

Editors

Timothy J. Barth, Moffett Field, CA

Michael Griebel, Bonn

David E. Keyes, New York

Risto M. Nieminen, Espoo

Dirk Roose, Leuven

Tamar Schlick, New York

Alfred Schmidt
Kunibert G. Siebert

Design of Adaptive Finite Element Software

The Finite Element Toolbox ALBERTA

With 30 Figures

 Springer

Alfred Schmidt
Zentrum für Technomathematik
Fachbereich Mathematik/Informatik
Universität Bremen
Bibliothekstr. 2
28359 Bremen, Germany
e-mail: schmidt@math.uni-bremen.de

Kunibert G. Siebert
Institut für Mathematik
Universität Augsburg
Universitätsstraße 14
86159 Augsburg, Germany
e-mail: siebert@math.uni-augsburg.de

Library of Congress Control Number: 2004113298

Mathematics Subject Classification (2000):
65M50, 65M60, 65N22, 65N30, 65N50, 65Y99, 68U20

ISSN 1439-7358
ISBN 3-540-22842-X Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable for prosecution under the German Copyright Law.

The publisher and the authors accept no legal responsibility for any damage caused by improper use of the instructions and programs contained in this book and the CD-ROM. Although the software has been tested with extreme care, errors in the software cannot be excluded.

Springer is a part of Springer Science+Business Media
springeronline.com

© Springer-Verlag Berlin Heidelberg 2005
Printed in Germany

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Cover design: Friedhelm Steinen-Broo, Estudio Calamar, Spain
Cover production: *design & production*, Heidelberg
Typeset by the authors using a Springer \TeX macro package
Production: LE- \TeX Jelonek, Schmidt & Vöckler GbR, Leipzig

Printed on acid-free paper 46/3142/YL - 5 4 3 2 1 0

Preface

During the last years, scientific computing has become an important research branch located between applied mathematics and applied sciences and engineering. Nowadays, in numerical mathematics not only simple model problems are treated, but modern and well-founded mathematical algorithms are applied to solve complex problems of real life applications. Such applications are demanding for computational realization and need suitable and robust tools for a flexible and efficient implementation. Modularity and abstract concepts allow for an easy transfer of methods to different applications.

Inspired by and parallel to the investigation of real life applications, numerical mathematics has built and improved many modern algorithms which are now standard tools in scientific computing. Examples are adaptive methods, higher order discretizations, fast linear and non-linear iterative solvers, multi-level algorithms, etc. These mathematical tools are able to reduce computing times tremendously and for many applications a simulation can only be realized in a reasonable time frame using such highly efficient algorithms.

A very flexible software is needed when working in both fields of scientific computing and numerical mathematics. We developed the toolbox ALBERTA¹ for meeting these requirements. Our intention in the design of ALBERTA is threefold: First, it is a toolbox for fast and flexible implementation of efficient software for real life applications, based on the modern algorithms mentioned above. Secondly, in an interplay with mathematical analysis, ALBERTA is an environment for improving existent, or developing new numerical methods. And finally, it allows the direct integration of such new or improved methods in existing simulation software.

Before having ALBERTA, we worked with a variety of solvers, each designed for the solution of one single application. Most of them were based on data structures specifically designed for one single application. A combination of different solvers or exchanging modules between programs was hard to do.

¹The original name of the toolbox was ALBERT. Due to copyright reasons, we had to rename it and we have chosen ALBERTA.

Facing these problems, we wanted to develop a general adaptive finite element environment, open for implementing a large class of applications, where an exchange of modules and a coupling of different solvers is easy to realize.

Such a toolbox has to be based on a solid concept which is still open for extensions as science develops. Such a solid concept can be derived from a mathematical abstraction of problem classes, numerical methods, and solvers. Our mathematical view of numerical algorithms, especially finite element methods, is based on our education and scientific research in the departments for applied mathematics at the universities of Bonn and Freiburg. This view point has greatly inspired the abstract concepts of ALBERTA as well as their practical realization, reflected in the main data structures. The robustness and flexible extensibility of our concept was approved in various applications from physics and engineering, like computational fluid dynamics, structural mechanics, industrial crystal growth, etc. as well as by the validation of new mathematical methods.

ALBERTA is a library with data structures and functions for adaptive finite element simulations in one, two, and three space dimension, written in the programming language ANSI-C. Shortly after finishing the implementation of the first version of ALBERTA and using it for first scientific applications, we confronted students with it in a course about finite element methods. The idea was to work on more interesting projects in the course and providing a strong foundation for an upcoming diploma thesis. Using ALBERTA in education then required a documentation of data structures and functions. The numerical course tutorials were the basis for a description of the background and concepts of adaptive finite elements.

The combination of the abstract and concrete description resulted in a manual for ALBERTA and made it possible that it is now used world wide in universities and research centers. The interest from other scientists motivated a further polishing of the manual as well as the toolbox itself, and resulted in this book.

