



Series in Information and Computational Science

—80

Finite Element Language and Its Applications I

Liang Guoping (梁国平) Zhou Yongfa (周永发)
Translated by Gu Quan (古泉译)

(有限元语言及应用 I)



SCIENCE PRESS
Beijing



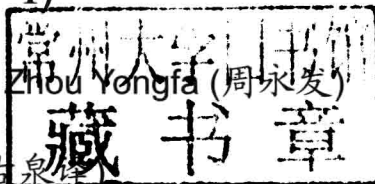
国家出版基金项目
NATIONAL PUBLICATION FOUNDATION

Series in Information and Computational Science 80

Finite Element Language and Its Applications I

(有限元语言及应用 I)

Liang Guoping (梁国平)



Translated by Gu Quan (古泉译)



SCIENCE PRESS
Beijing, China

Responsible Editors: Li Xin, Zhao Yanchao

Copyright© 2015 by Science Press
Published by Science Press
16 Donghuangchenggen North Street
Beijing 100717, P. R. China

Printed in Beijing

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of the copyright owner.

ISBN 978-7-03-046517-7

Editorial Board

Editor-in-Chief: Shi Zhongci

Vice Editor-in-Chief: Wang Xinghua Yu Dehao

Members:	Bai Fengshan	Bai Zhongzhi	Chen Falai
	Chen Zhiming	Chen Zhongying	Cheng Jin
	E Weinan	Guo Benyu	He Bingsheng
	Hou Yizhao	Shu C.-W.	Song Yongzhong
	Tang Tao	Wu Wei	Xu Jinchao
	Xu Zongben	Yang Danping	Zhang Pingwen

Preface to the Series

in Information and Computational Science

Since the 1970s, Science Press has published more than thirty volumes in its series Monographs in Computational Methods. This series was established and led by the late academician, Feng Kang, the founding director of the Computing Center of the Chinese Academy of Sciences. The monograph series has provided timely information of the frontier directions and latest research results in computational mathematics. It has had great impact on young scientists and the entire research community, and has played a very important role in the development of computational mathematics in China.

To cope with these new scientific developments, the Ministry of Education of the People's Republic of China in 1998 combined several subjects, such as computational mathematics, numerical algorithms, information science, and operations research and optimal control, into a new discipline called Information and Computational Science. As a result, Science Press also reorganized the editorial board of the monograph series and changed its name to Series in Information and Computational Science. The first editorial board meeting was held in Beijing in September 2004, and it discussed the new objectives, and the directions and contents of the new monograph series.

The aim of the new series is to present the state of the art in Information and Computational Science to senior undergraduate and graduate students, as well as to scientists working in these fields. Hence, the series will provide concrete and systematic expositions of the advances in information and computational science, encompassing also related interdisciplinary developments.

I would like to thank the previous editorial board members and assistants, and all the mathematicians who have contributed significantly to the monograph series on Computational Methods. As a result of their contributions the monograph series achieved an outstanding reputation in the community. I sincerely wish that we will extend this support to the new Series in Information and Computational Science, so that the new series can equally enhance the scientific development in information and computational science in this century.

Shi Zhongci
2005.7

Preface

This book is a new edition of the previous one *Finite Element Language*. A new part ‘applications’ has been added in the current version, as shown explicitly in the title. Finite Element Language (FEL) is a state of art modeling language used to solve partial differential equations (PDEs) by using finite element method (FEM) or finite volume method (FVM). This language is used to generate computer programs of FEM/FVM, by simply creating system-defined expressions for PDEs and its corresponding algorithms. The computer program of FEM/FVM (e.g., C or Fortran code) can be automatically generated by using the generator of this language.

When programming using FEL, the amount of code generated is reduced by more than 90 percent compared with that generated by other advanced language generators, thus it tremendously improves the efficiency of programming. Moreover, the system-defined expressions for PDFs and its algorithms are extremely easy for user to read, modify, update, re-use and maintain. FEL helps engineers and researchers to mainly focus on understanding their physics problems and creating the appropriate mathematical models by making them free from the tedious, time-consuming and error-prone coding work.

