Programming Finite Elements in Java™

Gennadiy Nikishkov

Programming Finite Elements in Java™

Springer

Gennadiy Nikishkov, DSc, PhD
The University of Aizu
Aizu-Wakamatsu, 965-8580
Japan

ISBN 978-1-84882-971-8 e-ISBN 978-1-84882-972-5
DOI 10.1007/978-1-84882-972-5
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2009942247

Cover design: eStudioCalamar, Figueres/Berlin

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The finite element method can be applied to problems in various fields of science and engineering. It is well established and its algorithms are presented in numerous publications. Many books are devoted to different aspects of the finite element method. Still, algorithms of the finite element method are difficult to understand, and programming of the finite element techniques is complicated.

Objective

This book focuses on algorithms of the finite element method and their programming. First, general equations of the finite element method for solving solid mechanics and thermal conductivity problems are introduced. Then, algorithms of the finite element method and their programming in Java™ are considered. In addition to solution methods, the book presents algorithms and programming approaches for mesh generation and visualization.

Why Java?

The Java language is selected for its numerous advantages: an object-oriented paradigm, multiplatform support, ease of development, reliability and stability, the ability to use legacy C or C++ code, good documentation, development-tool availability, etc. The Java runtime environment always checks subscript legitimacy to ensure that each subscript is equal to or greater than zero and less than the number of elements in the array. Even this simple feature means a lot to developers. As a result, Java programs are less susceptible to bugs and security flaws. Java also provides application programming interfaces (APIs) for development of GUI, and three-dimensional graphics applications.

I started programming finite elements in Fortran. Later I used Pascal, C, and C++, before settling on Java. Comparing these languages I found that programming finite elements in Java is not just efficient because the productivity is higher and the code contains fewer errors, but is also more pleasant.

An opinion exists that Java is not suitable for computational modeling and finite element programming because of its slow execution speed. It is true that Java is slower than C in performing "multiply-add" arithmetic inside double and triple loops. However, tuning of important Java code fragments provides computational speed comparable to that of C.

The attractive features of Java prevail over some of its drawbacks. In my opinion, Java is good for both learning finite element programming and for finite element code development with easy debugging, modification, and support. Further, Java is easy to understand even for those who do not program in Java. In most cases, methods performing computations can be easily used with minimum modification to procedures written in other languages such as C or C++.

For Whom Is This Book Written?

This book is an introductory text about finite element algorithms and especially finite element programming using an object-oriented approach. All important aspects of finite element techniques are considered – finite element solution, generation of finite element meshes, and visualization of finite element models and results.

The book is useful for graduate and undergraduate students for self-study of finite element algorithms and programming techniques. It can be used as a textbook for introductory graduate courses or in advanced undergraduate courses. I hope that the book will be interesting to researchers and engineers who are already familiar with finite element algorithms and codes, since the programming approaches of this book differ from other publications.

Organization

The book is organized into four parts. Part I covers general formulation of the finite element method. Chapter 1 introduces the finite element formulation in the one-dimensional case. Both the Galerkin method and variational formulations are considered. Chapter 2 presents finite element equations for heat transfer problems derived with the use of the Galerkin approach. Chapter 3 contains variational formulation of general finite element equations for solid mechanics problems. An object-oriented approach to development of the finite element code is discussed in Chapter 4.

Part II is devoted to algorithms and programming of the finite element solution of solid mechanics problems. Chapter 5 considers the class structure of the finite ele-

ment processor code. Data structures of the finite element model and corresponding Java class are presented in Chapter 6. Relations for elastic material and the corresponding class are given in Chapter 7. Chapters 8 and 9 describe an abstract class for a finite element and a class for numerical integration. Chapters 10–13 present algorithms and programming implementation for two- and three-dimensional isoparametric quadratic elements. Assembly and solution of the finite element equation system are discussed in Chapters 14–16. Chapter 17 is devoted to assembly of the global load vector. Computing stress increments is presented in Chapter 18. Solution and programming implementation of elastic–plastic problems is discussed in Chapter 19.

