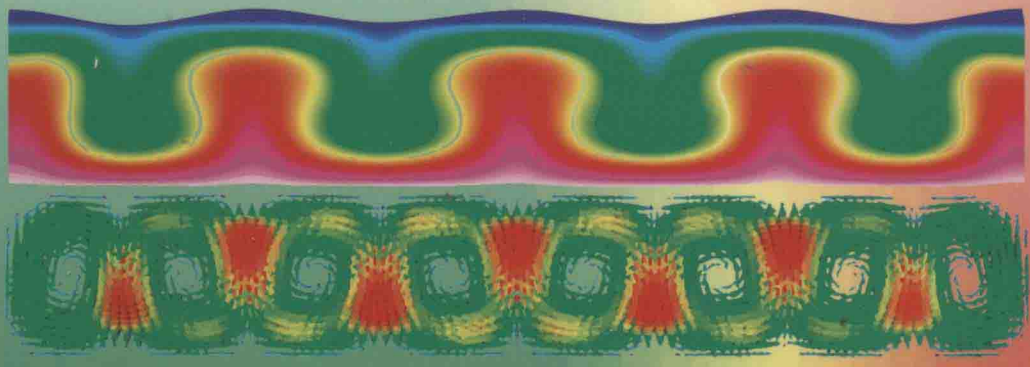
DMITRI KUZMIN • JARI HÄMÄLÄINEN

Finite Element Methods for Computational Fluid Dynamics

A Practical Guide



siam
Computational Science & Engineering

DMITRI KUZMIN
TU Dortmund University
Dortmund, Germany

JARI HÄMÄLÄINEN
Lappeenranta University of Technology
Lappeenranta, Finland

Finite Element Methods for Computational Fluid Dynamics

A Practical Guide

siam
Society for Industrial and Applied Mathematics
Philadelphia

This SIAM edition is partly based on the Finnish-language book *Elementtimenetelmä virtauslaskennassa* (in English: The Finite Element Method in Computational Fluid Dynamics) by J. Hämäläinen and J. Järvinen, CSC Press, 1994.

10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Kuzmin, Dmitri, author.
Finite element methods for computational fluid dynamics : a practical guide / Dmitri Kuzmin, Dortmund University of Technology, Dortmund, Germany, Jari Hämäläinen, Lappeenranta University of Technology, Lappeenranta, Finland.
pages cm. – (Computational science and engineering series)
"This SIAM edition is partly based on the Finnish-language book, Elementtimenetelmä virtauslaskennassa (in English: The finite element method in computational fluid dynamics) by J. Hämäläinen and J. Järvinen, CSC Press, 1994."
Includes bibliographical references and index.
ISBN 978-1-611973-60-0
1. Fluid dynamics–Mathematics. 2. Finite element method–Data processing. I. Hämäläinen, Jari, 1965- author. II. Title.
TA357.5.D37.K89 2014
620.1'0640151825–dc23

2014029504

Finite Element Methods for Computational Fluid Dynamics

A Practical Guide

Computational Science & Engineering

The SIAM series on Computational Science and Engineering publishes research monographs, advanced undergraduate- or graduate-level textbooks, and other volumes of interest to an interdisciplinary CS&E community of computational mathematicians, computer scientists, scientists, and engineers. The series includes both introductory volumes aimed at a broad audience of mathematically motivated readers interested in understanding methods and applications within computational science and engineering and monographs reporting on the most recent developments in the field. The series also includes volumes addressed to specific groups of professionals whose work relies extensively on computational science and engineering.

SIAM created the CS&E series to support access to the rapid and far-ranging advances in computer modeling and simulation of complex problems in science and engineering, to promote the interdisciplinary culture required to meet these large-scale challenges, and to provide the means to the next generation of computational scientists and engineers.