This book is organized as follows: Part I includes 5 chapters and appendix A - F. Chapter 1 discusses the description language for creating the expressions of PDFs. These expressions are used by the system to generate the element subroutines in FEM/FVM; Chapter 2 presents the fundamental method to create the FEM algorithms for solving problems in single-physics field; Contents presented in Chapter 3 are similar to that in Chapter 2 but involed with coupled problems in multi-physics fields; Chapter 4 introduces the strategy for building FEL which is based on the component-based-programming method. Details about five most commonly used component programs are also given; The FEM data structure is presented and discussed in Chapter 5. Appendix A to C provide some fundamental concepts and knowledge for finite element shap function and element types as well as the coordinate transformations and numerical integration. Appendix D and E contain the collection of keywords and some specific statements defined in FEL.

Part II includes six chapters, introducing the applications of FEL in solid mechanics, Navier-Stokes equation, Darcy flow, electromagnetic field, structural mechanics and thermal field problems, respectively. It is worth mentioning that the analytical examples in the book are only used to illustrate the specific applications of FEL,

some of the application results have not been strictly benchmarked so it is user's responsibility to perform the verification. The pre- and post-processing work is done based on FEPG.GID platform.

The history of FEL and the developed software FEPG can be tracked back to the 1980s. FEPG has been involved from the early version of only working on single CPU to the latest version which works on HPC and internet and provides user with very friendly GUI, thanks to the rapid development of modern simulation and high performance computing technologies.

The early users of FEPG have become its 'fans' and strong supporters or even participants. However, limited by the current situation of CAE industry in China, the promotion of FEL and FEPG face big challenges. We would like to take the opportunity when this book being published, to invite people in scientific computing community in China to join us in promoting FEPG and developing our own high performance finite element software.

Contents

Preface to the Series in Information and Computational Science

Preface

Introduction1

Chapter 1 Description Language of Differential Equation

Expression3

1.1 Writing PDE Files3

1.1.1 DEFI Information Segments4

1.1.2 FUNC Information Segments9

1.1.3 STIF Information Segments10

1.1.4 MASS Information Segments11

1.1.5 DAMP Information Segments12

1.1.6 LOAD Information Segments13

1.1.7 Insert Fortran Source Programs14

1.1.8 Examples17

1.2 Writing CDE Files18

1.2.1 DEFI Information Segments18

1.2.2 FUNC Information Segments21

1.2.3 SHIF Information Segments22

1.2.4 MASS Information Segments23

1.2.5 DAMP Information Segment23

1.2.6 LOAD Information Segments24

1.2.7 Insert Fortran Source Programs25

1.2.8 Examples25

1.3 Writing VDE Files26

1.3.1 Vector and Matrix Specification Statements27

1.3.2 ARRAY Specification Statements28

1.3.3 Tensor Operation Expressions28

1.3.4 Examples30

1.4 Writing FDE Files33

1.4.1 Writing Format of FDE Documents33

1.4.2 FVECT and FMATR Statements34

1.4.3 @l Operator Statements34

1.4.4 Common @l Operator Library35

1.4.5	@a Operator Statements	37
1.4.6	@w Operator Statements	38
1.4.7	@s Operator Statements	38
1.4.8	@r Operator Statements	38
1.4.9	Examples	39
1.5	Writing FBC Files	40
1.6	Writing GES Files	41
1.6.1	GES Files Structure	41
1.6.2	Rules for Writing GES Files	42
1.6.3	Examples	62
1.6.4	Element Subroutine	64
1.7	Writing GLT Files	66
1.7.1	DEFI Information Segments	66
1.7.2	VART Information Segments	68
1.7.3	Insert Fortran Source Programs	69
1.8	Writing Finite Volume Method Programs	70
1.8.1	GVS File Structure	71
1.8.2	Rules for Writing GVS Files	71
1.8.3	Rules for writing FVS Files	73
1.8.4	Basic flow of Finite Volume Method Program	76
1.8.5	Examples	76
1.8.6	FVS Files of Several Types of Elements	84
Chapter 2	Description Language of Algorithms in Single-physics Fields	99
2.1	NFE File Structure	99
2.2	Rules of writing NFE Files	100
2.2.1	DEFI Information Segments	100
2.2.2	COEF Information Segments	101
2.2.3	EQUATION Information Segments	101
2.2.4	SOLUTION Information Segments	105
2.2.5	Insert Fortran Source Programs	107
2.2.6	The Ending Character	107
2.3	NFE Algorithm Library	107
2.3.1	ell. nfe	108
2.3.2	nell. nfe	109
2.3.3	parb. nfe	111
2.3.4	par. nfe	111
2.3.5	nparb. nfe	112