Part III focuses on mesh generation for solution of two- and three-dimensional finite element problems. The block decomposition method used for mesh generation and general organization of the mesh generator is given in Chapter 20. Chapter 21 presents two-dimensional mesh generators. Chapter 22 describes three-dimensional mesh generation by sweeping a two-dimensional mesh. Chapters 23–25 contain algorithms and classes for pasting mesh blocks and various operations on mesh blocks, including their pasting for creation of a complex mesh of simple blocks.

Part IV describes algorithms for visualization of finite element methods and results. Chapter 26 introduces the Java 3D^{TM} API, which is used for rendering three-dimensional objects. Visualization algorithms for higher-order finite elements and visualization code structure are presented in Chapter 27. A scene graph for visualization of meshes and results is discussed in Chapter 28. Chapter 29 describes algorithms for creation of the model surface. Chapters 30 and 31 are devoted to subdivision of the model surface into polygons. Chapter 32 presents Java classes for results field, color-gradation strip, mouse interaction and lights.

Appendices A, B, and C contain brief instructions for preparing data for finite element programs that perform problem solution, mesh generation, and visualization of models and results. Appendix D provides examples of finite element analysis: mesh generation, problem solution, and visualization of results.

How This Book Differs from Others

There are many books about the finite element method. Some of them contain finite element program segments. Two qualities distinguish this book from other books on the finite element method.

First, programming in this book is based upon the Java programming language. In my opinion, Java is well suited for explaining programming of the finite element method. It allows compact and simple code to be written. This helps greatly because the reader has a chance to actually read finite element programs and to understand them.

Secondly, algorithms presented in this book are tightly connected to programming. The book is written around one finite element Java program, which includes solution of solid mechanics boundary value problems, mesh generation, and visu-

alization of finite element models and results. Presentation of computational algorithms is followed by a Java class or group of Java methods and then accompanied by code explanation.

Web Resources

The Java programs and examples presented in this book are available on the Web: `http://www.springer.com/978-1-84882-971-8`. Comments, suggestions, and corrections are welcome by e-mail: `fem.java@gmail.com`.

About the Author

Gennadiy Nikishkov got his Ph.D. and D.Sc. degrees from the Moscow Engineering Physics Institute (Technical University) in computational mechanics. He held a Professor position at the Moscow Engineering Physics Institute. He also had visiting positions at Georgia Institute of Technology (USA), Karlsruhe Research Center (Germany), RIKEN Institute of Physical and Chemical Research (Japan), GKSS Research Center (Germany), and the University of California at Los Angeles (USA). Currently, he is a Professor at the University of Aizu (Japan). His research interests include computational mechanics, computational fracture mechanics, computational nanomechanics, development of finite element and boundary element codes, scientific visualization, and computer graphics.

Acknowledgments

The author is grateful to Prof. Michael Cohen, University of Aizu, Japan, for his attentive reading of the book manuscript and his valuable suggestions, and to Dr. Yuriy Nikishkov, Georgia Institute of Technology, USA, for his advice related to development of the Java program. I also acknowledge my colleagues who contributed to this book through many discussions on computational methods over the years.

I thank the anonymous reviewers for their precious comments. Special thanks go to my editor Oliver Jackson and to Ms. Aislinn Bunning and Ms. Sorina Moosdorf for their help, which greatly improved the quality of the book. Finally, I express my loving appreciation to my wife Valentina for her support and encouragement.

Aizu-Wakamatsu, Japan *Gennadiy Nikishkov*
July 2009

Contents

Part I Finite Element Formulation

1 Introduction 3
1.1 Basic Ideas of FEM 3
1.2 Formulation of Finite Element Equations 4
1.2.1 Galerkin Method 5
1.2.2 Variational Formulation 8
1.3 Example of Shape-function Determination 9
Problems 10

2 Finite Element Equations for Heat Transfer 13
2.1 Problem Statement 13
2.2 Finite Element Discretization of Heat Transfer Equations 14
2.3 Different Type Problems 16
2.4 Triangular Element 17
Problems 19