Series Volumes

Kuzmin, Dmitri and Hämäläinen, Jari, *Finite Element Methods for Computational Fluid Dynamics: A Practical Guide*

Rostamian, Rouben, *Programming Projects in C for Students of Engineering, Science, and Mathematics*

Smith, Ralph C., *Uncertainty Quantification: Theory, Implementation, and Applications*

Dankowicz, Harry and Schilder, Frank, *Recipes for Continuation*

Mueller, Jennifer L. and Siltanen, Samuli, *Linear and Nonlinear Inverse Problems with Practical Applications*

Shapira, Yair, *Solving PDEs in C++: Numerical Methods in a Unified Object-Oriented Approach, Second Edition*

Borzì, Alfio and Schulz, Volker, *Computational Optimization of Systems Governed by Partial Differential Equations*

Ascher, Uri M. and Greif, Chen, *A First Course in Numerical Methods*

Layton, William, *Introduction to the Numerical Analysis of Incompressible Viscous Flows*

Ascher, Uri M., *Numerical Methods for Evolutionary Differential Equations*

Zohdi, T. I., *An Introduction to Modeling and Simulation of Particulate Flows*

Biegler, Lorenz T., Ghattas, Omar, Heinkenschloss, Matthias, Keyes, David, and van Bloemen Waanders, Bart, Editors, *Real-Time PDE-Constrained Optimization*

Chen, Zhangxin, Huan, Guanren, and Ma, Yuanle, *Computational Methods for Multiphase Flows in Porous Media*

Shapira, Yair, *Solving PDEs in C++: Numerical Methods in a Unified Object-Oriented Approach*

Preface

Computational fluid dynamics (CFD) is the art of solving partial differential equations that model the motion of fluids, as well as mass and heat transfer phenomena. This book describes one the most powerful numerical techniques for CFD, the finite element method. The finite element method is readily applicable to domains of complex geometrical shape and provides a great freedom in the choice of numerical approximations. Regrettably, only finite difference and/or finite volume methods are usually discussed in introductory courses, and most CFD codes are based on these methods. One of the reasons for this state of affairs is the lack of self-contained elementary textbooks on finite elements for fluids.

In 1994, the second author and his colleague Jari Järvinen wrote a Finnish-language guide to finite elements for CFD (*Elementtimenetelmä virtauslaskennassa* [95]) which became a popular textbook at Finnish universities. The revised and extended second edition appeared in 2006. The present text brings [95] up to date and adds a lot of new material.

The mission of our guidebook is to provide an informal introduction to finite element methods for CFD applications without burdening the reader with the underlying mathematical theory. Instead of tedious proofs of convergence for the Poisson equation in a unit square, the reader will find a clear and detailed presentation of state-of-the-art numerical algorithms for convection-dominated transport problems and the incompressible Navier–Stokes equations. The methods to be presented have been chosen among many others for their accuracy, robustness, and simplicity. Moreover, the book will equip the reader with a collection of fail-safe tools for enforcing physical constraints such as nonnegativity.

We anticipate that the readers of this book will have very diverse backgrounds. The first chapters are written for beginners and present elementary concepts assuming no prior knowledge of CFD or finite elements. The material covered in these chapters may be used in undergraduate-level courses for students majoring in mathematics or computational engineering. In the chapters that follow, basic methods are extended to increasingly complex flow problems. Following a practice-oriented approach, numerical algorithms for representative models are discussed in great detail. The best way to gain the full understanding of these methods is to implement them. Simple problems can be solved using just a few lines of MATLAB code. As a software development kit for more advanced applications, we recommend the open-source finite element library *Elmer* developed at Finnish center for scientific computing CSC - IT Center for Science (www.csc.fi/elmer). The sections describing *Elmer* in Chapters 2 and 5 were contributed by its chief developers Peter Råback, Mika Malinen, and Juha Ruokolainen (CSC). The authors gratefully acknowledge this contribution.