These notes are organized as follows: In Chapter 1 we describe the concepts of adaptive finite element methods and its ingredients like the domain discretization, finite element basis functions and degrees of freedom, numerical integration via quadrature formulas for the assemblage of discrete systems, and adaptive algorithms.

The second chapter is a tutorial for using ALBERTA without giving much details about data structures and functions. The implementation of three model problems is presented and explained. We start with the easy and straight forward implementation of the Poisson problem to learn about the basics of ALBERTA. The examples with the implementation of a nonlinear reaction-diffusion problem and the time dependent heat equation are more involved and show the tools of ALBERTA for attacking more complex problems. The chapter is closed with a short introduction to the installation of the

ALBERTA distribution enclosed to this book in a UNIX/Linux environment. Visit the ALBERTA web site

<http://www.alberta-fem.de/>

for updates, more information, FAQ, contributions, pictures from different projects, etc.

The realization of data structures and functions in ALBERTA is based on the abstract concepts presented in Chapter 1. A detailed description of all data structures and functions of ALBERTA is given in Chapter 3. The book closes with separate lists of all data types, symbolic constants, functions, and macros.

The cover picture of this book shows the ALBERTA logo, combined with a locally refined cogwheel mesh [17], and the norm of the velocity from a calculation of edge tones in a flute [4].

Starting first as a two-men-project, ALBERTA is evolving and now there are more people maintaining and extending it. We are grateful for a lot of substantial contributions coming from: Michael Fried, who was the first brave man besides us to use ALBERT, Claus-Justus Heine, Daniel Köster, and Oliver Kriessl. Daniel and Claus in particular set up the GNU configure tools for an easy, platform-independent installation of the software.

We are indebted to the authors of the gltools, especially Jürgen Fuhrmann, and also to the developers of GRAPE, especially Bernard Haasdonk, Robert Klöfkorn, Mario Ohlberger, and Martin Rumpf.

We want to thank the Department of Mathematics at the University of Maryland (USA), in particular Ricardo H. Nochetto, where part of the documentation was written during a visit of the second author. We appreciate the invitation of the Isaac Newton Institute in Cambridge (UK) where we could meet and work intensively on the revision of the manual for three weeks.

We thank our friends, distributed all over the world, who have pointed out a lot of typos in the manual and suggested several improvements for ALBERTA.

Last but not least, ALBERTA would not have come into being without the stimulating atmosphere in the group in Freiburg, which was the perfect environment for working on this project. We want to express our gratitude to all former colleagues, especially Gerhard Dziuk.

Bremen and Augsburg, October 2004

Alfred Schmidt and Kunibert G. Siebert

Contents

Introduction	1
1 Concepts and abstract algorithms	9
1.1 Mesh refinement and coarsening	9
1.1.1 Refinement algorithms for simplicial meshes	12
1.1.2 Coarsening algorithm for simplicial meshes	18
1.1.3 Operations during refinement and coarsening	20
1.2 The hierarchical mesh	22
1.3 Degrees of freedom	24
1.4 Finite element spaces and finite element discretization	25
1.4.1 Barycentric coordinates	26
1.4.2 Finite element spaces	29
1.4.3 Evaluation of finite element functions	29
1.4.4 Interpolation and restriction during refinement and coarsening	32
1.4.5 Discretization of 2nd order problems	35
1.4.6 Numerical quadrature	38
1.4.7 Finite element discretization of 2nd order problems	39
1.5 Adaptive Methods	42
1.5.1 Adaptive method for stationary problems	42
1.5.2 Mesh refinement strategies	43
1.5.3 Coarsening strategies	47
1.5.4 Adaptive methods for time dependent problems	49
2 Implementation of model problems	55
2.1 Poisson equation	56
2.1.1 Include file and global variables	56
2.1.2 The main program for the Poisson equation	57
2.1.3 The parameter file for the Poisson equation	59
2.1.4 Initialization of the finite element space	60
2.1.5 Functions for leaf data	60

2.1.6	Data of the differential equation	62
2.1.7	The assemblage of the discrete system	63
2.1.8	The solution of the discrete system	65
2.1.9	Error estimation	66
2.2	Nonlinear reaction–diffusion equation	68
2.2.1	Program organization and header file	69
2.2.2	Global variables	71
2.2.3	The main program for the nonlinear reaction–diffusion equation	71
2.2.4	Initialization of the finite element space and leaf data	72
2.2.5	The build routine	72
2.2.6	The solve routine	73
2.2.7	The estimator for the nonlinear problem	73
2.2.8	Initialization of problem dependent data	75
2.2.9	The parameter file for the nonlinear reaction–diffusion equation	79
2.2.10	Implementation of the nonlinear solver	80
2.3	Heat equation	93
2.3.1	Global variables	93
2.3.2	The main program for the heat equation	94
2.3.3	The parameter file for the heat equation	96
2.3.4	Functions for leaf data	98
2.3.5	Data of the differential equation	99
2.3.6	Time discretization	100
2.3.7	Initial data interpolation	100
2.3.8	The assemblage of the discrete system	101
2.3.9	Error estimation	105
2.3.10	Time steps	107
2.4	Installation of ALBERTA and file organization	111
2.4.1	Installation	111
2.4.2	File organization	111
3	Data structures and implementation	113
3.1	Basic types, utilities, and parameter handling	113
3.1.1	Basic types	113
3.1.2	Message macros	114
3.1.3	Memory allocation and deallocation	118
3.1.4	Parameters and parameter files	122
3.1.5	Parameters used by the utilities	127
3.1.6	Generating filenames for meshes and finite element data	127
3.2	Data structures for the hierarchical mesh	128
3.2.1	Constants describing the dimension of the mesh	128
3.2.2	Constants describing the elements of the mesh	129
3.2.3	Neighbour information	129

