27
C      ENDIF QDCB 28
C
C 100 CONTINUE QDCB 29
C
C      IF (IBBAR.EQ.0) RETURN QDCB 30
C
C.... ADD CONTRIBUTIONS TO FORM B-BAR QDCB 31
C
C      CONSTB = ONE/THREE QDCB 32
C
C      DO 200 J=1,NEN QDCB 33
C
C      J2      = 2*j QDCB 34
C      J2M1   = J2 - 1 QDCB 35
C
C      IF (IOPt.EQ.2) THEN QDCB 36
C          TEMP3 = CONSTB*(SHGBAR(3,J) - SHG(3,J)/R) QDCB 37
C          B(1,J2M1) = B(1,J2M1) + TEMP3 QDCB 38
C          B(2,J2M1) = B(2,J2M1) + TEMP3 QDCB 39
C          B(4,J2M1) = B(4,J2M1) + TEMP3 QDCB 40
C      ELSE QDCB 41
C          B(4,J2M1) = ZERO QDCB 42
C          B(4,J2)   = ZERO QDCB 43
C      ENDIF QDCB 44
C
C      TEMP1 = CONSTB*(SHGBAR(1,J) - SHG(1,J)) QDCB 45
C      TEMP2 = CONSTB*(SHGBAR(2,J) - SHG(2,J)) QDCB 46
C
C      B(1,J2M1) = B(1,J2M1) + TEMP1 QDCB 47
C      B(1,J2)   = B(1,J2) + TEMP2 QDCB 48
C      B(2,J2M1) = B(2,J2M1) + TEMP1 QDCB 49
C      B(2,J2)   = B(2,J2) + TEMP2 QDCB 50
C      B(4,J2M1) = B(4,J2M1) + TEMP1 QDCB 51
C      B(4,J2)   = B(4,J2) + TEMP2 QDCB 52
C
C 200 CONTINUE QDCB 53
C
C      RETURN QDCB 54
C
C      END QDCB 55
C
C***** SUBROUTINE QDCK(SHGBAR,W,DET,R,SHG,B,C,DMAT,DB,ELSTIF,CONSTK, QDCB 56
C      &           IBBAR,NEN,NINT,IOPt,NESD,NROWSH,NROWB,NSTR,NEE) QDCB 57
C
C.... PROGRAM TO FORM STIFFNESS MATRIX FOR A CONTINUUM ELEMENT QDCB 58
C      WITH "NEN" NODES QDCB 59
C
C      NOTE: THE B-BAR OPTION IS RESTRICTED TO THE MEAN-DILATATION QDCB 60
C      FORMULATION. TO GENERALIZE TO OTHER FORMULATIONS, QDCB 61
C      REDIMENSION ARRAY "SHGBAR", AND REPLACE ROUTINES QDCB 62
C      "MEANSH" AND "QDCB". QDCB 63
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z) QDCB 64
C
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION QDCB 65
C
C      DIMENSION SHGBAR(3,NEN,1),W(1),DET(1),R(1),SHG(NROWSH,NEN,1), QDCB 66
C      &           B(NROWB,1),C(NROWB,1),DMAT(NROWB,1),DB(NROWB,1), QDCB 67
C      &           ELSTIF(NEE,1) QDCB 68
C
C.... CALCULATE MEAN VALUES OF SHAPE FUNCTION GLOBAL DERIVATIVES QDCB 69
C      FOR MEAN-DILATATIONAL B-BAR FORMULATION QDCB 70
C
C      IF (IBBAR.EQ.1) QDCB 71
C      &           CALL MEANSH(SHGBAR,W,DET,R,SHG,NEN,NINT,IOPt,NESD,NROWSH) QDCB 72
C
C.... LOOP ON INTEGRATION POINTS QDCB 73
C
C      DO 100 L=1,NINT QDCB 74
C      TEMP = CONSTK*W(L)*DET(L) QDCB 75

```

```

.. SET UP THE STRAIN-DISPLACEMENT MATRIX QDCK 30
CALL QDCB(SHG(1,1,L),SHGBAR,B,R(L),IOPT,NROWSH,NROWB,NEN,IBBAR) QDCK 31
.. SET UP THE CONSTITUTIVE MATRIX QDCK 32
CALL SETUPD(C,DMAT,TEMP,NSTR,NROWB) QDCK 33
.. MULTIPLY D*B QDCK 34
CALL MULTAB(DMAT,B,DB,NROWB,NROWB,NROWB,NSTR,NSTR,NEE,1) QDCK 35
.. MULTIPLY B(TRANSPOSE) * DB, TAKING ACCOUNT OF SYMMETRY, QDCK 36
    AND ACCUMULATE IN ELSTIF QDCK 37
CALL BTDB(ELSTIF,B,DB,NEE,NROWB,NSTR) QDCK 38
QDCK 39
QDCK 40
QDCK 41
QDCK 42
QDCK 43
QDCK 44
QDCK 45
QDCK 46
QDCK 47
00 CONTINUE QDCK 48
RETURN QDCK 49
END QDCK 50
QDCK 51
***** SUBROUTINE QDCKD(SHGBAR,W,DET,R,SHG,B,DL,STRAIN,C,STRESS,WORK, QDCKD 1
& ELRESF,CONSTK,IBBAR,NEN,NINT,IOPt,NROWSH, QDCKD 2
& NESD,NROWB,NEE,NSTR) QDCKD 3
QDCKD 4
.. PROGRAM TO FORM INTERNAL FORCE ("K*D") FOR A CONTINUUM ELEMENT QDCKD 5
    WITH "NEN" NODES QDCKD 6
    NOTE: THE B-BAR OPTION IS RESTRICTED TO THE MEAN-DILATATION QDCKD 7
    FORMULATION. TO GENERALIZE TO OTHER FORMULATIONS, QDCKD 8
    REDIMENSION ARRAY "SHGBAR", AND REPLACE ROUTINES QDCKD 9
    "MEANSH" AND "QDCB". QDCKD 10
    IMPLICIT DOUBLE PRECISION (A-H,O-Z) QDCKD 11
QDCKD 12
.. DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION QDCKD 13
DIMENSION SHGBAR(3,NEN,1),W(1),DET(1),R(1),SHG(NROWSH,NEN,1), QDCKD 14
& B(NROWB,1),DL(1),STRAIN(1),C(NROWB,1),STRESS(1), QDCKD 15
& WORK(1),ELRESF(1) QDCKD 16
    IF (IBBAR.EQ.1) QDCKD 17
    & CALL MEANSH(SHGBAR,W,DET,R,SHG,NEN,NINT,IOPt,NESD,NROWSH) QDCKD 18
    DO 100 L=1,NINT QDCKD 19
    TEMP = CONSTK*W(L)*DET(L) QDCKD 20
    QDCKD 21
.. LOOP ON INTEGRATION POINTS QDCKD 22
    QDCKD 23
    QDCKD 24
    QDCKD 25
    QDCKD 26
    QDCKD 27
    QDCKD 28
.. SET UP THE STRAIN-DISPLACEMENT MATRIX QDCKD 29
CALL QDCB(SHG(1,1,L),SHGBAR,B,R(L),IOPt,NROWSH,NROWB,NEN,IBBAR) QDCKD 30
.. CALCULATE STRAINS QDCKD 31
CALL MULTAB(B,DL,STRAIN,NROWB,NEE,NSTR,NEE,NSTR,1,1) QDCKD 32
.. CALCULATE STRESSES QDCKD 33
CALL MULTAB(C,STRAIN,STRESS,NROWB,NSTR,NSTR,NSTR,NSTR,1,1) QDCKD 34
.. CALCULATE ELEMENT INTERNAL FORCE QDCKD 35
CALL SMULT(TEMP,STRESS,STRESS,NSTR,NSTR,NSTR,1,1) QDCKD 36
CALL MULTAB(B,STRESS,WORK,NROWB,NSTR,NEE,NSTR,NEE,1,2) QDCKD 37
CALL MATADD(ELRESF,WORK,ELRESF,NEE,NEE,NEE,NEE,1,1) QDCKD 38
QDCKD 39
QDCKD 40
.. CALCULATE ELEMENT INTERNAL FORCE QDCKD 41
CALL SMULT(TEMP,STRESS,STRESS,NSTR,NSTR,NSTR,1,1) QDCKD 42
CALL MULTAB(B,STRESS,WORK,NROWB,NSTR,NEE,NSTR,NEE,1,2) QDCKD 43
CALL MATADD(ELRESF,WORK,ELRESF,NEE,NEE,NEE,NEE,1,1) QDCKD 44
QDCKD 45
QDCKD 46
00 CONTINUE QDCKD 47
RETURN QDCKD 48
END QDCKD 49
QDCKD 50
***** SUBROUTINE QDCRSFIELNO,ISIDE,PRESS,SHEAR,NSURF) QDCRSF 1
.. PROGRAM TO READ, WRITE AND STORE SURFACE FORCE DATA FOR THE QDCRSF 2
    FOUR-NODE QUADRILATERAL, ELASTIC CONTINUUM ELEMENT QDCRSF 3
    IMPLICIT DOUBLE PRECISION (A-H,O-Z) QDCRSF 4
.. DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION QDCRSF 5
DIMENSION IELNO(1),ISIDE(1),PRESS(2,1),SHEAR(2,1) QDCRSF 6
COMMON /IOUNIT/ IIN,IOUT,IRSIN,IRSOUT QDCRSF 7
QDCRSF 8
QDCRSF 9
QDCRSF 10
QDCRSF 11

```

```

C
      DO 100 N=1,NSURF
      IF (MOD(N,50).EQ.1) WRITE(IOUT,1000) NSURF
      READ(IIN,2000) IELNO(N),ISIDE(N),PRESS(1,N),PRESS(2,N),
      &           SHEAR(1,N),SHEAR(2,N)
      &           WRITE(IOUT,3000) IELNO(N),ISIDE(N),PRESS(1,N),PRESS(2,N),
      &           SHEAR(1,N),SHEAR(2,N)
100  CONTINUE
C
      RETURN
C
1000 FORMAT('1',
      & ELEMENT SURFACE FORCE DATA      ',//5X,'NSURF ) = ',I5///,
      & NUMBER OF SURFACE FORCE CARDS   '(NSURF ) = ',I5///
      & ELEMENT SIDE    2(   PRESSION   ' ),
      & 2(   SHEAR   ' ),
      & 5X,2(' NUMBER ),2(' NODE I      NODE J   '),/
2000 FORMAT(215,4F10.0)
3000 FORMAT(6X,15.7X,I2,3X,4(2X,E12.4))
END
*****SUBROUTINE QDCSHG(XL,DET,SHL,SHG,NINT,NEL,NEG,LQUAD)
C.... PROGRAM TO CALCULATE GLOBAL DERIVATIVES OF SHAPE FUNCTIONS AND
C..... JACOBIAN DETERMINANTS FOR A FOUR-NODE QUADRILATERAL ELEMENT
C
CCC XL(J,I)      = GLOBAL COORDINATES
CCC DET(L)        = JACOBIAN DETERMINANT
CCC SHL(1,I,L)    = LOCAL ("XI") DERIVATIVE OF SHAPE FUNCTION
CCC SHL(2,I,L)    = LOCAL ("ETA") DERIVATIVE OF SHAPE FUNCTION
CCC SHL(3,I,L)    = LOCAL SHAPE FUNCTION
CCC SHG(1,I,L)    = X-DERIVATIVE OF SHAPE FUNCTION
CCC SHG(2,I,L)    = Y-DERIVATIVE OF SHAPE FUNCTION
CCC SHG(3,I,L)    = SHL(3,I,L)
CCC XS(I,J)       = JACOBIAN MATRIX
CCC I             = LOCAL NODE NUMBER OR GLOBAL COORDINATE NUMBER
CCC J             = GLOBAL COORDINATE NUMBER
CCC L             = INTEGRATION-POINT NUMBER
CCC NINT          = NUMBER OF INTEGRATION POINTS, E.g. 1 OR 4
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION
C
LOGICAL LQUAD
DIMENSION XL(2,1),DET(1),SHL(3,4,1),SHG(3,4,1),XS(2,2)
COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
COMMON /IOUNIT/ IIN,IOUT,IRSIM,IRSOOUT
C
CALL MOVE(SHG,SHL,12*NINT)
C
DO 700 L=1,NINT
C
IF (.NOT.LQUAD) THEN
  DO 100 I=1,3
    SHG(I,3,L) = SHL(I,3,L) + SHL(I,4,L)
    SHG(I,4,L) = ZERO
100  CONTINUE
ENDIF
C
DO 300 J=1,2
DO 200 I=1,2
XS(I,J) = RWDOT(SHG(I,1,L),XL(J,1),3,2,4)
200  CONTINUE
300  CONTINUE
C
DET(L) = XS(1,1)*XS(2,2)-XS(1,2)*XS(2,1)
IF (DET(L).LE.ZERO) THEN
  WRITE(IOUT,1000) NEL,NEG
  STOP
ENDIF
C
DO 500 J=1,2
DO 400 I=1,2
XS(I,J) = XS(I,J)/DET(L)
400  CONTINUE
500  CONTINUE
C
DO 600 I=1,4
  TEMP = XS(2,2)*SHG(1,I,L) - XS(1,2)*SHG(2,I,L)
  SHG(2,I,L) = - XS(2,1)*SHG(1,I,L) + XS(1,1)*SHG(2,I,L)
  SHG(1,I,L) = TEMP
600  CONTINUE
C
700  CONTINUE
C
RETURN

```