The book will guide the reader through all the steps that are involved in solving the equations of fluid mechanics using continuous finite elements. The text is organized in nine chapters. Chapter 1 introduces the basic notation and mathematical foundations. Chapter 2 describes the general CFD philosophy and acquaints the reader with numeri-

cal methods for partial differential equations. In Chapter 3, some basic models of fluid flow and heat transfer are derived from physical conservation principles. An introduction to the Galerkin finite element method is given in Chapter 4 using the one-dimensional heat equation as a model problem. Chapter 5 extends the presented concepts to the two-dimensional heat equation and to the steady incompressible Navier–Stokes equations. In Chapters 6 and 7, we present a self-contained review of stabilization techniques and high-resolution schemes for convection-dominated transport problems. Chapter 8 addresses the numerical solution of the time-dependent incompressible Navier–Stokes equations and the implementation of the $k-\varepsilon$ turbulence model. In addition to numerous citations throughout the book, the reader will find a list of books for further reading in Chapter 9. To some extent, the eclectic choice of topics reflects the authors' personal research interests and teaching philosophy. Nevertheless, we hope that this excursion into the realm of finite element methods for CFD will be a useful experience for the reader and the first step toward solving real-life problems.

Our own interest in finite element methods for CFD goes back to our graduate studies at the University of Jyväskylä (Finland) in the 1990s. This book is an outcome of teaching and research activities which have been greatly influenced by the feedback of our students and colleagues. The methods and software that we develop are based on the work of many other people. We express our sincere gratitude to all individuals with whom we had the privilege to work during the last two decades. Special thanks go to Suncica Canic, Roland Glowinski, Heikki Haario, Jari Järvinen, Raino Mäkinen, Pekka Neittaanmäki, John Shadid, Friedhelm Schieweck, Mikhail Shashkov, Timo Tiihonen, Jari Toivanen, Stefan Turek, and to our late advisor Valery Rivkind. We also wish to thank Markus Hämäläinen, the son of the second author, for bringing the figures from the 20-year-old book [95] up to present-day standards. Last but not least, we thank our wives, Antonia and Taija, for patience and understanding.

Dmitri Kuzmin
Dortmund, Germany

Jari Hämäläinen
Lappeenranta, Finland

Contents

Chapter 1

Notation and Preliminaries

The mathematical models to be considered in this text are based on partial differential equations (PDEs) in $d \in \{1,2,3\}$ space dimensions. In this chapter, we introduce the basic mathematical notation that will be employed throughout the book. We also review some useful theorems of vector calculus and the basic definitions of functional analysis.

Blackboard bold capitals are commonly used to denote the sets of natural numbers $\mathbb{N}$, integers $\mathbb{Z}$, rational numbers $\mathbb{Q}$, real numbers $\mathbb{R}$, and complex numbers $\mathbb{C}$.

Lowercase roman, italic, or Greek letters refer to scalar quantities. For example, we denote the time variable by t and use the standard notation x_1, x_2, x_3 or x, y, z for the Cartesian coordinates of a point in the three-dimensional Euclidean space $\mathbb{R}^3$. Further real-valued variables or constants may be named, e.g., a, b, c or α, β, γ. The letters i, j, k, l, m, n are commonly reserved for integers (subscripts, superscripts, dimensions, etc.).

Lowercase boldface symbols denote vectors, i.e., elements of $\mathbb{R}^d$. The individual components of a given vector field are denoted by lowercase letters with subscripts. For example, the boldface symbol $\mathbf{x}$ is just shorthand notation for

$$\mathbf{x} = (x_1, \ldots, x_d)^T.$$

The subscript notation may also be used to pick out the nodal values of an approximate solution defined on a computational mesh. Superscripts usually refer to time levels or iteration steps. For example, $f_i^n \approx f(\mathbf{x}_i, t^n)$ may stand for the numerical value of a scalar function f at some point $\mathbf{x}_i \in \mathbb{R}^d$ and time instant $t^n \in \mathbb{R}$. If the computation of f_i^n requires several iterations, the provisional value calculated at the mth step is denoted by $f_i^{(m)}$.

Uppercase letters are reserved for matrices. The components of the square matrix

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{d1} & \cdots & a_{dd} \end{pmatrix}$$

are denoted by a_{ij}, where i is the row number and j is the column number.