3 FEM for Solid Mechanics Problems 21
3.1 Problem Statement 21
3.2 Finite Element Equations 23
3.3 Stiffness Matrix of a Triangular Element 26
3.4 Assembly of the Global Equation System 27
3.5 Example of the Global Matrix Assembly 29
Problems 30

4 Finite Element Program 33
4.1 Object-oriented Approach to Finite Element Programming 33
4.2 Requirements for the Finite Element Application 34
4.2.1 Overall Description 34
4.2.2 User Description 35
4.2.3 User Interface 35

4.2.4 Functions . . . 35
4.2.5 Other Requirements . . . 36
4.3 General Structure of the Finite Element Code . . . 36
Problems . . . 38

Part II Finite Element Solution

5 Finite Element Processor . . . 43
5.1 Class Structure . . . 43
5.2 Problem Data . . . 49
5.2.1 Data Statements . . . 49
5.2.2 Model Data . . . 51
5.2.3 Load Specification . . . 52
5.2.4 Data Example . . . 54
5.3 Data Scanner . . . 57
Problems . . . 61

6 Finite Element Model . . . 63
6.1 Data for the Finite Element Model . . . 63
6.2 Class for the Finite Element Model . . . 66
6.3 Adding New Data Item . . . 72
Problems . . . 72

7 Elastic Material . . . 75
7.1 Hooke's Law . . . 75
7.2 Class for a Material . . . 76
7.3 Class for Elastic Material . . . 79
Problems . . . 81

8 Elements . . . 83
8.1 Element Methods . . . 83
8.2 Abstract Class Element . . . 84
8.2.1 Element Data . . . 84
8.2.2 Element Constructor . . . 85
8.2.3 Methods of Particular Elements . . . 87
8.2.4 Methods Common to All Elements . . . 88
8.2.5 Container for Stresses . . . 90
8.3 Adding New Element Type . . . 91
Problems . . . 92

9 Numerical Integration . . . 93
9.1 Gauss Integration Rule . . . 93
9.2 Implementation of Numerical Integration . . . 95
Problems . . . 99

10 Two-dimensional Isoparametric Elements 101
10.1 Shape Functions 101
10.2 Strain–Displacement Matrix 104
10.3 Element Properties 107
10.4 Nodal Equivalent of the Surface Load 108
10.5 Example: Computing Nodal Equivalents of a Distributed Load 109
10.6 Calculation of Strains and Stresses 110
Problems 111

11 Implementation of Two-dimensional Quadratic Element 113
11.1 Class for Shape Functions and Their Derivatives 113
11.1.1 Element Degeneration 114
11.1.2 Shape Functions 115
11.1.3 Derivatives of Shape Functions 116
11.1.4 One-dimensional Shape Functions and Their Derivatives . . . 118
11.2 Class for Eight-node Element 118
11.2.1 Stiffness Matrix 119
11.2.2 Displacement Differentiation Matrix 121
11.2.3 Thermal Vector 122
11.2.4 Nodal Equivalent of a Distributed Load 123
11.2.5 Equivalent Stress Vector 125
11.2.6 Extrapolation from Integration Points to Nodes 126
11.2.7 Other Methods 127
Problems 128

12 Three-dimensional Isoparametric Elements 129
12.1 Shape Functions 129
12.2 Strain–Displacement Matrix 131
12.3 Element Properties 133
12.4 Efficient Evaluation of Element Matrices and Vectors 134
12.5 Calculation of Nodal Equivalents for External Loads 134
12.6 Example: Nodal Equivalents of a Distributed Load 136
12.7 Calculation of Strains and Stresses 138
12.8 Extrapolation of Strains and Stresses 138
Problems 139

13 Implementation of Three-dimensional Quadratic Element 141
13.1 Class for Shape Functions and Their Derivatives 141
13.1.1 Element Degeneration 141
13.1.2 Shape Functions 143
13.1.3 Derivatives of Shape Functions 144
13.1.4 Shape Functions and Their Derivatives for an Element Face . 147
13.2 Class for Twenty-node Element 149
13.2.1 Stiffness Matrix 150
13.2.2 Thermal Vector 152