```

) FORMAT('1','NON-POSITIVE DETERMINANT IN ELEMENT NUMBER ',I5,
& IN ELEMENT GROUP ',I5)
END
***** SUBROUTINE QDCSHL(SHL,W,NINT)
. PROGRAM TO CALCULATE INTEGRATION-RULE WEIGHTS, SHAPE FUNCTIONS
AND LOCAL DERIVATIVES FOR A FOUR-NODE QUADRILATERAL ELEMENT
      R,S = LOCAL ELEMENT COORDINATES ("XI", "ETA", RESP.)
      SHL(1,I,L) = LOCAL ("XI") DERIVATIVE OF SHAPE FUNCTION
      SHL(2,I,L) = LOCAL ("ETA") DERIVATIVE OF SHAPE FUNCTION
      SHL(3,I,L) = LOCAL SHAPE FUNCTION
      W(L) = INTEGRATION-RULE WEIGHT
      I = LOCAL NODE NUMBER
      L = INTEGRATION POINT NUMBER
      NINT = NUMBER OF INTEGRATION POINTS, EQ. 1 OR 4
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
. DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION
DIMENSION SHL(3,4,1),W(1),RA(4),SA(4)
COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
DATA RA/-0.50,0.50,0.50,-0.50/,SA/-0.50,-0.50,0.50,0.50/
G = ZERO
W(1) = FOUR
IF (NINT.EQ.4) THEN
  G = TWO/SQRT(THREE)
  W(1) = ONE
  W(2) = ONE
  W(3) = ONE
  W(4) = ONE
ENDIF
DO 200 L=1,NINT
  R = G*RA(L)
  S = G*SA(L)
  DO 100 I=1,4
    TEMPR = PT5 + RA(I)*R
    TEMPS = PT5 + SA(I)*S
    SHL(1,I,L) = RA(I)*TEMPS
    SHL(2,I,L) = TEMPR*SA(I)
    SHL(3,I,L) = TEMPR*TEMPS
  CONTINUE
  CONTINUE
  RETURN
END
***** SUBROUTINE QDCSTR(SHG,SHGBAR,B,R,DL,STRAIN,C,STRESS,PSTRN,PSTRS,
& NRWSH,NESD,NROWB,IBBAR,NEN,NED,NEE,NSTR,IOPT)
. PROGRAM TO CALCULATE STRESS, STRAIN AND PRINCIPAL VALUES AT AN
INTEGRATION POINT FOR A TWO-DIMENSIONAL CONTINUUM ELEMENT
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
. DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION
DIMENSION SHG(NRWSH,1),SHGBAR(3,1),B(NROWB,1),DL(NED,1),
& STRAIN(1),C(NROWB,1),STRESS(1),PSTRN(1),PSTRS(1)
COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
. SET UP STRAIN-DISPLACEMENT MATRIX
  CALL QDCB(SHG,SHGBAR,B,R,IOPT,NRWSH,NROWB,NEN,IBBAR)
. CALCULATE STRAINS
  CALL MULTAB(B,DL,STRAIN,NROWB,NEE,NSTR,NEE,NSTR,1,1)
. CALCULATE STRESSES
  CALL MULTAB(C,STRAIN,STRESS,NROWB,NSTR,NSTR,NSTR,NSTR,1,1)
. CALCULATE PRINCIPAL STRAINS; ACCOUNT FOR ENGINEERING SHEAR STRAIN
  STRAIN(3) = PT5*STRAIN(3)
  CALL PRINC(NESD,STRAIN,PSTRN)
  STRAIN(3) = TWO*STRAIN(3)
  PSTRN(3) = TWO*PSTRN(3)
. CALCULATE PRINCIPAL STRESS
      QDCSHG 67
      QDCSHG 68
      QDCSHG 69
      QDCSHG 70
      QDCSHL 1
      QDCSHL 2
      QDCSHL 3
      QDCSHL 4
      QDCSHL 5
      QDCSHL 6
      QDCSHL 7
      QDCSHL 8
      QDCSHL 9
      QDCSHL 10
      QDCSHL 11
      QDCSHL 12
      QDCSHL 13
      QDCSHL 14
      QDCSHL 15
      QDCSHL 16
      QDCSHL 17
      QDCSHL 18
      QDCSHL 19
      QDCSHL 20
      QDCSHL 21
      QDCSHL 22
      QDCSHL 23
      QDCSHL 24
      QDCSHL 25
      QDCSHL 26
      QDCSHL 27
      QDCSHL 28
      QDCSHL 29
      QDCSHL 30
      QDCSHL 31
      QDCSHL 32
      QDCSHL 33
      QDCSHL 34
      QDCSHL 35
      QDCSHL 36
      QDCSHL 37
      QDCSHL 38
      QDCSHL 39
      QDCSHL 40
      QDCSHL 41
      QDCSHL 42
      QDCSHL 43
      QDCSHL 44
      QDCSHL 45
      QDCSHL 46
      QDCSHL 47
      QDCSHL 48
      QDCSTR 1
      QDCSTR 2
      QDCSTR 3
      QDCSTR 4
      QDCSTR 5
      QDCSTR 6
      QDCSTR 7
      QDCSTR 8
      QDCSTR 9
      QDCSTR 10
      QDCSTR 11
      QDCSTR 12
      QDCSTR 13
      QDCSTR 14
      QDCSTR 15
      QDCSTR 16
      QDCSTR 17
      QDCSTR 18
      QDCSTR 19
      QDCSTR 20
      QDCSTR 21
      QDCSTR 22
      QDCSTR 23
      QDCSTR 24
      QDCSTR 25
      QDCSTR 26
      QDCSTR 27
      QDCSTR 28
      QDCSTR 29
      QDCSTR 30
      QDCSTR 31
      QDCSTR 32
      QDCSTR 33
      QDCSTR 34

```

```

C      CALL PRINC(NESD,STRESS,PSTRS)          QDCSTR 3
C      IF (IOPt.EQ.0) THEN                   QDCSTR 3
C          STRESS(4) = ZERO                 QDCSTR 3
C          STRAIN(4) = - ( C(4,1)*STRAIN(1) + C(4,2)*STRAIN(2)
C                           + C(4,3)*STRAIN(3) )/C(4,4)    QDCSTR 4
C      ENDIF                                QDCSTR 4
C
C      IF ( (IOPt.EQ.1) .AND. (IBBAR.EQ.0) ) THEN   QDCSTR 4
C          STRESS(4) = ZERO                 QDCSTR 4
C          STRAIN(4) = C(4,1)*STRAIN(1) + C(4,2)*STRAIN(2)
C                           + C(4,3)*STRAIN(3)    QDCSTR 4
C      ENDIF                                QDCSTR 4
C
C      RETURN                               QDCSTR 4
C
***** SUBROUTINE QDCSUFIELNO,IEN,X,XL,ISIDE,MAT,TH,PRESS,SHEAR,ELRESF, BRHS,LM,FAC,NSURF,NEN,NSD,NED,NEE,IOPt ***** QDCSUFI
C
C.... PROGRAM TO COMPUTE CONSISTENT SURFACE LOADS FOR THE QDCSUFI
C        FOUR-NODE QUADRILATERAL, ELASTIC CONTINUUM ELEMENT QDCSUFI
C
C        NOTE: TWO-POINT GAUSSIAN QUADRATURE IS EMPLOYED QDCSUFI
C
C        IMPLICIT DOUBLE PRECISION (A-H,O-Z) QDCSUFI
C
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION QDCSUFI
C
C        DIMENSION Z(2),WORK(2),IELNO(1),IEN(NEN,1),X(NSD,1),XL(NESD,1),
C        &           ISIDE(1),MAT(1),TH(1),PRESS(2,1),SHEAR(2,1),
C        &           ELRESF(NED,1),BRHS(1),LM(NED,NEN,1) QDCSUFI
C        COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE QDCSUFI
C
C        Z(2) = PT5/SQRT(THREE) QDCSUFI
C        Z(1) = - Z(2)          QDCSUFI
C
C        DO 300 K=1,NSURF          QDCSUFI
C        NEL = IELNO(K)          QDCSUFI
C        CALL LOCAL(IEN(1,NEL),X,XL,NEN,NSD,NSD)          QDCSUFI
C        CALL CLEAR(ELRESF,NEE)          QDCSUFI
C        I = ISIDE(K)          QDCSUFI
C        J = I + 1          QDCSUFI
C        IF (J.EQ.5) J = 1          QDCSUFI
C        DX = XL(1,J) - XL(1,I)          QDCSUFI
C        DY = XL(2,J) - XL(2,I)          QDCSUFI
C        M = MAT(NEL)          QDCSUFI
C        TEMP = PT5*FAC*TH(M)          QDCSUFI
C
C        DO 200 L=1,2          QDCSUFI
C        SHI = PT5 + Z(L)          QDCSUFI
C        SHJ = PT5 + Z(L)          QDCSUFI
C        P = SHI*PRESS(1,K) + SHJ*PRESS(2,K)          QDCSUFI
C        S = SHI*SHEAR(1,K) + SHJ*SHEAR(2,K)          QDCSUFI
C
C        IF (IOPt.EQ.2) THEN          QDCSUFI
C            R = SHI*XL(1,I) + SHJ*XL(1,J)          QDCSUFI
C            P = P*R          QDCSUFI
C            S = S*R          QDCSUFI
C        ENDIF          QDCSUFI
C
C        WORK(1) = TEMP*(- P*DY + S*DX)          QDCSUFI
C        WORK(2) = TEMP*( P*DX + S*DY)          QDCSUFI
C
C        DO 100 N=1,2          QDCSUFI
C        ELRESF(N,I) = ELRESF(N,I) + SHI*WORK(N)          QDCSUFI
C        ELRESF(N,J) = ELRESF(N,J) + SHJ*WORK(N)          QDCSUFI
C 100 CONTINUE          QDCSUFI
C 200 CONTINUE          QDCSUFI
C
C        CALL ADDRHS(BRHS,ELRESF,LM(1,1,NEL),NEE)          QDCSUFI
C
C 300 CONTINUE          QDCSUFI
C
C        RETURN          QDCSUFI
C
***** SUBROUTINE TRUSS(ITASK,NPAR,MP,NEG) ***** TRUSS 1
C
C.... PROGRAM TO SET STORAGE AND CALL TASKS FOR THE TRUSS 2
C        THREE-DIMENSIONAL, ELASTIC TRUSS ELEMENT TRUSS 3
C
C        DOUBLE PRECISION TIME TRUSS 4
C
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION TRUSS 5
C                                         TRUSS 6
C                                         TRUSS 7
C                                         TRUSS 8

```