The above subscript notation is also adopted for second-rank tensors denoted by boldface symbols with double overbars. The jth column of the tensor

$$\bar{\bar{\boldsymbol{\sigma}}} = \begin{pmatrix} \sigma_{11} & \cdots & \sigma_{1d} \\ \vdots & \ddots & \vdots \\ \sigma_{d1} & \cdots & \sigma_{dd} \end{pmatrix}$$

is given by the vector $\boldsymbol{\sigma}_j = (\sigma_{1j}, \ldots, \sigma_{dj})^T$. Using this notation, we can write

$$\bar{\bar{\boldsymbol{\sigma}}} = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_d).$$

The *dot product* and *dyadic product* of two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$ are defined by

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + \cdots + u_d v_d = \sum_{i=1}^{d} u_i v_i$$

and

$$\mathbf{u} \otimes \mathbf{v} = \begin{pmatrix} u_1 v_1 & \cdots & u_1 v_d \\ \vdots & \ddots & \vdots \\ u_d v_1 & \cdots & u_d v_d \end{pmatrix},$$

respectively. The dot product of a tensor $\bar{\bar{\boldsymbol{\sigma}}} \in \mathbb{R}^{d \times d}$ and a vector $\mathbf{n} \in \mathbb{R}^d$ is a vector

$$\bar{\bar{\boldsymbol{\sigma}}} \cdot \mathbf{n} = (\boldsymbol{\sigma}_1 \cdot \mathbf{n}, \ldots, \boldsymbol{\sigma}_d \cdot \mathbf{n})^T,$$

while the *double dot product* of two tensors $\bar{\bar{\boldsymbol{\sigma}}}, \bar{\bar{\boldsymbol{\varepsilon}}} \in \mathbb{R}^{d \times d}$ is a scalar quantity

$$\bar{\bar{\boldsymbol{\sigma}}} : \bar{\bar{\boldsymbol{\varepsilon}}} = \sum_{i=1}^{d} \sum_{j=1}^{d} \sigma_{ij} \varepsilon_{ji}.$$

The first and second derivative of a function $f : \mathbb{R} \to \mathbb{R}$ are denoted by $f'(x)$ and $f''(x)$, respectively. The first partial derivative of $f : \mathbb{R}^d \to \mathbb{R}$ with respect to (w.r.t.) the independent variable x_i is denoted by $\frac{\partial f}{\partial x_i}$. The vector of first partial derivatives is denoted by

$$\nabla = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_d} \right)^T.$$

The *gradient* of a scalar-valued function $f : \mathbb{R}^d \to \mathbb{R}$ is defined as

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d} \right)^T.$$

The *Laplacian* of f represents the sum of second partial derivatives

$$\Delta f = \sum_{j=1}^{d} \frac{\partial^2 f}{\partial x_j^2}.$$

The *divergence* of a vector field $\mathbf{u} : \mathbb{R}^d \to \mathbb{R}^d$ is defined as the dot product

$$\nabla \cdot \mathbf{u} = \frac{\partial u_1}{\partial x_1} + \cdots + \frac{\partial u_d}{\partial x_d} = \sum_{i=1}^{d} \frac{\partial u_i}{\partial x_i}.$$

Note that the gradient of a scalar is a vector, while the divergence of a vector is a scalar. In particular, it is easy to verify that $\nabla \cdot \nabla f = \Delta f$, i.e., $\Delta = \nabla^2$.

The gradient of a vector field $\mathbf{u}$ is a tensor containing all partial derivatives:

$$\nabla \mathbf{u} = \begin{pmatrix} \frac{\partial u_1}{\partial x_1} & \cdots & \frac{\partial u_1}{\partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial u_d}{\partial x_1} & \cdots & \frac{\partial u_d}{\partial x_d} \end{pmatrix} = (\nabla u_1, \ldots, \nabla u_d)^T.$$

The divergence of a second-rank tensor $\bar{\bar{\sigma}}$ is a vector field defined by the formula

$$\nabla \cdot \bar{\bar{\sigma}} = (\nabla \cdot \sigma_1, \ldots, \nabla \cdot \sigma_d)^T.$$

A composition of two functions can be differentiated by the *chain rule* of calculus,

$$(f \circ g)' = (f' \circ g) \cdot g',$$

where the prime mark is used to denote the total derivatives, i.e., the Jacobian matrices.