2.3.6	npar. nfe	114
2.3.7	wave. nfe	117
2.3.8	wavev. nfe	118
2.3.9	newmark. nfe	119
2.3.10	waveexp. nfe	120
2.3.11	nwave. nfe	121
2.3.12	nnewmark. nfe	123
2.3.13	nwaveexp. nfe	126
2.3.14	str. nfe	127
2.3.15	nstr. nfe	127
2.3.16	hypls. nfe	128
2.3.17	cbsexp. nfe	129

**Chapter 3 Description Language of Coupling FEM Algorithms
in Multi-Physics Fields**

3.1	Writing GCN Files	132
3.1.1	Writing Pattern	132
3.1.2	Examples	133
3.2	GCN Library	134
3.3	Writing MDI Files	134
3.3.1	Writing Pattern	134
3.3.2	Examples	136

Chapter 4 Component-based Program Design Method

4.1	Finite Element Program Structure and Component-based Program Design Method	137
4.1.1	Program Structure	137
4.1.2	Component-based Program Design Method	138
4.2	Five Component Programs	144
4.2.1	START Component Program	144
4.2.2	BFT Component Program	153
4.2.3	E Component Program	160
4.2.4	SOLV Solver	175
4.2.5	U Component Program	199

Chapter 5 Data Structure of Finite Element Method

5.1	Component Description of Input Data for Finite Element Computation	205
5.1.1	Data Entry Form	205
5.1.2	Read/Write Format of Table Files	206
5.2	Input Data of FEM in Single-physics Problem	206

- 5.2.1 Coordinate Data Table206
 - 5.2.2 Table of Node Specification Number 207
 - 5.2.3 Table of Specified Nodal Displacement and Nodal Load
Information 207
 - 5.2.4 Table of Initial Values 208
 - 5.2.5 Element Information Data 208
- 5.3 Display and Query of FEM Input Data 209
- 5.4 PRE Files 209
 - 5.4.1 Linear Steady State Examples 211
 - 5.4.2 Nonlinear Transient Examples 214
 - 5.4.3 Examples of Coupling Problems in Multi-physics Fields 221
 - 5.4.4 Automatic Writing and Modification of Files 228
- 5.5 Results Display POS Files 229
- Appendix A Interpolation Functions and Element Types 231**
 - A.1 1D Lagrange Element 231
 - A.1.1 1D Linear Element 231
 - A.1.2 1D Quadratic Element 232
 - A.2 2D Lagrange Element 233
 - A.2.1 Four Nodal Rectangular Element 233
 - A.2.2 Nine-Node Rectangular Element 234
 - A.2.3 Eight Nodal Rectangular Element 236
 - A.2.4 Three Nodal Triangle Element 237
 - A.2.5 Six Triangle Element 238
 - A.3 3D Lagrange Element 239
 - A.3.1 Eight-node Hexahedral Element 240
 - A.3.2 27-node Hexahedral Element 240
 - A.3.3 20-node Hexahedral Element 242
 - A.3.4 Four nodal Tetrahedron Element 243
 - A.3.5 Ten-node Tetrahedron Element 243
 - A.3.6 Six-node Triangular Prism Element 244
- Appendix B Isoparametric Element 246**
 - B.1 Integral Transformation for Rectangular
Coordinates of Natural Coordinates 249
 - B.2 Integral Transformation for Rectangular
Coordinates of Natural Coordinates 251
- Appendix C Numerical Integration 253**
 - C.1 Gaussian Integral 253
 - C.2 Nodal Integral 256

**Appendix D Appendix D Term Collections of
 Finite Element Program 258**

**Appendix E Collections of Key Words of
 Finite Element Language 260**

Symbol Table 265

References 266

Index 267

Introduction

In the 1950's, scientists in structural mechanics in the united states invented finite element method. The main principles of this method are originally based on variational method and piecewise low-order interpolation polynomial. However these two basic principles no longer meet user's needs due to the vast finite element applications in physics and engineering over the past decades. They have been then developed based on partial differential equation weak form and arbitrary approximation function space.