```

LOGICAL LDYN          TRUSS   9
DIMENSION NPAR(1), MP(1)    TRUSS  10
COMMON /BPOINT/ MFIRST, MLAST, MTOT, IPREC   TRUSS 11
& COMMON /DPOINT/ MPSTEP, MPDPRT, MPSRPT, MPHPLT, MPITER, MPALPH, MPBETA,   TRUSS 12
& COMMON /HPL0TC/ MPGAMM, MPDT, MPDOUT, MPIDHS, MPVPRD, MPDPRD, MPA, MPV   TRUSS 14
& COMMON /INFO/  NPLPTS, LOCPLT, TIME      TRUSS 15
& COMMON /SP0INT/ IEXEC, IACODE, LDYN, IREADR, IWRITR, IPRTIN, IRANK,   TRUSS 16
& COMMON /SP0INT/ NUMSEQ, NDOUT, NSD, NUMNP, ND0F, NLVECT, NLFTFTN, NP0TSLF,   TRUSS 17
& COMMON /SP0INT/ NUMEG                  TRUSS 18
& COMMON /SP0INT/ MPD, MPX, MPID, MPF, MPG, MPG1, MPDIAG, MPNGRP,   TRUSS 19
& COMMON /SP0INT/ MPALHS, MPBRHS        TRUSS 20
COMMON A(1)           TRUSS 21
                                         TRUSS 22
MW      = 1             TRUSS 23
MDET    = 2             TRUSS 24
MSHL    = 3             TRUSS 25
MSHG    = 4             TRUSS 26
MXS     = 5             TRUSS 27
MRHO    = 6             TRUSS 28
MRDPM   = 7             TRUSS 29
MRDPK   = 8             TRUSS 30
MAREA   = 9             TRUSS 31
MC      = 10            TRUSS 32
MGRAV   = 11            TRUSS 33
MIEN    = 12            TRUSS 34
MMAT    = 13            TRUSS 35
MLM     = 14            TRUSS 36
MISHST  = 15            TRUSS 37
MSOUT   = 16            TRUSS 38
MELEFM  = 17            TRUSS 39
MXL     = 18            TRUSS 40
MW0RK   = 19            TRUSS 41
MB      = 20            TRUSS 42
MDMAT   = 21            TRUSS 43
MDB     = 22            TRUSS 44
MVL     = 23            TRUSS 45
MAL     = 24            TRUSS 46
MELRES  = 25            TRUSS 47
MDL     = 26            TRUSS 48
MSTRN   = 27            TRUSS 49
MSTRS   = 28            TRUSS 50
MF0RCE  = 29            TRUSS 51
                                         TRUSS 52
NTYPE   = NPAR( 1)      TRUSS 53
NUMEL   = NPAR( 2)      TRUSS 54
NUMAT   = NPAR( 3)      TRUSS 55
NEN     = NPAR( 4)      TRUSS 56
NS0UT   = NPAR( 5)      TRUSS 57
ISTPRT  = NPAR( 6)      TRUSS 58
LFBODY  = NPAR( 7)      TRUSS 59
                                         TRUSS 60
IF (NPAR(8).EQ.0) NPAR(8) = 2
                                         TRUSS 61
NINT    = NPAR( 8)      TRUSS 62
IMASS   = NPAR( 9)      TRUSS 63
IMPEXP  = NPAR(10)      TRUSS 64
                                         TRUSS 65
SET ELEMENT PARAMETERS
NED     = 3             TRUSS 66
NEE     = NEN*NED      TRUSS 67
NESD   = 3              TRUSS 68
NROWSH = 2              TRUSS 69
NEESQ  = NEE*NEE       TRUSS 70
NROWB  = 1              TRUSS 71
NSTR   = 1              TRUSS 72
                                         TRUSS 73
                                         TRUSS 74
                                         TRUSS 75
                                         TRUSS 76
IF (ITASK.EQ.1) THEN
.... SET MEMORY POINTERS
NOTE: THE MP ARRAY IS STORED DIRECTLY AFTER THE NPAR ARRAY,
      BEGINNING AT LOCATION MPNPAR + 16 OF BLANK COMMON.
      THE VARIABLE "JUNK" IS NOT USED SUBSEQUETLY.
JUNK    = MP0INT('MP      ,29      ,0      ,0      ,1)    TRUSS 80
                                         TRUSS 81
                                         TRUSS 82
                                         TRUSS 83
                                         TRUSS 84
                                         TRUSS 85

```

```

C
MP(MW) = MPOINT('W',NINT,0,0,IPREC) TRUSS 86
MP(MDET) = MPOINT('DET',NINT,0,0,IPREC) TRUSS 87
MP(MSHL) = MPOINT('SHL',NROWSH,NEN,NINT,IPREC) TRUSS 88
MP(MSHG) = MPOINT('SHG',NROWSH,NEN,NINT,IPREC) TRUSS 89
MP(MXS) = MPOINT('XS',NESD,NINT,0,IPREC) TRUSS 90
MP(MRHO) = MPOINT('RHO',NUMAT,0,0,IPREC) TRUSS 91
MP(MRDPM) = MPOINT('RDAMPM',NUMAT,0,0,IPREC) TRUSS 92
MP(MRDPK) = MPOINT('RDAMPK',NUMAT,0,0,IPREC) TRUSS 93
MP(MAREA) = MPOINT('AREA',NUMAT,0,0,IPREC) TRUSS 94
MP(MC) = MPOINT('C',NROWB,NROWB,NUMAT,IPREC) TRUSS 95
MP(MGRAV) = MPOINT('GRAV',NESD,0,0,IPREC) TRUSS 96
MP(MIEN) = MPOINT('IEN',NEN,NUMEL,0,1) TRUSS 97
MP(MMAT) = MPOINT('MAT',NUMEL,0,0,1) TRUSS 98
MP(MLM) = MPOINT('LM',NED,NEN,NUMEL,1) TRUSS 99
MP(MLELFM) = MPOINT('LELEFFM',NEE,NEE,0,IPREC) TRUSS 100
MP(MXL) = MPOINT('XL',NESD,NEN,0,IPREC) TRUSS 101
MP(MWORK) = MPOINT('WORK',16,0,0,IPREC) TRUSS 102
MP(MB) = MPOINT('B',NROWB,NEE,0,IPREC) TRUSS 103
MP(MDMAT) = MPOINT('DMAT',NROWB,NROWB,0,IPREC) TRUSS 104
MP(MDB) = MPOINT('DB',NROWB,NEE,0,IPREC) TRUSS 105
MP(MVL) = MPOINT('VL',NED,NEN,0,IPREC) TRUSS 106
MP(MAL) = MPOINT('AL',NED,NEN,0,IPREC) TRUSS 107
MP(MELRES) = MPOINT('ELRESF',NEE,0,0,IPREC) TRUSS 108
MP(MDL) = MPOINT('DL',NED,NEN,0,IPREC) TRUSS 109
MP(MSTRN) = MPOINT('STRAIN',NROWB,0,0,IPREC) TRUSS 110
MP(MSTRS) = MPOINT('STRESS',NROWB,0,0,IPREC) TRUSS 111
MP(MFORCE) = MPOINT('FORCE',NROWB,0,0,IPREC) TRUSS 112
IF (NSOUT.EQ.0) THEN TRUSS 113
  MP(MISHST) = JUNK TRUSS 114
  MP(MSOUT) = JUNK TRUSS 115
ELSE TRUSS 116
  MP(MISHST) = MPOINT('ISHIST',3,NSOUT,0,1) TRUSS 117
  MP(MSOUT) = MPOINT('SOUT',NSOUT+1,NPLPTS,0,1) TRUSS 118
ENDIF TRUSS 119
MP(MELEFM) = MPOINT('LELEFFM',NEE,NEE,0,IPREC) TRUSS 120
MP(MXL) = MPOINT('XL',NESD,NEN,0,IPREC) TRUSS 121
MP(MWORK) = MPOINT('WORK',16,0,0,IPREC) TRUSS 122
MP(MB) = MPOINT('B',NROWB,NEE,0,IPREC) TRUSS 123
MP(MDMAT) = MPOINT('DMAT',NROWB,NROWB,0,IPREC) TRUSS 124
MP(MDB) = MPOINT('DB',NROWB,NEE,0,IPREC) TRUSS 125
MP(MVL) = MPOINT('VL',NED,NEN,0,IPREC) TRUSS 126
MP(MAL) = MPOINT('AL',NED,NEN,0,IPREC) TRUSS 127
MP(MELRES) = MPOINT('ELRESF',NEE,0,0,IPREC) TRUSS 128
MP(MDL) = MPOINT('DL',NED,NEN,0,IPREC) TRUSS 129
MP(MSTRN) = MPOINT('STRAIN',NROWB,0,0,IPREC) TRUSS 130
MP(MSTRS) = MPOINT('STRESS',NROWB,0,0,IPREC) TRUSS 131
MP(MFORCE) = MPOINT('FORCE',NROWB,0,0,IPREC) TRUSS 132
ENDIF TRUSS 133
C.... TASK CALLS TRUSS 134
C
IF (ITASK.GT.6) RETURN TRUSS 135
GO TO (100,200,300,400,500,600),ITASK TRUSS 136
C 100 CONTINUE TRUSS 137
C.... INPUT ELEMENT DATA ('INPUT____')
C
CALL TRUST1(A(MP(MSHL)),A(MP(MW)),A(MP(MRHO)),TRUSS 138
  & A(MP(MRDPM)),A(MP(MRDPK)),A(MP(MAREA)),TRUSS 139
  & A(MP(MC)),A(MP(MGRAV)),A(MP(MIEN)),TRUSS 140
  & A(MP(MMAT)),A(MPID),A(MP(MLM)),TRUSS 141
  & A(MPDIAG),A(MP(MISHST)),TRUSS 142
  & NTYPE,NUMEL,NUMAT,NEN,NSOUT,ISTPR,TRUSS 143
  & LFBODY,NINT,IMASS,IMPEXP,NROWSH,NROWB,TRUSS 144
  & NESD,NDOF,NED,IPRTIN,LDDY) TRUSS 145
C
RETURN TRUSS 146
C 200 CONTINUE TRUSS 147
C.... FORM ELEMENT EFFECTIVE MASS AND ASSEMBLE INTO GLOBAL TRUSS 148
C LEFT-HAND-SIDE MATRIX ('FORM_LHS') TRUSS 149
C
CALL TRUST2(A(MP(MELEFM)),A(MP(MIEN)),A(MPX),TRUSS 150
  & A(MP(MXL)),A(MP(MMAT)),A(MP(MDET)),TRUSS 151
  & A(MP(MSHL)),A(MP(MSHG)),A(MP(MRDPM)),TRUSS 152
  & A(MP(MRDPK)),A(MP(MAREA)),A(MP(MRHO)),TRUSS 153
  & A(MP(MW)),A(MP(MWORK)),A(MP(MB)),TRUSS 154
  & A(MP(MC)),A(MP(MDMAT)),A(MP(MDB)),TRUSS 155
  & A(MPALHS),A(MPDIAG),A(MP(MLM)),TRUSS 156
  & A(MP(MXS)),TRUSS 157
  IMPEXP,IMASS,NUMEL,NEESQ,NEN,NSD,TRUSS 158
  NESD,NINT,NEG,NROWSH,LDDY,NED,TRUSS 159
  NROWB,NSTR,NEE) TRUSS 160
C
RETURN TRUSS 161
C 300 CONTINUE TRUSS 162
C.... FORM ELEMENT RESIDUAL-FORCE VECTOR AND ASSEMBLE INTO GLOBAL TRUSS 163
C RIGHT-HAND-SIDE VECTOR ('FORM_RHS') TRUSS 164
C TRUSS 165
C TRUSS 166
C TRUSS 167

```