For example, let $f = f(\mathbf{x}, t)$ be a scalar-valued function of the space coordinates $\mathbf{x} \in \mathbb{R}^3$ and time $t \in \mathbb{R}$. The time rate of change along the path $(\mathbf{x}(t), t)$ is given by

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \nabla f \cdot \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} + \frac{\partial f}{\partial t}.$$

In fluid dynamics, the trajectory $\mathbf{x}(t)$ of a fluid particle moving with a prescribed velocity $\mathbf{u}$ is determined by integrating the ordinary differential equation (ODE)

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{u}.$$

In this case, the so-called *substantial derivative* (alias *material derivative*)

$$\frac{Df}{Dt} = \frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f \tag{1.1}$$

coincides with the total time derivative of $f(\mathbf{x}, t)$ along the path defined by $\mathbf{x}(t)$.

The integral of $f(\mathbf{x}, t)$ over a time-dependent bounded domain $\Omega(t)$ with a moving boundary $\partial\Omega(t)$ evolves in accordance with the *Reynolds transport theorem*

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega(t)} f \, \mathrm{d}\mathbf{x} = \int_{\Omega(t)} \frac{\partial f}{\partial t} \, \mathrm{d}\mathbf{x} + \int_{\partial\Omega(t)} f \mathbf{u} \cdot \mathbf{n} \, \mathrm{d}s, \tag{1.2}$$

where $\mathbf{n}$ denotes the outward-pointing unit normal vector. The surface integral can be transformed into a volume integral using *Green's formula*:

$$\int_{\partial\Omega} f \mathbf{u} \cdot \mathbf{n} \, \mathrm{d}s = \int_{\Omega} \mathbf{u} \cdot \nabla f \, \mathrm{d}\mathbf{x} + \int_{\Omega} f \nabla \cdot \mathbf{u} \, \mathrm{d}\mathbf{x}. \tag{1.3}$$

This useful identity represents a generalization of the integration-by-parts formula for functions of a single real variable. Assuming that the flux $\mathbf{g} = f\mathbf{u}$ is differentiable, Green's formula can be easily inferred from the *divergence theorem*

$$\int_{\Omega} \nabla \cdot \mathbf{g} \, \mathrm{d}\mathbf{x} = \int_{\partial\Omega} \mathbf{g} \cdot \mathbf{n} \, \mathrm{d}s \tag{1.4}$$

and the generalized *product rule*

$$\nabla \cdot (f\mathbf{u}) = \mathbf{u} \cdot \nabla f + f \nabla \cdot \mathbf{u}. \tag{1.5}$$

Knowledge of the above theorems and transformation rules is essential for the derivation of mathematical models and numerical algorithms to be presented in this guidebook.

In our presentation, we tacitly assumed that all integrals and partial derivatives exist, at least in a certain generalized sense. This assumption relies on the right choice of functional spaces, so we close this chapter with some definitions from functional analysis.

Let $\Omega \subset \mathbb{R}^d$ be an open set. The space of functions that are square integrable in Ω is denoted by $L^2(\Omega)$. The formal mathematical definition of this space is as follows:

$$L^2(\Omega) = \left\{ v : \Omega \to \mathbb{R} \,\middle|\, \int_\Omega |v(\mathbf{x})|^2 \,\mathrm{d}\mathbf{x} < \infty \right\}. \tag{1.6}$$

The space $L^2(\Omega)$ is a Hilbert space. The L^2 scalar product and norm are defined by

$$(w, v) = \int_\Omega w(\mathbf{x}) v(\mathbf{x}) \,\mathrm{d}\mathbf{x}, \qquad w, v \in L^2(\Omega), \tag{1.7}$$

$$\|v\| = \sqrt{\int_\Omega |v(\mathbf{x})|^2 \,\mathrm{d}\mathbf{x}}, \qquad v \in L^2(\Omega). \tag{1.8}$$