In physics and engineering, boundary value problem of partial differential equations (PDEs) in general can be expressed as follows.

In the solving domain Ω , the partial differential is satisfied by

$$A(u) = \left\{ \begin{array}{c} A_1(u) \\ A_2(u) \\ \vdots \end{array} \right\} = 0 \quad (\Omega) \quad (1)$$

on the boundary $\partial\Omega$, the boundary value condition is satisfied

$$B(u) = \left\{ \begin{array}{c} B_1(u) \\ B_2(u) \\ \vdots \end{array} \right\} = 0 \quad (\partial\Omega) \quad (2)$$

Where A and B are differential operators, u is unknown function.

In general, these partial differential equations have no analytical solution and usually are solved by using numerical methods. Finite element method is one of these numerical methods that plays a major role in solving various PDEs effectively by providing the weak form solution.

The PDE (1) is multiplied by the test function δu and then integrated on Ω , this gives

$$\int_{\Omega} A^T(u) \delta u d\Omega = \int_{\Omega} A_i(u) \delta u d\Omega = 0 \quad (3)$$

Where δu is the arbitrary function.

Assuming $A(u)$ is a smooth function and the integral equation (3) is satisfied for any δu , then PDE (1) is satisfied at any point of Ω . This is because if at some points or some subdomains of Ω $A(u)$ doesn't satisfy (1), namely $A(u) \neq 0$, an appropriate function δu can be found to make the integral form of equation (3) doesn't to equal zero. It is obvious that when $A(u)$ is a smooth function, equations (1) and (3) are equivalent. But in some cases, for example, the problem of 3-D harmonic electromagnetic field, these two are not equivalent. In many cases, by performing integration by parts to equation (3) and using boundary value condition (2), another form can be obtained:

$$\int_{\Omega} D^T(u)C(\delta u)d\Omega + \int_{\partial\Omega} F^T(u)E(\delta u)dS = 0 \quad (4)$$

Where C , D , E , F are differential operators. Note that the derivative order of unknown functions may be lower than the differential operator in equation (1), therefore only the continuity of lower order of function u is required. Equation (4) is called the weak form of differential equation (1) with the boundary condition (2). Weak form (4) may also be obtained using Petrov-Galerkin method.

Finite element approximate solution u is written as

$$u = \sum_{i=1}^n N_i u_i \quad (5)$$

The variational variable (namely dummy variable) is written as

$$\delta u = \sum_{i=1}^n N_i \delta u_i \quad (6)$$

Where u_i is the unknown variable to be solved, δu_i is the dummy variable corresponding to u_i , N_i is the basis function and the finite element space is composed of the basis functions N_i ($i = 1, 2, \dots, n$).

By Plugging (5) and (6) into (3), finite dimensional algebra equations can be obtained from the arbitrary selection of dummy variable δu_i , therefore the problem of solving PDEs is converted to the problem of solving finite dimensional algebra equations.

As seen from (5) and (6), the approximating solution space and test function space have the same basic functions. In this book, methods with different basis functions for these two spaces generally are not considered.

Chapter 1

Description Language of Differential Equation Expression

The description of differential equation, a basic content of Finite Element Language (FEL), is presented in this chapter. File that uses pdf as its extension name, referred to as PDE file, is adopted by FEL to describe the differential equation expression based on weak form. The automatic generating system of FEL generates element subroutines to calculate element stiffness matrix, element damping matrix, element load vector, etc, by using this file.

GES file, the most fundamental file, provides all formulas of FEM such as shape function, numerical integration, etc. It is used to generate element subroutine. PDE file allows use to obtain shape function and the formula of numerical integration from formula library, thus makes the creation easy and simple.

The files of CDE, VDE and FDE are designed to save time for creating PDE expressions. The main content of CDE file is for complex variable differential equation expression. And the main content of VDE file is for tensor expression of differential equation. FDE file can access operator formula library for operator expression.

The main content of FBC file is for differential equation expression of boundary condition (the second-type and third-type boundary conditions). The automatic generating system of element subroutine generates subroutines to calculate element stiffness matrix, damping matrix, load vector, etc for boundary conditions based on this file. The following sections are dedicated to the creation of these files.