```

CALL TRUST3(A(MP(MMAT )),A(MP(MIEN )),A(MPDPRD  ),  

&          A(MP(MDL )),A(MPVPRD ),A(MP(MVL )),  

&          A(MPA  ),A(MP(MAL )),A(MP(MRDPK )),  

&          A(MP(MRDPM )),A(MP(MRH0 )),A(MP(MGRAV )),  

&          A(MP(MELRES)),A(MPX  ),A(MP(MXL )),  

&          A(MP(MDET )),A(MP(MSHL )),A(MP(MSHG )),  

&          A(MPG1  ),A(MP(MWORK )),A(MP(MAREA )),  

&          A(MP(MW )),A(MP(MELEFM)),A(MP(MB )),  

&          A(MP(MSTRN )),A(MP(MC )),A(MP(MDMAT )),  

&          A(MP(MSTRS )),A(MP(BRHS )),A(MP(MLM )),  

&          A(MP(MXS )),  

&          NUMEL ,NED  ,NEN  ,NDOF ,LDYN ,NEE  

&          IMASS ,NESD ,LFBODY ,NSD  ,NINT ,NROWSH,  

&          NEG  ,NROWB )  

      RETURN  

) CONTINUE  

. CALCULATE AND PRINT ELEMENT STRESS/STRAIN OUTPUT ('STR_PRNT')  

  IF (ISTPRT.EQ.0)  

    CALL TRUST4(A(MP(MMAT )),A(MP(MIEN )),A(MPD  

&          A(MP(MDL )),A(MPX  ),A(MP(MXL )),  

&          A(MP(MDET )),A(MP(MSHL )),A(MP(MSHG )),  

&          A(MP(MXS )),A(MP(MWORK )),A(MP(MB )),  

&          A(MP(MSTRN )),A(MP(MC )),A(MP(MSTRS )),  

&          A(MP(MFORCE)),A(MP(MAREA )),  

&          NINT ,NUMEL ,NEN  ,NDOF ,NED  ,NSD  

&          NESD ,NROWSH,NEG  ,NROWB ,NEE )  

      RETURN  

) CONTINUE  

. CALCULATE AND STORE ELEMENT TIME-HISTORIES ('STR_STOR')  

  IF (NSOUT.GT.0)  

    CALL TRUST5(A(MP(MISHST)),A(MP(MSOUT )),A(MP(MMAT )),  

&          A(MP(MIEN )),A(MPD  ),A(MP(MDL )),  

&          A(MPX  ),A(MP(MXL )),A(MP(MDET )),  

&          A(MP(MSHL )),A(MP(MSHG )),A(MP(MXS )),  

&          A(MP(MB )),A(MP(MSTRN )),A(MP(MC )),  

&          A(MP(MSTRS )),A(MP(MFORCE)),A(MP(MAREA )),  

&          NSOUT ,NEN  ,NDOF ,NED  ,NSD ,NESD ,  

&          NROWSH,NINT ,NEG  ,NROWB ,NEE )  

      RETURN  

) CONTINUE  

. PLOT ELEMENT TIME-HISTORIES ('STR_PLOT')  

  IF (NSOUT.GT.0)  

    CALL HPLOT(A(MP(MISHST)),A(MP(MSOUT )),NSOUT ,3,NTYPE )  

    RETURN  

END
*****SUBROUTINE TRUST1(SHL  ,W   ,RHO  ,RDAMPM, RDAMPK, AREA  ,  

&          C   ,GRAV ,IEN  ,MAT  ,ID  ,LM  ,  

&          IDIAG ,ISHIST ,NUMAT ,NEN  ,NSOUT ,ISTPRT,  

&          NTYPE ,NUMEL ,NEN  ,NSD  ,NDOF ,IPRTIN,LDYN )  

&          LFBODY ,NINT ,IMASS ,IMPEXP,NROWSH,NROWB ,  

&          NESD ,NDOF ,NED  ,IPRTIN,LDYN )  

PROGRAM TO READ, GENERATE AND WRITE ELEMENT DATA FOR THE  

THREE-DIMENSIONAL, ELASTIC TRUSS ELEMENT  

IMPLICIT DOUBLE PRECISION (A-H,O-Z)  

DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION  

LOGICAL LDYN  

DIMENSION SHL(NROWSH,NEN,1),W(1),RHO(1),RDAMPM(1),RDAMPK(1),  

&          AREA(1),C(NROWB,NROWB,1),GRAV(NESD),IEN(NEN,1),MAT(1),  

&          ID(NDOF,1),LM(NED,NEN,1),IDIAG(1),ISHIST(3,1)  

&          COMMON /IOUNIT/ IIN,IOUT,IRSIN,IRSOUP  

WRITE(IOUT,1000) NTYPE,NUMEL,NUMAT,NEN,NSOUT,ISTPRT,LFBODY,NINT  

IF (LDYN) WRITE(IOUT,2000) IMASS,IMPEXP  

CALL TRUSHL(SHL,W,NINT,NEN)  

CALL TRUSPR(RHO,RDAMPM, RDAMPK, AREA,C,NUMAT)

```



```

IF ( LDYN .AND. (IMASS.NE.2) ) THEN TRUST2 41
  .... FORM MASS MATRIX TRUST2 42
  CONSTM = (ONE + RDAMPM(M)*COEFF4)*AREA(M)*RHO(M)
  & IF (CONSTM.NE.ZERO) CALL CNTM(SHG,XL,W,DET,ELEFFM,WORK,
    & CONSTM,IMASS,NINT,NR0NSH,NE$D,NEN,NED,NEE,.FALSE.)
  ENDIF TRUST2 43
  IF ( (.NOT.LDYN) .OR. (IMPEXP.EQ.0) ) THEN TRUST2 44
  .... FORM STIFFNESS MATRIX TRUST2 45
  CONSTK = (COEFF4*RDAMPK(M) + COEFF5)*AREA(M)
  & CALL TRUSK(W,DET,SHG,XS,XL,B,C(1,1,M),DMAT,DB,ELEFFM,
    & CONSTK,NEN,NINT,NE$D,NR0NSH,NROWB,NSTR,NEE)
  ENDIF TRUST2 46
  . ASSEMBLE ELEMENT EFFECTIVE MASS MATRIX INTO GLOBAL TRUST2 47
  LEFT-HAND-SIDE MATRIX TRUST2 48
  CALL ADDLHS(ALHS,ELEFFM,IDIAG,LM(1,1,NEL),NEE,LDIAG) TRUST2 49
  0 CONTINUE TRUST2 50
  RETURN TRUST2 51
END TRUST2 52
***** SUBROUTINE TRUST3(MAT ,IEN ,DPRED ,DL ,VPRED ,VL ,
  & A ,AL ,RDAMPK,RDAMPM,RHO ,GRAV ,
  & ELRESF,X ,XL ,DET ,SHL ,SHG ,
  & G1 ,WORK ,AREA ,W ,ELEFFM,B ,
  & STRAIN,C ,DMAT ,STRESS ,BRHS ,LM ,
  & XS ,
  & NUMEL ,NED ,NEN ,NDOF ,LDYN ,NEE ,
  & IMASS ,NE$D ,LFBODY ,NSD ,NINT ,NR0NSH ,
  & NEG ,NROWB ) TRUST3 53
  . PROGRAM TO CALCULATE RESIDUAL-FORCE VECTOR FOR THE TRUST3 54
  THREE-DIMENSIONAL ELASTIC TRUSS ELEMENT AND TRUST3 55
  ASSEMBLE INTO THE GLOBAL RIGHT-HAND-SIDE VECTOR TRUST3 56
  IMPLICIT DOUBLE PRECISION (A-H,O-Z) TRUST3 57
  . DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION TRUST3 58
  LOGICAL LDYN,FORMMA,FORMKD,ZERODL,ZEROGL,LNODE3 TRUST3 59
  DIMENSION MAT(1),IEN(NEN,1),DPRED(NDOF,1),DL(NED,1),VPRED(NDOF,1),TRUST3 60
  & VL(NED,1),A(NDOF,1),AL(NED,1),RDAMPK(1),RDAMPM(1),TRUST3 61
  & RH0(1),GRAV(1),ELRESF(1),X(NSD,1),XL(NESD,1),DET(1),TRUST3 62
  & SHL(NR0NSH,NEN,1),SHG(NR0NSH,NEN,1),G1(1),WORK(1),TRUST3 63
  & AREA(1,W(1)),ELEFFM(NEE,1),B(NROWB,1),STRAIN(1),TRUST3 64
  & C(NROWB,NROWB,1),DMAT(NROWB,1),STRESS(1),BRHS(1),TRUST3 65
  & LM(NED,NEN,1),XS(NESD,1),TRUST3 66
  COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE TRUST3 67
  DO 500 NEL=1,NUMEL TRUST3 68
  FORMMA = .FALSE. TRUST3 69
  FORMKD = .FALSE. TRUST3 70
  M = MAT(NEL)
  . NOTE: FOR STATIC ANALYSIS MPDPRD = MPD, HENCE REFERENCE TRUST3 71
  TO ARRAY "DPRED" WILL ACCESS THE CONTENTS OF ARRAY "D". TRUST3 72
  CALL LOCAL(IEN(1,NEL),DPRED,DL,NEN,NDOF,NED) TRUST3 73
  IF (LDYN) THEN TRUST3 74
  CALL LOCAL(IEN(1,NEL),VPRED,VL,NEN,NDOF,NED) TRUST3 75
  CALL LOCAL(IEN(1,NEL),A,AL,NEN,NDOF,NED) TRUST3 76
  DO 200 J=1,NEN TRUST3 77
  DO 100 I=1,NED TRUST3 78
  DL(I,J) = DL(I,J) + RDAMPK(M)*VL(I,J) TRUST3 79
  AL(I,J) = AL(I,J) + RDAMPM(M)*VL(I,J) TRUST3 80
  00 CONTINUE TRUST3 81
  00 CONTINUE TRUST3 82
  CALL ZTEST(AL,NEE,ZERODL) TRUST3 83
  & IF ( (.NOT.ZERODL) .AND. (IMASS.NE.2) .AND. (RHO(M).NE.ZERO) ) TRUST3 84
    & FORMMA = .TRUE. TRUST3 85
  ELSE TRUST3 86

```

Sec. 11.5 DLEARN User's Manual