13.2.3 Nodal Equivalent of a Distributed Load . . . 153
13.2.4 Equivalent Stress Vector . . . 154
13.2.5 Extrapolation from Integration Points to Nodes . . . 155
13.2.6 Other Methods . . . 156
Problems . . . 158

14 Assembly and Solution . . . 161
14.1 Disassembly and Assembly . . . 161
14.1.1 Disassembly of Vectors . . . 161
14.1.2 Assembly of Vectors . . . 163
14.1.3 Assembly Algorithm for Matrices . . . 164
14.2 Displacement Boundary Conditions . . . 166
14.2.1 Explicit Specification of Displacement Boundary Conditions 166
14.2.2 Method of Large Number . . . 167
14.3 Solution of Finite Element Equations . . . 167
14.4 Abstract Solver Class . . . 168
14.5 Adding New Equation Solver . . . 170
Problems . . . 171

15 Direct Equation Solver . . . 173
15.1 LDU Solution Method . . . 173
15.2 Assembly of Matrix in Symmetric Profile Format . . . 174
15.3 LDU Solution Algorithm . . . 178
15.4 Tuning of the LDU Factorization . . . 182
Problems . . . 186

16 Iterative Equation Solver . . . 187
16.1 Preconditioned Conjugate Gradient Method . . . 187
16.2 Assembly of Matrix in Sparse-row Format . . . 188
16.3 PCG Solution . . . 193
Problems . . . 196

17 Load Data and Load Vector Assembly . . . 199
17.1 Data Describing the Load . . . 199
17.2 Load Data Input . . . 201
17.3 Load Vector Assembly . . . 207
17.4 Element Face Load . . . 209
Problems . . . 211

18 Stress Increment, Residual Vector and Results . . . 213
18.1 Computing Stress Increment . . . 213
18.2 Residual Vector . . . 215
18.3 Results . . . 217
18.4 Solution of a Simple Test Problem . . . 219
Problems . . . 220

19 Elastic–Plastic Problems 223
19.1 Constitutive Relations for Elastic–Plastic Material 223
19.2 Computing Finite Stress Increments 225
19.2.1 Determining Elastic Fraction of Stress Increment 226
19.2.2 Subincrementation for Computing Stress Increment 226
19.3 Material Deformation Curve 227
19.4 Implementation of Elastic–Plastic Material Relations 228
19.5 Midpoint Integration of Constitutive Relations 234
19.6 Nonlinear Solution Procedure 239
19.6.1 Newton–Raphson Method 240
19.6.2 Initial Stress Method 241
19.6.3 Convergence Criteria 242
19.7 Example: Solution of an Elastic–Plastic Problem 243
Problems 245

Part III Mesh Generation

20 Mesh Generator 249
20.1 Block Decomposition Method 249
20.2 Class Structure 250
20.3 Mesh-generation Modules 252
20.4 Adding New Module 253
Problems 254

21 Two-dimensional Mesh Generators 257
21.1 Rectangular Block 257
21.2 Mesh Inside Eight-node Macroelement 261
21.2.1 Algorithm of Double-quadratic Transformation 261
21.2.2 Implementation of Mesh Generation 264
21.3 Example of Mesh Generation 269
Problems 270

22 Generation of Three-dimensional Meshes by Sweeping 271
22.1 Sweeping Technique 271
22.2 Implementation 272
22.2.1 Input Data 272
22.2.2 Node Numbering 275
22.2.3 Element Connectivities and Nodal Coordinates 276
22.3 Example of Mesh Generation 279
Problems 281

23 Pasting Mesh Blocks 283
23.1 Pasting Technique 283
23.2 Implementation 284
23.2.1 Data Input 284
23.2.2 Finding Coincident Nodes 286

23.2.3 Pasting . . . 287
Problems . . . 288

24 Mesh Transformations . . . 289
24.1 Transformation Relations . . . 289
24.2 Implementation . . . 291
24.2.1 Input Data . . . 291
24.2.2 Performing Transformations . . . 293
24.3 Example of Using Transformations . . . 295
Problems . . . 296