3.2.4	Element indices	130
3.2.5	The BOUNDARY data structure	130
3.2.6	The local indexing on elements	132
3.2.7	The MACRO_EL data structure	132
3.2.8	The EL data structure	134
3.2.9	The EL_INFO data structure	135
3.2.10	The NEIGH , OPP_VERTEX and EL_TYPE macros	137
3.2.11	The INDEX macro	137
3.2.12	The LEAF_DATA_INFO data structure	138
3.2.13	The RC_LIST_EL data structure	140
3.2.14	The MESH data structure	141
3.2.15	Initialization of meshes	143
3.2.16	Reading macro triangulations	144
3.2.17	Writing macro triangulations	150
3.2.18	Import and export of macro triangulations from/to other formats	151
3.2.19	Mesh traversal routines	153
3.3	Administration of degrees of freedom	161
3.3.1	The DOF_ADMIN data structure	162
3.3.2	Vectors indexed by DOFs: The DOF*_VEC data structures	164
3.3.3	Interpolation and restriction of DOF vectors during mesh refinement and coarsening	167
3.3.4	The DOF_MATRIX data structure	168
3.3.5	Access to global DOFs: Macros for iterations using DOF indices	170
3.3.6	Access to local DOFs on elements	171
3.3.7	BLAS routines for DOF vectors and matrices	173
3.3.8	Reading and writing of meshes and vectors	173
3.4	The refinement and coarsening implementation	176
3.4.1	The refinement routines	176
3.4.2	The coarsening routines	182
3.5	Implementation of basis functions	183
3.5.1	Data structures for basis functions	184
3.5.2	Lagrange finite elements	190
3.5.3	Piecewise constant finite elements	191
3.5.4	Piecewise linear finite elements	191
3.5.5	Piecewise quadratic finite elements	195
3.5.6	Piecewise cubic finite elements	200
3.5.7	Piecewise quartic finite elements	204
3.5.8	Access to Lagrange elements	206
3.6	Implementation of finite element spaces	206
3.6.1	The finite element space data structure	206
3.6.2	Access to finite element spaces	207
3.7	Routines for barycentric coordinates	208

3.8	Data structures for numerical quadrature	210
3.8.1	The QUAD data structure	210
3.8.2	The QUAD_FAST data structure	212
3.8.3	Integration over sub-simplices (edges/faces)	215
3.9	Functions for the evaluation of finite elements	216
3.10	Calculation of norms for finite element functions	221
3.11	Calculation of errors of finite element approximations	222
3.12	Tools for the assemblage of linear systems	224
3.12.1	Assembling matrices and right hand sides	224
3.12.2	Data structures and function for matrix assemblage	227
3.12.3	Data structures for storing pre-computed integrals of basis functions	236
3.12.4	Data structures and functions for vector update	243
3.12.5	Dirichlet boundary conditions	247
3.12.6	Interpolation into finite element spaces	248
3.13	Data structures and procedures for adaptive methods	249
3.13.1	ALBERTA adaptive method for stationary problems	249
3.13.2	Standard ALBERTA marking routine	255
3.13.3	ALBERTA adaptive method for time dependent problems	256
3.13.4	Initialization of data structures for adaptive methods	260
3.14	Implementation of error estimators	263
3.14.1	Error estimator for elliptic problems	263
3.14.2	Error estimator for parabolic problems	266
3.15	Solver for linear and nonlinear systems	268
3.15.1	General linear solvers	268
3.15.2	Linear solvers for DOF matrices and vectors	272
3.15.3	Access of functions for matrix-vector multiplication	274
3.15.4	Access of functions for preconditioning	275
3.15.5	Multigrid solvers	277
3.15.6	Nonlinear solvers	282
3.16	Graphics output	285
3.16.1	One and two dimensional graphics subroutines	286
3.16.2	gltools interface	290
3.16.3	GRAPE interface	293
	References	295
	Index	301
	Data types, symbolic constants, functions, and macros	311
	Data types	311
	Symbolic constants	311
	Functions	312
	Macros	315