```

C      CALL CLEAR(AL,NEE)                                     TRUST
C      ENDIF                                                 TRUST
C      CALL ZTEST(DL,NEE,ZERODL)                               TRUST
C      IF (.NOT.ZERODL) FORMKD = .TRUE.                         TRUST
C      CALL ZTEST(GRAV,NESD,ZEROG)                             TRUST
C      IF ((.NOT.ZEROG).AND. (LFBODY.NE.0).AND. (RHO(M).NE.ZERO)
C & .AND. (IMASS.NE.2)) THEN                                TRUST
C          FORMMA = .TRUE.                                    TRUST
C          DO 400 I=1,NED                                     TRUST
C             WORK(I) = GRAV(I)*G1(LFBODY)                   TRUST
C          DO 300 J=1,NEN                                     TRUST
C             AL(I,J) = AL(I,J) - WORK(I)                     TRUST
300      CONTINUE                                         TRUST
C      400      CONTINUE                                     TRUST
C      ENDIF                                                 TRUST
C      IF (FORMMA.OR.FORMKD) THEN                            TRUST
C          CALL CLEAR(ELRESF,NEE)                            TRUST
C          CALL LOCAL(IEN(1,NEL),X,XL,NEN,NSD,NESD)           TRUST
C          LNODE3 = .TRUE.                                  TRUST
C          IF (NEN.EQ.3 .AND. IEN(2,NEL).EQ.IEN(3,NEL)) LNODE3 = .FALSE. TRUST
C          CALL TRUSHG(XL,DET,SHL,SHG,XS,NEN,NINT,NEL,NEG,LNODE3) TRUST
C          IF (FORMMA) THEN                                 TRUST
C..... FORM INERTIAL AND/OR BODY FORCE                  TRUST
C          CONSTM = - AREA(M)*RHO(M)                      TRUST
C          CALL CNTM(AREA(XL,W,DET,AL,ELEFFM,WORK,ELRESF,CONSTM,IMASS,
C & NINT,NROWSH,NESD,NEN,NED,NEE)                      TRUST
C          ENDIF                                              TRUST
C          IF (FORMKD) THEN                                 TRUST
C..... FORM INTERNAL FORCE                           TRUST
C          CONSTK = - AREA(M)                            TRUST
C          CALL TRUSKD(W,DET,SHG,XS,XL,B,DL,STRAIN,C(1,1,M),DMAT,
C & STRESS,WORK,ELRESF,CONSTK,NEN,NINT,NROWSH,
C & NESD,NROWB,NEE)                                    TRUST
C          ENDIF                                              TRUST
C          CALL ADDRHS(BRHS,ELRESF,LM(1,1,NEL),NEE)          TRUST
C          ENDIF                                              TRUST
C 500      CONTINUE                                         TRUST
C          RETURN                                           TRUST
C          END                                                 TRUST
***** SUBROUTINE TRUST4(MAT ,IEN ,D ,DL ,X ,XL ,
C & DET ,SHL ,SHG ,XS ,XINT ,B ,
C & STRAIN,C ,STRESS ,FORCE ,AREA ,
C & NINT ,NUMEL ,NEN ,NDOF ,NED ,NSD ,
C & NESD ,NROWSH ,NEG ,NROWB ,NEE ) ,TRUST
C.... PROGRAM TO CALCULATE AND PRINT STRESS, STRAIN AND FORCE FOR THE
C.... THREE-DIMENSIONAL, ELASTIC TRUSS ELEMENT          TRUST
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                TRUST
C.... DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION TRUST
C      LOGICAL LNODE3                                     TRUST
C      DIMENSION MAT(1),IEN(NEN,1),D(NDOF,1),DL(NED,1),X(NSD,1),
C & XL(NESD,1),DET(1),SHL(NROWSH,NEN,1),SHG(NROWSH,NEN,1),
C & XS(NESD,1),XINT(NESD,1),B(NROWB,1),STRAIN(1),
C & C(NROWB,NROWB,1),STRESS(1),FORCE(1),AREA(1)        TRUST
C      NNTOT = 24                                         TRUST
C      NN = 0                                            TRUST
C      DO 300 NEL=1,NUMEL                                TRUST
C      M = MAT(NEL)                                     TRUST
C      CALL LOCAL(IEN(1,NEL),D,DL,NEN,NDOF,NED)          TRUST
C      CALL LOCAL(IEN(1,NEL),X,XL,NEN,NSD,NESD)          TRUST
C      LNODE3 = .TRUE.                                  TRUST
C      IF (NEN.EQ.3 .AND. IEN(2,NEL).EQ.IEN(3,NEL)) LNODE3 = .FALSE. TRUST
C      CALL TRUSHG(XL,DET,SHL,SHG,XS,NEN,NINT,NEL,NEG,LNODE3) TRUST

```

```

LOOP OVER INTEGRATION POINTS          TRUST4 31
DO 200 L=1,NINT                      TRUST4 32
CALCULATE COORDINATES OF INTEGRATION POINTS   TRUST4 33
DO 100 I=1,NESD                      TRUST4 34
XINT(I,L) = ROWDOT(SHG(NROWSH,1,L),XL(I,1),NROWSH,NESD,NEN)
) CONTINUE                            TRUST4 35
, CALCULATE STRESS, STRAIN AND FORCE    TRUST4 36
CALL TRUSR(SHG(1,1,L),XS(1,L),B,DL,STRAIN,C(1,1,M),STRESS,
&           FORCE,AREA(M),NROWSH,NESD,NROWB,NEN,NEE)   TRUST4 37
, PRINT STRESS, STRAIN AND FORCE      TRUST4 38
CALL TRUSPT(XINT(1,L),STRESS,FORCE,STRAIN,NN,NNTOT,NEG,NEL,L)   TRUST4 39
) CONTINUE                            TRUST4 40
) CONTINUE                            TRUST4 41
, RETURN                               TRUST4 42
END                                  TRUST4 43
*****SUBROUTINE TRUST5(ISHIST,SOUT,MAT,IEN,D,DL,XL,DET,SHL,SHG,XS,B,STRAIN,C,STRESS,FORCE,AREA,WORK,NSD,NED,NSOUT,NEN,NDOF,NEE,NROWSH,NINT,NEG,NROWB)*****
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
PROGRAM TO CALCULATE AND STORE ELEMENT TIME-HISTORIES FOR THE TRUST5 8
THREE-DIMENSIONAL, ELASTIC TRUSS ELEMENT   TRUST5 9
IMPLICIT DOUBLE PRECISION (A-H,O-Z)        TRUST5 10
DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION   TRUST5 11
REAL SOUT                                TRUST5 12
LOGICAL LNODE3                            TRUST5 13
DIMENSION ISHIST(3,1),SOUT(NSOUT+1,1),MAT(1),IEN(NEN,1),D(NDOF,1),TRUST5 14
& DL(NED,1),X(NSD,1),XL(NESD,1),DET(1),SHL(NROWSH,NEN,1),TRUST5 15
& SHG(NROWSH,NEN,1),XS(NESD,1),B(NROWB,1),STRAIN(1),TRUST5 16
& C(NROWB,NROWB,1),STRESS(1),FORCE(1),AREA(1),WORK(1)   TRUST5 17
COMMON /HPLOTC/ NPLPTS,LOCPLT,TIME       TRUST5 18
SOUT(1,LOCPLT) = REAL(TIME)               TRUST5 19
DO 100 I=1,NSOUT                          TRUST5 20
NEL = ISHIST(1,I)                         TRUST5 21
INTPT = ISHIST(2,I)                       TRUST5 22
NCOMP = ISHIST(3,I)                       TRUST5 23
M = MAT(NEL)                             TRUST5 24
CALL LOCAL(IEN(1,NEL),D,DL,NEN,NDOF,NED)   TRUST5 25
CALL LOCAL(IEN(1,NEL),X,XL,NEN,NSD,NEED)   TRUST5 26
LNODE3 = .TRUE.                           TRUST5 27
IF ( NEN .EQ. 3 .AND. IEN(2,NEL).EQ.IEN(3,NEL) ) LNODE3 = .FALSE.   TRUST5 28
CALL TRUSHG(XL,DET,SHL,SHG,XS,NEN,NINT,NEL,NEG,LNODE3)   TRUST5 29
CALL TRUSR(SHG(1,1,INTPT),XS(1,INTPT),B,DL,STRAIN,C(1,1,M),
&           STRESS,FORCE,AREA(M),NROWSH,NESD,NROWB,NEN,NEE)   TRUST5 30
WORK(1) = STRESS(1)                      TRUST5 31
WORK(2) = FORCE(1)                       TRUST5 32
WORK(3) = STRAIN(1)                      TRUST5 33
SOUT(I+1,LOCPLT) = REAL(WORK(NCOMP))     TRUST5 34
) CONTINUE                                TRUST5 35
, RETURN                               TRUST5 36
END                                     TRUST5 37
*****SUBROUTINE TRUSB(B,SHG,XS,NEN,NESD,NROWB,NROWSH)*****
1
2
3
4
5
6
7
8
9
10
PROGRAM TO SET UP THE STRAIN-DISPLACEMENT MATRIX FOR THE TRUSB 2
THREE-DIMENSIONAL, ELASTIC TRUSS ELEMENT   TRUSB 3
IMPLICIT DOUBLE PRECISION (A-H,O-Z)        TRUSB 4
DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION   TRUSB 5
DIMENSION B(NROWB,1),SHG(NROWSH,1),XS(1)   TRUSB 6

```

```

C      DO 200 J=1,NEN          TRUSB
C      K = (J - 1)*NESD        TRUSB
C      DO 100 I=1,NESD         TRUSB
C      B(1,K+I) = SHG(1,J)*XS(I) TRUSB
100  CONTINUE                 TRUSB
C 200 CONTINUE                 TRUSB
C      RETURN                  TRUSB
C      END                     TRUSB
C*****SUBROUTINE TRUSHG(XL,DET,SHL,SHG,XS,NEN,NINT,NEL,NEG,LNODE3) TRUSHG
C... PROGRAM TO CALCULATE GLOBAL DERIVATIVES OF SHAPE FUNCTIONS TRUSHG
C     AND JACOBIAN DETERMINANTS FOR THE THREE-DIMENSIONAL, TRUSHG
C     ELASTIC TRUSS ELEMENT TRUSHG
CCCCXL(J,L) = GLOBAL COORDINATES OF NODAL POINTS TRUSHG
CCCCSHL(1,I,L) = LOCAL ("XI") DERIVATIVE OF SHAPE FUNCTION TRUSHG
CCCCSHL(2,I,L) = SHAPE FUNCTION TRUSHG
CCCCSHG(1,I,L) = GLOBAL ("ARC-LENGTH") DERIVATIVE OF SHAPE FTN TRUSHG
CCCCSHG(2,I,L) = SHL(2,I,L) TRUSHG
CCCCXS(J,L) = JTH COMPONENT OF THE LOCAL DERIVATIVE TRUSHG
C           OF THE POSITION VECTOR; THEN SCALED TO TRUSHG
C           DIRECTION COSINE TRUSHG
CDET(L) = EUCLIDEAN LENGTH OF XS TRUSHG
CI = LOCAL NODE NUMBER TRUSHG
CJ = GLOBAL COORDINATE NUMBER TRUSHG
CL = INTEGRATION-POINT NUMBER TRUSHG
CNINT = NUMBER OF INTEGRATION POINTS TRUSHG
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Z) TRUSHG
C
C... DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION TRUSHG
C
LOGICAL LNODE3 TRUSHG
DIMENSION XL(3,1),DET(1),SHL(2,NEN,1),SHG(2,NEN,1),XS(3,1) TRUSHG
COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE TRUSHG
COMMON /IOUNIT/ IIN,IOUT,IRSI,IRSOUT TRUSHG
C
CALL MOVE(SHG,SHL,2*NEN*NINT) TRUSHG
C
DO 400 L=1,NINT TRUSHG
IF (.NOT.LNODE3) THEN TRUSHG
  TEMP = PT5*SHG(1,3,L) TRUSHG
  SHG(1,1,L) = SHG(1,1,L) + TEMP TRUSHG
  SHG(1,2,L) = SHG(1,2,L) + TEMP TRUSHG
  TEMP = PT5*SHG(2,3,L) TRUSHG
  SHG(2,1,L) = SHG(2,1,L) + TEMP TRUSHG
  SHG(2,2,L) = SHG(2,2,L) + TEMP TRUSHG
ENDIF TRUSHG
DET(L) = ZERO TRUSHG
C
DO 100 J=1,3 TRUSHG
XS(J,L) = ROWDOT(SHL(1,J,L),XL(J,1),2,3,NEN) TRUSHG
DET(L) = DET(L) + XS(J,L)**2 TRUSHG
100  CONTINUE TRUSHG
C
DET(L) = SQRT(DET(L)) TRUSHG
C
IF (DET(L).LE.ZERO) THEN TRUSHG
  WRITE(IOUT,1000) NEL,NEG TRUSHG
  STOP TRUSHG
ENDIF TRUSHG
C
DO 200 J=1,3 TRUSHG
XS(J,L) = XS(J,L)/DET(L) TRUSHG
200  CONTINUE TRUSHG
C
DO 300 I=1,NEN TRUSHG
SHG(1,I,L) = SHL(1,I,L)/DET(L) TRUSHG
300  CONTINUE TRUSHG
C
400 CONTINUE TRUSHG
C
RETURN TRUSHG
C
1000 FORMAT('1','NON-POSITIVE DETERMINANT IN ELEMENT NUMBER ',I5, TRUSHG
&           'IN ELEMENT GROUP ',I5) TRUSHG
END TRUSHG
C*****SUBROUTINE TRUSHL(SHL,W,NINT,NEN) TRUSHL
C... PROGRAM TO CALCULATE INTEGRATION-RULE WEIGHTS, SHAPE FUNCTIONS TRUSHL
C     AND LOCAL DERIVATIVES FOR A TWO OR THREE NODE, TRUSHL
C     ONE-DIMENSIONAL ELEMENT TRUSHL

```