Note that the value of $\|v\|$ is finite by definition of the functional space $L^2(\Omega)$, while the value of (w, v) is finite by the *Cauchy–Schwarz inequality*

$$|(w, v)| \le \|w\| \|v\|, \qquad w, v \in L^2(\Omega). \tag{1.9}$$

The space of functions that are continuous on Ω is denoted by $C(\Omega)$ or $C^0(\Omega)$. If all partial derivatives of order $k \in \mathbb{N}$ exist and are continuous, the function belongs to the space $C^k(\Omega)$. A function v is said to have *compact support* if $v \equiv 0$ outside a bounded subset $\Omega' \subset \Omega$ which has a positive distance from the boundary $\partial\Omega$. The space of infinitely differentiable functions that have compact supports is denoted by $C_0^\infty(\Omega)$.

Generalized partial derivatives of a function $v \in L^2(\Omega)$ can be defined using multiplication by test functions $w \in C_0^\infty(\Omega)$ and integration by parts. Let $\mathbf{e}_i \in \mathbb{R}^d$, $i \in 1, \ldots, d$, denote the ith unit vector in $\mathbb{R}^d$. If v is differentiable in the classical sense, the application of Green's formula (1.3) to the vector field $\mathbf{u} = v\mathbf{e}_i$ and scalar field $f = w$ yields

$$\int_\Omega w \frac{\partial v}{\partial x_i} \,\mathrm{d}\mathbf{x} = -\int_\Omega \frac{\partial w}{\partial x_i} v \,\mathrm{d}\mathbf{x}. \tag{1.10}$$

The surface integral vanishes due to the assumption that $w \in C_0^\infty(\Omega)$ has a compact support.

In light of the above, it is natural to define a generalized (weak) partial derivative of $v \in L^2(\Omega)$ w.r.t. x_i as a function $\partial_i v \in L^2(\Omega)$ such that

$$\int_\Omega w \partial_i v \,\mathrm{d}\mathbf{x} = -\int_\Omega v \frac{\partial w}{\partial x_i} \,\mathrm{d}\mathbf{x} \qquad \forall w \in C_0^\infty(\Omega). \tag{1.11}$$

Higher-order generalized partial derivatives are defined in a similar way. Let

$$\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$$

be a vector of nonnegative integers. The sum of its components is denoted by

$$|\boldsymbol{\alpha}| = \alpha_1 + \cdots + \alpha_d.$$

In analogy to definition (1.11), a square integrable function

$$D^\alpha v = \frac{\partial^{|\alpha|} v}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}} \in L^2(\Omega)$$

represents a generalized (weak) partial derivative of order $|\alpha|$ if

$$\int_\Omega w D^\alpha v \, \mathrm{d}\mathbf{x} = (-1)^{|\alpha|} \int_\Omega v D^\alpha w \, \mathrm{d}\mathbf{x} \qquad \forall w \in C_0^\infty(\Omega). \tag{1.12}$$

If the partial derivative $D^\alpha v$ exists in the usual sense of differential calculus, this formula can be derived using repeated integration by parts. Thus, a classical partial derivative (if it exists) is always equal to its generalized counterpart defined in the sense of distributions.

Since we are interested in solving PDEs, we will need to assume the existence of (at least) first generalized partials. The *Sobolev space*

$$H^1(\Omega) = \{v \in L^2(\Omega) \mid D^\alpha v \in L^2(\Omega),\ |\alpha| = 1\}$$

contains all square integrable functions with square integrable first generalized partials.

If the function v is continuous on $\bar{\Omega}$, its boundary values are defined by the restriction $v|_{\partial\Omega}$. However, two functions $v, w \in L^2(\Omega)$ are equivalent if they differ on a set of measure zero. By the *trace theorem* of functional analysis, there exists a bounded linear operator

$$\gamma : H^1(\Omega) \to L^2(\partial\Omega): \quad \gamma v = v|_{\partial\Omega} \quad \forall v \in H^1(\Omega) \cap C(\bar{\Omega}).$$

Hence, the trace γv represents a well-defined generalization of $v|_{\partial\Omega}$ to $v \in H^1(\Omega)$. The space of H^1 functions with vanishing traces on $\partial\Omega$ is denoted by

$$H_0^1(\Omega) = \{v \in H^1(\Omega) \mid \gamma v = 0\}.$$

Generalized derivatives and Sobolev spaces are widely used in texts on functional analysis and finite element methods for PDEs. We will also refer to them occasionally in this guidebook and introduce additional definitions in the course of presentation.