1.1 Writing PDE Files

The main content of PDE file is to prepare differential equation expression. System will automatically generate element subroutines for calculating element stiffness matrix, element mass matrix, element damping matrix, etc. according to this file. The structure of PDF file is demonstrated as follows.

(1) User needs to create 6 segments at most whose keywords of information segments are DEFI, FUNC, STIF, MASS, DAMP and LOAD respectively. According

to different cases, user can write each segment of information as above-mentioned sequence.

This language defines that each segment of information (hereinafter referred to as information segment) should begin with its keyword and end with a blank line. After all information segments, the terminator END should be used for ending.

(2) This language allows user to insert Fortran source program. The Fortran code statements led by characters \$C6, \$C0 and \$CV can be inserted at some specific information segments. In addition, this language allows user to add Fortran source program beginning with keyword FORT at the end of PDF files. The methods of inserting and adding Fortran source codes will be specifically addressed in the following sections.

(3) This language defines character “\” (backslash) as continuation line. The characters after continuation line “\” are skipped and the next line is taken as the continuation line for the current line. Hence, backslash “\” also can be used as comment line. User can add comment line beginning with character “\” at any position in PDE file.

The following sections will describe how to create each information segment. All sections are accompanied by some examples. We will take Poisson equation for example to illustrate how to create PDE file for a practical problem in section 1.1.8.

1.1.1 DEFI Information Segment

DEFI information segment of PDE file needs 10 lines of information at most, and its general format is as follows:

```
DISP=name of unknown function, name of unknown function, ..., name of
  unknwn function
COORD=name of coordinate variable, name of coordinate variable, ...,
  name of coordinate variable
COEF=name of nonlinear-coefficient function, name of nonlinear-
  coefficient function, ..., name of nonlinear-coefficient function
FUNC=name of custom function, name of custom function, ..., name of
  custom function
MATE=name of material parameter, name of material parameter, ...,
  name of material parameter
SHAP=character of element geometry type, number of element nodes
GAUS=character of element geometry type or number of Gauss integrat-
  ion point
MASS=character of element geometry type, mass density
DAMP=character of element geometry type, damping coefficient
```


LOAD=expression, expression, ..., expression

By leaving a blank space (or equal sign) on the right side of DISP, user can write any number of the unknown function names of differential equation which need to be solved. Each unknown function name corresponds to one unknown function. All unknown function names should be separated by a blank space, comma or semicolon.

By leaving a blank space (or equal sign) on the right side of COOR, coordinate variable names can be provided without quantitative limitation. Each coordinate variable name corresponds to one coordinate variable. All coordinate variable names should be separated by a blank space, comma or semicolon.

By leaving a blank space (or equal sign) on the right side of COEF, coefficient variable names can be provided without quantitative limitation. All coefficient variable names should be separated by a blank space, comma or semicolon. Each coefficient variable name corresponds to one coefficient of differential equation. This line must be ignored if there is no coefficient variable. For example, user needs to create this line for nonlinear problem or coupled problem.

By leaving a blank space (or equal sign) on the right side of FUNC, user-defined function names can be written without quantitative limitation. Each function name corresponds to one user-defined function. All function names should be separated by a blank space, comma or semicolon. This language also provides the feature of substitution function. This line is used to define function name and it must be followed by the corresponding FUNC information segment. After defining function expression, all same expression of each information segment is replaced by user-defined function name. This makes information creation easy and also provides convenience for reading and modifying. Note that the information at FUNC line should be consistent with the FUNC information segment followed. Even though their keywords are the same, information of FUNC line is just one line required in DEFI information segment. Its main contents are provided in the FUNC information segment. If the virtual work equation is not complicate, the whole FUNC information segment may be ignored.

Leaving a blank space (or equal sign) on the right side of MATE, the material parameters are given without quantitative limitation. All material parameters are separated by a blank space or semicolon but not comma. The default values corresponding to material parameters are separated by a space or semicolon, which are written in the brace after (works as well without brace). Note that the default values cannot be separated by comma as well. The default values of material parameters which are not required by the program will be regarded as 0.0. These material parameters can be used in other information segments. After adding MATE, user can