```

R = LOCAL ELEMENT COORDINATE ("XI")
SHL(1,I,L) = LOCAL ("XI") DERIVATIVE OF SHAPE FUNCTION
SHL(2,I,L) = SHAPE FUNCTION
W(L) = INTEGRATION-RULE WEIGHT
I = LOCAL NODE NUMBER
L = INTEGRATION-POINT NUMBER
NINT = NUMBER OF INTEGRATION POINTS, EQ. 1, 2 OR 3
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION
DIMENSION SHL(2,NEN,1),W(1),RA(3)
COMMON /CONSTS/ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE
DATA RA/.00,1.00,0.00/
&      FIVE9/0.5555555555555555/,EIGHT9/0.8888888888888888/
IF (NINT.EQ.1) THEN
  W(1) = TWO
  G = ZERO
ENDIF
IF (NINT.EQ.2) THEN
  W(1) = ONE
  W(2) = ONE
  G = ONE/SQRT(THREE)
ENDIF
IF (NINT.EQ.3) THEN
  W(1) = FIVE9
  W(2) = FIVE9
  W(3) = EIGHT9
  G = SQRT(THREE/FIVE)
ENDIF
DO 100 L=1,NINT
  R = G*RACL
  SHL(1,1,L) = - PT5
  SHL(1,2,L) = PT5
  SHL(2,1,L) = PT5*(ONE - R)
  SHL(2,2,L) = PT5*(ONE + R)
  IF (NEN.EQ.3) THEN
    SHL(1,3,L) = - TWO*R
    SHL(2,3,L) = ONE - R**2
    TEMP = - PT5*SHL(2,3,L)
    SHL(1,1,L) = SHL(1,1,L) + R
    SHL(1,2,L) = SHL(1,2,L) + R
    SHL(2,1,L) = SHL(2,1,L) + TEMP
    SHL(2,2,L) = SHL(2,2,L) + TEMP
  ENDIF
  100 CONTINUE
  RETURN
END
*****SUBROUTINE TRUSK(W,DET,SHG,XS,XL,B,C,DMAT,DB,ELSTIF,CONSTK,
  & NEN,NINT,NESD,NROWSH,NROWB,NSTR,NEE)
  . PROGRAM TO FORM STIFFNESS MATRIX FOR THE
    THREE-DIMENSIONAL, ELASTIC TRUSS ELEMENT
  IMPLICIT DOUBLE PRECISION (A-H,O-Z)
  . DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION
  DIMENSION W(1),DET(1),SHG(NROWSH,NEN,1),XS(NESD,1),XL(NESD,1),
  & B(NROWB,1),DB(NROWB,1),ELSTIF(NEE,1)
  . LOOP ON INTEGRATION POINTS
  DO 100 L=1,NINT
    TEMP1 = CONSTK*W(L)*DET(L)
    . SET UP THE STRAIN-DISPLACEMENT MATRIX
    CALL TRUSB(B,SHG(1,1,L),XS(1,L),NEN,NESD,NROWB,NROWSH)
    . SET UP THE CONSTITUTIVE "MATRIX"
    DMAT = C*TEMP1
    . MULTIPLY DMAT * B
    TRUSL   6
    TRUSL   7
    TRUSL   8
    TRUSL   9
    TRUSL  10
    TRUSL  11
    TRUSL  12
    TRUSL  13
    TRUSL  14
    TRUSL  15
    TRUSL  16
    TRUSL  17
    TRUSL  18
    TRUSL  19
    TRUSL  20
    TRUSL  21
    TRUSL  22
    TRUSL  23
    TRUSL  24
    TRUSL  25
    TRUSL  26
    TRUSL  27
    TRUSL  28
    TRUSL  29
    TRUSL  30
    TRUSL  31
    TRUSL  32
    TRUSL  33
    TRUSL  34
    TRUSL  35
    TRUSL  36
    TRUSL  37
    TRUSL  38
    TRUSL  39
    TRUSL  40
    TRUSL  41
    TRUSL  42
    TRUSL  43
    TRUSL  44
    TRUSL  45
    TRUSL  46
    TRUSL  47
    TRUSL  48
    TRUSL  49
    TRUSL  50
    TRUSL  51
    TRUSL  52
    TRUSL  53
    TRUSL  54
    TRUSL  55
    TRUSL  56
    TRUSL  57
    TRUSL  58
    TRUSL  59
    TRUSL  60
    TRUSL  61
    TRUSL  62
    TRUSL  63
    TRUSL  64
    TRUSL  65
    TRUSK   1
    TRUSK   2
    TRUSK   3
    TRUSK   4
    TRUSK   5
    TRUSK   6
    TRUSK   7
    TRUSK   8
    TRUSK   9
    TRUSK  10
    TRUSK  11
    TRUSK  12
    TRUSK  13
    TRUSK  14
    TRUSK  15
    TRUSK  16
    TRUSK  17
    TRUSK  18
    TRUSK  19
    TRUSK  20
    TRUSK  21
    TRUSK  22
    TRUSK  23
    TRUSK  24
    TRUSK  25
    TRUSK  26
    TRUSK  27

```

```

C      CALL SMULT(DMAT,B,DB,NROWB,NROWB,NSTR,NEE,1)          TRUSK
C.... MULTIPLY B(TRANSPOSE) * DB, TAKING ACCOUNT OF SYMMETRY,   TRUSK
C      AND ACCUMULATE IN ELSTIF                                TRUSK
C      CALL BTDB(ELSTIF,B,DB,NEE,NROWB,NSTR)                   TRUSK
C 100 CONTINUE                                              TRUSK
C      RETURN                                              TRUSK
C      END                                                 TRUSK
C*****SUBROUTINE TRUSKD(W,DET,SHG,XS,XL,B,DL,STRAIN,C,DMAT,STRESS,WORK,ELRESF,CONSTK,NEN,NINT,NROWSH,NESD,NROWB,NEE)*****; TRUSKD
C....PROGRAM TO FORM INTERNAL FORCE (" -K*D") FOR THE      TRUSKD
C      THREE-DIMENSIONAL, ELASTIC TRUSS ELEMENT               TRUSKD
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                  TRUSKD
C....DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION TRUSKD
C      DIMENSION W(1),DET(1),SHG(NROWSH,NEN,1),XS(NESD,1),XL(NESD,1),    TRUSKD
C      &           B(NROWB,1),DL(1),WORK(1),ELRESF(1)                 TRUSKD
C....LOOP ON INTEGRATION POINTS                           TRUSKD
C      DO 100 L=1,NINT                                     TRUSKD
C      TEMP = CONSTK*W(L)*DET(L)                         TRUSKD
C....SET UP THE STRAIN-DISPLACEMENT MATRIX             TRUSKD
C      CALL TRUSB(B,SHG(1,1,L),XS,NEN,NESD,NROWB,NROWSH)    TRUSKD
C....CALCULATE STRAIN                                 TRUSKD
C      STRAIN = RCDOT(B,DL,NROWB,NEE)                    TRUSKD
C....CALCULATE STRESS                               TRUSKD
C      STRESS = C*STRAIN                                TRUSKD
C....CALCULATE ELEMENT INTERNAL FORCE                TRUSKD
C      STRESS = TEMP*STRESS                            TRUSKD
C      CALL SMULT(STRESS,B,WORK,NROWB,1,1,NEE,1)        TRUSKD
C      CALL MATADD(ELRESF,WORK,ELRESF,NEE,NEE,NEE,NEE,1,1) TRUSKD
C 100 CONTINUE                                              TRUSKD
C      RETURN                                              TRUSKD
C      END                                                 TRUSKD
C*****SUBROUTINE TRUSPR(RHO,RDAMPM,RDAMPK,AREA,C,NUMAT)*****; TRUSPR
C....PROGRAM TO READ, WRITE AND STORE PROPERTIES FOR      TRUSPR
C      THREE-DIMENSIONAL, ELASTIC TRUSS ELEMENT            TRUSPR
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                  TRUSPR
C....DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION TRUSPR
C      DIMENSION RHO(1),RDAMPM(1),RDAMPK(1),AREA(1),C(1)    TRUSPR
C      COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE TRUSPR
C      COMMON /IOUNIT/ IIN,IOUT,IR$IN,IR$OUT                 TRUSPR
C      DO 100 N=1,NUMAT                                    TRUSPR
C      IF (MOD(N,50).EQ.1) WRITE(IOUT,1000) NUMAT          TRUSPR
C      READ(IIN,2000) M,E,RHO(M),RDAMPM(M),RDAMPK(M),AREA(M) TRUSPR
C      WRITE(IOUT,3000) M,E,RHO(M),RDAMPM(M),RDAMPK(M),AREA(M) TRUSPR
C      C(M) = E                                         TRUSPR
C 100 CONTINUE                                              TRUSPR
C      RETURN                                              TRUSPR
C 1000 FORMAT('1', //5X,'MATERIAL SET DATA', //5X,'NUMBER OF MATERIAL SETS', //5X,'SET', //5X,'YOUNG', //5X,'MASS', //5X,'MASS', //5X,'STIFFNESS', //5X,'AREA', //5X,'NUMBER', //5X,'MODULUS', //5X,'DENSITY', //5X,'DAMPING', //5X,'DAMPING', //) TRUSPR
C 2000 FORMAT(15.5X,7F10.0)                                TRUSPR
C 3000 FORMAT(4X,I5,3X,5(2X,1PE10.4))                   TRUSPR
C      END                                                 TRUSPR
C*****SUBROUTINE TRUSPT(XINT,STRESS,FORCE,STRAIN,NN,NNTOT,NEG,NEL,LINT)*****; TRUSPT
C....PROGRAM TO PRINT STRESS, STRAIN AND FORCE FOR THE    TRUSPT

```