List of Variables

α	multi-index
α_{ij}	correction factor
β	thermal expansion coefficient
Γ	boundary of a domain
δ_{ij}	Kronecker's delta
ε	emissivity, diffusion coefficient, turbulent dissipation rate
$\bar{\bar{\varepsilon}}$	strain rate tensor,
$\varkappa$	von Karman constant
μ	dynamic viscosity
μ_T	eddy viscosity
ν	kinematic viscosity
Φ	limiter function
ρ	density

List of Variables, continued

σ	Stefan–Boltzmann constant
$\sigma_k, \sigma_\varepsilon$	constants in the k–ε turbulence model
$\underline{\underline{\boldsymbol{\sigma}}}$	Cauchy stress tensor
τ	stabilization parameter
$\underline{\underline{\tau}}$	Reynolds stress tensor
τ_w	wall shear stress
Ω	bounded domain
c_p	specific heat capacity at constant pressure
C_μ, C_1, C_2	constants in the k–ε turbulence model
e	internal energy
E	roughness coefficient
f_{ij}	numerical flux
$\mathbf{f}$	flux, body force
g	gravitational constant
$\mathbf{g}$	gravitational force
h	heat source, mesh size
h_c	heat transfer coefficient
h_r	effective heat transfer coefficient
$\mathbf{I}$	identity tensor
k	heat conductivity, turbulent kinetic energy
$\mathbf{n}$	unit outward normal
p	static pressure
Pe	Péclet number
$\mathbf{q}$	heat flux
Re	Reynolds number
s	source term
S	control surface
T	temperature
$\mathbf{u}$	velocity
u_τ	friction (shear) velocity
$\mathbf{v}$	velocity
V	control volume
y	dimensional wall distance
y^+	dimensionless wall distance

List of Abbreviations

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
ADI	alternating direction implicit
ALE	arbitrary Lagrangian Eulerian
AMG	algebraic multigrid
BDF	backward difference formula
BiCGStab	biconjugate gradient stabilized
BW	Beam–Warming
CCS	compressed column storage
CDR	convection-diffusion-reaction

List of Abbreviations, continued

CES	compressed edge storage
CFD	computational fluid dynamics
CFL	Courant–Friedrichs–Lewy
CG	conjugate gradient
CGS	conjugate gradient squared
CPU	central processing unit
CRS	compressed row storage
CV	control volume
DAE	differential-algebraic equation
DC	discontinuity capturing
DG	discontinuous Galerkin
DMP	discrete maximum principle
DNS	direct numerical simulation
DOF	degree of freedom
FCT	flux-corrected transport
FDM	finite difference method
FEM	finite element method
FSI	fluid-structure interaction
FVM	finite volume method
GLS	Galerkin least squares
GMG	geometric multigrid
GMRES	generalized minimal residual
GPU	graphics processing unit
GUI	graphical user interface
ILU	incomplete lower-upper
LBB	Ladyzhenskaya–Babuška–Brezzi
LED	local extremum diminishing
LES	large eddy simulation
LP	linearity preserving
LW	Lax–Wendroff
MC	monotonized centered
MPI	message passing interface
MOL	method of lines
ODE	ordinary differential equation
PDE	partial differential equation
PGP	pressure gradient projection
PPE	pressure Poisson equation
PSC	pressure Schur complement
RANS	Reynolds-averaged Navier–Stokes
SD	streamline diffusion
SGS	subgrid scale
SI	system of units
SOR	successive overrelaxation
SU	streamline upwind
SUPG	streamline upwind Petrov–Galerkin
TDMA	tridiagonal matrix algorithm
TFQMR	transpose-free quasi-minimal residual
TG	Taylor–Galerkin
TVD	total variation diminishing