25 Copying, Writing and Reading Mesh Blocks . . . 297
25.1 Copying . . . 297
25.2 Writing Mesh to File . . . 299
25.3 Reading Mesh from File . . . 300
Problems . . . 302

Part IV Visualization of Meshes and Results

26 Introduction to Java 3D™ . . . 305
26.1 Rendering Three-dimensional Objects . . . 305
26.2 Scene Graph . . . 306
26.3 Scene Graph Nodes . . . 307
26.3.1 Group Nodes . . . 307
26.3.2 Leaf Nodes . . . 308
26.4 Node Components . . . 309
26.4.1 Geometry . . . 309
26.4.2 Appearance and Attributes . . . 311
Problems . . . 311

27 Visualizer . . . 313
27.1 Visualization Algorithm . . . 313
27.2 Surface of the Finite Element Model . . . 314
27.3 Subdivision of Quadratic Surfaces . . . 315
27.4 Class Structure of the Visualizer . . . 315
27.5 Visualizer Class . . . 316
27.6 Input Data . . . 318
27.6.1 Input Data File . . . 318
27.6.2 Class for Data Input . . . 319
Problems . . . 322

28 Visualization Scene Graph . . . 325
28.1 Schematic of the Scene Graph . . . 325
28.2 Implementation of the Scene Graph . . . 326
28.3 Shape Objects . . . 328
Problems . . . 331

29 Surface Geometry . . . 333
29.1 Creating Geometry of the Model Surface . . . 333
29.2 Surface Faces . . . 335
29.3 Surface Edges and Nodes . . . 338
29.4 Modification of Nodal Coordinates . . . 340
Problems . . . 342

30 Edge and Face Subdivision . . . 343
30.1 Subdivision for Quality Visualization . . . 343
30.2 Edge Subdivision . . . 344
30.3 Face Subdivision . . . 347
Problems . . . 352

31 Surface Subdivision . . . 353
31.1 Subdivision of the Model Surface . . . 353
31.2 Subdivision of Faces into Triangles . . . 356
31.3 Arrays for Java 3D . . . 359
Problems . . . 362

32 Results Field, Color Scale, Interaction and Lights . . . 363
32.1 Results Field . . . 363
32.2 Color Scale . . . 368
32.3 Mouse Interaction . . . 370
32.4 Lights and Background . . . 372
32.5 Visualization Example . . . 373
Problems . . . 375

A Data for Finite Element Solver . . . 377
A.1 Data Statements . . . 377
A.1.1 Data Statement . . . 377
A.1.2 Comment Statement . . . 377
A.1.3 Including File . . . 377
A.1.4 End Statement . . . 378
A.2 Model Data . . . 378
A.2.1 Parameters . . . 378
A.2.2 Material Properties . . . 378
A.2.3 Finite Element Mesh . . . 379
A.2.4 Displacement Boundary Conditions . . . 379
A.3 Load Specification . . . 380
A.3.1 Load Step Name . . . 380
A.3.2 Parameters . . . 380
A.3.3 Nodal Forces . . . 381
A.3.4 Surface Forces . . . 381
A.3.5 Surface Forces Inside a Box . . . 381
A.3.6 Nodal Temperatures . . . 382

B Data for Mesh Generation 383
B.1 Mesh-generation Modules 383
B.2 Rectangular Mesh Block 384
B.3 Mesh Inside Eight-node Macroelement 384
B.4 Three-dimensional Mesh by Sweeping 384
B.5 Reading Mesh from File 385
B.6 Writing Mesh to File 385
B.7 Copying Mesh 385
B.8 Mesh Transformations 385
B.9 Connecting Two Mesh Blocks 386

C Data for Visualizer 387
C.1 Visualization Data 387
C.2 Input Data 387

D Example of Problem Solution 389
D.1 Problem Statement 389
D.2 Mesh Generation 390
D.3 Problem Solution 391
D.4 Visualization 393

References 397

Index 399