```

THREE-DIMENSIONAL, ELASTIC TRUSS ELEMENT           TRUSPT   4
IMPLICIT DOUBLE PRECISION (A-H,O-Z)               TRUSPT   5
.. DEACTIVATE ABOVE CARD(S) FOR SINGLE-PRECISION OPERATION TRUSPT   6
DIMENSION XINT(3)                                TRUSPT   7
COMMON /IOUTUNIT/ IIN,IOUT,IRSI,IRSOUT            TRUSPT   8
NN = NN + 1                                      TRUSPT   9
IF (MOD(NN,NNTOT),EQ.1) THEN                   TRUSPT  10
  WRITE(IOUT,1000) NEG                         TRUSPT  11
  NN = 1                                         TRUSPT  12
ENDIF                                           TRUSPT  13
WRITE(IOUT,2000) NEL,LINT,XINT,STRESS,FORCE,STRAIN TRUSPT  14
RETURN                                          TRUSPT  15
DO FORMAT('1',
  & ELEMENT STRESSES AND STRAINS', //5X,          TRUSPT  16
  & ELEMENT GROUP NUMBER . . . X1 . . . X2 . . . (NEG) = , I5//,
  & ELEMENT INT. PT.      X1 . . . X2 . . . X3      TRUSPT  17
  & STRESS FORCE STRAIN /          , 5X,           TRUSPT  18
  & NUMBER NUMBER COORD. COORD. COORD. COORD. ')  TRUSPT  19
00 FORMAT(/2X,I5,7X,I2,8X,3(1PE10.2),5X,3(1PE10.2)) TRUSPT  20
END                                              TRUSPT  21
***** SUBROUTINE TRUSR(SHG,XS,B,DL,STRAIN,C,STRESS,FORCE,AREA,
  & NROWSH,NESD,NROWB,NEN,NEE)                  TRUSPT  22
  & TRUSR  23
.. PROGRAM TO CALCULATE STRESS, STRAIN AND FORCE AT AN INTEGRATION POINT
  & FOR THE THREE-DIMENSIONAL, ELASTIC TRUSS ELEMENT      TRUSPT  24
  & IMPLICIT DOUBLE PRECISION (A-H,O-Z)                   TRUSPT  25
.. DEACTIVATE ABOVE CARD(S) FOR SINGLE PRECISION OPERATION TRUSPT  26
DIMENSION SHG(NROWSH,1),XS(NESD,1),B(NROWB,1),DL(1)      TRUSPT  27
COMMON /CONSTS/ ZERO,PT1667,PT25,PT5,ONE,TWO,THREE,FOUR,FIVE TRUSPT  28
.. SET UP STRAIN-DISPLACEMENT MATRIX                 TRUSPT  29
  CALL TRUSB(B,SHG,XS,NEN,NESD,NROWB,NROWSH)          TRUSPT  30
.. CALCULATE STRAIN                                  TRUSPT  31
  STRAIN = RCDOT(B,DL,NROWB,NEE)                      TRUSPT  32
.. CALCULATE STRESS                                 TRUSPT  33
  STRESS = C*STRAIN                                  TRUSPT  34
.. CALCULATE FORCES                                TRUSPT  35
  FORCE = AREA*STRESS                             TRUSPT  36
RETURN                                           TRUSPT  37
END                                              TRUSPT  38

```

References Section 11.2

1. E. Cuthill, "Several Strategies for Reducing the Bandwidth of Matrices," in *Sparse Matrices and Their Applications*, eds. D. J. Rose and R. A. Willoughby. New York: Plenum Press, 1972.
2. W.-H. Liu and A. H. Sherman, "Comparitive Analysis of the Cuthill-McKee and the Reversed Cuthill-McKee Ordering Algorithms for Sparse Matrices," *SIAM Journal on Numerical Analysis*, 13 (1976), 198–213.
3. N. E. Gibbs, W. G. Poole, Jr., and P. K. Stockmeyer, "An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix," *SIAM Journal on Numerical Analysis*, 13 (1976), 236–250.
4. N. E. Gibbs, W. G. Poole, and P. K. Stockmeyer, "A Comparison of Several Bandwidth and Profile Reduction Algorithms," *Association for Computing Machinery Transactions of Mathematical Software*, 2 (1976), 322–330.
5. G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore: The Johns Hopkins

Index

- Absolute stability, 525
Accuracy, 462
Active column equation solver, 554, 633
Algorithm for constructing interpolation functions, 176–77
Algorithmic damping ratio, 505
 α -method, 532
Amplification factor, 466
Amplification matrix, 492
Amplitude decay, 505
Assembly algorithm, 43
Assembly operator, 44
A-stable, 525
Aubin-Nitsche method, 190
Augmented matrix, 32
Average acceleration method, 494–95
Axial force, 367
Axial strain, 367
Axisymmetric shells (*see* Shells of revolution, rings and tubes)
Axisymmetry, 101–3
- Bandwidth, 23
 \bar{B} -approach, 232
Barlow curvature points, 50
Barlow stress points, 31
Basis, 463
Basis functions, 9
Bazzi-Anderheggen ρ -method, 551
Beams (*see also* Bernoulli-Euler beam theory):
assumptions, 363–64
cross-section properties, 367
element stiffness matrix and load vector, 375
local-global transformations, 367
matrix formulation, 373
strain-displacement equations, 366
strong form, 371
variational equation, 369
weak form, 372
Bending moments, 367
Bernoulli-Euler beam theory, 48–51
Best approximation property, 186
Bilinear quadrilateral element, 112
Biquadratic Lagrange element, 129
Blank common, 633
- Block power method, 577
Body force, 76
Bossak's method, 550
Boundary, 59
Boundary conditions, 2
Boundary heat flux calculations 107
Boundary traction calculations, 107
Brick elements, 123, 136
BTCS method, 480
Bubble function, 130, 134
Bubnov-Galerkin method, 8
Bulk modulus, 192
- $C^k(\Omega)$, 52
 $C_b^k(\Omega)$, 52
Capacity, 419
Capacity matrix, 422
Cauchy stress tensor, 76
 C^0 -elements, 110
 C^1 -elements, 110
Central difference method, 494–95
Chain rule, 44
Change of variables formula:
Dirac delta function, 158
one dimension, 44
three dimension, 140
two dimensions, 138

characteristic velocity, 510
 Cholesky decomposition, 644
 circular plates, 328–32, 339–42,
 346, 347, 350, 358
 used unit interval, 2
 efficient of heat transfer, 71
 factors, 149–50
 allocation schemes, 530
 impacted column equation
 solver, 554
 impacted column storage, 633
 incompatible elements, 110
 completeness of finite element
 functions, 110–11
 completeness of function spaces,
 265
 conditional consistency, 481
 conditional stability, 466
 conductivities, 60
 conductivity matrix, 60
 forming elements, 110
 conservation of total energy, 457
 consistency, 462
 inconsistent mass, 436, 507
 constitutive equation:
 elastostatics, 76
 heat conduction, 60
 strained media problems, 192
 constrained variational problem,
 194
 constraint ratio, 209, 223, 289,
 324
 continuously differentiable functions, 52
 continuous pressure elements,
 215–16
 convection-diffusion equation,
 161
 convergence, 462
 convergence criterion for eigenvalues, 592
 correctors, 553, 562, 657
 couples, 367
 crack elements, 159 (*see also*
 Singular elements)
 crank-Nicolson method, 460, 480
 creeping flow, 193
 critically damped, 524
 critical sampling frequency, 493
 crossed triangles, 224, 286
 our factorization, 485, 636
 cubic beam element, 520
 cubic four-node element in one
 dimension, 128
 cubic Hermite shape functions, 49
 cubic triangular element, 169

Curvature, 50, 314, 367

 Dahlquist's theorem, 525
 Data structure, 633
 Deflation, 626
 Degeneration, 120, 125, 126,
 180–81
 Density, 419, 423
 Derivatives of shape functions,
 146–50, 174
 Destination array (*see* ID array)
 Deviatoric components, 233
 Diagonal scaling, 642
 Dilatational components, 233
 Dipole, 50
 Dirac delta function, 24, 158
 Direct stiffness method, 41
 Discontinuous pressure elements,
 210–14
 Discrete Kirchhoff approach, 359
 Discrete Poisson equation for
 pressure, 203
 Discrete Rayleigh-Ritz approxi-
 mation, 585
 Discrete Rayleigh-Ritz reduction,
 574
 Discretization, 7, 65
 Displacement, 366
 Displacement difference equation
 form, 527
 Displacement vector, 11, 76
 Distributions, 25
 Divergence theorem, 60
 DKQ, 359, 361, 362
 DKT, 361
 DLEARN coding techniques, 632
 DLEARN examples:
 dynamic analysis of a plane
 strain cantilever beam:
 description, 705–6
 input file, 706
 output, 717–29
 implicit-explicit dynamic analy-
 sis of a rod:
 description, 715–16
 input file, 717
 output file, 717–29
 planar truss:
 description, 691–92
 input file, 692
 output, 693–704
 static analysis of a plane strain
 cantilever beam:
 description, 705
 input file, 705

DLEARN input instructions:
 boundary conditions data, 672
 coordinate data, 667
 element data:
 three-dimensional, elastic
 truss element, 687
 two-dimensional, isotropic
 elasticity element, 680
 execution control, 664
 input data echo, 663
 kinematic initial condition data,
 678
 load-time functions, 677
 nodal history data, 667
 prescribed nodal forces and
 kinematic boundary condi-
 tions, 673
 time sequence data, 666
 DLEARN program listing,
 734–96
 DLEARN program structure:
 global control, 651
 initialization phase, 651–53
 solution phase, 653–59
 DLEARN storage in blank com-
 mon:
 dynamic analysis data, 646
 equation system data, 650
 static analysis data, 647
 time sequence and time history
 data, 645–46
 DLEARN storage requirements
 for four-node element,
 647–50
 DLEARN subroutine index,
 729–34
 Domain, 1
 Doubly curved shells:
 element force vector, 396
 element stiffness matrix, 396
 fiber coordinate systems, 387
 geometry, 384
 kinematics, 388
 lamina coordinate systems, 385
 reduced constitutive equation,
 389
 strain-displacement matrix, 392
 stress resultants, 399
 Douglas, 27
 Drilling degrees of freedom, 404
 Driven cavity flow, 230–31,
 282–85
 DuFort-Frankel method, 481
 Dupont, 27
 Dynamic storage allocation,
 633–44

- Eigenvalue problems:
 buckling of a thin beam, 431
 free vibration of an elastic rod, 430
 free vibration of a thin beam, 433
 generalized, 570
 standard, 571
 standard error estimates, 433
- Elastic coefficients, 76
- Elastic membrane, 428
- Elastodynamics (*see* Hyperbolic problems)
- Elastostatics:
 axisymmetric formulation, 101–3
 element displacement vector, 91–92
 element force vector, 90
 element stiffness matrix, 90
 element strain-displacement matrix:
 axisymmetric case, 102
 three-dimensional case, 90
 two-dimensional case, 90
- Galerkin formulation, 84
 matrix formulation, 87
 strong form, 77
 summary of important equations, 98–99
 weak form, 78
- Element body forces, 162–63
- Element boundary forces, 161–63
- Element-by-element (EBE) implicit methods, 483
- Element force vector, 41
- Element groups, 633
- Element nodes array (*see* IEN array)
- Elements, 20
- Element stiffness implementation, 151–56
- Element stiffness matrix, 41
- Elements with variable numbers of nodes, 132–35
- Empty set, 58
- Energy inner product, 186, 273
- Energy method (*see* Stability via the energy method)
- Energy norm, 186, 273
- Energy stability, 472
- Enriched bilinear displacements–constant pressure quadrilateral, 259
- Equation of motion, 423
- Equilibrium equations, 77
- Equivalence theorem, 221, 330
- Error, 186
- Error equation, 470
- Error estimates:
 elliptic boundary-value problems, 189
 elliptic eigenvalue problems, 433
 semidiscrete Galerkin approximations, 456
- Error in the derivative, 29
- Essential boundary conditions, 6
- Estimation of eigenvalues, 452
- Euclidean basis vectors, 85–86
- Euclidean decomposition of a second-rank tensor, 78
- Euler–Lagrange equations, 5
- Explicit methods, 461
- Explicit predictor–corrector methods, 553
- Exponential shape functions, 47
- Factorization, 637
- Fiber, 384
- Fiber numerical integration, 398
- Finite difference equations, 479
- Finite difference stencil, 31
- Finite element, 20
- Finite element domain, 20
- Finite Taylor expansion, 28
- Flop, 642
- Force vector, 11
- Forward difference method, 460
- Forward Euler method, 460
- Forward reduction, 32, 639
- Fourier coefficients, 463
- Fourier law, 60
- Fox–Goodwin method, 493
- Fractional-step algorithm, 474
- Frames (*see* Beams)
- FTCS method, 479
- Function spaces, 8
- Fundamental lemma of the calculus of variations, 6
- Galerkin equation, 9
- Galerkin method, 8
- Gauss elimination:
 example, 35
 hand-calculation algorithm, 33
- Gaussian quadrature (*see* Numerical integration)
- Gear's methods, 526
- Generalized derivative, 17
- Generalized displacements, 243
- Generalized Fourier law, 60
- Generalized functions, 25
- Generalized Hooke's law, 76
- Generalized Jacobi method, 578
- Generalized solution, 4
- Generalized step function, 21
- Generalized trapezoidal methods:
 commutative diagram, 465
 convergence, 468
 equations, 460
 implementations, 460–61
 modal reduction to SDOF form 462
- SDOF model problem, 464
- stability, 465–67
- Geometric stiffness, 432
- Ghost eigenvalues, 594
- Givens method, 572, 619
- Green's function, 25
- Growth/decay estimates, 457
- $H^k(\Omega)$, 54
- Half-bandwidth, 23
- Heat conduction:
 axisymmetric formulation, 101
 element force vector, 69
 element stiffness matrix, 69
 element temperature vector, 71
 Galerkin formulation, 64
 matrix formulation, 67
 strong form, 61
 summary of important equations, 99–100
 weak form, 61
- Heat equation, 61, 419, 422
- Heat flux, 107
- Heat flux vector, 60
- Heat supply, 60
- Heaviside function, 25
- Hermite shape functions, 49
- Hermitian matrix, 564
- Heterosis plate element, 335
- Heterosis shell element, 401
- Higher-order elements, 126
- Higher-order mass, 446, 507
- Hilber–Hughes–Taylor method (*see* α -method)
- Hilbert projection theorem, 280
- Hilbert space, 266
- Homogeneity:
 elastic coefficients, 155
 elastostatics, 76
 heat conduction, 60
- Hooke's law, 76

ubolt's method, 529
urglass modes, 239, 254
urglass stabilization operator, 254
urglass stiffness, 251
useholder-QR method, 572, 582
drostatic pressure, 193
perbolic problems:
matrix formulation, 424
emidiscrete Galerkin formula-
tion, 424
strong form, 423
weak form, 423

array:
definition:
 elastostatics, 85
 heat conduction, 66
example:
 elastostatics, 94
 heat conduction, 72–73

N array:
definition, 71

example:
 elastostatics, 94
 heat conduction, 72–73
plicit-explicit element mesh
 partitions, 461
plicit-explicit methods, 553
plicit methods, 461
ompatible elements, 110, 243
ompatible modes, 243
ompressible elasticity, 192–93
lex-free notation, 63
rtial inner product, 584
initesimal rigid-body motions, 88

initesimal strain tensor, 76
tial condition, 418
tial strain, 105
tial stress, 104–5
tial-stress stiffness matrix, 104, 432
ier product, 264
egration by parts, 60
terior element boundaries, 68
terpolation estimate, 189
terpolation functions, 9
terpolation property, 114
verse function theorem, 119
verse iteration, 579
ons-Guyan reduction, 576
OFLEX, 361

Isoparametric elements, 118, 271
Isotropy:
 elastic coefficients, 155
 elastostatics, 83
 heat conduction, 60

Jacobian determinant, 119
Jacobi method, 572
Joints, 20

Kinematic boundary conditions, 563, 655
Kinematic condition of incom-
pressibility, 193
Kinetic energy, 512
Kirchhoff mode concept, 324, 353
 k,m -regular, 189, 269
Knots, 20
Kronecker delta, 21
Krylov sequence, 586

$L_2(\Omega)$, 54
Lagrange elements, 130, 138, 139
Lagrange-multiplier method, 195
Lagrange plate elements, 327
Lagrange polynomials, 127, 176
Lagrange shell elements, 400
Lagrange-type interpolation over
tetrahedra, 171
Lagrange-type interpolation over
triangles, 166–69

Lamé parameters, 83, 192
Lamina, 384
Lanczos algorithm, 582–90
 example, 590
 summary (table), 588
Lanczos vectors, 586
LANSEL eigenvalue package, 600–29

Lax equivalence theorem, 470
LBB condition, 208
Leap frog method, 480
Least squares, 227
Limitation principle, 226
Linear acceleration method, 493
Linear multistep (LMS) methods
 for first-order equations, 523
Linear multistep (LMS) methods
 for second-order equations, 526

Linear nonconforming triangle, 250
Linear one-dimensional finite ele-
ment, 37
Linear spaces, 263
Linear tetrahedral element, 126,
170
Linear triangular element, 120,
167
Linear triangular plate element,
355

LM array:
 definition:
 elastostatics, 92
 heat conduction, 72
 one-dimensional model prob-
 lem, 42

example:
 elastostatics, 94
 heat conduction, 72–73

Lobatto element, 440
Lobatto quadrature, (see Numeri-
cal integration)
Local spurious modes, 287
Local truncation error, 468, 529
Location matrix (see LM array)
Locking, 323
Locking element, 293
LORA, 345, 351
Loss of orthogonality, 595
Lumped mass, 436–45, 507
 nodal quadrature, 436
 row-sum, 444
 special lumping, 445

Macaulay bracket, 25
Macroelement, 224, 259
Mass, consistent (see Consistent
mass)
Mass, higher-order (see Higher-
order mass)
Mass, lumped, 436–45, 507 (see
also Lumped mass)
Mass matrices for shell elements,
564
Mass matrix, 426
Matched methods, 505
Matrix equations, 11
Mean-dilatation approach, 232
Mean incompressibility, 161
Mean-value theorem, 28
Mechanisms, 240
Memory manager, 644

- Memory pointer dictionary, 631, 644
 Mesh, 7
 Mesh locking, 208
 Mesh parameter, 189
 Mesh partitions, 552
 Midpoint rule, 460
 Minimum potential energy principle, 188
 Misconvergence, 594
 Mixed boundary-value problem of linear elastostatics, 77
 Mixed formulation of elasticity: element arrays, 204–6
 Galerkin formulation, 200
 matrix formulation, 200–204
 strong form, 198
 weak form, 199
 Mixed method, 195, 197
 Modal analysis, 487, 540
 Moment tensor, 314
 Multicorrector iteration, 656
 Multiple eigenvalues, 595
- Natural boundary conditions, 6
 Natural coordinates, 112
 Natural norm, 265
 Nearly incompressible case, 217
 Newmark method: commutative diagram, 494
 displacement-difference equation form, 527
 equations, 490–91
 error equation, 496
 high-frequency behavior, 498–500
 implementation, 491
 predictors, 491
 stability conditions, 492–93
 truncation error, 496
 viscous damping, 500
 Newton's law of heat transfer, 71
 Nodal points, 20
 Nodes, 20
 Nonconforming elements, 110, 242
 Nonlocking element, 293
 Norm, 265
 Numerical dispersion, 504
 Numerical dissipation, 504
 Numerical integration: Gaussian quadrature, 141–45
 Lobatto quadrature, 440
 rules for tetrahedra, 172, 174
 rules for triangles, 172–74
 Simpson's rule, 141
 trapezoidal rule, 140
- One-dimensional model problem: element force vector, 41
 element stiffness matrix, 41
 Galerkin formulation, 9
 matrix formulation, 11
 strong form, 3
 summary of important equations, 100
 weak form, 4
 One-step multivalue methods, 492
 One-to-one, 118
 Onto, 118
 Open set, 57–59
 Open unit interval, 2
 Operator splitting, 552
 Optimal collocation methods, 531
 Optimally constrained, 300
 Order of accuracy, 30, 468
 Order of convergence, 30
 Orthogonality, 264, 571
 Orthonormality, 571
 Overconstrained, 300
 Overdamped, 524
 Overshoot, 537
- Parabolic problems: matrix formulation, 421
 semidiscrete Galerkin formulation, 420
 strong form, 419
 weak form, 419
- Parallel processing, 486
 Parent domain, 112
 Parent tetrahedron, 170
 Parent triangle, 165
 Park's method, 526
 Partitioned form, 573
 Pascal triangles, 139
 Patch test, 238, 248, 256, 259–61
 Penalty formulation of incompressible elasticity, 217, 289
 Penalty method, 196
 Pergola roof, 334
 Perpendicular, 264
 Petrov-Galerkin method, 9
 Pinched cylinder, 401
 Plane strain, 83, 103, 237
 Plane stress, 83, 103
- Plate theory (*see* Reissner-Mindlin plate theory)
 Poisson-Kirchhoff theory, 310, 311
 Poisson's ratio, 83
 Positive-definite matrix, 23
 Post processing, 107
 Potential energy, 188
 Preconditioned conjugate gradients (*table*), 485
 Predictor-corrector algorithms, 473, 476
 Predictor-multicorrector algorithms, 562
 Predictors, 553, 562, 654–55
 Prescribed boundary displacements, 77
 Prescribed boundary heat flux, 61
 Prescribed boundary temperature, 61
 Prescribed boundary tractions, 77
 Pressure modes, 207, 277
 Pressure smoothing, 227
 Principal invariants, 498, 528
 Principal roots, 529
 Profile, 554, 633
 Projects, 261–62
 Pseudonormal, 384
- QUAD4, 345, 351, 362, 395
 Quadratic tetrahedral element, 171
 Quadratic three-node element in one dimension, 128
 Quadratic triangular element, 136, 168
 Quasi-uniform, 269
- Range, 1
 Rank check, 257, 637
 Rank deficiency, 191, 239, 278, 332–34
 Rate of convergence, 468
 Rayleigh damping, 426, 492
 Rayleigh quotient, 435, 452
 Rectangular plates, 338–40, 346–49, 363
 Reduced integration, 221, 327
 Reduced integration beam elements, 376
 Reduced system, 571
 Region of absolute stability, 525
 Regular, 269
 Regularized element array, 486

- Esner-Mindlin plate theory:
 assumptions, 310
 boundary conditions, 324–26
 constitutive equation, 313
 convergence criteria, 322
 element stiffness matrix and
 kinematics, 311–12
 matrix formulation, 319
 rain-displacement equations,
 strong form, 317
 variational equation, 315
 weak form, 318
 active error, 36
 active period error, 505
 ordered Crout EBE preconditioner, 486
 idual, 585
 idual bending flexibility, 378
 idual forces, 656
 toring orthogonality, 598
 urn of banished Ritz vectors,
 ombic plates, 346–47, 351,
 gs (*see* Shells of revolution,
 z value, 585
 z vector, 585
 tation, 314, 366
 uth-Hurwitz criterion, 530
 yal road method (*see* Fox-Goodwin method)
- il'yev's method, 481
 warz inequality, 264
 OF model problem, 464
 ctor element, 159–60
 ective orthogonalization methods, 598
 ective reduced integration,
 MILOOF, 361
 niorthogonality condition, 598
 endipity elements, 135, 138,
 t closure, 58
 t intersection, 58
 t union, 58
 ape functions, 9, 112–37,
 quadrilaterals and bricks,

tetrahedra, 170–71
 triangles, 165–70
 Shape function subroutines,
 Shear constraints, 323
 Shear correction factors, 391
 Shear force, 314, 367

Newmark methods, 556
 predictor-corrector methods,

Standard element families,

Standard error estimate, 190
 Statically condensed elastic coefficient matrix, 103
 Static condensation, 246, 573
 Static load patterns, 574
 Stiffly stable, 526
 Stiffness matrix, 11
 Stokes flow, 193
 Strain-displacement equations, 76
 Strain energy, 187, 512
 Strain projection, 232, 236
 Strain tensor, 76

- two-node linear rod element, 513–14
T1, 342, 362
Torsionless axisymmetric analysis, 236
Torsionless axisymmetric case, 101
Total energy, 512
Total potential energy function, 194
Tractions, 77
Transition element, 159–60
Transverse displacement, 314
Trapezoidal rule (*see* Average acceleration method)
Trial solutions, 3
Trial vectors, 574, 585
Triangular coordinates, 166
Triangular elements, 121, 136, 138, 139, 167–69, 180–81
Tridiagonal matrix, 589
Trilinear hexahedral element, 123
Truncation errors, 496
- Tubes (*see* Shells of revolution, rings and tubes)
Twist, 367
Twisted ribbon, 351, 353, 354
Twisting moment, 367
Two-point boundary-value problems, 2
- Unconditional stability, 466
Underconstrained, 300
Underdamped, 524
Unified single-step methods, 552
Uniform reduced integration, 221, 327, 414
Unit outward normal vector, 57–58
Unit roundoff error, 595
Unit step function, 25
Upper triangular matrices, 636
- Variational crimes, 8
- Variational equation, 4
Variations, 4
Virtual displacement principle, 78
Virtual work principle, 4, 78
Viscous damping, 426
Von Neumann method, 479, 521
- Wave equation, 427, 506
Weak solution, 4
Wedge-shaped elements, 125, 171–72
Weighted residual methods, 9
Weighting functions, 4
Wilson- θ method, 530
Winkler foundation, 428
- Young's modulus, 83
- Zero-energy modes (*see* Rank deficiency)